Traffic Management and Control of Automated Guided Vehicles Using Artificial Neural Networks

by

Manuel Romano dos Santos Pinto Barbosa

A thesis submitted in fulfilment of the requirements for

the Degree of Doctor of Philosophy at the University of Warwick

Department of Engineering

University of Warwick

August 1997

Summary

An industrial traffic management and control system based on Automated Guided Vehicles faces several combined problems. Decisions must be made concerning which vehicles will respond, or are allocated to each of the transport orders. Once a vehicle is allocated a transport order, a route has to be selected that allows it to reach its target location. In order for the vehicle to move efficiently along the selected route it must be provided with the means to recognise and adapt to the changing characteristics of the path it must follow. When several vehicles are involved these decisions are interrelated and must take into account the coordination of the movements of the vehicles in order to avoid collisions and maximise the performance of the transport system. This research concentrates on the problem of routing the vehicles that have already been assigned destinations associated with transport orders.

In nearly all existing AGV systems this problem is simplified by considering there to be a fixed route between source and destination workstations. However if the system is to be used more efficiently, and particularly if it must support the requirements of modern manufacturing strategies, such as Just-in-Time and Flexible Manufacturing Systems, of moving very small batches more frequently, then there is a need for a system capable of dealing with the increased complexity of the routing problem.

The consideration of alternative paths between any two workstations together with the possibility of other vehicles blocking routes while waiting at a particular location, increases enormously the number of alternatives that must be considered in order to identify the routes for each vehicle leading to an optimum solution. Current methods used to solve this type of problem do not provide satisfactory solutions for all cases, which leaves scope for improvement. The approach proposed in this work takes advantage of the use of Backpropagation Artificial Neural Networks to develop a solution for the routing problem. A novel aspect of the approach implemented is the use of a solution derived for routing a single vehicle in a physical layout when some pieces of track are set as unavailable, as the basis for the solution when several vehicles are involved. Another original aspect is the method developed to deal with the problem of selecting a route between two locations based on an analysis of the conditions of the traffic system, when each movement decision has to be made. This lead to the implementation of a step-by-step search of the available routes for each vehicle.

Two distinct phases can be identified in the approach proposed. First the design of a solution based on an ANN to solve the single vehicle case, and subsequently the development and testing of a solution for a multi-vehicle case. To test and implement these phases a specific layout was selected, and an algorithm was implemented to generate the data required for the design of the ANN solution.

During the development of alternative solutions it was found that the addition of simple rules provided a useful means to overcome some of the limitations of the ANN solution, and a "hybrid" solution was originated. Numerous computer simulations were performed to test the solutions developed against alternatives based on the best published heuristic rules. The results showed that while it was not possible to generate a globally optimal solution, near optimal solutions could be obtained and the best hybrid solution was marginally better than the best of the currently available heuristic rules.

To my mother,

and my wife Elizabeth,

and in found memory of my grandfather.

Table of Contents

List of Figures

Chapter 5

Chapter 6

Chapter 7

Chapter 8

List of Tables

Chapter 5

Chapter 6

Chapter 7

Chapter 9

Acknowledgements

I would like to express my profound gratitude to my supervisors, Dr T.C. Goodhead and Dr. E.L. Hines, for their advice and encouragement throughout this work.

I would also like to thank my colleagues at the University of Porto for their support to initiate and complete this work, and in particular to Prof. Fernando Gomes de Almeida for the invaluable discussions and advice.

I would also like to acknowledge the British Council and the Junta Nacional de Investigação Científica e Tecnológica for the financial support that made it possible the accomplishment of this work.

Finally I am grateful to my friends and my family.

Declaration

The accompanying dissertation entitled "Traffic Management and Control of Automated Guided Vehicles Using Artificial Neural Networks" is submitted in support of an application for the degree of Doctor of Philosophy at the University of Warwick.

The observations and interpretations described in this thesis are those of the author, except where acknowledgement has been made to results and ideas previously published.

None of the work has been, or is being, submitted for any other degree or diploma to this or any other institution. A part of this work has formed the basis of one conference paper.

I hereby declare that the above statements are true.

(Manuel Romano dos Santos Pinto Barbosa)

August, 1997

Abbreviations

AGV	Automated Guided Vehicle
AGVS	Automated Guided Vehicle System
ANN	Artificial Neural Network
ANN_0_LH0	Artificial neural network based solution, without heuristic, but with complete look-ahead strategy
ANN_H_LH0	Artificial neural network based solution, with heuristic and complete look-ahead strategy
ANN_H_LH1	Artificial neural network based solution, with heuristic and limited look-ahead strategy
ANN_H_NLH	Artificial neural network based solution, with heuristic but without look-ahead strategy
BENVO_LH0	Solution based on an heuristic which favours a consistent direction of the traffic flow ("benevolent counterclockwise flow policy"), and with complete look-ahead strategy
BENVO_LH1	Solution based on an heuristic which favours a consistent direction of the traffic flow ("benevolent counterclockwise flow policy"), and with limited look-ahead strategy
BENVO_POL	Solution based on an heuristic which favours a consistent direction of the traffic flow ("benevolent counterclockwise flow policy") only (i.e. without look-ahead strategy)
BP ANN	Backpropagation Artificial Neural Network
FMS	Flexible Manufacturing System
GA	Genetic Algorithms
GRADI_LH0	Pure gradient solution with complete look-ahead strategy
GREED_LH0	Solution based on the "greedy policy" heuristic and with complete look-ahead strategy

GREED_LH1	Solution based on the "greedy policy" heuristic and with limited look-ahead strategy
GREED_POL	Solution based on the "greedy policy" heuristic only (i.e. without look-ahead strategy)
JIT	Just in Time
RAN_0_LH0	Solution based on random choices, without heuristic and complete look-ahead strategy
RAN_0_NLH	Solution based on random choices, without heuristic and without any look-ahead strategy
RAN_H_LH0	Solution based on random choices, with heuristic and complete look-ahead strategy
TSP	Travelling Salesman Problem

Chapter 1 | Introduction

The development of modern manufacturing concepts and the technology that supports them gives rise to an increased usage of automation at various levels within a manufacturing system. The goal to remain competitive in a more global open market was one of the reasons for these developments. The capability to respond more rapidly and efficiently to an increased demand for a higher variety of products was seen as a major requirement. The impact on manufacturing systems affects all its functions and activities, including at its operational level where they come under pressure to reduce lead-times, work with smaller batches, and adapt efficiently to different product mix. Dealing appropriately with these issues is a major step towards an overall manufacturing system whose response capability and competitiveness is not impaired by its operational level. Manufacturing concepts emerged to respond to the new challenges that manufacturing systems face at various levels. Flexible Manufacturing Systems (FMS) and Just-in-Time (JIT) illustrate well the new concepts and deserve a special reference due to their impact at the operations level, and particularly on the flow of materials.

Among the various issues that contribute to an increase in the operational flexibility and response capability of a manufacturing system, the transport system presents specific problems that remain to be addressed. The increased level of transport flexibility made possible by the use of driverless, self-powered, individual vehicles brings an increased level of complexity to the problem associated with making decisions to efficiently control the transport system. Recognising the advantages of a transport system with an increased level of flexibility and the limitations of current existing systems, the main

objective of this research work is to investigate whether more efficient solutions to the problem of traffic management and control of automated guided vehicles can be developed. The limitations of current solutions for this type of problem and the recent developments in other areas of research directed the work to the application of artificial neural networks in seeking a solution to this problem.

The development of a solution for this problem would be applicable in FMS type or any other manufacturing environment where possible alternative processing routes for the components or parts can be considered, and frequent deliveries of smaller quantities of materials are in demand. In such cases the implementation of the transport system assumes an increased importance due to the intensified repercussions it has on the job scheduling system. The flexibility associated with the connections or paths linking any two load/unload locations in the physical layout, and the capability of implementing these paths efficiently become important characteristics of the transport system.

The concept of manufacturing flexibility recognises the importance of the operational level in the capability of a manufacturing system to respond adequately to the new challenges. This flexibility can be characterised from various perspectives, being most often associated with manufacturing systems producing a range of similar products or variants of a product, and where the cost of the products or components is less dependent on variations in production volumes, processing route sequences, product design, and also accommodating more easily the introduction of new equipment. The implementation of these systems can benefit from the identification of similarities within the range of products and its classification into product families, which leads to an ability to design more efficient layouts.

Systems developed which are representative of the use of these organisational concepts generally use highly automated systems. The three main components usually present, are the machines or equipment used to process the materials into products, the transport and storage system, and an overall computer control system responsible for the planning and control of all systems. The use of programmable equipment and the possibility of preparing most of the operations in an off-line mode contributes significantly to a rapid change between products at a workstation and consequently allowing for smaller batches. The use of an automated transport and storage system gives the necessary control

over the flow of materials or tools. Different alternatives can be considered for an automated transport system [Ref. 1] but automated guided vehicles (AGVs) offer great potential to satisfy the flexibility requirements. They combine the flexibility of independent vehicles with a control system suitable for integration in the overall computerised control system.

To make it possible for driverless vehicles to move within a manufacturing layout various issues must be addressed. They can be related to the problems of guiding a vehicle along its path, the selection of a route that leads the vehicle to its destination, the co-ordination of the movements of the various vehicles, the allocation of transport requests to the vehicles, while satisfying the safety requirements for the operation of the all system. Different approaches are possible to deal with these problems, as is described in the next chapter (Chapter 2). However it is possible to distinguish two perspectives from which a solution for these problems is sought, and that are dependent on the characteristics of the vehicles and the role of the centralised control system in charge of the various vehicles. At one extreme we can think of vehicles capable of recognising and following paths according to the directions received from the centralised control system that has the responsibility to optimise the decisions such as allocation, routing, and traffic management of all vehicles. At the other extreme we can have vehicles with a higher decision making ability, that makes them able to recognise and react within their environment, and in a way that allows them to move in less restricted paths and being less dependent on the directions received by a central system. Possibly they can even deal directly with other vehicles, select and plan their route, and decide how to respond to specific transport orders. These later systems are more frequently associated with 'autonomous mobile robots' which continue to be an active area of research [Ref. 2]. The systems commercially available and frequently used in current manufacturing systems are closer to the first type where the vehicles match adequately the designation of 'automated guided vehicles'. The use of 'free-ranging' vehicles represents the more recent evolution of these vehicles and originated from the elimination of the restrictions to move along pre-defined paths which define a physical layout by means of underground wires.

Although AGVs have the potential to provide the required flexibility and controllability of the transport system, there are still fundamental problems which must be solved to take full advantage of their usage. An increased flexibility of the transport system allowing alternative routes between

locations in the physical layout creates the opportunity for an optimization of the transport system and consequently of the entire manufacturing system through its interaction with the job scheduling system. However when various vehicles are involved, their allocation to transport orders, the selection of a route among the several possible for each vehicle, and the co-ordination of the movements of all vehicles, are all interrelated problems whose complexity increases enormously when large systems are considered. The complexity will be even higher if the job scheduling and vehicle management problems are considered simultaneously.

The solutions for these type of problems normally rely on the use of heuristic rules to support the decisions to be made, either for the job scheduling or AGVs related problems [i.e. Ref. 3, to Ref. 11]. In current AGVs systems these problems are frequently simplified either because only a small number of vehicles is required, or by imposing restrictions such as unidirectionality of paths and fixed paths between locations, which allows the use of simple heuristic rules. While heuristic rules provide a solution in real time, for a complex problem they do not normally guarantee it will be the optimum solution. Furthermore, specific rules are tested and validated for specific operating conditions which may not always be exactly the ones faced in the real system. Obtaining even good solutions is therefore dependent on the selection of the most appropriated heuristic for each specific case.

In spite of the benefits already obtained with current solutions for the implementation of AGVs as the major transport system in a manufacturing environment, it is clear that they could take advantage of more efficient alternatives. This is even more significant when larger systems are envisaged and more vehicles are needed to satisfy the requirements of moving smaller quantities of materials more frequently. A transport system capable of satisfying these requirements provides a better support for the implementation of FMS and the pursuit of JIT objectives. The benefits of such a system, and the belief that the recent developments experienced in other areas of research such as artificial neural networks (ANNs) could be applied in an alternative solution for this problem, were the main motivation for this research work. Traffic management and control of AGVs includes the problems described so far related with AGV systems, but concentrating on the higher planning level of co-ordinating multiple vehicles rather than on the vehicle's lower control level.

Artificial neural networks (ANNs) can be considered as a problem solving technique with specific characteristics and a large application area. The distinguishing features were inspired by the superior performance of the human brain in applications such as image recognition, when compared with the possible solutions implemented in conventional von Neumann style computer systems. The superior capability is even more striking when only incomplete information or data is available, and when it is known that the elements, or neurons, in the brain work at much slower rates, i.e. few tens of milliseconds [Ref. 12] than the processors in computer systems. An explanation for the differences in performance lies on the use of many, i.e. 10^{11} in the human brain [Ref. 13], simpler elements working in parallel [Ref. 13]. This is one of the main distinguish features of ANN systems, which are sometimes described as parallel distributed processing systems [Ref. 14]. One of the difficulties associated with the traffic management and control of AGVs is the high number of alternatives that must be considered when a decision has to be made, and its strong dependency on the size of the system. The processing capability of an ANN provided by the multiple elements working in parallel, seemed therefore well suited to the requirements of this problem.

Another feature of ANNs inspired by an analogy with the human brain, is the use of a learning scheme through which knowledge about a problem is incorporated into the system. An ANN adapts itself to produce a solution to a problem by being exposed to examples or instances of that problem, rather than using an explicit model of the problem and its solution. Even if enough processing capability is available to solve a problem of allocating, routing, and co-ordinating the movement of multiple vehicles, there is still a complex problem which is how to relate and take account of the various factors characterising the AGV system, and the interrelation of the decisions to be made. The capability of ANNs to learn these relations without the need for its explicit identification was considered to be a significant advantage in the application of ANNs to this problem. If an ANN is capable of recognising patterns, it could possibly be capable of recognising the actual traffic conditions of the system and identify the appropriate actions or decisions to be made.

There appeared to be a high potential offered by the principles of ANNs to overcome the limitations of current solutions for AGVs systems. Exploring the application of ANNs to the AGV traffic

management and control problems, became then a major direction for this research work as is described in the next chapters.

The following chapter, Chapter 2, presents a characterisation of the various issues related to a traffic management and control system for AGVs. Along with the alternatives available to deal with the different aspects of its implementation, it will also be important to characterise its interaction with other systems. Although 'autonomous mobile robots' represent a higher level of evolution than current AGV systems, they both face similar types of problems and a more general framework [Ref. 2] will be used to characterise these systems.

In Chapter 3 the different types of ANNs and the issues related to their application to a particular problem are presented. Although ANNs are inspired by the human brain, the development and implementation of ANNs hardly relate to the real neurons in terms of the number of elements used and how they work. In part this is due to our limited knowledge of how the neurons in the brain really work, and also because the models used to develop and implement ANNs nevertheless provide encouraging solutions and alternatives to many problems in science and engineering. Particularly due to the developments in the last decade, ANNs have drawn enormous attention from researchers which has led to a great variety of ANN types and their applications to specific problems. In this characterisation special attention is given to ANNs more directly related to the traffic management and control problem.

With a clearer perspective of the importance and type of problem faced in an AGV traffic management and control system, together with the various issues related to the development of a solution based on ANNs, Chapter 4 concentrates on the characterisation of the complexity and specific formulation of the problem. Backpropagation (BP) ANN is selected as the type of ANN used to model the routing problem of multiple AGVs in a physical layout. The relative advantages and limitations of ANNs over other possible alternatives including Genetic Algorithms and Fuzzy Logic are also discussed.

The following chapters, Chapter 5 to Chapter 10, describe the approach derived to solve the problem and the work done to evaluate and test the solutions obtained. Chapter 5 presents and discusses

alternative representations of the single vehicle case into a BP ANN. Developing a BP ANN solution requires appropriate data about the problem and Chapter 6 describes the process used, and the characteristics of the generated data. Chapter 7 presents the results obtained with the BP ANN solutions for the single vehicle case and Chapter 8 proceeds with its application to a multi-vehicle case. At this stage it was perceived that a 'look-ahead' strategy would fit well our approach of using the single vehicle solution iteratively to each vehicle in turn, when multiple vehicles are involved. The evaluation of performance when multiple vehicles are involved is made by comparison with alternative methods, and these are also described. Chapter 9 presents the measures of performance used, the tests performed and the results obtained in the multiple vehicle's case. Having tested both the single and the multi-vehicle cases it became appropriate to discuss the overall approach and results obtained, together with the recommendations for further work, which is done in Chapter 10. Finally the major conclusions are collected in Chapter 11.

Chapter 2

Traffic Management and Control of AGVs

The implementation of a transport system based on AGVs involves a variety of issues ranging from the specific characteristics of its basic elements, to the interaction or impact it will have on other systems, such as job scheduling. Traffic management and control assists in the implementation of interrelated functions such as the co-ordination of multiple vehicles, assigning a transport task to each vehicle, and selecting a travel route for each vehicle. The implementation of these functions can be related to other characteristics of the AGV transport system, such as the type of vehicles or other design parameters. Representing the highest control level of the AGV system it will have a direct influence in the integration with other systems (i.e. job scheduling).

An overview of the characteristics of the basic elements of a transport system based on AGVs is presented first as it helps in clarifying the functions of the traffic management and control system. The alternatives range from the AGV systems used initially in FMS installations to the more sophisticated 'autonomous mobile robots'. The issue of 'autonomy' together with a hierarchical structure are described in [Ref. 2] and were found useful in the characterisation of the problems faced by both systems. Subsequently focus will be on the type of decisions made and alternatives selected at the design stage of a transport system based on AGVs for an FMS. They play a major role in the performance of AGV transport systems and are directly related to the selection of specific alternatives for the traffic management and control system. Finally the interaction of the traffic management and control system with the job scheduling system is analysed. The main concern is with its application in an FMS environment whose characteristics must be clearly identified.

2.1 Components of a transport system based on AGVs

A number of issues must be addressed when using a transport system based on AGVs to accomplish the basic functions of handling materials in a manufacturing environment, including:

- loading/unloading materials for transportation at specific pick up/deposit stations;
- guaranteeing a safe operation of the vehicles;
- identifying the position of each vehicle in the system;
- selecting the route a vehicle should follow to arrive at its destination;
- co-ordinating the movement of the various vehicles in order to avoid collisions and congestions;
- allocating vehicles to transport requests.

An 'automated guided vehicle system' (AGVS) represents a materials handling system addressing these issues, and is normally considered as an alternative in the implementation of the transport system for an FMS. Although different solutions to the implementation of AGV systems have been used, commercially available systems normally [Ref. 1, Ref. 15] involve the following basic elements:

- self-propelled, driverless, independently operated vehicles;
- specific methods to guide the vehicles within the physical layout;
- a computer based control system that communicates with the vehicles.

The characteristics of each of these elements are obviously interrelated. The level of autonomy of each vehicle in terms of decision making, recognising and interacting within its environment depends on how these elements are implemented. These systems normally follow a strategy that concentrates most of the analysis and decision making responsibilities in external computer systems, rather than on the on-board computer control system of each vehicle. The vehicles or AGVs can therefore be considered as 'automated' or 'automatic' vehicles which follow directions from an external system. These terms apply even more appropriately to the early systems where the vehicles movement was dependent on a physical definition of the paths, most often using an embedded wire or a reflective

paint strip. In more recent systems 'free-ranging' AGVs [Ref. 16] were developed which overcome this movement limitation.

The principle used in wire guided vehicles is to have two inductive sensors, one on either side of the vehicle, to detect the magnetic field originated by an alternate current flowing through an insulated wire, buried about 25 millimetres deep [Ref. 16] in the shop floor. The signal from the sensors provides the reference position to the controller of the steering mechanism allowing it to follow the energised wire. When guide wires are replaced by painted strips on the floor, optical sensors are used. Out of range of these guide paths the vehicles will normally automatically stop for safety reasons; unless the vehicles capability to estimate its position based on its driving mechanism is used in a 'dead reckoning' mode [Ref. 1], which might be effective over shorter distances and useful, for example, when docking.

The use of a guide wire to control the position of the vehicle also provides the means to direct the vehicle along the selected route when the vehicle approaches a branching point. Using wire loops energised with different frequencies makes it possible to indicate which one the vehicle should follow. Alternatively the wire loops are energised with only one frequency and selected branches are switched on/off to leave only the ones which correspond to the selected route.

In AGVS with 'free-ranging' vehicles one of the solutions used to define the position of the vehicles is through lasers mounted on the vehicles that scan reflective sensors positioned around the factory walls. Based on a map of the layout and the position of each vehicle the external control system has the information required to direct the vehicles. Using radio waves to communicate position and instructional data to the vehicles has also being referred to as another alternative [Ref. 16]; a vision system could be used to locate the vehicles in the physical layout.

Having most of the capability to make decisions implemented in an external computer system, the communication links with the vehicles assume particular importance. Radio control is a common solution to implement this communication link. It allows two-way communication between the on-board computer and a remote computer, independently of the position of the vehicle. When wire-

guided vehicles are used, the wire grid can also include in-floor communication lines that facilitate the reception of the signals at specific communication points [Ref. 15].

Although the design of the vehicles must satisfy the specific needs of each application, they are normally equipped with batteries, an on-board computer, a steering and driving system, communication devices, sensors to detect the presence of other vehicles (i.e. optical sensors, ultrasonic systems), a safety bumper, emergency stop buttons, warning lights and bells, and other devices needed for their operation such as a on-board control panel which allows manual vehicle control, vehicle programming, and other functions [Ref. 1].

Apart from the on-board sensors for safety and traffic control a 'zone blocking' concept is also frequently used to avoid collisions between the vehicles [Ref. 1]. It consists of dividing the layout into separate zones and preventing more than one vehicle entering a zone. Also for safety reasons AGVs can only travel at slow speed but typically in the range of 10 to 70 m/min [Ref. 15].

One main difficulty when using AGVs is the automatic transfer on to and off the vehicle, especially if there are a wide variety of loads [Ref. 16]. The lower positioning accuracy of the vehicles must be matched to the normally higher accuracy of the loading/unloading stations, and this normally requires the design of special devices at the docking stations. In addition to the positioning interface problem it is also necessary to co-ordinate the operation of the transfer devices (i.e. motorised fixturing such as elevators or rollertables [Ref. 15]) on-board the vehicle and on the load/unload station.

Various alternatives are available for the mechanical design of the vehicles based on the number of steering/driving wheels and fixed wheels, allowing for forward, reverse and also side travel. Three-wheeled vehicles with a single steering/driving wheel and two fixed rolling castors are typical, with larger vehicles containing two steering/driving wheels [Ref. 15]. Omnidirectional vehicles can also be designed using three or four steering/driving wheels independently powered or not; the higher manoeuvrability provided by these systems is normally exploited for the development of 'autonomous' vehicles [Ref. 17].

Having a fleet of vehicles capable of moving, and methods used to identify and guide them around the facilities, a system is now required which has the responsibility to make decisions relative to the management of the AGVS, including vehicle allocation to transport requests, route selection and co-ordination of the movements of all vehicles, in order for the transport system to accomplish its functions and achieve its objectives. When integrated in a highly automated factory the solution normally consists of a remote computer system which monitors floor equipment status, supervises vehicle traffic, and has the software required to make the necessary decisions that are subsequently transmitted to the vehicles. The computer system normally also provides a visual display of the current situation. In [Ref. 15] an example is presented where a three level control system is implemented in a wire-guided system. The first level consists of a main (host) computer which monitors, supervises and make most of the decisions. The second level consists of several microcomputers located on the shop floor and whose functions are to generate the guidewire frequencies, process communications between the vehicles and the main computer, and often control factory floor equipment. The third level is implemented within each vehicle at the on-board microcomputer which co-ordinates the vehicle's functions (pre-programmed command outputs, initiates tasks by activating drives, transfer mechanisms, directional motors, transmit status data to the main computer through the regional microcomputers). In other less automated systems the alternatives involve using the vehicle's on-board control panel to manually dispatch the vehicles [Ref. 1]. It is also possible to have systems where the vehicles can be directly called and stopped at the load/unloading stations and subsequently programmed and sent to its next destination.

The methods used by the computer system in making the decisions required to manage the fleet of vehicles will be addressed separately in a later section because of its interaction with the job scheduling system and with the specific design of the all AGVS.

The next section will focus on the developments associated with an approach to AGVS which is based on an increased autonomy of the vehicles relative to a main computer system, and therefore closer to the developments in the area of 'autonomous mobile robots' [Ref. 2].

2.2 Autonomous mobile vehicles

The AGVS described so far (section 2.1) are representative of commercially available alternatives that have been used successfully as a material handling system satisfying the requirements of FMS implementations. However they are still limiting the flexibility potential that can be explored in an FMS. These limitations are apparent when wire guided vehicles are used but increasing the capability of the vehicles to autonomously make-decisions, rather than acting as pre-programmed devices, is a major contribution to cost-effectively extend the use of FMS [Ref. 2]. If each vehicle has the ability to autonomously deal with unexpected events, such as recognising and avoiding obstacles, plan its own route and co-ordinate its movements directly with other vehicles which may interfere with its path, being able to interact directly with the pick up and deposit stations, and even take part in the assignment of transport requests, then the concept of transport flexibility is greatly extended. Transport systems based on the development of autonomous mobile vehicles follow a strategy based on an increased level of intelligence at the vehicles rather than on relying mostly on external computer systems. The research and developments in mobile robots and artificial intelligence are therefore intimately related to the development of transport systems based on autonomous mobile vehicles. Although such systems have limited commercial availability at present [Ref. 17] extensive research has been devoted to them [Ref. 2].

An increased mobility and manoeuvrability of an autonomous vehicle must be complemented with sensor and control systems capable of endowing it with the perception of its environment and the capability to react to it. A multiple sensor system is suggested in [Ref. 17] differentiating between sensors used within the vehicle to monitor the operation of its mechanisms (i.e. incremental encoders in the drive wheels), sensors used on-board the vehicles to inspect its surrounding environment (i.e. vision systems, proximity sensors), and world-based sensor systems (i.e. infrared, ultra-sonic, laser or radiotelemetry principles) to provide a model of the facilities. This perception-cognition-control processing is typically implemented using a three level hierarchical structure [Ref. 2, Ref. 17] consisting of a strategic planning level, a tactical planning level, and an operational level.

When applied to autonomous mobile vehicles these three levels are sometimes called differently by different researchers. An interesting terminology described in [Ref. 2] derives from an analogy with the distribution of responsibilities in human crews controlling 'manned' moving vehicles and which include the planner level, the navigator level, and the pilot level. The basic function at the planner's level is to specify a global path whose main 'regions' or locations are passed to the lower level (navigator level). The planner defines this global path in response to an input consisting of instructions or a task to be performed, and using information describing the global environment conditions from a map of the facilities and global sensors. Path planning, strategic planning, and long range planning have also been used to describe this planning level.

At the navigator's level, also sometimes referred to [Ref. 2] as obstacle avoidance, tactical planning, guidance execution, or intermediate range, the main objective is to produce a more refined planning of the path between the locations identified in the plans drafted on the level above. It consists of a sequence of actions which will be passed on to the next lower level, i.e. the pilot level, for implementation. A more detailed map and information from local sensors together with global information, is required in order to recognise unknown obstacles and to report abnormal situations to the planning system.

Finally at the pilot's level, also sometimes referred to as local navigation, operative planning, manoeuvre planning, or short range, the instructions received from the navigator's level are translated into mechanical actions taking into account a detailed view of the vehicle's immediate range of action, obtained from local sensors. Below the pilot's level is the controller which implements the tasks decided by the pilot and acts directly upon the environment.

When these autonomous vehicles are used as part of the transport system in a manufacturing environment, particular aspects which are not explicitly addressed in the hierarchical structure above described must be considered. These include [Ref. 17] the analysis of the transport requests and its allocation to vehicles, the docking/undocking operations, and the loading/unloading of the materials.

This hierarchical structure illustrates well the complexity of the problems faced when these tasks are to be solved on-board a vehicle which operates autonomously, that is without a link to a human operator. Knowledge from several areas of research, including sensor systems, computer systems, control, and artificial intelligence, must be brought together and further developed in order to increase the rudimentary forms of autonomy present in commercially available AGVS, such as the ones described in the previous section (section 2.1). Although many research efforts have been made to solve specific problems, such as local path planning [Ref. 18 to Ref. 24], and research projects have been undertaken [Ref. 17] to develop these systems, AGVS based on autonomous vehicles are not yet implemented in industrial environments.

2.3 Design and operation of AGV systems and its interaction with other elements in an FMS

In the previous sections the main objective was to describe the technological aspects that characterise the main components of a transport system based on automated guided vehicles and the possibility of using vehicles with a higher level of autonomy. However other problems arise when considering the selection and implementation of this type of transport system in an FMS facility. The interaction of the AGVS with other systems becomes apparent not only in terms of the issues directly related with the design aspects such as path layout, but also in the solutions used to manage the fleet of vehicles to accomplish its functions.

The requirements for the transport system result from an analysis and characterisation of the transport requests. The type and quantities of materials to be moved, the origin/destination locations and the expected material flows, are the main aspects that must be taken into account. The FMS configuration and scheduling procedures used for its operation directly affect the materials flow and therefore must also be considered in the AGVS design. Furthermore the AGVS must satisfy the requirements for integration with other automated systems, such as tool management, fixturing, and auxiliary machines, frequently used in FMS.

The different implementations of FMS can be characterised in relation to various aspects but a distinction is usually made between dedicated and random FMS. While in a dedicated FMS machines are arranged in a flow configuration, random FMS's are capable of producing a higher variety of parts in random order [Ref. 1, Ref. 5]. It should therefore be expected that random FMS's use different scheduling procedures and require higher flexibility from the transport system. Due to the complexity of the problems involved the solutions are normally based on heuristic rules such as the ones described in [Ref. 3]. A further clarification is made in [Ref. 6] of some terms used in relation to the scheduling procedures. Off-line scheduling is described as consisting of scheduling all operations of available jobs for the entire scheduling period while on-line scheduling concentrates on scheduling operations one at a time as they are needed. A distinction is also made between on-line and real-time scheduling where in the latter a short-term decision process is involved to generate and up-date schedules based on the current status of the system and overall requirements. Dynamic scheduling is used to emphasise the dynamic nature of real-time scheduling problems. The type of scheduling procedures used directly affects the transport request patterns and therefore it will have implications on the design and operation of the AGVS.

With the set of requirements obtained from this analysis it will be possible to specify the design parameters of the AGVS including the number of vehicles and the characteristics of the path layout. Various alternatives can be evaluated such as using uni or bidirectional tracks, paths with one or more ways, arranged in one or several loops, or in grid configurations, the number and type of crossing junctions, the location of the docking/undocking stations relative to the path layout, having deviation zones that can be used to temporarily park vehicles obstructing paths, the number of pick-up and deposit locations, the size of input/output buffers used at the pick-up/deposit stations, loading each vehicle with one or more materials each time, and also the specific solutions used to manage the AGVS when in operation. Answering these questions is normally a very complex problem [i.e. Ref. 25], even if the material flow patterns are well known and characterised. This complexity derives from the high number of interrelated parameters involved, along with its interdependence of the specific job scheduling procedures used [Ref. 5, Ref. 6, Ref. 25, Ref. 26].

Ideally the decisions concerning the scheduling of jobs should be made taking into account also the possibility of implementing the associated transport tasks. Both the allocation in time of jobs to machines and the allocation of vehicles to transport tasks should be considered simultaneously in order to optimise the performance of the overall manufacturing system. However each of these problems per se turn out to be of high combinatorial complexity in most cases.

Two distinct approaches have been followed when dealing with these problems. One sacrifices the transport flexibility by imposing restrictions on the path layout design, in favour of a simpler control system. Alternatively the problems are normally treated separately using heuristic rules.

The simplifications on the path layout design allow for the use of analytical [Ref. 27] tools to model the problem. Simulation models are also extensively used [Ref. 4, Ref. 27 to Ref. 30] to test and evaluate design solutions, including specific rules for the scheduling of jobs and the allocation of vehicles. The simplifications imposed at the design stage imply a reduction in the transport flexibility and consequently of the overall system. Frequently these simplifications consist of allowing only unidirectional tracks [Ref. 31], assuming fixed routes between workstations, optimising the path layout design into a single loop [Ref. 32 to Ref. 34], and also exploring the use of control zones where only one vehicle is allowed at a time [Ref. 35]. This trend to explore the advantages of using simpler control systems to co-ordinate the AGVs has lead to the design of non-overlapping loops or cells [Ref. 36 to Ref. 39]. The idea is to have only one vehicle circulating in each loop and serving the workstations that are arranged around that loop. The interface between adjacent loops is provided by transit areas.

When the objective is to take advantage of increasing levels of transport flexibility the solutions normally consist of using scheduling procedures based on heuristic rules. The job scheduling problem is often considered separately of the vehicle's allocation and route selection problem. Alternatively a combination of rules can be used at a dynamic or real-time scheduling level to dispatch jobs to machines and allocate vehicles to transport tasks [Ref. 3, Ref. 4]. Specific rules can be derived that relate job scheduling and vehicle allocation problems. However in these cases the allocation of

vehicles does not directly take into account the route selection problem when multiple routes are available.

Even if the routing problem is considered separately from the vehicle allocation and job dispatching, it can turn out to be of a high combinatorial complexity. If this complexity can be solved, then a centralised control system could find a solution that selects a route for each one of multiple vehicles when multiple paths are available, while avoiding collisions and satisfying a defined overall system's performance measure. The alternative to this centralised control system normally consists of having the vehicles deciding independently but using the same rules which are designed to obtain an overall good performance. While each vehicle concentrates on achieving its own objective, it uses rules which are intended to serve a common good [Ref. 40].

2.4 Conclusions

This survey highlights the type of decisions faced by a traffic management and control system for AGVs, its interaction with job shop scheduling, and its functions within an evolution towards more autonomous vehicles. Allocation of vehicles, route selection and overall vehicle's co-ordination are the main functions of such a system. They can be placed at the higher levels of a hierarchical structure designed for autonomous vehicles, above local navigation or obstacle avoidance. Its interaction with job shop scheduling is mainly through dispatching rules and independently of the route selection problem. The high combinatorial complexity associated with either of these problems is most often dealed with using heuristic rules.

With a clearer understanding of the type of problems involved with traffic management and control of AGVs, it is possible to proceed with an analysis of ANNs as a problem solving technique which is presented in next chapter, Chapter 3.

Chapter 3

ANNs as a Problem Solving Technique

Artificial Neural Networks (ANNs) have drawn the attention of researchers from various fields, particularly in the last decade. These efforts led to new developments and extensions of ANN's paradigms and to its application as an alternative solution to a great variety of problems. This last approach was followed in the current work that explores the application of available ANNs to the particular case of traffic management and control of AGVs. In line with this perspective this chapter presents a characterisation of ANNs starting with its main principles and the associated benefits that can be exploited in its application as a problem solving technique. The evolution experienced by ANNs since they first appeared produced a great diversity of paradigms having particular characteristics that must be taken into account in the process of applying them to specific problems.

Backpropagation (BP) ANNs represent the most widely used ANNs and therefore special attention is given to its characterisation. Hopfield type ANNs are also discussed due to its inherent capability of modelling optimisation type problems illustrated by the extensive work concerning its application, among others, to the Travelling Salesman Problem (TSP) and job shop scheduling problems.

3.1 ANNs basic principles and characteristics

Current ANNs have little to do with the complexity of the human brain, but nevertheless its development sought inspiration from our current understanding of how the human brain works and why it clearly outperforms available computer systems in some tasks, such as pattern recognition. The extremely high parallelism observed in the human brain as opposed to the traditional serial processing

on von Neuman style computer systems, is the underlying principle to explain this superiority and the basis for the development of mathematical models or devices associated with ANNs [Ref. 13].

Another distinguishing feature of ANNs is the process of incorporating or programming, the required information into these systems. Again, up to a certain extent a similar process to the one that can be associated with learning in the human brain is pursued in the implementation of ANNs. Rather than using an explicit identification and understanding of a given problem, ANNs model it based on limited information such as examples of the problem and the corresponding solutions. This model of the problem also benefits from the inherent parallelism of ANNs where the information is not only associated with each of the multiple processing elements but also with the links connecting these elements. The dispersion of the information gives these systems high robustness and fault tolerance characteristics, in addition to the increased computational power of parallel processing systems and its flexibility in adapting to a particular problem.

ANNs main features derive from being a interlinked structure of simple processing elements working in parallel and using particular methods to represent a given problem into this structure. These processing elements also sometimes called, processing units, nodes or artificial neurons, typically implement a non-linearity which is considered to be an essential feature [Ref. 13]. The different ANNs paradigms represent alternative solutions in the definition of the links or connections between the various processing elements, the functions implemented on these processing elements, and the mechanism that relates their output with their respective input connections. When considering this mechanism over the global structure an ANN can be viewed as implementing a mapping function between a defined Input and Output vectors. Therefore when considering alternative ANNs it must be taken into account that a representation of the problem into such a structure is always required.

Although variations occur in the terminology used to describe the mathematical models associated with the implementation of these multiprocessor structures the following terms and definitions [Ref. 12] are useful to identify the essential aspects of an ANN and understand how they are implemented in different ANNs. This terminology is centred on the processing elements which represent the basics of ANNs.

Every processing element has a single output, and a number of inputs. These inputs represent the individual connections receiving the output of other processing elements, and possibly its own in some ANNs. A measure of the strength of the individual connections is defined by a quantity called its *weight (w)*. The *net input* to a processing element is typically calculated by summing the input values multiplied by their respective weights. The *activation value* represents a function of the net input, and possibly previous states of the processing element, but is typically equal to the calculated net input. An *output function*, also sometimes called *transfer function*, is applied to the net input resulting in the *output value* of the processing element. The process of programming these structures to model a particular problem is called the *learning or training process* and is typically accomplished by the modification of the weight values. When considering the evolution over time of these systems, the weight changes can be described by a set of differential equations representing the *learning law*. Subsequent to the learning process, the ANNs can be used to provide information, or solutions, about the encoded problem which is normally described as a *recall process*.

When multiple processing elements are organised to build an ANN structure, a variety of alternatives are typically used originating several ANN's paradigms as can be found in [Ref. 12 to Ref. 14, Ref. 41 to Ref. 45]. Basic differences occur in the definition of the topology or interconnection patterns, and in the selection of the output functions and learning law.

At the interconnection pattern's level a distinction is normally made between ANNs which allow information to flow back within the structure, and feedforward ANNs which do not allow recurrent or feedback flow of information. Another distinction results from the connectivity patterns between the many processing elements when they are organised in one or more layers. Alternatives are available that allow interconnection between elements in the same layer while others do not. Connection patterns might also be defined with *hidden elements*, that are processing elements not directly connected to the processing elements associated with the input or output of the ANN structure.

Although there are no restrictions to the type of output functions that can be explored [Ref. 14] and implemented on the processing elements, the threshold and sigmoid type functions are typically used

[Ref. 12 to Ref. 14]. They represent a form of nonlinear functions which is a main requirement for ANN's [Ref. 13], and which gives them the potential to model nonlinear problems. The use of sigmoid type functions, such as the logistic and hyperbolic tangent functions, instead of hard limiting or step threshold functions, has the advantage of using a continuous differentiable output function which is required in the implementation of specific learning laws. Moreover it allows the interpretation of the output from the processing elements to include a certain degree or level of confidence based on the processing element activation values [Ref. 14], rather than considering only two extreme values. Probabilistic type functions are also an alternative that has been explored in ANN's paradigms [Ref. 45].

The objective of the learning process is to incorporate knowledge about a problem into the ANN structure. This is commonly achieved through an appropriate selection of the weights of the connections linking the processing elements. A distinction is normally made between supervised and unsupervised learning. Supervised learning, or learning with a 'teacher', requires external information in the form of desired solutions to specific examples or instances of the problem. A measure of the difference between these desired responses and the ANN's output is used to calculate the weight changes. Unsupervised learning does not rely on an external teacher but rather on internal control to self-organise the presented data and discover its emergent characteristics [Ref. 43].

Hebbian learning [Ref. 14, Ref. 43] represents one of the original learning rules which illustrates well the idea of including experience in the learning process. This is achieved by adpating the connectivity patterns to reflect the influence of the processing elements which participate persistently in the activation of another processing element. In its simpler form the weight associated with a connection between two processing elements change in proportion to the product of their activity levels [Ref. 14, Ref. 43]. This simpler form can even be extended to a more general formulation [Ref. 14] to cover a variety of rules commonly used, such as the 'delta' rule which represents a type of supervised learning.

Another characteristic of the learning process using specific learning rules in ANNs is the iterative nature of the process which can often be viewed as searching for a particular set of values in a multi-

dimension (equal to the number of weights or connections) space. The number of calculations involved and the difficulties to solve the explicit mathematical representation associated with this search is normally dealt with iterative techniques. An example which illustrates the need for an iterative search is provided by Backpropagation type ANNs, one of the most common ANNs, that implements an iterative search algorithm based on gradient descent. The capability of the learning process to converge to a desired solution is an important issue in these type of networks. Another important aspect particularly when feedback or recurrent ANNs are used is the requirement for stability due to the dynamic nature of their operation.

The various aspects related with the structure of an ANN can be combined to produce a great variety of ANN models. What type of solutions can be expected from their application to a particular problem, and what type of problems can be cast into a ANN structure is discussed next.

In a more general view an ANN represents a process for mapping an input vector to an output vector without the explicit need for the identification of the underlying function. Based on data from the problem it implements an approximation to that function. As such it is difficult to envisage a problem that can not be represented and modelled by an ANN structure. The limitations emerge normally at the design stage where a high number of parameters is involved and no explicit rules or techniques are available [Ref. 12 to Ref. 14] to select and completely define the most suitable ANN model of the problem. Obtaining the desired level of performance becomes highly dependent on the specific design and also on the data used. Although it can be viewed as a general problem solving technique its application to specific problems rely on extensive testing and experimentation.

Regardless of the design issue the unique characteristics of ANNs, which include distributed processing over multiple elements and its inherent parallelism, make them especially capable of dealing with a set of problems generally identified with pattern recognition or classification. Image recognition is an example of these type of problems that illustrates well the power of ANNs when they are used as an associative memory [Ref. 12, Ref. 13]. In this case, the inherent capability of a ANN model to retrieve the complete information (i.e. image) even when the input data is not complete or has

been corrupted, is difficult to implement through a serial program running in conventional computers [Ref. 13].

Another set of problems which deserve extensive attention from the ANN's research community are optimisation type problems. The original idea [Ref. 46, Ref. 47] was implemented in Hopfield type ANNs, and consists of modelling the dynamic behaviour of an ANN structure using an energy function that is minimized by the changes occurring in the ANN. This energy function provides a more natural, or direct means to accommodate in an ANN the concept of a cost function and constraints normally associated with the formulation of an optimisation problem.

A special type of mapping is provided by unsupervised ANN's which are capable of extracting patterns from the data presented to the network, without the need for external information indicating whether they are correct or what they should be. This type of mapping can be used for a variety of problems including clustering, encoding, and feature mapping [Ref. 13]. The capability of feature mapping to associate similar input patterns to the physical organisation of the output elements is of particular interest to model problems that can be formulated in a two dimension space such as the well known Travelling Salesman Problem (TSP) and other similar problems [Ref. 13, Ref. 48 to Ref. 53].

A great variety of problems can therefore be approached by the several ANNs models available. The performance obtained varies and is highly dependent on a specific design. In order to have a clearer evaluation it is necessary to concentrate on the details of specific ANNs paradigms. Taking into consideration the type of problems associated with a traffic management and control system for AGVs, as described in Chapter 2, it is plausible to think that they could be formulated as a pattern recognition or an optimisation type problem. Backpropagation (BP) type ANNs were considered because they represent one of the most commonly used ANNs that has been successfully used in many typical pattern recognition problems. The direct association of Hopfield type ANNs with optimisation type problems, particularly the TSP and scheduling problems, together with the opportunity to compare a different ANN structure with BP ANNs, were the main reasons that led to their selection for a more detailed analysis. These more detailed analysis is presented in the next two sections, starting with BP ANNs.

3.2 Backpropagation type ANNs

The development of the Backpropagation ANNs played a central role in ANN's research and application areas. It represents a supervised learning type ANN, whose distinguishing feature lies in the learning algorithm that overcame the limitations of learning in hidden processing elements based on the global output error measure. Up to its development the basic processing element, the perceptron, was already available but it could only solve classification or decision problems which were linearly separable [Ref. 13]. Provided the input and output pairs or patterns used during learning were linearly separable the perceptron's learning rule was proved to converge in a finite number of steps to accomplish the desired association [Ref. 13]. However a simple problem like the Boolean exclusive 'OR' function could not be modelled by a perceptron. The use of hidden layers could provide the extra flexibility for separating more complex decision regions. However a method was required to allocate an error measure to the output of the processing elements in these hidden layers, required to calculate the changes to the weights of their respective input connections. Backpropagation provided such an algorithm.

The structure of this Backpropagation ANN [Ref. 13, Ref. 14] consists of a multi-layer, feedforward network, with a supervised learning algorithm. Being based on perceptrons it is often also referred to as the feedforward multi-layer perceptron network. The learning algorithm uses the generalised delta rule [Ref. 12] and can be viewed as implementing the minimisation of a cost function, representing an error measure between the desired outputs and the ANN current outputs, through an iterative steepest-descent gradient technique [Ref. 12 to Ref. 14]. The changes to the weights are made in proportion to the gradient of the function at that location. One requirement of these networks is the use of a continuous differentiable output function in the processing elements; nonlinear sigmoid functions such as the logistic function, are normally used.

The basic learning procedure in BP ANNs [Ref. 12] consists of presenting an input pattern to the ANN, process these values through the network using the current weight values, calculate the error terms for the output processing elements based on the current and the desired output values, update the

weights on the output layer and update the weights on the hidden layer. When the error is acceptably small for each of the training patterns, learning can be discontinuous.

Backpropagation (BP) ANNs illustrate the use of an ANN structure that is capable of approximating a mapping function from an input to an output vector. Although the number of hidden processing elements needed to implement this approximation is not known in general, it has been proved that only one hidden layer is enough to approximate any continuous function [Ref. 13].

How well the mapping implemented in a BP ANN suits a particular problem depends on many issues. Starting with the representation of the problem into the BP ANN structure, there might be different possible encodings of Input and Output vectors that can be associated with information about the problem. Different encodings translate into different patterns presented to the network. In addition to the selection of an encoding scheme, it has to be decided what and how many examples of the problem should be used.

Having found a representation scheme that defines the processing elements in the network's input and output layers, the design process continues with the selection of the parameters which allow the network to accomplish the desired mapping. The number of hidden layers and hidden elements must be selected. The learning process being an iterative procedure requires the specification of the number of steps, or minimum error levels, that allow learning to stop; specifying a learning rate is also required to control the steps in each iteration. In addition to the learning rate a momentum term is often used that includes in each weight change a contribution from the previous time step.

When the training or learning process ends, it becomes necessary to evaluate the performance of the BP ANN when patterns that were never used during training are presented at its input. This gives an indication of the generalisation obtained. The difficulties in designing a BP ANN solution derive from the various decisions and their interrelation, and the non existence of well proven methods or techniques to assist in these decisions apart from extensive replication of experiments and evaluation.

Despite the design difficulties its potential for solving different problems is illustrated by the variety of applications that have been reported [Ref. 13, Ref. 41] and which include among others image compression, speech recognition, signal prediction and forecasting, sonar target recognition, recognising hand-written numeric postal codes, navigation of a car, encoding problems. A number of solutions more directly related with job scheduling and AGVS or mobile robots have also been published [Ref. 54 to Ref. 59].

Various developments [Ref. 13, Ref. 45, Ref. 60] have been implemented to improve the performance of BP ANNs and reduce the training times. These include other minimisation procedures than simplest gradient descent [Ref. 13], involving methods which make use of the second-derivative (i.e. second-order methods) of the gradient [Ref. 45], or using heuristics to adapt the learning rate as during learning [Ref. 60]. The results obtained are nevertheless highly problem dependent [Ref. 45].

3.3 Hopfield type ANNs

A fundamental difference between Hopfield ANNs and the BP ANNs described in the previous section (section 3.2), is the use of feedback or recurrent connections that link the output from every processing element to the input of all other processing elements. The dynamic behaviour inherent to these feedback connections exist even when the network is in a recall mode, being necessary to wait until the network stabilises to extract information from its processing elements. In a trained BP ANN a recall was deterministically defined. Another characteristic is the freedom to choose between a synchronous or asynchronous updating of the processing elements.

There are two main applications for these networks, one as an associative memory and the other to solve optimisation type problems [Ref. 13, Ref. 41, Ref. 46]. The application to optimisation type problems has a particular interest deriving from the direct relationship that can be established between the cost function representing the optimisation problem and the function controlling the dynamic behaviour of the Hopfield ANN. The original application to the Travelling Salesman Problem (TSP) [Ref. 46] and subsequent developments [Ref. 61 to Ref. 64] which include applications to job shop

scheduling and other similar problems [Ref. 65 to Ref. 73] illustrate its importance as an alternative approach to solve very difficult problems due to the high combinatorial complexity involved.

As an associative memory, Hopfield ANN use the Hebb rule [Ref. 13] to select the weights based on the patterns to be stored. After a new pattern is initially imposed to the network, it will iterate in discrete time steps until changes no longer occur, corresponding to the stored pattern. The storage capacity of these networks has been investigated and various upper bound limits to the number of patterns that can be stored by a number of processing elements have been presented [Ref. 13].

The application of Hopfield type ANNs to solve optimisation problems is based on expressing the dynamic behaviour of the network by a function which always decreases, or remains constant, as the system evolves. Hopfield [Ref. 46, Ref. 47] proved that such a function exists provided the weights were symmetric, and making an analogy with physical systems called it an energy function. Initially binary processing elements, with asynchronous updating were considered, subsequently replacing the binary elements with a continuous function (sigmoid) a continuous version of the Hopfield ANN was developed. The continuous version formulation contains the discrete version as a special case [Ref. 12]. These functions are also called Lyapunov functions which results from satisfying the Lyapunov method for convergence and stability [Ref. 12, Ref. 13, Ref. 43].

With the introduction of the energy function the emphasis can now turn to changing the weights in the Hopfield ANN in a more direct association with the representation of a problem by a cost function and its constraints, rather than training based on example patterns. The design problem now becomes matching the problem's cost function and constraints to the energy function of the Hopfield ANN. The evolution of the network minimising its energy function will also minimise the problem's cost function. The difficulties associated with this process lie on the requirement to include in the problem's cost function terms representing the constraints or restrictions of the problem. The minima obtained by the network might not correspond exactly to the minima of the problem's cost function because of the influence of the constraints terms. Moreover it might be difficult to guarantee that the restrictions are completely satisfied.

These type of problems were highlighted by [Ref. 74] which could not replicate the results obtained by the original application of the Hopfield ANN to model the TSP problem. These problems were overcame later with a re-analysis of the Hopfield ANNs, and the re-definition of the coefficients used in the cost function [i.e. Ref. 61, Ref. 63, Ref. 13]. Although Hopfield ANN can be used to solve optimisation type problems it must be taken into consideration the difficulties associated with the formulation of a cost function that guarantees the desired performance.

3.4 Conclusions

From this overview of ANNs it seems clear that they can be viewed as a general problem solving technique, with specific characteristics that can be used with advantage to solve problems. However it seems also clear, particularly from an analysis of the two ANNs models described in more detail, that designing a ANN solution which performs as desired, is in itself a non trivial problem.

The description made of the BP and Hopfield ANNs also shows that different approaches can be followed when modelling a problem into a ANN solution. Although they both can be considered as implementing a technique to search for minima of a function, one is based only on changing the weights (BP) while the other also depends on the states of the processing elements (Hopfield ANN). While BP ANNs require examples and respective solutions about the problem, Hopfield ANNs involve the formulation of a cost function. Considering the type of decisions associated with a traffic management and control problem as described in the previous chapter, Chapter 2, it can be envisaged that pattern recognition and optimisation are involved and therefore both ANN alternatives could be considered. However as in many other ANN's applications the success obtained will be highly dependent on a specific design and problem formulation. The formulation issue will be dealt with in next chapter, Chapter 4.

Chapter 4

Characterization of the Problem

The previous analysis of the various aspects of traffic management and control of AGVs (Chapter 2), and of the characteristics of ANNs as a problem solving technique (Chapter 3), provided better grounds on which a more detailed formulation of the problem and the selection of a specific ANN to model it can be made. The main issues involved will be whether to separate or not the job shop scheduling and the traffic management problems, and also whether the problem should be formulated as an optimisation type problem or as involving a pattern recognition problem. The performance obtained when modelling a problem into an ANN structure is not only dependent on the type and complexity of the problem. It is also important the particular representation into the ANN structure, which in itself is one of the design parameters involved. A detailed analysis and formulation of the problem become thus even more significant. In the next section this detailed analysis is presented. Subsequently the approach chosen to model the problem into an ANN model is selected and described.

4.1 Problem formulation

One major issue in a detailed formulation is the relationship between the job shop scheduling problem and the traffic management and control problem. It is obvious that the job shop problem and the vehicle allocation/dispatching problems are in fact intimately related and could be formulated as only one optimization problem. Another possibility is to consider the problems independently in which case the traffic management and control system would be concerned mainly with the efficiency and

capability of the transport system to satisfy the requirements of the scheduling system. In such a case its major functions would be route selection, vehicle dispatching and traffic management (Chapter 2).

A first indication of the type of problem can be given by a general formulation of the problem as involving the optimum (or near optimum) allocation/dispatching and route selection of 'p' AGVs among 'm' workstations in order to accomplish 'n' tasks. This general formulation implied an optimization type problem and also the consideration of the job shop scheduling problem together with the vehicle allocation or dispatching and route selection problem. As an optimization problem it can be considered an np-complete problem [Ref. 75], as the number of alternatives when at a decision point increases exponentially with the size of the problem thus limiting the practical use of mathematical optimization techniques.

A more realistic insight on what is involved in the formulation of the problem when both systems are considered, either simultaneously or independently, can be obtained by the analysis of the type of decisions involved and the conditions on which these systems operate. When job shop scheduling and traffic management and control are considered simultaneously, the major elements of this analysis can be summarised as follows:

- the scheduling problem can be considered as a real-time, dynamic scheduling system for a job shop type FMS operating in a random mode (Chapter 2);

- all the information which defines the possible sequence, or alternative sequences, in which the operations of each job must be executed is defined previous to the release of the job orders. Other information such as due-dates, quantity of production, priorities, are also defined in the pre-release plans;

- scheduling jobs will consist of selecting for each part the workstation most suitable for the next operation from among all the possible alternatives, and the starting time for its execution. The transport time between workstations must also be taken into account, and will be dependent on the availability of AGVs, the selected route and the traffic conditions at each instant;

- the decisions concerned with this problem are affected by the dynamic evolution of the system during its operation. Factors which contribute to this dynamic behaviour are: uncertainties related to workstations or AGV breakdown, variable execution times at the workstations, variable transport times, availability of resources, change of production objectives and priorities;

- the type of decisions in the scheduling, vehicle allocation and dispatching problem could be for each operation of each job related to:

  - which workstation should the part go to?

  - when it should go?

  - which AGV should be selected to transport the part?

  - which route the AGV should follow?

  - also of all the parts, which should be selected first?

- several criteria could be used to measure the performance of a manufacturing system and its scheduling system being most important [Ref. 3]: meeting due-date, maximisation of system/machine utilization, minimisation of work-in-progress.

When the problems are treated independently, with the traffic management and control system concerned with the operation and performance of the transport system to satisfy the transport orders or requirements of the job shop scheduling system, the following aspects must be considered:

- the characterisation of the physical path network, such as uni or bidirectional paths, number of crossings and alternative ways out of a crossing, docking or overtaking points;

- alternative paths to link each pair of workstations;

- route planning: selects a unique route for any given vehicle, and can be viewed as choosing the shortest path, or the shortest time for a vehicle to reach its destination. In order to account for time

dependent events and avoid track congestion it must be planned in a dynamic mode considering multiple objective functions and constraints;

- vehicle dispatching: decides which vehicle to choose for a particular operation. Again in order to account for time dependent events such as blocking, and the current location and status of the vehicles (i.e. the nearest vehicle may not be the one which will take the least time to execute the transport order), the change of priorities of the transport orders, it should be solved using a dynamic model;

- traffic management: deals with the control of the various vehicles in the route network;

All aspects of the transport system are interrelated and should be considered at the same time. A key issue appears to be the ability to quickly and accurately forecast the presence of vehicle conflicts in order to prevent them through flexible route planning, vehicle dispatching and traffic control.

As described before (Chapter 2) most systems normally use heuristic rules to deal with these problems, which in the case of a traffic management and control system basic functions the various levels of complexity or alternatives considered normally include [Ref. 40].

- the use of restricted paths;

- the use of global rules that every vehicle, with equal priorities and taking no account of future implications of its decision, will apply each time it has to decide which of the alternative paths it should follow (or stay at the same location) in the next move step when the desired one (i.e. the one closer to the target) is blocked;

- the use of a centralised system which is able to evaluate the overall system status and decide the best next step for each vehicle and/or decide when and what type of general rules should be applied in order to satisfy an overall objective.

From the previous analysis it seemed very difficult to envisage a solution or technique that can cope with this high combinatorial complexity, to produce an optimum solution, taking into account the high number of variables involved. Even if that is possible than there is a time constraint which must be taken into account due to the dynamic characteristics of manufacturing systems, which most probably would turn that optimum solution not adequate to the current existing conditions.

Formulating the problem as an optimisation problem normally involves defining a cost function which includes all relevant variables, and which is to be maximised/minimised according to optimisation criteria. Too many variables and constraints are involved which makes the problem very complex to formulate and solve as an optimisation problem by identifying and selecting the best alternative/solution among the possible ones. Separating the job shop scheduling problem from the traffic management and control problem reduces the number of variables. However even when separated from the scheduling system, the decision problems associated with vehicle allocation, route selection and traffic management represent highly complex problems, particularly when they are considered simultaneously in highly flexible transport systems.

Based on these considerations an approach was followed in which the two problems are separated and only considering directly the route selction problems. The objective was to develop a system that was capable of dealing efficiently with flexible routing so that the job scheduling system could reliable count on a transport system which was not impairing the flexibility of the all manufacturing system.

The problem was then considered as a route selection problem in a system with multiple vehicles and where multiple routes are available. Each vehicle has assigned a target location and requires movement directions that lead it to the target or destination location.

This problem can be formulated as an optimisation type problem involving the definition of a cost function that when minimised would represent global optimum performances. Alternatively, when thinking of vehicles moving around a network of paths in which conflicts and congestions must be prevented and avoided it seems also reasonable to consider that the vehicle's movements can be associated with patterns which if identified could be the basis to avoid those conflicts and therefore

provide the means to improve the performance of the system. This later approach fits well in the objective of having the capability to respond rapidly and adapt in real-time to dynamically changing environments. The availability of ANNs (i.e. BP ANNs) which have been extensively used in pattern recognition problems provide the opportunity to explore this later approach. The problem has therefore been considered has a recognition type problem. Its formulation will then involve the characterisation of the status of the transport system and of the associated movement decisions for the vehicles.

4.2 BP ANN and alternative techniques

Having favoured the formulation of the problem as involving the recognition of patterns, the Backpropagation type ANN was selected to model it. It represents one of the most commonly used ANN [Ref. 12 to Ref. 14, Ref. 41, Ref. 45] in a variety of problems, in spite of its associated problems. These can be associated with the high number of design parameters involved, the lack of explict rules which guarantee their adequate specification, the possibility to converge to local minima, the slow convergence, long training times. However it still remains a highly effective paradigm [Ref. 45] and capable of providing accurate results in mapping nonlinear relationships. Its rapid computation in recall mode assumes particular importance in the envisaged approach to model a real-time route selection system in a dynamic environment. This provides an advantage over other more recent ANNs which compete with BP ANNs, such as Radial Basis Function Networks [Ref. 45].

There are other alternatives to ANNs, as a problem solving technique, that could be considered to model the route selection problem. One of these is represented by Genetic Algorithms which represent search algorithms based on the mechanics of natural selection and natural genetics [Ref. 76, Ref. 77]. They also require the codification of the problem in a specifc format, typically a string of '0's and '1's, and the definition of a cost function. Particular instances of the problem and respective cost values are generated. The algorithm is very effective in searching over that space of solutions. They can be viewed more as an optimisation type technique requiring a number of iterations to arrive at a good solution. However the proposed formulation of the route selection problem is to be used in a more dynamic and real-time mode, in which case the fast recall of BP ANNs fits better. Another alternative

which have many similarities to ANNs are Fuzzy Logic systems and an application to AGVs dynamic scheduling has been reported [Ref. 78]. In particular they also present theoretical and practical changes requiring a great deal of experimentation to determine the membership functions [Ref. 45]. It was decided therefore to proceed with the application of BP ANN to model the route selection problem and that is the objective of the following chapters.

Chapter 5 | Strategy for the Application of Backpropagation to the Single Vehicle Case

This chapter describes the approach chosen to model the single vehicle case using a backpropagation type ANN. It starts with an explanation of the basic idea behind the solution envisaged for the problem, and which consists of making use of the solution for the single vehicle case as the basis for a solution of the multiple vehicle case. Next it focuses on the process of representing the problem of the single vehicle case using a backpropagation type ANN. In order to implement and test ANN based on the representations proposed it was necessary to generate test data, thus the algorithm and conditions used to generate the required training and testing data are also described.

5.1 Basic Approach

The process of modelling a problem using a backpropagation type ANN must take various factors into consideration. Of most relevance are the ones related to the type and formulation of the problem, including those related to its representation using an ANN architecture. Both sets of factors will have implications for the size of the ANN and also on its ability to provide the desired solutions to the problem. The approach chosen and described below was derived from considerations relative to three factors:

- local versus global information about the system;
- single versus multi-vehicle cases;
- and characterization of specific transport system layouts.

5.1.1 Local versus global information

In the single vehicle case we would wish to obtain the most general formulation of the solution to the problem. However the implications this formulation will have on the implementation of an ANN architecture, particularly on the number of input and output elements required, must be considered. As a result an approach was chosen in which it was possible to model progressively more complex formulations of the problem. The more complex formulations are associated with the consideration of several vehicles, and also with the size of the physical transport system layout. The larger the transport system, the more information can be considered in the problem representation (i.e. the status of the elements in the physical layout), and consequently a larger ANN would typically be required.

The approach chosen consists of developing a solution for a single vehicle case which can be used as part of the solution in a multiple vehicle system. It is also based on an assumption that in a larger system not all the information about the transport system has the same importance when a vehicle has to make a local movement decision. For example if a vehicle is well away from its target location then what happens there, at that time, might be of little interest compared with the status of the elements in its immediate vicinity. It is reasonable to expect that more information than just the information of the adjacent elements must be considered. It also seems reasonable to expect that a limited area around a vehicle may provide information to support decisions that are intended to satisfy an overall common good. If it was possible to identify and model all the information which characterizes the transport system then it would be possible to plan paths for each vehicle, optimally satisfying the global objectives. Furthermore it must be possible to generate these solutions in an acceptably short time because in a practical system conditions change with time. Therefore the approach chosen is based on the belief, that a more global and general solution could be developed from a solution to a more localized problem.

5.1.2 Specific layout

At this point the main objective is to develop a control system capable of making real time decisions to manage the movements of multiple AGVs in any layout. It is also assumed that the logistical requirements were determined independently by the production schedule. However in order to proceed with the design and implementation of a Backpropagation ANN it was decided to start with a specific layout. A transport system was used that was based on a rectilinear grid consisting of "nodes" linked by "tracks". Inspired by what a practical case would require, two types of nodes were considered: the origin or destination nodes which are associated with machine locations and therefore with only two alternative movement directions; and the other junction nodes where a routing decision may be required. A track is a bi-directional path, one vehicle wide, that links any two nodes; and all tracks are of the same length. The minimum size of layout that contains all possible junction types is shown in Figure 5- 1. This layout is representative of a much larger grid and the results obtained from it are expected to be applicable to larger problems.

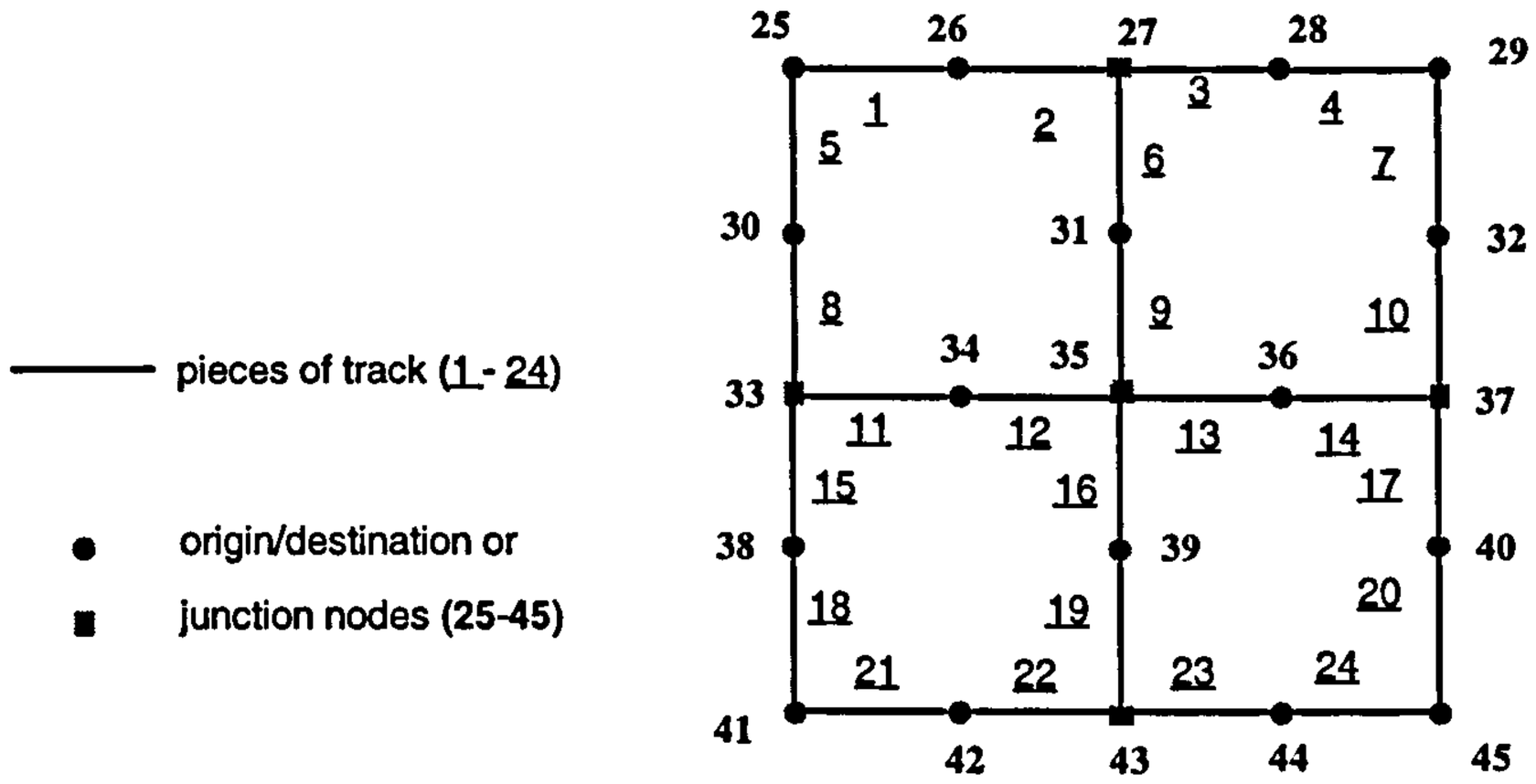


Figure 5- 1: Physical layout.

5.1.3 Progressing from single to multi-vehicle cases

On the basis of modelling with progressively greater complexity, the simplest problem is to consider the case of routing a single vehicle from an origin to a destination. This is likely to be of little practical interest but is a building block in our progress toward a solution to the multiple vehicle case. In order to make this progress possible it is vital to consider the interaction with the decisions of other

vehicles. This will introduce not only the need to evaluate the status of the system at each stage but, also what will happen at a later stage.

In this strategy we will firstly consider the possibility of using the solution for the one vehicle case iteratively for each vehicle when they have to make a decision. The interaction between the vehicles is accounted for by temporarily assigning unavailability states to those tracks which other vehicles occupy, or are intending to occupy. This method can be used as a look ahead policy and, together with simple heuristic rules, to provide solutions to successfully route each vehicle to its destination.

In a multiple vehicle situation a solution based only on the ANN solution to the single vehicle case will most likely lead to solutions which optimize individual paths rather than global or overall co-ordination of the several vehicles. In order to overcome this limitation, information will be required relative to future occupation of tracks. The strategy proposed to account for this limitation is to use a look ahead policy and a hybrid system based on simple rules to interpret, analyse and update the information required for the single vehicle neural network (Figure 5- 2). Various alternatives relative to the interpretation and usage of the unavailable tracks can be explored in order to develop a system to provide close to optimal solutions in a multiple vehicle scenario.

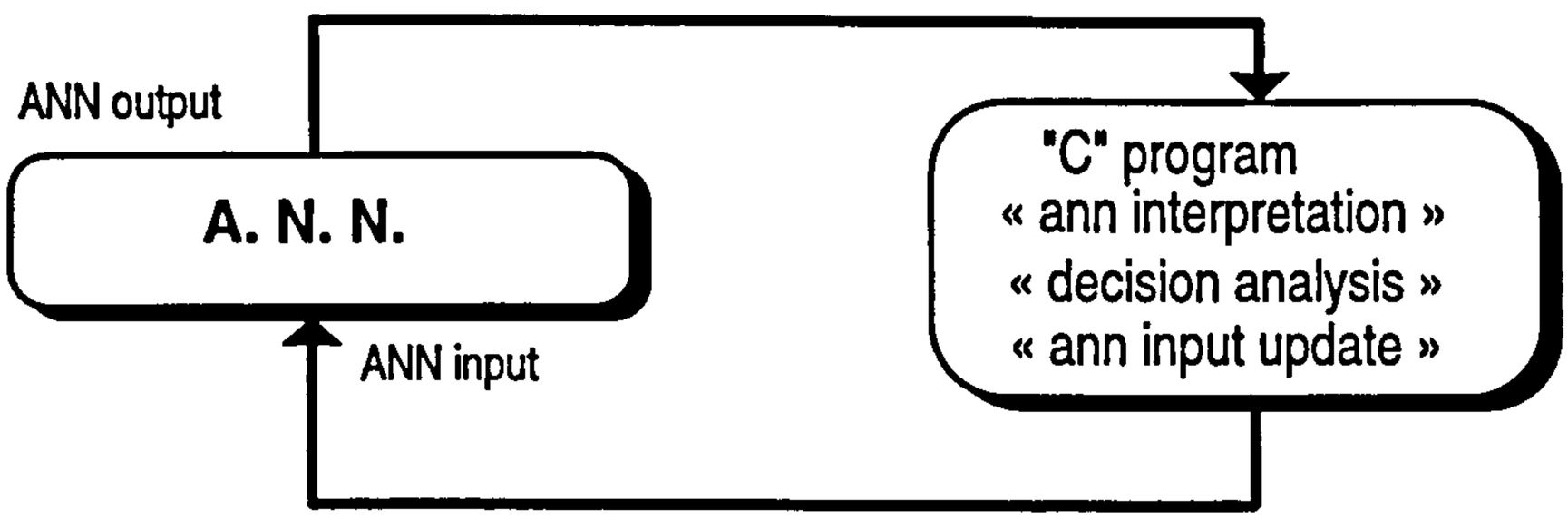


Figure 5- 2: Hybrid solution in a multiple vehicle scenario.

5.1.4 Development of the BP ANN solution

Following the established methodology and the chosen transport system layout (Figure 5- 1) the development of a BP ANN model of the single vehicle case was undertaken. Three main issues were involved:

- choosing a representation of the problem which is suitable for the BP ANN architecture;
- generating data for training and testing the ANN;
- optimizing the values of the design parameters of the ANN through successive training and testing stages.

One of the most important aspects in the development of a solution using ANNs is the particular representation or mapping of the problem onto an ANN architecture. This involves the identification of what type of information the neural network is given and what information is obtained from its output. By defining the type of information and a specific encoding we are defining how the neural network will perceive the information about the problem and therefore also its ability to distinguish different scenarios and learn how to provide solutions to them. If it is possible to find a representation and encoding of the information that makes it easier to distinguish between different cases, then the ability of BP ANN's to identify the desired pattern associations between its input and output elements demonstrated in various applications [i.e. Ref. 12] can be better exploited.

Once a representation is selected, the next phase is to select the values of the design parameters for the backpropagation network so that its performance is optimized. The basic process is mainly based on successive experimentation with arbitrarily set values. Some heuristic rules have been reported [Ref. 12, Ref. 13, Ref. 60, Ref. 79 to Ref. 81] but they are normally problem dependent. The procedure used to select the backpropagation design parameters (*number of hidden layers* and *hidden elements*, *learning rate*, *momentum term*, *epoch size*, *number of training iterations*, *different initial random values*) in these experiments, was to test each one independently for a set of pre-defined values, evaluate performance and define new values to be tested.

For the generation of the required training and testing data an exhaustive search type algorithm was implemented as described in section (5.3). For the particular layout chosen (Figure 5- 1) pseudo-random numbers were used to generate pairs of origin and destination nodes and the tracks which would be set as being temporarily unavailable. All possible paths, without repeated tracks or nodes, linking the origin and destinations nodes were found and all necessary information was stored in files for use during the training and testing phases of the ANN design.

5.2 Representation of the problem in a way which is amenable to solution using a Backpropagation ANN

The representation of the problem in a BP ANN consists basically of the definition of the ANN input and output elements. In most cases there will be various alternatives as to how to represent the solutions to a problem using an ANN. Variations might occur on what information about the problem, and what specific encoding of that information, should be provided to and obtained from the ANN. For the single vehicle case and the specific layout selected for the chosen methodology it is possible to consider an analysis of different possible representations based on the following two considerations:

- will the ANN be designed so its output can provide a complete specification (i.e. all the tracks to be used) of the proposed solution path? Or will it be designed instead as an output which only provides the direction to be followed in the move from the current node and therefore requiring iterative use of the ANN at each current node until the destination node is reached?

- when various solutions, i.e. paths linking the two nodes are possible, which ones should be used to train the ANN? Will only the optimum routes, i.e. the ones corresponding to the shortest length in the single vehicle case be used, or alternatively will "solution paths" with different lengths be used? It must be noted that in a 'grid type' layout there are usually many equally short routes between two nodes, specially if only one vehicle is involved and all tracks are available.

These considerations can be analysed from the perspective of the design of a BP ANN that has the ability to provide information in a single vehicle case, and also from a perspective of how useful that

information will be in a multiple vehicle scenario. We have to investigate whether or not the use of different types of information, such as when paths of different lengths are considered, provides a BP ANN which works best, i.e. that we can find the optimum solution paths easily. However, the explicit inclusion of more information about a particular problem into a BP ANN requiring more elements normally leads to a larger network (ANN), and consequently will normally require larger amounts of data for training and testing it. In the envisaged "hybrid" approach to the multiple vehicle case, a BP ANN which has the ability to distinguish between and select paths of different lengths can provide support for the consideration of a higher number of alternatives in the movements of the several vehicles. On the other hand increasing the number of alternatives to be analysed when making decisions may contribute to the approach suffering from scalability problems. Based on this analysis it is not possible to discard any of the approaches.

5.2.1 Representation 1: complete path specification and using only optimum paths

In this first representation of the problem of the single vehicle case using a BP ANN, the output elements must represent information which identifies a path linking two nodes in a specific layout. The input elements must be capable of representing the state of each track in the layout and an identification of which nodes have to be linked by the solution path. Also only optimum solution paths will be used to train and test this ANN.

The representation chosen and particular encoding used consists of only binary elements and can be interpreted as follows (see also Figure 5- 3):

Input: 40 elements whose values can be "0" or "1".

The first 24 each correspond to one section of track (i.e. first element corresponds to track number 1), a "0" value means available, a "1" value means unavailable. The other 16 elements (25 to 40) represent each one of the existing nodes which can be used as origin or destination nodes. Only two of these nodes can have a value "1", representing the nodes to be linked, the others should be "0". No distinction is made as to which is the origin or the destination node because the path should be "reversible".

Output: 24 elements whose values are in the interval [0, 1] and are associated with each of the existing tracks in the physical layout, i.e. the first element corresponds to track number 1. Each value will be converted, after specifying a threshold value, into a value of "1" which indicates a track which is used in the solution path, and a "0" value indicating a track not used in the solution path.

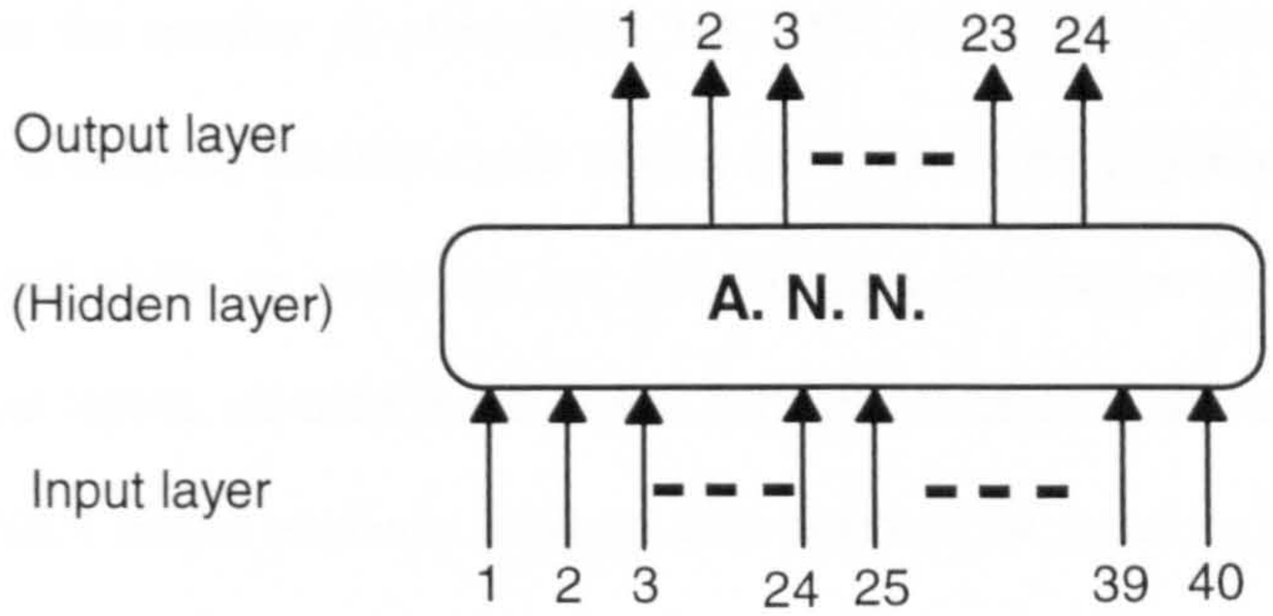


Figure 5- 3: Backpropagation neural network Representation 1

An example showing the input/desired output vectors in a particular case (and encoding) as represented in Figure 5- 4 is described next,

INPUT vector: elements 1,...,24: 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0

elements 25,...,40: 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0

DESIRED OUTPUT vector:

elements 1,...,24: 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0

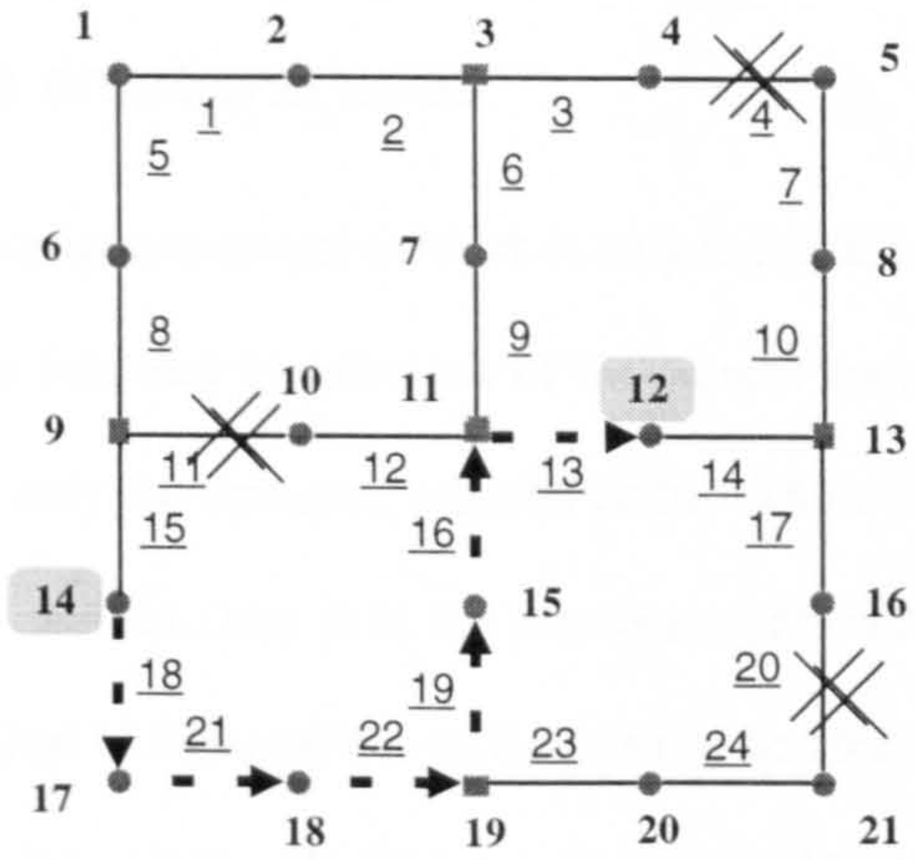


Figure 5- 4: Optimum solution path linking origin node (14) with destination node (12) when tracks (4, 11, 20) are unavailable.

One problem with this representation which can be anticipated by considering the example shown is the relatively large number of elements in the output, and even more in the input, which do not change state when cases change.

In this representation the number of elements in the ANN output were chosen to minimise the information required to uniquely identify a path linking two nodes in the physical layout. When paths which include repeated tracks or nodes are not allowed, an identification of which of the tracks, defined in the physical layout, are used in the path is sufficient to characterize a path. The minimum number of the BP ANN output elements required will therefore be equal to the number of tracks defined in the physical layout. The possibility of the ANN producing impossible solutions, i.e. indicating tracks which do not form a continuous path between the origin and destination nodes must be considered because it only depends on how well the BP ANN learned the possible combinations of solution paths.

The number of elements defined in the input of the BP ANN was based on the minimum number of binary elements required to represent the state of each track defined in the physical layout with the additional assumption that no distinction is made between the origin and the destination node.

5.2.2 Representation 2: next step specification and only optimum solutions used

Here the output of the backpropagation neural network is only required to give an indication relative to the direction which should be followed to move out of the current node. As in Representation 1 the training/testing data includes only the optimum solution paths. The input information indicating that the tracks are not available is also the same as in the previous representation but the input information indicating the nodes to be linked is different due to the need to be able to identify which is the origin node and which is the destination node. Having chosen to provide only information relative to two nodes it was possible to use an encoding system which uses less elements than in the encoding used in Representation 1. A five bit binary word was chosen to represent a number associated with a node in the numbering system in the physical layout. One group of five elements (bits) was used to represent

the origin node, followed by a second group of five elements (bits) associated with the destination node. In a step by step solution the origin node is updated, after each stage step it becomes the current node. This made it necessary to use five bit binary words to represent all of the twenty one existing nodes (i.e. origin/destination and nodes at junctions). In a step by step definition of the "solution path" we must take into account the fact that the current node may be one of the junction nodes. This particular encoding is described as follows (see also Figure 5- 5):

Input: 34 elements whose values can be "0" or "1".

The first 24 correspond to each one of the existing tracks on the physical layout. A value of "1" means track unavailable, a value of "0" means track available. The other 10 elements (25 to 29, and 30 to 34 in Figure 5- 5) represent two five bit binary numbers, which are the numbers associated with the origin (or current) node and destination node respectively in the physical layout.

Output: 2 elements whose values can vary in the interval [0, 1]. These values are associated with two co-ordinates in the plane (x,y). The length and direction of the vector defined by these values will be used to define which of the five alternatives is indicated by the backpropagation neural network: move *upwards*, *downwards*, *right*, *left* or *stop*. This interpretation of the neural network output is illustrated in Figure 5- 6. It can be seen that with this interpretation we have reduced the possibility of having impossible solutions and also it profits from a consideration of the continuous values given by the neural network instead of converting each output element to a "0" or "1" using a threshold value of 0.5 (i.e. "0" if less than 0.5; "1" if equal or greater than 0.5). The continuous values of the output elements of the network result from the requirement to use a continuous function to produce an output at the nodes, such as the sigmoid function which produces a range of values from 0 to 1. When we are interested in a binary type information from an ANN output element a threshold value (i.e. 0.5) is specified to convert the continuous values obtained from each ANN output element to binary information. However in this representation the two ANN output elements are associated with the two co-ordinates of a point in the xy-plane which defines a vector from its origin to that point. If we consider the origin of the xy-plane to be the current node where one of the possible moving alternatives out of that one must be selected, we can associate the vector obtained from the ANN to a moving direction. There is no need to apply a threshold directly to the values obtained from the ANN

output elements because any point in the plane can be used to define the direction vector. Threshold values in this case will only be required to classify the angle and the length of the moving direction vector to restrain the moving directions to the four available (move up, down, right, or left) or to stop. It is possible then with this interpretation to make use of a wider range of values obtained from the ANN output elements than if threshold values were applied to convert each ANN output element to a binary format.

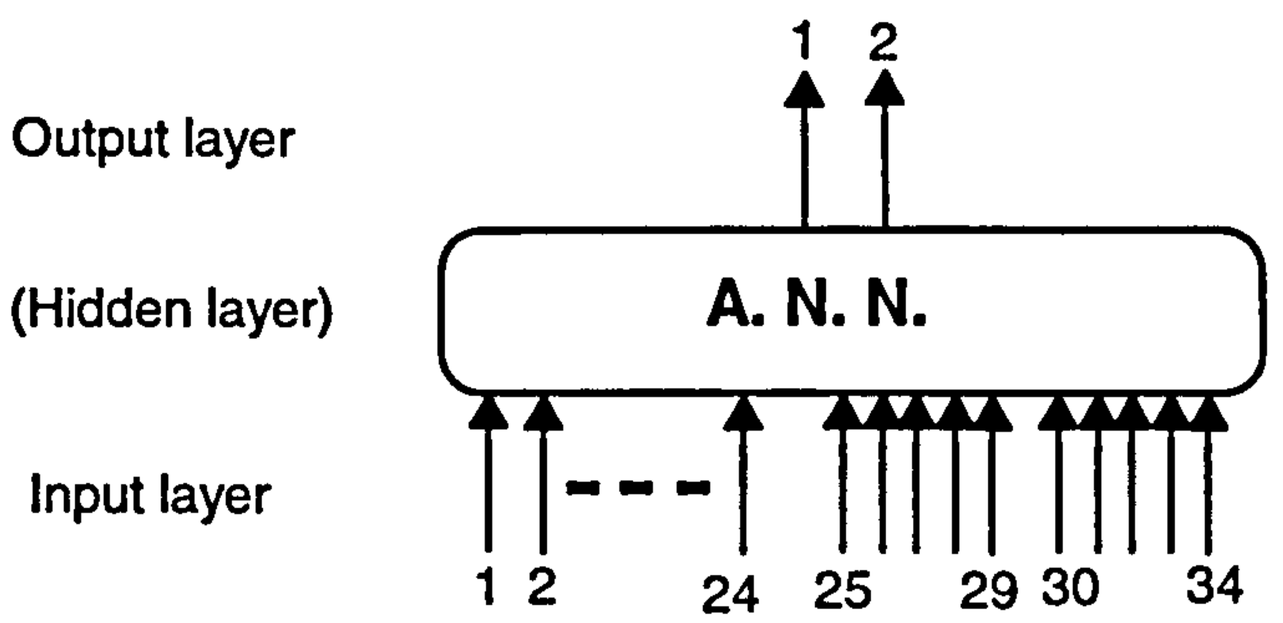


Figure 5- 5: Backpropagation neural network Representation 2

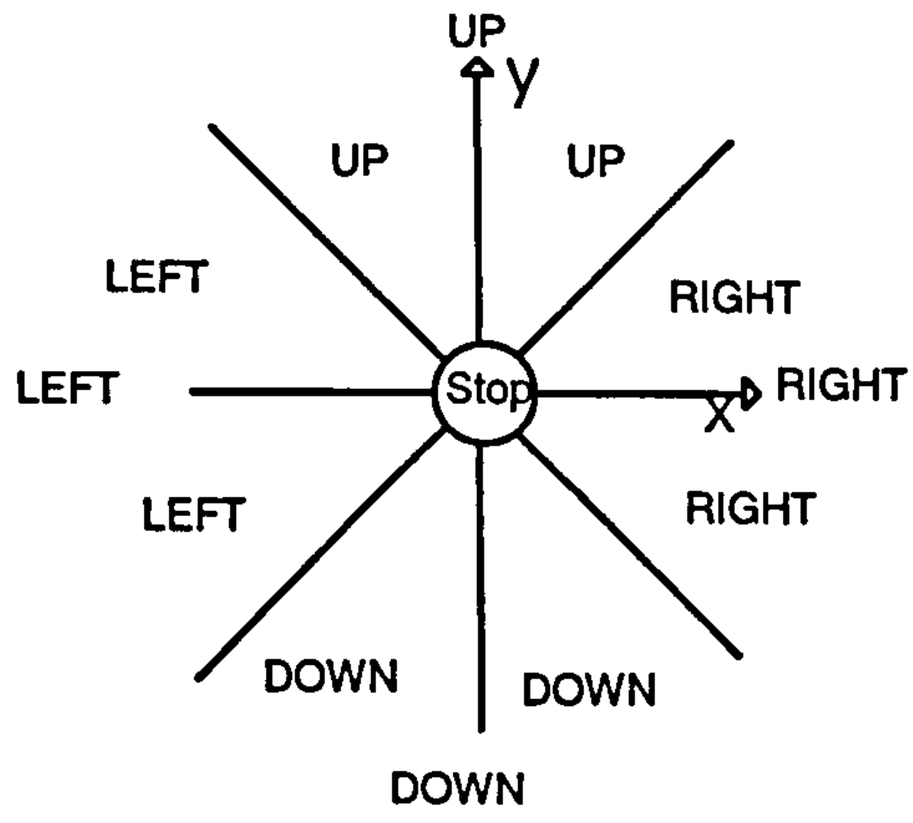


Figure 5- 6: Interpretation of neural network output (Representation 2)

Using the same case as the one used to illustrate Representation 1, the ANN of Representation 2 could be used in a sequence of steps as described below to provide the solution for this particular case (see Table 5- 1 and Figure 5- 7).

Table 5- 1: Exemplification of input/desired output elements of the ANN Representation 2.

Origin: node 14 Target: node 12	**INPUT** vector elements:			Desired **OUTPUT**	
	1 ... 24	25...29	30...34	(x , y)	(direction)
step 1	000100000010000000010000	**01110**	01100	**0.5 , 0**	**Down**
step 2	000100000010000000010000	**10001**	01100	**1 , 0.5**	**Right**
step 3	000100000010000000010000	**10010**	01100	**1 , 0.5**	**Right**
step 4	000100000010000000010000	**10011**	01100	**0.5 , 1**	**Up**
step 5	000100000010000000010000	**01111**	01100	**0.5 , 1**	**Up**
step 6	000100000010000000010000	**01010**	01100	**1 , 0.5**	**Right**

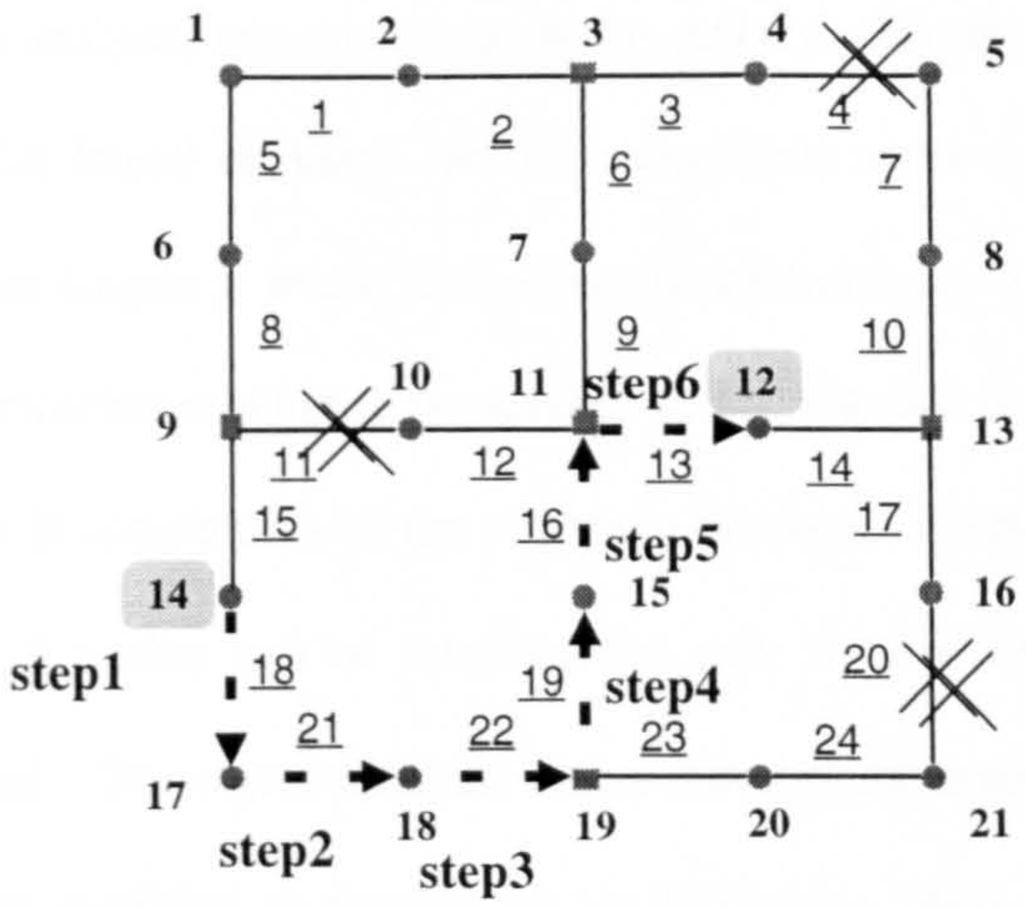


Figure 5- 7: Optimum solution path linking origin node (14) with destination node (12) when tracks (4, 11, 20) are unavailable.

One of the characteristics of this representation relative to the one described previously, is the reduced number of elements, and consequently number of different combinations, of the output of the ANN. This also means training/testing the ANN will be more effective because fewer patterns are required. By reducing the possible combinations of the output the possibility of the ANN indicating an impossible solution is reduced, and also even if an impossible solution is given it will be easier to detect. This is an important aspect and combines well with the goal of using the ANN in a multiple vehicle situation: local decisions are made that can take into account other vehicles intentions using look ahead strategies (section 5.1.3, page 39).

5.2.3 Representations that make use of cases involving consideration of solution paths with different lengths

In the previous two representations the idea was to have a backpropagation ANN which was capable of learning how to generate optimum solutions by training it only with cases where the solution was optimum. However in a multiple vehicle situation it is likely that not all vehicles will always be able to travel along a minimum distance path due to interference from other vehicles. It is therefore necessary to consider sub-optimal (i.e. longer than minimum distance) paths using an external system, that is the system which analyses and complements the ANN decisions in a "hybrid" solution (Figure 5- 2). Even if the ANN based on these two representations were trained with cases which have solution paths of different lengths it would not be very beneficial in an "hybrid" solution because there would be no direct information relating to the quality of the solution path provided by the ANN, i.e. how much longer a path is compared with the minimum distance. Neither is it possible to indicate to the ANN what "level" of quality can be tolerated for each vehicle, i.e. how much longer than the minimum can be allowed. The representations discussed next are based on Representation 1 and 2 previously presented and modified to account for its limitations when different quality solutions are used.

The philosophy used here was that training a network only with optimum solutions may not be as effective as training it with both optimal and suboptimal solutions, provided that the suboptimal solutions are identified with a level of quality. This approach provides potentially far more training information (there are far more larger routes to travel between any two nodes) and takes advantage of the feature that BP ANN's can be used to distinguish between solutions of different quality.

The first alternative was to assume that in a multiple vehicle situation the external system which uses the ANN could decide that for each vehicle only the best solutions must be considered; or alternatively that any of a limited number of the best, or closest to best possible solutions are acceptable. This analysis can be applied either in the complete path specification or in the step-by-step approach, as indicated respectively in the Figure 5- 8 and Figure 5- 9. The representations are exactly the same as

in representations 1 and 2 respectively except for the inclusion of more elements in the Input of each ANN to account for an explicit indication of the acceptability quality (i.e. length) of the paths.

Of the different possible codifications for this information relating to the new elements of the ANN INPUT, one possibility is to add only one binary element. When its value is set to "1" it indicates to the ANN that we are only interested in one of the best solutions. When its value is set to "0" it indicates to the ANN that any possible solution, or any of a number (i.e. 4) of the best possible solutions, can be accepted.

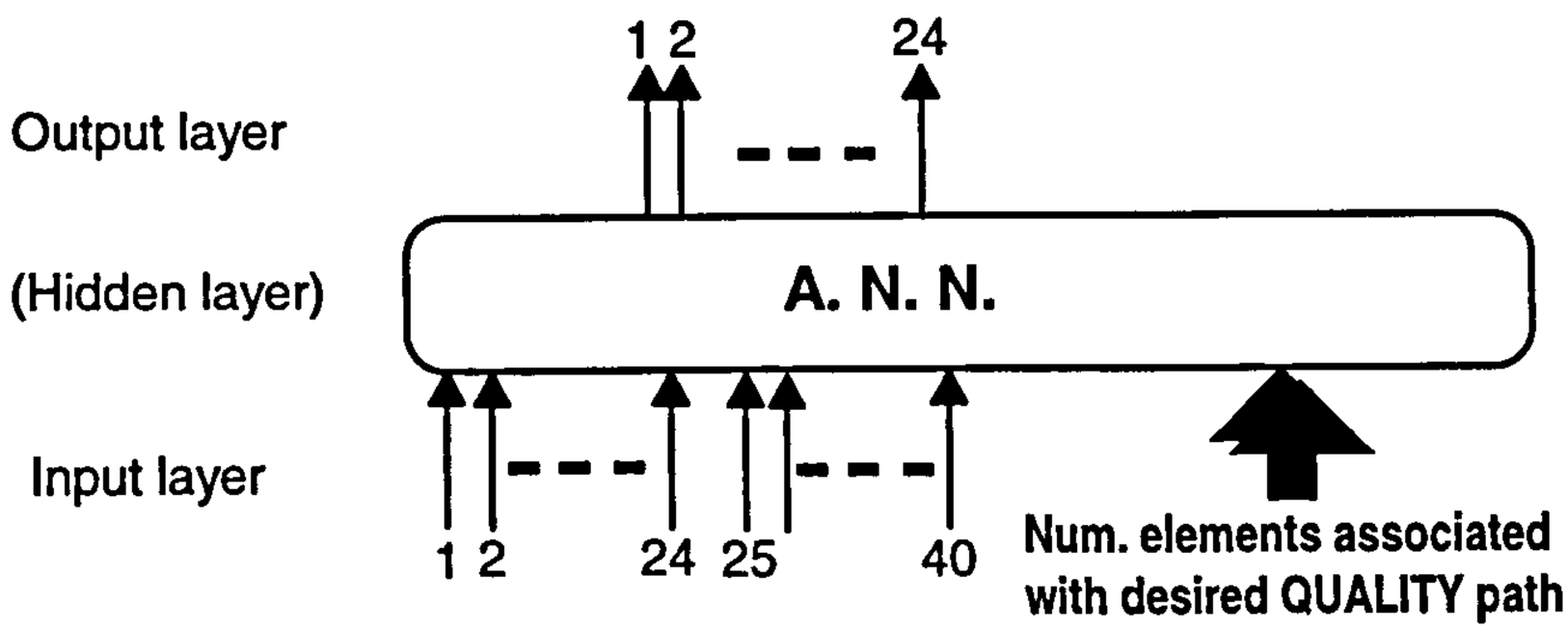


Figure 5- 8: Backpropagation ANN based on modification of Representation 1 to include Input elements associated with the quality allowed for the solution path.

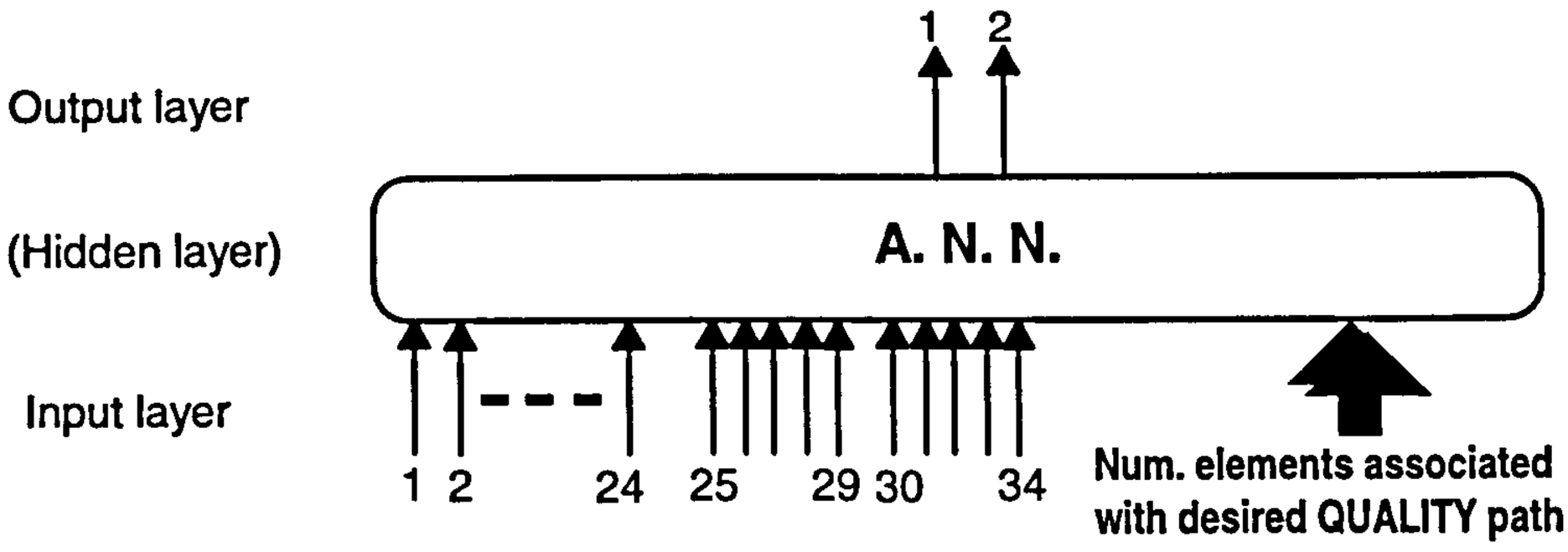


Figure 5- 9: Backpropagation ANN based on modification of Representation 2 to include Input elements associated with the quality allowed for the solution path.

While this approach has potential for providing alternative decisions when used as the ANN basis for an "hybrid" solution to the multi-vehicle case, it has not being pursued in favour of the approach which only uses optimum solutions. The reasons behind this decision can be related to an indication from the

experiments with ANNs which favoured a step-by-step approach and the implications of the use of extra information, i.e. alternative decisions leading to sub-optimal solution paths, on the external system of the "hybrid" approach. Basically it is possible to think of two different schemes. One which uses an ANN which was developed to provide movement decisions which lead to a shortest path, and an external system which analyses that decision when made for all vehicles involved, decides whether they can be implemented and solves possible conflicts rapidly using simple heuristic rules. Alternatively we can use an ANN which is able of providing alternative movement decisions, i.e. along the optimum or sub-optimum paths. In this case the external system would have more alternatives to be considered and evaluated when applied to all vehicles involved, with a consequent increase in processing required. In a step-by-step approach the conditions of the system change at each step and therefore the current optimum solution may change in the next step. When considered over all steps this will eventually account for the impossibility of completing a path along the optimum solution and using instead a sub-optimal solution path. The first scheme was therefore selected and the approach which is based on using different quality solutions was considered no further.

5.3 Program for generation of data

Having established a specific physical layout and a methodology to map the single vehicle case onto a backpropagation type ANN, the next step was to generate data for training and testing the ANN. For this purpose it was necessary to devise a means of finding optimal and suboptimal solutions to many vehicle routing problems in many different layouts. The only difference from the task that the fully trained neural network had to deal with was that the time taken to generate solutions was not critical. Purely manual means were however far too slow so it was decided to model a variety of layouts and routing problems using a general purpose simulation package and solutions were found laboriously, but reliably, using an exhaustive search algorithm. A simulation package called WITNESS [Istel Ltd, Redditch, England] was used because it had a graphical user interface that allowed the validity of the solutions to be easily confirmed and it was relatively quick to learn. Since it was PC based, the very high number of computations needed for the search algorithm made it slow to generate solutions, but it could be arranged to run automatically. The main features of the program implemented in WITNESS were that it:

(1) allowed the specification of the two nodes to be linked

(2) allowed the state of the tracks within the physical layout to be varied

(3) was capable of finding all possible paths linking the two nodes.

The information generated must then be presented in the formats defined for the input and output elements of the ANN. To account for all proposed representations of the single vehicle case in a backpropagation ANN, the information included in each case consists of:

(1) the identification of two location nodes in the physical layout

(2) the tracks which were set as unavailable

(3) the tracks used in the path linking the two location nodes

(4) the corresponding path length.

The main parts of the program are described next with the aid of flowcharts (Figure 5- 10 to Figure 5- 12), and considering the following conditions:

- a definition of a physical layout as defined in Figure 5- 1
- repeated nodes, or tracks, are not allowed in a solution path.

The program starts with a specification of the number of cases to be generated by asking the user to specify the number of different layout states, i.e. number of selections of tracks available/not available, and the number of pairs of origin and destination nodes generated and tested in each layout. Then it is possible to specify manually, i.e. by the user, or randomly using a pseudo-random generator function, the selections for each case, see also Figure 5- 11:

- the number of tracks to be set as unavailable
- each of the track numbers to be set as unavailable
- the origin node number
- and the destination node number.

Based on the specifications for each case the program then calculates the number of possible ways out of each node, i.e. how many tracks at each node are available, and starts the search algorithm to find all possible solution paths. This exhaustive search algorithm, as illustrated in Figure 5- 12, consists of starting a path at the origin node and tries to include consecutively adjacent nodes in the path by testing a possible way out of the current node until it either gets to the destination node, a node already included on the current path, or to a node with no other possible way out. When a way out of a node is used the total number of ways out of that node is decreased by one. When it gets to the destination node a path is completed, and a new path is started which includes all the nodes of the path just completed except the destination node. The search continuous at that point in the new path by testing the number of ways still available out of the current node. The nodes which are released from the current path must have the number of ways out reset to the initial value so they can be included and tested when other paths may approach these nodes from other directions. The process ends when all ways out of the last node in the completed path have been tested and the last node has become the origin node. This algorithm can be included on what are usually [Ref. 82, Ref. 83] designed by "depth-first-search" exhaustive search algorithms associated with graphs in graphs theory, but in this application it involves a much higher number of calculations because every node as to be checked several times [Ref. 82, Ref. 83].

This program was used to generate a large number of paths to be used in the design of the ANNs and a more complete characterization of the cases generated is presented in Chapter 6 next. An example of the data generated using this program is presented in Appendix A.

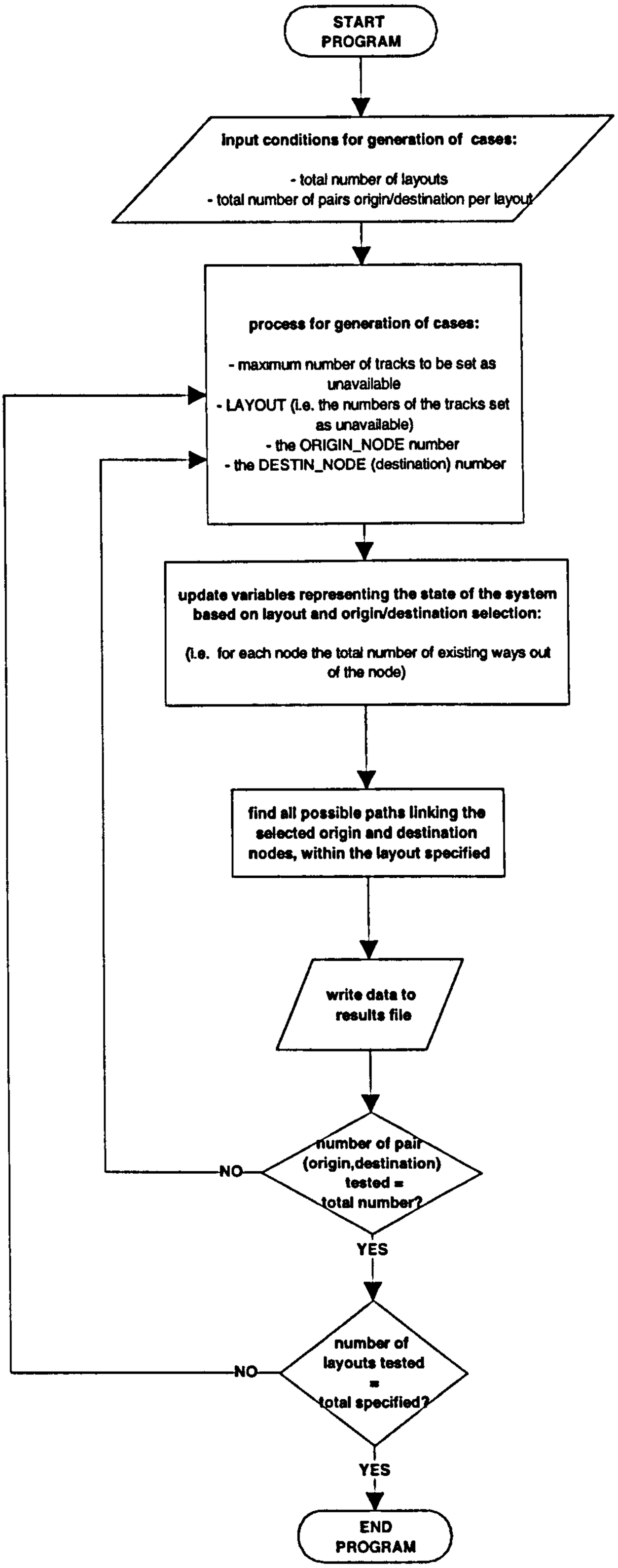


Figure 5- 10: Program used to generate training and testing data: main phases.

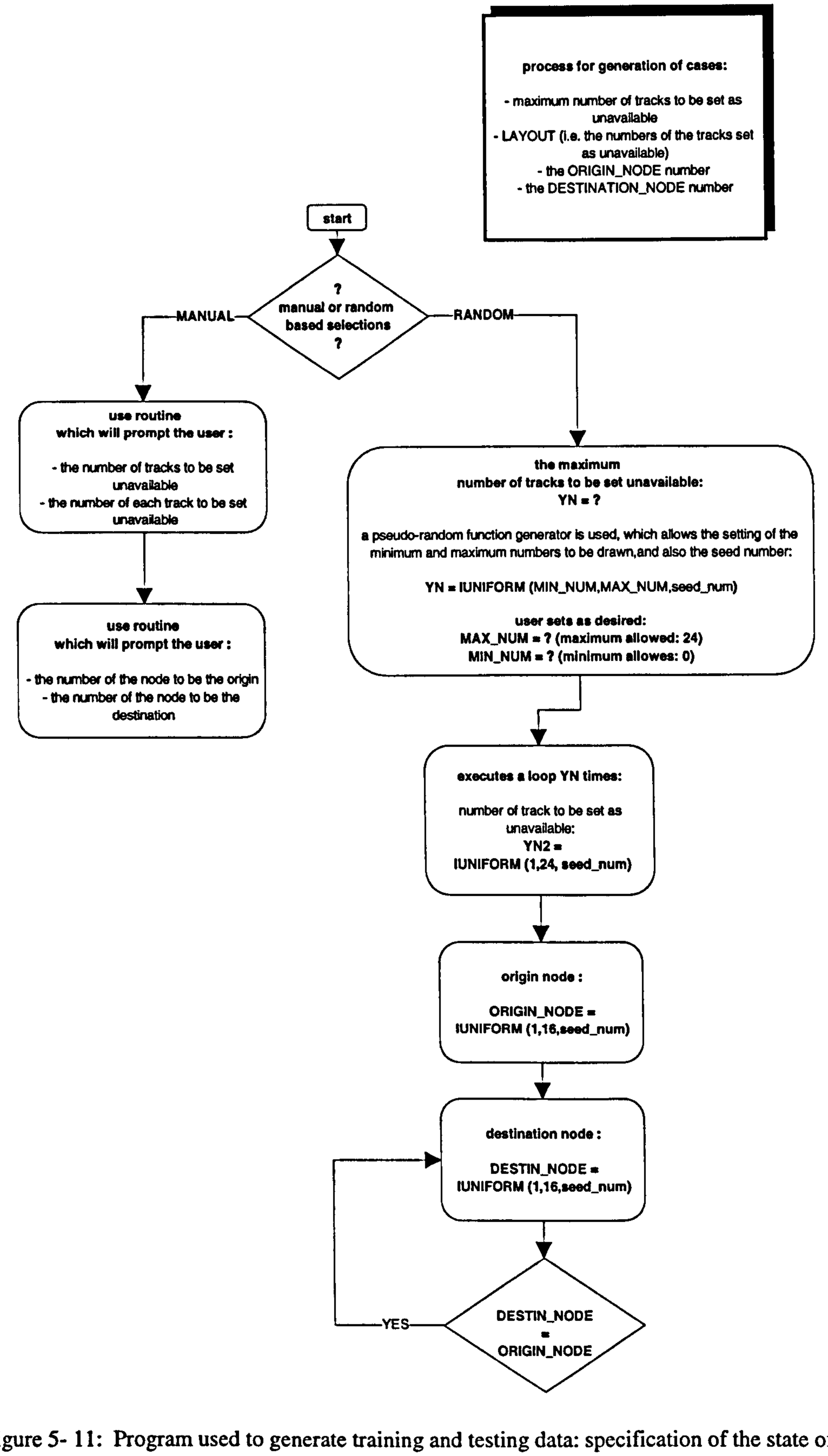


Figure 5- 11: Program used to generate training and testing data: specification of the state of the tracks in the layout, origin and destination nodes.

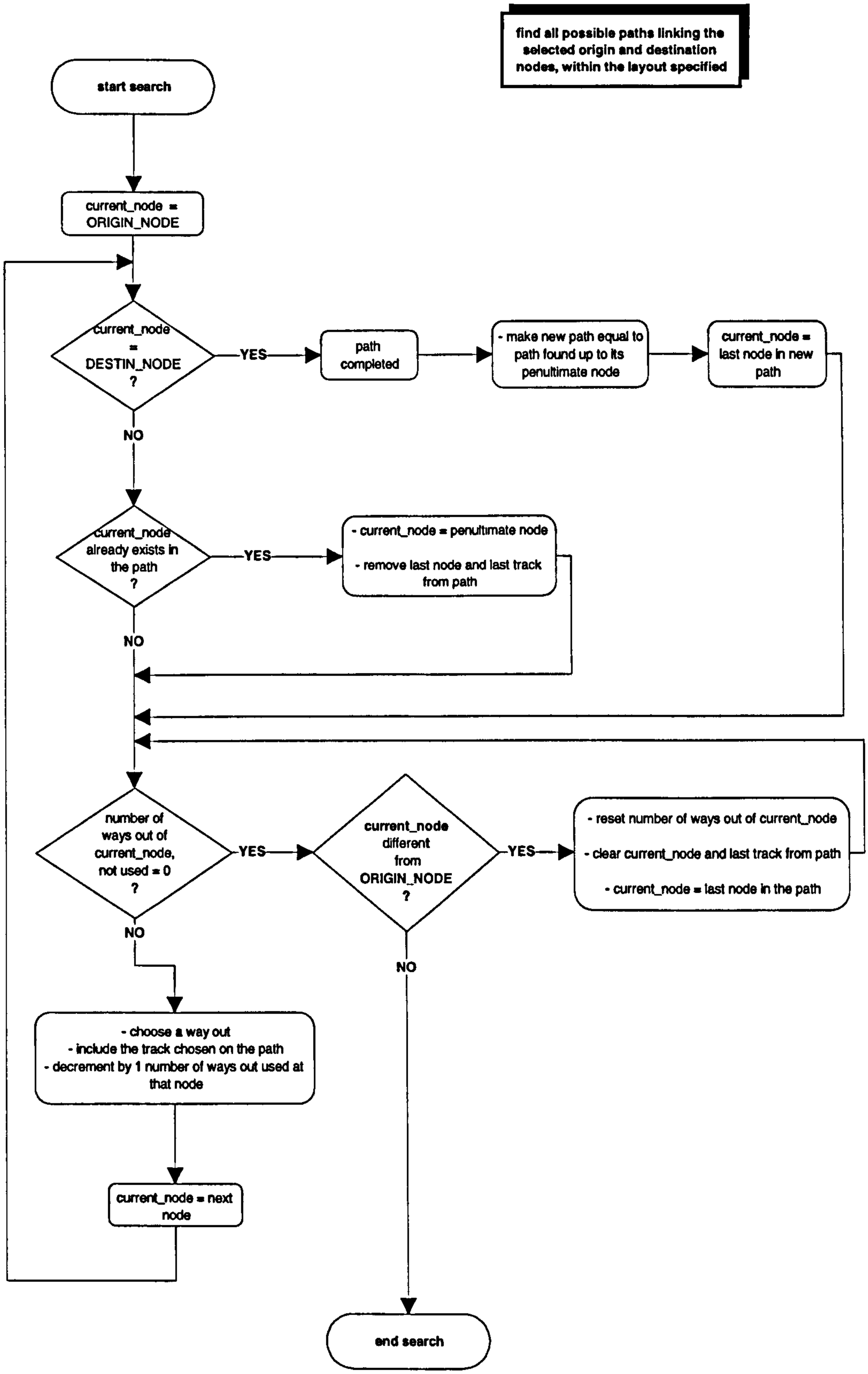


Figure 5- 12: Program used to generate training and testing data: algorithm to search for all possible paths.

Chapter 6

Characterization and Validation of Training and Testing Data

The data used for training and testing is the basis for developing a BP ANN and therefore it must be characterized before describing the experiments done to select the design parameters of the BP ANN. The separation of the data into training and testing sets, is required in order to evaluate the performance of the BP ANN when exposed to new data (i.e. testing set), that is data which has not been used during learning of the BP ANN. This chapter concentrates on the various aspects that can be related to a characterization of the data obtained from the program for generation of data described in the previous chapter, Chapter 5.

BP ANN data, whether used for training or testing, consists of a number of cases each consisting of a problem instance and its corresponding solution. These sets of data can be considered as representing a limited number of examples and the desired solution of all the possible occurrences in the problem and from which the BP ANN will learn how to provide solutions to other instances of the problem. It is therefore important to characterize the global set of cases possible in relation to the set of cases used for training or testing the BP ANN. One aspect to look at is the size of the data sets used but it is also important to evaluate how the individual cases represent different possible classes or types of information contained in the global set of all possible cases.

A case can be characterized, when it is associated with the formulation of the problem and also when it is associated with the representation of the problem for use with an ANN which has specific requirements for input and output elements. First we will present a case derived from the formulation of the problem of the single vehicle case with aspects of the generated data which are relevant to all

representations and which can help in a perception of the size of the problem. The particular aspects of the data when it has been converted into a format suitable for an ANN based on the representations considered are presented subsequently.

6.1 General aspects of the generated data

The problem of the single vehicle case consists of routing a vehicle from an origin to a destination when some tracks are set unavailable (chapter 5, section 5.1). There are two aspects to look at when trying to characterize the total number of different cases that can occur in that problem: one is the total number of different problem instances that can be created when all possible combinations of origin/destination nodes and tracks states are specified, the other is the set of all possible different paths which contain any solution for any possible specification of the problem. Both of these sets will depend on the physical layout and the estimates presented below correspond to the physical layout of Figure 5- 1 with the following considerations:

- there are 16 nodes available each time, from which a pair of (origin, destination) nodes can be chosen, with the condition that the origin and destination nodes cannot be the same,

- the 5 nodes at junctions are not considered as possible origin or destination nodes in the general formulation of the problem,

- there are 24 tracks which can be set as unavailable, but only a maximum of 4 tracks can be set as unavailable at the same time,

- the solution paths cannot contain repeated nodes or tracks.

On the basis of these assumptions the total number of cases possible, in terms of different specifications of origin/destination pairs and different track states, can be calculated based on the number of combinations possible for blocked tracks and pair of nodes chosen, as follows:

$$\left(C_0^{24} + C_1^{24} + C_2^{24} + C_3^{24} + C_4^{24}\right) * C_2^{16} = 1554120$$

The calculation of the total number of different solutions (i.e. possible paths) involves a different analysis because the selection of tracks to form a path cannot simply be based on different combinations of the tracks available. However it should be noted that the total number of different paths possible will be a maximum when all tracks are available. A calculation of all possible paths with lengths ranging from one to eighteen tracks (maximum possible), assuming that all tracks are available on the physical layout is presented in Table 6- 2. Comparing the values in this table with the one calculated above we can see that the case of our single vehicle problem, as formulated, involves mapping a much larger set of cases (1554120), into a smaller, by several orders of magnitude, set of solution paths (1068). Also it should be noted that increasing the number of tracks set as unavailable at a time, further increases the difference in size between the two sets by enlarging the number of different combinations of layout states while reducing the number of different solutions paths that are possible.

Table 6- 1: Total number of different paths with lengths 1 to 18 tracks, considering the 16 origin and destination nodes and all 24 tracks available.

Num. of Tracks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Num. of Paths	8	22	16	44	28	68	72	100	96	76	96	138	56	44	48	92	48	16
Total Num. of Paths = 1068																		

The data sets used for training and testing were generated using the program for the generation of data described in section 5.3. Various sets of data were collected by setting the options for pseudo-random generation of the origin/destination pairs of nodes and the tracks unavailable. Taking into consideration the approach chosen (Chapter 5, section 5.1) that associates vehicles to unavailable tracks, the number of tracks to be set as unavailable was generated between 1 and 4. In our envisaged solution the four tracks set as unavailable would represent information that vehicles using the ANN would use relative to the intentions of movement of four other vehicles. Allowing five vehicles in the system, a maximum of four tracks set as unavailable seemed reasonable for the layout considered (Figure 6- 1). Furthermore it also provides the opportunity for us to evaluate the performance of the approach when a larger number of vehicles are involved, and experiments with up to 10 vehicles were performed using the ANN developed with this data. A case in the data generated can be considered to

be the pair of nodes to be linked when a number of tracks are set as unavailable. To each case corresponds a number of paths that link the two nodes. There maybe none, one, or more paths of the same or different lengths.

Three sets were selected from the data collected: a training data set, a testing_1 data set, and a testing_2 data set. The testing data sets were selected to contain different cases and randomly comprised about 10% of the number of cases in the training set. Two sets of testing data were selected because the specification of a problem instance consists of the specification of a layout state and a pair of nodes origin/destination. Data set testing_1, contains cases where the layout state is one of the already used cases in the training data set but with a different pair of nodes origin/destination. Data set testing_2, contains cases which are different from the ones in the training set only in terms of layout state.

It should be noted that when a distinction is made as to which is the origin node and which is the destination node, the number of cases generated can be duplicated by swapping the nodes origin/destination and considering the respective "forward" and "reverse" solution paths. This was in fact the case for Representation 2 and it was used also in order to evaluate the possibility of relating the direction of the paths with the performance of the ANN for that representation. The number of cases used in the training and testing data is summarized in Table 6- 3. which contains also values corresponding to the bi-direction cases. It should also be noted that because the program for generating the data does not check for repeated cases, the data sets generated contained some repeated cases. However these were less than 5 % of the cases generated. The number of pairs of origin/destination generated for each layout state indicated in Table 6- 3 was also not always the same, because they were collected from different data sets. Appendix A contains drawings representing each of the different layout states used in training and testing.

Table 6- 2: Number of cases generated.

Data sets:	TOTAL	Training	Testing_1	Testing_2
NUMBER OF CASES				
- no distinction of origin/destination nodes:	1137	985	111	41
- doubled by swapping origin/destination nodes:	2274	1970	222	82
Number of layouts states:	85	74	70	11

As indicated previously there might be cases where a specific pair of origin and destination nodes in a specific layout state can be linked by more than one solution path and also there might be more than one optimum path in some cases. The characteristics described below were based on considering only one of the best possible solutions for each case, when more than one was possible, and is addressed later in section 7.3.1.

The objective was to relate the use of the individual elements defined in the physical layout with the origin/destination pairs of nodes, the layout states, and one of the optimum solution paths associated with these cases. Each Figure represents one of the characteristics chosen in relation to the data sets selected: training, testing_1 and testing_2; and considering the cases resulting from "forward" and "reverse" paths.

For the layout states and origin/destination pairs of nodes these characteristics are:

- the number of times each track was set as unavailable,
- the number of tracks set as unavailable at the same time,
- the number of times each node was used as an origin and as a destination node,
- and also a matrix with the number of times each pair of nodes origin/destination was used.

And for the solution paths:

- the lengths of the solution paths,
- the number of times each track was used in a solution path.

Next are presented graphical representations (Figure 6- 2 to Figure 6- 7) of these characteristics and their analysis. The number of each node and track used in the graphs corresponds to the ones represented in Figure 6- 1.

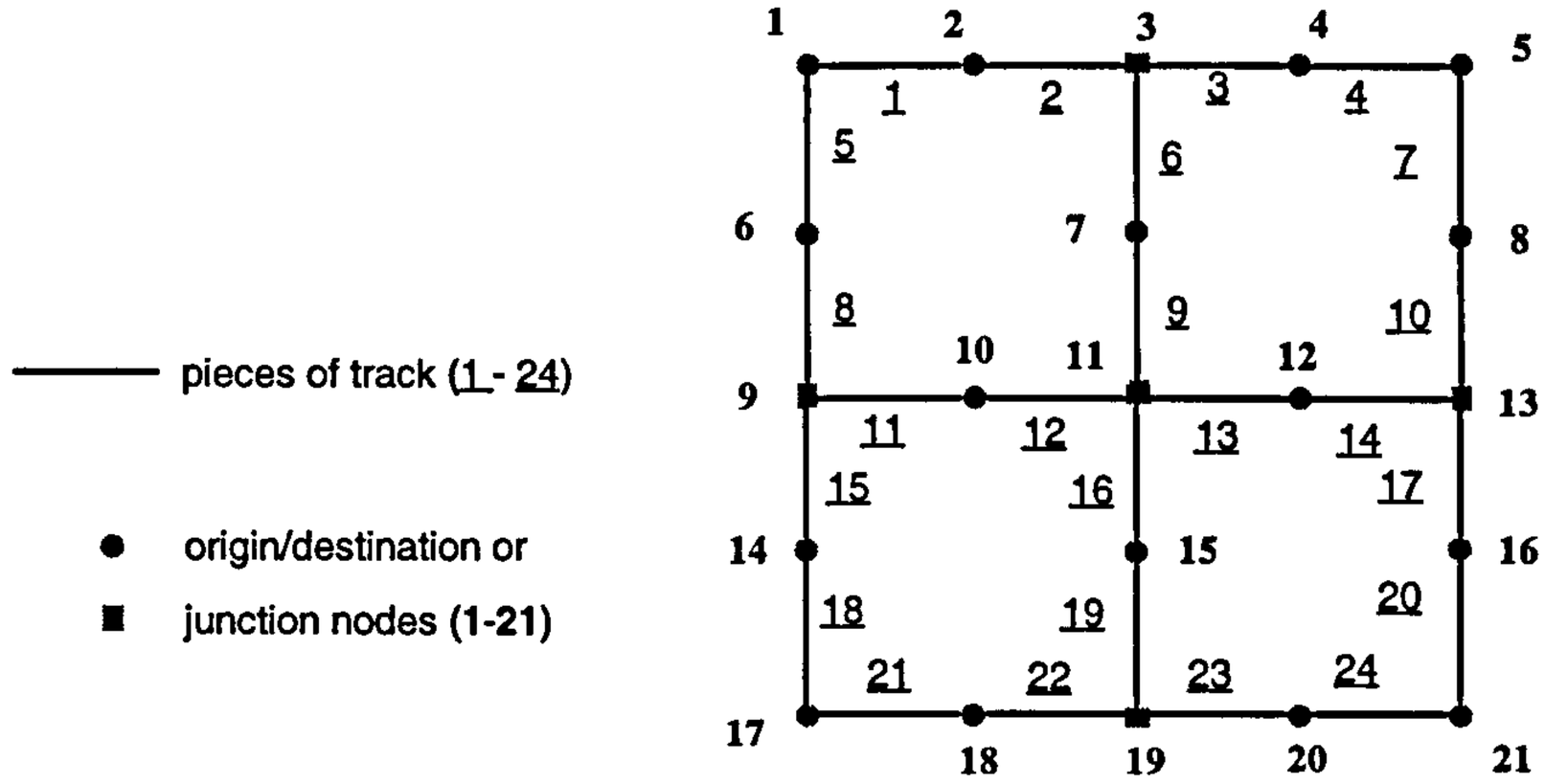


Figure 6- 1: Representation of the identification numbers for nodes and tracks in the physical layout.

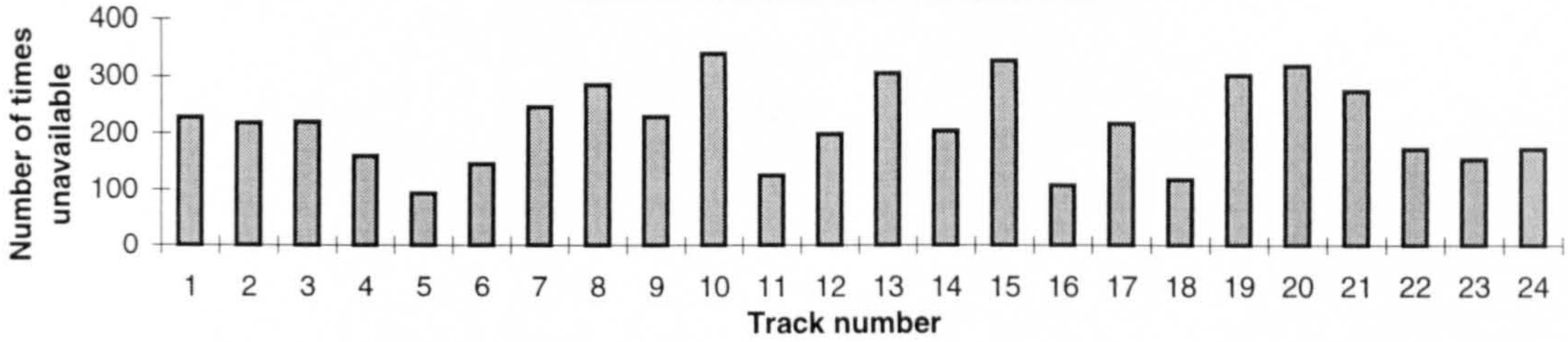
The graphs in Figure 6- 2 show distributions of the number of times each track was set unavailable in all the cases used for training, testing_1 and testing_2 data sets respectively. The scales used are not the same but they are set in relation to the relative size of the training and testing data sets. This is convenient to compare the evolution of the distributions, however for the values on each scale it must be noted that the maximum values on the scales of the graphs representing the testing cases are 10% of the ones on the graphs representing the cases used for training. The information represented on these graphs seems to confirm what could be expected by the generation of a number of layout states using pseudo-random numbers generation functions. The tracks were not equally used in either of the data sets, but it is plausible that this could be achieved with the generation of a higher number of cases. Nevertheless all tracks have been used extensively in relation to the maximum and minimum values obtained. The distribution of the testing_1 data set presents a similar distribution to the training data set. Which corresponds to what might be expected because it is based on cases using the same layout states. In relation to the testing_2 data set it shows considerable differences with some tracks never used, but this can be related to the limited number of layouts used (11) on this data set, and which are different from the layouts used on the other data sets.

Figure 6- 3 shows the other graphs used to characterize the layout states in terms of the number of tracks set unavailable at the same time. In the training data set the maximum number of cases consisted of cases which had 3 tracks unavailable, but overall it seems apparent that the number of tracks set unavailable is reasonably well spread between the minimum, 1, and the maximum of 4 tracks. As in the graphs of Figure 6- 3 and due to the same reasons, the graph associated with testing_1 data set is again very similar to the graph associated with the training data. However the cases in the testing_2 data set are clearly more with 4 tracks unavailable.

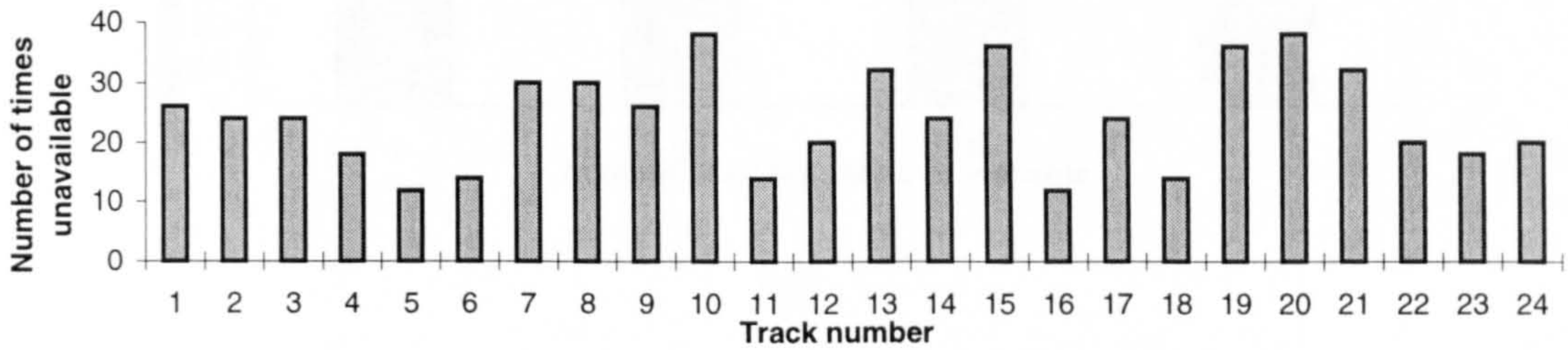
The graphs on Figure 6- 4 and the matrices on Figure 6- 5 show the other issue related to the specification of an instance of the problem, and which derives from the selection of pairs of origin and destination nodes. All three graphs show that all nodes have been used in a pair of origin and destination nodes, except the nodes at junctions which were specifically excluded. Also the variations in the number of times each node was used seem plausible in terms of corresponding to a limited number of cases that were generated using pseudo-random numbers. The differences now observed when comparing the graphs associated with the training data set and the one associated with the testing_1 data set were expected because what is being analysed now is what differentiates these testing cases from the cases used for training.

The graphs of Figure 6- 6 and Figure 6- 7 characterise the paths that represent solutions to the problem instances or cases generated. It only considers one of the solutions if more than one was available. The distributions related to the length of the solution paths (Figure 6- 6) present a comparable evolution around the same, or similar values. That is a maximum, or close to maximum number of cases with length 4, most of the cases within lengths 1 to 8 and only a reduced number of cases with lengths longer than 8, or impossible paths (i.e. length equal 0). It is interesting to note that paths with length 4 represent half of the maximum optimum distance (i.e. length 8) between two nodes located at opposite extremes in the physical layout. It is also interesting to note that in the training case (Figure 6- 6 (a)), paths with length 3 and 5 are fewer than with length 2 and 6, which is in line with the estimates made for the number of different paths possible with those lengths in a layout where all tracks available (Table 6- 1). However the number of cases selected for testing_1 data set include more solution paths with lengths 3 and 5 than 2 and 6.

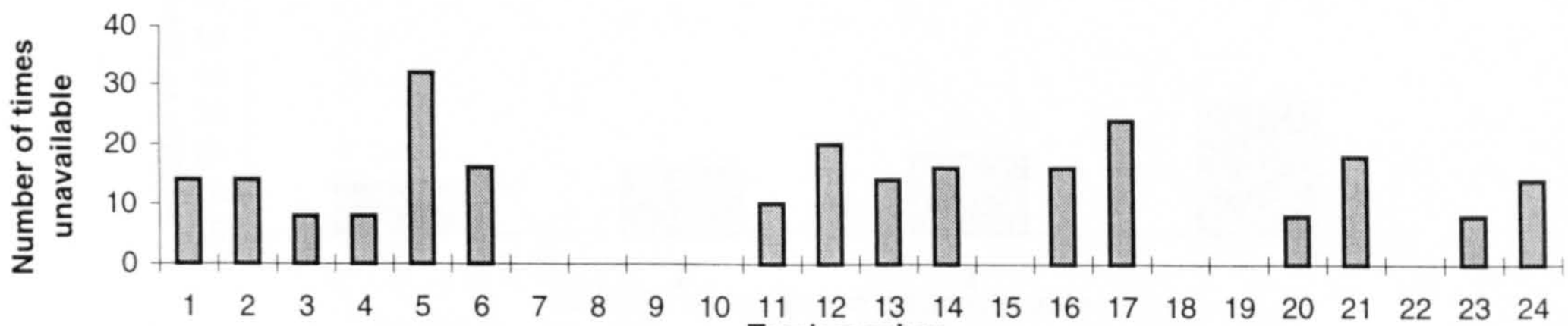
In relation to Figure 6- 7 where the graphs represent the number of times each track was used as part of a solution path, all three graphs show distributions well spread over all tracks particularly in the training and testing_1 data sets. The distribution associated with the testing_2 data set presents somewhat higher variations, which might be related to a reduced number of cases being considered.



(a)

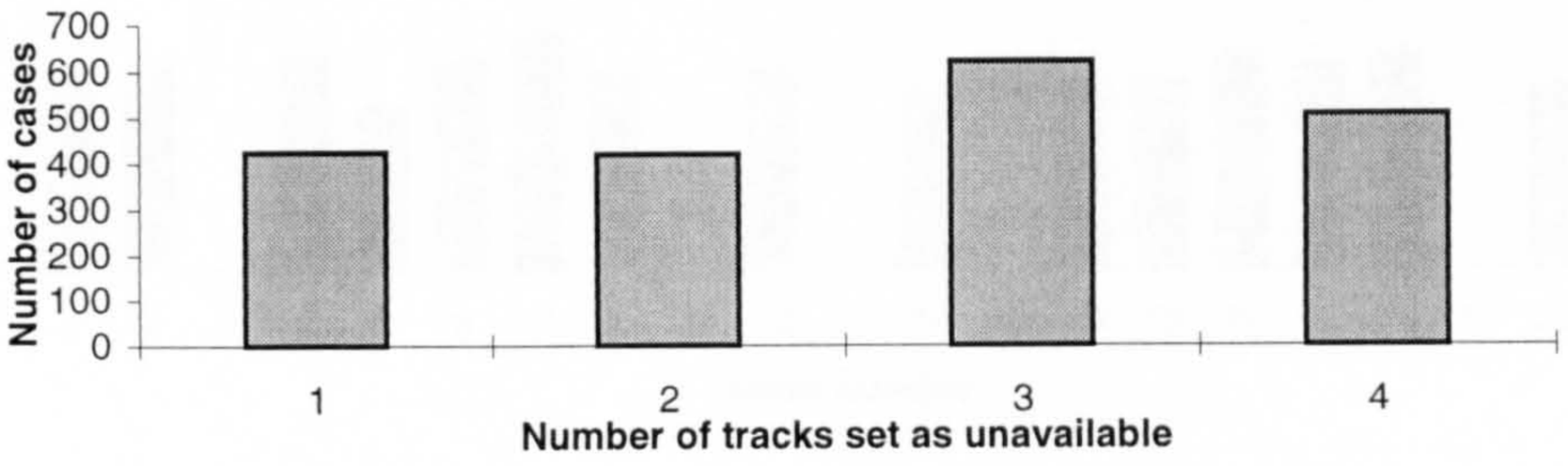


(b)

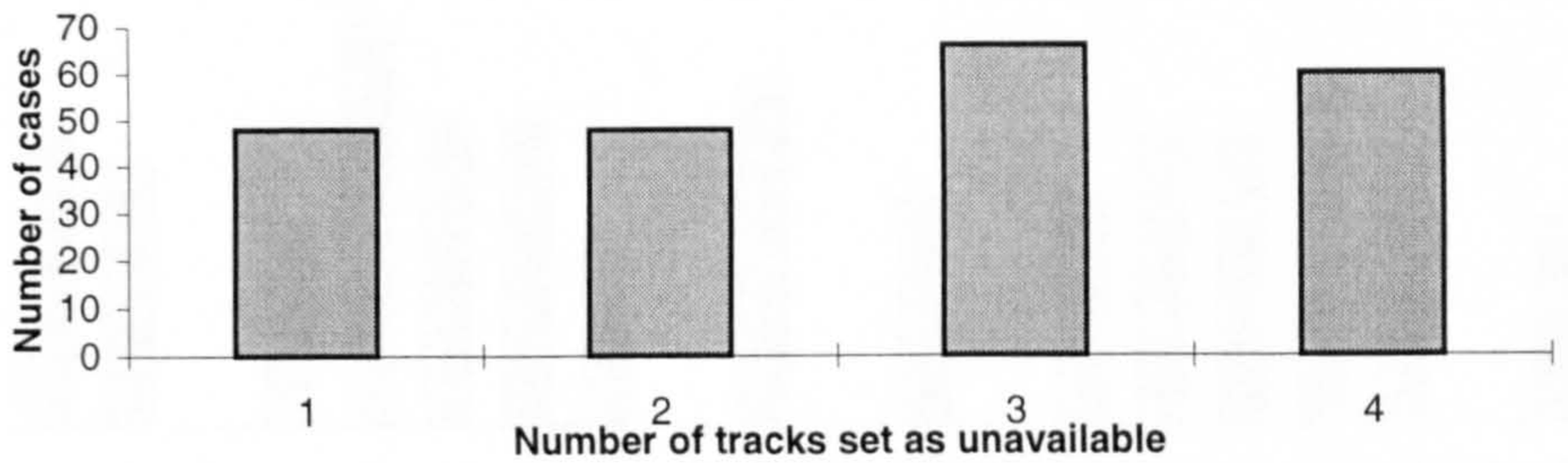


(c)

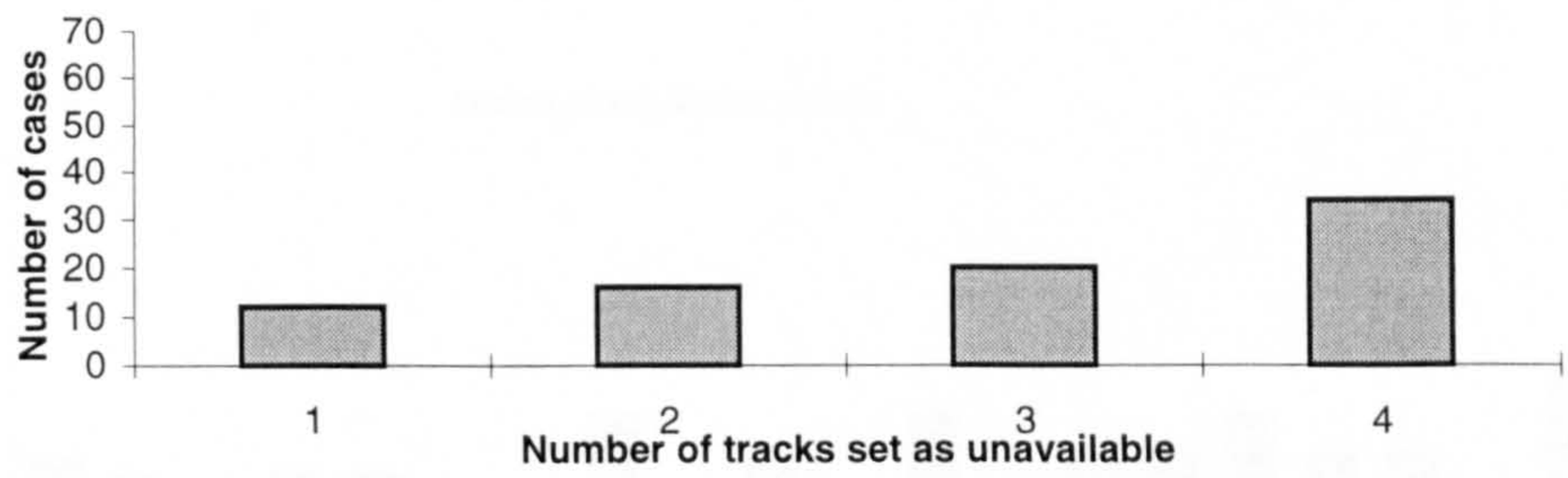
Figure 6- 2: Number of times each track was set as unavailable(forward and reverse cases), (a)training, (b) testing_1, (c) testing_2.



(a)

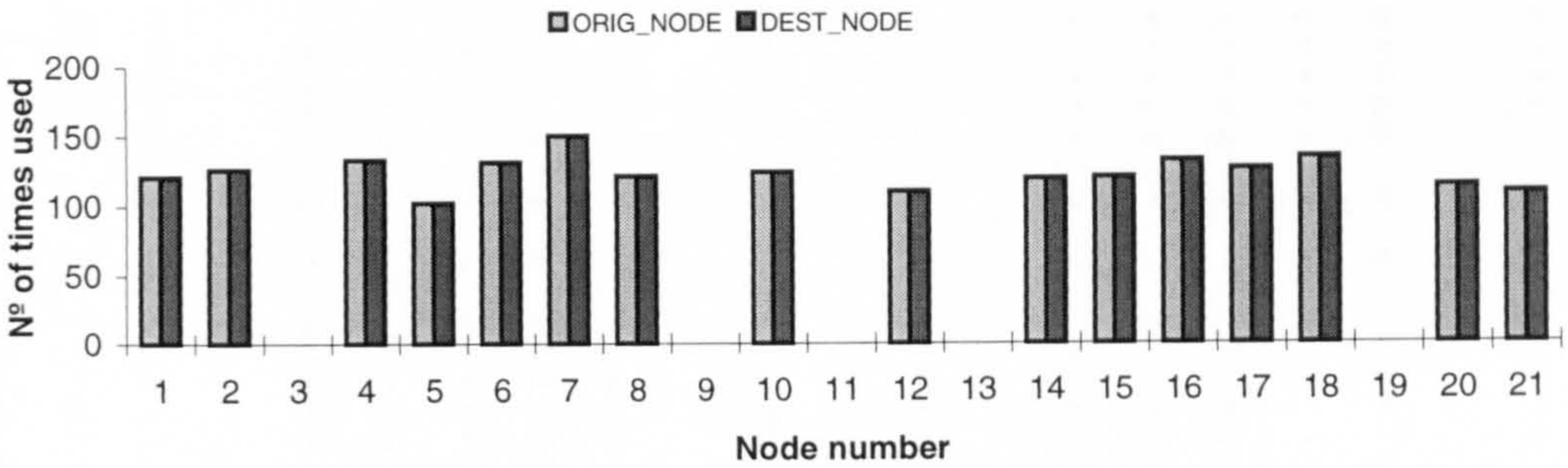


(b)

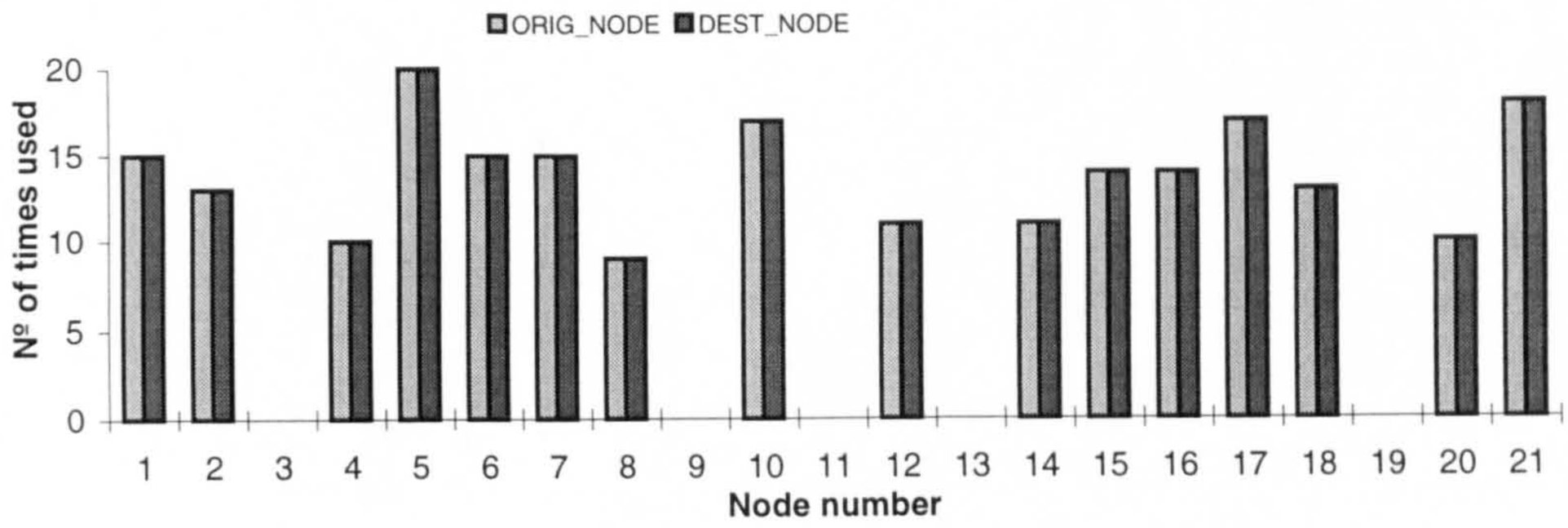


(c)

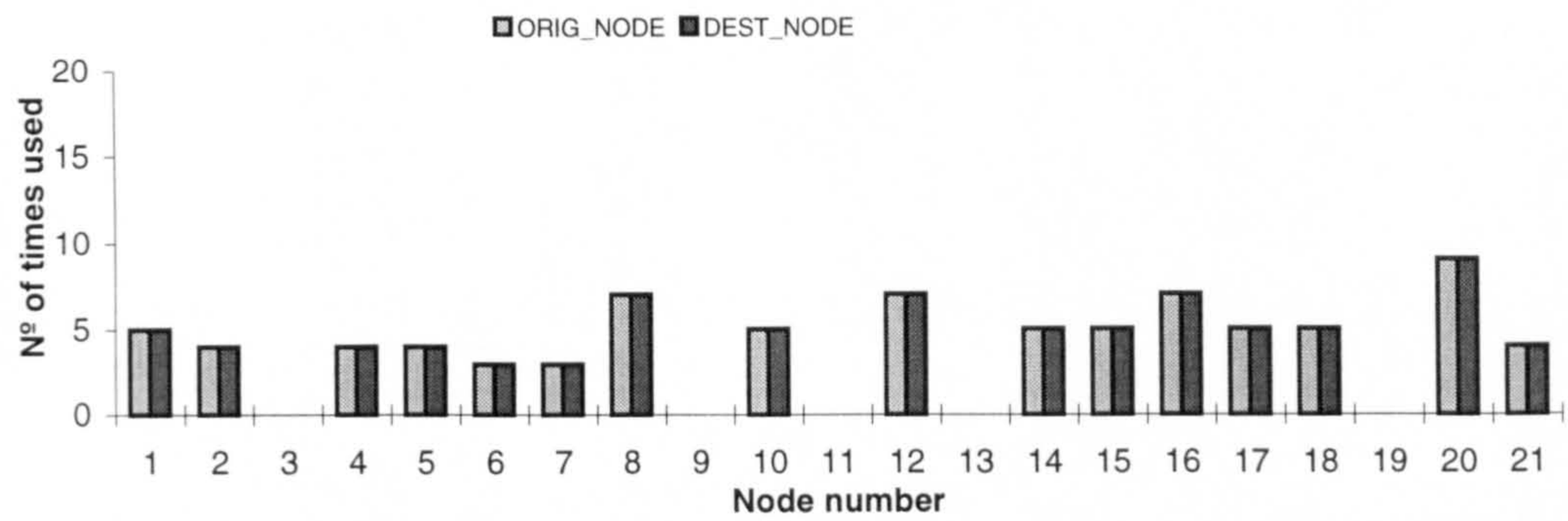
Figure 6- 3: Number of tracks set as unavailable (forward and reverse cases), (a) training, (b)testing_1, (c) testing_2.



(a)



(b)



(c)

Figure 6- 4: Number of times each node was used (forward and reverse cases), (a) training, (b)testing_1, (c) testing_2.

	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1		10		10	6	6	4	5		10		9		9	9	13	8	7		7	8
N_OR2	10			9	14	7	11	8		7		1		8	10	8	9	9		7	8
N_OR3																					
N_OR4	10	9			6	10	7	7		12		7		10	5	10	19	8		9	4
N_OR5	6	14		6		13	10	5		6		5		4	4	8	2	3		6	11
N_OR6	6	7		10	13		11	9		9		11		8	6	8	8	7		9	9
N_OR7	4	11		7	10	11		8		8		9		17	14	12	12	10		8	9
N_OR8	6	8		7	6	9	8			4		6		9	11	10	6	12		11	10
N_OR9																					
N_OR10	10	7		12	6	9	8	4				9		10	8	11	9	12		7	3
N_OR11																					
N_OR12	9	1		7	5	11	9	6		9				9	8	5	8	11		9	3
N_OR13																					
N_OR14	9	8		10	4	8	17	9		10		9			10	6	9	4		3	3
N_OR15	9	10		5	4	6	14	11		8		8		10		9	5	7		6	8
N_OR16	13	8		10	8	8	12	10		11		5		6	9		5	12		7	8
N_OR17	8	9		19	2	8	12	6		9		8		9	5	5		9		9	8
N_OR18	7	9		8	3	7	10	12		12		11		4	7	12	9			11	12
N_OR19																					
N_OR20	7	7		9	6	9	8	11		7		9		3	6	7	9	11			4
N_OR21	8	8		4	11	9	9	10		3		3		3	8	8	8	12		4	

(a)

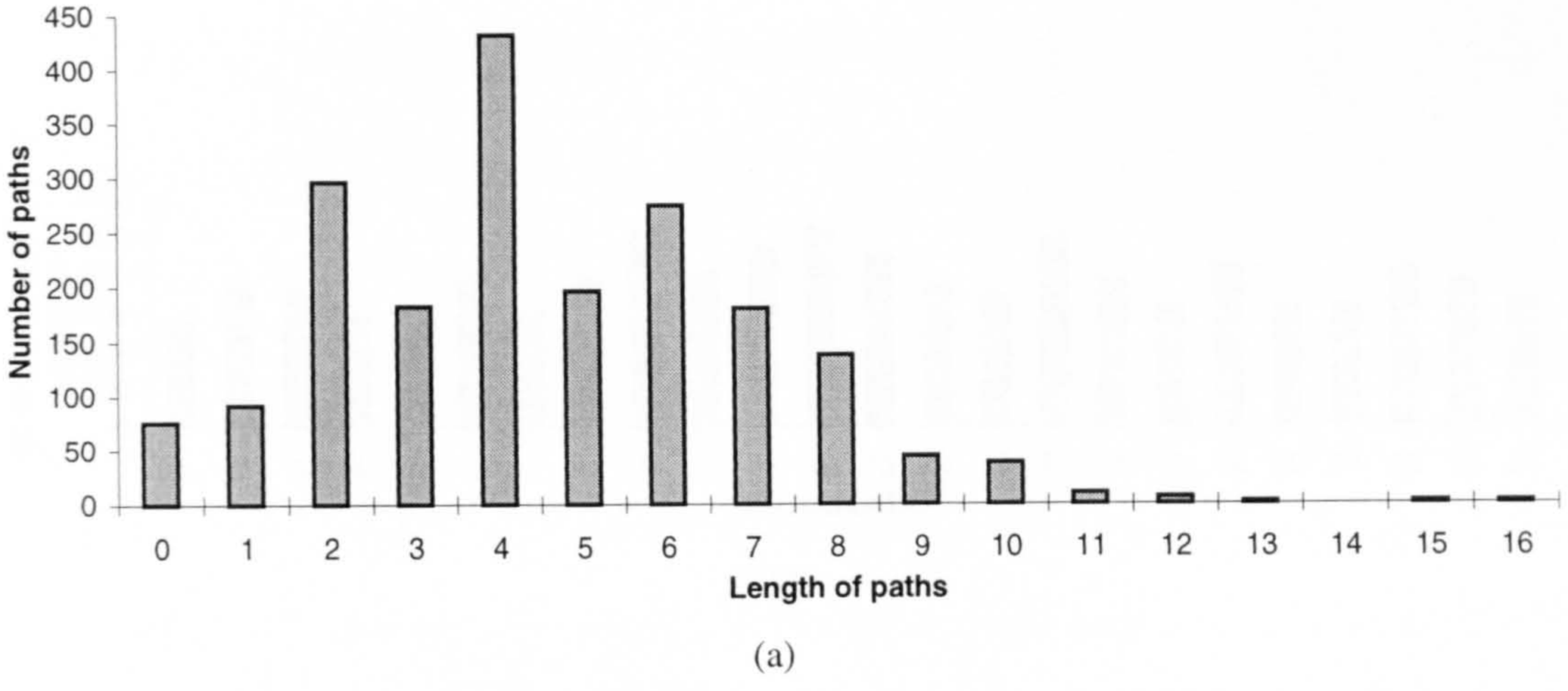
	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1		1		1		1	1	1		1		1		1	1		2	4			
N_OR2	1			1	2	4				1		2								1	1
N_OR3																					
N_OR4	1	1			2		1			1				1	2	1					
N_OR5		2		2		1						3		1	2	2	2	2		1	2
N_OR6	1	4			1		1	1							2	1	3			1	
N_OR7	1			1		1				4				1	3	1	1	1		1	
N_OR8	1					1				1				1	1		1			1	2
N_OR9																					
N_OR10	1	1		1			4	1				1		1	1		2	2			2
N_OR11																					
N_OR12	1	2			3					1							1	1			2
N_OR13																					
N_OR14	1			1	1		1	1		1					1	2					2
N_OR15	1			2	2	2	3	1		1				1							1
N_OR16				1	2	1	1							2			2	1		1	3
N_OR17	2				2	3	1	1		2		1				2		1		2	
N_OR18	4				2		1			2		1				1	1				1
N_OR19																					
N_OR20		1			1	1	1	1								1	2				2
N_OR21		1			2			2		2		2		2	1	3		1		2	

(b)

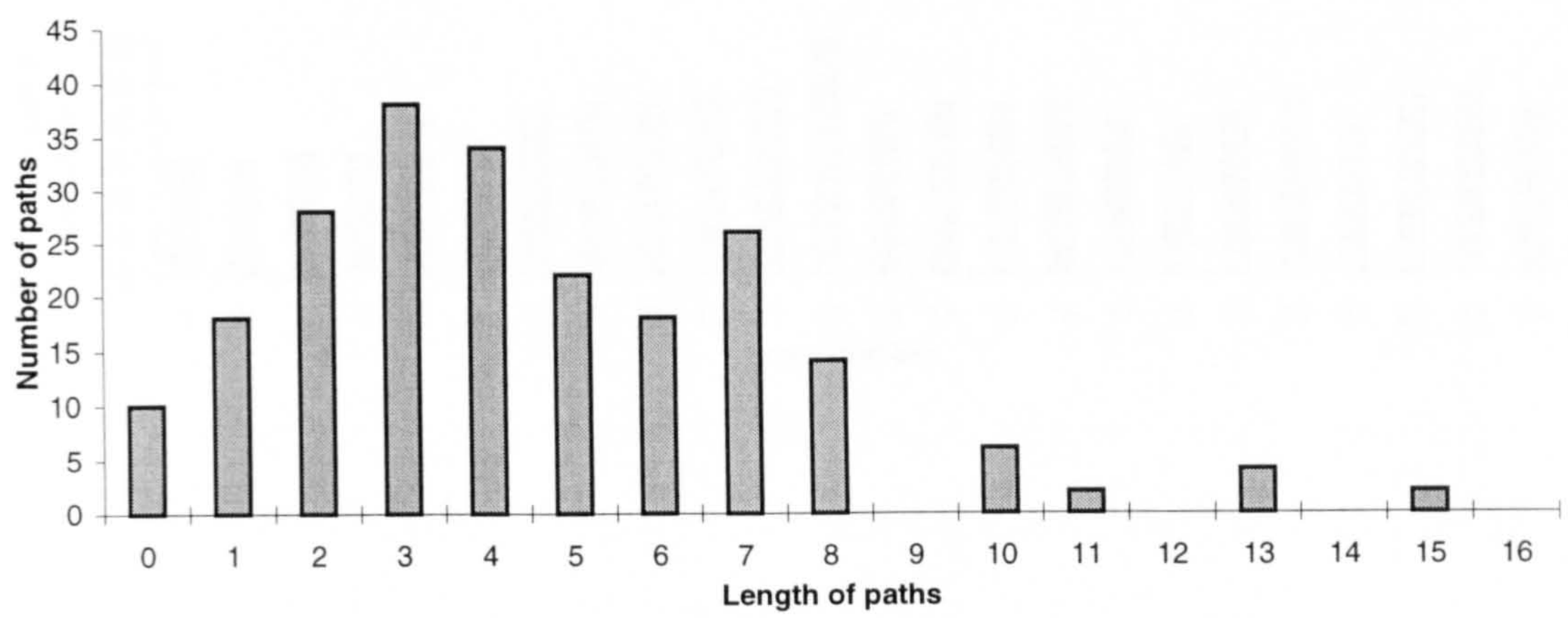
	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1						1	1							1						2	
N_OR2							1	1								1					1
N_OR3																					
N_OR4										1		2			1						
N_OR5						1				2							1				
N_OR6	1				1											1					
N_OR7	1	1																1			
N_OR8		1								1		1			1	1		1			1
N_OR9																					
N_OR10				1	2			1						1							
N_OR11																					
N_OR12				2				1								1		1		2	
N_OR13																					
N_OR14	1									1					1		2				
N_OR15				1				1						1		1				1	
N_OR16		1				1		1				1			1		1			1	
N_OR17					1									2		1				1	
N_OR18							1	1				1								1	1
N_OR19																					
N_OR20	2											2			1	1	1	1			1
N_OR21		1						1										1		1	

(c)

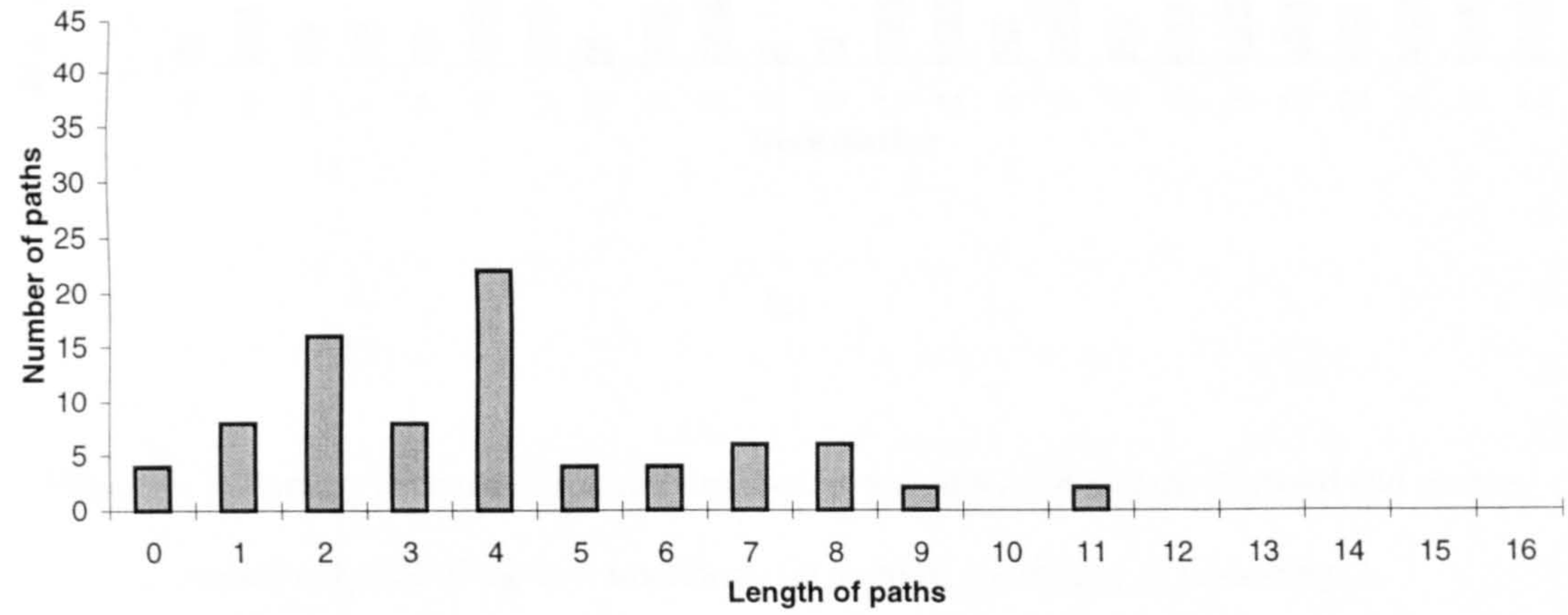
Figure 6- 5: Combinations of pairs origin/destination nodes used (forward and reverse cases), (a)training, (b) testing_1, (c) testing_2.



(a)

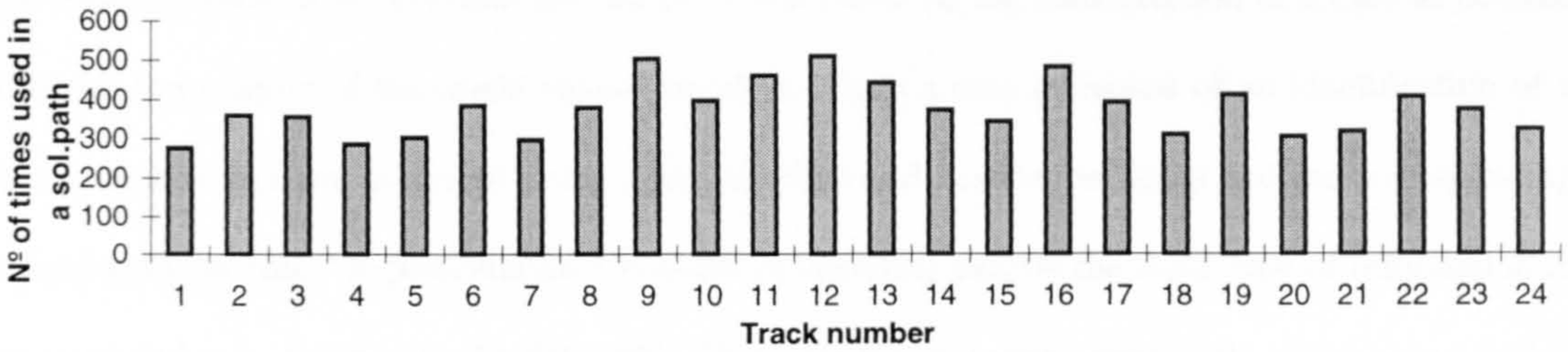


(b)

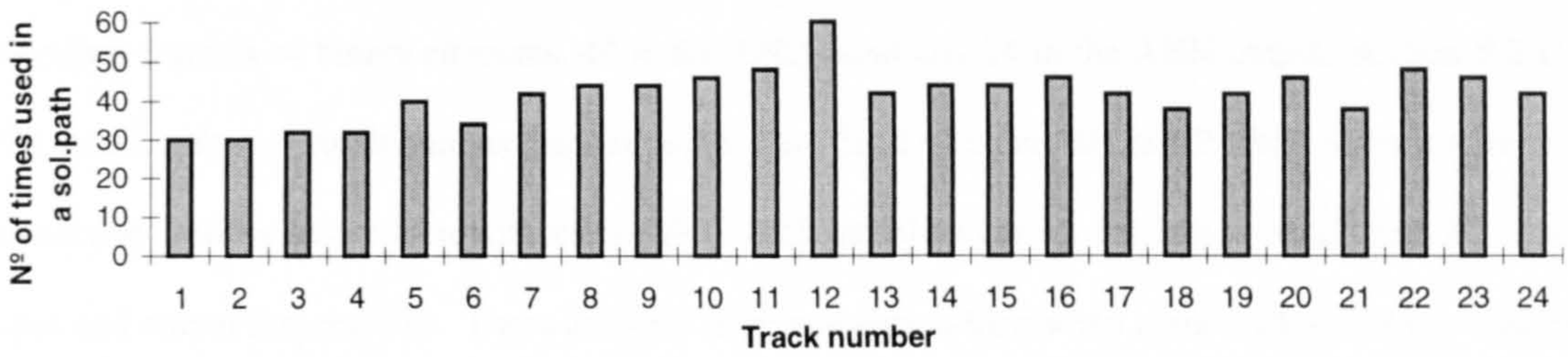


(c)

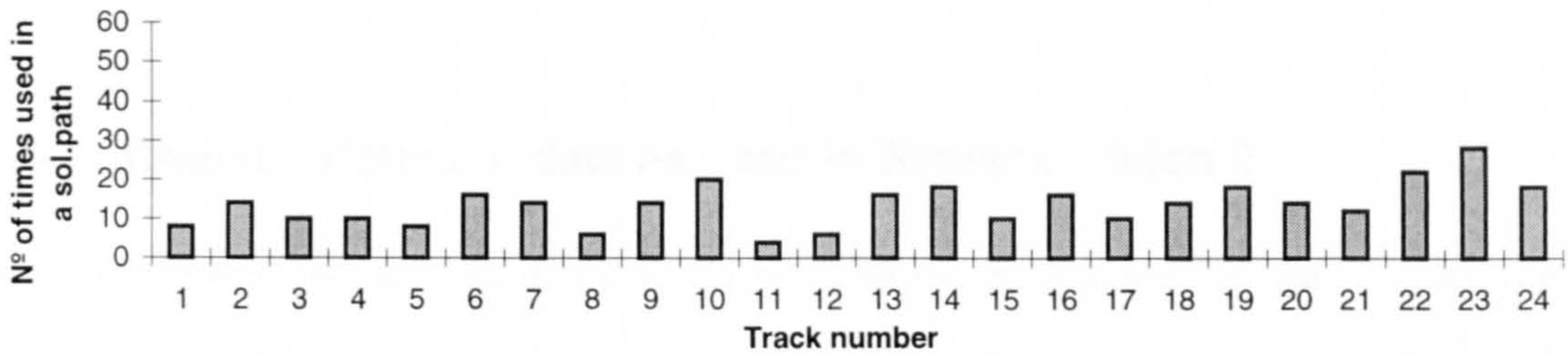
Figure 6- 6: Number of paths with different lengths (length 0: impossible path; forward and reverse cases; only one of the best solutions); (a) training, (b) testing_1, (c) testing_2.



(a)



(b)



(c)

Figure 6- 7: Number of times each track was used as part of a solution path (forward and reverse cases; only one of the best solutions); (a) training, (b) testing_1, (c) testing_2.

6.2 Characteristics of data as used in Representation 1

The analysis made in the previous section (6.1) was based on the consideration of a case as derived from the formulation of the single vehicle problem. Such a case consisted of an identification of a given situation in terms of layout state, a pair of origin and destination nodes and the corresponding desired solution path. Representation 1 consists of encoding exactly the same type of information in the input and output elements of a BP ANN. Therefore the characterization of the training and testing data sets presented in section 6.1 completely describes the data generated when converted to a format suitable to a ANN derived from Representation 1. There is however one aspect which must be noted and that results from the encoding of the information as defined in Representation 1. This particular encoding consists of binary elements, 40 in the ANN input and 24 in the ANN output (section 5.2.1). The total number of combinations that can result from these elements of the BP ANN when we are not concerned with its association with the single vehicle problem can be calculated as 2^{40} and 2^{24} at its input and output respectively. These are very large numbers, which when compared with the estimates for the number of possible paths and instances of the problem considered in the previous section (6.1) indicate that only a reduced set of the different codifications possible in this representation is of interest to encode solutions for the single vehicle problem.

6.3 Characteristics of data as used in Representation 2

This representation uses the same data characterized in the two previous sections, but then transforms each complete solution path into as many cases as the number of tracks (or steps) used in the solution path. A case in Representation 2 can not be interpreted exactly as a case derived from the formulation of the problem or in Representation 1. It has to be defined according to the input and output elements of the ANN for Representation 2. This will have implications for the number of cases and the other characteristics of the training and testing data sets.

The information associated with the input to the ANN is of the same type as in the ANN in Representation 1: the layout state is defined by the tracks set as unavailable, and a pair of origin and destination nodes. However since the path is defined in successive steps it now becomes necessary to

account for the location from where the search for the path is started. Also at each step the current node becomes the origin node number which makes it necessary to consider the nodes at junctions. These new conditions result in it being possible that there are a higher number of different cases at the ANN input. Considering the 5 junction nodes in the physical layout as possible current node numbers the total number of different combinations of origin/current nodes and destination nodes can be calculated as follows:

$$P_2^{16} + 5*16 = 320$$

When the different combinations of layout states, with a maximum of 4 tracks set as unavailable each time are considered (as in section 6.1), the total number of different cases can be calculated as follows:

$$\left(C_0^{24} + C_1^{24} + C_2^{24} + C_3^{24} + C_4^{24}\right)*320 = 4144320$$

While the number of different input cases increases in this representation the number of different solutions that can be interpreted from the ANN output reduces significantly. The output vector only has 2 elements which are interpreted as two co-ordinates in an xy-plane, and therefore the total number of different combinations reduces to 5 different alternatives: move to the Right, Left, Up or Down, or Stop.

These values represent estimates for the size of the input and output global set of different cases which may be of interest in the single vehicle problem. However, just as with Representation 1 a larger number of possibilities is obtained when considering the global set of different combinations that result from considering the 34 binary elements in the input of the BP ANN, which provide 2^{34} different combinations possible. And this is still a large number when compared with the cases that may be of interest to the problem and calculated above.

The total number of cases which result from a conversion of the training and testing sets presented before are summarised in Table 6- 3.

Table 6- 3: Number of cases as used in Representation 2 ANNs.

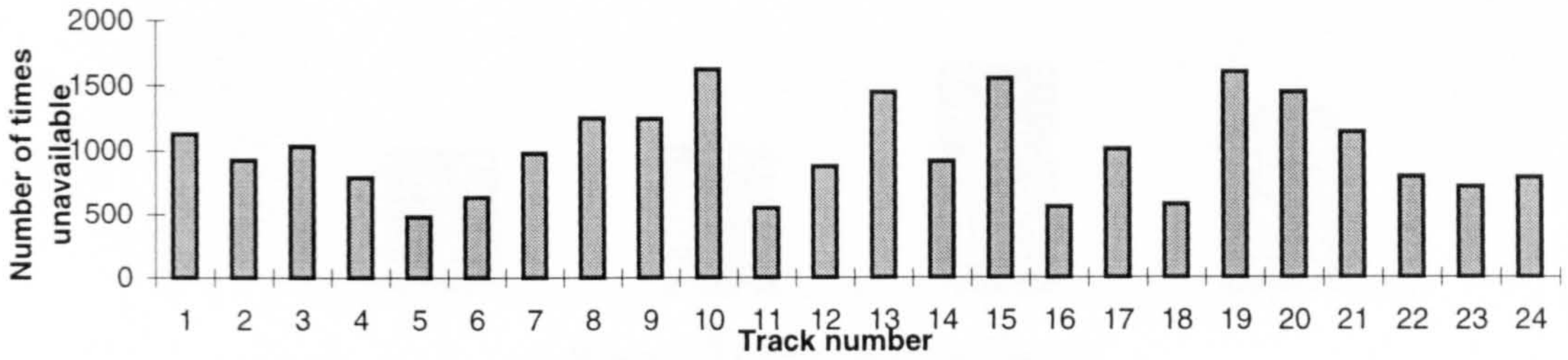
Data sets:	TOTAL	Training	Testing_1	Testing_2
NUMBER OF CASES (forward and reverse paths considered)	10408	9068	1010	330

Figure 6- 8 to Figure 6- 11 show the graphs representing the characteristics of the data that result from the separation of the complete path into steps. These graphs do not include tracks used to move out of each node in all cases because the paths linking the two nodes are the same in both representations. Also in a step-by-step approach each case will involve movement of one track length or else none if it is impossible to link the two nodes.

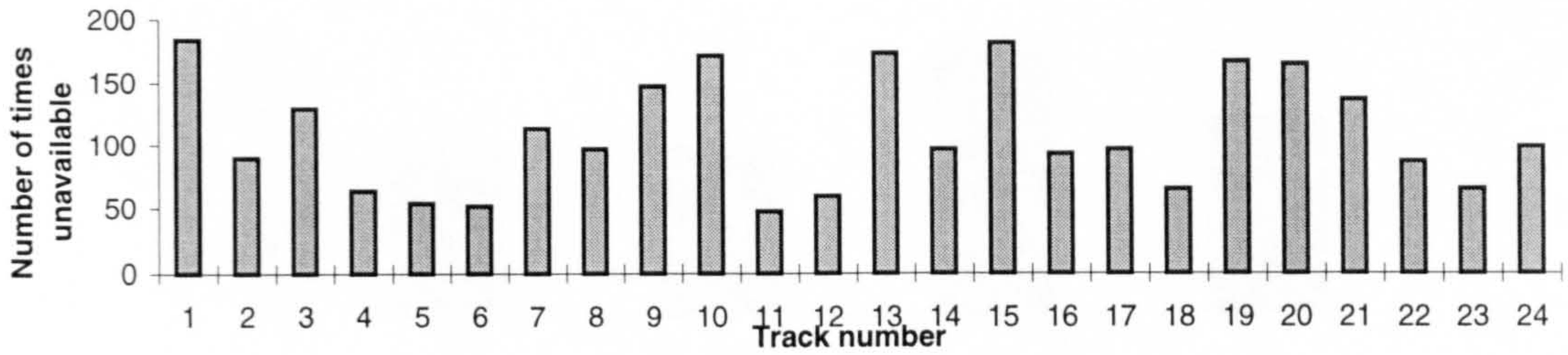
The characteristics which are susceptible to change are the ones related to the input elements in the ANN, including the ones related to the layout state and pairs of origin destination nodes. For example the cases with longer paths will result in the corresponding layout states being used in more cases; also there are now nodes which can be used only as origin nodes, such as the node at junctions. Furthermore even when using reverse paths a node is not used equally as an origin and destination node.

From the graphs of Figure 6- 8 it can be seen that the number of times each track is set as unavailable increased because the number of cases also increased. However apart from marginal changes associated with some tracks, in overall terms the distributions are comparable to the ones represented in Figure 6- 2. The same can be concluded in relation to the graphs on Figure 6- 9 which represent the number of tracks set as unavailable at the same time.

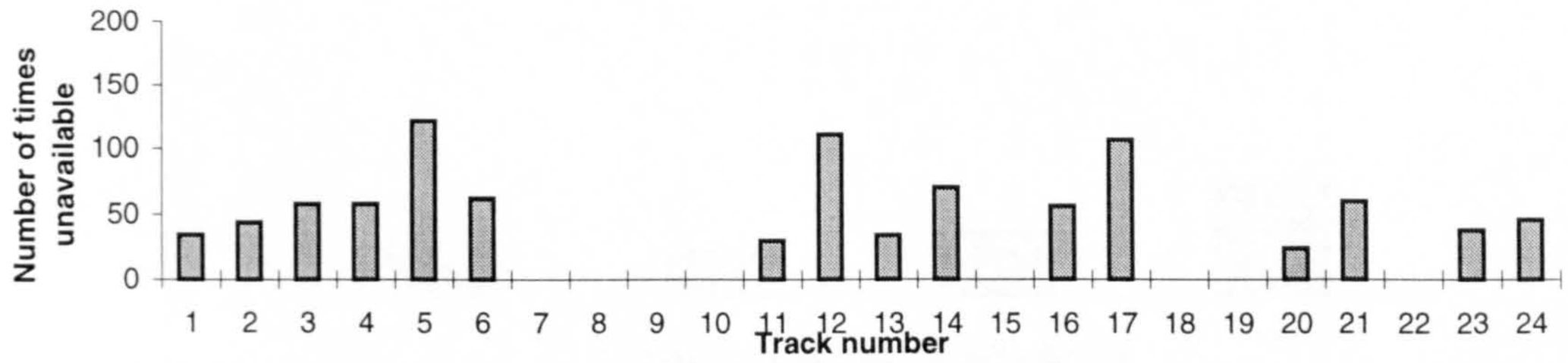
In the graphs of Figure 6- 10 representing how each node was used it can be seen that each node was used more times, due to an increase in the number of cases considered. It can also be seen that there is an increase in the number of times each node was used as a destination node relative to its use as an origin node, except when the nodes at junctions are considered.



(a)



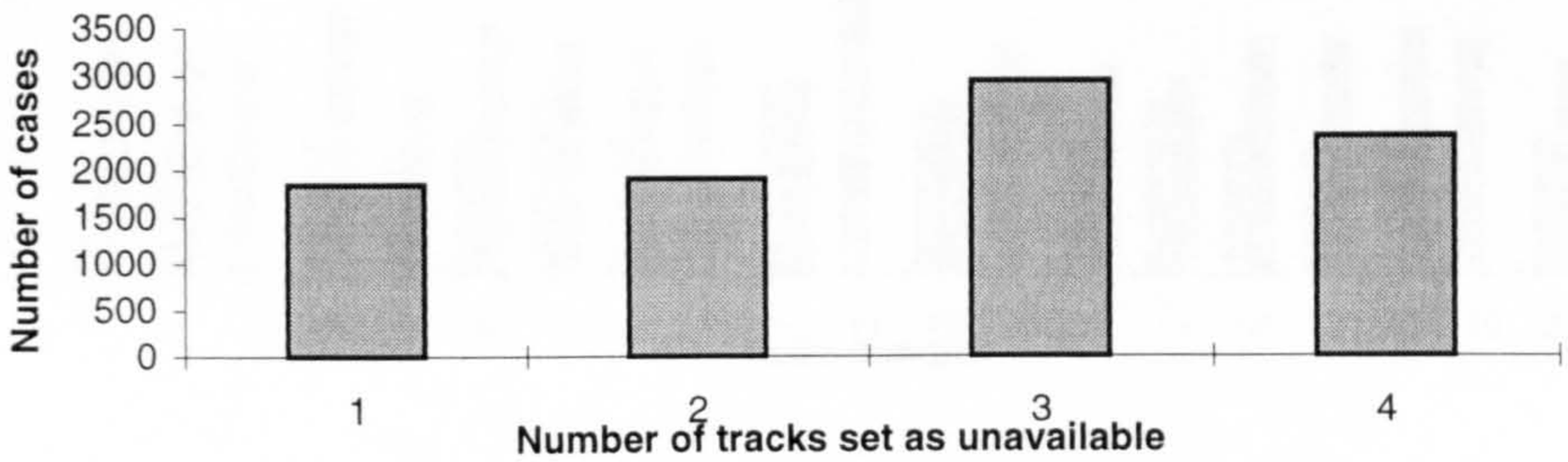
(b)



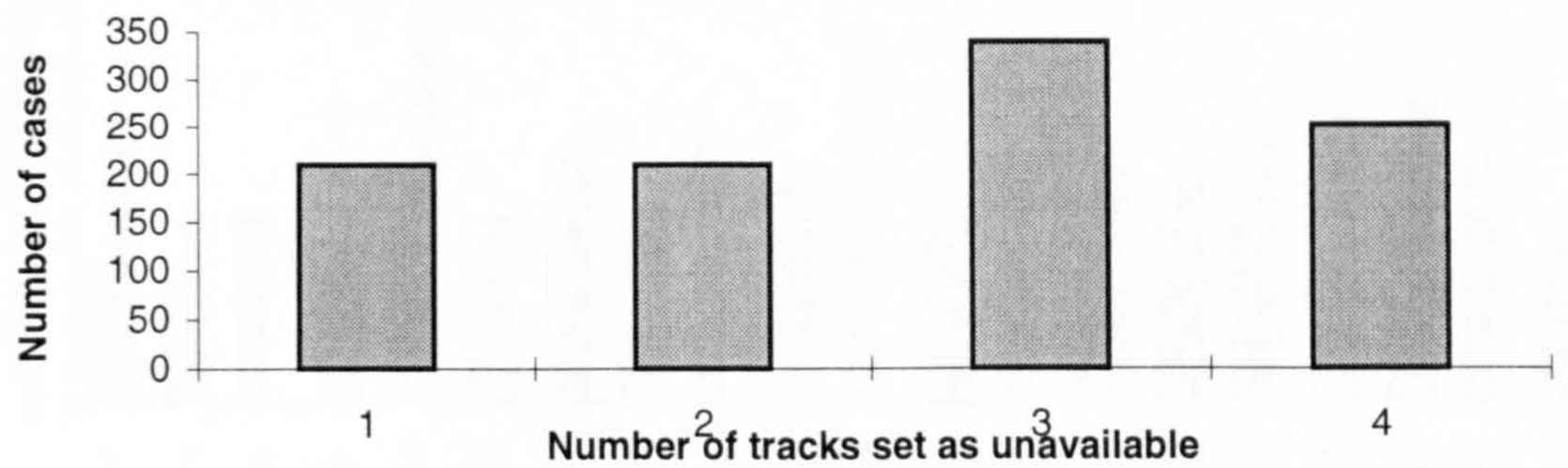
(c)

Figure 6- 8: Number of times each track was set as unavailable (forward and reverse cases),

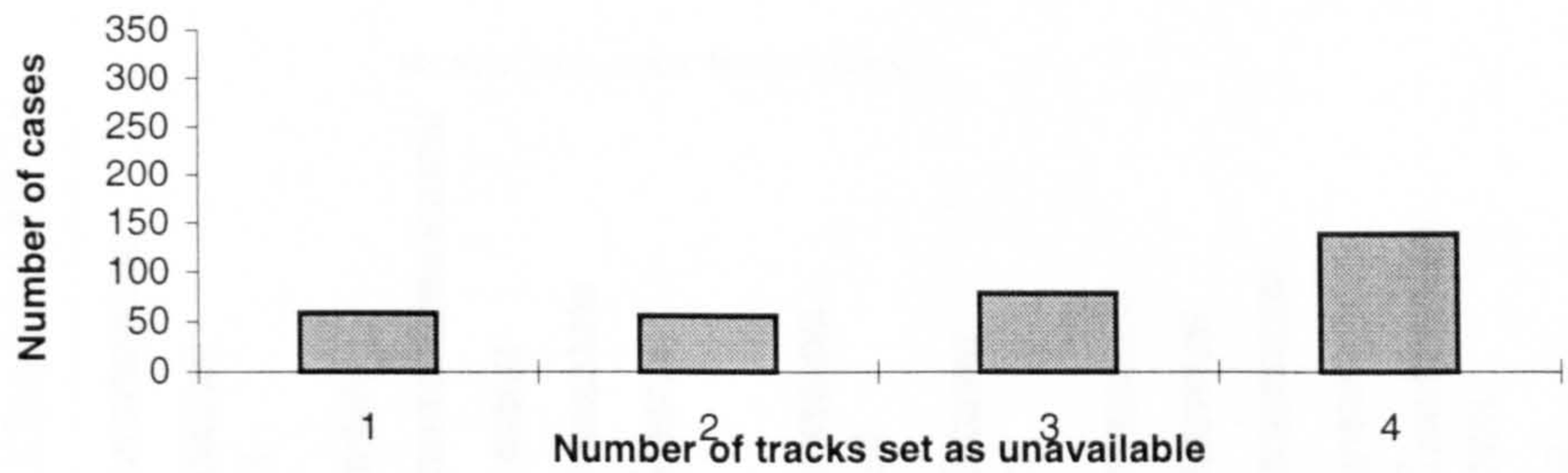
(a)training, (b) testing_1, (c) testing_2.



(a)

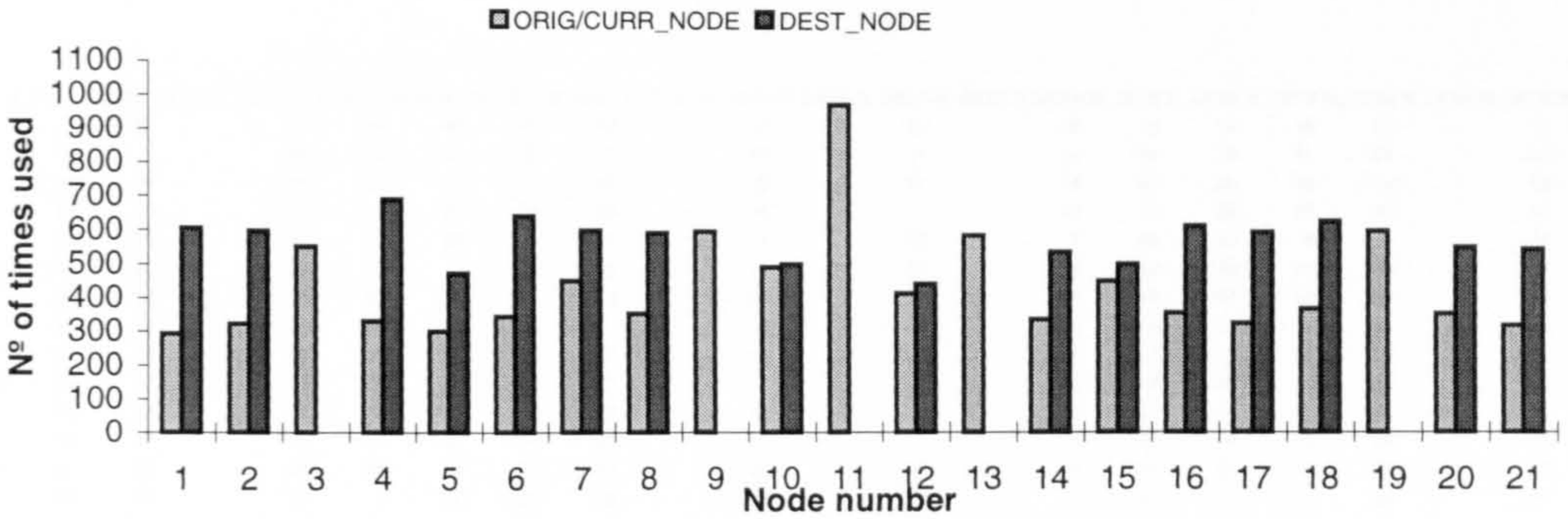


(b)

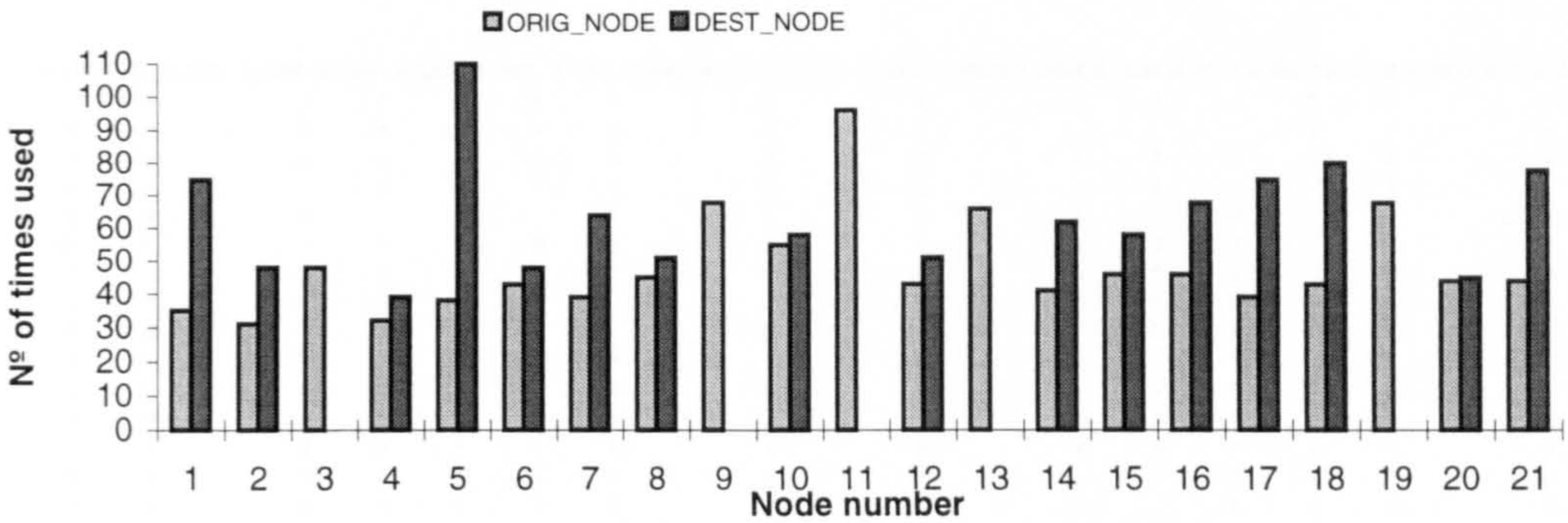


(c)

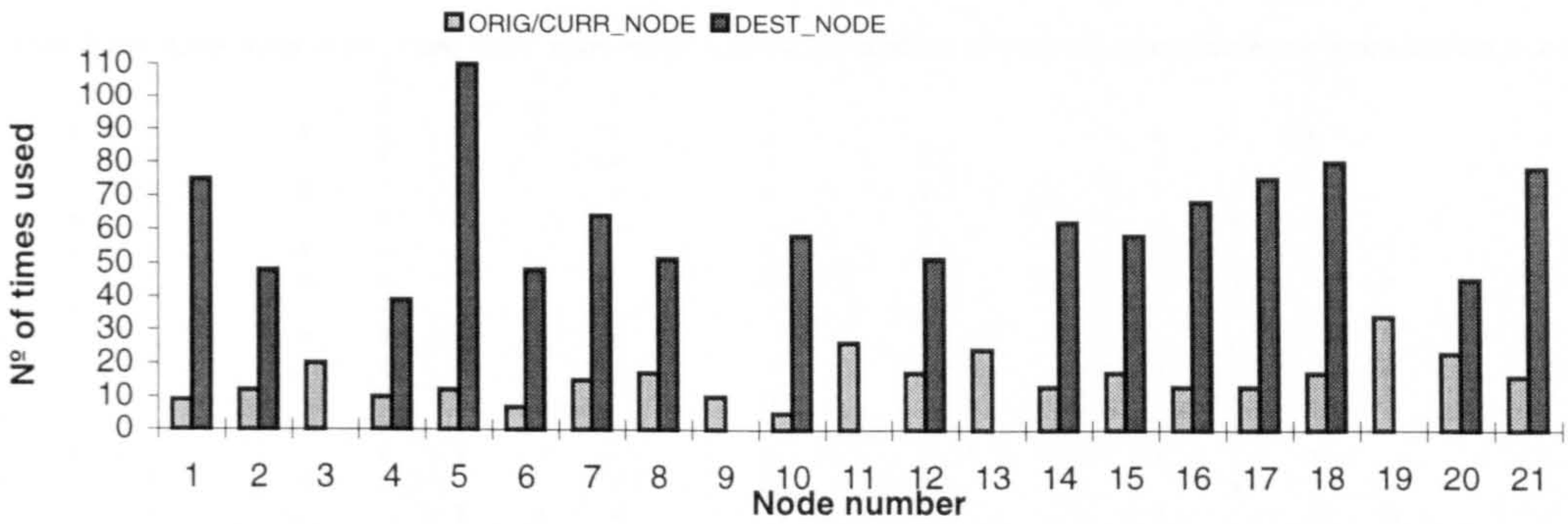
Figure 6- 9: Number of tracks set as unavailable (forward and reverse cases), (a) training, (b)testing_1, (c) testing_2.



(a)



(b)



(c)

Figure 6- 10: Number of times each node was used (forward and reverse cases), (a) training , (b)testing_1, (c) testing_2.

	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1		42		25	20	45	14	12		20		12		20	15	16	15	13		13	11
N_OR2	54			26	30	41	23	17		15		7		18	18	19	11	13		14	16
N_OR3	49	85		80	45	43	43	27		20		11		18	27	29	18	19		16	20
N_OR4	27	36			47	27	23	31		15		10		12	13	29	20	12		14	13
N_OR5	21	30		47		20	22	34		11		13		7	10	31	6	6		19	21
N_OR6	65	36		24	19		18	16		26		17		25	14	14	22	18		17	12
N_OR7	30	54		58	23	31		18		24		20		29	39	27	29	27		19	20
N_OR8	18	20		44	51	15	19			12		18		13	18	38	11	15		29	30
N_OR9	61	36		32	11	84	35	23		49		25		66	23	17	53	40		22	15
N_OR10	41	28		39	14	61	40	23				32		48	28	26	39	34		20	15
N_OR11	50	53		72	32	64	106	47		75		75		50	81	56	59	62		44	44
N_OR12	22	11		31	27	31	30	46		29				19	21	49	21	26		25	24
N_OR13	24	20		43	48	25	29	85		21		36		17	21	85	16	23		41	48
N_OR14	24	20		19	6	26	26	17		30		16			22	9	57	38		14	11
N_OR15	20	28		29	13	18	42	24		29		29		24		21	38	54		41	38
N_OR16	21	17		20	24	16	23	41		15		19		14	16		13	25		40	50
N_OR17	19	16		23	6	21	17	16		27		15		48	23	10		45		21	17
N_OR18	16	15		20	8	16	19	23		26		22		40	26	21	62			28	24
N_OR19	18	22		26	13	19	30	28		29		28		37	44	34	57	68		68	55
N_OR20	13	14		16	14	19	18	27		15		19		16	22	38	26	37			57
N_OR21	12	13		14	20	16	18	33		9		16		13	18	43	19	30		42	

(a)

	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1		6		1	1	6	4	2		2		2		1	1		3	5			1
N_OR2	5			2	3	5	4	2		2		2					2	1		1	2
N_OR3	5	6		5	5	1	5	3		4		2		1	1	2	2	3		1	2
N_OR4	3	3			7	1	1	3		2		2		2	2	3		2			1
N_OR5	2	3		5		1		3		1		4		2	3	4	2	4		1	3
N_OR6	10	5			1		4	2		2		2		1	3	1	5	5		1	1
N_OR7	3	3		3	3	1		1		7		1		2	4	3	2	3		2	1
N_OR8	2	1		3	12	1				1		3		2	4	3	3	4		2	4
N_OR9	9	2			3	8	6	3		5		2		7	3	1	8	5		3	3
N_OR10	4	3		1	3	5	7	4				3		6	3	1	3	4		3	5
N_OR11	4	4		2	10	5	10	4		11		6		5	8	5	3	6		4	7
N_OR12	1	2			11	1	2	3		2				2	3	4	2	3		1	6
N_OR13	1	1		3	14	1	2	6		1		6		3	4	6	3	4		2	9
N_OR14	5	1		1	3	3	4	1		3					2	3	8	3		2	2
N_OR15	3	1		2	6	3	3	1		4		3		2		2	2	7		2	5
N_OR16	1			3	6	1	2	4		1		3		4	2		4	2		2	11
N_OR17	5	1		1	3	3	3	1		2		1		4	2	5		4		4	
N_OR18	6	1		1	4		2	1		2		2		4	2	5	8			4	1
N_OR19	4	2		2	6	1	2	1		3		3		5	5	6	7	9		6	6
N_OR20	1	2		2	4	1	2	2		1		1		4	3	6	5	2			8
N_OR21	1	1		2	5		1	4		2		3		5	3	8	3	2		4	

(b)

	N_DE1	N_DE2	N_DE3	N_DE4	N_DE5	N_DE6	N_DE7	N_DE8	N_DE9	N_DE10	N_DE11	N_DE12	N_DE13	N_DE14	N_DE15	N_DE16	N_DE17	N_DE18	N_DE19	N_DE20	N_DE21
N_OR1					2	2	1							1			1			2	
N_OR2	2				2	1	2	1								1	1			1	1
N_OR3	2	3		2	2	1	2	2		1					2		1			1	1
N_OR4		1			1			2		1		2			2		1				
N_OR5		1		2		1		2		2		2			1		1				
N_OR6	2				2											1	1			1	
N_OR7	2	2		2	1	1		1		1					2			1		1	1
N_OR8		1		2	3	1				3		3			1	1		1			1
N_OR9	1				2	1				2				1		1	1			1	
N_OR10				1	2			1						1							
N_OR11	1	1		2	3	1	1	2		4		3			2			2		3	1
N_OR12				2	3	1	1	2		3						1		2		2	
N_OR13				2	3	1	1	5		3		4				2		2			1
N_OR14	2				2	1				2					1	1	3			1	
N_OR15	1	1		1	1			1		1		3		1		1		1		4	1
N_OR16		1				1	1	3				1			1		1	2		1	1
N_OR17	1				2	1				1				3	1	2				2	
N_OR18	1				1	1	1	1		1		1		1	1	2	2			3	1
N_OR19	2	1			1	1	1	1		1		3		1	3	3	2	5		7	2
N_OR20	2	1				1	1	1				2			2	4	2	4			3
N_OR21		1				1	1	2							1	4	1	3		2	

(c)

Figure 6- 11: Combinations of pairs origin/destination nodes used (forward and reverse cases), (a)training, (b) testing_1, (c) testing_2.

6.4 Overall conclusions

A characterization of the different aspects of the data sets such as the ones associated with the specification of an instance of the problem (i.e. origin and destination nodes, availability of tracks), and the solution paths (i.e. path lengths, tracks used in the solution path) was presented. This analysis has shown that the method of generating the data appears fair and that the amount of training and testing data produced, although small in comparison to the total possible cases is acceptable.

Chapter 7

Experiments and Results with ANN Solutions for the Single Vehicle Case

Having data available for training and testing BP ANNs, experiments were performed in order to decide on the design parameters of the ANN under test. This chapter describes these experiments and the results obtained with the ANNs solutions to the single vehicle case. The ANNs were developed using the NeuralWorks Professional II [NeuralWare,Inc.,Pittsburgh, USA], ANN software package . During this investigation it was found that the use of simple heuristic rules could significantly improve the performance of the ANN solution and thus an "hybrid solution" is proposed.

7.1 Design parameters of a BP ANN

The codification and number of Input and Output elements is only one of the design parameters that must be specified in a Backpropagation ANN. The other elements are the number of hidden layers and hidden neurons, the learning coefficient and momentum term, the epoch size and number of training iterations.

The selection of the values for these parameters so that the best performance of the ANN is obtained, normally depends on the particularities of each application. Even though some guidelines are reported [i.e. Ref. 12, Ref. 13, Ref. 80] on how to proceed in the selection of the parameters, they cannot replace the replication of training and evaluation cycles. The way chosen to proceed here was starting with a set of values for each parameter, train the network, evaluate performance and repeat the process with new values until the results obtained were "satisfactory", that is the new values consistently result in the network performing no better than with values previously tried.

The following considerations were taken into account when deciding on how to choose the initial values and how to proceed from that point onwards with the tests:

- always used a standard backpropagation ANN with a sigmoid transfer function and the generalized delta-rule [Ref. 12, Ref. 14];

- number of hidden layers and neurons

- generally one hidden layer is sufficient [Ref. 12];

- only one hidden layer is enough to approximate any *continuous* function, provided it has enough neurons [Ref. 13];

-sometimes, specifying more than one layer of hidden neurons can be more efficient, i.e. the network learns faster [Ref. 12];

- it is quite possible that a smaller network, whose minimum error distance is greater than zero can predict better than a larger network with a zero error distance [Ref. 84];

- too few neurons in the hidden layer can lead to underfitting. Too many neurons can contribute to overfitting [Ref. 60];

- the number of hidden neurons may lie between the total number of input and output elements [Ref. 80, Ref. 81];

- a widely used heuristic rule is that a network should average at least 10 samples per weight (connection) in the network [Ref. 84, Ref. 80, Ref. 81].

Based on these considerations it was decided to test an ANN with one and two hidden layers. For the number of hidden elements it was decided to start with 5 neurons and proceed iteratively with an increase of this number as long as performance increases or reducing it when performance starts to decline.

- initial values of weights

- replicate training with different small initial values for the weights reduces the possibility of the solution found corresponding to a local minimum [Ref. 12, Ref. 13]

- in order to reproduce the results obtained it is required to know the initial values of the weights

Different random seed numbers were used to ensure the initial values were different and the results could be reproduced.

- learning rate

- a large learning rate leads to unstable learning [Ref. 60, Ref. 12, Ref. 13]

- a small learning rate require longer to train [Ref. 60, Ref. 12, Ref. 13]

- momentum term, includes in the new weight change a fraction of the last weight change:

- if set to "0" the weight change does not include a fraction of last weight change

- if set to "1" the weight change is the same as last weight change.

The experiments done include ANNs based on representations 1 and 2. In both cases the performance of each ANN was evaluated after being trained, in terms of its capability to produce a value for each of its output elements that correspond to the value indicated in the desired output element when tested with testing data sets. However when considering the main objective of producing paths it was also necessary to evaluate how that performance was representative of a desired solution path.

7.2 Initial experiments with ANNs based on representations 1 and 2

Initial experiments were done in order to test whether an ANN based on representations 1 and 2 presented considerable differences and if they indicated one as the more promising alternative. These initial experiments were performed using an initial set of data. Only one of the two possible paths ("forward" and "reverse") that use the same tracks to link two nodes was considered in these patterns. When converted to the format of Representations 1 the data sets used contained 449 training patterns and 50 testing patterns. When converted to Representation 2 they represented 2105 training patterns

and 228 testing patterns. After the initial experiments described in this section the data sets were complemented with cases generated later and resulted in the data sets described in Chapter 6 which were used in the experiments described in the next section.

All the experiments described in this section were based on a data format which derived from an initial ordering of the elements defined in the physical layout as represented in Figure 7- 1.

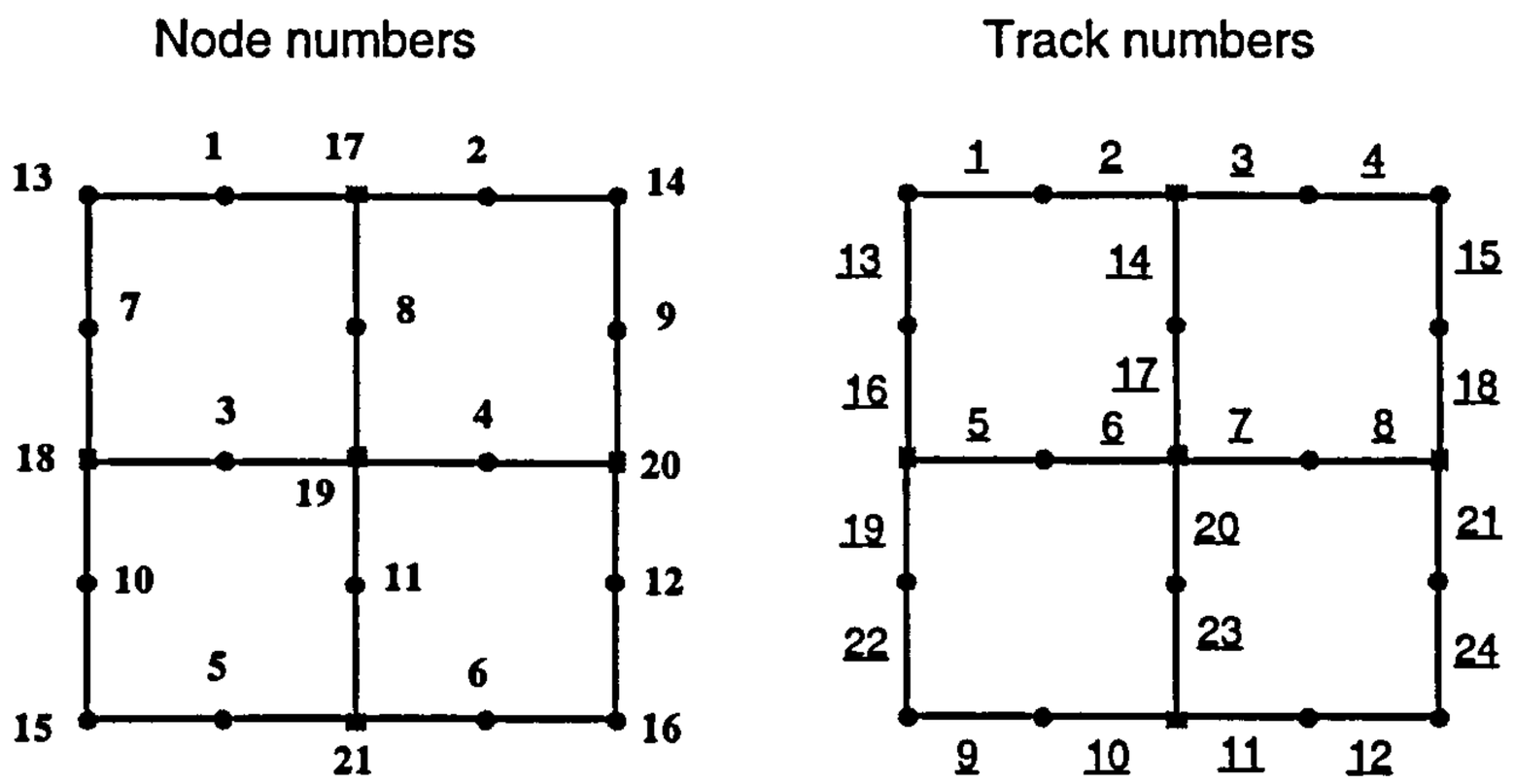


Figure 7- 1: Initial numbering (LAYOUT_1) of nodes and tracks in the physical layout.

Starting with ANNs based on Representation 1 a first indication, of its limitations in terms of its ability to produce the desired complete solution paths in the test cases, was apparent when testing ANNs with different numbers of hidden elements (Table 7- 1). This occurred even when its performance, measured by the number of correct elements in the ANN output, was relatively high. Observing the results in Table 7- 1 we can see that the maximum number of correct paths was 16%, obtained using an ANN with 50 elements in the hidden layer. These results were consistent when different initial values for the weights were generated.

A possible interpretation of this relatively poor performance relates to the reduced number of training cases used or having not yet experimented with values for the other design parameters which could provide an optimized ANN. However, when relating to the particular representation and encoding of these ANN's it is apparent that an extremely high number of correct elements identified at the ANN Output will be required in order to produce solutions from which valid paths could be extracted. Also

when analysing the Input patterns, with a maximum of 4 tracks being set unavailable and two nodes to be linked, we can see that at least 36 of the 40 elements are in a state ("zero") that does not change in the various patterns. The similarity of the Input elements may result in an increased difficulty of the ANN to distinguish and identify valid solutions. It was thus decided to test ANNs based on Representation 2 which uses an encoding with fewer elements and is therefore likely to be less sensitive to the problems encountered in the ANNs tested thus far.

Table 7- 1: Testing number of elements in one hidden layer (ANNs: Representation 1)

ANN: input=40;output=24;learning rate=0.3;momentum term=0.2;epoch=16;100000 iterations				
Num. elem. hidden layer	(%) Correct elements TEST	TRAINING	(%) Correct desired paths TEST	TRAINING
5	80	83	4	6
25	82	94	12	54
50	**84**	**95**	**16**	**77**
100	84	95	14	77

The experiments performed with ANNs based on Representation 2 considered two measures to evaluate the performance of each ANN. One was the number of correct movement decisions that could be interpreted from a direction vector associated with the two output elements of the ANN. The other was the number of correct paths obtained by linking origin and destinations nodes using the tracks resulting from the movement decisions based on the ANN output direction vector. Both these measures were compared against the desired output for each case in the test or training sets. A more detailed analysis that includes checking other possible solution paths such as a longer path or even an equivalent shorter path but using different tracks, was left to a later stage and is described in the next section.

The initial experiments with ANNs based on Representation 2 were used to investigate the number of hidden elements and also hidden layers which resulted in the best performances. As can be seen in Table 7- 2, the (%) of correct complete paths was generally higher than the ones obtained with the ANNs based on Representation 1.

Table 7- 2: Testing number of elements in one or two hidden layers (Representation 2)

ANN: input=34;output=2;learning rate=0.3;momentum term=0.2;epoch=16;100000 iterations				
Hidden layer (s)	(%) Correct step decisions TEST	TRAINING	(%) Correct desired paths TEST	TRAINING
5	66	60	33	27
25	68	77	37	45
50	70	81	37	48
20	**72**	**76**	**42**	**45**
10	66	70	35	33
5+5	54	63	12	23
10+5	59	62	23	26
5+10	47	46	12	13

Fixing the ANNs structure to one hidden layer and 20 hidden elements the same approach was followed to investigate whether changing values for the learning rate, and momentum term, epoch size and number of training cycles could improve these results. Table 7- 3 gives an indication of the results obtained with different learning rates and momentum terms. From these tests there seemed not to be one set of values which significantly improves the percentage of correct paths. Testing for three more different epoch sizes and different learning iterations the best value obtained for correct desired paths was 47% obtained with an epoch size of 50 and 100 000 iterations in a ANN with 20 hidden elements and learning rate and momentum term values of 0.7 and 0.6 respectively.

Table 7- 3: Results obtained with different learning rates and momentum terms (Representation 2)

ANN: input=34;output=2;hidden layer=20;epoch=16;100000 iterations		
(Learning rate ; Momentum term)	(%)Correct step decision	(%)Correct desired paths
(0.1 ; 0)	56	21
(0.7 ; 0)	71	35
(0.9 ; 0)	69	37
(0.5 ; 0)	65	33
(0.3 ; 0)	66	33
(0.7 ; 0.9)	69	40
(0.7 ; 0.1)	71	30
(0.7 ; 0.6)	73	44
(0.7 ; 0.4)	61	28
(0.7 ; 0.3)	71	42
(0.3 ; 0.6)	72	37
(0.3 ; 0.2)	72	42

While these initial results do not represent a large number of desired correct paths it was considered appropriate to investigate whether:

- they could be confirmed and improved using a larger set of data,

- through a more detailed analysis of the impossible solutions, simple heuristics could be derived that improve its performance,

To investigate these issues more cases were generated and were combined with the cases already used to produce the data sets described previously in Chapter 6. The next section describes the experiments performed.

7.3 Further experiments with ANNs based on Representation 2

This new set of experiments concentrated first on the values of the design parameters using the new data sets. The same values were used in the ANNs now being trained and the results obtained were similar to the ones obtained previously. Table 7- 4 presents the results, in terms of number of correct step decisions and complete paths, obtained from training ANNs with 20 hidden elements, an epoch size of 50, and different combinations of learning rate, momentum term, and training iterations.

The results in this table include ANNs trained first with cases which only use paths starting at one of the two possible nodes, and then duplicating the number of cases by considering the "forward" and "reverse" paths for each case. The testing data sets were also organized in "forward" and "reverse" paths depending on the position of the origin and destination nodes in the physical layout. In a "forward" path the destination node will always be to the right or below the origin node, considering the top and left corner as the one corresponding to node number 13 in the physical layout numbering of Figure 7- 1 (see previous section).

While the experiments are not exhaustive it is possible to observe that in general the best results are marginally better than the results obtained in the initial set of experiments and they are obtained when using both "reverse" and "forward" cases during training. The results obtained when using the data sets without separating the cases into "reverse" and "forward" are marginally worse than the best one obtained in the initial set of experiments. In this case using a larger set of cases did not seem to be sufficient to obtain a best performance.

Using "forward" and "reverse" cases during training shows a more consistent improvement of performance when testing both "forward" and "reverse" cases, relative to using cases consisting only of one of the possible paths between two nodes. This is not unexpected because the training data includes the same number of cases in each of the two paths in opposite directions.

There is however an exception which is when testing cases containing different layouts from the ones used for training, and using only "forward" paths (testing2_F, third ANN relative to second ANN in Table 7- 4). The number of correct complete paths given by the ANN, when trained using the same conditions but with both "reverse" and "forward" cases, is lower than when trained with cases using only one path. While this difference seems significant, it is much less so when only correct "step" decisions are considered. This makes it more plausible that the poor performance may derive from peculiarities of this set of testing data and training conditions. A more detailed analysis of the individual cases in the test set showed that the cases correctly identified by each ANN were not coincident. However in spite of this, if all equivalent shorter paths were considered, the percentile difference between these two solutions would only be of 2%. This 2% corresponds to 16 correct paths found in one solution against 15 in the other when 41 cases are tested. It seems therefore possible to conclude that using "forward" and "reverse" cases has potential for improving performance relative to only using one of the possible paths between each of the pair of nodes considered.

The first two columns of results in Table 7- 4, obtained using ANNs trained with only one of the two possible paths between two nodes, show better performances when testing the "forward" cases (Testing1_F, Testing2_F) than when testing the "reverse" cases (Testing1_R, Testing2_R). This might be related to the fact that cases used for training were used as generated without a separation between "reverse" and "forward" paths, which was done with the testing cases. The "forward" cases present better results which might be related to the training cases used including more of this type of paths. That came as a result of the process used to identify one of the two nodes as the origin node which simply picked it up the node with the lower identification number.

Table 7- 4: Results of the 2 testing data sets, both with only "forward"(_F) or "reverse"(_R) cases.

ANN: input=34; output=2; hidden layer=20; epoch=50 ("stop" decision when modulus of movement direction vector < than 0.1)												
	TRAINING DATA CASES USED:											
	only one path each case				two paths each case ("forward" and "reverse")							
Learning rate:	0.7		0.3		0.3		0.7		(*)		(**)	
Moment. term:	0.6		0.6		0.6		0.6		0		0	
Training iter.:	100000		100000		100000		100000		100000		300000	
TEST DATA:	steps (%)	paths (%)	steps (%)	paths (%)	steps (%)	paths (%)	steps (%)	paths (%)	steps (%)	paths (%)	steps (%)	paths (%)
Testing1_F	74	40	72	39	74	45	69	38	57	24	67	35
Testing1_R	59	22	54	16	76	46	69	40	54	17	69	32
Testing2_F	70	34	69	39	68	34	72	39	58	22	73	51
Testing2_R	52	24	50	20	78	46	72	39	62	24	72	37
(*)(**) Learning rate changed during training.												

The number of correct complete paths which have been used as a performance measure thus far only considered as correct solutions, the paths that were exactly the same as the ones indicated in the desired output vector; or a "Stop" decision at the origin node when a path linking the two nodes was not possible. However there are cases where multiple solutions are possible, either equivalently shorter or longer. We must therefore investigate whether the ANN's incorrect solutions in the results already obtained may in fact correspond to possible paths linking the origin and destination nodes. At the same time the analysis of the impossible solutions may prove useful in deriving heuristics capable of preventing them and therefore being used as an alternative approach to increase the ANN performance due to the limitations of the optimization of the ANN design parameters through more training and testing experiments, possibly due to a lack of training data. The next section describes the results obtained when analysing, in detail, the solutions obtained with the ANNs.

7.3.1 Detailed analysis of solution paths with ANNs based on Representation 2

The detailed analysis consists of an identification of three types of possible and three types of impossible solutions based only on the evaluation of complete paths linking the origin and destination nodes. The possible solutions are:

- one associated with the desired output vector solution,

- an equally short solution to the one associated with the desired output vector, but using different tracks,

- a path capable of linking the origin and destination nodes but using more tracks.

And the impossible solutions are where:

- movement direction indicates a track not defined in the physical layout,

- movement direction indicates a track temporarily set as not available,

- deadlock situation characterised either by the movement direction indicating movement into a track already used in the path or indicating the need to stop at a node different from the destination node.

The detailed analysis showed that the majority of the possible solutions correspond to the solution associated with the desired output vectors. That is the ANN very seldom finds a correct solution which is either longer than the one selected as the desired output vector or an equivalent solution but using different tracks. One reason that may explain these results has to do with using only one of the possible solutions when more than one was possible for each case in the training set. Table 7- 5 presents results obtained when evaluating the three types of possible solutions.

Table 7- 5: Results considering the three possible solutions. Where "Des.", "Equ.", and "Long" represent respectively in relation to the desired output vector: exactly the same, same length but using different tracks, and longer paths.

ANN: input=34: output=2: hidden layer=20: epoch=50 ("stop" decision when modulus of movement direction vector < than 0.1)																	
	TRAINING DATA CASES USED:																
	only one path each case						two paths each case ("forward" and "reverse")										
Learn. rate:	0.7			0.3			0.3			0.7			(*)			(**)	
Mom. term:	0.6			0.6			0.6			0.6			0			0	
Train. iter.:	100000			100000			100000			100000			100000			300000	
TEST DATA:	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)
Testing1_F	39.6	1.8	0.9	38.7	2.7	1.8	45	2.7	0	-	-	-	-	-	-	-	-	-
Testing1_R	-	-	-	-	-	-	45.9	1.8	0.9	-	-	-	-	-	-	-	-	-
Testing2_F	34.1	4.9	0	39	0	0	34.1	2.4	0	39	2.4	0	21.9	0	0	48.8	2.4	0
Testing2_R	23.4	0	2.4	19.5	0	0	46.3	4.9	0	39	2.4	2.4	24.4	0	0	36.6	4.9	0
(*)(**) Learning rate changed during training.																		

Table 7- 6: Results considering the three types of impossible solutions. Where "Imp1", "Imp2", and "Imp3", represent respectively: movement into a non-defined track, movement into a track set unavailable, movement into a repeated track or stop before destination node.

ANN: input=34: output=2: hidden layer=20: epoch=50 ("stop" decision when modulus of movement direction vector < than 0.1)																	
	TRAINING DATA CASES USED:																
	only one path each case						two paths each case ("forward" and "reverse")										
Learn. rate:	0.7			0.3			0.3			0.7			(*)			(**)	
Mom. term:	0.6			0.6			0.6			0.6			0			0	
Train. iter.:	100000			100000			100000			100000			100000			300000	
TEST DATA:	Imp1 (%)	Imp2 (%)	Imp3 (%)	Imp1 (%)	Imp2 (%)	Imp3 (%)	Imp1 (%)	Imp2 (%)	Imp3 (%)	Imp1 (%)	Imp2 (%)	Imp3 (%)	Imp1 (%)	Imp2 (%)	Imp3 (%)	Imp1 (%)	Imp2 (%)	Imp3 (%)
Testing1_F	26.1	20.7	10.8	28.8	18.9	9.0	22.5	20.7	9.0	-	-	-	-	-	-	-	-	-
Testing1_R	-	-	-	-	-	-	15.3	25.2	10.8	-	-	-	-	-	-	-	-	-
Testing2_F	19.5	19.5	21.9	21.9	21.9	17.1	19.5	19.5	24.4	24.4	24.4	9.8	51.2	12.2	14.6	29.3	9.8	4.9
Testing2_R	24.4	24.4	24.4	41.5	21.9	17.1	17.1	24.4	7.3	17.1	12.2	26.8	48.8	14.6	9.8	21.9	19.5	14.6
(*)(**) Learning rate changed during training.																		

Analysing the results (Table 7- 6) in terms of different types of impossible solutions it can be concluded that all three are significant and in the majority of the cases the number of "deadlock" type solutions is lower. This can be verified by the results in the third group of columns in (Table 7- 6) where three of the four data sets tested have lower values for the deadlock solutions and which range from 7.3% to 10.8%, against the two other types of impossible solutions which range respectively from 15.3% to 22.5% and 20.7% to 25.2%. In the other data set of that column the number of deadlock type impossible solutions is 24% against 19.5% for each of the other two types of impossible solutions.

While "deadlock" situations are of very little interest, in a practical situation moving into a non-existing track or a track which is temporarily blocked cannot be accepted. It was therefore decided to test an alternative interpretation of the ANN output which could prevent these situations from occurring. This heuristic was applied only when the movement direction indicated a non-existent or blocked track and consisted first of selecting a movement direction based only on the value of the element of the output vector which pointed to an available track. In the case that an alternative direction could not be implemented then it would stop at the current node. Tables (Table 7- 7 and Table 7- 8) present the results obtained when these alternative decisions are taken into account. The general increase in performance now places the number of correct paths almost always near or above the 50 % margin. However it must be noted that the number of "deadlock" situations, the only impossible situations now allowed, rose substantially which means that there are still a significant number of cases not solved.

Table 7- 7: Results in terms of the three types of possible decisions obtained using a heuristic to prevent movement to a non-existing or unavailable track.

ANN: input=34; output=2; hidden layer=20; epoch=50 ("stop" decision when modulus of movement direction vector < than 0.1)																	
	TRAINING DATA CASES USED:																
	only one path each case						two paths each case ("forward" and "reverse")										
Learn. rate:	0.7			0.3			0.3			0.7			(*)			(**)	
Mom. term:	0.6			0.6			0.6			0.6			0			0	
Train. iter.:	100000			100000			100000			100000			100000			300000	
TEST DATA:	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)	Des. (%)	Equ. (%)	Long (%)
Testing1_F	53.2	2.7	1.8	45.9	3.6	2.7	57.7	4.5	0	-	-	-	-	-	-	-	-	-
Testing1_R	-	-	-	-	-	-	52.3	3.6	1.8	-	-	-	-	-	-	-	-	-
Testing2_F	43.9	4.9	0	41.5	0	0	46.3	2.4	0	53.7	2.4	0	31.7	4.9	2.4	61.0	2.4	0
Testing2_R	29.3	0	2.4	29.3	0	0	56.1	4.9	0	43.9	4.9	2.4	39.0	4.9	0	46.3	7.3	2.4
(*)(**) Learning rate changed during training.																		

Table 7- 8: Results in terms of impossible decisions (type "deadlock") obtained using a heuristic to prevent movement to a non-existing or unavailable track.

ANN: input=34; output=2; hidden layer=20; epoch=50 ("stop" decision when modulus of movement direction vector < than 0.1)						
	TRAINING DATA CASES USED:					
	only one path each case		two paths each case ("forward" and "reverse")			
Learn. rate:	0.7	0.3	0.3	0.7	(*)	(**)
Mom. term:	0.6	0.6	0.6	0.6	0	0
Train. iter.:	100000	100000	100000	100000	100000	300000
TEST DATA:	Imp3 (%)	Imp3 (%)	Imp3 (%)	Imp3 (%)	Imp3 (%)	Imp3 (%)
Testing1_F	42.3	47.8	37.8	-	-	-
Testing1_R	-	-	42.3	-	-	-
Testing2_F	51.2	58.5	51.2	43.9	61.0	36.6
Testing2_R	68.3	70.1	39.0	48.8	56.1	43.9
(*)(**) Learning rate changed during training.						

7.3.2 Experiments using different identification numbers for the elements in the physical layout (ANNs Representation 2)

The last set of these experiments was intended to investigate whether the order in which numbers were chosen to identify each track and node in the physical layout would make any difference to the performance of the ANNs, i.e. is the network learning the numbering pattern rather than 'real solutions'. Using different identification numbers for tracks and nodes the encoding, and therefore the "patterns" presented to ANN, appear different. All previous experiments were based on order numbering (Figure 7- 1) found convenient when implementing the program for generation of data, but three others were generated for comparison. One was based on a sequential ordering starting with the lowest number associated with a track or node at the top left corners (Figure 7- 2). The other two were based on random numbers (Figure 7- 3 and Figure 7- 4). The results obtained are presented in Table 7- 9 , and Table 7- 10.

The sequential ordering LAYOUT_3 which identifies each element starting with lower numbers at the top left corner and using successively higher identification numbers for the elements to the right and down gives better results generally, except for testing2_R cases. The use of ordering numbers that tend to associate the same movement direction with the relative position of an element identified by a lower number and another with a higher number (i.e. move right and/or down to get to an element with a higher identification number) may be one of the reasons why LAYOUT_3 generally performs better. Obviously this depends on availability of the tracks which may require a movement in an opposite direction.

One other relevant issue might be the association of each track element in the ANN INPUT with the relative position of the tracks in the physical layout. An order numbering which is more favourable to the use of similar (i.e. lower and higher) identification numbers with adjacent tracks might allow an easier identification of patterns.

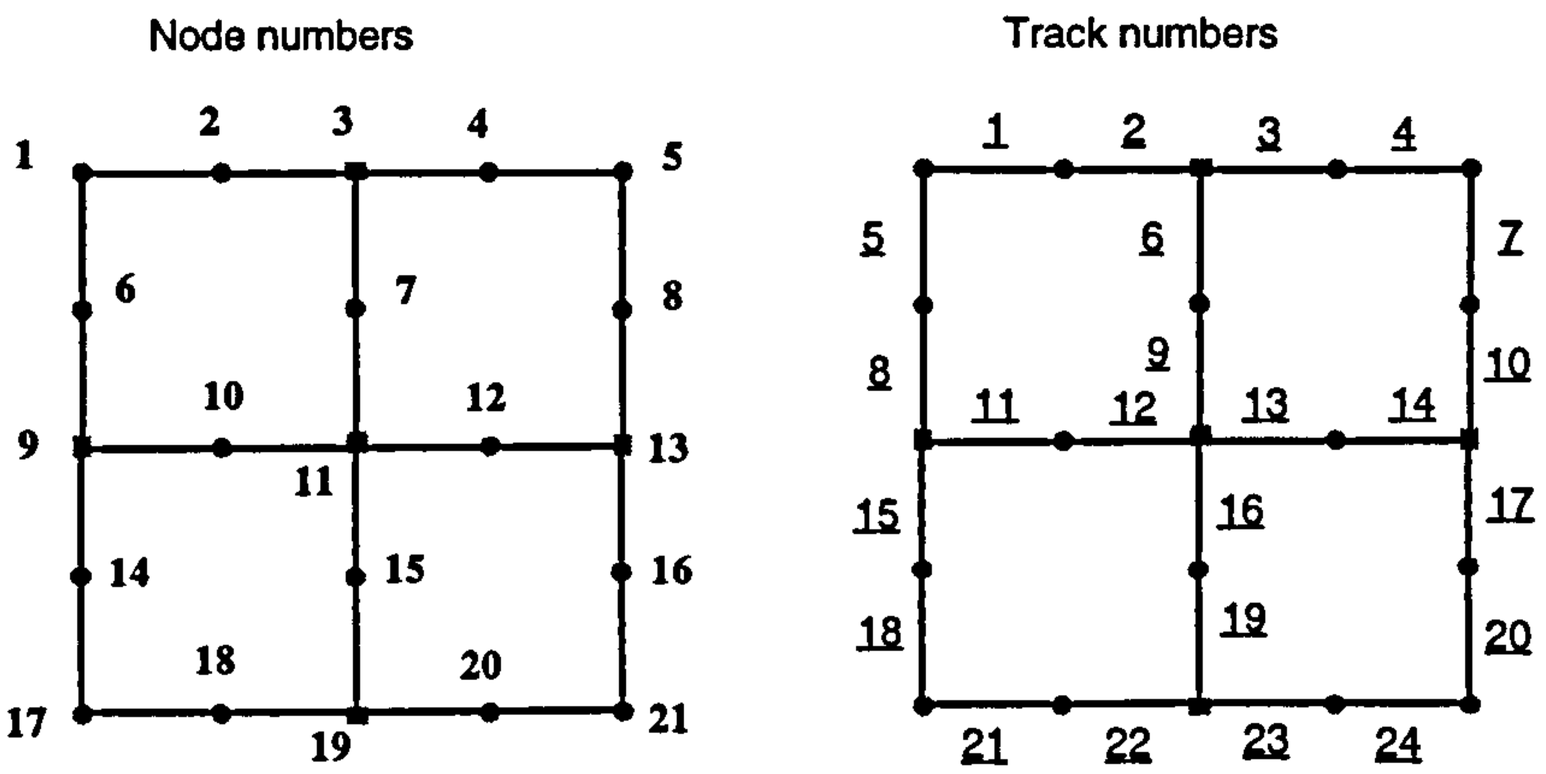


Figure 7- 2: Sequential numbering (LAYOUT_3) of nodes and tracks in the physical layout.

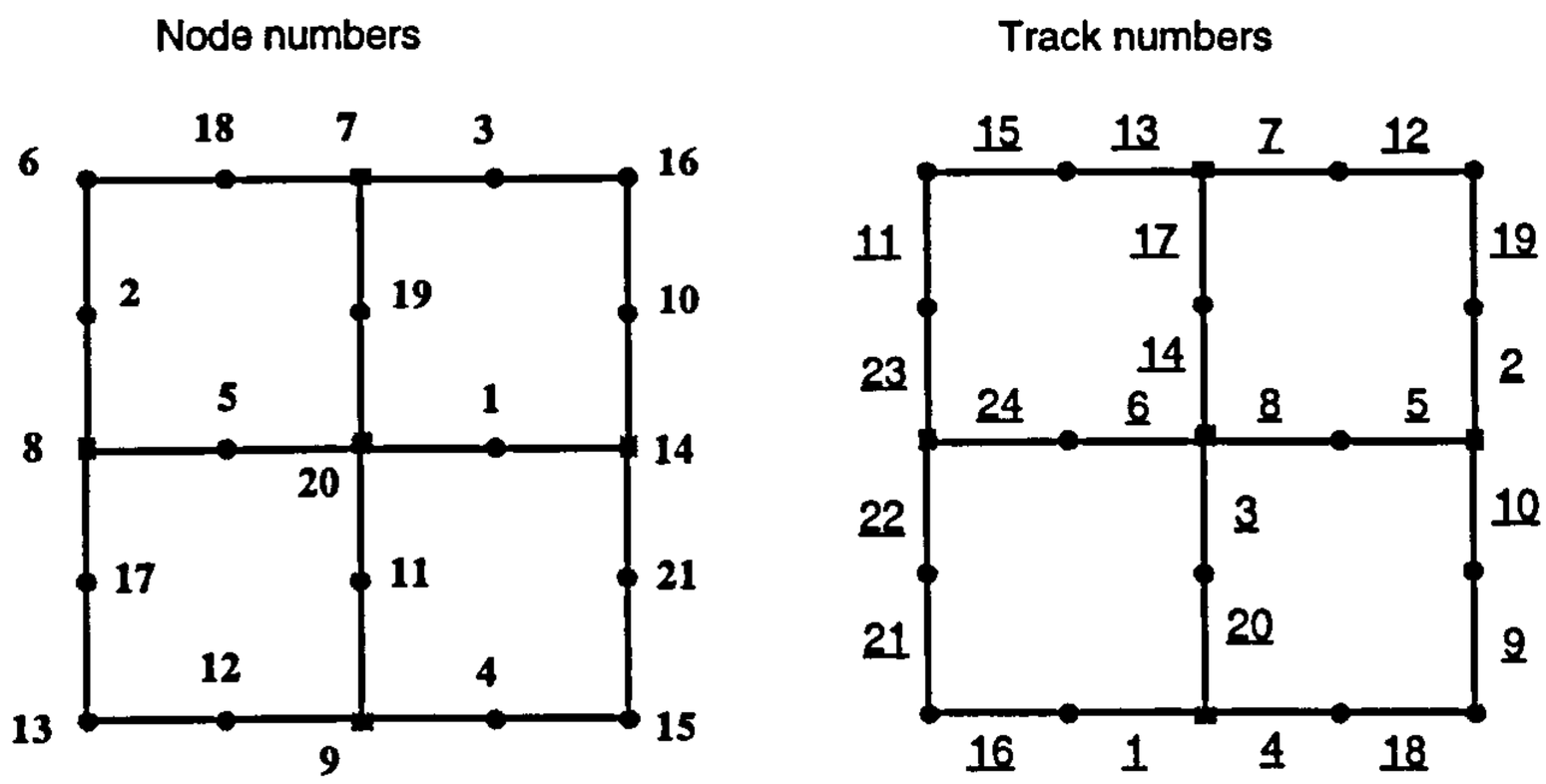


Figure 7- 3: Random numbering (LAYOUT_2) of nodes and tracks in the physical layout.

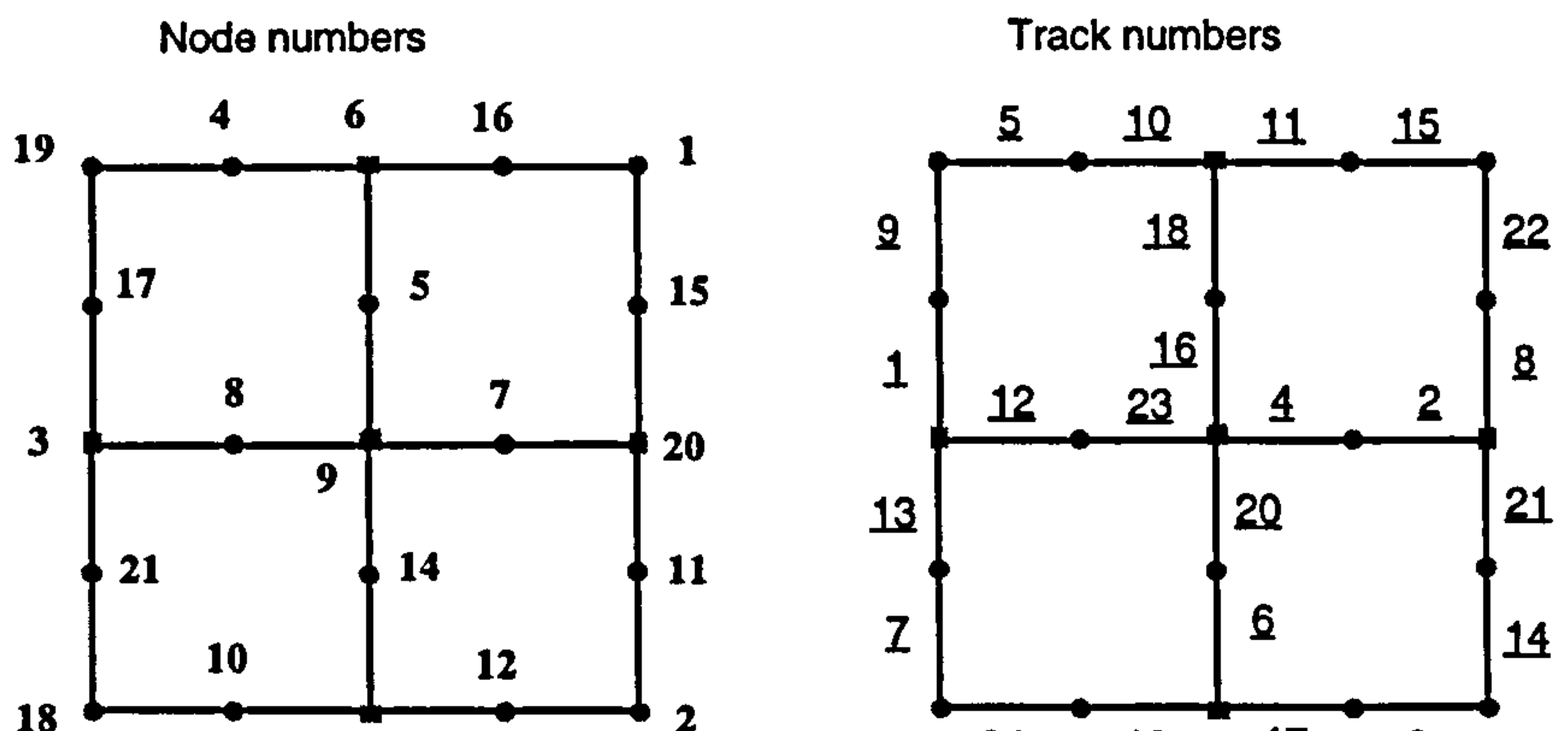


Figure 7- 4: Random numbering (LAYOUT_4) of nodes and tracks in the physical layout.

Table 7- 9: Results (correct shorter paths) obtained with four different order numbers for the elements in the physical layout. Only the movement direction vector derived from the ANN was used.

ANN: input=34; output=2; hidden layer=20; learning rate=0.3; moment.term=0.6; epoch=50; 100000 iterations ("stop" decision when modulus of movement direction vector < than 0.1)				
Testing data separated in:	Training data cases (forward and reverse) converted to ordering number as in: "Layout_1"	"Layout_2"	"Layout_3"	"Layout_4"
	(%)	(%)	(%)	(%)
Testing1_F	45	28	59	43
Testing1_R	46	33	52	40
Testing2_F	34	37	66	44
Testing2_R	46	34	42	44

Table 7- 10: Results (correct shorter paths) obtained with two different sequential order numbers for the elements in the physical layout. Using the interpretation of the ANN output to prevent movement into non-existent or unavailable tracks.

ANN: input=34; output=2; hidden layer=20; learning rate=0.3; moment.term=0.6; epoch=50; 100000 iterations ("stop" decision when modulus of movement direction vector < than 0.1)		
Testing data separated in:	Training data cases (forward and reverse) converted to ordering number as in: "Layout_1"	"Layout_3"
	(%)	(%)
Testing1_F	62.2	64.8
Testing1_R	55.9	60.3
Testing2_F	48.7	73.2
Testing2_R	61	56.1

7.4 Summary of results/conclusions

At this stage an ANN solution based on a step-by-step approach (i.e. Representation 2) was implemented which is able to find 42% to 66% of correct solution paths on the two testing data sets used. These figures rise to 56% and 73% when a simple heuristic is used to prevent moving into impossible tracks (Table 7- 9, Table 7- 10). These results were obtained with a limited number of experiments and more exhaustive tests could be performed to try and optimize the representations proposed and ANNs tested.

However it seems clear that success rate near 100% in terms of correct complete paths is a difficult task for the ANNs proposed and tested. One of the reasons is related to the difficulty in finding a representation which unambiguously identifies each layout state. Another aspect is the need to use an indirect measure of the ANN performance in relation to the objectives of implementing possible paths. One other issue can be related to the volume of training data used. A reduced number of training cases was generated and used in relation to the total number of possible combinations that can be associated either with the problem formulation or even more with the representations and encodings used in the ANNs proposed. Using a larger number of cases for training it is possible that the "deadlock" situations or movement into an impossible to use track could be better distinguished and identified by the ANNs. We have also obtained an indication (see Table 7- 9, Table 7- 10) that these situations can be improved by the use of simple heuristics to complement the ANNs solutions. It may therefore be questioned which of the approaches should be followed in order to significantly increase the performances already obtained. In order to decide on this issue it must not be forgotten that the ultimate objective is to develop a solution for the multiple vehicle case and the significance of the levels of performance of any solution have to be related to its use in a multiple vehicle case. It is envisaged that the use of additional simple heuristics (i.e. preventing a vehicle from moving backwards to avoid deadlock situations, and recognising the destination node when it passes by) can improve the results already obtained in a single vehicle case even if not always producing optimum paths. It is also acknowledged that the use of such rules have to be tested in a multi-vehicle situation. Furthermore when multiple vehicles are involved it may be expected that not all vehicles will be allowed to implement their own individual "optimum" solutions. These aspects taken together are the basis on

which we proceed to the multiple vehicle case based on the results obtained so far with the step-by-step approach (i.e. Representation 2). The approach based on 'complete paths', i.e. Representation 1 will no longer be pursued.

Chapter 8 Strategy for the Application of Backpropagation to the Multi-Vehicle Case

The solution envisaged for the multiple vehicle case is based on the Artificial Neural Network (ANN) solution previously developed for the single vehicle case. This chapter describes the various alternatives developed and the strategies chosen for performance comparison.

The ANN for the single vehicle case consists of a BP type ANN which determines the movement direction out of a node when given information relating to the current node, target node, and the status (i.e. available, not available) of the tracks. In a multiple vehicle case the same ANN is used iteratively by each vehicle, one at a time, with the track status information representing the movement intentions of the vehicles which had already decided about their movement. This basic approach for the multiple vehicle case was employed to investigate "look-ahead" strategies which consist of analysing the implications of the movement of a vehicle with respect to the movement of other vehicles.

Evaluation of the performance of the solutions developed was effected by comparison with two global heuristic rules [Ref. 40]: the "greedy policy" and the "benevolent counterclockwise flow policy"; and also solutions based on the use of pseudo-random number generators to specify the next movement direction for each vehicle. The "greedy policy" and the "benevolent counterclockwise flow policy" represent rules which are applied equally to every vehicle when a routing decision has to be made. While the "greedy policy" is based on always selecting the available movement direction which leads to a location closer to the destination, the "benevolent counterclockwise flow policy" uses a process in which the movement alternatives are evaluated according to each ones location in a pre-defined

referential system. This referential system is centred on the vehicles's current location and its quadrants numbered in a counterclockwise direction starting from the one which contains the location of the destination node. Each movement alternative is assigned the number of the quadrant which contains its location. The movement alternative selected will be the one which has a lower quadrant number and is available. Therefore the main idea behind this policy is to induce a consistent direction for the traffic flow, which in this case is counterclockwise but it should not be no different if a clockwise direction is selected.

The two global heuristic rules were extended with the application of the "look-ahead" strategies developed for the ANN based solutions. All together, including the random choices, 14 cases have been proposed for testing and evaluation.

8.1 Artificial Neural Network based solutions

The ANN solution for the single vehicle case, which was chosen for use in the multiple vehicle case, consists of a backpropagation type ANN with 34 input neurons, an hidden layer, and 2 neurons in the output layer. The neurons in the input layer represent information about the track status (i.e. available or not available), and the origin and destination nodes. The two output neurons represent the co-ordinates of a vector which is used to indicate the movement direction out of the current node. A schematic representation of the ANN is represented in Figure 8- 1. Figure 8- 2 represents the interpretation of the two output elements as the co-ordinates of a vector which indicates the direction of the next move by transforming all possible directions in the plane into one of the five alternatives: UP, RIGHT, DOWN, LEFT, and STOP. The STOP alternative is defined when the modulus of the output vector is below a specified minimum value. In the experiments performed so far this value was set to zero (0). The exact encoding and conditions of training and testing were described previously in Chapter 7 for the single vehicle case.

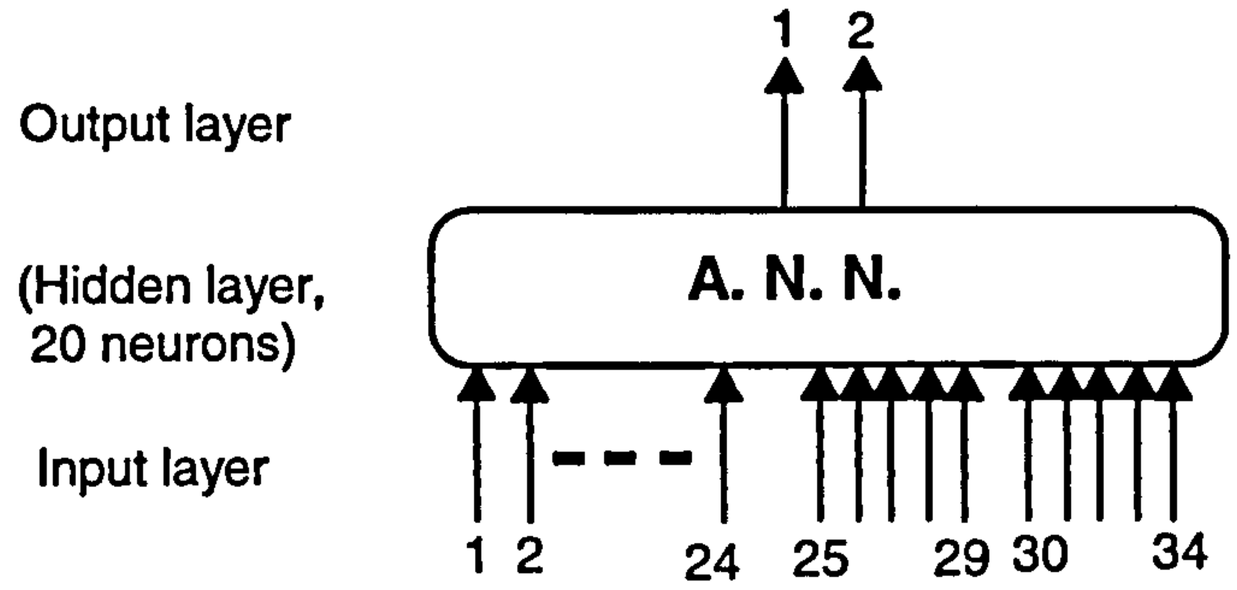


Figure 8- 1: Backpropagation artificial neural network representation.

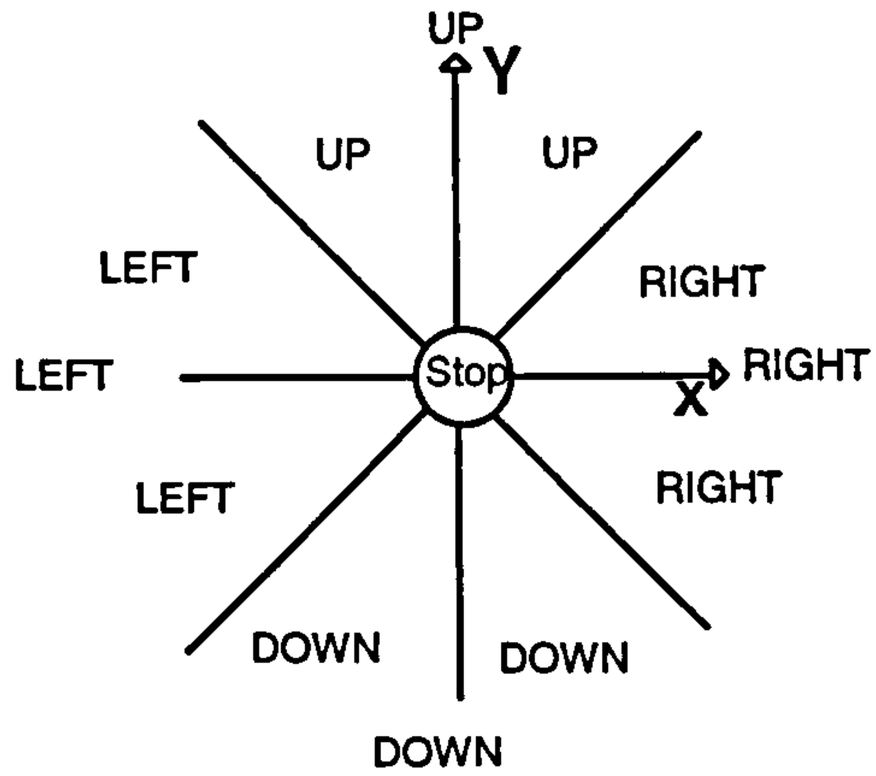


Figure 8- 2: Interpretation of neural network output.

This ANN was trained with 1970 optimum (shortest) paths and in the paths tested (304) it could find a complete, optimum, path in 42 to 66 % of those cases (Chapter 7). In the rest of the cases the majority consisted of an indication from the ANN to move onto a track that does not exist in the physical layout or is unavailable. In a real system it would be necessary to guarantee that all decisions made are possible, even if they do not lead to the optimum solution. Using simple rules together with the ANN solution ("hybrid solution") it is possible to achieve this, as is described in the following section.

8.1.1 Hybrid solution

The implementation of the "hybrid" solution was done with the aid of a program written in the "C" language which provides an interface with the neural network software. The program reads the ANN

output, analyses it, makes decisions and updates the ANN input based on these decisions. Figure 8- 3 shows a schematic representation of the "hybrid" solution.

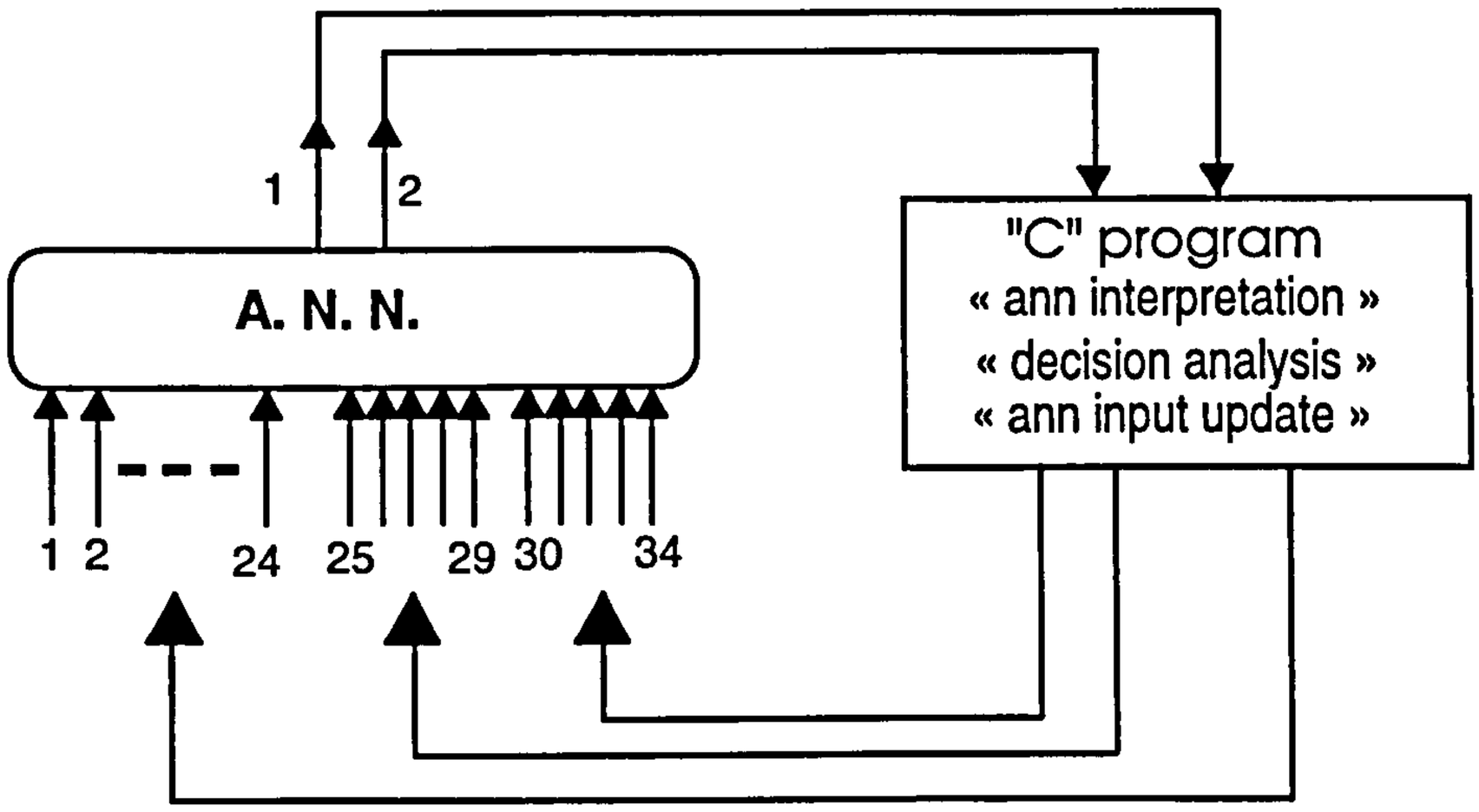


Figure 8- 3: Hybrid solution.

Such a scheme is useful not only to improve the utility of the ANN solution for the single vehicle case with unavailable pieces of tracks as already seen in Chapter 7, but it seems even more necessary to implement a solution for the multiple vehicle case because it provides a way to account for the interaction of the vehicle's movements that result from decisions based on the ANN. Since it is important to improve the performance of the ANN in a single vehicle case with blocked tracks, it seems at least as important to evaluate what the percentage of good solutions obtained represent in a multiple vehicle case where the dynamics of track status, due to the movements of the vehicles, assumes greater importance.

The interface ("C" program in Figure 8- 3) to the ANN was then developed to allow the implementation of specific interpretations of the ANN output vector, the update of the ANN input, and also the consideration of specific "look-ahead" strategies. This led to the implementation and testing of four different strategies based on an ANN for the multiple vehicle case. Before describing these four strategies in the following sections, the main parts of the program developed will be described as represented by the flowcharts in Figure 8- 4 to Figure 8- 8.

8.1.1.1 Program development

The method provided by the software package used to develop the ANN models to present data to, and access data from the network is through the use of a procedure written using the "C" program language, that is attached to the network. This program can only be executed from within the main program where it is possible to select it as an option in the specification of the learning or testing procedures. The communication between the user written program and the main program is achieved through the use of static data pointers. A listing of the code used is presented in Appendix C.

The main phases of the program implemented using these facilities are represented in Figure 8- 4, where letters 'A', 'B', 'C', and 'D' are used to identify the main parts, and the boxes bold-faced represent parts of the program which are described in more detail in Figure 8- 5 to Figure 8- 8. It consists basically of using the possibilities of access to the output, and presenting information to the input of the network during the recall or test phases of the network; after the output access and before the input update, an evaluation and analysis is performed according to the strategies defined. Due to restrictions on size of the programs attached to the main program it was sometimes necessary to use different programs which implement the modifications required by different strategies. Using the identification letters in Figure 8- 4 the description of the program is presented next.

<u>Phase A</u> ⇒ Represents the initial phase of the program where a number of parameters specifying the conditions of the simulation run have to be defined. These parameters are listed in the input box of Figure 8- 4 and are basically related with the specification of a number of vehicles in the system, a number of time steps for the system to run continuously, and the selection of strategies defined to evaluate, analyse and take decisions for each vehicle. The duration of the run is related to the number of network recall iterations and the number of vehicles in the system. Pseudo-random number generators are used and the specification of a seed number was introduced to allow the identification and selection of the series of pseudo-random numbers used.

The next parameter, the number of time steps for deadlock reference is related with the solution used to deal with deadlock situations. In a system running continuously it becomes necessary to identify

possible deadlock situations so they can be accounted for and overcome in order for the system to proceed. As represented in Figure 8- 8 the solution used to prevent the system from remaining in a deadlock situation for the duration of the test, was to limit the number of time steps allowed for each vehicle to complete a path. The number chosen for this limit was 200 which was long enough for a vehicle to accomplish its transport task, and after which even if it was not in an indefinitely deadlock situation it would not be of much interest in a practical situation. In an attempt to correct a possible deadlock situation a value was specified for the number of steps each vehicle takes before the system gives particular attention to that vehicle. This value is the number of time steps for deadlock reference, and the particular attention consist of temporarily defining a new destination node for the vehicles exceeding the deadlock reference time, and then re-assigning it the original destination node number.

The next parameter is the number of time steps that a vehicle could be waiting at a node before the track used by that vehicle, to get to that node, would be reset. This implied that a moving backwards alternative would be considered no differently than other movement alternatives when a decision was made as to what track to choose in the next movement.

The next two parameters are user specified, one is related to the selection of the alternative strategies implemented to decide and analyse the implications of the movements of the various vehicles, and the other is related to the interpretation of the ANN output. Three different alternative "look-ahead" strategies (complete, limited, and none) (Figure 8- 7) and an indication as to whether or not a set of rules applied to extract a movement decision from the ANN output (Figure 8- 5 and Figure 8- 6) could be selected. These alternatives are described in detail in the next four sections corresponding to the different solutions tested.

The last parameter provides the user with the alternative to use the two ANN output values in the system or replace them by two pseudo-random numbers making it possible to obtain results and compare the use of ANN with purely pseudo-random generators, when all other conditions are the same.

Following the specification of these initial parameters the system then proceeds with the generation of the starting cases for each vehicle. The initial origin and destination node numbers for each vehicle would be generated using pseudo-random generators. A new case for each vehicle would consist of the vehicle's current node as the origin node and a destination node selected using pseudo-random generators. A specific destination node can only be assigned to one vehicle. The solutions developed consisted of deciding what a vehicle should do, one vehicle at a time. In a multiple vehicle case, this requires assigning priorities to the vehicles. In the experiments done so far the initial priorities were arbitrarily pre-defined, but at each move the priorities were rotated to ensure that all vehicles would have overall equal priorities.

Phase B ⇒ With each vehicle assigned a location node in the physical layout, a destination node, and a priority, the system now proceeds with the iterative usage of the ANN by each vehicle, one at a time according to its priorities. Starting with the higher priority vehicle first, the ANN input will be updated to include the status of the layout (all available for the first vehicle) and the identification of the vehicle's current and destination nodes. The ANN output will then be analysed and interpreted according to the two possibilities described in Figure 8- 5 and Figure 8- 6. Basically they both consist of converting the two ANN output elements into the co-ordinates of a vector representing the moving direction. While in Figure 8- 5 ("with heuristic") a set of rules is used to test in turn other moving possibilities if the first one is not available, in Figure 8- 6 ("without heuristic") only the moving direction obtained from the ANN is considered. One of the five possible movement decisions for the vehicle, i.e. move right, down, left, up, or stop is then obtained and kept as an intention of movement for the vehicle. The layout state is then updated based on the intention of the previous vehicle(s) to occupy a track and on the next vehicle's current location and destination nodes. This process is repeated until all vehicles have selected their intended next move.

Phase C ⇒ It is now necessary for the system to analyse the movement intentions of each vehicle and decide whether it will be implemented or not. Each vehicle will have at most two possibilities either move along its chosen direction or stop at the current node, unless its intended decision is to stop in which case it will be allocated its current node. The decision for each vehicle,

whether to move or not, can therefore be made taking into consideration, or not, the implications its move will have on movements of the others vehicles. Three alternatives are implemented (Figure 8-7), a complete "look-ahead" strategy, a limited "look-ahead" strategy, and not using a "look-ahead" strategy. The complete look-ahead strategy, which analyses the vehicle's intention to move along a selected track will lead to a node occupied by either vehicle, or wanted by other vehicle(s) movement to its full extent. In case the node is not occupied but wanted by other vehicles, the other vehicles will be stopped at their current nodes to allow the movement of the first vehicle. If the node is occupied then it will be checked if the occupying vehicle wants to move out of the node and if that is possible, i.e. the same procedure is applied to that vehicle until a decision is made whether the movements are possible or not. The vehicles are analysed first by allocating the vehicles with a "stop" intention, then the vehicles which reach the destination nodes with the intended move, and then the remaining vehicles in order of their assigned priorities. The limited "look-ahead" strategy differs from the complete "look-ahead" strategy in that only two nodes ahead of the current node of the vehicle in analysis are checked. The alternative which does not use a "look-ahead" strategy consists of assigning the vehicles its intentions to move along the directions selected in phase B. These directions will always be possible without further checks for collisions at nodes because they are taken into consideration during phase B when "no look-ahead" strategy is selected. The characterization of these three strategies is completed in the following sections describing the solutions tested.

<u>Phase D</u> ⇒ At this stage the paths are evaluated in order to account for and deal with their current situations (Figure 8- 8) such as vehicles terminating their paths, or deadlock type situations. One possibility is for vehicles which terminate their paths to be assigned a new destination node to link its current location with a new path. Another possibility is for vehicles which have used up the 200 maximum number of time steps allowed and must therefore be accounted for as deadlock situations (i.e. paths not solved), to be assigned new destinations. Two other situations must be detected, one represents vehicles whose paths are overcoming the deadlock reference time and will be assigned a temporary new destination node, and the other represents vehicles which have already been assigned a temporary destination node for the maximum time allowed and will be re-assigned the original destination nodes.

For each vehicle, having being defined a current node and a destination node, the priorities will be rotated to ensure all vehicles will have overall equal priorities. After updating the results files with the current values, the process is repeated from phase B until the number of iterations specified is completed.

To test the functionality of the program each function or routine was tested independently and also during the execution of the program with messages alerting us to impossible solutions such as having more than one vehicle at a node.

The following four sections represent the alternatives tested based on this program and which consist of using the following selections for the ANN output interpretation and "look-ahead" strategy:

- ANN with "heuristic" and complete "look-ahead" strategy;
- ANN with "heuristic" and limited "look-ahead" strategy;
- ANN with "heuristic" and no "look-ahead" strategy;
- ANN only, i.e. without "heuristic", but complete "look-ahead" strategy.

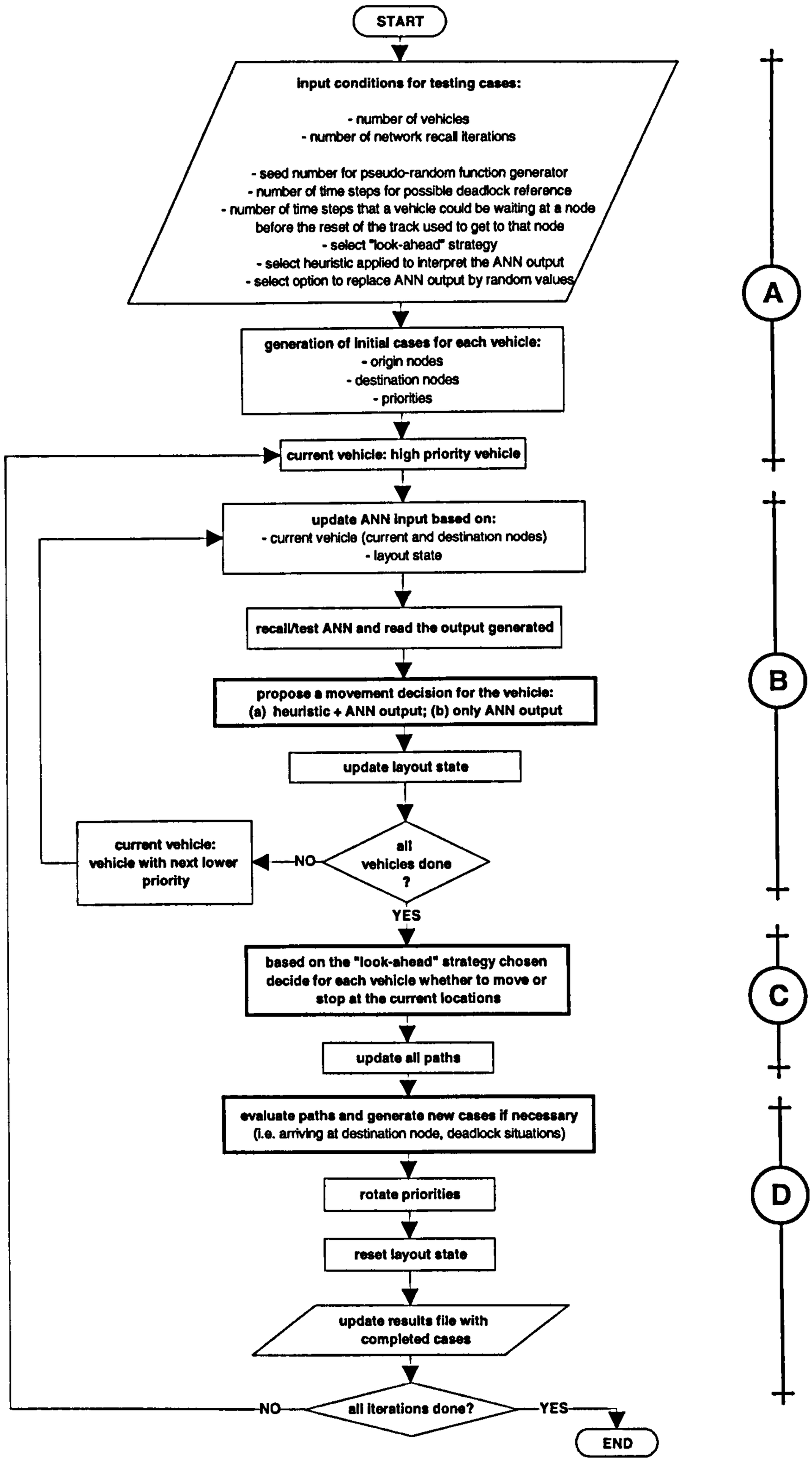


Figure 8- 4: Interface program used to implement "hybrid solutions"; identification of main phases.

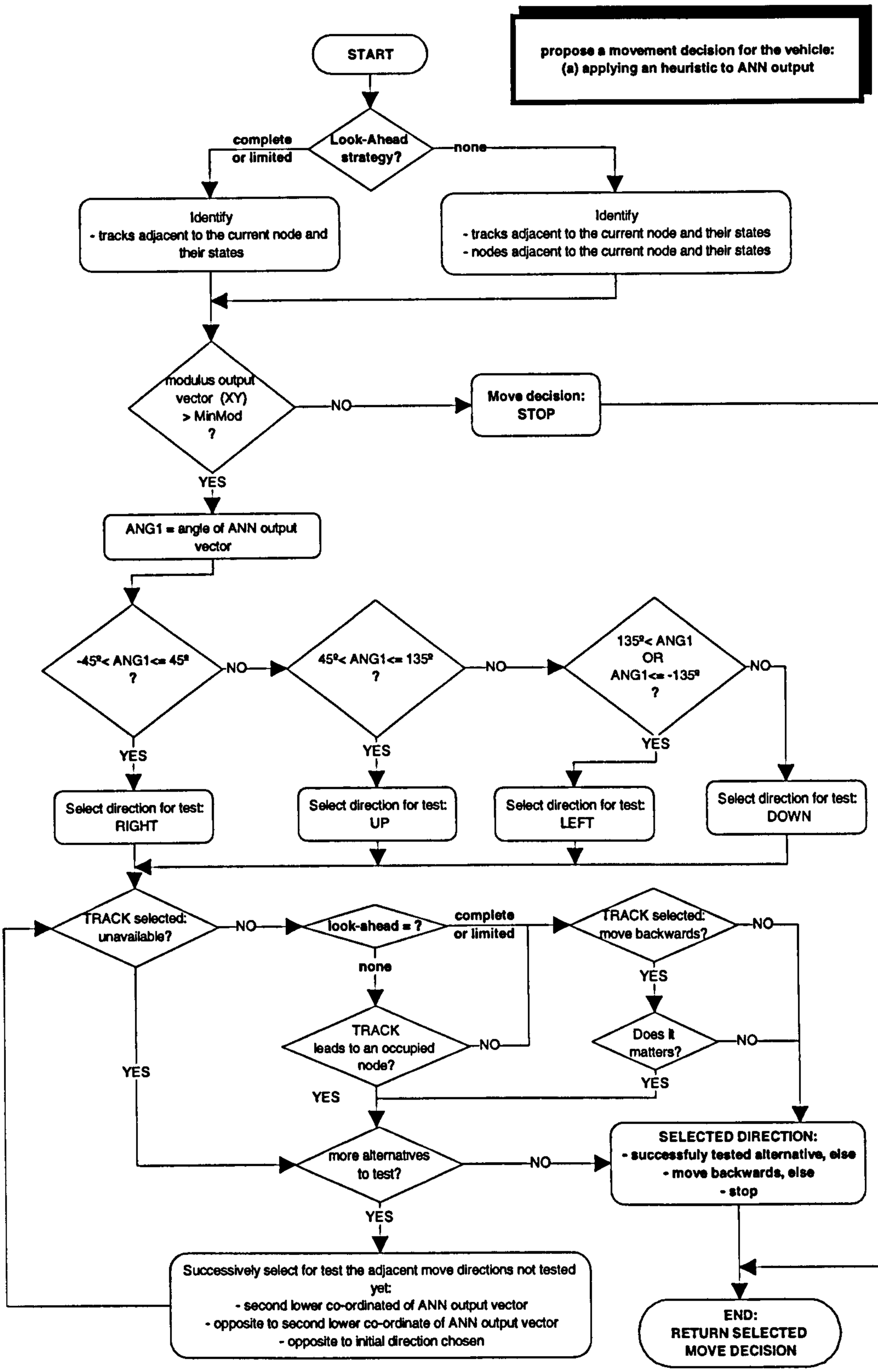


Figure 8- 5: Interface program used to implement "hybrid solutions"; propose movement decision (with heuristic applied to the ANN output).

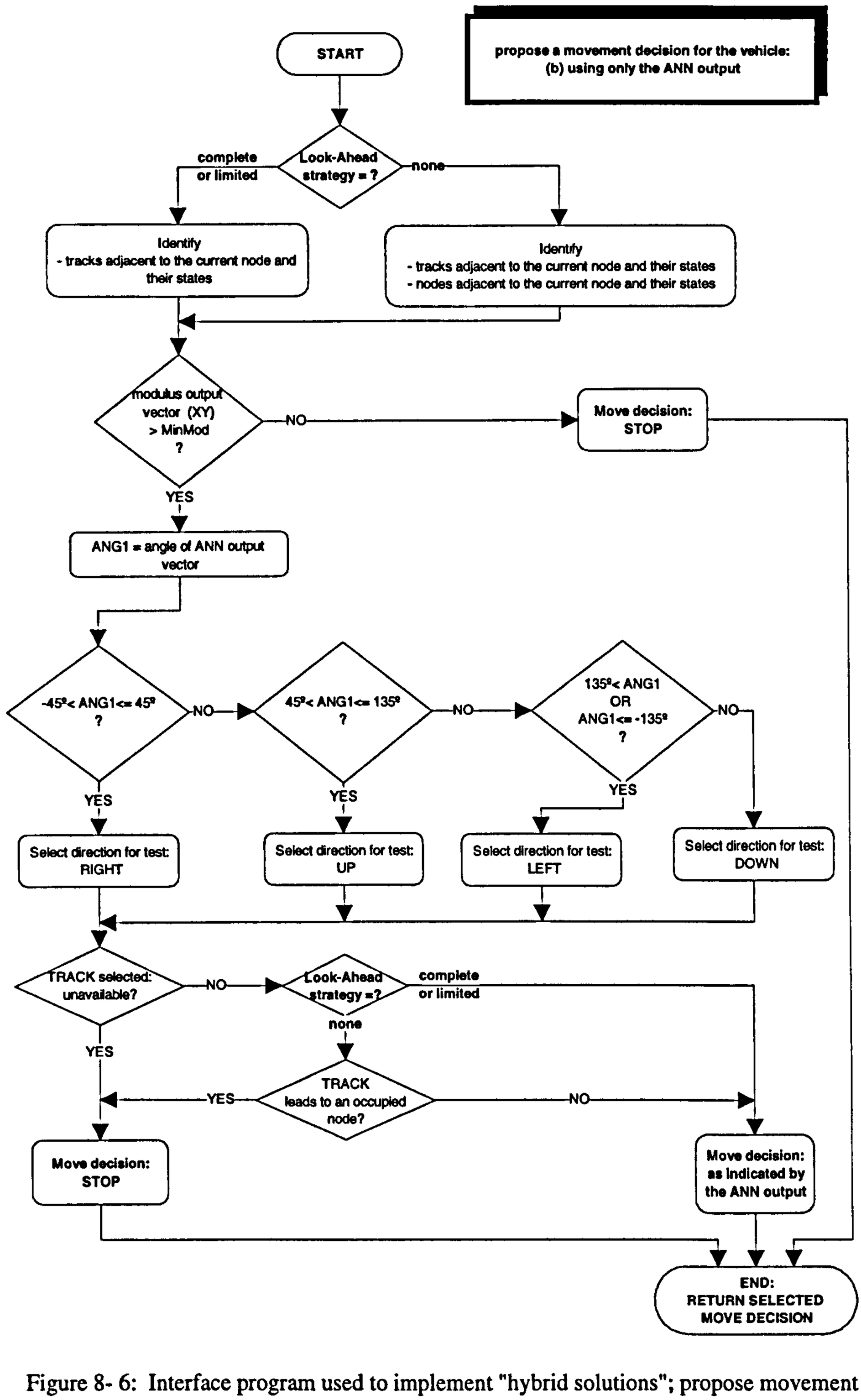


Figure 8- 6: Interface program used to implement "hybrid solutions"; propose movement decision (considering only the ANN output, if not possible then stop).

based on the "look-ahead" strategy chosen decide for each vehicle whether to move or stop at the current location

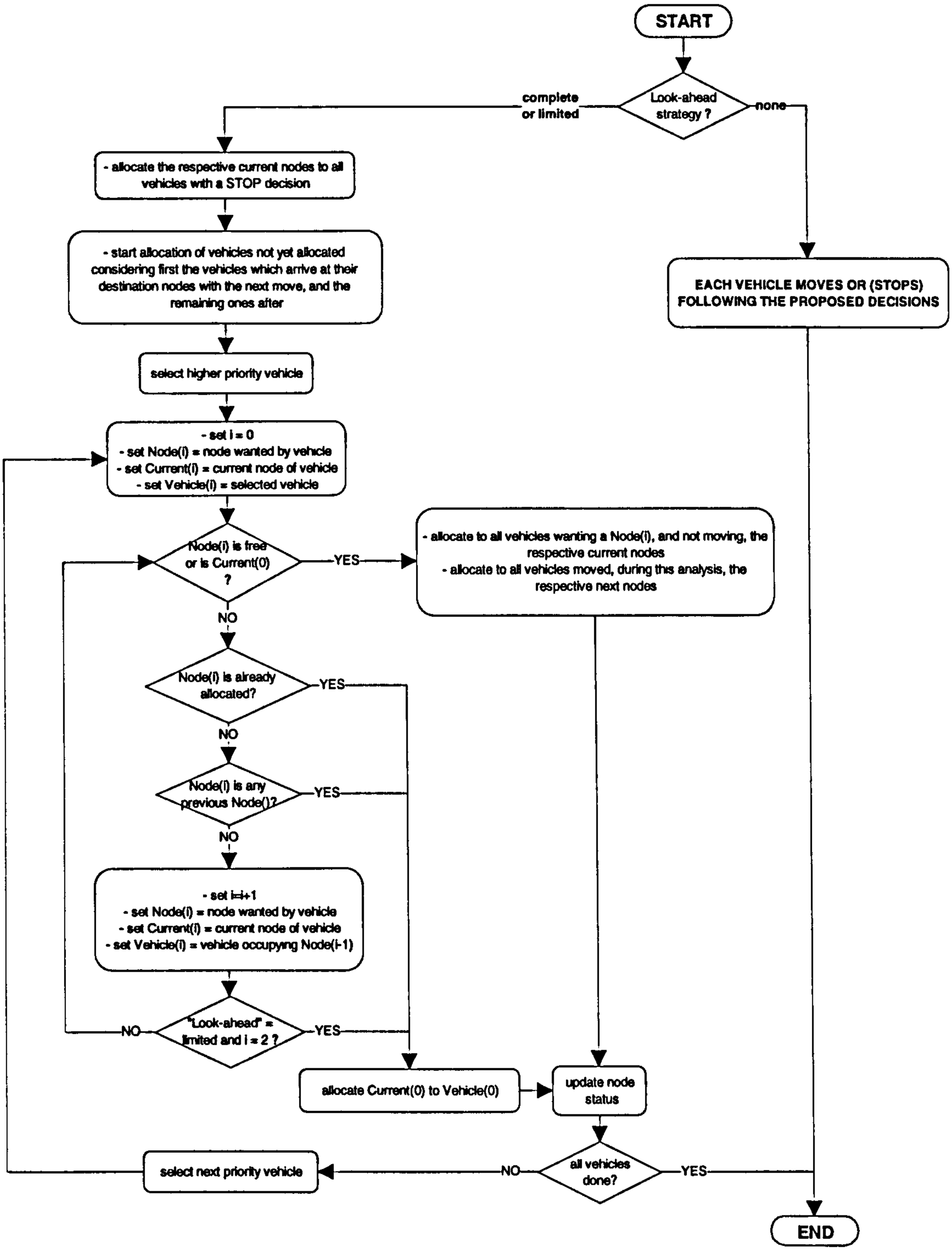


Figure 8- 7: Interface program used to implement "hybrid solutions"; decide whether to implement or not the movement previously proposed for each vehicle (with complete, limited, or no "look-ahead").

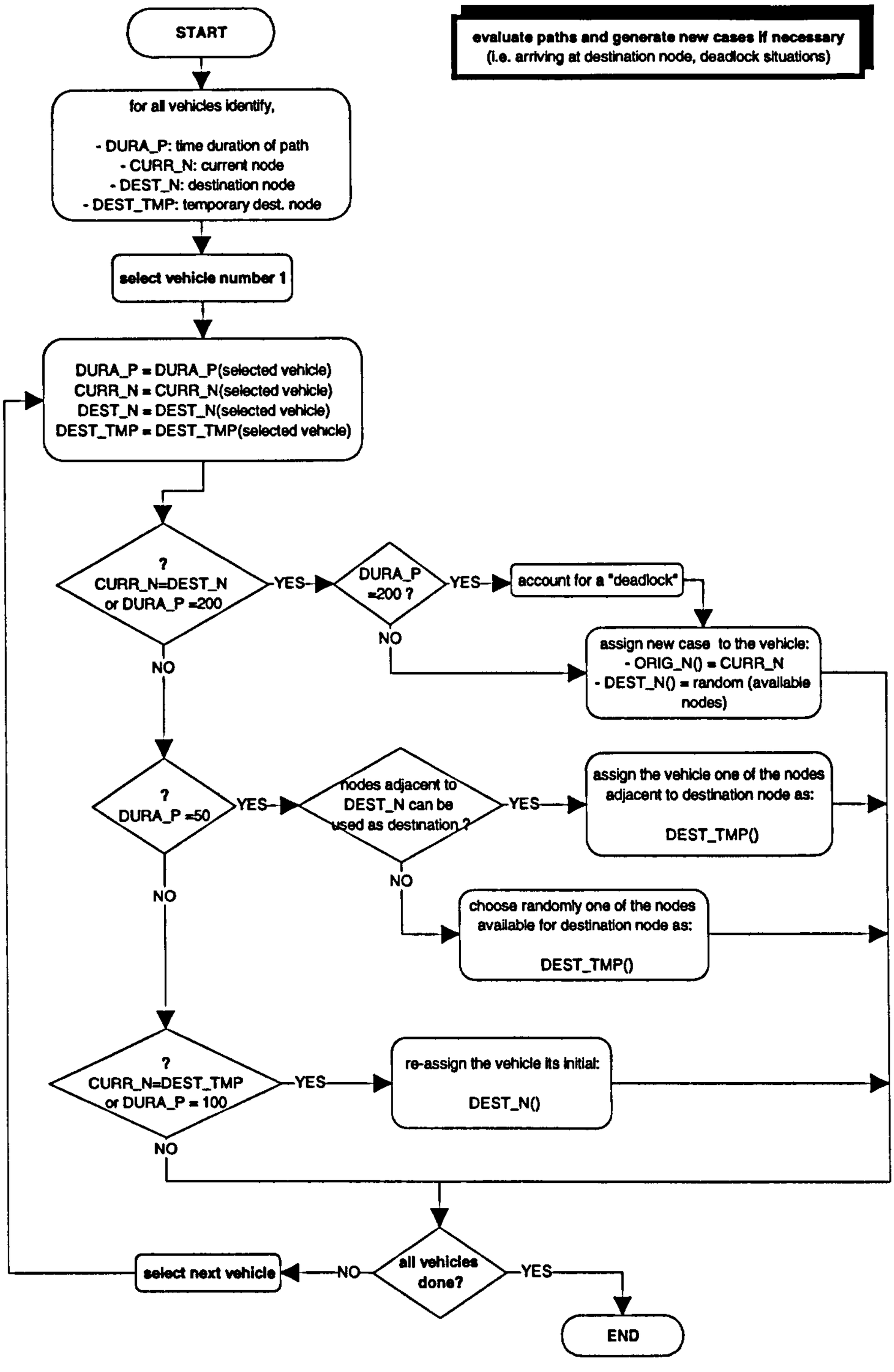


Figure 8- 8: Interface program used to implement "hybrid solutions"; evaluate paths and generate cases if necessary.

8.1.2 ANN with heuristic and complete look-ahead strategy

In this case, first the solution indicated by the output vector of the ANN is complemented with rules that basically prevent a decision to move to a unavailable track being taken; and also the vehicle does not stop unless no other movement alternative is possible. The exact rules implemented are described as follows (see Figure 8- 9, Figure 8- 5 and also Figure 8- 7):

- first choose the direction indicated by the output vector of the ANN and test to see if it is possible;

- second, if the first alternative is not possible, then follow the direction most nearly corresponding to the ANN output element that it is possible to implement;

- third, if neither the first nor the second alternatives are possible then select the direction opposite to the direction of the ANN output vector co-ordinate considered in the second alternative;

- fourth, if a possible direction has not yet been found, choose the remaining possibility;

- fifth, if none of the previous alternatives are possible then stop at current location.

The possibility is also considered of having repeated tracks and nodes in the path. Moreover going back to the previous track/node had to be the last possible alternative but it is possible to specify a time limit after which no distinction is made between tracks being used or not. The STOP at current node alternative in the cases tested was only considered if all directions out of the node were blocked because a value of zero (0) was defined for the minimum threshold value of the output vector modulus.

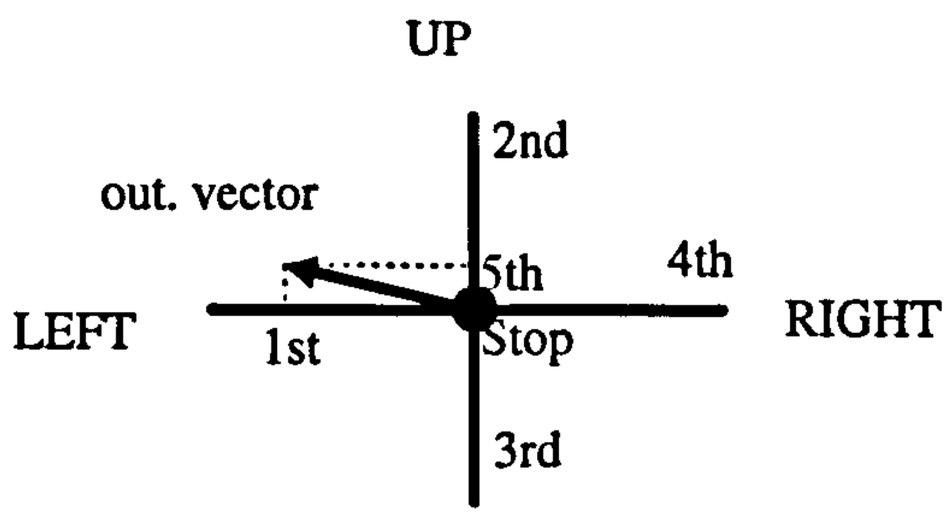


Figure 8- 9: Sequence of decisions depending on the availability of the adjacent tracks and an example of the ANN output vector.

Using this first set of rules, each vehicle, one at a time according to the current priorities, makes its decision about what movement it desires to implement next. Once all vehicles have a decision the system analyses whether there will be conflicts, i.e. more than one vehicle desires to move to the same next node. A "look-ahead" strategy is used which consists basically of preventing these conflicts by assigning the nodes to only one vehicle while the other(s) who want the same node will wait at their current nodes. Each vehicle will therefore have only two possibilities: either move to its chosen track (or next node) or stop at its current node. The exact rules implemented are:

- first give priority to a vehicle which will arrive at a target node if its movement intention is implemented. If more than one vehicle is in that situation, consider the priorities already assigned to the vehicles at the start of that iteration to establish a priority order between those vehicles;

- second consider the priorities already assigned for the remaining vehicles, to decide which ones will have priority over the others, and therefore move, or remain at the current location, whenever conflicts exist in desired next node locations;

- both these rules have to consider the implications of a decision (whether or not to move a vehicle) in all other vehicles movement. These implications were investigated considering all the possible implications of the movement of one vehicle on all other existing vehicles.

An example illustrating this "look-ahead" strategy is represented in Figure 8- 10 and is described next:

- is the node (nB) wanted by the vehicle (A) either free, occupied, or do other vehicles want to go to that node?

- if node (nB) is free, or if it is free but wanted by other vehicles, then move vehicle (A) to node (nB), release node (nA), and stop the other vehicles who wanted node (nB),

- if node (nB) is occupied can the occupying vehicle (i.e. B) move out to let it be free?

- if vehicle (B) cannot move out because node (nC) was already assigned definitely to vehicle (C), then stop vehicle (A) definitely at node (nA),

- if vehicle (C) is still not definitely assigned to a node can it move out to its desired location, i.e. node (nE)?

- repeat the process until all vehicle movements, or stops, needed to make initial vehicles decision possible are investigated and solved according to the rules presented above.

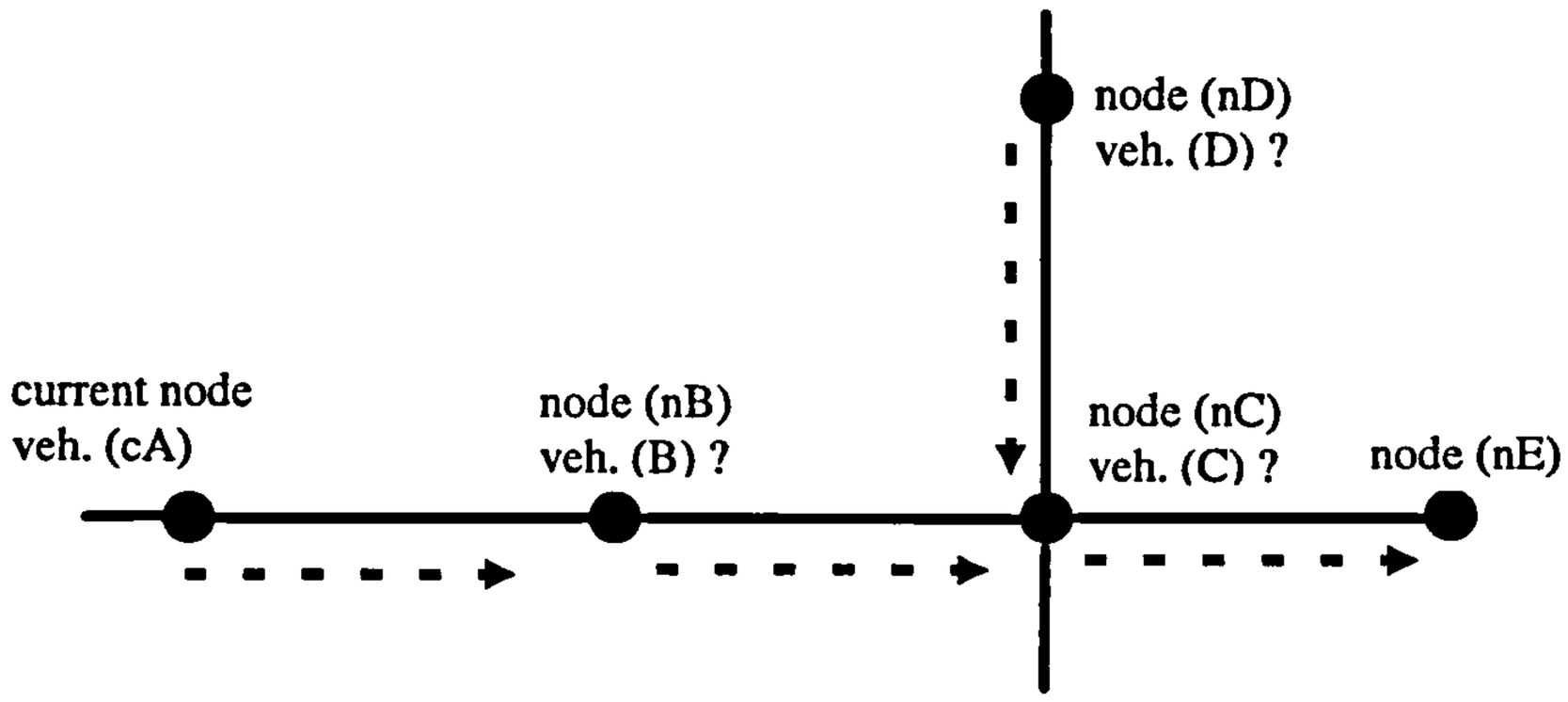


Figure 8- 10: Example of look-ahead analysis implications.

The philosophy behind this strategy was to test a look-ahead type solution and the benefits it could bring compared to other more local and distributed approaches. If the look-ahead analysis has to consider all possible implications of all possible movements of a vehicle in respect of all other vehicles then its complexity increases very rapidly in line with an increase in the number of vehicles in the system. For example if all movement alternatives for each vehicle were taken into account, the number of combinations possible would increase significantly with an increase in the number of vehicles. The approach would then suffer from the same scalability limitations as other solutions based on exhaustive search type algorithms.

However in the "look-ahead" strategies implemented so far only two alternatives are considered for each vehicle, either move along its selected direction or stop at the current node. In this case the complexity of the "look-ahead" strategy can be directly related to the number of vehicles in the system: the number of operations involved when a vehicle is investigating the possibility of implementing its next move will be maximum when all vehicles are successively affected by that initial movement, and will consist of the number of operations performed to investigate the node where the first vehicle intends to move, repeated by each vehicle as it becomes the vehicle moving into an adjacent node. As such it can be assumed that the complexity increases linearly with the number of vehicles in the system. Nevertheless it was decided to implement a more limited version of the "look-ahead" strategy which is described next and in which the maximum complexity involved is limited independently of the number of vehicles in the system.

8.1.3 ANN with heuristic and limited look-ahead strategy

This solution is basically identical to that presented in (section 8.1.2) except in the extent of the implications of each vehicle's movement on the other vehicles. It uses exactly the same heuristic to interpret and complement the ANN output vector. The ANN is used iteratively by each vehicle according to its assigned priority, and all the vehicles keep their decisions as an intention to move to the next selected node. The "look-ahead" strategy rules are implemented in exactly the same way except that in the complete "look-ahead" strategy the implication of a vehicle move in the other vehicles is considered in all its extension, in the limited "look-ahead" strategy the extension of these implications is limited to two nodes away from the vehicles current node. In the limited "look-ahead" strategy if there is a vehicle in the node desired by the first vehicle, and for the second vehicle to move out of that node it was necessary to move a third vehicle, then no further implications are investigated and the initial vehicle stays at its current node. An example of this limited "look-ahead" strategy is represented in Figure 8- 11 and is described as follows:

- is the node (nB) wanted by the vehicle (A) either free, occupied, or do other vehicles want to go to that node?

- if node (nB) is free, or if it is free but wanted by other vehicles, then move vehicle (A) to node (nB), release node (nA), and stop the other vehicles which wanted node (nB),

- if node (nB) is occupied can the occupying vehicle (i.e. B) move out to let it be free?

- if there is a vehicle in (nC) then vehicle A will stop at its current node (nA). However if node (nC) is currently free but other vehicles also want to go there then they will be stopped at their current locations to make vehicle movements possible (veh.A and veh.B). If the movement of a vehicle affects more than two nodes ahead of the desired node then the current vehicle stops, remaining at its current location.

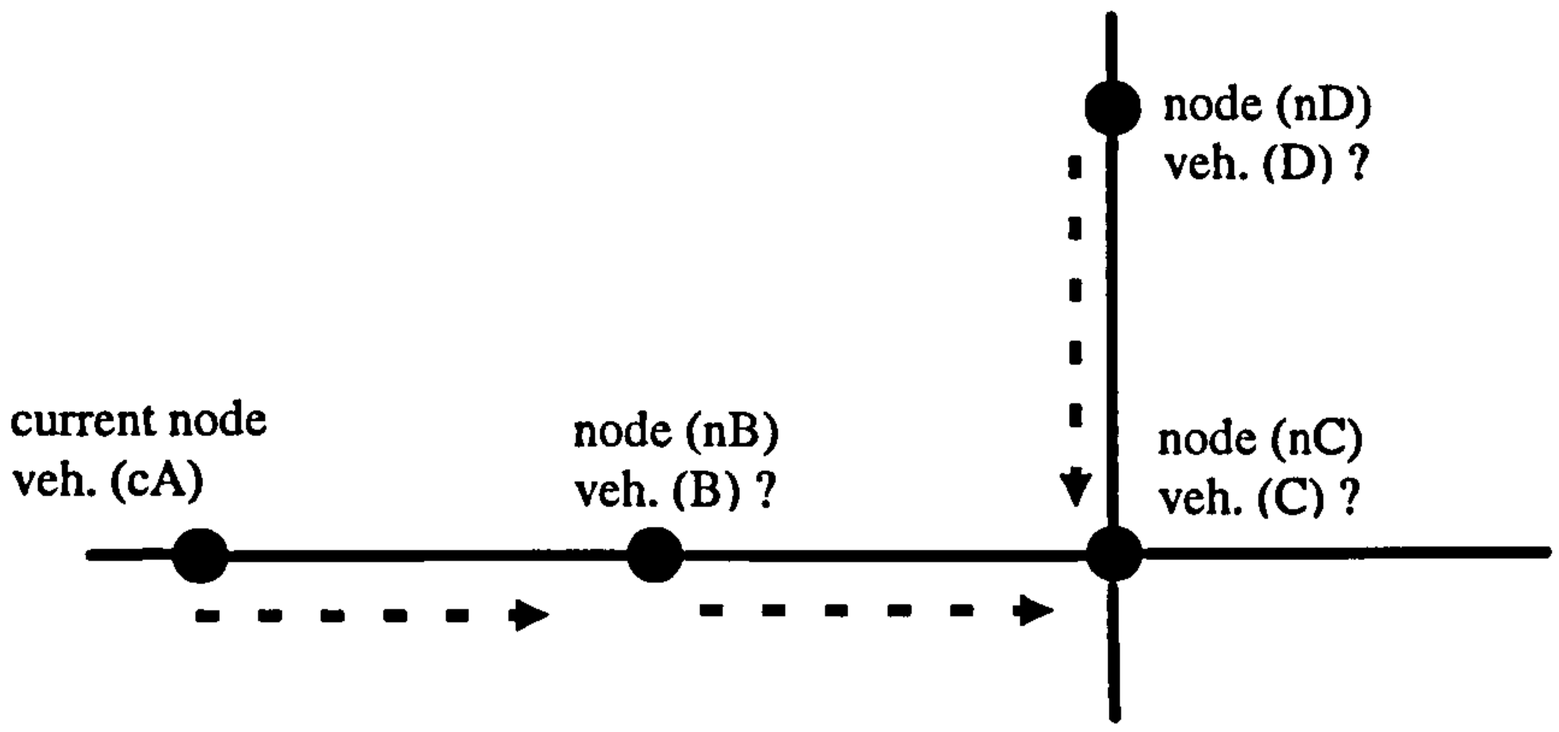


Figure 8- 11: Example of limited "look-ahead" analysis implications.

8.1.4 ANN with heuristic and without look-ahead strategy

This approach using the ANN plus heuristic in a multiple vehicle case differs from the previous strategy in that no look-ahead or analysis of movement implications is made. The only heuristic used is to complement the ANN output vector as described in Figure 8- 9. The decisions about movement for each vehicle are defined almost exactly as in a single vehicle case, except that:

- each vehicle uses the ANN with heuristic, in the sequence defined by its currently assigned priority,

- the tracks set as unavailable in the ANN input vector represents the tracks already selected by the vehicles with higher priority,

- the simple rules used with the ANN solution take into consideration the movement of all the vehicles that have already made a decision, to avoid conflicts in the nodes wanted by each vehicle,

- each vehicle's moves are decided sequentially according to priority, but the movements are executed at the same time.

The implications of each vehicle movement is now only considered through the updating of the track status in the ANN input vector. The higher the priority a vehicle has assigned to it the less information is considered, relative to other vehicle movements, by the ANN and the heuristic implemented to avoid ANN decisions of moving onto unavailable or non existing tracks. In the limit, the higher priority vehicle has all tracks available when its decision about movement is being considered. However the vehicle will not implement a movement direction which leads to a currently occupied node but which

could be free by the time the vehicle would get there had that direction been chosen. After all vehicles have been moved according to the decisions made its priorities are rotated to guarantee that all vehicles will have equal priorities overall, and the process is then repeated.

8.1.5 ANN without heuristic but with look-ahead strategy

This case consists of using the ANN output vector without heuristics, or simple rules to prevent impossible solutions but with the use of a complete "look-ahead" strategy. It is implemented exactly as in the first ANN case (ANN with heuristic and with look-ahead) except that the heuristic which interprets and complements the ANN output vector is replaced by the movement direction indicated by the output of the ANN as represented in Figure 8- 12 (and also Figure 8- 6). If that movement is impossible then the decision will be to stop the vehicle at its current location. There are only two alternatives: move according to ANN output vector or stop.

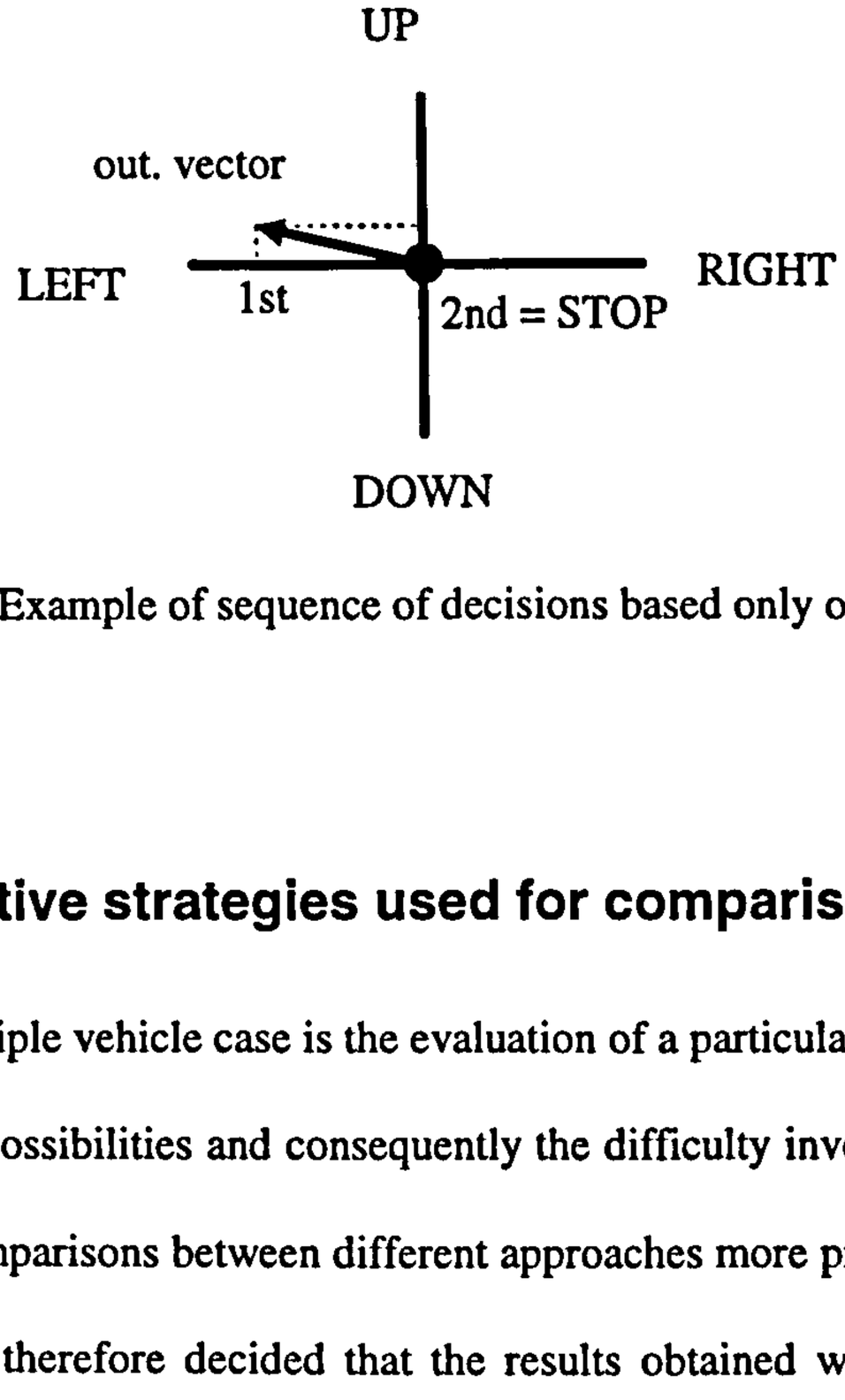


Figure 8- 12: Example of sequence of decisions based only on ANN output vector.

8.2 Other alternative strategies used for comparison

One major issue in the multiple vehicle case is the evaluation of a particular solution or approach. The increase in the number of possibilities and consequently the difficulty involved in finding the optimal solutions, makes use of comparisons between different approaches more practical than the comparison with an optimum. It was therefore decided that the results obtained with the solutions developed would be compared with other well known heuristic type solutions. Two heuristic type policies were used to evaluate the results obtained with ANN based solutions: the "greedy policy" and the

"benevolent counterclockwise flow policy" [Ref. 40]. These two policies were selected because they can be included in a strategy for controlling AGV traffic which has similarities with the proposed approaches based on ANN. None of the alternatives is based on a global optimization strategy, but rather on having the vehicles deciding autonomously on their routes using rules which are intended to serve a common good [Ref. 40], in a system where multiple routes are possible. While various rules could be derived and used in the same strategy, the "greedy policy" being a pragmatic implementation of a logical objective of always trying to get closer to the destination seemed an obvious choice to be tested. The other rule proposed by [Ref. 40] favouring a consistent direction had been shown to perform close to optimal and therefore it seemed satisfactory to test it also against the proposed solutions.

A solution in which the rules were replaced by the use of pseudo-random numbers was also implemented and tested because it was felt that it could provide a lower bound on the performance of any solution and therefore provide more information about the relative merits of the different solutions.

Another approach also implemented consists of the modification of the "greedy policy" that only considers two alternatives for each vehicle and which are: moving according to the first selected direction or else stop at current node.

Combining these four types of solutions with the "look-ahead" strategies and considering, when appropriate, the heuristic used with the ANN output vector, 10 more different cases have been implemented and are described in the following sections.

8.2.1 Greedy Policy and Benevolent Counterclockwise Flow Policy

The two rules or policies used to compare the results obtained with the ANN based approaches are applied equally to all vehicles in the system. As mentioned above the "greedy policy" results in each vehicle always trying to move into a location nearer to destination while the "benevolent

counterclockwise flow policy" tries to get nearer to destination node but using consistently the same direction which in this case is "counterclockwise". As in the solutions developed so far an iterative process is also involved consisting, at each step, of:

- selecting the higher priority vehicle;

- making a decision about its movement, based on the system status (nodes occupied) and the particular rules;

- updating system status (occupied nodes);

- selecting the second highest priority vehicle;

- making a decision about its movement, based on the system status (nodes occupied) and the particular rules;

- updating system status (occupied nodes);

- and so on until all vehicles have been considered.

After all vehicles have been moved, according to the decisions made, its priorities are rotated to guarantee that all vehicles will have equal priorities overall, and the process is then repeated.

The specific rules for the "greedy policy" consist of:

- among all unblocked immediately adjacent nodes, choose the node closest to the target node;

- if more than one node is equally close, choose randomly between them;

- if all adjacent nodes are occupied, wait at current node.

The specific rules for the "benevolent counterclockwise flow policy" are:

- define a referential (xy) co-ordinated axes centred on the current node and sequentially number (in a counterclockwise direction) the four quadrants in that referential system;

- locate the target node on the quadrants;

- move to the adjacent node which is on the same quadrant as the target node;

- if that node is occupied move to the node in the next quadrant (add 1 to the quadrant number);

- repeat previous step until an unblocked adjacent node is found;

- if all adjacent nodes are occupied wait at current node.

Comparing these two policies with the ANN plus simple rules approach they use more limited information about the system because they only consider the information about the immediate or adjacent nodes. Conversely, in the ANN information has been incorporated about all tracks in the system.

8.2.2 Greedy Policy and Benevolent Counterclockwise Flow Policy with look-ahead strategy

It was decided to extend the methods used by [Ref. 40] with the "look-ahead" strategy approaches already developed for the ANN based approaches. When this extension is applied to the greedy policy, each vehicle takes a decision using basically the same rules as the greedy policy. However instead of occupying the node selected based on the greedy policy rules, it keeps it as an intention movement which blocks the track used to arrive at that node. The next vehicle will then decide, based on the information about the tracks not available because they are reserved by the vehicles with higher priorities. After all vehicle's movements have been decided, then exactly the same "look-ahead" strategy as with the ANN based solutions is used:

- "greedy policy" with "look-ahead" strategy;

- "greedy policy" with limited "look-ahead" strategy.

Using the same approach with the benevolent counterclockwise flow policy the following strategies have also been implemented:

- "benevolent counterclockwise flow policy" with "look-ahead";

- "benevolent counterclockwise flow policy" with limited "look-ahead".

8.2.3 Pure gradient with look-ahead

Both the "greedy policy" and the "benevolent counterclockwise flow policy" consist of rules which always allow each vehicle to move in one of four possible alternative directions. It was therefore felt

that it would be comparing "like" with "like" if the ANN solution without heuristic (i.e. move along the direction of the ANN output vector, else stop) could be compared with a solution which only considers two alternative movements for each vehicle. A solution was therefore tested which was implemented in exactly the same way as the "greedy policy" except that if the closest node to the target is not available at the moment of decision it will stop. It is referred to here as "pure gradient" because it always tries to go to a closer to target path.

8.2.4 Random solutions

Testing solutions based on random decisions seemed to be worthy of consideration as they can provide a performance benchmark. A solution which consistently performs worse than one based on random decisions can be discarded.

The implementation of a random solution was realised by replacing the system which makes a decision about one of the four alternative movements out of a node, by a pseudo-random generator. In order to compare the same alternatives in the same conditions three cases based on random solutions have been implemented and tested as follows:

- "random choices" with heuristic with "look-ahead" strategy, which implements the same heuristic as the one defined to complement the ANN output vector (Figure 8- 9, except that the vector co-ordinates are randomly generated); and uses the same complete "look-ahead" strategy;

- "random choices" without heuristic with "look-ahead" strategy, which considers only two alternative movements out of a node (the one indicated by the randomly generated output vector, or stop), but uses the complete "look-ahead" strategy;

- "random choices" without heuristic without "look-ahead" strategy, which considers only two alternative movements out of a node (the one indicated by the randomly generated output vectors, or stop), and it does not use a "look-ahead" strategy.

The ten strategies just presented together with the four ANN based strategies presented in section 8.1 represent the fourteen strategies which were tested and evaluated as described in the following chapter, Chapter 9.

Chapter 9

Experiments and Results with Solutions for the Multi-Vehicle Case

The main objectives of this chapter are to describe the tests performed for the multi-vehicle case and to present the analysis and evaluation of the results obtained. The alternative solutions tested were described in the previous chapter, Chapter 8: based on ANNs, the global heuristic rules, and on the generation of pseudo-random numbers.

Four different measures were used to evaluate and compare the performance of these alternatives. One was a metric defined to give an overall measure of the time taken to complete all paths relative to an absolute lower bound length of those paths defined by the length of the minimum geometric distance with no traffic congestion. A measure of throughput representing the number of paths (origin, destination) which were completed was also considered. To give an idea how the number of paths found by a solution were distributed relative to each one's absolute minimum distance (or time) a cumulative (%) distribution of the number of paths completed within the same time as the minimum geometric distance with no traffic congestion is presented; or within a certain number of time units more than that minimum absolute time, expressed as a ratio. No one strategy was based on the implementation of an exhaustive search of all possible solutions which would allow the selection of the one performing the best, due to the complexity of the problem as it increases in size. Therefore there was no guarantee that all paths would always be found and so there was a need to analyse the number of cases not solved within the maximum time allowed.

9.1 Test conditions

Tests were done under the same conditions for all the 14 cases previously described and which will be designated from now on as represented next (Table 9- 1):

Table 9- 1: Identification of curves in the graphs.

ALTERNATIVE SOLUTIONS	**Look-ahead strategy:**		
	complete	**limited**	**none**
ANN based solutions, **- ANN output and 'heuristic':**	ANN_H_LH0	ANN_H_LH1	ANN_H_NLH
- only ANN output:	ANN_0_LH0		
Greedy Policy:	GREED_LH0	GREED_LH1	GREED_POL
Pure Gradient:	GRADI_LH0		
Benev. CounterClockwise Flow Policy:	BENVO_LH0	BENVO_LH1	BENVO_POL
Solutions based on random choices, **- with 'heuristic':**	RAN_H_LH0		
- without 'heuristic':	RAN_0_LH0		RAN_0_NLH

Experiments consisted of having a pre-defined number of vehicles in a layout, just like the one used for the single vehicle case (Figure 9- 1), with each vehicle being automatically assigned an origin and destination node as required, and this system running continuously for a pre-defined time duration.

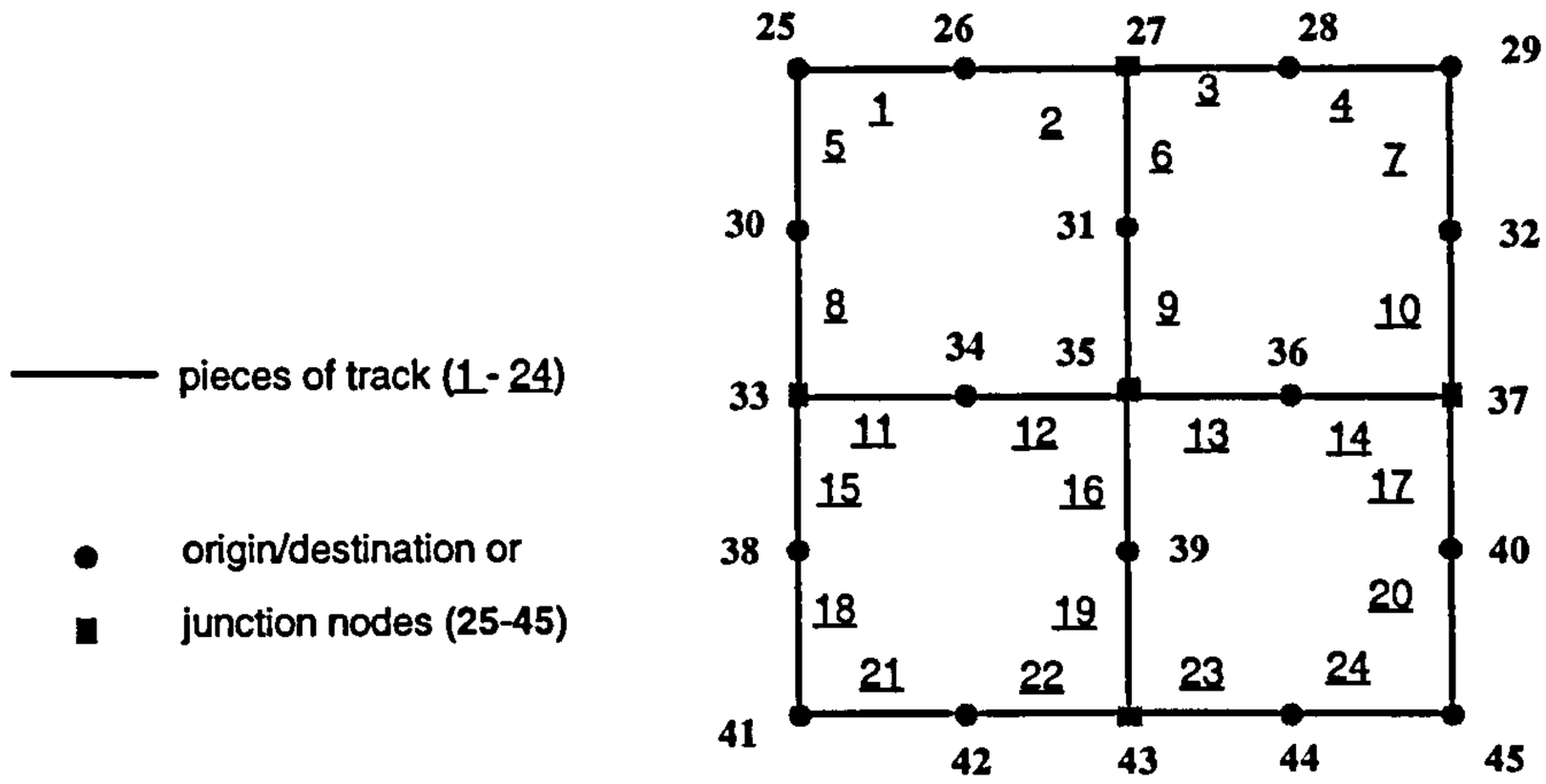


Figure 9- 1: Layout used for the tests.

Tests were made with the number of vehicles ranging from 1 to 10. Defining a unit of time as the time taken by a vehicle to travel along one track, with all tracks of the same length and all vehicles travelling with the same velocity. The duration of each test was 10000 units of time. When a vehicle had a decision to stop (i.e. is blocked) at a node, it would stay there until all vehicles move, i.e. for 1 unit of time. This assumption might not be accurate but it is adequate to test the method.

The initial pairs of origin and destination nodes were randomly generated using a pseudo-random number generator with the seed value set to 5. The destination nodes were generated from the 16 machine location nodes, and not from the nodes at crosses (Figure 9- 1). A destination node could not be assigned to more than one vehicle. When a vehicle completed a path it was assigned a new path which consisted of the current node and a new randomly generated destination node. This means that if the average path length is 4 units (or tracks) and there are no stops at the nodes, one vehicle would complete 2500 paths within the duration of the test (10000 units of time).

As already mentioned when describing the program which implements the interface to the ANN in the hybrid solution (section 8.1.1) a number of issues had to be resolved in order to overcome "deadlock"

situations (as described in Chapter 8) in a system running continuously. There is no guarantee that a situation would not occur where the decisions successively taken by a vehicle will result in with the vehicle always repeating, in cycles, a path which never includes the assigned destination node. Or even if this situation does not last indefinitely it is reasonable to accept that after a time it can be considered to be of little or no interest. At this point it was decided that a way to get out of these possible deadlock situations was to limit the time allowed for a vehicle to complete a path. This limit was set to 200 units of time which is more than 10 times the length of the longest possible path that does not include repeated tracks or nodes in the physical layout chosen. It was considered to provide a long enough time horizon to characterize the paths resulting from each test, and in a practical case other maximum limits could be selected and other strategies applied to these vehicles.

When a vehicle reached 200 units of time a deadlock situation was assumed, a new destination is assigned to the vehicle and a new path is started from the current node to the newly assigned destination node. However before the 200 time units have elapsed a change was introduced into the system with the objective of avoiding a possible deadlock situation. The motivation behind this was to the alter the behaviour of the vehicle(s) by using different rules. This change consisted of assigning, temporarily, a new destination node to the vehicle while keeping in memory its original destination node. This new temporary destination node is maintained for a limited number of time units, or until the vehicle reaches it, after which the initial destination node is re-assigned to the vehicle. If the vehicle visits the initial destination node while searching for the temporary destination node it recognizes it and completes the initial path. Once again the time allowed for a vehicle to search for a path before this change is introduced, and the time before the initial destination node is re-assigned, were arbitrarily defined as 50 time units each. It was considered that in this physical layout more than twice the maximum path length, not containing repeated tracks or nodes, was a reasonable estimate for the reference time. In a practical case this time would be estimated based on the characteristics of the particular layout.

The temporary destination node is selected from one of the nodes adjacent to the target node, or else another of the available nodes is randomly selected. In any case the selected node cannot be the one in use as destination node for other vehicles, nor the current node of the vehicle under analysis. Also it

will be selected from the 1 to 16 nodes considered as locations of machines, except when, due to the large number of vehicles in the system, only one of the nodes at junctions is available.

A path is considered to be completed either when a vehicle arrives at the target node or uses up the 200 units of time allowed to find the target. In this last case it would be counted as a 'non solved' path with a 200 units of time length.

One other issue relevant in the ANN based solutions, is related to the alternative to move backwards through the track previously used to get to the current node. The heuristic applied to complement the decisions obtained from the ANN output vector, considered moving backwards as the last movement alternative. However the possibility to specify a time after which the moving backwards alternative is 'forgotten' in the sense that it is considered no different from the others was also implemented. The reason behind this is to allow a vehicle which stays for several units of time waiting at the same node to follow the preferred moving direction even if moving backwards. It is just a way of assuming that after a certain time without moving it means that the traffic conditions may have changed and therefore the reasons that led to the current location are no longer valid. After several experiments with different values for this waiting time, it was decided to set it to 3 units of time, as this was the minimum figure which allowed the vehicle to use the backward move effectively.

The results were collected in files which contained all the information about each path executed by each vehicle. A program (described in Appendix D) was written in "C" to analyse the data according to the measures of performance defined and it was used in all cases to collect the data required.

Table 9- 2 shows how the total number of paths executed varied in the tests performed, while in Table 9- 3 is represented an average estimate for the value of the path lengths (travel + waiting times).

Table 9- 2: Total number of paths executed during the 10000 time units duration of the tests performed with each of the alternative solutions. Number of vehicles ranged from 1 to 10.

	1_VEH	2_VEH	3_VEH	4_VEH	5_VEH	6_VEH	7_VEH	8_VEH	9_VEH	10_VEH
ANN_H_LH0	2508	4676	6427	7871	8827	9800	10155	10189	10303	9574
ANN_H_LH1	2508	4676	6444	7764	8620	9184	9546	9336	9044	8156
ANN_H_NLH	2508	4483	5107	5677	6221	6210	6050	5838	5566	5349
ANN_0_LH0	862	1001	967	919	744	1060	1003	946	954	995
GREED_LH0	2569	4009	6269	7838	9002	9697	9666	9690	8695	8264
GREED_LH1	2569	4009	6497	7665	8629	9039	8815	8000	6557	5968
GREED_POL	2569	1871	5917	6637	7652	8089	8149	8212	7628	7258
GRADI_LH0	2569	1368	914	982	856	745	765	707	725	721
BENVO_LH0	502	935	1815	2634	3850	4926	5863	6371	6526	5103
BENVO_LH1	502	935	1989	2366	4151	4622	5206	5049	4804	3966
BENVO_POL	502	543	1820	2362	4147	4904	6194	6565	6623	6510
RAN_H_LH0	551	1090	1566	2026	2457	2825	3223	3401	3620	3926
RAN_0_LH0	137	254	373	430	532	600	639	722	761	785
RAN_0_NLH	137	219	337	434	504	555	589	647	716	773

Table 9- 3: Average length (including waiting time) per vehicle, of the paths executed during the 10000 time units duration of the tests performed with each of the alternative solutions. Number of vehicles ranged from 1 to 10.

	1_VEH	2_VEH	3_VEH	4_VEH	5_VEH	6_VEH	7_VEH	8_VEH	9_VEH	10_VEH
ANN_H_LH0	3.99	4.28	4.67	5.08	5.66	6.12	6.89	7.85	8.74	10.44
ANN_H_LH1	3.99	4.28	4.66	5.15	5.80	6.53	7.33	8.57	9.95	12.26
ANN_H_NLH	3.99	4.46	5.87	7.05	8.04	9.66	11.57	13.70	16.17	18.70
ANN_0_LH0	11.60	19.98	31.02	43.53	67.20	56.60	69.79	84.57	94.34	100.50
GREED_LH0	3.89	4.99	4.79	5.10	5.55	6.19	7.24	8.26	10.35	12.10
GREED_LH1	3.89	4.99	4.62	5.22	5.79	6.64	7.94	10.00	13.73	16.76
GREED_POL	3.89	10.69	5.07	6.03	6.53	7.42	8.59	9.74	11.80	13.78
GRADI_LH0	3.89	14.62	32.82	40.73	58.41	80.54	91.50	113.15	124.14	138.70
BENVO_LH0	19.92	21.39	16.53	15.19	12.99	12.18	11.94	12.56	13.79	19.60
BENVO_LH1	19.92	21.39	15.08	16.91	12.05	12.98	13.45	15.84	18.73	25.21
BENVO_POL	19.92	36.83	16.48	16.93	12.06	12.23	11.30	12.19	13.59	15.36
RAN_H_LH0	18.15	18.35	19.16	19.74	20.35	21.24	21.72	23.52	24.86	25.47
RAN_0_LH0	72.99	78.74	80.43	93.02	93.98	100.00	109.55	110.80	118.27	127.39
RAN_0_NLH	72.99	91.32	89.02	92.17	99.21	108.11	118.85	123.65	125.70	129.37

9.2 Performance measures

The data collected from the various cases was evaluated using the following four types of measures:

- METRIC, representing an overall measure of the time (travel + waiting) to execute all paths, including the ones which were not solved, relative to the time to travel directly along minimum geometric distance paths between origin and destination nodes. Obviously an absolute minimum value for this metric will be 1, which corresponds to no traffic congestion (no stops or waiting at a node) and all vehicles accomplish their paths using the minimum geometric distance. The lower the value of this metric the closer a solution will be to a minimum overall length of the executed paths. This metric is given by:

$$METRIC = \frac{\sum_{i=1}^{NPaths} TravelTime_i + \sum_{i=1}^{NPaths} WaitingTime_i}{\sum_{i=1}^{NPaths} T\_Min\_Path_i}$$

Where,

NPaths: is the total number of paths completed (including not solved)

TravelTime: is the travel time

WaitTime: is the waiting time

T_Min_Path: is the time to travel the path along minimum geometric distance without traffic congestion.

- THROUGHPUT, representing an average measure of the number of paths completed every unit of time by all vehicles in the system. This measure is intended to represent how effective an increase in the number of vehicles in the system can be in terms of increasing the number of transport orders accomplished. Increasing the number of vehicles in the system increases the means available to accomplish transport orders. However it also increases the possibility of having traffic congestion which in turn may affect the time to complete the transport orders. If increasing the number of vehicles in the system does not cause any traffic congestion then it can be assumed that the number of transport orders accomplished will increase linearly with the number of vehicles in the system. It is therefore reasonable to assume a linear increase until traffic congestion starts to have an effect. While one vehicle cannot experience traffic congestion, it is possible that 2 vehicles could. An overall

absolute best performance of a solution is a linear increase of the throughput with the number of vehicles and starting with the optimum performance for one vehicle.

The *'THROUGHPUT'* value for each test was derived as an average value considering the number of paths (i.e. transport orders) completed by each of the vehicles involved and the total time needed by each of the vehicles to accomplish their respective orders. This *'THROUGHPUT'* value was calculated using the *'THROUGHPUT'* formula presented below where the average time per order for each of the vehicles involved in the test is calculated first. Adding these average times per order for all vehicles and dividing by the number of vehicles involved in the test we obtain the average time per order and per vehicle. When taking its inverse, this average time per order and per vehicle, becomes an average, over all vehicles and transport orders, of the number of orders completed per unit of time per vehicle. Multiplying the result obtained in the previous step by the number of vehicles involved in the test yields the *'THROUGHPUT'* value; it represents an average value of the number of transport orders accomplished each time unit by all vehicles involved in the test.

$$THROUGHPUT = \left(\frac{\sum_{i=1}^{NumVeic} \left(\frac{TotalTime}{NumPaths} \right)_i}{NumVeic} \right)^{-1} * NumVeic$$

Where,

NumVeic: is the number of vehicles in the system

NumPaths: is the total number of paths, or transport orders, completed by each vehicle in the system

TotalTime: is the total time taken to complete all paths by each vehicle

- NUMBER OF BAD SOLUTIONS, is the number of solutions, i.e. paths linking the origin and destination nodes, which were not accomplished before the maximum time allowed (200 units of time) was reached.

- DISTRIBUTION of K-RATIO, represents the number of paths that can be included in one of the classes defined for the *'K'* ratio. This measure was developed to provide a more detailed characterization of the individual paths executed in each set of tests. The *'K'* ratio represents the value obtained when dividing the time required to execute a path, using one of the strategies tested, by the time of the path along the minimum geometric distance without traffic congestion. The classes considered for the *'K'* ratio are represented by the values ranging from 1.0 (minimum) to 200.0 (maximum), with increments of one tenth (0.1). This distribution is represented as an accumulated (%) distribution. For each path a *'K'* value is calculated using the following expression, which is rounded to one decimal place so it can be associated with one of the *'K'* classes considered:

$$k_i = \frac{TimePath_i}{T\_Min\_Path_i}$$

Where, *TimePath:* is the total time (travel + waiting) to execute path *'i'*

T_Min_Path: is the time to travel the path along minimum geometric distance without traffic congestion, linking the origin and destination nodes as in path *'i'*.

While the previous measures of performance (*'metric'* and *'throughput'*) represented an aggregate measure of performance which included all paths executed, the distribution of the *'K'* values for each strategy represents individually the information for each of the paths executed. In either of these situations an overall lower bound on quality (i.e. time duration) is used which is based on a hypothetical possibility of executing all transport orders along the minimum distance paths without stopping (i.e. traffic congestion). Obviously this will not always be possible, especially if several vehicles are involved, and is used because the global optimum solution is not available and the objective is to evaluate each strategy by comparing it with the others when the same measures are used.

9.3 Graphical analysis of results

The results obtained with each of the tests performed were collected for presentation in the form of graphs which are organized sequentially for each of the described measures of performance. The first

graph shows data from all 14 cases tested and after selecting, when appropriate, from that graph only the cases to be compared in more detail and with a more suitable scale.

The legend in the graphics representing each of the cases tested can be interpreted as indicated in Table 9- 1.

9.3.1 Graphs of *'metric'* performance measure

The graphs in Figure 9- 2 to Figure 9- 9 represent the data of the *'metric'* performance measure described in section (9.2). Each point in these graphs was obtained by adding the total time of all paths executed during 10000 time units (Table 9- 2) and dividing it by the time taken to complete the same transport orders along the minimum distance path and with no traffic congestion.

The values for the *'metric'* obtained when testing all type of solutions ranges from a maximum of 37.6 to a minimum of 1. An obvious example where this minimum value of 1 is expected is in cases using greedy policy type solutions with one vehicle in the system.

Starting with the graph in Figure 9- 2 representing all cases tested, the major conclusions can be summarized as follows:

- there is a clear distinction between the best cases and the worst cases, which is represented by a difference of at least a factor of 7 in the *'metric'* values (from below 5 to above 35),

- the worst values were obtained in cases which only consider two alternative decisions at each node: either movement along a direction chosen according to a specific rule or stop if that movement is not possible (i.e. RAN_0_NLH, RAN_0_LH0, GRADI_LH0, ANN_0_LH0),

- the group representing the best cases, is limited by a lower bound (best overall values) which in general (exception with 1 vehicle where strategies based on the greedy policy also reach the best value) were obtained with cases using the complete 'look-ahead' strategy. The larger values in this group are the values obtained with a random type strategy with 'look-ahead' (RAN_H_LH0) except when the number of vehicles is 1 or 2,

- while in the best case the curve shows a continuous increase from 1 to 10 vehicles, other strategies especially the ones not based on the ANN with heuristics, present a significant alteration to the curve when the number of vehicles is equal to 2.

A more detailed analysis of the worse cases shows the ones based on random choices as being generally worse, especially in cases with low numbers of vehicles in the system. With 8 and 9 vehicles in the system the pure gradient performs on a par with the two random based solutions; and with 10 vehicles it becomes the worse solution as determined by this *'metric'* measure. The direct ANN solution, i.e. without heuristic, performs the best with a large number of vehicles in the system (more than 5), being close to the pure gradient solution with a lower number of vehicles except with 1 and 2 vehicles. With only 1 vehicle obviously the pure gradient will perform optimally as it always chooses the nearest nodes and for these there are no conflicts with other vehicles.

A detailed analysis of the best cases becomes more clear when we consider the graph (Figure 9- 3) with only those cases using a higher resolution scale. The ANN solution with heuristic and 'look-ahead' (ANN_H_LH0) and the greedy policy with 'look-ahead' (GREED_LH0) are better than any of the others (Figure 9- 3), in particular better than the solution based on the benevolent counterclockwise flow policy (BENVO_LH0). With only 1 vehicle the solution based on the greedy-policy is slightly better than the ANN solution, however with 2 vehicles the ANN is clearly better. With 3, 4, 5, and 6 vehicles the two solutions are very close one to the other. However the ANN solutions can be seen to be significantly better in the cases with a larger number of vehicles (7 to 10).

The particularly bad behaviour of the benevolent counterclockwise flow policy deserves a special reference as it has been shown to perform closely to optimum in a uniform rectilinear grid type layout [Ref. 40]. The differences in performance now verified using this solution might be due to the comparatively reduced number of ways out that can be considered in most of the nodes in the current layout. In a uniform rectilinear grid type layout all the nodes except those on the outside limits of the layout have four alternative ways out, while in the current layout the machine nodes have only two ways out [see Figure 5-1 in pp. 39]. Therefore the use of a consistent flow direction in these

circumstances will tend to favour movements in the backward direction from the destination targets, rather than on a consistent movement around blocked paths to arrive at the destination targets.

A similar behaviour to the one observed previously with solutions using the complete 'look-ahead' strategy can be observed in Figure 9- 4 with the curves representing the best solutions with only a limited 'look-ahead' strategy. In overall terms the ANN solution (ANN_H_LH1) continues to perform better with a number of vehicles equal to 2 or even better when the number of vehicles is more than 6.

When comparing the solutions which do not use any 'look-ahead' strategy a different behaviour, from the one observed in the previous analysis, can be observed in the graph of Figure 9- 5. However the ANN solution (ANN_H_NLH) only becomes significantly better with a number of vehicles equal to 2. Above that number of vehicles the greedy-policy becomes increasingly better with an increase in the number of vehicles in the system. The ANN solution is worse than the benevolent counterclockwise solution (BENVO_POL) when the number of vehicles increase above 7.

In Figure 9- 6 we can observe the three cases based on random choices. They show a significant difference between using an heuristic to increase the number of alternative movement solutions considered by a vehicle when at a node, and only using two alternatives: either move to the direction chosen or else stop.

The last graphs in Figure 9- 7, Figure 9- 8, and Figure 9- 9 show respectively how the use of a 'look-ahead' strategy (complete, limited, or none) affects performance when solutions based on the ANN, the greedy policy, or the benevolent counterclockwise flow policy are used. While in the ANN type solutions the use of a 'look-ahead' strategy always clearly improves performance. In the greedy policy type solutions this improvement is very significant when two vehicles are in the system but in general the differences in performance are less significant; the solution not using any 'look-ahead' performs ahead of the solution using limited 'look-ahead' when eight or mores vehicles are in the system. The solutions based on the benevolent counterclockwise flow policy have the peculiarity of performing badly both when a high and a reduced number of vehicles are in the system. Also, except when two vehicles are in the system, the solution not using any 'look-ahead' generally performs the best.

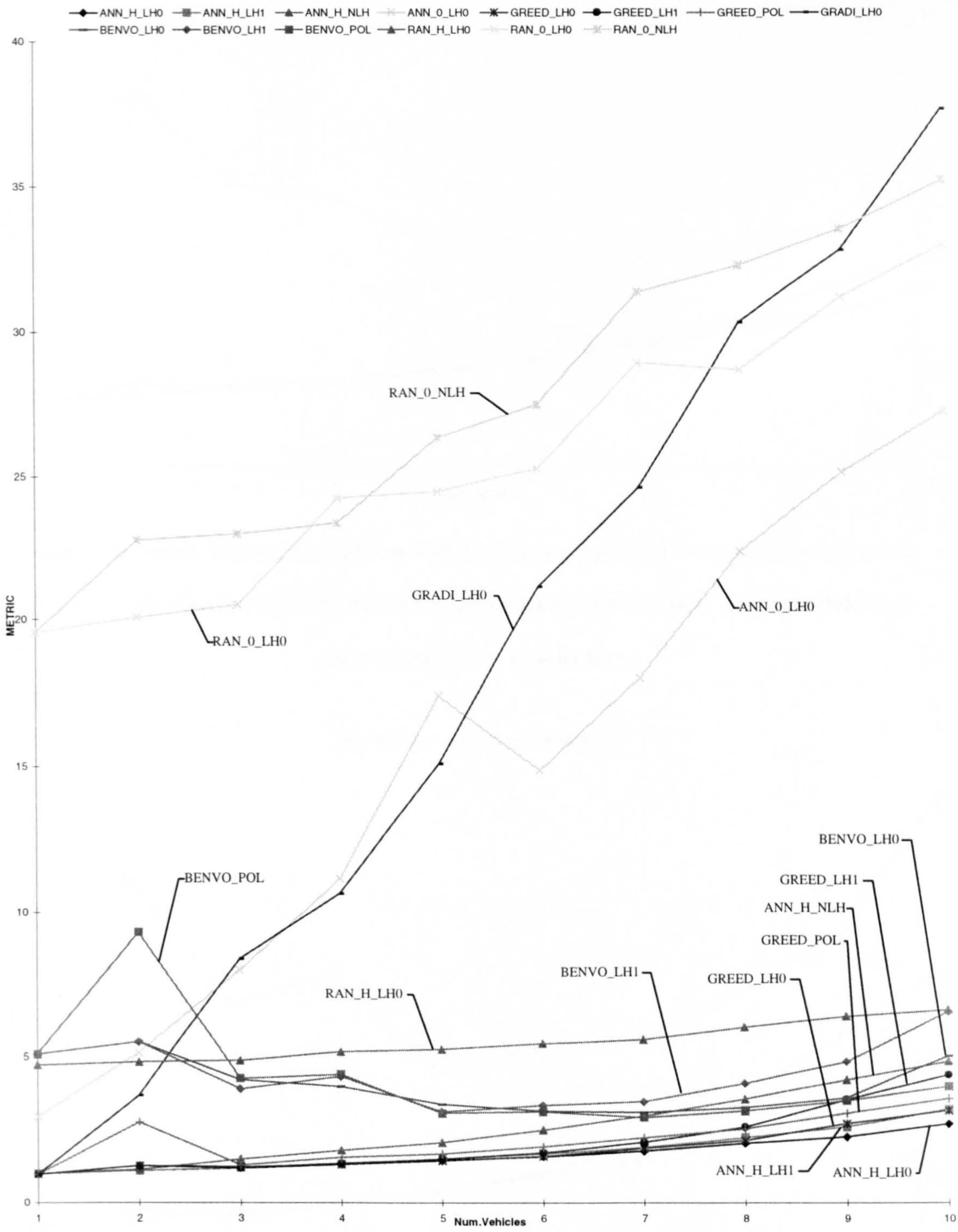


Figure 9- 2: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. All strategies, 1 to 10 vehicles.

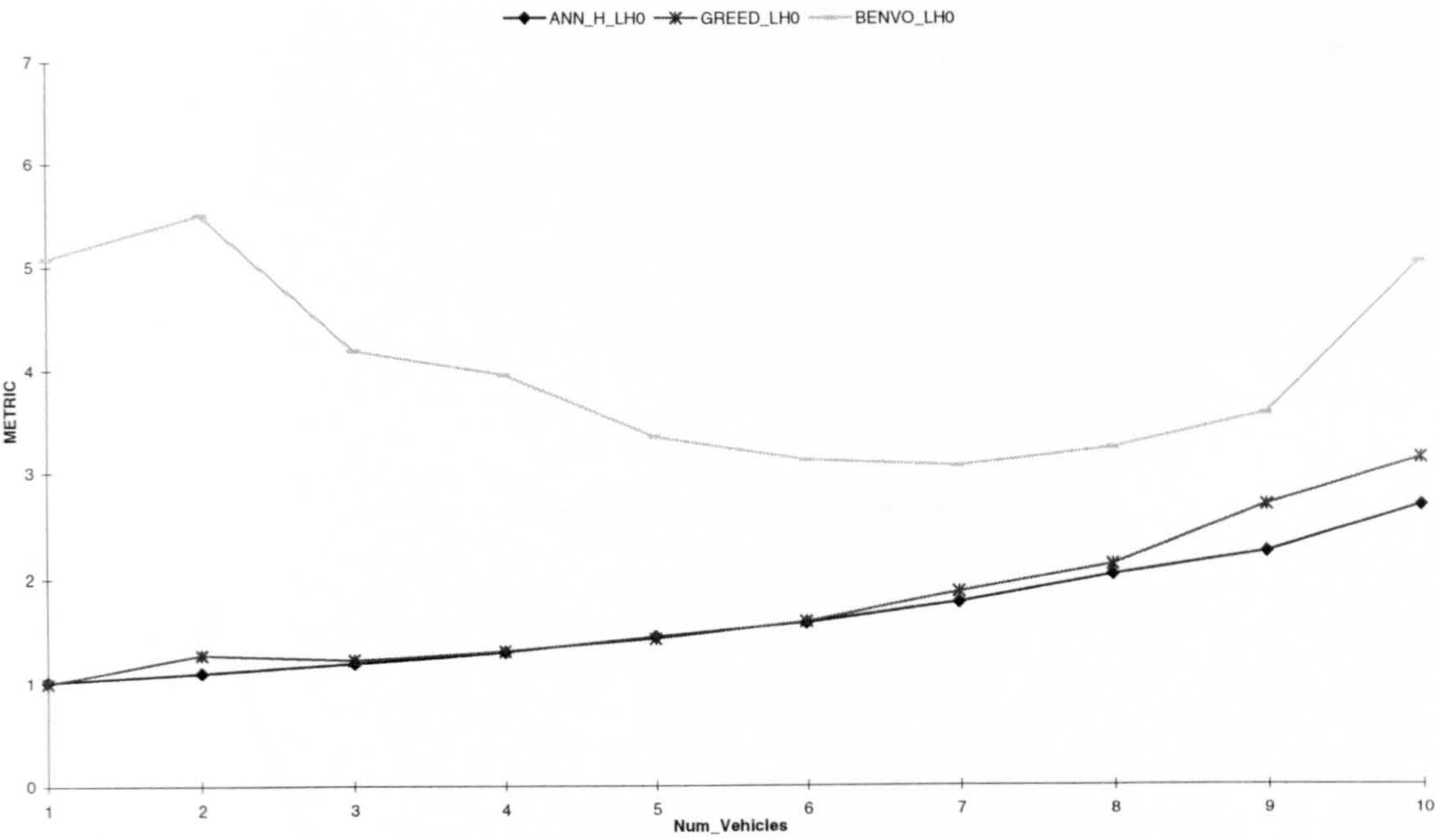


Figure 9- 3: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. ANN, Greedy Policy, Benevolent Counterclockwise Flow Policy, **with complete** 'look-ahead' strategy, 1 to 10 vehicles.

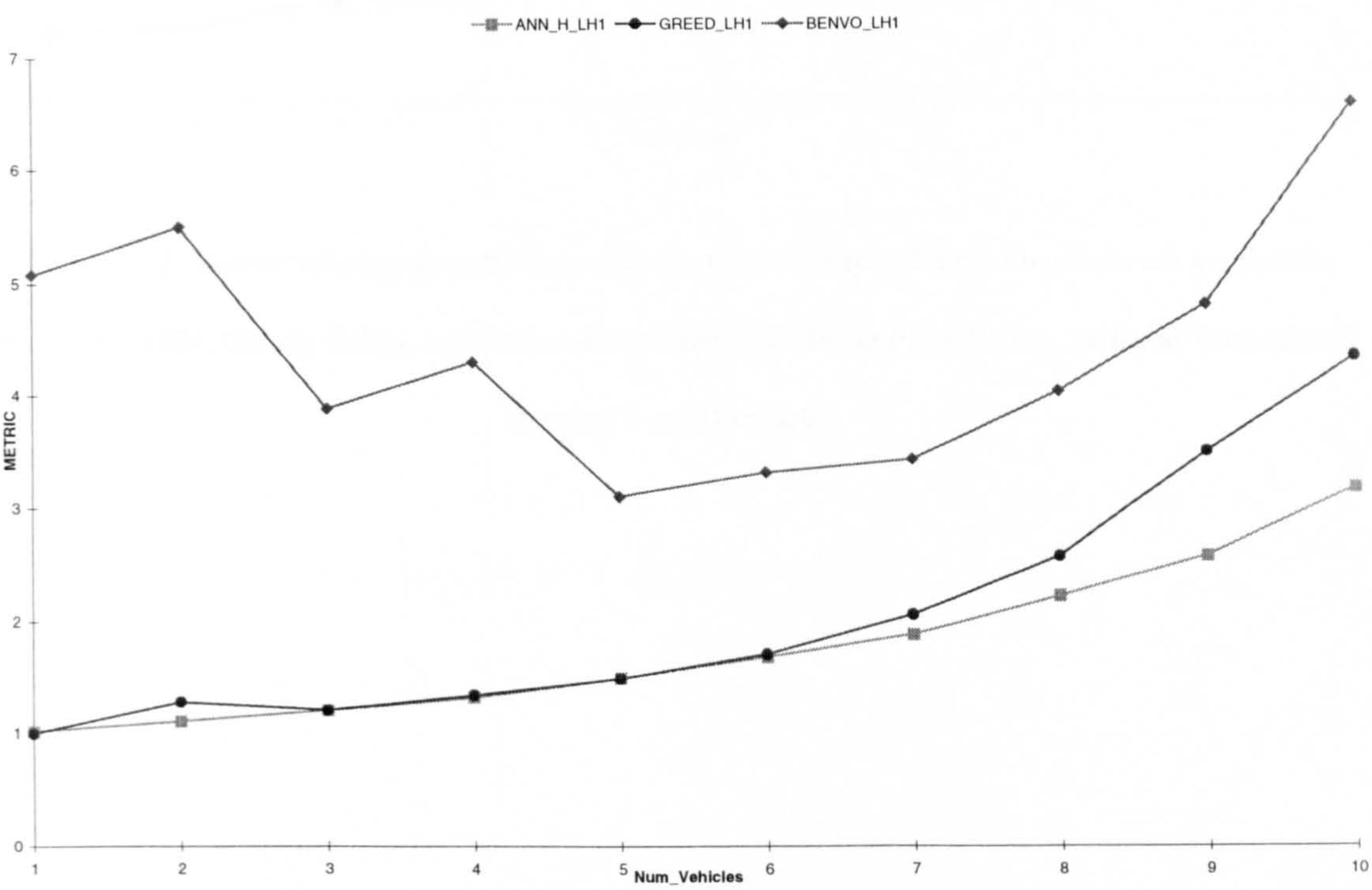


Figure 9- 4: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. ANN, Greedy Policy, and Benevolent Counterclockwise Flow Policy, **with limited** 'look-ahead' strategy, 1 to 10 vehicles.

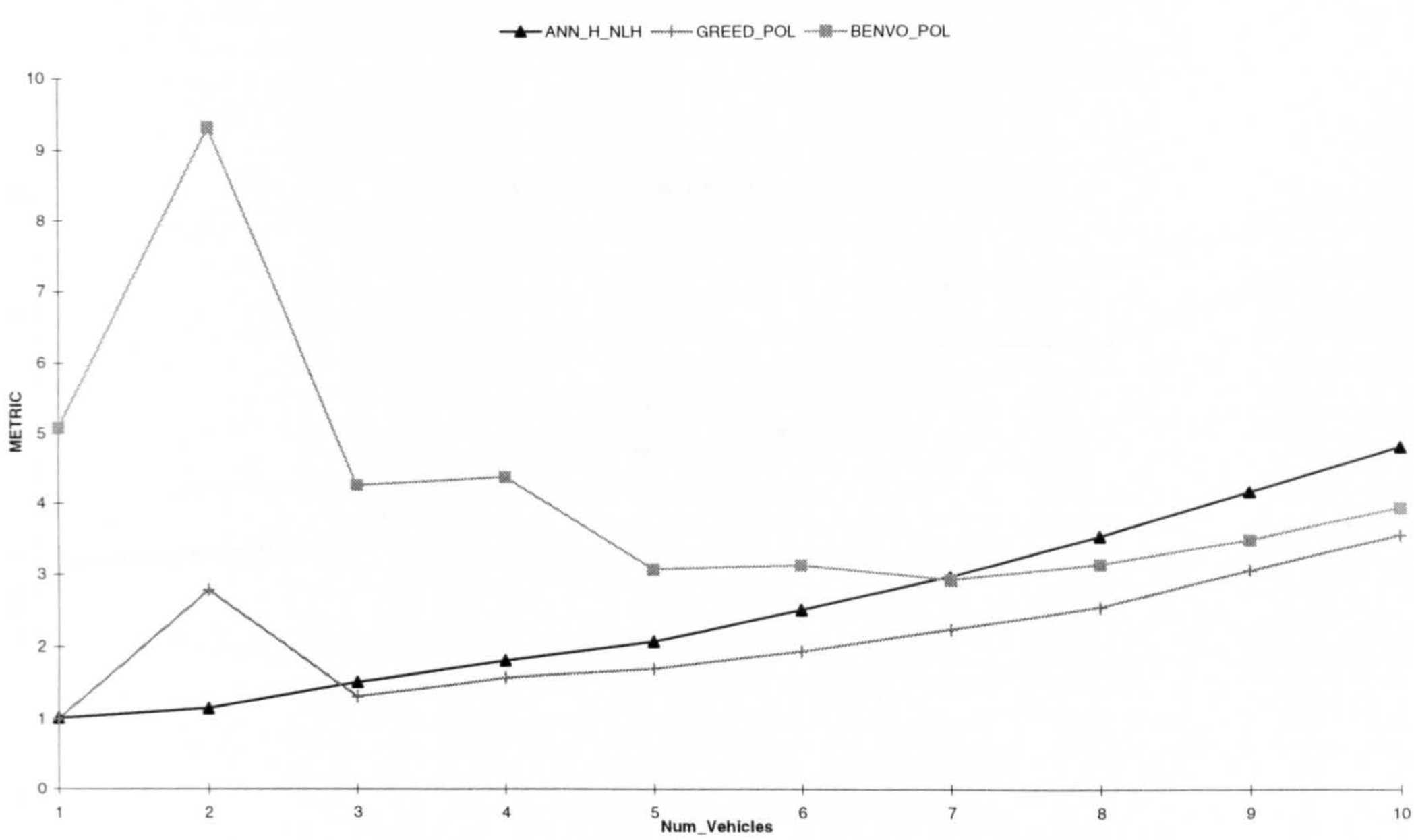


Figure 9- 5: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. ANN, Greedy Policy, and Benevolent Counterclockwise Flow Policy, **without** 'look-ahead' strategy, 1 to 10 vehicles.

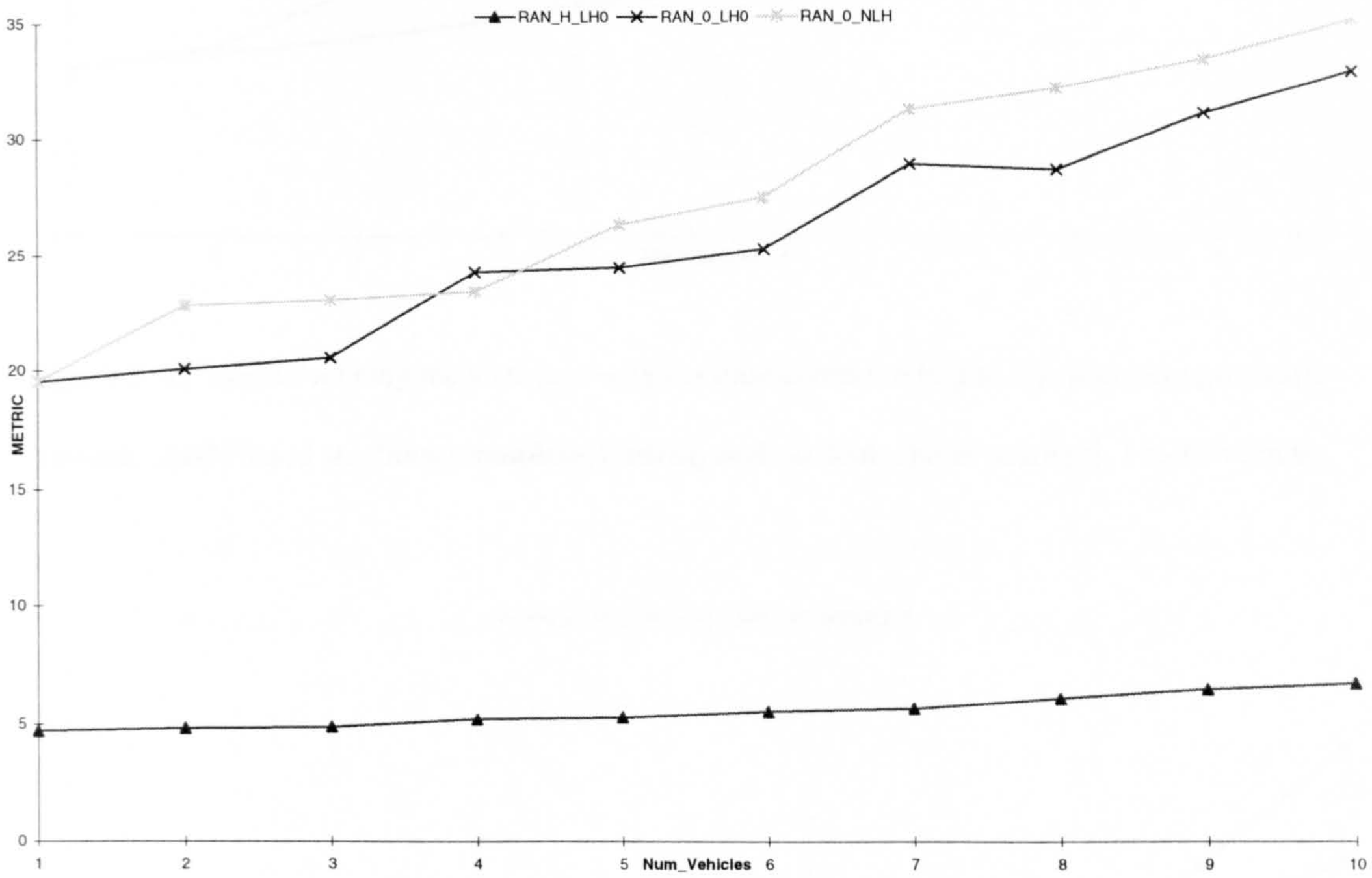


Figure 9- 6: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. Solutions based on random choices, 1 to 10 vehicles.

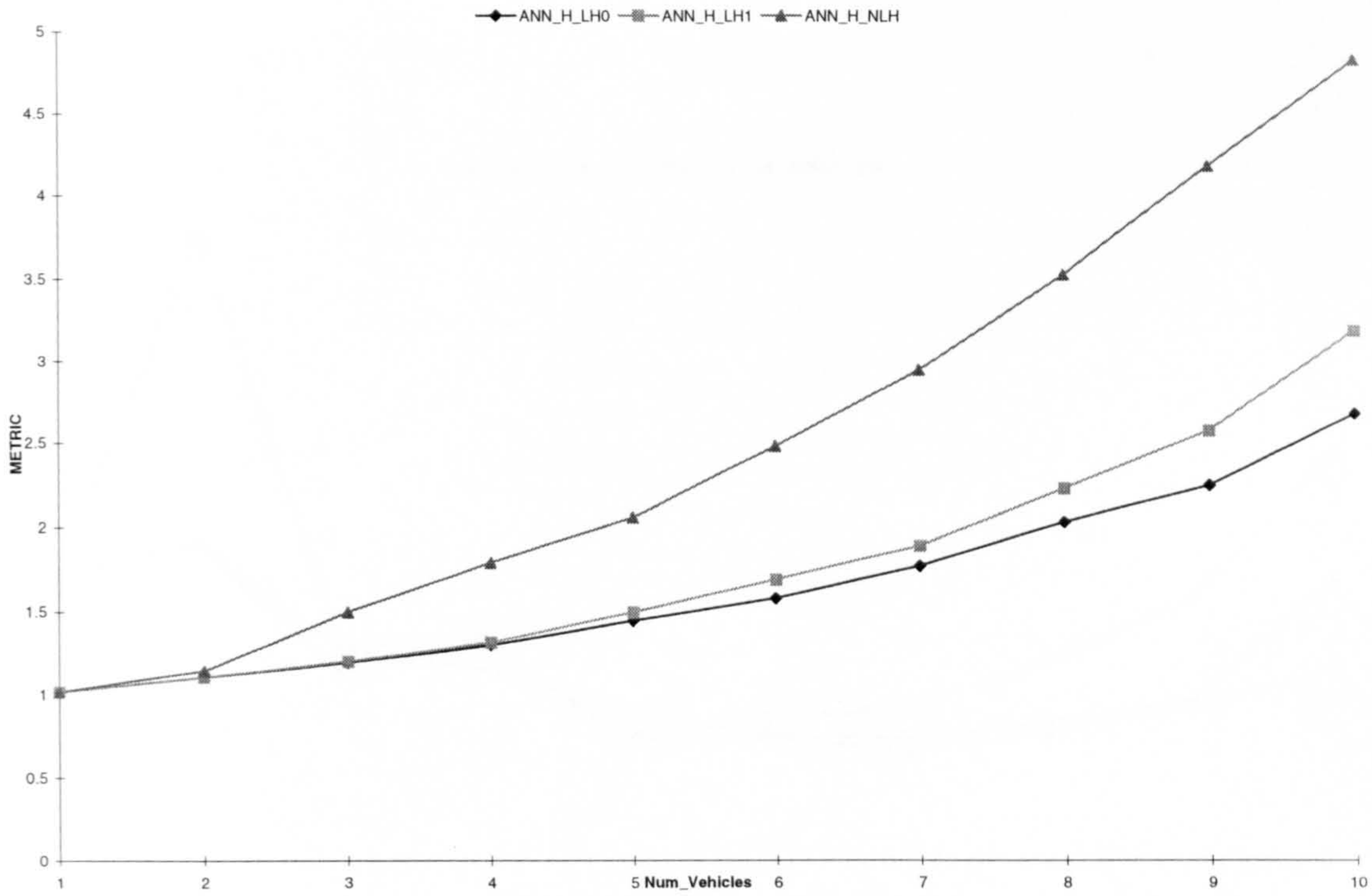


Figure 9- 7: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. ANN based solutions (**complete**, **limited, and no** 'look-ahead' strategy), 1 to 10 vehicles.

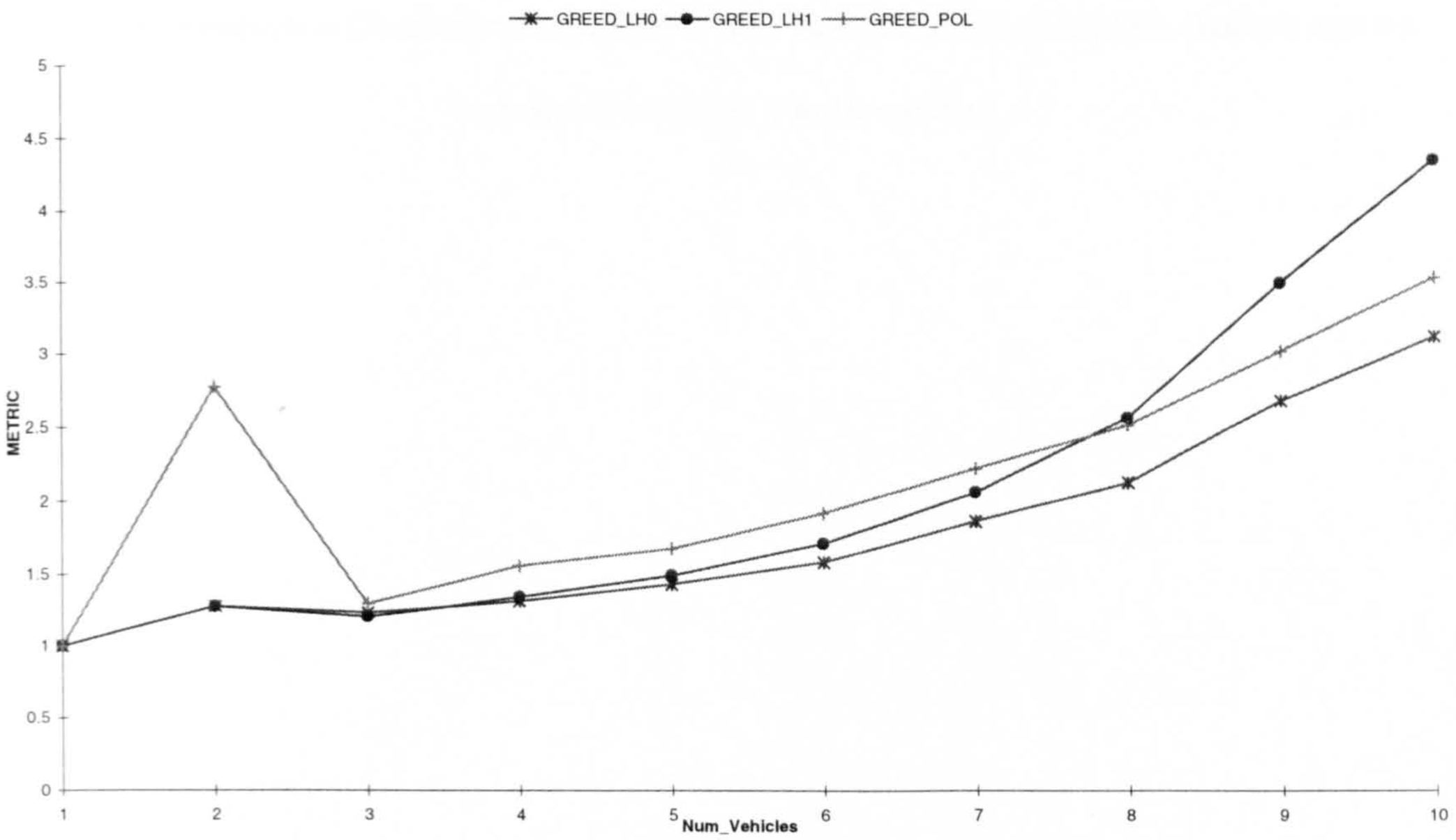


Figure 9- 8: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. Greedy Policy based solutions (**complete**, **limited, and no** 'look-ahead' strategy), 1 to 10 vehicles.

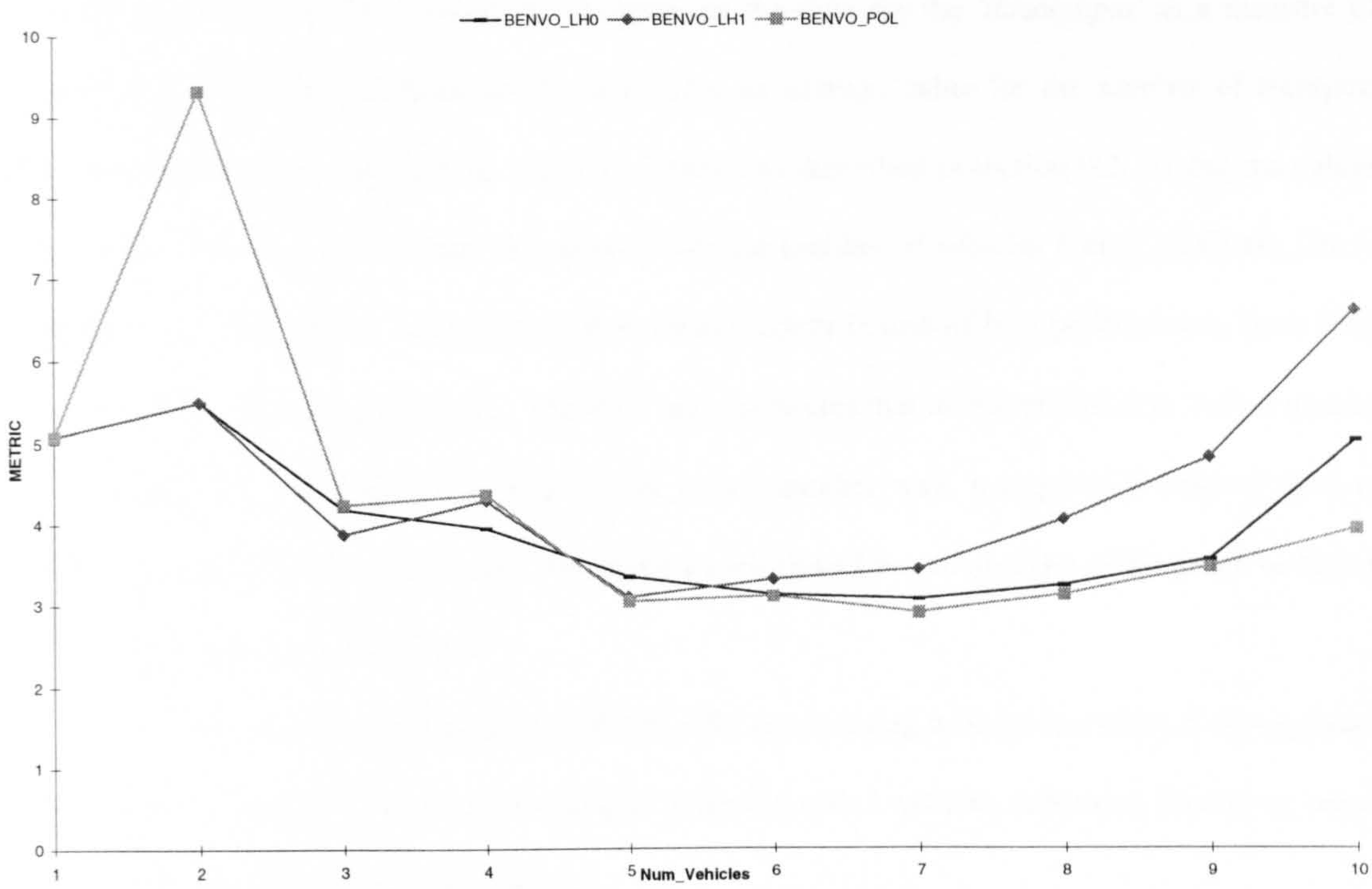


Figure 9- 9: *'metric'* relating the total time with the time corresponding to the minimum geometric distance. Benevolent Counterclockwise Flow Policy based solutions (**complete**, **limited, and no** 'look-ahead' strategy), 1 to 10 vehicles.

9.3.2 Graphs of *'throughput'* performance measure

The graphs in Figure 9- 10 to Figure 9- 18 represent the data for the *'throughput'* as a measure of performance. Each value in these graphs represents an average value for the number of transport orders accomplished in a unit of time, and was obtained as described in section 9.2. Using the values representing a linear increase in the *'throughput'* with the number of vehicles (curve LINEAR_GR in the graphs), the *'throughput'* values range in an overall upper bound of best performance, from 0.25 with 1 vehicle to 2.5 with 10 vehicles. The 0.25 value indicates that in one unit of time only a quarter of a transport order has been performed, or put it in a another way, it requires 4 units of time to complete a transport order, on average. The major conclusions from an analysis of the graph in Figure 9- 10, can be summarised as follows:

- as expected the straight line (LINEAR_GR) representing a linear increase of *'throughput'*, starting with the maximum value of *'throughput'* achieved with 1 vehicle, represents clearly an upper bound never reached by any of the solutions

- also as expected, it can be seen that most of the curves representing solutions which perform better (i.e. higher values) show an initial increase of *'throughput'* with an increase in the number of vehicles, but after a certain number of vehicles the *'throughput'* starts to decrease

- there is a group of solutions which perform rather badly, either in terms of low values (i.e. less than 0.1) of *'throughput'* with a low number of vehicles and also because an increase in the number of vehicles does not significantly change these low figures of *'throughput'* values

- as in the *'metric'* graphs the solutions based on 'look-ahead' strategies seem to perform the best, and present significant increases in the *'throughput'* values with up to 6 vehicles. With more than 6 vehicles only the curve ANN_H_LH0 shows a small increase which starts to decay significantly with 10 vehicles

- the curves representing the solutions based on the greedy policy show, as in the *'metric'* graphs also, a particular behaviour when the number of vehicles is 2.

A more detailed analysis of these situations can be made by observing the other plots of the *'throughput'* graph as presented next:

- observing Figure 9- 11, of the solutions with the complete 'look-ahead' strategy the solution based on ANN with heuristic (ANN_H_LH0) has a similar behaviour to the equivalent solution based on the greedy policy (GREED_LH0). Both of them perform better than the equivalent solution based on the benevolent counterclockwise flow policy (BENVO_LH0). The ANN solution performs significantly better than the greedy policy solution when the number of vehicles in the system is either 2, or more than 6. Being quite similar to the rest of the points of the curves. Other than achieving higher values for the *'throughput'* measure the ANN solutions also achieves its maximum *'throughput'* later, with 9 vehicles, whereas the solution based on the greedy policy starts to look saturated with 6 vehicles in the system and significantly decreasing with 9 vehicles.

- in Figure 9- 12, a similar behaviour to that previously observed in Figure 9- 11, can be observed with the curves representing the three type of solutions, ANN, greedy policy, and benevolent counterclockwise flow policy, but with a limited 'look-ahead' strategy. The ANN (ANN_H_LH1) solution remains the overall best or close to the best performance, in terms of maximum *'throughput'* values. However the ANN solution now starts to decline when the number of vehicles exceeds 7, and the solution based on the greedy policy (GREED_LH1) decreases significantly after 6 vehicles. The solution based on the benevolent counterclockwise flow policy (BENVO_LH1) still remains the one which performs the worst of the three.

- in Figure 9- 13, we can observe the curves representing the same three solutions, ANN, greedy policy, and benevolent counterclockwise flow policy, as before but now with no 'look-ahead' strategy. A similar behaviour to the one observed when analysing the *'metric'* performance with the same solutions is observed. The ANN solution (ANN_H_NLH) significantly outperforms the other two solutions, the greedy policy (GREED_POL) and the benevolent counterclockwise flow policy (BENVO_POL), when the number of vehicles equals 2. With the number of vehicles ranging from 3 to 6 the ANN solution performs worse than the greedy policy and better than the benevolent counterclockwise flow policy. With the number of vehicles increasing above 6 the ANN solution becomes the worse of the three. Also in terms of the number of vehicles where each strategy reaches its maximum *'throughput'* the ANN is the one which reaches that value first, at 5 vehicles.

- in Figure 9- 14, it can be seen that the solutions based on only two alternatives for movement of a vehicle when at a node (moving in the first direction selected, or else stopped), the pure gradient (GRADI_LH0), the ANN solution (ANN_0_LH0), and a solution based on random choices (RAN_0_LH0), all perform rather badly, with low values of *'throughput'* except for the pure gradient solution with 1 vehicle in the system. The worse of the three is the solution based on random choices up to 8 vehicles, after which the pure gradient solution becomes the one with lower *'throughput'* values. The ANN solution is close to a flat curve and performs the best of the three for 6 to 10 vehicles.

- Figure 9- 15 represents the curves of the solutions based on random choices (RAN_0_LH0, RAN_0_NLH, RAN_H_LH0). Here the one using an heuristic which considers all possible movements out of a node for a vehicle is clearly the best.

- Figure 9- 16, Figure 9- 17, and Figure 9- 18 represent respectively solutions based on the ANN, greedy policy, and benevolent counterclockwise flow policy, when different 'look-ahead' strategies (complete, limited, and none) are used. ANN based solutions always improve performance when using a 'look-ahead' strategy. Considering the solutions based on the greedy policy in Figure 9-17, it shows that not using a 'look-ahead' strategy is still the worst solution, in general, but to a less extent than in the ANN type solutions. The complete 'look-ahead' strategy performs the best and the limited 'look-ahead' strategy is surpassed by the solution without a 'look-ahead' when eight or more vehicles are in the system. The solutions based on the benevolent counterclockwise flow policy (Figure 9- 18) are all three very similar, with the solution not using a 'look-ahead' strategy performing the best when six or more vehicles are in the system.

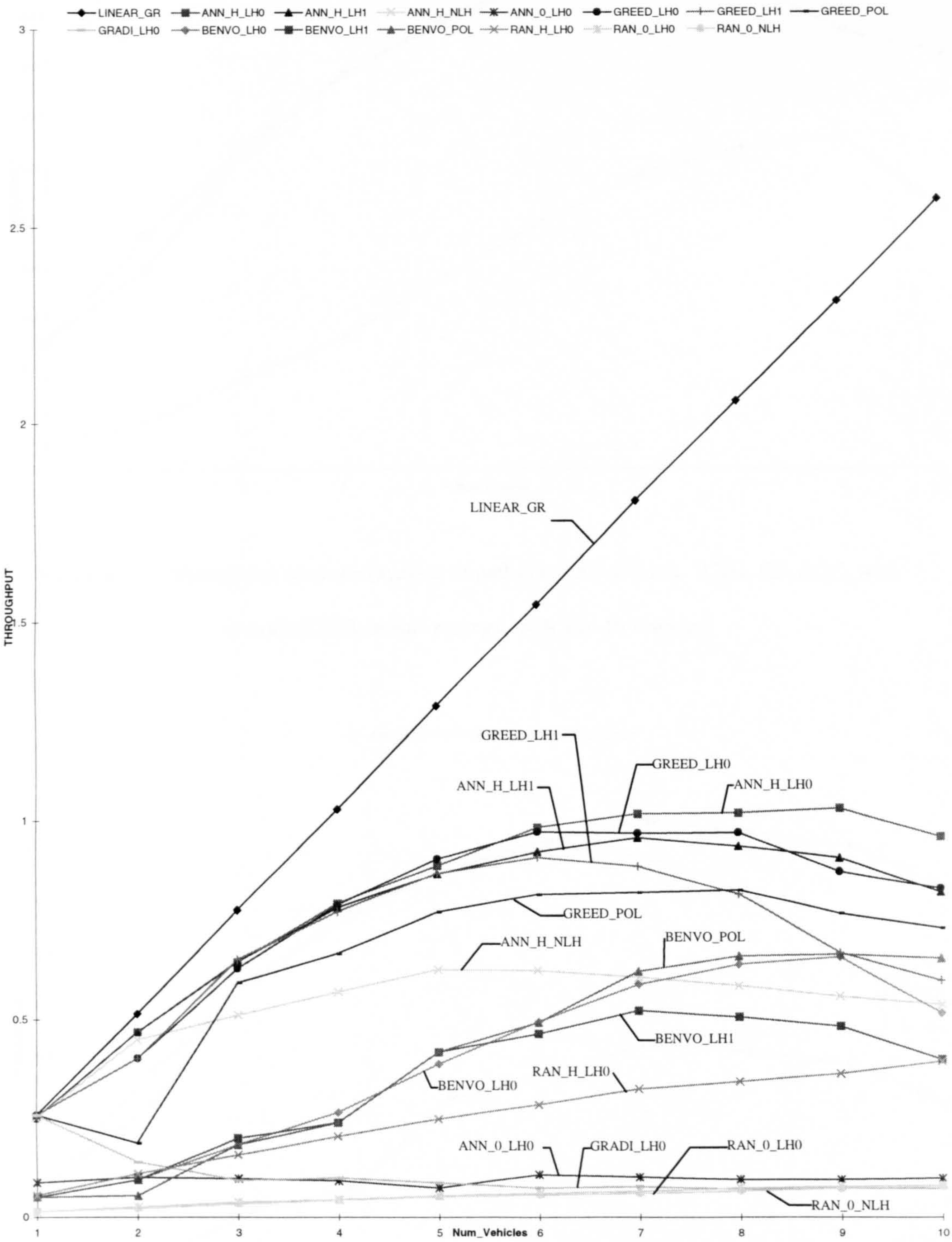


Figure 9- 10: *'throughput'* measure (number of paths per unit of time). All strategies, with 1 to 10 vehicles.

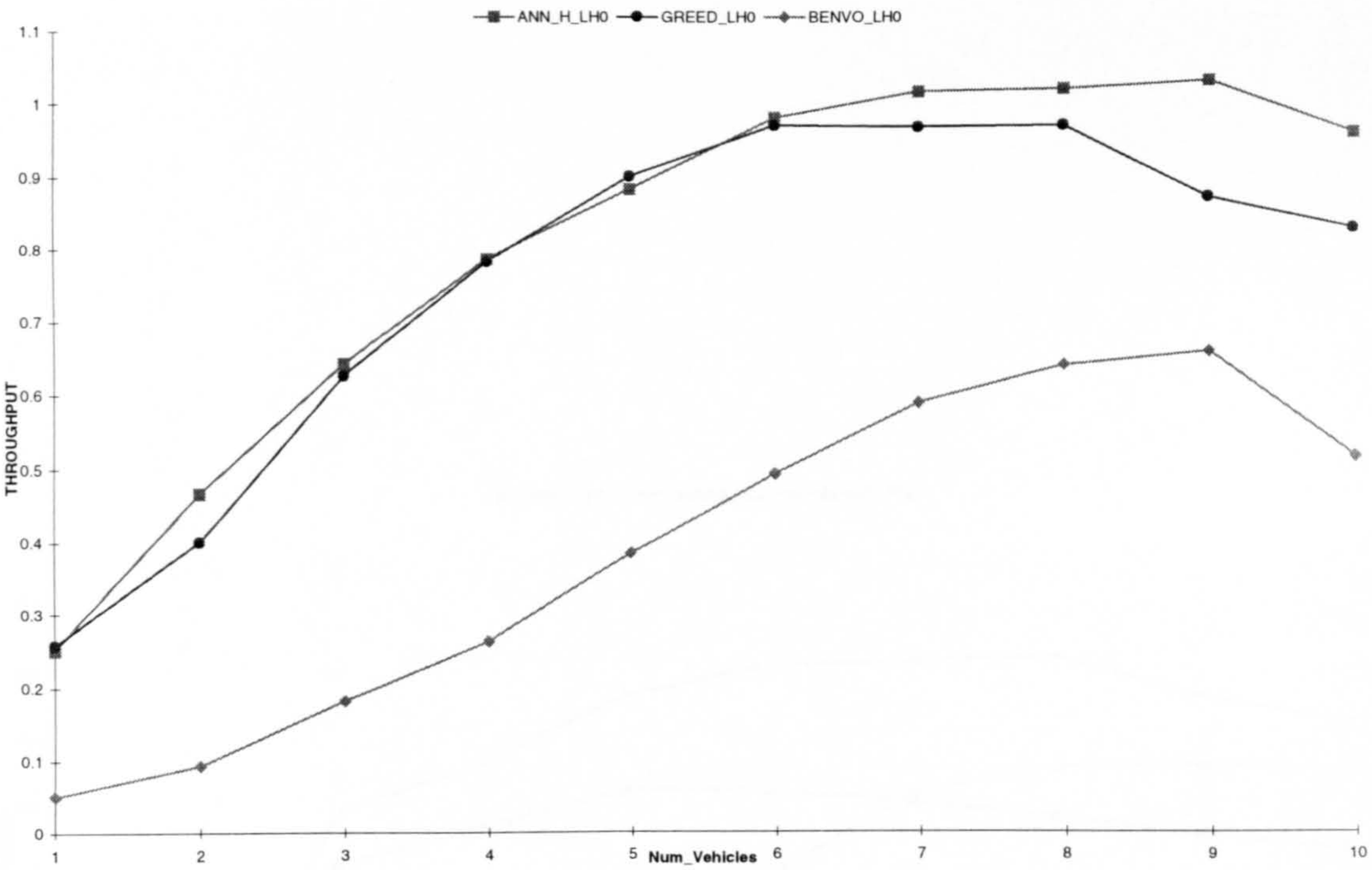


Figure 9- 11: *'throughput'* measure (number of paths per unit of time). ANN, GP, BCC, **with complete** 'look-ahead' strategy, with 1 to 10 vehicles.

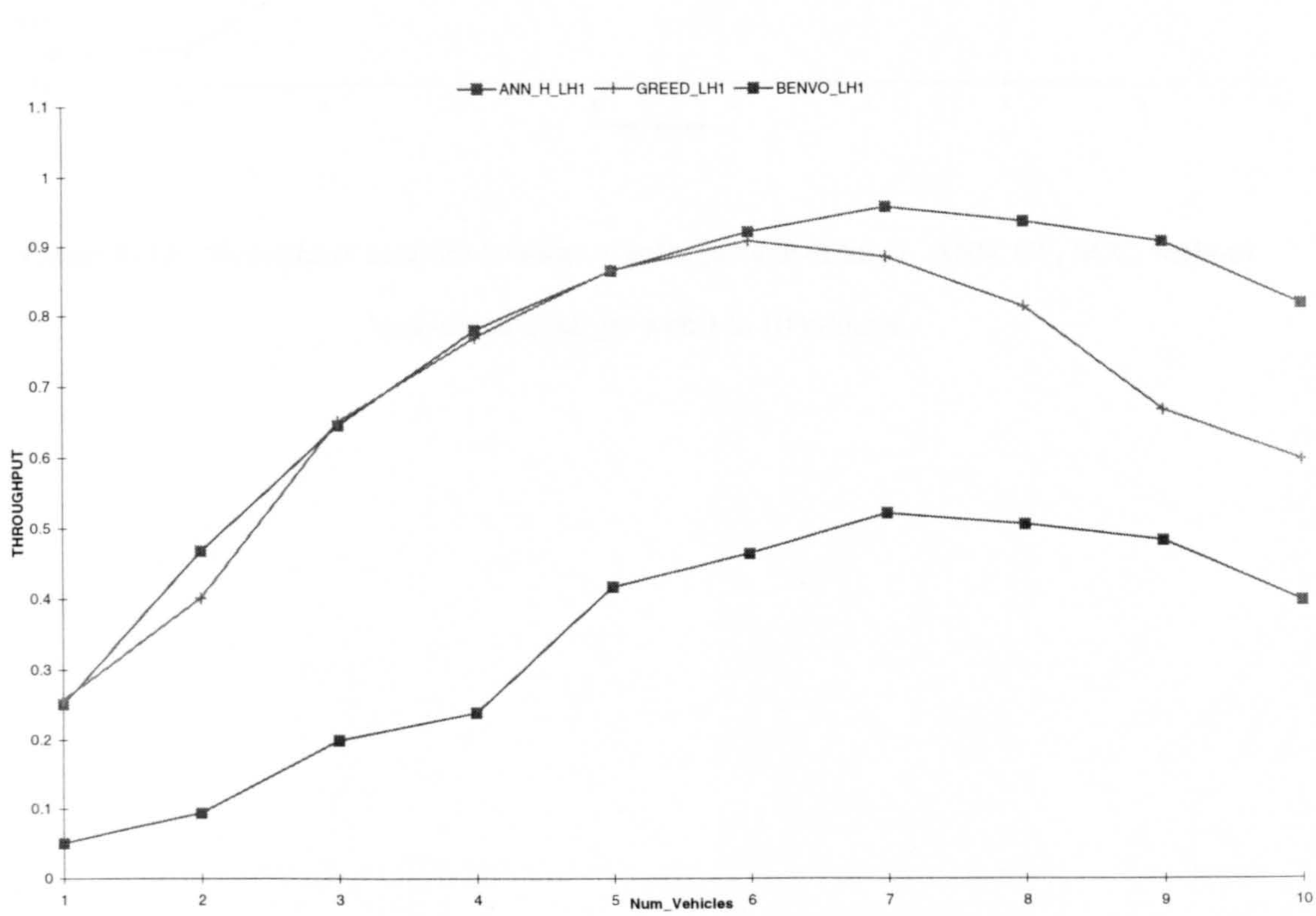


Figure 9- 12: *'throughput'* measure (number of paths per unit of time). ANN, GP, BCC, **with limited** 'look-ahead' strategy, with 1 to 10 vehicles.

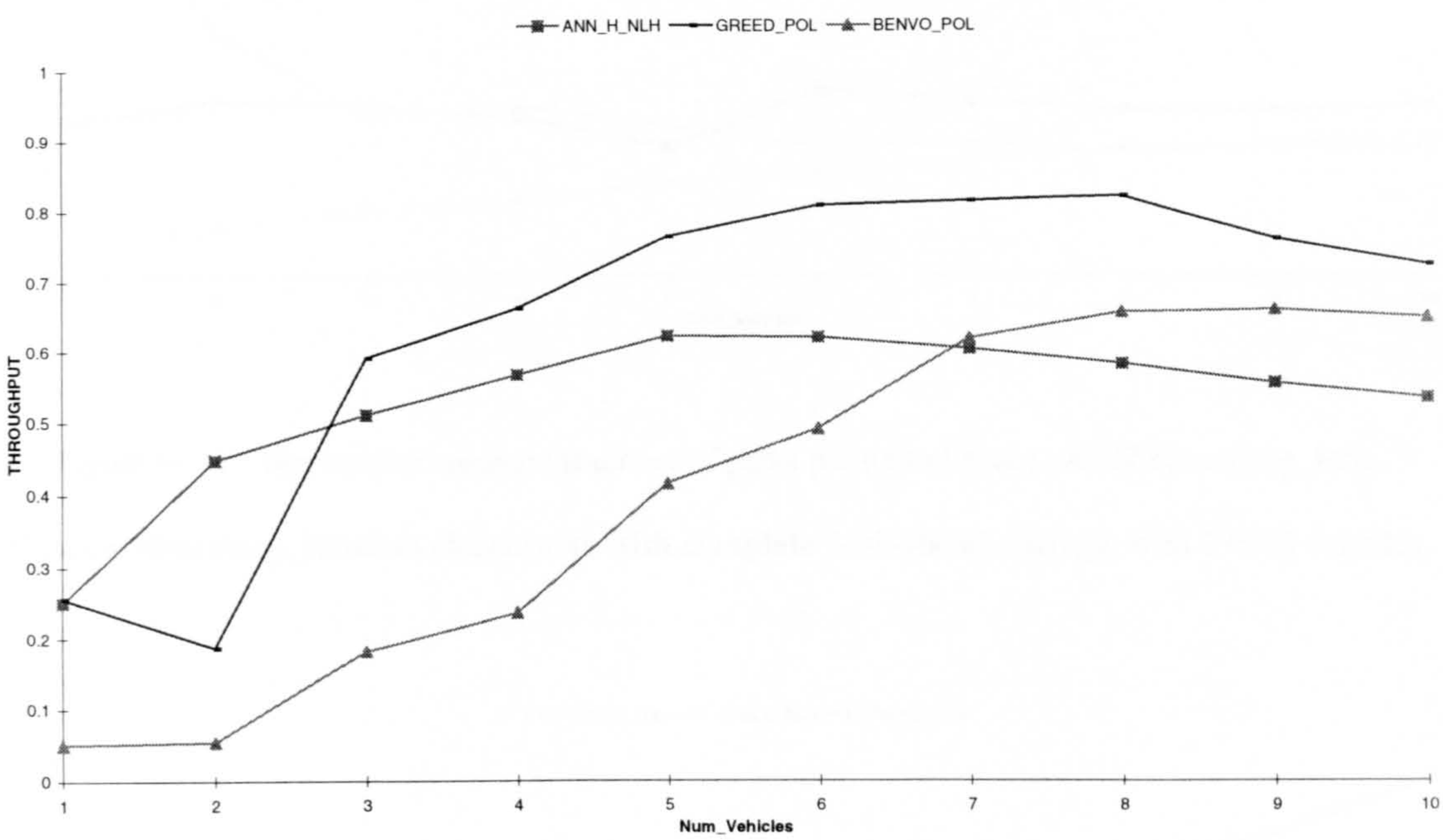


Figure 9- 13: *'throughput'* measure (number of paths per unit of time). ANN, GP, BCC, **without** 'look-ahead' strategy, with 1 to 10 vehicles.

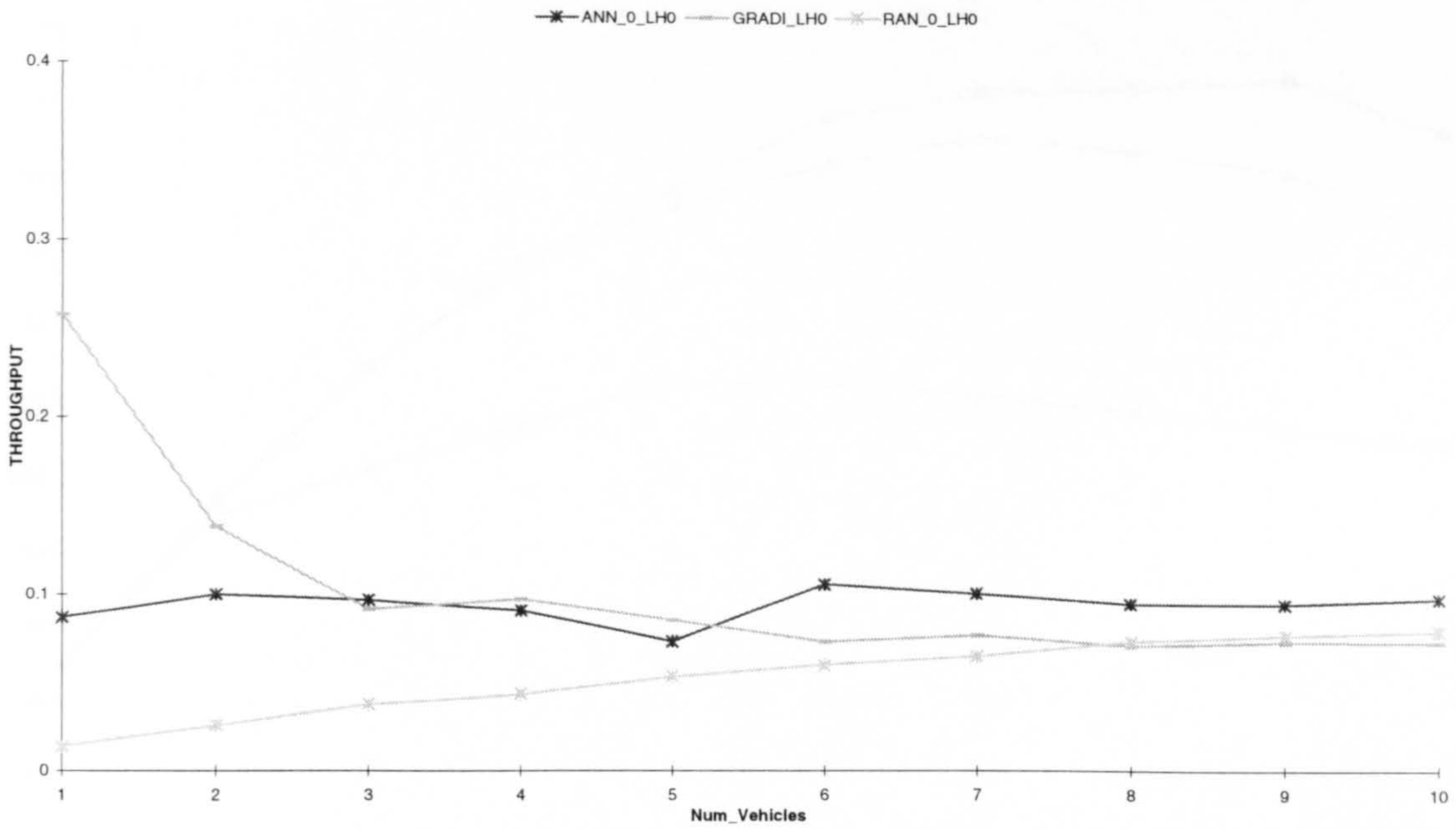


Figure 9- 14: *'throughput'* measure (number of paths per unit of time). ANN direct/stop, Pure Gradient direct/stop, Random choice/stop, **with complete** 'look-ahead' strategy, with 1 to 10 vehicles.

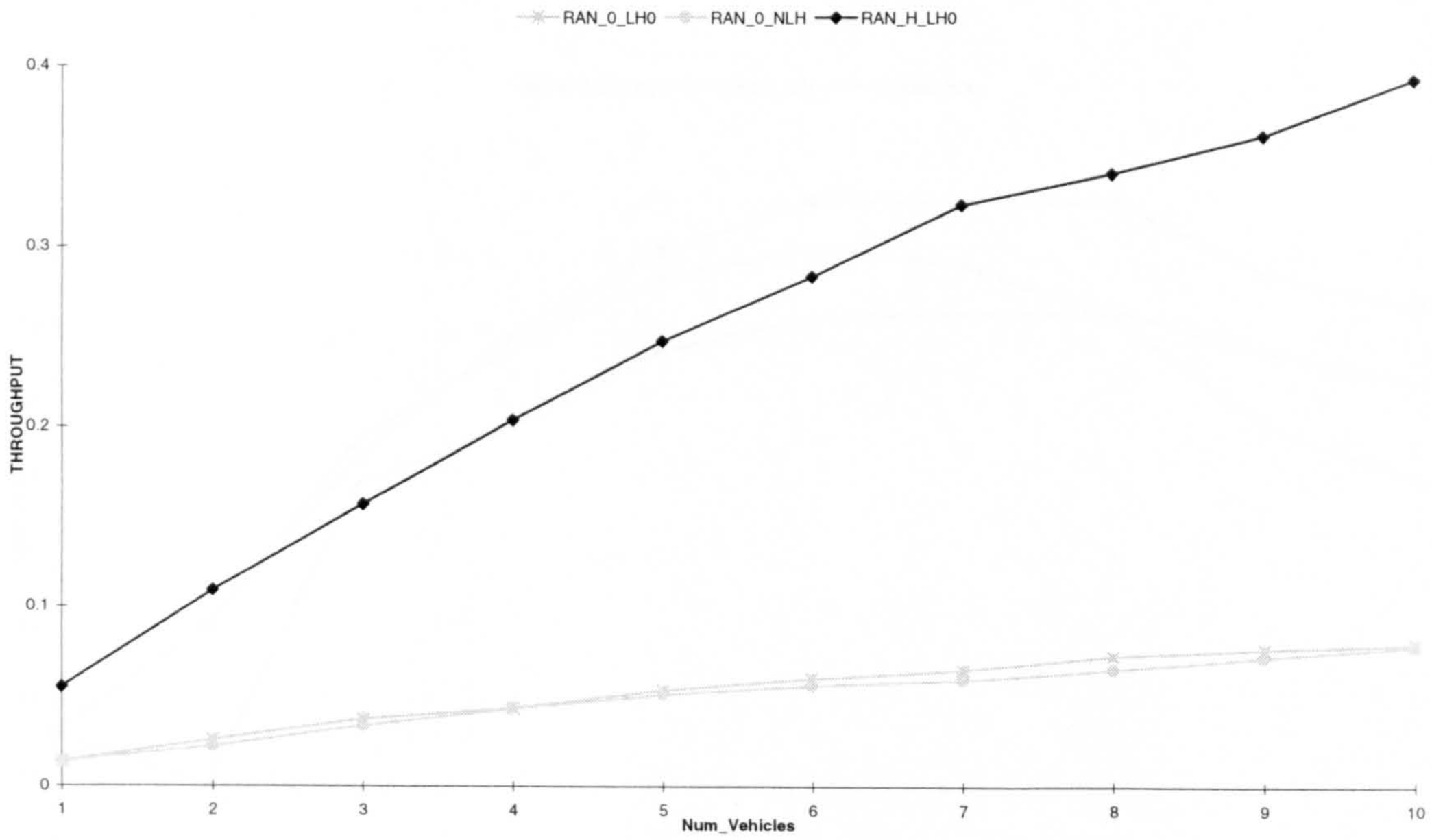


Figure 9- 15: *'throughput'* measure (number of paths per unit of time). Solutions based on random choices, with 1 to 10 vehicles.

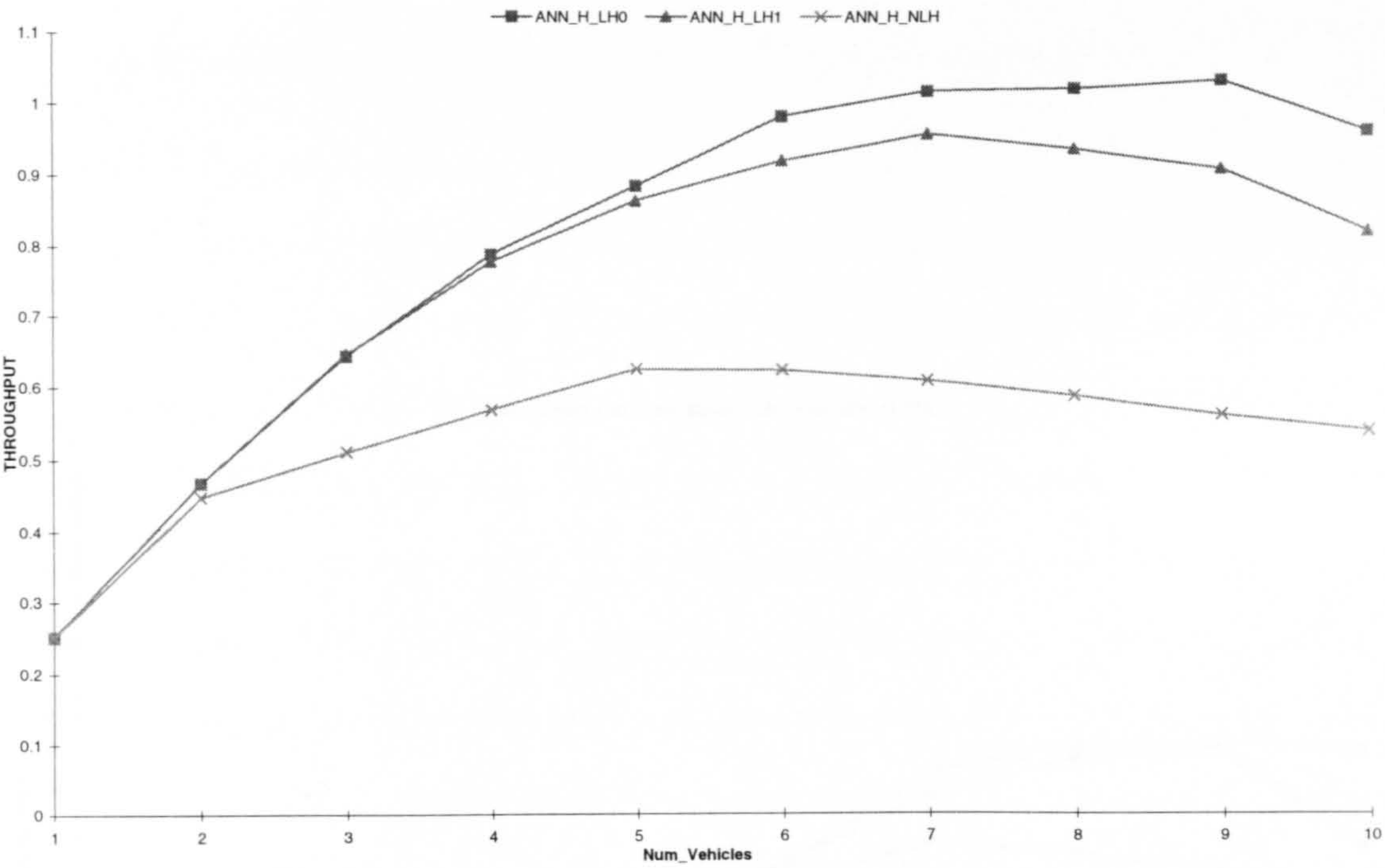


Figure 9- 16: *'throughput'* measure (number of paths per unit of time). ANN based solutions (**complete, limited, and no** 'look-ahead' strategy), with 1 to 10 vehicles.

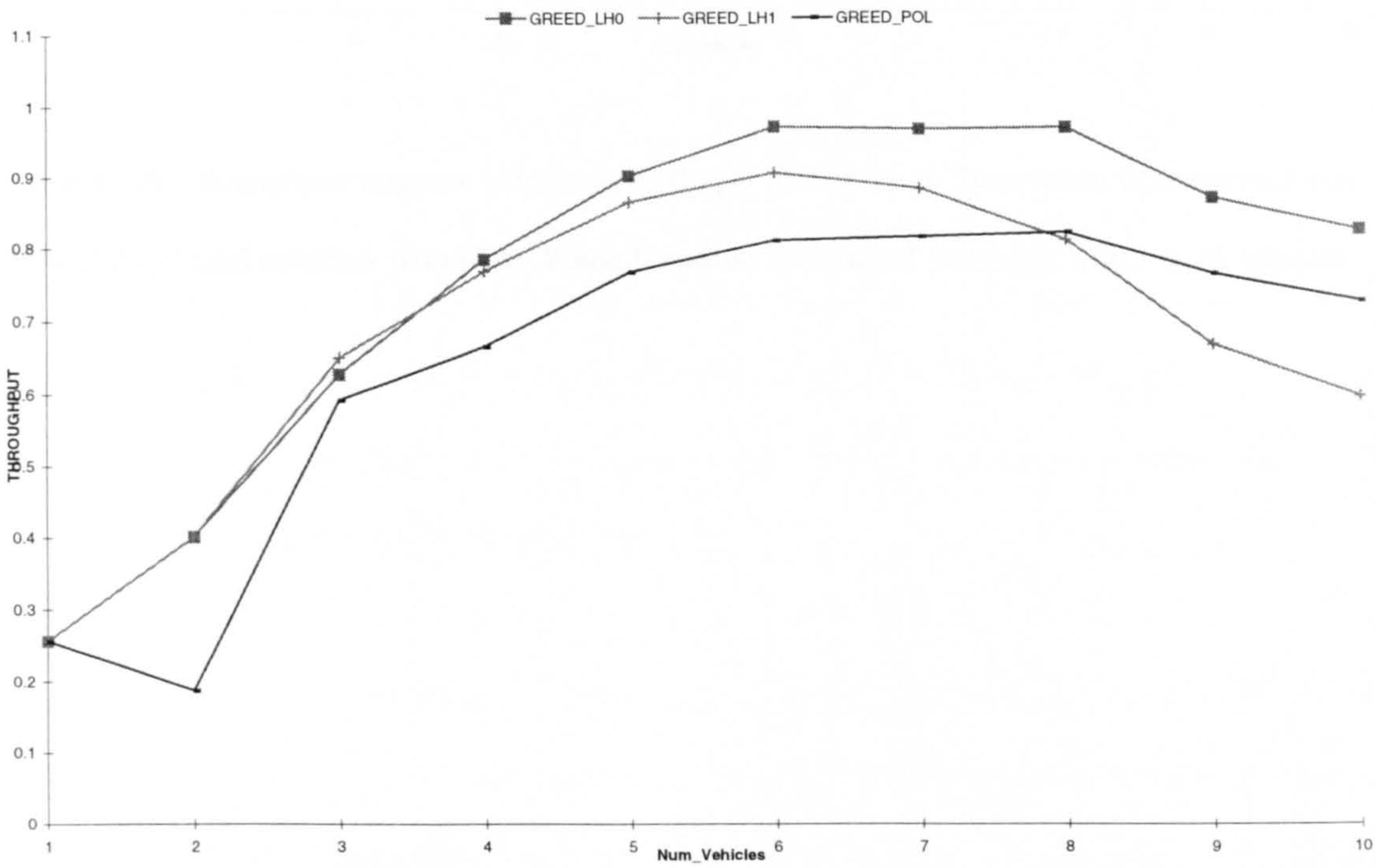


Figure 9- 17: *'throughput'* measure (number of paths per unit of time). Greedy Policy based solutions (**complete, limited, and no** 'look-ahead' strategy), with 1 to 10 vehicles.

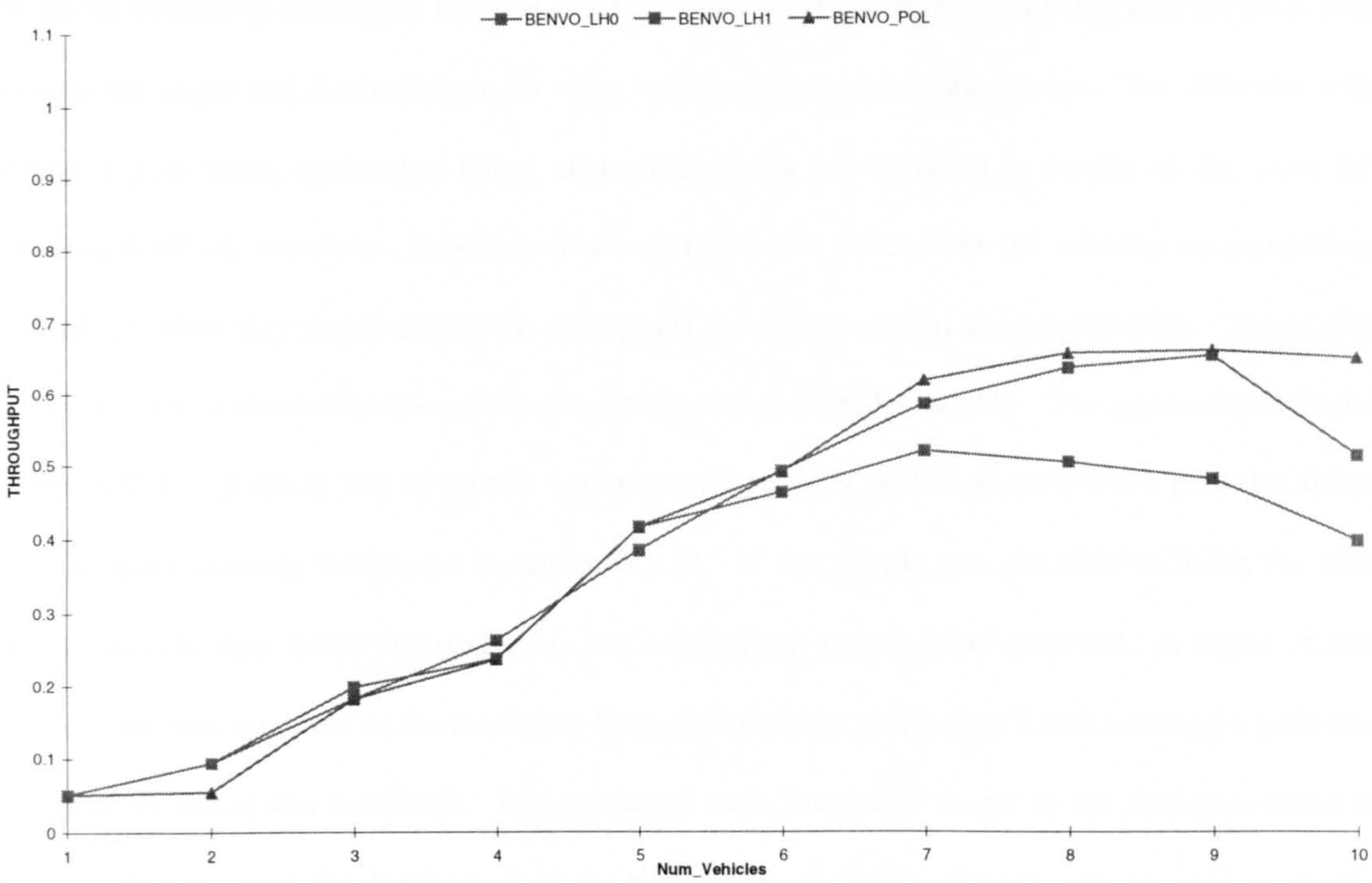


Figure 9- 18: *'throughput'* measure (number of paths per unit of time). Benevolent Counterclockwise Flow Policy based solutions (**complete, limited, and no** 'look-ahead' strategy), with 1 to 10 vehicles.

9.3.3 Graphs of number of bad solutions (*'bad paths'*)

None of the strategies implemented so far can guarantee that a path will always be found because they do not implement an exhaustive search algorithm capable of investigating all possible alternatives and thus deciding if a path exists or not. Such an approach is limited by the size of the problem. Therefore we need to investigate how the strategies tested perform in terms of being able to find a path between the origin and destination nodes when various vehicles are in the system. The difficulty with finding a path when approaches based on heuristic rules are involved is related to the need for detecting deadlock situations. Deadlock situations may occur either when the vehicles are completely blocked, or when they move around the same paths and would remain there indefinitely. These may happen in various circumstances and are not always easy to identify quickly. The approach employed to deal with this problem was to specify a maximum time for a vehicle to complete a path that it has been assigned (already introduced in section 8.1.1). If the vehicle was not able to finish the path within that time then it was assumed some sort of deadlock situation had occurred. A figure of 200 units of time was specified as the maximum time allowed after which time it was assumed a path was not found, or that it was unsuitable. The number of such *'bad paths'* found by the strategies tested in these conditions, and the distribution of the times taken to find each path, can be used to decide whether the approach deserves consideration or not, as a valid solution in a real system. The 200 units of time were arbitrarily chosen only to allow enough time to evaluate the performance of the solution over a sufficiently long period of time; taking into consideration the maximum distance of 8 units of time (8 tracks) between the nodes in the layout used (Figure 9- 1) without traffic congestion. In a real case other maximum time limits could be imposed, and other methods used to identify and solve deadlock situations which could occasionally occur.

The following histograms in Figure 9- 19 show, for each of the strategies tested, the number of '*bad paths*', i.e. paths not found in less than 200 units of time. They present separately the total number of '*bad paths*' found when each strategy was tested with 1 to 10 vehicles at a time. Each strategy will have a maximum of 10 columns if '*bad paths*' were found in each of the cases tested while varying the number of vehicles. Figure 9- 20 also show the same information, that is the total number of '*bad*

paths', as in Figure 9- 19 but without separating out the cases for each strategy on the basis of the number of vehicles in the system.

The major conclusions from an analysis of the information contained in these histograms can be summarised as follows:

- it is possible to distinguish the cases which clearly perform worse by a factor of no less than 10, and these correspond to strategies which only consider two alternative decisions at each node: move along the first selected direction, or else stop. Also in general the number of '*bad paths*' in these cases increase with the number of vehicles in the system.

- there is a group of strategies which perform the best with none, or less than 5 '*bad paths*', which includes the solutions based on ANN with heuristics

- the number of '*bad paths*' in the group of solutions based on the 'greedy policy' and the 'benevolent counterclockwise flow policy' is of some significance, ranging from 41 to 135, when compared with the best solutions. Also in these solutions, the number of '*bad paths*' does not always increases with the number of vehicles in the system. Particularly with 2 vehicles in the system the 'greedy policy' type solutions seems to perform rather badly.

In a more detailed analysis of the worst cases it can be seen that the solutions based on random choices (RAN_0_LH0 and RAN_0_NLH) perform better than the pure gradient (GRADI_LH0), which is the one performing worst, and the solution based on the ANN (ANN_0_LH0). This might be explained by the lower number of total paths executed by the solutions based on random choices, and consequently with longer paths, when compared with the pure gradient and the solution based on ANN. If the number of '*bad paths*' was presented as a (%) of the total paths executed then the solutions based on random choices would become the worst:

- RAN_0_NLH = 30 (%); RAN_0_LH0 = 26 (%)

- GRADI_LH0 = 21 (%)

- ANN_0_LH0 = 20 (%).

When directly comparing the solutions based on the 'greedy policy' and the 'benevolent counterclockwise flow policy' it can be seen that the solutions based on the 'greedy policy' always perform better in terms of number of '*bad paths*':

- GREED_LH0=41; GREED_LH1=69; GREED_POL=41

- BENVO_LH0=80; BENVO_LH1=135; BENVO_LH0=77.

Observing more closely the results of the greedy policy type solutions all '*bad paths*' found when the number of vehicles in the system is 6 or less correspond to the 2 vehicles cases. This particular behaviour is even more visible in the greedy policy (GREED_POL) cases in which all 41 '*bad paths*' were found when 2 vehicles were used.

The strategies using the ANN together with the heuristic to allow for more alternatives to be considered at each node are the ones with a higher success rate of completed paths in terms of arriving at the specified destination nodes. The ANN with 'look-ahead' (ANN_H_LH0) has found all paths. The ANN solution with limited 'look-ahead' strategy (ANN_H_LH1) only has 5 '*bad paths*' and all with 10 vehicles in the system. The ANN without 'look-ahead' strategy (ANN_H_NLH) only has 1 bad path when tested with 4 vehicles.

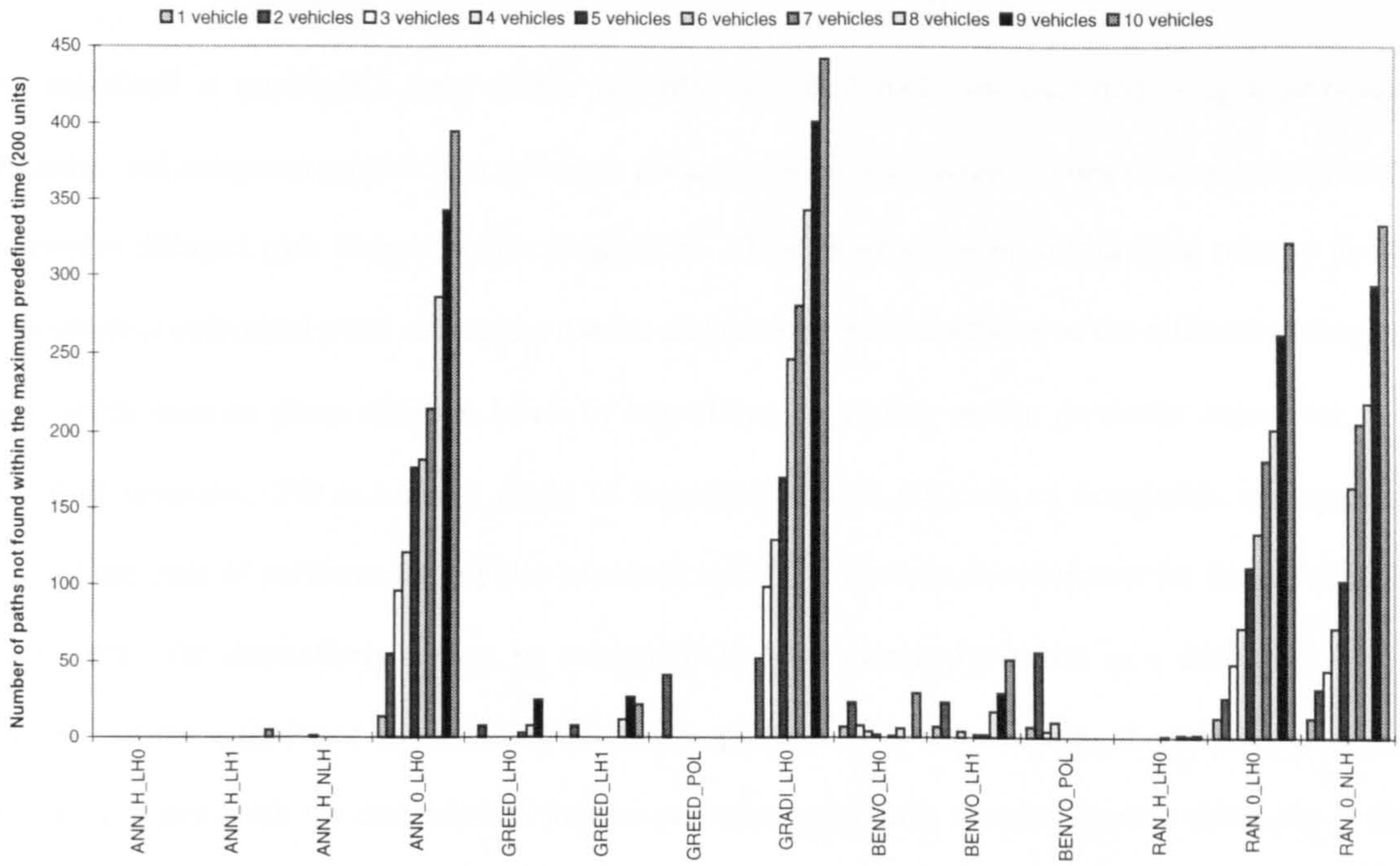


Figure 9- 19: Total number of bad solutions for each of 1 to 10 cases. All strategies.

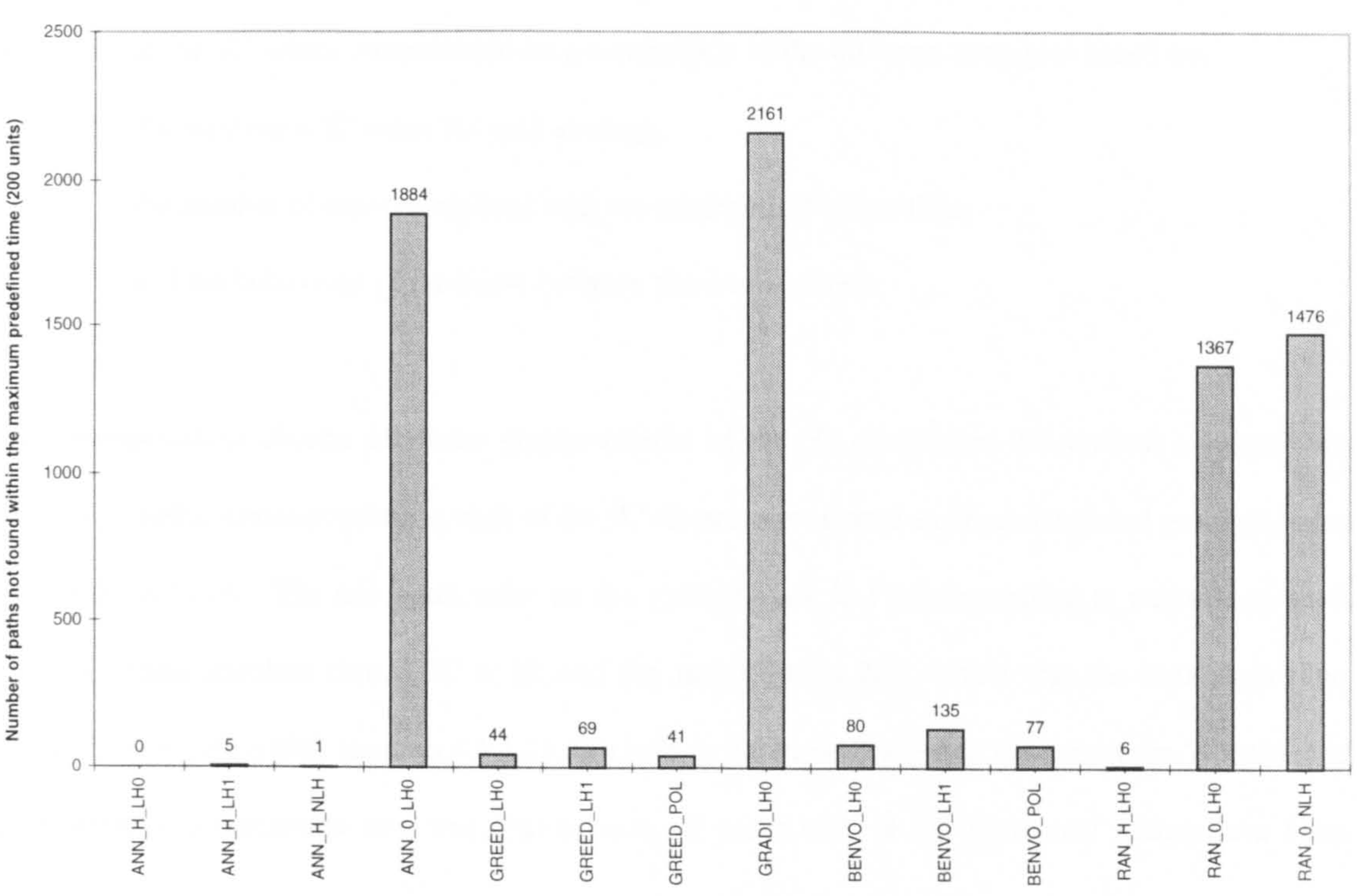


Figure 9- 20: Total number of bad solutions on all 1 to 10 vehicles cases. All strategies.

9.3.4 Graphs of cumulative distribution of *'K'* ratio

As explained in section 9.2 these graphs describe how each path was executed using a particular strategy, and compares relative to a reference lower bound time duration, corresponding to each one's minimum distance path without traffic congestion. These graphs contain information relative to the execution of individual paths and can be used to characterize various aspects of the different strategies; and which may be given different levels of importance depending on the particular objectives in a practical situation. For example it might be important to have not only an acceptable aggregate or overall measure of performance but also to expect small variations in the estimates for the duration of each path. Or alternatively it may be acceptable to have greater variations in a few cases if the duration of the majority of the cases is close to an optimum. When considering the interaction of the transport system with the allocation of jobs to machines scheduling system, smaller variations in the times to execute paths seems important in order to reduce the probability of disruption in the scheduling system. However the exact values on which different alternatives must be considered depends on the particular implementation and its objectives. Therefore the analysis of the graphs representing the *'K'* values concentrated on a comparison of the different strategies based on:

- the maximum *'K'* value for each strategy,
- the number of cases completed with the minimum absolute time,
- and the behaviour of the curve between these two values.

The representation chosen for these graphs consist of the *'K'* classes on the vertical axis and the number of paths, corresponding to each of the *'K'* classes represented as an accumulated percentage on the horizontal axis. The minimum value on the vertical scale is 1 corresponding to cases completed with minimum absolute time (*'K'* = 1); and the maximum is 200, which was the maximum time allowed to execute a path (section 8.1.1.1). To help in the identification of the maximum *'K'* value for each strategy a horizontal line was drawn when all cases (100 % on horizontal scale) have been classified.

As in the presentation of previous graphs it was decided first to include all 14 strategies tested in a graph, and the curves were then selected to be observed in more detail, separately, using a more

suitable scale. A set of all the graphs considered is included in Appendix E. The ones containing all strategies are presented first and are followed by the graphs with complete 'look-ahead', limited 'look-ahead', and no 'look-ahead' strategies. A selection from these set of graphs corresponding to cases with two and three vehicles is presented in the following figures (Figure 9- 21 to Figure 9- 24). The first two represent all strategies while the other two represent the complete 'look-ahead' strategies in a more suitable scale.

From an analysis of the graphs containing all 14 strategies it is possible to distinguish three main types of solutions:

- a group of curves which differ significantly from the others by almost always presenting the highest maximum *'K'* values, lowest percentages of paths with *'K'* = 1, and with the remaining values on the curves spreading more clearly. These curves (RAN_0_NLH, RAN_0_LH0) correspond to the strategies which generate its movement decisions based on random numbers and which consider only one of two alternative movement decisions at a node: move along the selected direction or stop.

- a second group of curves which also present, in general, maximum *'K'* values close to or equal to the highest observed, but with a much higher number of paths executed with *'K'* = 1, and also a smaller dispersion of the remaining values on the curves when compared with the two curves previously analysed. This group consists of the other two strategies: the pure gradient (GRADI_LH0) and one based on the ANN solution (ANN_0_LH0), that only consider one of the alternative movement decisions at a node.

- the remaining strategies can be included in a group of curves which have in common mainly the spread of the curves and which is clearly more concentrated than either of the two previous groups. The figures for maximum *'K'* values do not always present a common pattern but can vary significantly, i.e. from the minimum to the highest possible, depending on the number of vehicles and on the strategies. In general the number of cases with *'K'* = 1, also varies but to a lesser degree than the maximum *'K'* values. This group of curves includes what can be considered to be the best curves in terms of the number of cases with *'K'* = 1 and in terms of the lowest of maximum *'K'* values. The strategies based on the ANN with complete and limited 'look-ahead', present in general the lowest, or close to the lowest of the maximum *'K'* values. They are occasionally outperformed by the "greedy policy" (GREED_POL), and once by the "benevolent counterclockwise flow policy" (BENVO_POL),

but these perform very poorly (with the maximum possible *'K'* value) when the number of vehicles in the system is equal to 2. The curve representing the strategy which is based on random decisions with heuristics (RAN_H_LH0) is also included in this group and presents what can be generally considered to be the worst of the solutions in this group, either in terms of number of cases with *'K'* = 1 and the spread of the other values on the curve. In terms of maximum *'K'* values it performs with high values in general, but these are not always the highest.

Observing in more detail the solutions based on the ANN with heuristics, the greedy policy and the benevolent counterclockwise flow policy, each using a complete 'look-ahead' strategy, the major conclusions can be summarised as follows:

- the solution based on the benevolent counterclockwise flow policy(BENVO_LH0) clearly performs worse in terms of the maximum *'K'* values, the number of cases with *'K'* = 1, and with a greater dispersion of the remaining values on the curve,

- the solutions based on the greedy policy (GREED_LH0) and the ANN with heuristic (ANN_H_LH0) present curves which look very similar in most cases, but with the ANN type solution performing with much lower maximum *'K'* values, except in three situations: when only one or five vehicles are in the system, in which case the solution based on the greedy policy performs the best; and also in cases with six vehicles where both strategies have the same maximum *'K'* value. With an increase in the number of vehicles the number of cases with *'K'* = 1 is marginally higher with the greedy policy but the ANN type solution remains the one with the lowest maximum *'K'* values and with the other values on the curve more concentrated around lower *'K'* values.

Similar conclusions are derived from an analysis of the graphs with these three types of solutions but with limited 'look-ahead' strategy (ANN_H_LH1, GREED_LH1, BENVO_LH1). The ANN type solution performs the best generally except in cases with seven vehicles, where it becomes second best, and with eight vehicles where it becomes only marginally different from the greedy policy, which is the best one.

When these three types of solutions are tested without 'look-ahead' (ANN_H_NLH, GREED_POL, BENVO_POL) then a different behaviour is observed. The strategy based on the ANN continues to

be clearly better in cases with two vehicles in the system, and marginally better in cases with six vehicles. Of the remaining cases the greedy type policy performs the best overall, with the benevolent counterclockwise flow policy marginally outperforming the ANN type solution almost always in cases with more than 6 vehicles in the system.

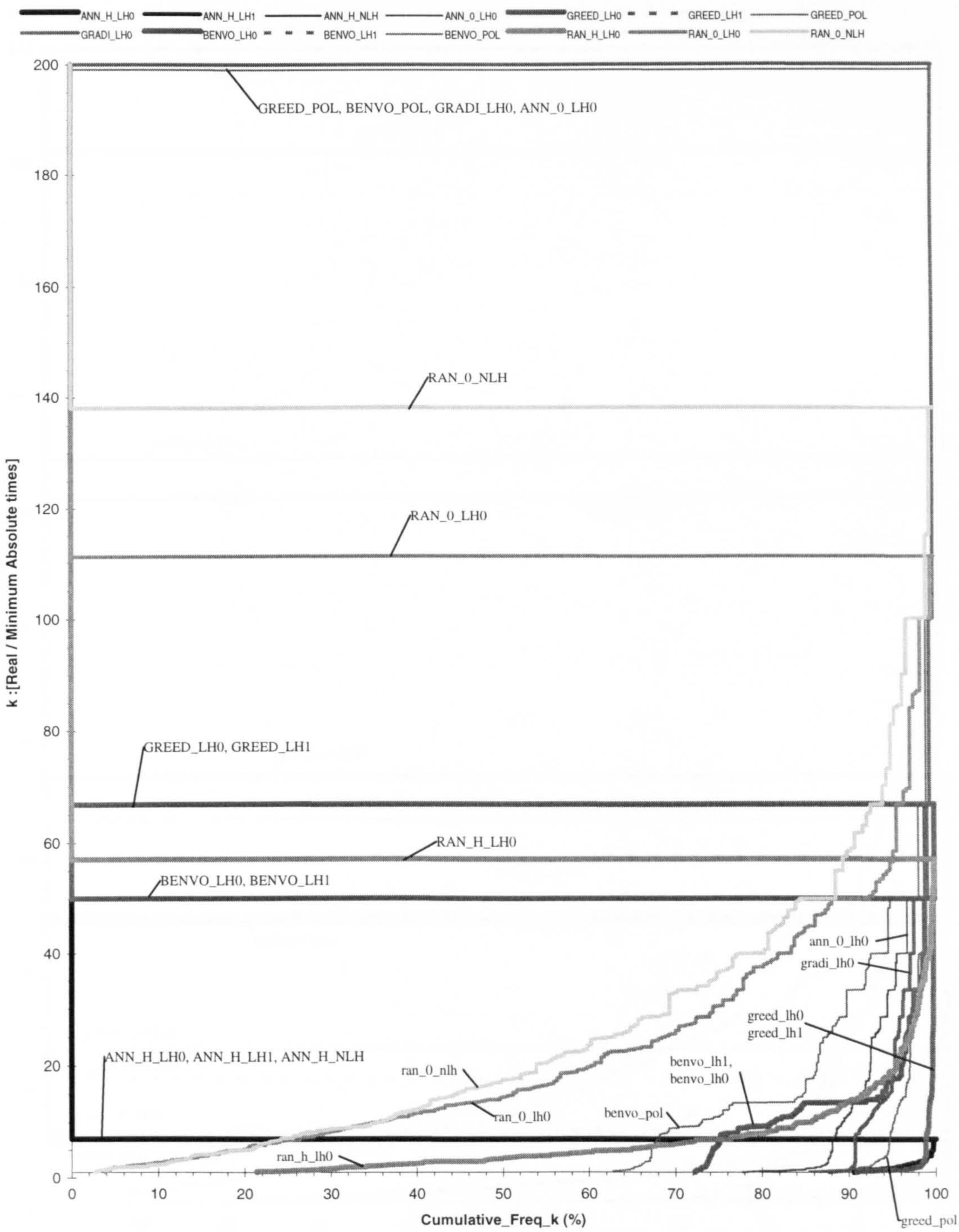


Figure 9- 21: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, two vehicles.

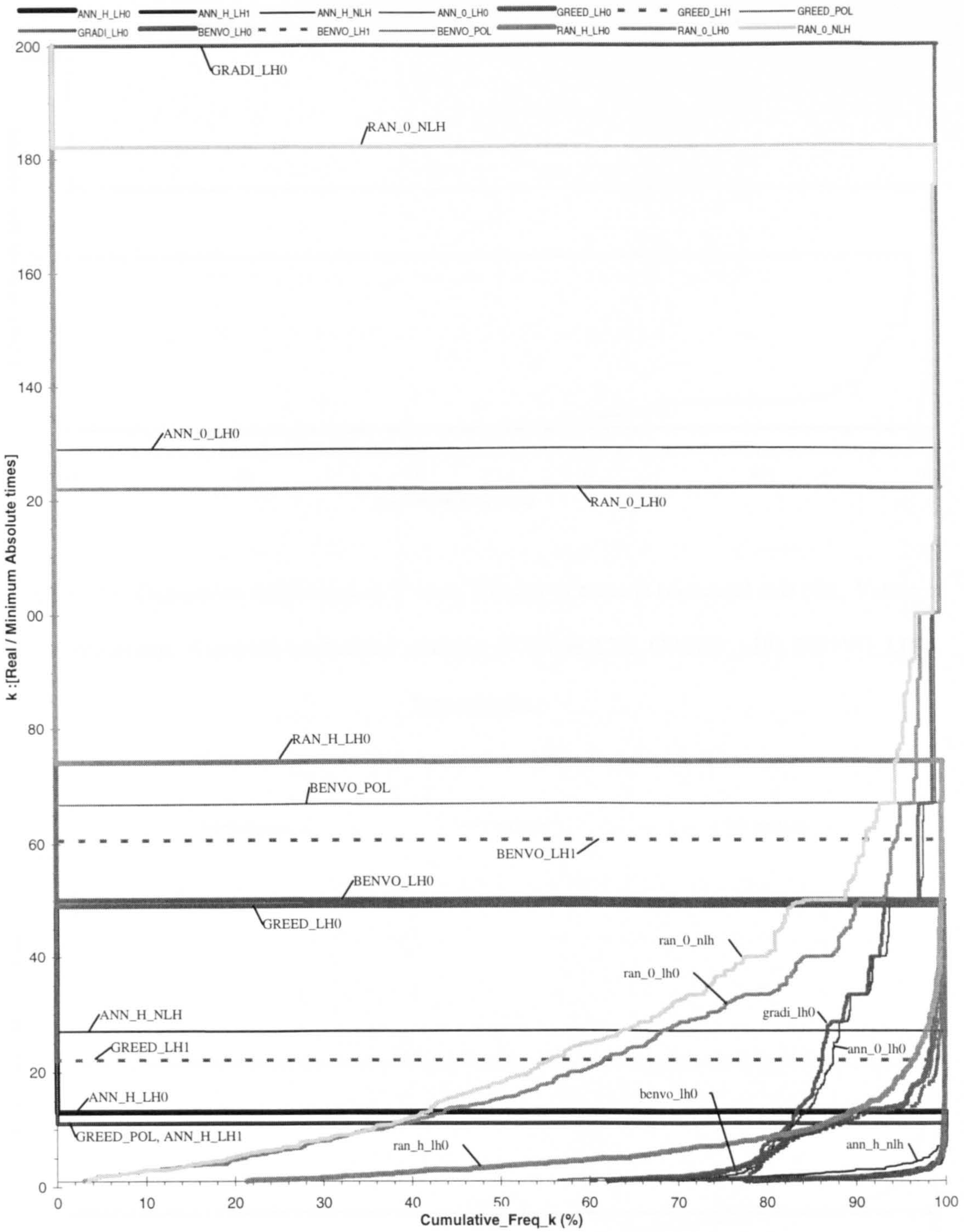


Figure 9- 22: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, three vehicles.

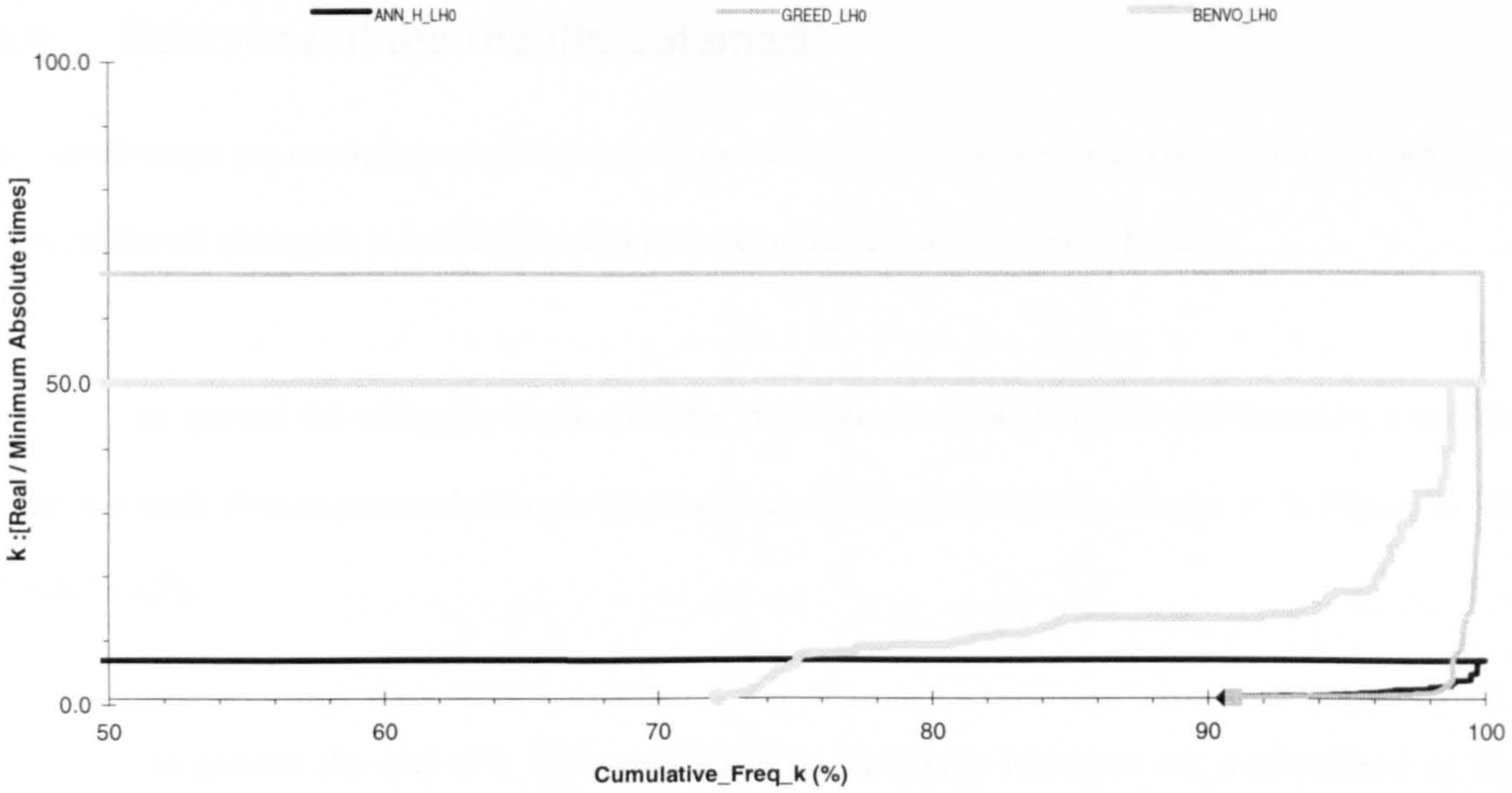


Figure 9- 23: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Two vehicles.

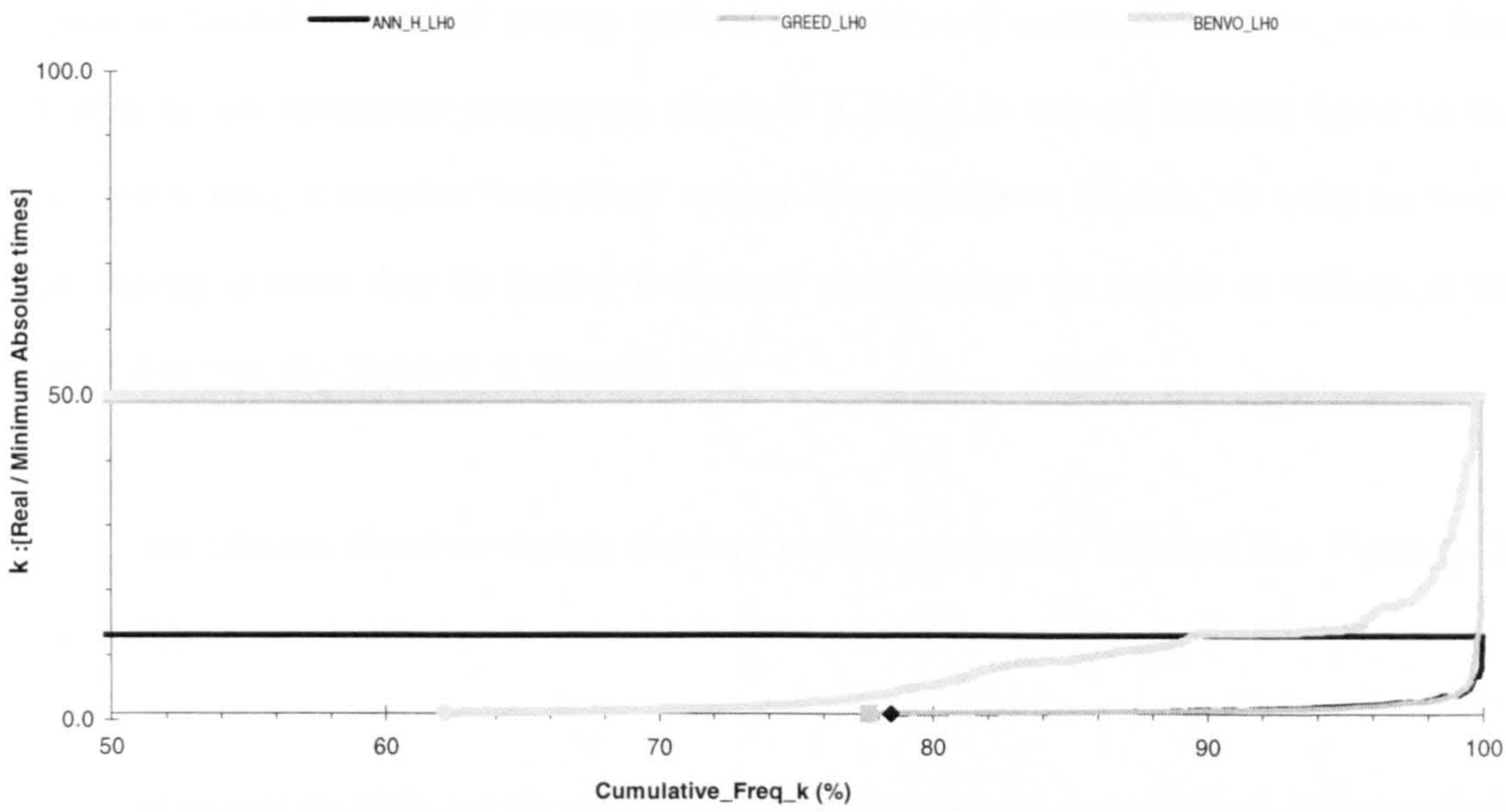


Figure 9- 24: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Three vehicles.

9.4 Overview of the results obtained

In overall terms the results obtained using the four measures to evaluate and compare the performance of the different strategies in a multiple vehicle system can be summarized as follows:

- in general the strategies which consider more than one possibility for movement by a vehicle when at a node always present better performances in all four measures (i.e. Figure 9- 2, Figure 9- 10, Figure 9- 22);

- in general the use of a 'look-ahead' strategy normally increases the performance of the solutions. The solutions based on ANN's always increase performance with an increase of the 'look-ahead' range (i.e. none, limited, complete) (i.e. Figure 9- 7, Figure 9- 16). This consistent improvement in performance is not always realised when using solutions based on the greedy policy or the benevolent counterclockwise flow policy. This is especially so when a high number of vehicles are involved. In these cases, the solutions based on the benevolent counterclockwise flow policy using a complete or limited 'look-ahead' strategy perform no better, and sometimes are even worse than without using any 'look-ahead' strategy (i.e. Figure 9- 9, Figure 9- 18); the solutions based on the greedy policy using a complete 'look-ahead' strategy always performs the best, but using no 'look-ahead' strategy is better than the limited 'look-ahead' strategy when the number of vehicles in the system is 8 or more (i.e. Figure 9- 8, Figure 9- 17);

- the solutions based on random decisions perform consistently the worst (i.e. Figure 9- 2, Figure 9- 10);

- in general the ANN and the greedy policy type solutions with complete 'look-ahead' perform similarly in overall terms and present the best performances. That is except in two situations where the ANN type solution clearly outperforms the greedy policy type solution. One is when the number of vehicles in the system is two, where the ANN performs the best in all four measures (i.e. Figure 9- 3, Figure 9- 11, Figure 9- 19, Figure 9- 23). The other is in relation to the number of paths not found

within the time allowed (i.e. 'bad paths'), with the ANN type solution performing better in general (i.e. Figure 9- 19, and Figure 9- 20),

- when 2 vehicles are involved, the particularly poor performances of the greedy and benevolent counterclockwise flow policies comparatively to the ANN type solutions becomes significantly worse if no 'look-ahead' strategy is used (i.e. Figure 9- 7, Figure 9- 8, Figure 9- 9, Figure 9- 23),

- the benevolent counterclockwise flow policy performs relatively badly in comparison with the greedy type and ANN solutions, but tends to improve when the number of vehicles in the system increases (i.e. Figure 9- 3, Figure 9- 21).

Chapter 10 | Discussion and Recommendations for Further Work

Having presented and tested the approach envisaged for the development of a solution based on ANNs to route the vehicles in an AGV traffic management and control system, it is now possible to evaluate and discuss the results obtained and how they can be extended to a more general and 'real' AGV system. One of the main issues of the discussion is the use of the solution for the single vehicle case as the basis of the solution of the multi-vehicle case. Another issue is related to the applicability of a ANN to model and to solve these two cases (i.e. single and multi-vehicle cases), and the role of the simple heuristics used in a "hybrid" solution. The solutions were developed for a well defined and specific layout, it must therefore be discussed whether they can be applied to different layouts. Furthermore it must be discussed whether the solutions proposed produce results which can be accepted when a 'real' AGV system is considered.

10.1 Using solutions to a simpler problem as the basis of solutions for more complex problems

One of the main characteristics of the approach based on ANN's is the use of the simpler solution for the single vehicle case as the basis of the solution for the multi-vehicle case. One advantage of this approach is that it does not require a search for the optimal solutions in a multi-vehicle case, but at the expense of only being able to produce close to optimal solutions instead of the optimal solutions. As discussed in Chapter 4, finding the optimal solution in a multi-vehicle system is not a trivial problem, even if we are not including time constraints other than obtaining data for training/testing a ANN. The additional option for a vehicle to stop and wait at a node for as much time as desired, significantly

increases the number of possibilities that can be evaluated in the search for the optimum solution(s). The approach developed avoids this problem and produces an alternative that performs competitively relative to other alternatives which are based on finding close to optimal solutions. Furthermore the approach has potential for improvement because data representing global optimal solutions can still be used provided it is available.

Another advantage of this approach is its potential for being used in a larger system, i.e. larger number of tracks and nodes. If the assumption is made that a limited area around the node, where a movement decision is required, contains information capable of supporting a specific movement decision then it is reasonable to expect that such an area can be identified and an ANN trained to identify movement directions for a vehicle at the nodes within that area (i.e. the centre node). As the vehicles move around in this larger layout, the limited area considered would be updated to represent information around the current location of the vehicle requiring a movement decision. It is even possible to consider training different ANNs to account for the particular characteristics of the nodes in the physical layout (i.e. nodes with four, three, or two possible ways out), each to be used in a specific node within that limited area.

The results obtained with the specific layout proved that the approach is feasible and it performs comparatively well in relation to other alternatives. The question now is whether the use of ANNs contribute significantly to the implementation and performance or not. This is the object of discussion in next section.

10.2 The role of ANNs in the approach developed

The use of ANNs within the approach developed sought after a suitable and convenient form which allows for the consideration of more information than just that relating to the elements in the immediate vicinity of the node where a movement decision is required. In the systems implemented this information represents the status of the tracks (i.e. available or not available) in the physical layout, and was used as an indication of a vehicle's desire to move along a specific track. As was pointed out in the previous section, this approach cannot be considered as an example of a strategy

which bases its decisions on a complete analysis of the entire system with the objective of arriving at an overall optimum solution. It is more appropriate to include the approach developed in the type of strategies which are based on the use of heuristic rules that are applied equally to every vehicle in the system. However the use of ANNs provides an extension to these strategies, because it is now possible to think in terms of the ANNs decisions being made having to account for a more complete characterization of the system. That is the vehicle's location or intention to use specific tracks and the knowledge of the best decision at that instant for that vehicle to move along an individual optimum route. If ANNs can be designed to perform as expected then they are a crucial element to support the implementation of this approach.

One of the major problems faced with using ANNs to model and solve the single vehicle case, was the difficulty in designing an ANN that always produced a movement decision at its output which was feasible within the physical layout restrictions (i.e. not leading to a non-existant or temporarily unavailable track). This issue is intrinsically linked to the representation of the problem onto the ANN and one solution was not found which proved more effective than the one based on a step-by-step definition of the complete route for the vehicle (ANN in Appendix B). The main problem can be related to the number of input/output elements in the ANN which do not change state in each case, making it difficult to distinguish between different patterns. The step-by-step representation significantly improved the capability of the ANNs to find correct complete paths, achieving 66% correct paths identified in the testing data (see Chapter 7). It must be noted that these results were obtained with ANNs trained with a reduced set of data and it can therefore be expected that using a larger set of data they can be improved. The same ANN was able to find 87% of the paths linking all possible combinations of nodes defined in the physical layout, assuming all tracks were available. This improvement in performance when all tracks are available revealed the capability of this ANN to recognise the locations of the origin and destination nodes more easily than to distinguish between the state of the tracks from 'available' to 'unavailable'. This is related to the less effective contribution of the ANN input elements associated with the tracks which are set as unavailable', relative to all the input elements. Although using a larger training data set would help the ANN to improve its performance when tracks are set unavailable, achieving a 100% effectiveness remains a very difficult task for a ANN based on this representation. In order to take advantage of the performance achieved

by this ANN it will always be required that its movement decisions are analysed to account for their feasibility. This was achieved through the implementation of a system external to the ANN which uses simple rules to complement the ANN decisions. The objective of these rules was always to favour the selection of the movement direction associated with the ANN output vector, or one of its co-ordinates, and only if those movement decisions could not be implemented would the remaining movement alternatives be considered. While the results obtained using the ANN on its own were not conclusive in relation to a 100% effectiveness, when used in this "hybrid" solution it was possible to solve the single vehicle case in 100% of the cases tested (see in Chapter 9 performance measures of ANN solutions with "heuristics" and 1 vehicle). This success of the "hybrid" solution provided the support for proceeding with the approach in a multi-vehicle case.

The discussion of the approach and results obtained in a multi-vehicle case must account for the fact that, the optimum solution was not available and therefore the evaluation was based on comparing performances of different strategies (Chapter 8) used in similar conditions (Chapter 9). Another aspect in the approach to the multi-vehicle case was the development and use of a strategy which analysed, the implications of allocating the intended movement of a vehicle based on the movements of the other vehicles. The objective was to try to bias the search for a global optimum in the approaches where vehicles decide independently one at a time. With this purpose a "look-ahead" strategy (Chapter 8) was derived and tested. The discussion of the approach based on ANNs and the approaches used for comparison must therefore account for the effect of using this strategy.

The results obtained proved that in general using the "look-ahead" strategy better performances are obtained. To a certain extent this might be expected because it provided for the co-ordination of the movement of the vehicles with interactions between them instead of each vehicle deciding independently. Furthermore it also allowed vehicles which were one track away from the respective target nodes to have higher priorities to terminate their paths.

The use of a "look-ahead" strategy proved particularly beneficial for solutions based on the ANN with heuristics (i.e. "hybrid solution", Chapter 8). This might be explained by the way possible collisions are dealt with in these different cases. The ANN only has information relative to track availability,

and not of nodes occupancy. It is therefore valid from the perspective of the ANN that two vehicles decide to move along two available tracks but leading to the same node. To avoid this collision in the "look-ahead" strategy provisions are made to allocate the movement to the higher priority vehicle, and stop the other vehicle at its current node. Furthermore a vehicle which terminates its path with that movement is given higher priority. When no "look-ahead" strategy is used, a check is made on the validity of the ANN choice based not only on the selected track availability but also on the occupancy of the node at the end of that track. If the movement is not possible then the remaining movement alternatives are considered as described in Chapter 8. The "look-ahead" strategy can then be seen as been more favourable to the decisions based on track status and as such is more closely related to the knowledge expected from the ANN. It must not be forgotten however, that the heuristics required to restrain the direction associated with the ANN output vector are essential to produce feasible movement directions. The strategies which always consider various movement alternatives always performed better than the ones which only consider one alternative (i.e. "pure gradient" and ANN without heuristics, Chapter 8).

It therefore seems reasonable to consider that the "look-ahead" strategy acts more as an extension of the ANN principles rather than providing for its limitations in finding the correct decisions and its performance can thus be associated with the use of an ANN in an "hybrid solution". When considering a 'real' AGV system it is more likely that the tracks will not be of the same length, or that the vehicles might be slightly delayed, and thus making the representation of information in the ANN relative to the node occupancy of lower importance than the track status.

It was expected that the ANN solution could take advantage of its knowledge of a larger area of the physical layout to clearly outperform the other strategies. However the results obtained were not very conclusive. When comparing the best strategies, i.e. the ANN with heuristics ("hybrid solution") and the "greedy policy" (Chapter 9), both with a "look-ahead" strategy, the ANN based solution is only marginally better. While this may be seen from one perspective as proof that a ANN based solution can be used as part of a valid and best approach, on the other hand the question might be raised as to whether the ANN has just learned the "greedy policy". To be more conclusive about this question

would require the use of a larger data set to train the ANN in order to try to achieve a higher performance from the ANN and also testing different physical layouts.

Testing different layouts would also prove useful to clarify another issue related to the generally poor performance of the approach based on the implementation of a consistent direction of traffic flow (i.e. "benevolent counterclockwise flow policy", Chapter 8 and 9). It is clear that this approach performs badly in the specific layout used and this might be related to the majority of the nodes having less than 4 ways out (or adjacent tracks).

Although it has been shown that a solution based on BP ANN is possible it must now be discussed whether or not the solution is also acceptable to route the vehicles in a 'real' AGV system.

10.3 Applicability of the "hybrid solution" to a 'real' AGV system

One major issue in applying any of the approaches which performed best, in a 'real' AGV system is to concede that some paths, or transport orders will be delayed and may possibly enter a deadlock situation. The solutions based on ANN's with heuristics and "look-ahead" strategy are the ones which perform best in deadlock situations. However, no guarantee can be made that always 100% of the paths are accomplished within a specified maximum time. In a 'real' system that might not be acceptable because of the interaction of the transport system with the job scheduling system. It might be possible for this approach to be used in a simulation mode which would allow for a number of steps to be simulated and thus making it possible to identify paths which are drifting significantly from an acceptable expected time duration. Once these cases are identified then methods need to be derived to avoid or correct them; for example changing priorities. Alternatively specific rules could be developed and tested for these situations. There will always be the possibility of instability occurring and, the possibility to simulate the system would help to identify such situations. The graphs representing the distribution of the relative duration of each path ("k-ratio", Chapter 9) shows that a minor percentage of paths significantly exceed the minimum absolute paths. These graphs can be used when deriving and testing solutions that bring down the maximum "k-ratios" and spread the overall length more evenly among all paths.

A major concern is then whether it is possible to run the simulation in an on-line mode or whether the time needed for the simulation requires it to be made off-line, that is all tests and decisions are made prior to the AGV starting to move. This is related to the time available and the time required to do the simulations.

The time constraints of the proposed "hybrid solution" with complete "look-ahead" are mainly defined by the interface program which was implemented using the "C" programming language, because the ANN part runs almost instantaneously. This interface program implements simple rules which makes it run very quickly. A few measures of the time taken to produce solutions in a worst case condition, i.e. with 10 vehicles in the system, allowed more than 100 steps to be executed in a minute. Considering the response capability of a 'real' transport system, where the speed of the vehicles is normally limited for safety reasons to 1 m/s, and considering the lengths of the tracks, it is plausible that the proposed system can be used in an on-line mode. Furthermore it might even be considered to be worthwhile to explore the use of the data collected from the paths already performed to train the ANN. It seems therefore that the approach proposed has potential for application in a 'real' AGV system.

10.4 Further work

Further work will be required to verify the performance of the approach when the following cases are considered:

- physical layouts with a different organisation of its elements,

- larger systems.

The solutions developed and tested were based on a specific physical layout, i.e. it has a well defined number of nodes and pieces of track, and these were associated in a particular configuration. This determines the number of alternatives out of a node, and the possible paths linking every pair of nodes. Using layouts which are of similar size, but with a different configuration, could be used to establish

whether the use of ANN in the solution developed gives it the capability to perform just as well independently of the characteristics of the physical layout.

Further testing of the approach on larger physical layouts is required, i.e. layouts with a larger number of nodes and pieces of tracks. This will help to evaluate whether the physical layout considered, in the solution developed, can be used as the minimum local area around a vehicle that must be considered by an ANN representation to obtain good performance overall. It would also provide for the possibility of identifying and testing other sizes and configurations for this minimum local area.

Another issue is whether it is possible to increase the performance of the ANN solution when data representing cases with the respective overall optimum solution is available. The main problem will be to develop an algorithm which can provide this data, particularly when large numbers of vehicles and larger systems are involved. One possibility to make the development of this algorithm more feasible would be to place a limit on the number of times a vehicle could be waiting at a node, and investigate the possible solutions when different values for this limit are considered.

Chapter 11 | Conclusions

The development of traffic management and control systems capable of addressing the problems involved more efficiently can significantly extend the potential provided by the use of AGVs to implement the main transport system within a manufacturing environment. The main objective of this research was to derive more efficient solutions to the complex problems that arise from exploiting the increased levels of flexibility provided by transport systems based on AGVs. The work culminated with the development of a novel approach for the routing problem that was based on BP ANNs and the use of simple heuristic rules. The results obtained with this novel approach were better than the results obtained using the best published heuristic rules [Ref. 40] in the cases tested, and its iterative and localised characteristics give it potential to be used in different and larger cases.

The routing flexibility provided by the use of AGVs is one of the aspects which assumes particular relevance in the ability of the manufacturing system to respond adequately to changes determined by the dynamic behaviour of its manufacturing operations, or to the changes determined at the systems planning level. This concept can be extended to the consideration of transport of increasingly smaller quantities more frequently, and consequently requiring a larger number of vehicles. This increases the complexity of the problems faced by the traffic management and control system, particularly the routing problem.

The complexity of this problem arises from the huge increase of alternatives that must be evaluated as the problem increases in size, and as such is also representative of many other problems in related or different areas, including the shortest path and single-loop guide path layout problems [Ref. 25], sequencing, scheduling and network type problems [i.e. Ref. 75]. Very often these problems can be

formulated as optimisation problems, and although various approaches have been used in an attempt to solve these problems, normally they have limitations either related to the size of the problem, or the number of variables which must be considered to model the problem. The use of methods which at best can only guarantee a close to optimum solution is frequently the only alternative [i.e. Ref. 25] and was also the approach chosen in this research. A solution for the specific case of routing vehicles was derived and as such it cannot be considered as directly applicable to other problems of the same complexity. It can nevertheless be considered as another example of success of heuristic solutions and of the capabilities of ANNs to implement complex nonlinear mappings.

The work focuses on the use of Backpropagation type ANNs as an alternative to the conventional methods of solution. The motivation for the use of this type of ANNs was their ability to recognise, and distinguish between, patterns. If a system had the ability to identify specific conditions in the traffic system then it could provide better grounds on which decisions could be made in relation to, for example, the routing problem.

The approach derived is based on the development of a BP ANN solution to the simpler case of routing a single vehicle from its current location to another location in a physical layout, when some of the tracks are set as unavailable. Subsequently the solution for this single vehicle case is used as the basis for the solution of the multiple vehicle case. Each time a vehicle has to make a movement decision its current and destination locations are input information to the BP ANN designed for the single vehicle case. Based on this information and on the tracks current state of occupancy due to the intentions of movement of the remaining vehicles, the BP ANN produces movement decisions for the current vehicle. Using this scheme iteratively for each vehicle, movement decisions for all vehicles can be obtained which take into account each one's intended movements.

The overall approach used illustrated the possibility of obtaining good solutions to the complex problem of routing multiple vehicles using data from a simpler one vehicle problem. Finding optimal solutions for a multi-vehicle route selection problem is in general a very difficult task even if we are not imposed time constraints to produce a solution. The solution proposed cannot be guaranteed to be optimal, but it was shown it performs comparatively well. Another advantage lies in its inherent

localised application that favours a more rapid response to dynamic changes, and allows the possibility for extension to larger systems.

Defining the layout area associated with the input information to the BP ANN as a limited area of influence that must be considered, and repeats itself in a larger layout, the extension of this approach to larger systems is immediate by considering this area has moving with the vehicles using the BP ANN. The capabilities of BP ANNs to learn and recognise different patterns provide an added advantage to refine this solution. For example by redefining entirely the limited area that should be considered with a corresponding redesign of the BP ANN, or even by using BP ANNs which were specifically designed to account for the differences in the type of nodes (i.e. two, three, four ways out of the node). Alternatively it can also be envisaged its application in cases where the vehicles are restricted to move in limited areas of a larger layout with specific locations for exchanging loads crossing to other areas.

Although a hybrid solution was required it was illustrated that BP ANN played an important role in arriving at a solution which, in spite of being localised, it can also include more global information about the traffic conditions.

The major conclusions drawn from the work done can be summarised as follows:

- the use of BP ANN as part of a solution to the routing problem of AGVs is possible and it plays an important role in an hybrid solution, with the results obtained showing that it has advantages over other existing solutions;

- it is difficult to find a representation of the problem suitable for a BP ANN implementation and capable of satisfying all constraints of the routing problem, but this can nevertheless be overcome by the use of simple rules;

- the approach developed can be extended to larger and more complex layouts, although further work is required in order to evaluate its performance in those cases.

Bibliography

Ref. 1: Groover, M.P.; "Automation, Production Systems, and Computer Integrated Manufacturing", Prentice-Hall International Editions, 1987.

Ref. 2: Meystel, A.; "Autonomous Mobile Robots, Vehicles with Cognitive Control", World Scientific Publishing Co. Pte. Ltd., 1991.

Ref. 3: Gupta, Y.P., Gupta, M.C., Bector, C.R.; "A review of scheduling rules in flexible manufacturing systems", Int. J. Computer Integrated Manufacturing, Vol. 2, N°. 6, pp. 356-377, 1989.

Ref. 4: Liu, C.-M., Duh, S.-H.; "Study of AGVS design and dispatching rules by analytical and simulation methods", Int. J. Computer Integrated Manufacturing, Vol. 5, N°. 4&5, pp. 290-299, 1992.

Ref. 5: Sabuncuoglu, I., Hommertzheim, D.L.; "Esperimental investigation of FMS machine and AGV scheduling rules against the mean flow-time criterion", Int. J. Production Research, Vol.30, N°.7, pp. 1617-1635, 1992.

Ref. 6: Sabuncuoglu, I., Hommertzheim, D.L.; "Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system", Int. J. Production Research, Vol.30, N°.5, pp. 1059-1079, 1992.

Ref. 7: Occeña, L.G., Yokota, T.; "Modelling of an automated guided vehicle system (AGVS) in a just-in-time (JIT) environment", Int. J. Production Research, Vol.29, N°.3, pp. 495-511, 1991.

Ref. 8: Liu, P.-S., Fu, L.-C.; "Planning and Scheduling in a Flexible Manufacturing System Using a Dynamic Routing Method for Automated Guided Vehicles", IEEE, pp. 1584-1589, 1989.

Ref. 9: Egbelu, P.J.; "Route selection and flow control in a multi-stage manufacturing system with heterogeneous machines within stages", Int. J. Production Research, Vol.28, N°.11, pp. 2137-2155, 1990.

Ref. 10: Egbelu, P.J.; "Machining and material flow system design for minimum cost production", Int. J. Production Research, Vol.28, N°.2, pp. 353-368, 1990.

Ref. 11: Ro, I.-K., Kim, J.-I.; "Multi-criteria operational control rules in flexible manufacturing systems (FMSs)", Int. J. Production Research, Vol.28, N°.1, pp. 47-63, 1990.

Ref. 12: Freeman, J.A., Skapura, D.M.; "Neural Networks: Algorithms, Applications, and Programming Techniques.", Addison-Wesley Publishing Company, Inc., 1992.

Ref. 13: Hertz, J.A., Palmer, R.G., Krogh, A.S.; "Introduction to the theory of neural computation", Addison-Wesley Publishing Company, 1991.

Ref. 14: Rumelhart, D.E., McClelland, and the PDP Research Group; "Parallel Distributed Processing. Explorations in the Microstructure of Cognition, vol.1: Foundations.", The MIT Press, 1986.

Ref. 15: Ranky, P.G.; "Flexible Manufacturing Cells and Systems in CIM", CIMware Ltd, 1990.

Ref. 16: Greenwood, N.R.; "Implementing Flexible Manufacturing Systems", MacMillan Education Ltd., 1988.

Ref. 17: Rembold, U., Nnaji, B.O., Storr, A.; "Computer Integrated Manufacturing and Engineering", Addison-Wesley Publishing Company, 1994.

Ref. 18: Borenstein, J., Koren, Y.; "Real-Time Obstacle Avoidance for Fast Mobile Robots", IEEE Transactions on Systems, Man, and Cybernetics, Vol.19, N°.5, pp. 1179-1187, September/October, 1989.

Ref. 19: Borenstein, J., Koren, Y.; "The Vector Field Histogram--Fast Obstacle Avoidance for Mobile Robots", IEEE Transactions on Robotics and Automation, Vol.7, N°.3, pp. 278-288, June, 1991.

Ref. 20: Kweon, I., Kuno, Y., Watanabe, M., Onoguchi, K.; "Behavior-Based Mobile Robot using Active Sensor Fusion", Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, pp. 1675-1682, France - May 1992.

Ref. 21: Skewis, T., Lumelsky, V.; "Experiments with a Mobile Robot Operating in a Cluttered Unknown Environment", Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, pp. 1482-1487, France - May 1992.

Ref. 22: Barbehenn, M., Hutchinson, S.; "Learning Conditional Effects of Actions for Robot Navigation", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, pp. 260-265, California - April 1991.

Ref. 23: Chen, P.C., Hwang, Y.K.; "Practical Path Planning among Movable Obstacles", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, pp. 444-449, California - April 1991.

Ref. 24: Chan, W.K., Lew, S.C.; "Local Path Optimisation of Free-ranging Automated Guided Vehicles", Computers in Industry 14, pp. 361-365, 1990.

Ref. 25: Guzman, M.C. DE, Prabhu, N., Tanchoco, J.M.A.; "Complexity of the AGV shortest path and single-loop guide path layout problems", Int. J. Production Research, Vol.35, No.8, pp. 2083-2092, 1997.

Ref. 26: Srinivasan, M.M., Bozer Y.A.; "Which one is responsible for WIP: the workstations or the material handling system?", Int. J. Production Research, Vol.30, N°.6, pp. 1369-1399, 1992.

Ref. 27: Mahadevan, B., Narendran, T.T., "Design of an automated guided vehicle-based material handling system for a flexible manufacturing system", Int. J. Production Research, Vol.28, N°.9, pp. 1611-1622, 1990.

Ref. 28: Mahadevan, B., Narendran, T.T.; "A hybrid modelling approach to the design of an AGV-based material handling system for an FMS", Int. J. Production Research, Vol.32, N°.9, pp. 2015-2030, 1994.

Ref. 29: Rembold, B., Tanchoco, J.M.A.; "A modular framework for the design of material flow systems", Int. J. Production Research, Vol.32, N°.1, pp. 1-21, 1994.

Ref. 30: Vosniakos, G.-C., Davies, B.J.; "On the path layout and operation of an AGV system serving an FMS", Int. J. Advanced Manufacturing Technology, N°4, pp. 243-262, 1989.

Ref. 31: Kaspi, M., Tanchoco, J.M.A.; "Optimal flow path design of unidirectional AGV systems", Int. J. Production Research, Vol.28, N°.6, pp. 1023-1030, 1990.

Ref. 32: Bartholdi, J.J., Platzman, L.K.; "Decentralized Control of Automated Guided Vehicles on a Simple Loop", IIE Transactions, Vol.21, N°.1, pp. 76-80, March 1989.

Ref. 33: Tanchoco, J.M.A., Sinriech, D.; "OSL-optimal single-loop guide paths for AGVS", Int. J. Production Research, Vol.30, N°.3, pp. 665-681, 1992.

Ref. 34: Malmborg, C.J.; "Heuristic, storage space minimization methods for facility layouts served by looped AGV systems", Int. J. Production Research, Vol.32, N°.11, pp. 2695-2710, 1994.

Ref. 35: Malmborg, C.J.; "A model for the design of zone control automated guided vehicle systems", Int. J. Production Research, Vol.28, N°.10, pp. 1741-1758, 1990.

Ref. 36: Faraji, M., Batta, R.; "Forming cells to eliminate vehicle interference and system locking in an AGVS", Int. J. Production Research, Vol.32, N°.9, pp. 2219-2241, 1994.

Ref. 37: Huang, C.; "Design of material transportation system for tandem automated guided vehicle systems", Int. J. Production Research, Vol.35, N°4, pp. 943-953, 1997.

Ref. 38: Lin, J.T., Chang, C.C.K., Liu, W.-C.; "A load-routeing problem in a tandem-configuration automated guided-vehicle system", Int. J. Production Research, Vol.32, N°.2, pp. 411-427, 1994.

Ref. 39: Lin, J.T., Dgen, P.-K.; "An algorithm for routeing control of a tandem automated guided vehicle system", Int. J. Production Research, Vol.32, N°.12, pp. 2735-2750, 1994.

Ref. 40: Grossman, D.D.; "Traffic Control of Multiple Robot Vehicles.", IEEE Journal of Robotics and Automation, vol. 4, N°5, pp. 491-497, October 1988.

Ref. 41: Lippmann, R.P.; "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-22, April 1987.

Ref. 42: Khanna, T.; "Foundations of Neural Works", Addison-Wesley Publishing Company, 1990.

Ref. 43: Simpson, P.K.; "Artificial Neural Systems, Foundations, Paradigms, Applications and Implementations.", Elsevier Science Ltd, 1990.

Ref. 44: Aleksander, I., Morton, H.; "An Introduction to Neural Computing", Chapman And Hall, 1990.

Ref. 45: Wasserman, P.D.; "Advanced Methods in Neural Computing", Van Nostrand Reinhold, 1993.

Ref. 46: Hopfield, J.J., Tank, D.W.; " "Neural" Computation of Decisions in Optimization Problems", Biological Cybernetics 52, pp.141-152, 1985.

Ref. 47: Tank, D.W., Hopfield, J.J.; " Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Transactions on Circuits and Systems Vol. 33, N°5, pp. 533-541, May 1986.

Ref. 48: Durbin, R., Willshaw, D.; "An analogue approach to the travelling salesman problem using an elastic net method", Nature, Vol. 326, pp. 689-691, 16 April 1987.

Ref. 49: Wong, W.S., Funka-Lea, C.A.; "An Elastic Net Solution to Obstacle Avoidance Tour Planning", Int. Joint Conf. on Neural Networks, Vol. 3, pp. III-799-804, 17-21 June 1990, San Diego, CA, USA, 1990.

Ref. 50: Gielen, C., Glasius, R.; "Coordination of movements by self-organizing neural networks", Proc. Int. Conf. Intelligent Autonomous Systems 2, Vol. 1, pp. 275-283, 11-14 December 1989, Amsterdam, Netherlands.

Ref. 51: Glasius, R., Komoda, A., Gielen, S.C.A.M.; "Neural Network Dynamics for Path Planning and Obstacle Avoidance", Neural Networks, Vol.8, N°. 1, pp. 125-133, Elsevier Science Ldd, 1995.

Ref. 52: Hemani, A., Postula, A.; "Cell Placement by Self-Organisation", Neural Networks, Vol.3, pp. 377-383, 1990.

Ref. 53: Hao, G., Lai, K.K.; "Solving the AGV Problem via a Self-Organizing Neural Network", Journal of the Operational Research Society, 47, pp. 1477-1493, 1996.

Ref. 54: Chryssolouris, G., Lee, M., Domroese, M.; "The Use of Neural Networks in Determining Operational Policies for Manufacturing Systems", Journal of Manufacturing Systems, Vol.10, N°. 2, pp. 166-175, 1991.

Ref. 55: Potvin, J.-Y., Shen, Y., Rousseau, J.-M.; "Neural Networks For Automated Vehicle Dispatching", Computers Ops. Research, Vol. 19, pp. 267-276, 1992.

Ref. 56: Tuijnman, F., Krose, B.J.A.; "Neural Networks for Collision Avoidance between Autonomous Mobile Robots", Proc. Int. Conf. Intelligent Autonomous Systems 2, vol.1, pp. 407-416, 11-14 Dec. 1989, Amsterdam, Netherlands, 1989.

Ref. 57: Sim, S.K., Yeo, K.T., Lee, W.H.; "An expert neural network system for dynamic job shop scheduling", Int. J. Production Research, Vol.32, N°.8, pp. 1759-1773, 1994.

Ref. 58: Meng, M., Kak, A.C.; "Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models", IEEE Control Systems, pp. 30-39, October 1993.

Ref. 59: Nagata, S., Sekiguchi, M., Asakawa, K.; "Mobile Robot Control by a Structured Hierarchical Neural Network", IEEE Control Systems Magazine, pp. 69-76, April 1990.

Ref. 60: Demuth, H., Beale, M.;"Neural Network Toolbox User's Guide", The MathWorks Inc., 1994.

Ref. 61: Aiyer, S.V.B., Niranjan, M., Fallside, F.; "A Theoretical Investigation into the Performance of the Hopfield Model", IEEE Transactions On Neural Networks, Vol. 1, N°. 2, pp. 204-215, June 1990.

Ref. 62: Bersini, H., Saerens, M., Sotelino, L.G.; "Hopfield Net Generation, Encoding and Classification of Temporal Trajectories", IEEE Transactions on Neural Networks, Vol. 5, N°. 6, pp. 945-953, November 1994.

Ref. 63: Lee, B.W., Sheu, B.J.; "Modified Hopfield Neural Networks for Retrieving the Optimal Solution", IEEE Transactions on Neural Networks, Vol. 2, N°.1, pp.137-142, January 1991.

Ref. 64: Samad, T., Harper, P.; "High-order Hopfield and Tank optimization networks", Parallel Computing 16, pp.287-292, 1990.

Ref. 65: Sivak, J.A.; "A Neural Network for Resource Allocation", Conf. Proc. Autofact 89, Oct.30-Nov.2, pp.29/11-22, Detroit, Michigan, 1989.

Ref. 66: Pourboghrat, F.; "Neural Networks Application in Autonomous Path Generation for Mobile Robots", Proceedings of SPIE - The Int. Soc. for Optical Engineering, Vol. 1396, pp. 243-252, Midwest'90, 1990.

Ref. 67: Bostel, A.J., Gan, W.W., Sagar, V.K., See, C.H.; "Generation of Optimal Routes in a Neural Network Based AGV Controller", Intelligent Systems Engineering, 5-9 September, Conference Publication N° 395, pp. 165-176, IEE, 1994.

Ref. 68: Bostel, A.J., Sagar, V.K.; "Dynamic control systems for AGVs", Computing & Control Engineering Journal, pp. 168-177, August 1996.

Ref. 69: Abe, S., Kawakami, J., Hirasawa, K.; "Solving Inequality Constrained Combinatorial Optimization Problems by the Hopfield Neural Networks", Neural Networks, Vol. 5, pp. 663-670, Pergamon Press Ltd, 1992.

Ref. 70: Lo, Z.-P., Bavarian, B.; "Multiple Job Scheduling With Artificial Neural Networks", Computers Elect. Engineering, Vol. 19, N°2, pp. 87-101, Pergamon Press Ltd, 1993.

Ref. 71: Sabuncuoglu, I., Gurgun, B.; "A neural network model for scheduling problems", European Journal of Operation Research, Vol. 93, N°. 2, pp. 288-299, September 6, 1996.

Ref. 72: Smith, K., Palaniswami, M., Krishnamoorthy, M.; "Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem", European Journal of Operation Research, Vol. 93, N°. 2, pp. 300-316, September 6, 1996.

Ref. 73: Fox, G., Gurewitz, E., Wong, Y.; "A Neural Network Approach to Multi-vehicle Navigation", Intelligent Control and Adaptive Systems, SPIE, Vol. 1196, pp. 164-169, 1989.

Ref. 74: Wilson, G.V., Pawley, G.S.; "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank". Biological Cybernetics, 58, pp.63-70, 1988.

Ref. 75: Garey, M.R., Johnson, D.S.; "Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, New York, 1979.

Ref. 76: Goldberg, D.E.; "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, Inc., 1989.

Ref. 77: Shibata, T., Fukuda, T., Kosuge, K., Arai, F.; "Path-planning For Multiple Mobile Robots By Using Genetic Algorithms", Robotics, Mechatronics and Manufacturing Systems, T. Takamori and K. Tsuchiya (Editors), Elsevier Science Publishers B.V. (North-Holland), pp. 409-415, 1993.

Ref. 78: Yuan, Y., Wu, Z.; "Algorithm of Fuzzy Dynamic Programming in AGV Scheduling", Proc. Int. Conf. on Computer Integrated Manufacturing, ICCIM'91, pp. 405-408, 2-4 October 1991, Singapore.

Ref. 79: Guiver, J.P., Klimasauskas, C.;"NeuralWorks: Networks II", NeuralWare, Inc.,1990.

Ref. 80: NeuralWare, Inc.;"Using NeuralWorks; an extended tutorial for NeuralWorks Prof.II/Plus and NeuralWorks Explorer", 1992.

Ref. 81: NeuralWare, Inc.; "Neural Computing, A Technology Handbook for Prof. II/Plus and NeuralWorks Explorer", 1994.

Ref. 82: Sedgewick, Robert; "Algorithms", 2nd Edition, Addison-Wesley Publishing Company, 1988.

Ref. 83: Moret, B.M.E., Shapiro, H.D.; " Algorithms from P to NP, vol.1, Design & Efficiency", The Benjamin/Cummings Publishing Company, Inc., 1991.

Ref. 84: Weiss, S.M., Kulikowski, C.A.; "Computer Systems That Learn: Classification and Prediction from Statistics, Neural Nets, Machine Learning, and Expert Systems", Morgan Kaufmann Publishers, 1991.

Ref. 85: Sontag, Eduardo D.; "Capabilities and Training of Feedforward Nets", Academic Press, Inc., pp. 303-321, 1991.

Appendix A | Layouts Used in Training and Testing Data Sets

This appendix contains examples of cases generated by the program for generation of data and also the drawings of the layouts corresponding to the cases used in the training data set and the two testing data sets (Testing1 and Testing2).

1.1 Example of data obtained from program for data generation

The examples shown next represent the cases found using the program for generation of data when tracks are set as available or not available (i.e. state "0" or "1") and two nodes are specified as the nodes to be linked. The format of the data corresponds to an identification of the tracks and nodes of the physical layout as represented in Figure A- 1 and a codification of the input and output elements as defined in Representation 1 (Chapter 5). This codification consists of each case having two lines with binary elements ("0" or "1"). The first line represent has 40 elements, of which the first 24 represent each the tracks defined in the physical layout (a value of "0" or "1" means respectively track available or not available), and the last 16 elements correspond each to the possible origin/destination nodes (a value of "1" means node to be linked). The 24 binary elements of the second line represent the 24 tracks of the physical layout to indicate which are used in the solution path ("0" not used, "1" used). The third line of each case represents the path length (decimal notation) assuming each track is 10 units long.

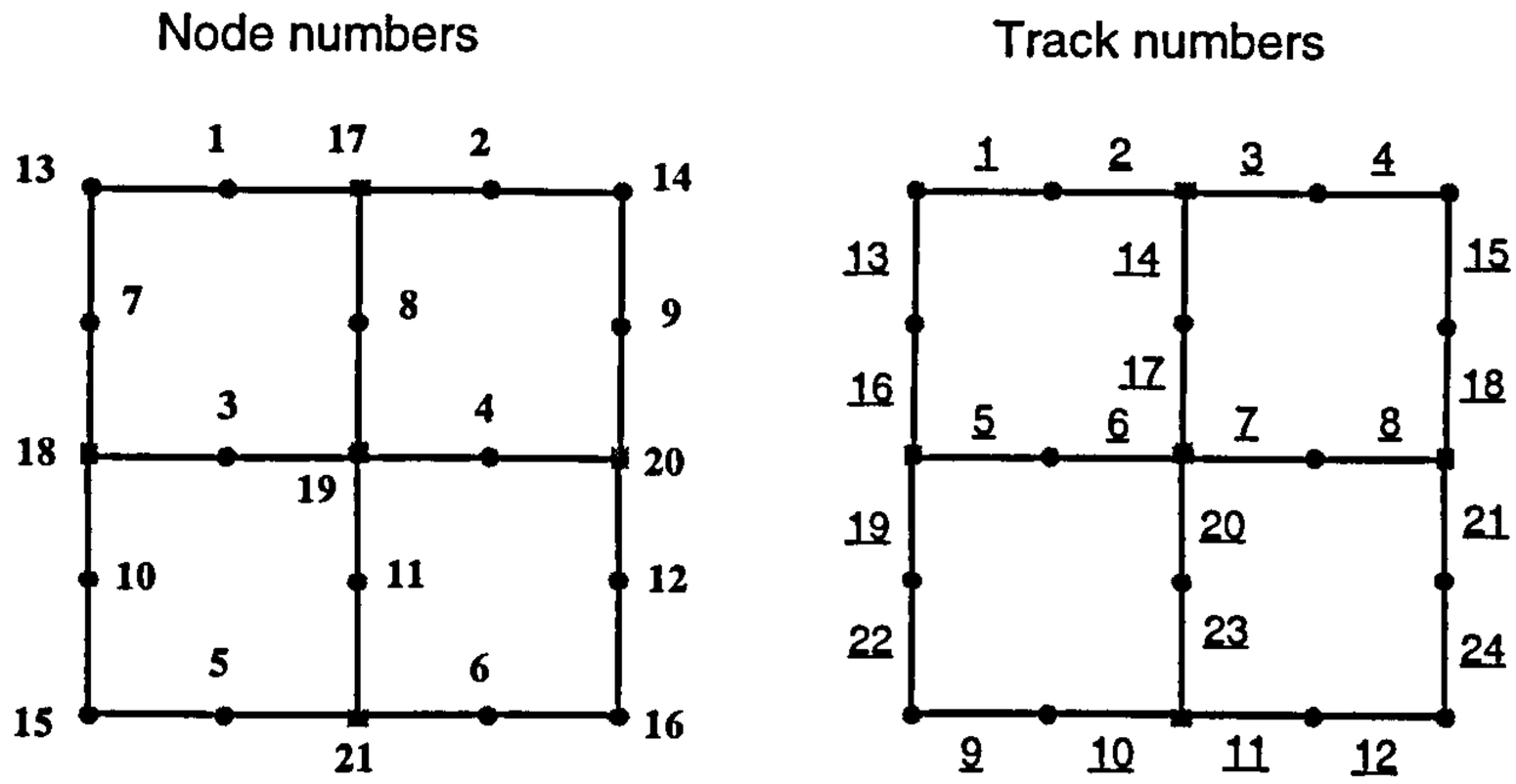


Figure A- 1: Numbering of nodes and tracks in the physical layout.

Examples with four tracks blocked in each case:

```
!Inp.(1)( 1,1):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
&0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0
&30
!---------------
!Inp.(1)( 1,2):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
&0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
&40
!---------------
!Inp.(2)( 1,2):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
&1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0
&60
!---------------
!Inp.(1)( 1,4):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
&0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
&40
!---------------
!Inp.(1)( 1,5):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
&0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0
&50
!---------------
!Inp.(2)( 1,5):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0
&1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0
&110
!---------------
!Inp.(1)( 1,6):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
&0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
&40
!---------------
!Inp.(1)( 1,8):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0
&50
!---------------
!Inp.(2)( 1,8):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 0 0 1 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0
&110
!---------------
!Inp.(1)( 1,9):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
&0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0
&70
!---------------
!Inp.(1)( 1,10):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
&20
```

Examples with three tracks blocked in each case:

```
!Inp.(1)( 1,1):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
&0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0
&30
!---------------
!Inp.(1)( 1,2):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
&0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
&40
!---------------
!Inp.(2)( 1,2):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
&1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0
&60
!---------------
```

```
!Inp.(1)( 1,4):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
&0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
&40
!---------------
!Inp.(1)( 1,5):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
&0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0
&50
!---------------
!Inp.(2)( 1,5):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
&1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0
&110
!---------------
!Inp.(1)( 1,6):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
&0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
&40
!---------------
!Inp.(1)( 1,8):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0
&50
!---------------
!Inp.(2)( 1,8):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0
&110
!---------------
!Inp.(1)( 1,9):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
&0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 0
&70
!---------------
!Inp.(1)( 1,10):
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
&20
!---------------
!Inp.(1)( 2,1):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
&0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
&50
!---------------
!Inp.(1)( 2,2):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
&0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0
&60
!---------------
!Inp.(2)( 2,2):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
&1 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0
&120
!---------------
!Inp.(1)( 2,3):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
&30
!---------------
!Inp.(1)( 2,4):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
&0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
&40
!---------------
!Inp.(1)( 2,5):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
&1 0 0 0 1 1 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0
&100
!---------------
!Inp.(2)( 2,5):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
&0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0
&80
!---------------
!Inp.(1)( 2,6):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
&0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1
&40
!---------------
!Inp.(1)( 2,7):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0
&0 1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
&60
!---------------
!Inp.(2)( 2,7):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&40
!---------------
!Inp.(1)( 2,8):
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
&0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&10
```

Examples with two tracks blocked in each case:

```
!Inp.(1)( 1,1):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
&30
!---------------
!Inp.(2)( 1,1):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
&50
!---------------
!Inp.(3)( 1,1):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
```

```
&1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
&90
!---------------
!Inp.(4)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0
&130
!---------------
!Inp.(1)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
&10
!---------------
!Inp.(2)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0
&70
!---------------
!Inp.(3)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0
&110
!---------------
!Inp.(4)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0
&150
!---------------
!Inp.(1)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
&50
!---------------
!Inp.(2)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0
&130
!---------------
!Inp.(3)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
&50
!---------------
!Inp.(4)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 0
&110
!---------------
!Inp.(5)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0
&110
!---------------
!Inp.(6)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
&70
!---------------
!Inp.(7)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
&110
!---------------
!Inp.(1)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0
&100
!---------------
!Inp.(2)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0
&80
!---------------
!Inp.(3)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0
&60
!---------------
!Inp.(4)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0
&120
!---------------
!Inp.(5)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0
&80
!---------------
!Inp.(6)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0
&60
!---------------
!Inp.(1)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0
&130
!---------------
!Inp.(2)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 0 0 0 1 1 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0
&90
!---------------
!Inp.(3)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0
&70
!---------------
!Inp.(4)( 1,5):
```

```
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 0 1 1 0 0 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0
&130
!---------------
!Inp.(5)( 1,5):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 0 1 1 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0
&90
!---------------
!Inp.(6)( 1,5):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
&30
!---------------
!Inp.(1)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0
&50
!---------------
!Inp.(2)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0
&130
!---------------
!Inp.(3)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0
&90
!---------------
!Inp.(4)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
&30
!---------------
!Inp.(5)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0
&110
!---------------
!Inp.(6)( 1,8):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
&1 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 1 0
&150
!---------------
!Inp.(1)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
&40
!---------------
!Inp.(2)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0
&60
!---------------
!Inp.(3)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0
&80
!---------------
!Inp.(4)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
&100
!---------------
!Inp.(5)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 1 1 1 0 1 1 0
&120
!---------------
!Inp.(6)( 1,9):
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0
&140
```

Examples with one track blocked in each case:

```
!Inp.(1)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1
&140
!---------------
!Inp.(2)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 1
&120
!---------------
!Inp.(3)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 0 1 0 0 1 1
&100
!---------------
!Inp.(4)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1
&120
!---------------
!Inp.(5)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0
&140
!---------------
!Inp.(6)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0
&100
!---------------
!Inp.(7)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0
```

```
&40
!---------------
!Inp.(1)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 0 0 1
&110
!---------------
!Inp.(2)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0
&70
!---------------
!Inp.(3)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0
&30
!---------------
!Inp.(4)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 1
&130
!---------------
!Inp.(5)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1
&170
!---------------
!Inp.(6)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0
&130
!---------------
!Inp.(7)( 1,2):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
&1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0
&130
!---------------
!Inp.(1)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1
&30
!---------------
!Inp.(2)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0
&70
!---------------
!Inp.(3)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 1 0 0
&110
!---------------
!Inp.(4)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1
&130
!---------------
!Inp.(5)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1 0 0 1 0
&130
!---------------
!Inp.(6)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 0 0 1 0 0
&130
!---------------
!Inp.(7)( 1,3):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
&1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 1
&170
!---------------
!Inp.(1)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0
&70
!---------------
!Inp.(2)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0
&170
!---------------
!Inp.(3)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1 0
&150
!---------------
!Inp.(4)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0
&50
!---------------
!Inp.(5)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1
&110
!---------------
!Inp.(6)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
&50
!---------------
!Inp.(7)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 1
&110
!---------------
!Inp.(8)( 1,4):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
```

```
&1 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1
&170
!---------------
!Inp.(1)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0
&70
!---------------
!Inp.(2)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0
&170
!---------------
!Inp.(3)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1 0
&150
!---------------
!Inp.(4)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0
&50
!---------------
!Inp.(5)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1
&110
!---------------
!Inp.(6)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1
&50
!---------------
!Inp.(7)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 1
&110
!---------------
!Inp.(8)( 1,5):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
&1 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1
&170
!---------------
!Inp.(1)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1
&130
!---------------
!Inp.(2)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1
&90
!---------------
!Inp.(3)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
&90
!---------------
!Inp.(4)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0
&150
!---------------
!Inp.(5)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0
&70
!---------------
!Inp.(6)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&1 1 1 1 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0
&110
!---------------
!Inp.(7)( 1,6):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
&0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0
&70
!---------------
!Inp.(1)( 1,7):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&20
!---------------
!Inp.(2)( 1,7):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1
&100
!---------------
!Inp.(3)( 1,7):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0
&60
!---------------
!Inp.(4)( 1,7):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1
&140
!---------------
!Inp.(5)( 1,7):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0
&140
!---------------
!Inp.(1)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 0 1
&120
!---------------
!Inp.(2)( 1,8):
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1
&120
!---------------
!Inp.(3)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
&60
!---------------
!Inp.(4)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&0 1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 1
&120
!---------------
!Inp.(5)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0
&100
!---------------
!Inp.(6)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1
&160
!---------------
!Inp.(7)( 1,8):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
&1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
&60
!---------------
!Inp.(1)( 1,9):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&20
!---------------
!Inp.(2)( 1,9):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1
&100
!---------------
!Inp.(3)( 1,9):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0
&60
!---------------
!Inp.(4)( 1,9):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1
&140
!---------------
!Inp.(5)( 1,9):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0
&140
!---------------
!Inp.(1)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
&0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1
&100
!---------------
!Inp.(2)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1
&160
!---------------
!Inp.(3)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1
&60
!---------------
!Inp.(4)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1
&160
!---------------
!Inp.(5)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
&20
!---------------
!Inp.(6)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 0
&120
!---------------
!Inp.(7)( 1,10):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
&1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 0
&160
!---------------
!Inp.(1)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1
&160
!---------------
!Inp.(2)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 1 0
&80
!---------------
!Inp.(3)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
&20
!---------------
!Inp.(4)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
&0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0 1
&140
!---------------
```

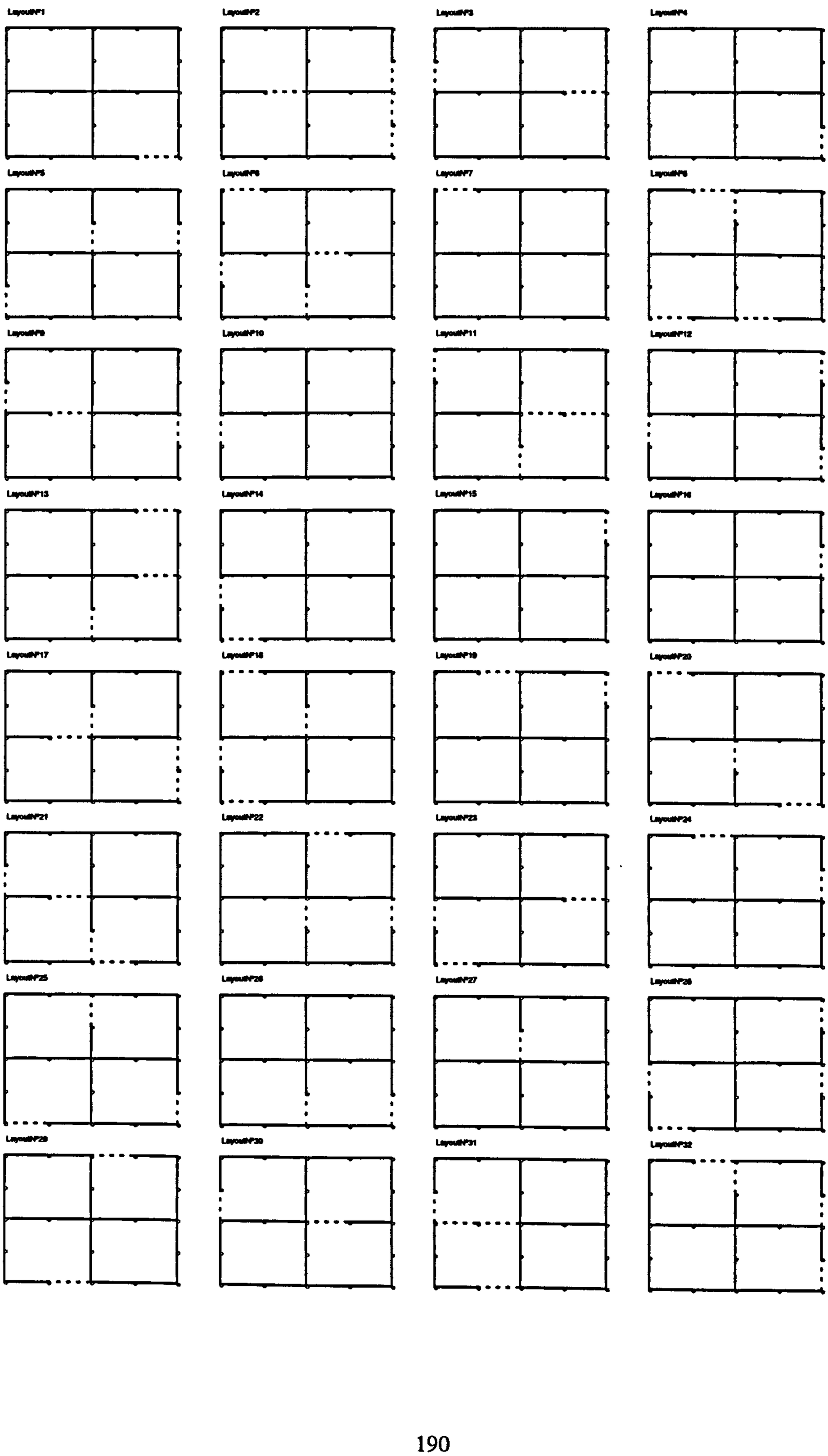
```
!Inp.(5)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
&0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1
&120
!---------------
!Inp.(6)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
&1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
&60
!---------------
!Inp.(7)( 2,1):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
&1 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 0
&120
!---------------
!Inp.(1)( 2,2):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
&0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
&20
!---------------
!Inp.(2)( 2,2):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
&0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 1 1
&100
!---------------
!Inp.(3)( 2,2):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
&0 0 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1
&140
!---------------
!Inp.(4)( 2,2):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1
&140
!---------------
!Inp.(5)( 2,2):
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
&1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 1
&140
```

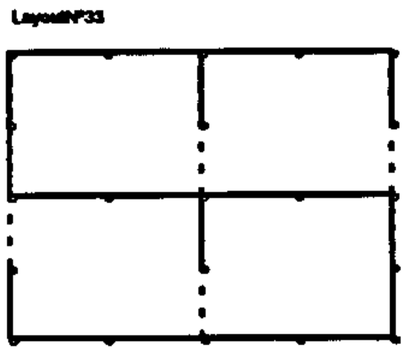
Example with all tracks available and nodes to be linked (3 and 14)

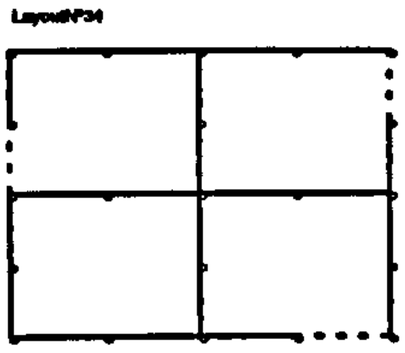
```
!Inp.(1)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 0 1
&170
!---------------
!Inp.(2)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
&50
!---------------
!Inp.(3)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 1
&90
!---------------
!Inp.(4)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0
&130
!---------------
!Inp.(5)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
&50
!---------------
!Inp.(6)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1
&170
!---------------
!Inp.(7)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 1
&150
!---------------
!Inp.(8)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1
&110
!---------------
!Inp.(9)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 0
&110
!---------------
!Inp.(10)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0
&110
!---------------
!Inp.(11)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
&70
!---------------
!Inp.(12)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
&110
!---------------
!Inp.(13)( 1,1):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
&1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1
&150
!---------------
```

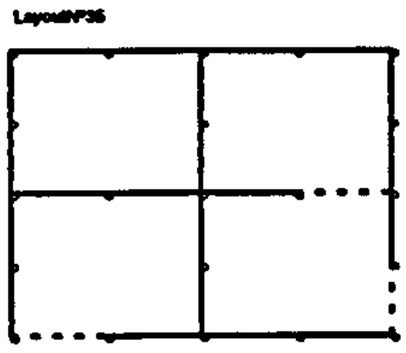
1.2 Drawings of layouts used

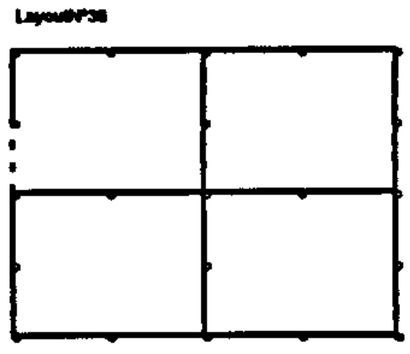
Training and testing_1 data sets:

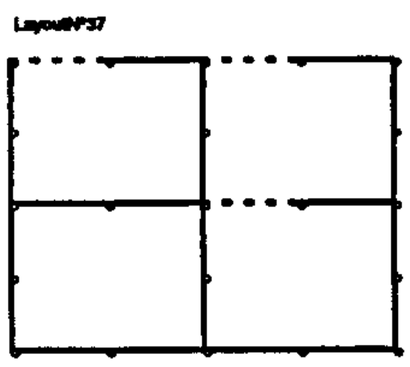


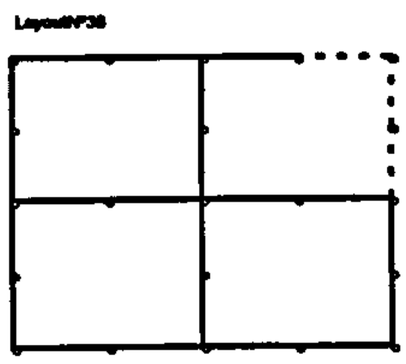


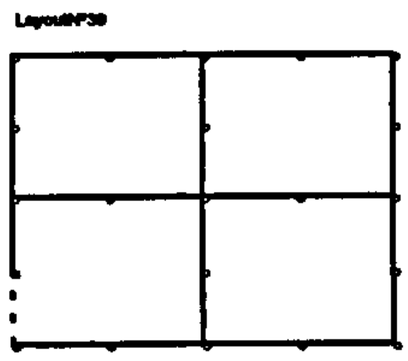


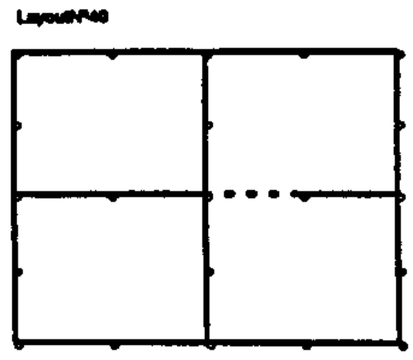


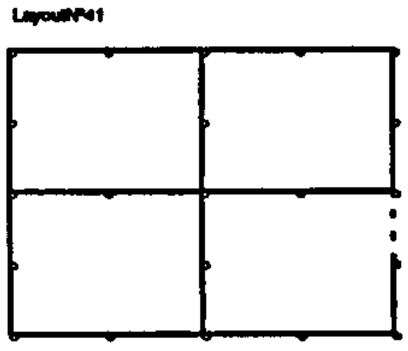


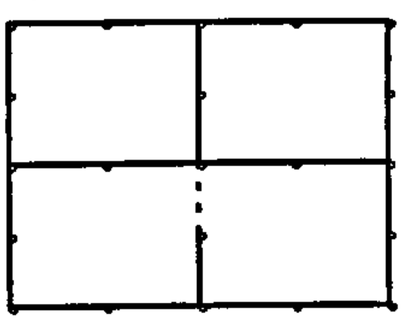


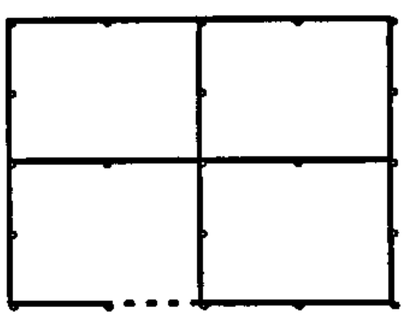


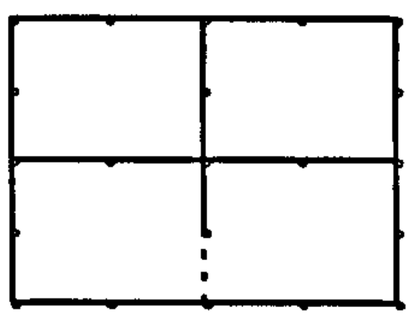


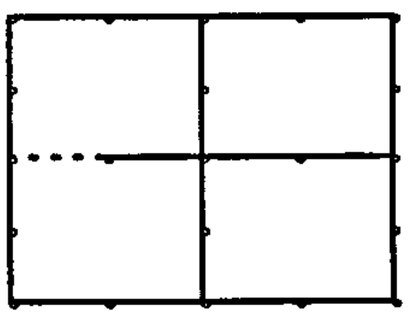


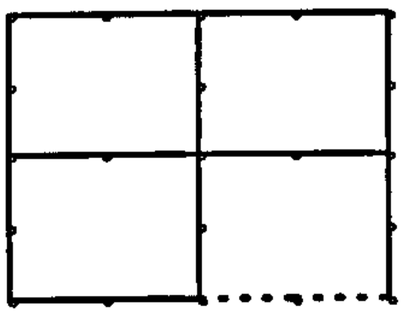


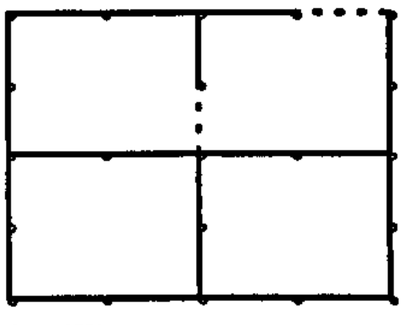


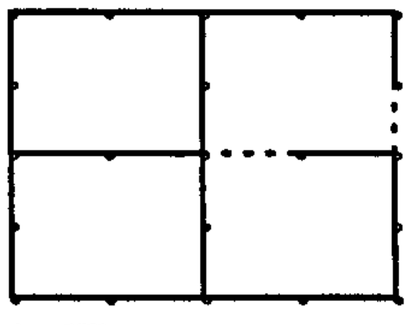


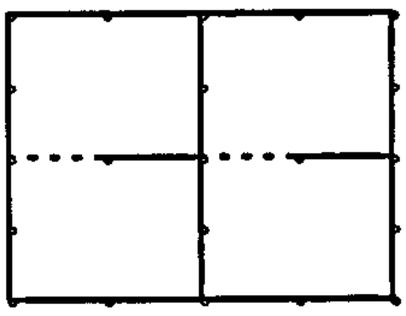


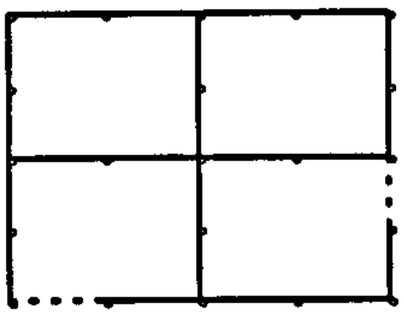


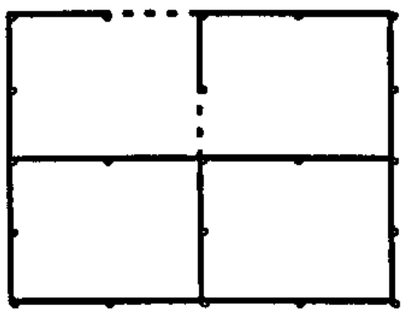


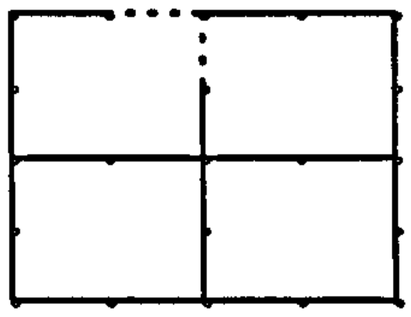


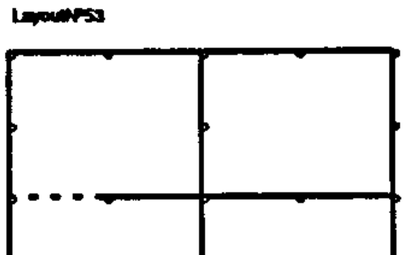


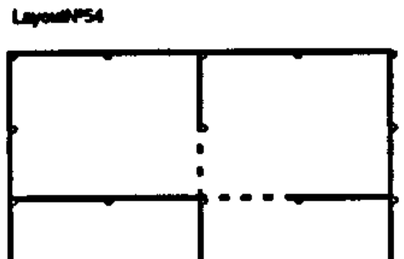


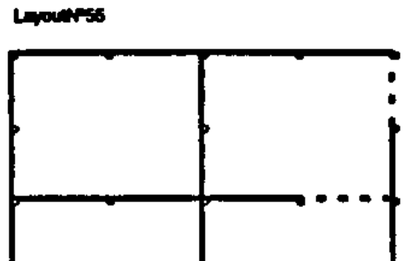


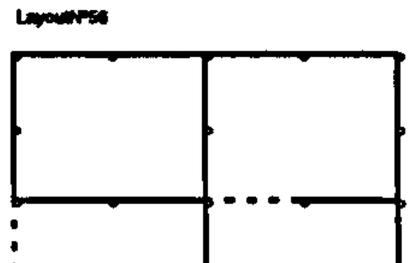


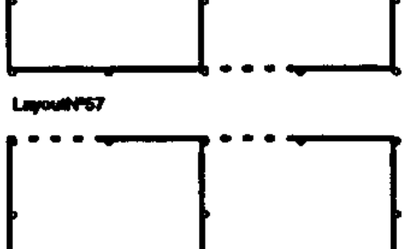


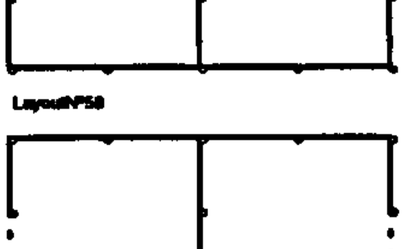


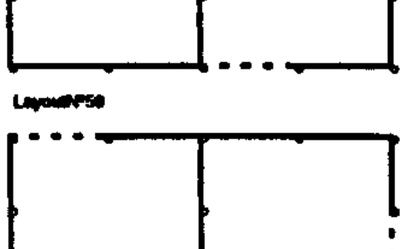


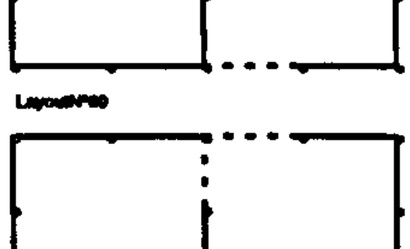


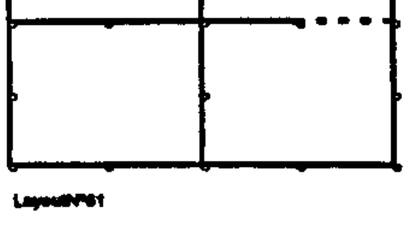


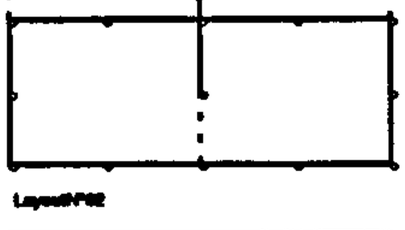


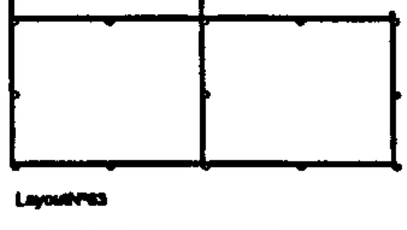


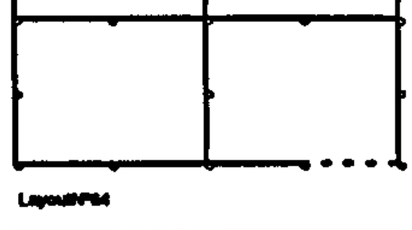


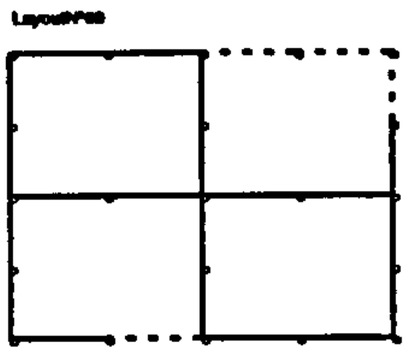


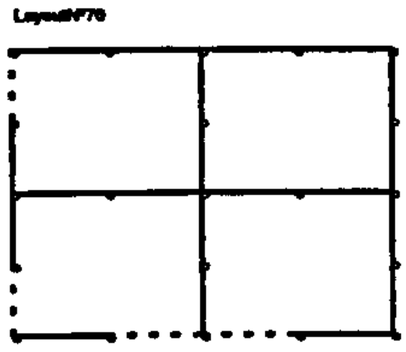


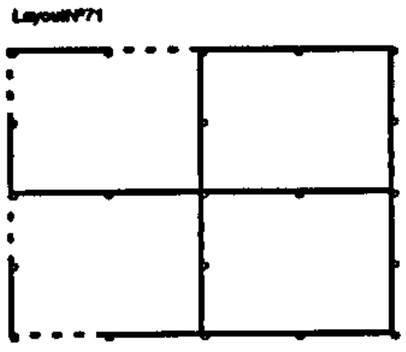


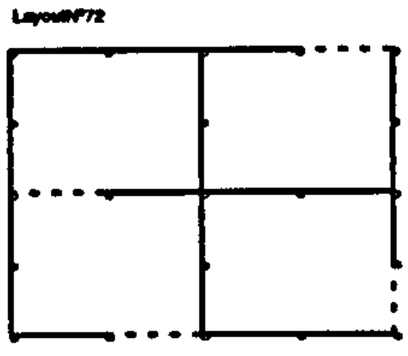


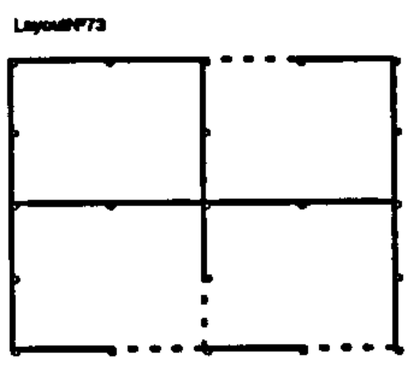


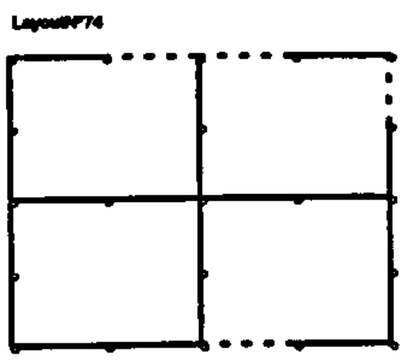




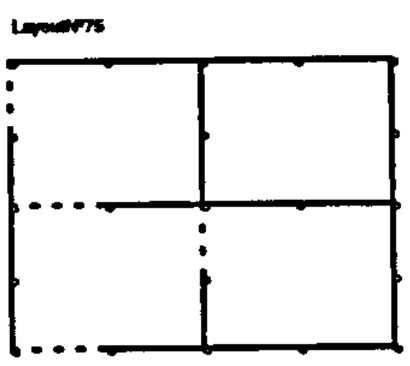


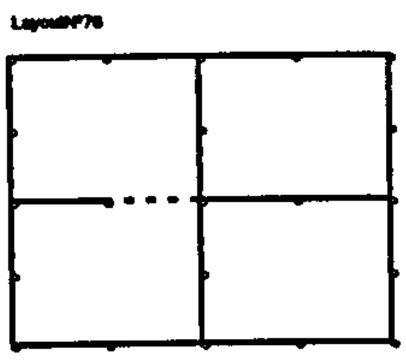


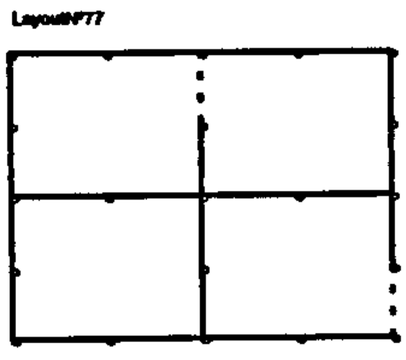


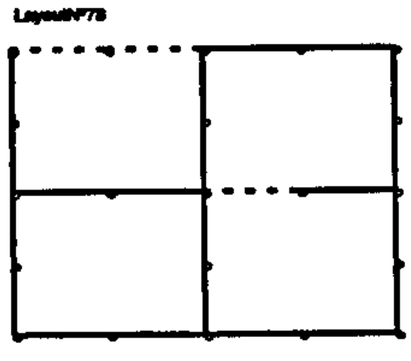


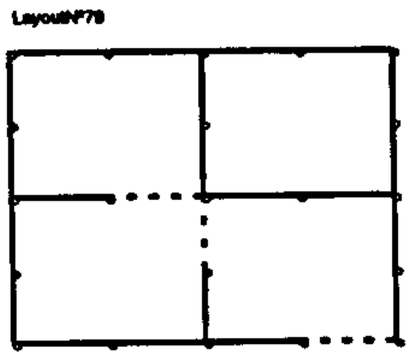
Testing_2 data set:

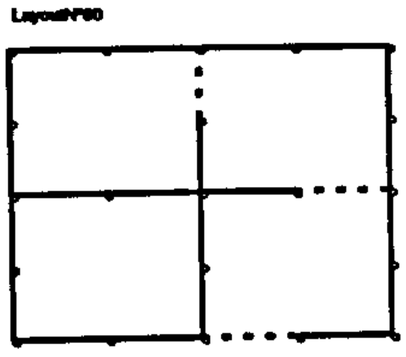


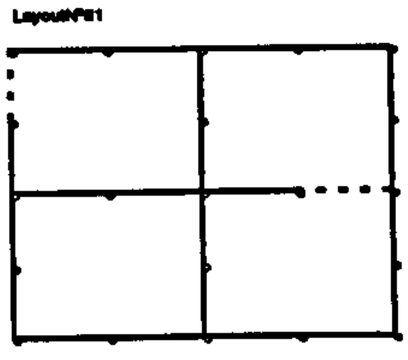


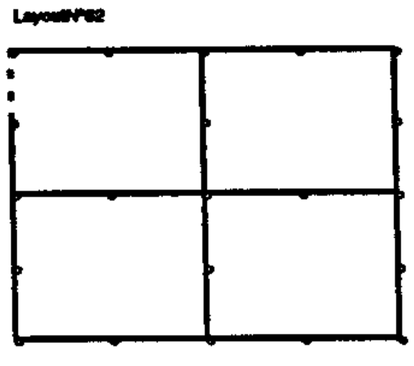


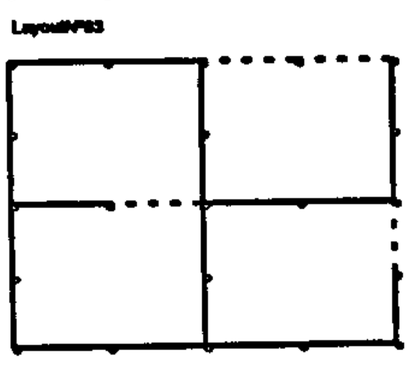


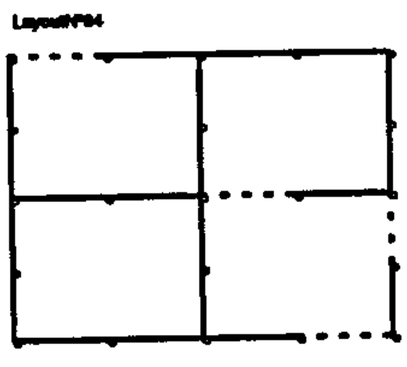


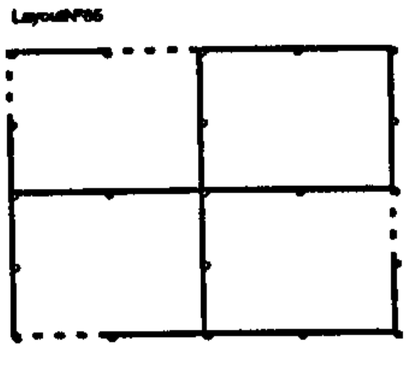












Appendix B

Artificial Neural Network Selected For Use in the Multi-Vehicle Case

The ANN selected for use in the multiple vehicle case is a backpropagation type ANN with an architecture based on REPRESENTATION 2, and LAYOUT_3 order numbering, and is represented in the Figure below. Only a few connections are represented and the elements are numbered as in the tables representing the weights associated with each connection. The ANN has 34 input elements (2, ..., 35), 20 hidden elements (36,... 55), and two output elements (56, 57). There is also a BIAS element which is connected to all other elements in the hidden and output layers. All other details are described in the following listing obtained from the package used to develop the ANN.

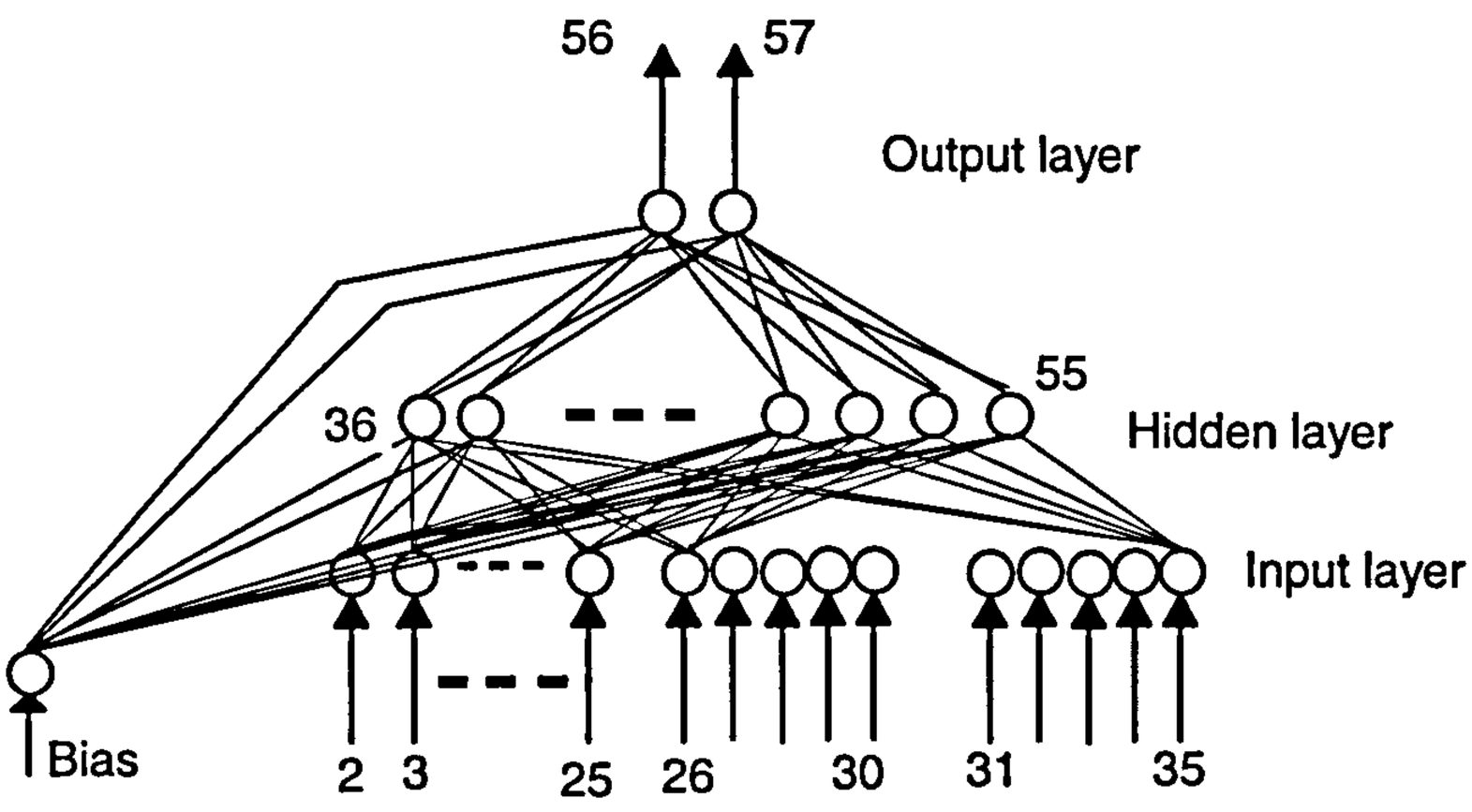


Figure B- 1: Schematic representation of BP ANN selected for use in the multi-vehicle case.

Output Elem.:																				
57	-0 2125	1.3006	3.5974	3.1251	-4.9264	0.0053	-0.065	-0.5404	1.016	0.0925	0.1859	0.7117	1.5514	-2.8014	-3.3067	0.6253	-0.0789	-0.2383	-0.0584	0.9967	-0.374
56	-0.3322	0.5511	0.5045	-1.021	-1.1201	2.8605	3.1951	-2.126	0.0089	-4.3171	3.0464	-0 6524	-1.0171	0 0682	0.5193	0.8864	3.1093	2.1596	4.2184	-1.6291	-2.1958
	1	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
	Bias	Ele. Hidden layer:																			

		Ele. Hidden layer:																		
		36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
Bias	1	-1.0668	-7.8892	-1.67	-4.1677	2.0397	-1.7947	2.7463	-0.9876	-4.7823	-3.3969	-1.3342	-1.6178	-6.0805	-8.6765	-1.109	-1.7762	-1.5036	-2.1868	-0.544	-1.2727
Input	2	-0.4809	-0.981	-0.5104	-0.3143	0.208	0.3442	0.0335	-0.3614	-1.8512	0.3066	-0.1636	-0.1692	-0.2127	-0.1585	-0.2533	0.033	-0.6464	-0.9425	-1.4636	-0.1312
Elements	3	-0.7730	-0.7849	-0.5993	-0.3232	-1.0988	0.1646	-0.2289	-0.1736	-0.64	-1.0669	0.155	-0.5345	-0.4046	0.4613	-0.3205	0.0619	-0.1058	-0.2433	0.5812	0.5266
	4	-0.4476	-1.6155	-0.4704	0.1953	-0.6035	0.6852	-0.0928	-0.1147	-1.1648	-0.3771	-0.1223	0.1032	-0.1826	0.5633	0.0774	-0.0717	-0.058	-0.5744	-0.3738	0.0962
	5	-0.3065	-0.4572	-1.3943	0.1876	0.1353	-1.6885	0.0705	-0.1263	0.9702	-0.9098	-0.0696	-0.8605	-0.0019	0.0456	-0.4883	-0.3034	-0.6139	0.0599	0.4073	-0.0835
	6	0.1028	-1.9364	0.2745	-0.5359	0.7041	0.0159	0.7411	-0.1107	0.0906	-0.4408	-0.0688	-0.4356	-1.0755	0.6482	-0.1844	0.1581	-0.2249	0.4902	0.0746	-0.3611
	7	-0.2356	-0.6955	0.0208	-0.5083	-0.212	0.6877	-0.3239	0.0212	-1.648	0.3243	-0.115	-0.5318	-0.5967	-0.3902	0.1031	-0.0646	0.352	-0.8936	-0.1244	0.0726
	8	0.1545	-0.3677	-0.7279	0.3328	0.5678	-0.6098	0.1929	0.0504	0.0937	0.182	-0.0415	0.2473	-0.0023	0.0157	-0.4179	-0.1795	-0.3156	-0.1579	-0.018	-0.2771
	9	0.6942	-0.6501	-0.0123	0.2474	0.4456	-0.5971	0.2764	-0.082	-0.1054	0.077	-0.4176	0.1787	0.1182	-3.1644	0.0204	-0.1394	-0.3941	0.1216	0.4395	-0.6525
	10	-0.5234	-0.4091	0.3056	-0.4321	0.1467	0.4496	-0.0873	-0.3726	-1.0056	0.0495	-0.044	-0.9684	-0.7432	-0.195	0.0807	-0.0129	0.5395	-0.3145	-0.0234	-0.0782
	11	-0.4207	0.065	-0.7762	-4.078	0.2592	-1.6876	-0.0865	-0.0862	-0.0722	-0.219	-0.3474	-0.2475	-0.152	-0.3505	-0.4006	-0.5905	-0.8103	-0.3255	0.7143	-0.1419
	12	-0.0358	-0.5825	-0.5999	0.3302	0.0754	-0.3307	-0.8232	-0.2152	-0.3536	-0.0485	-0.4631	-0.1765	-0.4662	-0.0635	-0.2404	-0.2939	-0.3472	-0.4074	-0.0155	-0.9555
	13	0.0882	-0.3359	-0.1018	-0.1208	-1.0995	0.1996	-0.8241	-0.0618	-0.8178	-0.5498	0.0722	0.1848	0.3425	-0.3758	0.1718	0.0933	0.2788	-0.0354	0.1744	0.3499
	14	0.0508	-0 2957	0.3798	-0.3068	0.4136	0.3383	-0.0092	-0.216	-0.9818	-0.1662	-0.4732	0.2895	-0.0272	0.1082	-0.0485	-0.3561	-0.6294	-0.3258	-0.8728	-0.4694
	15	0.0239	-0.2921	0 2373	-0.3258	0.1184	-1.4385	-0.6945	-0.0519	-0.4554	0.0531	-0.0958	-0.4625	-0.5301	0.2501	-0.3827	0.1069	-0.5697	-0.4348	-0.8233	0.2274
	16	0.0364	-1.7831	0.1287	0.2656	1.0441	-0.5345	-0.5654	-0.2226	-0.5539	0.0873	-0.6669	-0.2546	-0.4381	-0.237	-0.155	-0.2231	-0.0701	-0.5761	-1.5328	-0.866
	17	-0.3317	-0.5384	-0.8275	0.3665	-0.6252	0.0616	0.7445	-0.0248	-1.072	-0.0058	0.112	-0.2744	-0.5901	-0.7561	0.0488	0.3412	-0.1172	0.0837	-0.2255	0.0967
	18	-0.2053	0.1614	-0.3735	-0.1392	-0.7073	-0.0596	-0.2243	-0.0306	0.3692	-0.7985	-0.0596	-0.4286	-0.1826	0.0827	0.0147	-0.2258	-0.3567	0.5424	-0.9907	-0.3539
	19	0.0805	-0.1688	0.0368	-0.8396	-0.4905	-0.3361	-0.9202	-0.1302	0.1363	0.4232	0.016	0.3218	0.3339	-0.724	0.2013	-0.0772	0.0073	0.2101	0.0113	0.4781
	20	-0.2331	-0.3956	-0.0706	-0.5313	-0.804	0.7926	0.1801	-0.5031	-0.7909	-0.0699	-0.0481	0.4246	-0.1601	-0.9342	-0.1459	-0.1536	0.6543	0.1278	0.7109	0.1408
	21	-0.1650	-0.332	-0.7624	-0.8424	-0.3438	-0.489	0.2918	-0.0609	0.0486	-0.1955	-0.0019	-0.3197	-0.6018	0.012	-0.2613	0.2068	-0.503	-0.0797	0.1548	0.5348
	22	-0.0560	0.8515	0.2139	-0.4063	-0.0911	-0.1123	0.6829	-0.0152	0.0882	0.2271	-0.231	0.2254	-0.2413	-1.2016	-0.1941	-0.5893	-0.124	-0.3675	-0.2293	-0.2833
	23	-0.3590	0.5836	-0.0152	-1.2576	-1.5516	0.4293	0.2895	-0.0882	-1.0271	0.3769	-0.0267	0.226	-0 6407	-1.2429	-0.1694	0.0138	0.3436	-0.2855	-0.1006	0.1052
	24	-0.1800	-1.0199	-0.3002	-0.1059	0.2254	0.1883	0.3851	-0.1759	-0.7648	0.5823	-0.3876	0.1101	-0.4183	-1.5571	-0.2431	-0.0728	-0.1393	-0.5362	-0.7222	-0.9549
	25	0.2080	-0.5747	0.0361	-1.4072	-0.6431	-1.2557	-0.0884	0.2365	-0.1897	-0.3758	0.0286	0.4919	-0.1209	0.0326	-0.0687	0.0508	-0.5644	0.4077	-0.6452	0.6654
	26	-0.7782	-1.5014	3.6579	0.0823	1.0905	-1.2774	2.6572	-0.6878	-0.7237	1.6816	-0.2051	2.9484	-6.7289	-2.797	-0.5303	0.2825	-0.2516	2.0706	-1.308	1.3874
	27	-0.6810	1.6436	2.7082	1 8627	1.1723	-1.2221	1.1798	-0.3115	-0.04	-0.0009	0.3295	0.6975	-3.4123	-1.7763	0.1412	-0.5874	-0.0953	1.505	-1.7833	0.7244
	28	-1.0251	4.923	-4.4114	-5.2121	-2.8701	1.3156	-6.3961	-0.8455	3.7106	-5 6932	-1.166	-2.3426	-1.2914	3.1806	-0.0011	-4.5877	1.4451	-4.6577	-1.129	-2.1221
	29	-0.9004	4.842	-4.0402	-5.1077	-1.5579	-2.7208	-5.1107	-0.7572	-5.2704	-2.7924	-0.3007	-4.0604	-2.6272	3.7882	-1.751	1.3867	-2.9063	-6.3844	-0.0818	-1.7942
	30	0.5998	1.7957	0.1526	-3.2453	-2.7702	-3.3291	-1.3212	-0.3695	-1.5669	-2.9176	-0.624	6.2098	7.5111	-0.5421	-0.151	1.9881	-1.1038	4.0154	-1.2013	-0.2975
	31	0.5493	-7.0644	-5.0034	3.4661	0.142	-0.2661	-2.1418	0.7711	0.1191	-1.1044	0.2053	-3.465	4.595	2.9983	-0.3165	-0.1186	-0.2006	-1.1497	0.6199	0.6888
	32	0.3655	-2.9957	-1.7809	2.1385	-0.4261	0.1519	-0.6944	-0.1797	0.0879	-0.8746	-0.1954	-0.7247	2.5967	1.111	0.541	0.5269	0.3153	-0.8846	-0.5871	-0.3599
	33	-0.5425	-1.2643	-0.1214	1.1357	-5.9191	0.59	-2.4346	-0.4499	-1.4335	6.3179	-0.089	-0.9233	0.8245	0.5284	-0.9427	0.2925	-1.6051	-0.9003	-0.7142	1.9001
	34	-1.3694	-0.3668	-1.1714	0.9248	-1.3558	-1.5326	-4.6137	-0.0521	3.9426	-2.8915	0.6385	0.2199	1.4103	1.3489	-0.8383	-5.0954	-1.8914	0.9181	-0.3077	1.6441
	35	-0.3990	-0.372	-0.0767	-0 2878	-5.3742	-0 8779	4.1488	-0.4945	1.5553	4 0819	-0.6521	-0.1505	0.9014	0.2623	-0.5197	-2.8	-1.4063	0.0065	-0 2326	-3.0554

Table B- 1: Table representing the weights associated with each connection in the selected BP ANN.

```
Title: NxyFBc3.nnd (FEUP240395): H1=20;IN=34;OUT=2 (L=0.3;Mom=0.6;Epo=50)
       Display Mode: Network                          Type: Hetero-Associative
      Display Style: default
   Control Strategy: backprop              L/R Schedule: backprop
      100002 Learn                0 Recall                    0 Layer
          50 Aux 1                0 Aux 2                     0 Aux 3
L/R Schedule: backprop
   Recall Step                1          0          0          0          0
     Input Clamp         0.0000     0.0000     0.0000     0.0000     0.0000
     Firing Density    100.0000     0.0000     0.0000     0.0000     0.0000
     Temperature         0.0000     0.0000     0.0000     0.0000     0.0000
     Gain                1.0000     0.0000     0.0000     0.0000     0.0000
     Modifier            1.0000     0.0000     0.0000     0.0000     0.0000
   Learn Step              5000          0          0          0          0
     Coefficient 1       0.3000     0.0000     0.0000     0.0000     0.0000
     Coefficient 2       0.6000     0.0000     0.0000     0.0000     0.0000
     Coefficient 3       0.0000     0.0000     0.0000     0.0000     0.0000
     Temperature         0.0000     0.0000     0.0000     0.0000     0.0000
IO Parameters
         Learn Data: File Rand. (trfbw21c)
        Recall Data: File Seq. (tefw21c)
        Result File:  Desired Output, Output
     UserIO Program: userio
         I/P Ranges:     0.0000,     1.0000
         O/P Ranges:     0.0000,     1.0000
      I/P Start Col:          1              O/P Start Col:         35
       MinMax Table: default                    # entries:          0
Layer: 1
        PEs: 1                                        Sum: Sum
    Spacing: 5       F' offset: 0.00            Transfer: Linear
      Shape: Square                               Output: Direct
      Scale: 1.00     Low Limit: 0.00          Error Func: standard
     Offset: 0.00    High Limit: 9999.00            Learn: --None--
   Init Low: -0.100   Init High: 0.100       L/R Schedule: (Network)
   Winner 1: None                              Winner 2: None
   PE: Bias
       1.000 Error Factor
       0.000 Sum             1.000 Transfer          1.000 Output
           0 Weights      -105.941 Error             0.000 Current Error
Layer: In
        PEs: 34                                       Sum: Sum
    Spacing: 1       F' offset: 0.00            Transfer: Linear
      Shape: Square                               Output: Direct
      Scale: 1.00     Low Limit: -9999.00      Error Func: standard
     Offset: 0.00    High Limit: 9999.00            Learn: --None--
   Init Low: -0.100   Init High: 0.100       L/R Schedule: (Network)
   Winner 1: None                              Winner 2: None
   PE: 2
       1.000 Error Factor
       1.000 Sum             1.000 Transfer          1.000 Output
           0 Weights        -0.003 Error             0.000 Current Error
           Input PE  Input Value Weight Type Delta Weight
   PE: 3
       1.000 Error Factor
       0.000 Sum             0.000 Transfer          0.000 Output
           0 Weights         0.005 Error             0.000 Current Error
           Input PE  Input Value Weight Type Delta Weight
   PE: 4
       1.000 Error Factor
       1.000 Sum             1.000 Transfer          1.000 Output
           0 Weights         0.005 Error             0.000 Current Error
           Input PE  Input Value Weight Type Delta Weight
PE: 5
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.014 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 6
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.001 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 7
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.002 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 8
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.000 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 9
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.000 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 10
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.006 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 11
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.001 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 12
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.005 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 13
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.006 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 14
    1.000 Error Factor
    1.000 Sum             1.000 Transfer          1.000 Output
        0 Weights        -0.002 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 15
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.000 Error             0.000 Current Error
```

```
        Input PE  Input Value Weight Type Delta Weight
PE: 16
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.012 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 17
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.007 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 18
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.006 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 19
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.008 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 20
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.015 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 21
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.009 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 22
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.009 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 23
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.022 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 24
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.000 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 25
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.016 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 26
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.148 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 27
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.065 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 28
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.020 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 29
    1.000 Error Factor
    1.000 Sum             1.000 Transfer          1.000 Output
        0 Weights        -0.043 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 30
    1.000 Error Factor
    1.000 Sum             1.000 Transfer          1.000 Output
        0 Weights        -0.011 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 31
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.119 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 32
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.055 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 33
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights         0.008 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 34
    1.000 Error Factor
    1.000 Sum             1.000 Transfer          1.000 Output
        0 Weights         0.016 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
PE: 35
    1.000 Error Factor
    0.000 Sum             0.000 Transfer          0.000 Output
        0 Weights        -0.017 Error             0.000 Current Error
        Input PE  Input Value Weight Type Delta Weight
Layer: Hidden 1
      PEs: 20                                        Sum: Sum
  Spacing: 1        F' offset: 0.00            Transfer: Sigmoid
    Shape: Square                                Output: Direct
    Scale: 1.00      Low Limit: -9999.00      Error Func: standard
   Offset: 0.00     High Limit: 9999.00           Learn: Delta-Rule
 Init Low: -0.100    Init High: 0.100       L/R Schedule: (Network)
 Winner 1: None                                 Winner 2: None
PE: 36
    1.000 Error Factor
   -3.617 Sum             0.026 Transfer          0.026 Output
       35 Weights        -0.000 Error            -0.002 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.0668 V-a   +0.0014
               2   +1.0000   -0.4809 V-a   -0.0000
```

```
  3   +0.0000   -0.7730 V-a   -0.0000
  4   +1.0000   -0.4476 V-a   -0.0000
  5   +0.0000   -0.3065 V-a   -0.0000
  6   +0.0000   +0.1028 V-a   +0.0000
  7   +0.0000   -0.2356 V-a   -0.0000
  8   +0.0000   +0.1545 V-a   -0.0000
  9   +0.0000   +0.6942 V-a   +0.0000
 10   +0.0000   -0.5234 V-a   -0.0000
 11   +0.0000   -0.4207 V-a   -0.0001
 12   +0.0000   -0.0358 V-a   +0.0000
 13   +0.0000   +0.0882 V-a   -0.0001
 14   +1.0000   +0.0508 V-a   -0.0001
 15   +0.0000   +0.0239 V-a   +0.0000
 16   +0.0000   +0.0364 V-a   -0.0000
 17   +0.0000   -0.3317 V-a   +0.0000
 18   +0.0000   -0.2053 V-a   -0.0001
 19   +0.0000   +0.0805 V-a   +0.0000
 20   +0.0000   -0.2331 V-a   +0.0000
 21   +0.0000   -0.1650 V-a   -0.0001
 22   +0.0000   -0.0560 V-a   -0.0000
 23   +0.0000   -0.3590 V-a   -0.0000
 24   +0.0000   -0.1800 V-a   +0.0000
 25   +0.0000   +0.2080 V-a   +0.0016
 26   +0.0000   -0.7782 V-a   -0.0001
 27   +0.0000   -0.6810 V-a   +0.0015
 28   +0.0000   -1.0251 V-a   -0.0000
 29   +1.0000   -0.9004 V-a   -0.0000
 30   +1.0000   +0.5998 V-a   +0.0015
 31   +0.0000   +0.5493 V-a   +0.0016
 32   +0.0000   +0.3655 V-a   -0.0002
 33   +0.0000   -0.5425 V-a   -0.0001
 34   +1.0000   -1.3694 V-a   -0.0002
 35   +0.0000   -0.3990 V-a   +0.0000
PE: 37
    1.000 Error Factor
   -4.511 Sum              0.011 Transfer          0.011 Output
       35 Weights          0.000 Error             0.042 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -7.8892 V-a   +0.0000
               2   +1.0000   -0.9810 V-a   +0.0001
               3   +0.0000   -0.7849 V-a   -0.0000
               4   +1.0000   -1.6155 V-a   +0.0001
               5   +0.0000   -0.4572 V-a   -0.0000
               6   +0.0000   -1.9364 V-a   +0.0000
               7   +0.0000   -0.6955 V-a   +0.0000
               8   +0.0000   -0.3677 V-a   -0.0000
               9   +0.0000   -0.6501 V-a   -0.0000
              10   +0.0000   -0.4091 V-a   -0.0000
              11   +0.0000   +0.0650 V-a   -0.0000
              12   +0.0000   -0.5825 V-a   +0.0000
              13   +0.0000   -0.3359 V-a   -0.0000
              14   +1.0000   -0.2957 V-a   +0.0001
              15   +0.0000   -0.2921 V-a   +0.0000
              16   +0.0000   -1.7831 V-a   -0.0000
              17   +0.0000   -0.5384 V-a   -0.0000
              18   +0.0000   +0.1614 V-a   -0.0000
              19   +0.0000   -0.1688 V-a   +0.0000
              20   +0.0000   -0.3956 V-a   +0.0000
              21   +0.0000   -0.3320 V-a   -0.0000
              22   +0.0000   +0.8515 V-a   -0.0000
              23   +0.0000   +0.5836 V-a   -0.0001
              24   +0.0000   -1.0199 V-a   -0.0000
              25   +0.0000   -0.5747 V-a   -0.0000
              26   +0.0000   -1.5014 V-a   -0.0000
              27   +0.0000   +1.6436 V-a   -0.0001
              28   +0.0000   +4.9230 V-a   -0.0001
              29   +1.0000   +4.8420 V-a   +0.0001
              30   +1.0000   +1.7957 V-a   +0.0001
              31   +0.0000   -7.0644 V-a   +0.0000
              32   +0.0000   -2.9957 V-a   -0.0001
              33   +0.0000   -1.2643 V-a   -0.0000
              34   +1.0000   -0.3668 V-a   +0.0000
              35   +0.0000   -0.3720 V-a   -0.0000
PE: 38
    1.000 Error Factor
   -7.329 Sum              0.001 Transfer          0.001 Output
       35 Weights          0.000 Error             0.107 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.6700 V-a   -0.0000
               2   +1.0000   -0.5104 V-a   -0.0001
               3   +0.0000   -0.5993 V-a   +0.0000
               4   +1.0000   -0.4704 V-a   +0.0000
               5   +0.0000   -1.3943 V-a   -0.0000
               6   +0.0000   +0.2745 V-a   +0.0000
               7   +0.0000   +0.0208 V-a   +0.0000
               8   +0.0000   -0.7279 V-a   -0.0000
               9   +0.0000   -0.0123 V-a   +0.0000
              10   +0.0000   +0.3056 V-a   -0.0001
              11   +0.0000   -0.7762 V-a   -0.0003
              12   +0.0000   -0.5999 V-a   -0.0000
              13   +0.0000   -0.1018 V-a   -0.0000
              14   +1.0000   +0.3798 V-a   -0.0002
              15   +0.0000   +0.2373 V-a   +0.0000
              16   +0.0000   +0.1287 V-a   -0.0001
              17   +0.0000   -0.8275 V-a   -0.0000
              18   +0.0000   -0.3735 V-a   -0.0000
              19   +0.0000   +0.0368 V-a   +0.0000
              20   +0.0000   -0.0706 V-a   +0.0000
              21   +0.0000   -0.7624 V-a   -0.0000
              22   +0.0000   +0.2139 V-a   -0.0001
              23   +0.0000   -0.0152 V-a   -0.0000
              24   +0.0000   -0.3002 V-a   -0.0000
              25   +0.0000   +0.0361 V-a   +0.0003
              26   +0.0000   +3.6579 V-a   -0.0000
              27   +0.0000   +2.7082 V-a   -0.0001
              28   +0.0000   -4.4114 V-a   -0.0000
              29   +1.0000   -4.0402 V-a   +0.0000
              30   +1.0000   +0.1526 V-a   +0.0001
              31   +0.0000   -5.0034 V-a   +0.0003
              32   +0.0000   -1.7809 V-a   -0.0004
              33   +0.0000   -0.1214 V-a   -0.0003
              34   +1.0000   -1.1714 V-a   -0.0004
              35   +0.0000   -0.0767 V-a   -0.0000
PE: 39
    1.000 Error Factor
  -12.022 Sum              0.000 Transfer          0.000 Output
```

```
35 Weights          0.000 Error            -0.037 Current Error
 Input PE  Input Value Weight Type Delta Weight
     Bias   +1.0000   -4.1677 V-a   -0.0000
        2   +1.0000   -0.3143 V-a   +0.0007
        3   +0.0000   -0.3232 V-a   +0.0000
        4   +1.0000   +0.1953 V-a   +0.0000
        5   +0.0000   +0.1876 V-a   +0.0000
        6   +0.0000   -0.5359 V-a   -0.0000
        7   +0.0000   -0.5083 V-a   +0.0000
        8   +0.0000   +0.3328 V-a   +0.0000
        9   +0.0000   +0.2474 V-a   -0.0000
       10   +0.0000   -0.4321 V-a   +0.0007
       11   +0.0000   -4.0780 V-a   +0.0000
       12   +0.0000   +0.3302 V-a   -0.0000
       13   +0.0000   -0.1208 V-a   -0.0000
       14   +1.0000   -0.3068 V-a   +0.0000
       15   +0.0000   -0.3258 V-a   -0.0000
       16   +0.0000   +0.2656 V-a   +0.0007
       17   +0.0000   +0.3665 V-a   -0.0000
       18   +0.0000   -0.1392 V-a   +0.0000
       19   +0.0000   -0.8396 V-a   +0.0000
       20   +0.0000   -0.5313 V-a   -0.0000
       21   +0.0000   -0.8424 V-a   +0.0000
       22   +0.0000   -0.4063 V-a   +0.0007
       23   +0.0000   -1.2576 V-a   +0.0000
       24   +0.0000   -0.1059 V-a   -0.0000
       25   +0.0000   -1.4072 V-a   -0.0007
       26   +0.0000   +0.0823 V-a   -0.0000
       27   +0.0000   +1.8627 V-a   -0.0000
       28   +0.0000   -5.2121 V-a   +0.0000
       29   +1.0000   -5.1077 V-a   -0.0000
       30   +1.0000   -3.2453 V-a   -0.0007
       31   +0.0000   +3.4661 V-a   -0.0007
       32   +0.0000   +2.1385 V-a   +0.0007
       33   +0.0000   +1.1357 V-a   +0.0000
       34   +1.0000   +0.9248 V-a   +0.0007
       35   +0.0000   -0.2878 V-a   +0.0000
PE: 40
    1.000 Error Factor
   -3.619 Sum              0.026 Transfer           0.026 Output
       35 Weights         -0.004 Error             -0.138 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   +2.0397 V-a   -0.0010
               2   +1.0000   +0.2080 V-a   -0.0011
               3   +0.0000   -1.0988 V-a   -0.0000
               4   +1.0000   -0.6035 V-a   -0.0010
               5   +0.0000   +0.1353 V-a   +0.0000
               6   +0.0000   +0.7041 V-a   -0.0000
               7   +0.0000   -0.2120 V-a   -0.0000
               8   +0.0000   +0.5678 V-a   +0.0000
               9   +0.0000   +0.4456 V-a   +0.0000
              10   +0.0000   +0.1467 V-a   -0.0001
              11   +0.0000   +0.2592 V-a   -0.0000
              12   +0.0000   +0.0754 V-a   +0.0000
              13   +0.0000   -1.0995 V-a   -0.0000
              14   +1.0000   +0.4136 V-a   -0.0010
              15   +0.0000   +0.1184 V-a   -0.0000
              16   +0.0000   +1.0441 V-a   -0.0001
              17   +0.0000   -0.6252 V-a   +0.0000
              18   +0.0000   -0.7073 V-a   -0.0000
              19   +0.0000   -0.4905 V-a   +0.0000
              20   +0.0000   -0.8040 V-a   +0.0000
              21   +0.0000   -0.3438 V-a   -0.0000
              22   +0.0000   -0.0911 V-a   -0.0001
              23   +0.0000   -1.5516 V-a   +0.0000
              24   +0.0000   +0.2254 V-a   +0.0000
              25   +0.0000   -0.6431 V-a   +0.0002
              26   +0.0000   +1.0905 V-a   -0.0000
              27   +0.0000   +1.1723 V-a   +0.0001
              28   +0.0000   -2.8701 V-a   -0.0000
              29   +1.0000   -1.5579 V-a   -0.0010
              30   +1.0000   -2.7702 V-a   -0.0009
              31   +0.0000   +0.1420 V-a   +0.0002
              32   +0.0000   -0.4261 V-a   -0.0001
              33   +0.0000   -5.9191 V-a   -0.0000
              34   +1.0000   -1.3558 V-a   -0.0012
              35   +0.0000   -5.3742 V-a   -0.0000
PE: 41
    1.000 Error Factor
   -8.009 Sum              0.000 Transfer           0.000 Output
       35 Weights         -0.000 Error             -0.155 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.7947 V-a   -0.0001
               2   +1.0000   +0.3442 V-a   -0.0000
               3   +0.0000   +0.1646 V-a   -0.0000
               4   +1.0000   +0.6852 V-a   -0.0000
               5   +0.0000   -1.6885 V-a   +0.0000
               6   +0.0000   +0.0159 V-a   -0.0000
               7   +0.0000   +0.6877 V-a   -0.0000
               8   +0.0000   -0.6098 V-a   +0.0000
               9   +0.0000   -0.5971 V-a   +0.0000
              10   +0.0000   +0.4496 V-a   -0.0000
              11   +0.0000   -1.6876 V-a   -0.0000
              12   +0.0000   -0.3307 V-a   +0.0000
              13   +0.0000   +0.1996 V-a   -0.0000
              14   +1.0000   +0.3383 V-a   -0.0000
              15   +0.0000   -1.4385 V-a   -0.0000
              16   +0.0000   -0.5345 V-a   -0.0000
              17   +0.0000   +0.0616 V-a   +0.0000
              18   +0.0000   -0.0596 V-a   -0.0000
              19   +0.0000   -0.3361 V-a   -0.0000
              20   +0.0000   +0.7926 V-a   +0.0000
              21   +0.0000   -0.4890 V-a   -0.0000
              22   +0.0000   -0.1123 V-a   -0.0000
              23   +0.0000   +0.4293 V-a   -0.0000
              24   +0.0000   +0.1883 V-a   +0.0000
              25   +0.0000   -1.2557 V-a   +0.0000
              26   +0.0000   -1.2774 V-a   +0.0000
              27   +0.0000   -1.2221 V-a   -0.0000
              28   +0.0000   +1.3156 V-a   -0.0000
              29   +1.0000   -2.7208 V-a   -0.0000
              30   +1.0000   -3.3291 V-a   -0.0000
              31   +0.0000   -0.2661 V-a   +0.0000
              32   +0.0000   +0.1519 V-a   -0.0000
              33   +0.0000   +0.5900 V-a   -0.0000
              34   +1.0000   -1.5326 V-a   -0.0001
```

```
            35   +0.0000   -0.8779 V-a   +0.0000
PE: 42
    1.000 Error Factor
   -8.367 Sum              0.000 Transfer          0.000 Output
       35 Weights          0.000 Error             0.092 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   +2.7463 V-a   -0.0006
               2   +1.0000   +0.0335 V-a   +0.0003
               3   +0.0000   -0.2289 V-a   +0.0000
               4   +1.0000   -0.0928 V-a   +0.0000
               5   +0.0000   +0.0705 V-a   -0.0000
               6   +0.0000   +0.7411 V-a   -0.0000
               7   +0.0000   -0.3239 V-a   +0.0000
               8   +0.0000   +0.1929 V-a   -0.0000
               9   +0.0000   +0.2764 V-a   -0.0000
              10   +0.0000   -0.0873 V-a   +0.0002
              11   +0.0000   -0.0865 V-a   -0.0000
              12   +0.0000   -0.8232 V-a   -0.0000
              13   +0.0000   -0.8241 V-a   +0.0000
              14   +1.0000   -0.0092 V-a   +0.0000
              15   +0.0000   -0.6945 V-a   -0.0000
              16   +0.0000   -0.5654 V-a   +0.0002
              17   +0.0000   +0.7445 V-a   -0.0000
              18   +0.0000   -0.2243 V-a   +0.0000
              19   +0.0000   -0.9202 V-a   -0.0000
              20   +0.0000   +0.1801 V-a   -0.0000
              21   +0.0000   +0.2918 V-a   +0.0000
              22   +0.0000   +0.6829 V-a   +0.0003
              23   +0.0000   +0.2895 V-a   -0.0000
              24   +0.0000   +0.3851 V-a   -0.0000
              25   +0.0000   -0.0884 V-a   -0.0008
              26   +0.0000   +2.6572 V-a   -0.0000
              27   +0.0000   +1.1798 V-a   -0.0005
              28   +0.0000   -6.3961 V-a   -0.0000
              29   +1.0000   -5.1107 V-a   -0.0000
              30   +1.0000   -1.3212 V-a   -0.0008
              31   +0.0000   -2.1418 V-a   -0.0008
              32   +0.0000   -0.6944 V-a   +0.0002
              33   +0.0000   -2.4346 V-a   -0.0000
              34   +1.0000   -4.6137 V-a   +0.0002
              35   +0.0000   +4.1488 V-a   -0.0000
PE: 43
    1.000 Error Factor
   -2.862 Sum              0.054 Transfer          0.054 Output
       35 Weights          0.001 Error             0.018 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -0.9876 V-a   +0.0011
               2   +1.0000   -0.3614 V-a   +0.0003
               3   +0.0000   -0.1736 V-a   +0.0000
               4   +1.0000   -0.1147 V-a   +0.0003
               5   +0.0000   -0.1263 V-a   -0.0000
               6   +0.0000   -0.1107 V-a   +0.0000
               7   +0.0000   +0.0212 V-a   +0.0000
               8   +0.0000   +0.0504 V-a   -0.0000
               9   +0.0000   -0.0820 V-a   +0.0000
              10   +0.0000   -0.3726 V-a   -0.0000
              11   +0.0000   -0.0862 V-a   -0.0001
              12   +0.0000   -0.2152 V-a   -0.0000
              13   +0.0000   -0.0618 V-a   -0.0001
              14   +1.0000   -0.2160 V-a   +0.0002
              15   +0.0000   -0.0519 V-a   +0.0000
              16   +0.0000   -0.2226 V-a   -0.0000
              17   +0.0000   -0.0248 V-a   -0.0000
              18   +0.0000   -0.0306 V-a   -0.0001
              19   +0.0000   -0.1302 V-a   +0.0000
              20   +0.0000   -0.5031 V-a   +0.0000
              21   +0.0000   -0.0609 V-a   -0.0001
              22   +0.0000   -0.0152 V-a   -0.0000
              23   +0.0000   -0.0882 V-a   -0.0001
              24   +0.0000   -0.1759 V-a   -0.0000
              25   +0.0000   +0.2365 V-a   +0.0011
              26   +0.0000   -0.6878 V-a   -0.0001
              27   +0.0000   -0.3115 V-a   +0.0009
              28   +0.0000   -0.8455 V-a   -0.0001
              29   +1.0000   -0.7572 V-a   +0.0002
              30   +1.0000   -0.3695 V-a   +0.0013
              31   +0.0000   +0.7711 V-a   +0.0012
              32   +0.0000   -0.1797 V-a   -0.0003
              33   +0.0000   -0.4499 V-a   -0.0001
              34   +1.0000   -0.0521 V-a   -0.0000
              35   +0.0000   -0.4945 V-a   +0.0000
PE: 44
    1.000 Error Factor
  -11.675 Sum              0.000 Transfer          0.000 Output
       35 Weights          0.000 Error             0.210 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -4.7823 V-a   +0.0001
               2   +1.0000   -1.8512 V-a   +0.0000
               3   +0.0000   -0.6400 V-a   -0.0000
               4   +1.0000   -1.1648 V-a   +0.0000
               5   +0.0000   +0.9702 V-a   -0.0000
               6   +0.0000   +0.0906 V-a   +0.0000
               7   +0.0000   -1.6480 V-a   +0.0000
               8   +0.0000   +0.0937 V-a   +0.0000
               9   +0.0000   -0.1054 V-a   +0.0000
              10   +0.0000   -1.0056 V-a   +0.0000
              11   +0.0000   -0.0722 V-a   +0.0000
              12   +0.0000   -0.3536 V-a   -0.0000
              13   +0.0000   -0.8178 V-a   +0.0000
              14   +1.0000   -0.9818 V-a   +0.0000
              15   +0.0000   -0.4554 V-a   +0.0000
              16   +0.0000   -0.5539 V-a   +0.0000
              17   +0.0000   -1.0720 V-a   -0.0000
              18   +0.0000   +0.3692 V-a   +0.0000
              19   +0.0000   +0.1363 V-a   -0.0000
              20   +0.0000   -0.7909 V-a   -0.0000
              21   +0.0000   +0.0486 V-a   +0.0000
              22   +0.0000   +0.0882 V-a   +0.0000
              23   +0.0000   -1.0271 V-a   +0.0000
              24   +0.0000   -0.7648 V-a   -0.0000
              25   +0.0000   -0.1897 V-a   -0.0000
              26   +0.0000   -0.7237 V-a   +0.0000
              27   +0.0000   -0.0400 V-a   +0.0001
              28   +0.0000   +3.7106 V-a   +0.0000
              29   +1.0000   -5.2704 V-a   +0.0000
              30   +1.0000   -1.5669 V-a   +0.0000
```

```
              31   +0.0000   +0.1191 V-a   -0.0000
              32   +0.0000   +0.0879 V-a   +0.0001
              33   +0.0000   -1.4335 V-a   +0.0000
              34   +1.0000   +3.9426 V-a   +0.0001
              35   +0.0000   +1.5553 V-a   -0.0000
PE: 45
    1.000 Error Factor
  -12.236 Sum             0.000 Transfer          0.000 Output
       35 Weights         0.000 Error            -0.143 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -3.3969 V-a   +0.0002
               2   +1.0000   +0.3066 V-a   -0.0000
               3   +0.0000   -1.0669 V-a   -0.0000
               4   +1.0000   -0.3771 V-a   +0.0000
               5   +0.0000   -0.9098 V-a   +0.0000
               6   +0.0000   -0.4408 V-a   +0.0000
               7   +0.0000   +0.3243 V-a   -0.0000
               8   +0.0000   +0.1820 V-a   +0.0000
               9   +0.0000   +0.0770 V-a   +0.0000
              10   +0.0000   +0.0495 V-a   +0.0002
              11   +0.0000   -0.2190 V-a   +0.0002
              12   +0.0000   -0.0485 V-a   +0.0000
              13   +0.0000   -0.5498 V-a   -0.0000
              14   +1.0000   -0.1662 V-a   -0.0000
              15   +0.0000   +0.0531 V-a   +0.0000
              16   +0.0000   +0.0873 V-a   +0.0002
              17   +0.0000   -0.0058 V-a   +0.0000
              18   +0.0000   -0.7985 V-a   -0.0000
              19   +0.0000   +0.4232 V-a   +0.0000
              20   +0.0000   -0.0699 V-a   +0.0002
              21   +0.0000   -0.1955 V-a   -0.0000
              22   +0.0000   +0.2271 V-a   -0.0000
              23   +0.0000   +0.3769 V-a   +0.0000
              24   +0.0000   +0.5823 V-a   +0.0000
              25   +0.0000   -0.3758 V-a   +0.0000
              26   +0.0000   +1.6816 V-a   +0.0002
              27   +0.0000   -0.0009 V-a   -0.0000
              28   +0.0000   -5.6932 V-a   +0.0000
              29   +1.0000   -2.7924 V-a   +0.0002
              30   +1.0000   -2.9176 V-a   -0.0001
              31   +0.0000   -1.1044 V-a   -0.0000
              32   +0.0000   -0.8746 V-a   +0.0002
              33   +0.0000   +6.3179 V-a   +0.0002
              34   +1.0000   -2.8915 V-a   +0.0002
              35   +0.0000   +4.0819 V-a   +0.0002
PE: 46
    1.000 Error Factor
   -2.387 Sum             0.084 Transfer          0.084 Output
       35 Weights         0.003 Error             0.045 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.3342 V-a   +0.0014
               2   +1.0000   -0.1636 V-a   +0.0010
               3   +0.0000   +0.1550 V-a   +0.0000
               4   +1.0000   -0.1223 V-a   +0.0010
               5   +0.0000   -0.0696 V-a   -0.0000
               6   +0.0000   -0.0688 V-a   +0.0000
               7   +0.0000   -0.1150 V-a   +0.0000
               8   +0.0000   -0.0415 V-a   -0.0000
               9   +0.0000   -0.4176 V-a   -0.0000
              10   +0.0000   -0.0440 V-a   -0.0000
              11   +0.0000   -0.3474 V-a   -0.0001
              12   +0.0000   -0.4631 V-a   -0.0000
              13   +0.0000   +0.0722 V-a   -0.0000
              14   +1.0000   -0.4732 V-a   +0.0010
              15   +0.0000   -0.0958 V-a   +0.0000
              16   +0.0000   -0.6669 V-a   -0.0000
              17   +0.0000   +0.1120 V-a   -0.0000
              18   +0.0000   -0.0596 V-a   -0.0000
              19   +0.0000   +0.0160 V-a   +0.0000
              20   +0.0000   -0.0481 V-a   -0.0000
              21   +0.0000   -0.0019 V-a   -0.0000
              22   +0.0000   -0.2310 V-a   +0.0000
              23   +0.0000   -0.0267 V-a   -0.0001
              24   +0.0000   -0.3876 V-a   -0.0000
              25   +0.0000   +0.0286 V-a   +0.0005
              26   +0.0000   -0.2051 V-a   -0.0001
              27   +0.0000   +0.3295 V-a   +0.0004
              28   +0.0000   -1.1660 V-a   -0.0001
              29   +1.0000   -0.3007 V-a   +0.0010
              30   +1.0000   -0.6240 V-a   +0.0015
              31   +0.0000   +0.2053 V-a   +0.0005
              32   +0.0000   -0.1954 V-a   -0.0002
              33   +0.0000   -0.0890 V-a   -0.0000
              34   +1.0000   +0.6385 V-a   +0.0008
              35   +0.0000   -0.6521 V-a   -0.0000
PE: 47
    1.000 Error Factor
    0.942 Sum             0.720 Transfer          0.720 Output
       35 Weights         0.016 Error             0.077 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.6178 V-a   +0.0050
               2   +1.0000   -0.1692 V-a   +0.0047
               3   +0.0000   -0.5345 V-a   -0.0000
               4   +1.0000   +0.1032 V-a   +0.0047
               5   +0.0000   -0.8605 V-a   -0.0000
               6   +0.0000   -0.4356 V-a   +0.0000
               7   +0.0000   -0.5318 V-a   +0.0000
               8   +0.0000   +0.2473 V-a   -0.0000
               9   +0.0000   +0.1787 V-a   -0.0000
              10   +0.0000   -0.9684 V-a   -0.0000
              11   +0.0000   -0.2475 V-a   -0.0002
              12   +0.0000   -0.1765 V-a   -0.0000
              13   +0.0000   +0.1848 V-a   -0.0002
              14   +1.0000   +0.2895 V-a   +0.0046
              15   +0.0000   -0.4625 V-a   +0.0000
              16   +0.0000   -0.2546 V-a   -0.0000
              17   +0.0000   -0.2744 V-a   -0.0000
              18   +0.0000   -0.4286 V-a   -0.0002
              19   +0.0000   +0.3218 V-a   +0.0000
              20   +0.0000   +0.4246 V-a   +0.0000
              21   +0.0000   -0.3197 V-a   -0.0002
              22   +0.0000   +0.2254 V-a   -0.0000
              23   +0.0000   +0.2260 V-a   -0.0000
              24   +0.0000   +0.1101 V-a   -0.0000
              25   +0.0000   +0.4919 V-a   +0.0006
              26   +0.0000   +2.9484 V-a   -0.0002
```

```
              27   +0.0000   +0.6975 V-a   +0.0006
              28   +0.0000   -2.3426 V-a   -0.0001
              29   +1.0000   -4.0604 V-a   +0.0045
              30   +1.0000   +6.2098 V-a   +0.0051
              31   +0.0000   -3.4650 V-a   +0.0006
              32   +0.0000   -0.7247 V-a   -0.0003
              33   +0.0000   -0.9233 V-a   -0.0000
              34   +1.0000   +0.2199 V-a   +0.0044
              35   +0.0000   -0.1505 V-a   -0.0000
PE: 48
    1.000 Error Factor
   -0.179 Sum             0.455 Transfer          0.455 Output
       35 Weights        -0.014 Error            -0.056 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -6.0805 V-a   -0.0049
               2   +1.0000   -0.2127 V-a   -0.0041
               3   +0.0000   -0.4046 V-a   +0.0000
               4   +1.0000   -0.1826 V-a   -0.0041
               5   +0.0000   -0.0019 V-a   +0.0000
               6   +0.0000   -1.0755 V-a   -0.0000
               7   +0.0000   -0.5967 V-a   -0.0000
               8   +0.0000   -0.0023 V-a   -0.0000
               9   +0.0000   +0.1182 V-a   +0.0000
              10   +0.0000   -0.7432 V-a   +0.0000
              11   +0.0000   -0.1520 V-a   +0.0003
              12   +0.0000   -0.4662 V-a   +0.0000
              13   +0.0000   +0.3425 V-a   +0.0001
              14   +1.0000   -0.0272 V-a   -0.0040
              15   +0.0000   -0.5301 V-a   -0.0000
              16   +0.0000   -0.4381 V-a   +0.0000
              17   +0.0000   -0.5901 V-a   +0.0000
              18   +0.0000   -0.1826 V-a   +0.0001
              19   +0.0000   +0.3339 V-a   -0.0000
              20   +0.0000   -0.1601 V-a   -0.0000
              21   +0.0000   -0.6018 V-a   +0.0001
              22   +0.0000   -0.2413 V-a   +0.0000
              23   +0.0000   -0.6407 V-a   -0.0001
              24   +0.0000   -0.4183 V-a   +0.0000
              25   +0.0000   -0.1209 V-a   -0.0010
              26   +0.0000   -6.7289 V-a   +0.0001
              27   +0.0000   -3.4123 V-a   -0.0008
              28   +0.0000   -1.2914 V-a   +0.0000
              29   +1.0000   -2.6272 V-a   -0.0042
              30   +1.0000   +7.5111 V-a   -0.0049
              31   +0.0000   +4.5950 V-a   -0.0011
              32   +0.0000   +2.5967 V-a   +0.0003
              33   +0.0000   +0.8245 V-a   +0.0002
              34   +1.0000   +1.4103 V-a   -0.0038
              35   +0.0000   +0.9014 V-a   +0.0000
PE: 49
    1.000 Error Factor
   -3.564 Sum             0.028 Transfer          0.028 Output
       35 Weights        -0.002 Error            -0.086 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -8.6765 V-a   -0.0007
               2   +1.0000   -0.1585 V-a   -0.0007
               3   +0.0000   +0.4613 V-a   -0.0000
               4   +1.0000   +0.5633 V-a   -0.0007
               5   +0.0000   +0.0456 V-a   +0.0000
               6   +0.0000   +0.6482 V-a   -0.0000
               7   +0.0000   -0.3902 V-a   -0.0000
               8   +0.0000   +0.0157 V-a   +0.0000
               9   +0.0000   -3.1644 V-a   -0.0000
              10   +0.0000   -0.1950 V-a   +0.0000
              11   +0.0000   -0.3505 V-a   +0.0000
              12   +0.0000   -0.0635 V-a   -0.0000
              13   +0.0000   -0.3758 V-a   +0.0000
              14   +1.0000   +0.1082 V-a   -0.0007
              15   +0.0000   +0.2501 V-a   -0.0000
              16   +0.0000   -0.2370 V-a   -0.0000
              17   +0.0000   -0.7561 V-a   +0.0000
              18   +0.0000   +0.0827 V-a   +0.0000
              19   +0.0000   -0.7240 V-a   -0.0000
              20   +0.0000   -0.9342 V-a   -0.0000
              21   +0.0000   +0.0120 V-a   +0.0000
              22   +0.0000   -1.2016 V-a   +0.0000
              23   +0.0000   -1.2429 V-a   +0.0000
              24   +0.0000   -1.5571 V-a   +0.0000
              25   +0.0000   +0.0326 V-a   -0.0000
              26   +0.0000   -2.7970 V-a   +0.0000
              27   +0.0000   -1.7763 V-a   +0.0000
              28   +0.0000   +3.1806 V-a   +0.0000
              29   +1.0000   +3.7882 V-a   -0.0007
              30   +1.0000   -0.5421 V-a   -0.0007
              31   +0.0000   +2.9983 V-a   -0.0000
              32   +0.0000   +1.1110 V-a   +0.0000
              33   +0.0000   +0.5284 V-a   -0.0000
              34   +1.0000   +1.3489 V-a   -0.0007
              35   +0.0000   +0.2623 V-a   -0.0000
PE: 50
    1.000 Error Factor
   -4.072 Sum             0.017 Transfer          0.017 Output
       35 Weights        -0.001 Error            -0.031 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.1090 V-a   -0.0000
               2   +1.0000   -0.2533 V-a   -0.0003
               3   +0.0000   -0.3205 V-a   -0.0000
               4   +1.0000   +0.0774 V-a   -0.0002
               5   +0.0000   -0.4883 V-a   -0.0000
               6   +0.0000   -0.1844 V-a   +0.0000
               7   +0.0000   +0.1031 V-a   -0.0000
               8   +0.0000   -0.4179 V-a   -0.0000
               9   +0.0000   +0.0204 V-a   +0.0000
              10   +0.0000   +0.0807 V-a   -0.0001
              11   +0.0000   -0.4006 V-a   -0.0001
              12   +0.0000   -0.2404 V-a   +0.0000
              13   +0.0000   +0.1718 V-a   -0.0000
              14   +1.0000   -0.0485 V-a   -0.0002
              15   +0.0000   -0.3827 V-a   +0.0000
              16   +0.0000   -0.1550 V-a   -0.0001
              17   +0.0000   +0.0488 V-a   +0.0000
              18   +0.0000   +0.0147 V-a   -0.0000
              19   +0.0000   +0.2013 V-a   +0.0000
              20   +0.0000   -0.1459 V-a   +0.0000
              21   +0.0000   -0.2613 V-a   -0.0000
              22   +0.0000   -0.1941 V-a   -0.0001
```

```
                23   +0.0000   -0.1694 V-a   -0.0002
                24   +0.0000   -0.2431 V-a   +0.0000
                25   +0.0000   -0.0687 V-a   +0.0005
                26   +0.0000   -0.5303 V-a   -0.0000
                27   +0.0000   +0.1412 V-a   +0.0001
                28   +0.0000   -0.0011 V-a   -0.0002
                29   +1.0000   -1.7510 V-a   -0.0002
                30   +1.0000   -0.1510 V-a   +0.0002
                31   +0.0000   -0.3165 V-a   +0.0005
                32   +0.0000   +0.5410 V-a   -0.0004
                33   +0.0000   -0.9427 V-a   -0.0001
                34   +1.0000   -0.8383 V-a   -0.0006
                35   +0.0000   -0.5197 V-a   +0.0000
PE: 51
    1.000 Error Factor
   -3.885 Sum              0.020 Transfer          0.020 Output
       35 Weights         -0.003 Error            -0.151 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.7762 V-a   -0.0009
               2   +1.0000   +0.0330 V-a   -0.0009
               3   +0.0000   +0.0619 V-a   -0.0000
               4   +1.0000   -0.0717 V-a   -0.0009
               5   +0.0000   -0.3034 V-a   +0.0000
               6   +0.0000   +0.1581 V-a   -0.0000
               7   +0.0000   -0.0646 V-a   -0.0000
               8   +0.0000   -0.1795 V-a   +0.0000
               9   +0.0000   -0.1394 V-a   +0.0000
              10   +0.0000   -0.0129 V-a   +0.0000
              11   +0.0000   -0.5905 V-a   -0.0002
              12   +0.0000   -0.2939 V-a   +0.0000
              13   +0.0000   +0.0933 V-a   -0.0001
              14   +1.0000   -0.3561 V-a   -0.0009
              15   +0.0000   +0.1069 V-a   -0.0000
              16   +0.0000   -0.2231 V-a   +0.0000
              17   +0.0000   +0.3412 V-a   +0.0000
              18   +0.0000   -0.2258 V-a   -0.0001
              19   +0.0000   -0.0772 V-a   +0.0000
              20   +0.0000   -0.1536 V-a   +0.0000
              21   +0.0000   +0.2068 V-a   -0.0001
              22   +0.0000   -0.5893 V-a   -0.0000
              23   +0.0000   +0.0138 V-a   +0.0001
              24   +0.0000   -0.0728 V-a   +0.0000
              25   +0.0000   +0.0508 V-a   +0.0001
              26   +0.0000   +0.2825 V-a   -0.0001
              27   +0.0000   -0.5874 V-a   +0.0001
              28   +0.0000   -4.5877 V-a   +0.0000
              29   +1.0000   +1.3867 V-a   -0.0010
              30   +1.0000   +1.9881 V-a   -0.0009
              31   +0.0000   -0.1186 V-a   +0.0001
              32   +0.0000   +0.5269 V-a   -0.0001
              33   +0.0000   +0.2925 V-a   -0.0000
              34   +1.0000   -5.0954 V-a   -0.0010
              35   +0.0000   -2.8000 V-a   -0.0000
PE: 52
    1.000 Error Factor
   -8.739 Sum              0.000 Transfer          0.000 Output
       35 Weights         -0.000 Error            -0.108 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.5036 V-a   -0.0000
               2   +1.0000   -0.6464 V-a   -0.0000
               3   +0.0000   -0.1058 V-a   -0.0000
               4   +1.0000   -0.0580 V-a   -0.0000
               5   +0.0000   -0.6139 V-a   +0.0000
               6   +0.0000   -0.2249 V-a   -0.0000
               7   +0.0000   +0.3520 V-a   -0.0000
               8   +0.0000   -0.3156 V-a   -0.0000
               9   +0.0000   -0.3941 V-a   +0.0000
              10   +0.0000   +0.5395 V-a   -0.0000
              11   +0.0000   -0.8103 V-a   -0.0000
              12   +0.0000   -0.3472 V-a   +0.0000
              13   +0.0000   +0.2788 V-a   -0.0000
              14   +1.0000   -0.6294 V-a   -0.0000
              15   +0.0000   -0.5697 V-a   -0.0000
              16   +0.0000   -0.0701 V-a   -0.0000
              17   +0.0000   -0.1172 V-a   +0.0000
              18   +0.0000   -0.3567 V-a   -0.0000
              19   +0.0000   +0.0073 V-a   +0.0000
              20   +0.0000   +0.6543 V-a   +0.0000
              21   +0.0000   -0.5030 V-a   -0.0000
              22   +0.0000   -0.1240 V-a   -0.0000
              23   +0.0000   +0.3436 V-a   +0.0000
              24   +0.0000   -0.1393 V-a   +0.0000
              25   +0.0000   -0.5644 V-a   -0.0000
              26   +0.0000   -0.2516 V-a   -0.0000
              27   +0.0000   -0.0953 V-a   -0.0000
              28   +0.0000   +1.4451 V-a   +0.0000
              29   +1.0000   -2.9063 V-a   -0.0000
              30   +1.0000   -1.1038 V-a   -0.0000
              31   +0.0000   -0.2006 V-a   -0.0000
              32   +0.0000   +0.3153 V-a   -0.0000
              33   +0.0000   -1.6051 V-a   -0.0000
              34   +1.0000   -1.8914 V-a   -0.0000
              35   +0.0000   -1.4063 V-a   +0.0000
PE: 53
    1.000 Error Factor
   -5.475 Sum              0.004 Transfer          0.004 Output
       35 Weights         -0.001 Error            -0.204 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -2.1868 V-a   -0.0012
               2   +1.0000   -0.9425 V-a   -0.0004
               3   +0.0000   -0.2433 V-a   +0.0000
               4   +1.0000   -0.5744 V-a   -0.0003
               5   +0.0000   +0.0599 V-a   +0.0000
               6   +0.0000   +0.4902 V-a   +0.0000
               7   +0.0000   -0.8936 V-a   -0.0000
               8   +0.0000   -0.1579 V-a   +0.0000
               9   +0.0000   +0.1216 V-a   -0.0000
              10   +0.0000   -0.3145 V-a   -0.0002
              11   +0.0000   -0.3255 V-a   -0.0008
              12   +0.0000   -0.4074 V-a   +0.0000
              13   +0.0000   -0.0354 V-a   -0.0004
              14   +1.0000   -0.3258 V-a   -0.0007
              15   +0.0000   -0.4348 V-a   +0.0000
              16   +0.0000   -0.5761 V-a   -0.0002
              17   +0.0000   +0.0837 V-a   +0.0000
              18   +0.0000   +0.5424 V-a   -0.0004
```

```
                19   +0.0000   +0.2101 V-a   +0.0000
                20   +0.0000   +0.1278 V-a   +0.0000
                21   +0.0000   -0.0797 V-a   -0.0004
                22   +0.0000   -0.3675 V-a   -0.0002
                23   +0.0000   -0.2855 V-a   -0.0000
                24   +0.0000   -0.5362 V-a   +0.0000
                25   +0.0000   +0.4077 V-a   +0.0000
                26   +0.0000   +2.0706 V-a   -0.0004
                27   +0.0000   +1.5050 V-a   -0.0005
                28   +0.0000   -4.6577 V-a   -0.0000
                29   +1.0000   -6.3844 V-a   -0.0006
                30   +1.0000   +4.0154 V-a   -0.0010
                31   +0.0000   -1.1497 V-a   +0.0000
                32   +0.0000   -0.8846 V-a   -0.0010
                33   +0.0000   -0.9003 V-a   -0.0004
                34   +1.0000   +0.9181 V-a   -0.0012
                35   +0.0000   +0.0065 V-a   +0.0000
PE: 54
    1.000 Error Factor
   -4.847 Sum              0.008 Transfer          0.008 Output
       35 Weights          0.001 Error             0.097 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -0.5440 V-a   +0.0003
               2   +1.0000   -1.4636 V-a   +0.0002
               3   +0.0000   +0.5812 V-a   +0.0000
               4   +1.0000   -0.3738 V-a   +0.0002
               5   +0.0000   +0.4073 V-a   -0.0000
               6   +0.0000   +0.0746 V-a   +0.0000
               7   +0.0000   -0.1244 V-a   +0.0000
               8   +0.0000   -0.0180 V-a   -0.0000
               9   +0.0000   +0.4395 V-a   -0.0000
              10   +0.0000   -0.0234 V-a   -0.0000
              11   +0.0000   +0.7143 V-a   -0.0000
              12   +0.0000   -0.0155 V-a   -0.0000
              13   +0.0000   +0.1744 V-a   +0.0000
              14   +1.0000   -0.8728 V-a   +0.0002
              15   +0.0000   -0.8233 V-a   +0.0000
              16   +0.0000   -1.5328 V-a   -0.0000
              17   +0.0000   -0.2255 V-a   -0.0000
              18   +0.0000   -0.9907 V-a   +0.0000
              19   +0.0000   +0.0113 V-a   -0.0000
              20   +0.0000   +0.7109 V-a   -0.0000
              21   +0.0000   +0.1548 V-a   +0.0000
              22   +0.0000   -0.2293 V-a   +0.0000
              23   +0.0000   -0.1006 V-a   -0.0000
              24   +0.0000   -0.7222 V-a   -0.0000
              25   +0.0000   -0.6452 V-a   +0.0001
              26   +0.0000   -1.3080 V-a   -0.0000
              27   +0.0000   -1.7833 V-a   +0.0001
              28   +0.0000   -1.1290 V-a   -0.0000
              29   +1.0000   -0.0818 V-a   +0.0002
              30   +1.0000   -1.2013 V-a   +0.0004
              31   +0.0000   +0.6199 V-a   +0.0001
              32   +0.0000   -0.5871 V-a   -0.0000
              33   +0.0000   -0.7142 V-a   -0.0000
              34   +1.0000   -0.3077 V-a   +0.0002
              35   +0.0000   -0.2326 V-a   -0.0000
PE: 55
    1.000 Error Factor
   -2.246 Sum              0.096 Transfer          0.096 Output
       35 Weights          0.009 Error             0.099 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -1.2727 V-a   +0.0034
               2   +1.0000   -0.1312 V-a   +0.0029
               3   +0.0000   +0.5266 V-a   +0.0000
               4   +1.0000   +0.0962 V-a   +0.0026
               5   +0.0000   -0.0835 V-a   -0.0000
               6   +0.0000   -0.3611 V-a   -0.0000
               7   +0.0000   +0.0726 V-a   +0.0000
               8   +0.0000   -0.2771 V-a   -0.0000
               9   +0.0000   -0.6525 V-a   -0.0000
              10   +0.0000   -0.0782 V-a   +0.0002
              11   +0.0000   -0.1419 V-a   +0.0009
              12   +0.0000   -0.9555 V-a   -0.0000
              13   +0.0000   +0.3499 V-a   +0.0007
              14   +1.0000   -0.4694 V-a   +0.0029
              15   +0.0000   +0.2274 V-a   -0.0000
              16   +0.0000   -0.8660 V-a   +0.0002
              17   +0.0000   +0.0967 V-a   -0.0000
              18   +0.0000   -0.3539 V-a   +0.0007
              19   +0.0000   +0.4781 V-a   -0.0000
              20   +0.0000   +0.1408 V-a   -0.0001
              21   +0.0000   +0.5348 V-a   +0.0007
              22   +0.0000   -0.2833 V-a   +0.0003
              23   +0.0000   +0.1052 V-a   +0.0001
              24   +0.0000   -0.9549 V-a   -0.0000
              25   +0.0000   +0.6654 V-a   -0.0006
              26   +0.0000   +1.3874 V-a   +0.0006
              27   +0.0000   +0.7244 V-a   +0.0002
              28   +0.0000   -2.1221 V-a   +0.0001
              29   +1.0000   -1.7942 V-a   +0.0031
              30   +1.0000   -0.2975 V-a   +0.0030
              31   +0.0000   +0.6888 V-a   -0.0006
              32   +0.0000   -0.3599 V-a   +0.0014
              33   +0.0000   +1.9001 V-a   +0.0002
              34   +1.0000   +1.6441 V-a   +0.0040
              35   +0.0000   -3.0554 V-a   -0.0001
Layer: Out
        PEs: 2                                      Sum: Sum
    Spacing: 5       F' offset: 0.00           Transfer: Sigmoid
      Shape: Square                              Output: Direct
      Scale: 1.00     Low Limit: -9999.00     Error Func: standard
     Offset: 0.00    High Limit: 9999.00          Learn: Delta-Rule
   Init Low: -0.100   Init High: 0.100      L/R Schedule: (Network)
   Winner 1: None                             Winner 2: None
   PE: 56
       1.000 Error Factor
      -1.077 Sum              0.254 Transfer          0.254 Output
          21 Weights         -0.048 Error            -0.254 Current Error
           Input PE  Input Value Weight Type Delta Weight
               Bias   +1.0000   -0.3322 V-a   -0.0162
                 36   +0.0262   +0.5511 V-a   -0.0003
                 37   +0.0109   +0.5045 V-a   -0.0002
                 38   +0.0007   -1.0210 V-a   -0.0002
                 39   +0.0000   -1.1201 V-a   -0.0003
                 40   +0.0261   +2.8605 V-a   -0.0009
```

```
                41   +0.0003   +3.1951 V-a   -0.0000
                42   +0.0002   -2.1260 V-a   +0.0000
                43   +0.0541   +0.0089 V-a   -0.0008
                44   +0.0000   -4.3171 V-a   -0.0001
                45   +0.0000   +3.0464 V-a   +0.0006
                46   +0.0842   -0.6524 V-a   -0.0014
                47   +0.7195   -1.0171 V-a   -0.0120
                48   +0.4554   +0.0682 V-a   -0.0072
                49   +0.0276   +0.5193 V-a   -0.0004
                50   +0.0167   +0.8864 V-a   -0.0004
                51   +0.0201   +3.1093 V-a   -0.0002
                52   +0.0002   +2.1596 V-a   -0.0000
                53   +0.0042   +4.2184 V-a   -0.0008
                54   +0.0078   -1.6291 V-a   -0.0001
                55   +0.0957   -2.1958 V-a   -0.0025
PE: 57
    1.000 Error Factor
   -0.306 Sum              0.424 Transfer          0.424 Output
       21 Weights          0.019 Error             0.076 Current Error
        Input PE  Input Value Weight Type Delta Weight
            Bias   +1.0000   -0.2125 V-a   +0.0058
              36   +0.0262   +1.3006 V-a   +0.0020
              37   +0.0109   +3.5974 V-a   +0.0000
              38   +0.0007   +3.1251 V-a   -0.0001
              39   +0.0000   -4.9264 V-a   -0.0000
              40   +0.0261   +0.0053 V-a   +0.0021
              41   +0.0003   -0.0650 V-a   -0.0001
              42   +0.0002   -0.5404 V-a   +0.0029
              43   +0.0541   +1.0160 V-a   +0.0017
              44   +0.0000   +0.0925 V-a   -0.0015
              45   +0.0000   +0.1859 V-a   +0.0000
              46   +0.0842   +0.7117 V-a   +0.0009
              47   +0.7195   +1.5514 V-a   +0.0057
              48   +0.4554   -2.8014 V-a   +0.0062
              49   +0.0276   -3.3067 V-a   +0.0001
              50   +0.0167   +0.6253 V-a   +0.0005
              51   +0.0201   -0.0789 V-a   +0.0021
              52   +0.0002   -0.2383 V-a   -0.0002
              53   +0.0042   -0.0584 V-a   +0.0035
              54   +0.0078   +0.9967 V-a   +0.0001
              55   +0.0957   -0.3740 V-a   +0.0015
```

Appendix C | Listing of the Code Used to Implement the Interface Program for the Hybrid Solution in a Multi-Vehicle case

This appendix contains a listing of the code used to implement the interface program to the ANN in a multi-vehicle case. The code was written and implemented using the Microsoft C package version 6.0 for MS-DOS (Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA).

The main phases of this program are described in Chapter 8, which basically provide the possibility to test the different strategies based on an ANN which were proposed for the multi-vehicle case.

The listing includes the main program (section 1.1) developed according to the requirements defined by the ANN software package (NeuralWorks Professional II [NeuralWare,Inc.,Pittsburgh, USA]) to implement a user defined interface to the ANN. A number of functions and variables were defined and implemented in a separate file and are also listed in a subsequent section, section 1.2.

1.1 Code Listing of Main Program

```
/*******      NN_ALLR6.c  27/08/96  *********/


#include "userutl.h"
#include "fuh_allr.c"
#ifndef SUN
#ifndef DLC
#include <stdlib.h>
#endif
#endif


/*************************************************************************/
/*     funcao UsrIO ()        para teste da interface com NWORKS         */
/*                                                                       */
/*                                                                       */
/*                                                                       */
/*            User I/O "C" Interface :                                   */
/*                                                                       */
/*                  - define input and output vectors                    */
/*                  - read ANN output                                    */
/*                  - interpret and make decisions                       */
/*                  - update ANN input                                   */
/*                                                                       */
/*        (funcionamento em continuo com geracao automatica e            */
/*            aleatoria dos NOS origem e destino para cada veiculo)  */
/*                                                                       */
/*************************************************************************/


/*************************************************************************/
/*                                                                       */
/*   NAO USAR OS NOMES DE VARIAVEIS: record, x_temp, y_temp  */
/*                                                                       */
/*************************************************************************/

static int flag_inicio = (0); /* permite identificar a primeira (inicio)*/
                              /*       especificacao de dados           */
static char flag_veiculo = ('N');

static int InitFlag = (0);

static long int Num_Iter = (0);

static long int tpo_abs = (0);

static int Semente =(1);

static int Lock_step =(0);/*referencia para deadlocks:max.percurso ate dest.*/
static int M_Array_Dim = (15); /* dimensao definida para os arrays melNO1/NO2/T*/

static long int DeadV_Cont = (0);
static long int flag_DCont [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int NotSolved  [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int NotSolv = (0);

static long int Solved [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int Solv_Tot = (0);
static long int NotGood [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int NotG_Tot = (0);

static int flag_Vcont [11] = (0,0,0,0,0,0,0,0,0,0,0);
static int flag_NOS0 [11] = (0,0,0,0,0,0,0,0,0,0,0);
static int flag_esp [11] = (0,0,0,0,0,0,0,0,0,0,0);

static int Lim_esp = (0);

static int No_lixo = (0);

static int MAQ_NOS [16]= (1,2,4,5,6,7,8,10,12,14,15,16,17,18,20,21);

static int ALEATOR =(0);
static float MAX_F01 = 32767.00000;


int UsrIO()                     /* handle NWORKS requests */
{
    char *sp,*sp1, Dir_temp, flag_lixo;
    int lixoNV, xforM1, M_wx, M_wy, M_wx1, no1_main, jfM1, zfM1, no_t1M,lix_contF;
int no_t2M, tr_usa_M, ind_k1, ind_k2, ind_k3, ind_k4, ind_k5, ind_k6;
int movNO2M0, lixoM1, lix_LM1, zero_lix, ifM1, ifM2, ifM3,f_R1,f_R2,tpo_lixo;

double  saida_x, saida_y;
char  buf [100];

FILE *fich_001; /*** FILE pointer a ser usado ficheiro de resultados**/
FILE *fich_002; /*** FILE pointer a ser usado ficheiro de resultados**/
FILE *fich_003; /*** FILE pointer a ser usado ficheiro de resultados**/

if ( InitFlag == 0 ) {

      /**** open any files which may be required at this point ****/
      fich_001 = fopen ("filetes1.nnr", "a");
      fich_002 = fopen ("filete01.nnr", "a");
      fich_003 = fopen ("filete03.nnr", "a");
      if ((fich_001 == NULL)||(fich_002 == NULL)||(fich_003 == NULL)){
          PutStr (" nao consegui abrir ficheiros ");
      }

      /* definir aqui o numero de veiculos que queremos considerar */
      PutStr ("Num_VeiT (min:1, max:10)= \n");
```

```
sp = GetStr ();
Num_VeiT = atoi(sp);
sprintf (buf,"%d", Num_VeiT);
PutStr (buf);
sp = GetStr ();

/* definir aqui o numero maximo de iteracoes da rede neuronal */
PutStr ("Num_Iter da rede neuronal ??? = ");
sp = GetStr ();
Num_Iter = atol(sp);
sprintf (buf,"%\ld ",Num_Iter);
PutStr (buf);
sp = GetStr ();

/* escolher a seed para geracao dos numeros pseudo-aleatorios */
PutStr ("Seed ???? = ");
sp = GetStr ();
Semente = atoi (sp);
sprintf (buf,"%\d ",Semente);
PutStr (buf);
sp = GetStr ();
srand (Semente);

/* escolher o numero maximo para referencia de possiveis deadlocks */
PutStr ("Mov/Wait Max. to Dealock [10....50]??? = ");
sp = GetStr ();
Lock_step  = atoi (sp);
sprintf (buf,"%\d ",Lock_step);
PutStr (buf);
sp = GetStr ();

/* definir o tempo de espera maximo de um veiculo num NO sem
fazer reset dos tramos usados */
PutStr ("Reset dos tramos apos Tempo espera [1...50] ??? = ");
sp = GetStr ();
Lim_esp   = atoi (sp);
sprintf (buf,"%\d ",Lim_esp);
PutStr (buf);
sp = GetStr ();

/* escolher a estrategia de look-ahead pretendida */
PutStr ("ESTRATEgia look_ahead: ALL(0), ONE(1), NONE (9)");
sp = GetStr ();
ESTRATE = atoi (sp);
sprintf (buf,"%\d ", ESTRATE);
PutStr (buf);
sp = GetStr ();

/* escolher a saida da rede como aleatoria */
PutStr ("ANN output = random ? [Yes=1, No=0]");
sp = GetStr ();
ALEATOR = atoi (sp);
sprintf (buf,"%\d ", ALEATOR);
PutStr (buf);
sp = GetStr ();

      fprintf (fich_001,"\n Num.Vehicles:%d\t Num. Iteracoes Rede NN:%\ld",Num_VeiT,Num_Iter);
      fprintf (fich_001,"\n Seed:%d\n Max. Iter. to Dealock:%d\n",Semente,Lock_step);
      fprintf (fich_001,"\n Reset dos tramos usados apos: %d\n",Lim_esp);
      fprintf (fich_001,"\n ESTRATEgia= %d\n",ESTRATE);

      fprintf (fich_002,"\n Num.Vehicles:%d\t Num. Iteracoes Rede NN:%\ld",Num_VeiT,Num_Iter);
      fprintf (fich_002,"\n Seed:%d\n Max. Iter. to Dealock:%d\n",Semente,Lock_step);
      fprintf (fich_002,"\n Reset dos tramos usados apos: %d\n",Lim_esp);
      fprintf (fich_002,"\n ESTRATEgia= %d\n",ESTRATE);

      fprintf (fich_003,"\n Num.Vehicles:%d\t Num. Iteracoes Rede NN:%\ld",Num_VeiT,Num_Iter);
      fprintf (fich_003,"\n Seed:%d\n Max. Iter. to Dealock:%d\n",Semente,Lock_step);
      fprintf (fich_003,"\n Reset dos tramos usados apos: %d\n",Lim_esp);
      fprintf (fich_003,"\n ESTRATEgia= %d\n",ESTRATE);

      InitFlag = 1;
}

IORTNCDE = 0;

if ((Num_VeiT < 1) || (Num_VeiT > Num_MaxV)){
      PutStr (" Error in Num_VeiT ");
      sprintf (buf,"Num_VeiT = %d \n",Num_VeiT);
      PutStr (buf);
      sp = GetStr();
      IORTNCDE = -1;
}

switch ( IOREQCDE) {
      case RQ_ATTENTION:
    break;
case RQ_REWIND:
    break;
case RQ_LSTART:
    break;
case RQ_LEARNIN:
    break;
case RQ_LEARNOUT:
    break;
case RQ_LEARNRSLT:
    break;
case RQ_LEND:
    break;
case RQ_RSTART:
    break;
case RQ_READ:

    /*************************************************************/
    /*    reset das variaveis de usadas no program               */
    /*                                                           */
    /*************************************************************/

    sp = "N";
```

```
if (flag_inicio == 0){

      tpo_lixo = 0;
      Veic_cont = 0;
      Veic_id = 0;
      no1_main = 0;
      tpo = 0;
      tpo_abs = 0;
      ftramo_zero ();      /*  reset das matrizes que representam */
      mat_RLUD();          /* o estado dos tramos (MATTH/VE e     */
      mtra_RLUD();         /*    MATR_R/L/U/D, MATR/L/U/D                */

      fnos_tliv();         /* inicializacao das matrizes que contem */
      ftra_liv();
      fnos_tblo();         /* num de saidas possiveis de cada no   */
      fsaip_ini();

      f_N_ocup ();   /* inicializacao de NOS_OCUP [][]  */
      fmest_no_0 (); /* reset da matriz estado dos nos (MEST_NO[][]) */
      fmat_noV0 ();        /* reset da matriz veiculos em nos (MVEI_NO[][])*/

      for (M_wx = 0; M_wx < M_Array_Dim; M_wx++){
          for (M_wy = 0; M_wy < Num_MaxV; M_wy++){
                vei_Pr [M_wx][M_wy] = 0;
                vei_Pr_id [M_wx][M_wy] = 0;
                melNO1[M_wx][M_wy] = 0;
                melNO2[M_wx][M_wy] = 0;
                melT [M_wx][M_wy] = 0;
                tipo_solV [M_wx][M_wy] = 0;
          }
          array_tra[M_wx] = 0;
      }
      flag_vei = 0;
      flag_vpri = 0;
      tipo_sol = 0;
      for (M_wx = 0; M_wx < Num_MaxV; M_wx ++){
          staint_noV [M_wx] = 0;
          endint_noV [M_wx] = 0;
          temp_seq [M_wx] = 0;
          int_ordA [M_wx][0] = 0;
          int_ordA [M_wx][1] = 0;
          flag_esp [M_wx] = 0;
      }

      for (M_wx = 0; M_wx < 5; M_wx ++){
          registo [M_wx] = 0;
          flag_est [M_wx] = 0;
          curr_nod [M_wx] = 0;
          start_nod[M_wx] = 0;
          end_nod  [M_wx] = 0;
      }

      /*********************************************************************/

      /* nos origem para os AGVs */
      M_wx = 0;
      for (ifM1 = 0; ifM1 < Num_VeiT; ifM1 ++){
          Salto_1:
          M_wx = MAQ_NOS [f_randoes (1,16)-1];

          ifM3 = 0;
          for (ifM2 = 0; ifM2 < Num_MaxV; ifM2 ++){
                if ((M_wx == staint_noV[ifM2])&&(ifM1 != ifM2)){
                    ifM3 = 1;
                }
          }
          if (ifM3 == 1){
                goto Salto_1;
          }

          staint_noV [ifM1] = M_wx;
      }

      /* nos destino para os AGVs */
      M_wx = 0;
      for (ifM1 = 0; ifM1 < Num_VeiT; ifM1 ++){
          Salto_2:
          M_wx = MAQ_NOS [f_randoes (1,16)-1];

          ifM3 = 0;
          for (ifM2 = 0; ifM2 < Num_MaxV; ifM2 ++){
                if ((M_wx==endint_noV[ifM2])||((M_wx == staint_noV[ifM2])&&(ifM1 == ifM2))){
                    ifM3 = 1;
                }
          }
          if (ifM3 == 1){
                goto Salto_2;
          }

          endint_noV [ifM1] = M_wx;
      }

      /**       actualizar posicoes iniciais de cada veiculo ****/

      for (M_wx = 0; M_wx < Num_VeiT; M_wx ++){
          melNO1 [0][M_wx] = staint_noV [M_wx];
      }

      /** actualizar o estado dos nos correspondentes
      as posicoes de cada veiculo ***/

      for (M_wx = 0; M_wx < Num_VeiT; M_wx ++){
          fmest_no_N (melNO1 [0][M_wx], 1);
          fmat_noV (melNO1 [0][M_wx], M_wx+1);
      }
      /* atribuir prioridades iniciais a cada veiculo   */
      for (M_wx = 1; M_wx <= Num_VeiT; M_wx ++){
          vei_Pr [0][M_wx-1] = M_wx;
```

```
    vei_Pr_id [0][M_wx-1] = M_wx;
}


/**       actualizar os valores iniciais de numero de no inicio e destino
a ser usado no vector de entrada de acordo com o veiculo escolhido***/

zero_lix = f_dec2_bin(staint_noV[vei_Pr[0][0]-1]);
if (zero_lix < 0){
    PutStr ("error in zero_lix");
}
for (M_wx = 0; M_wx < 5; M_wx++){
    start_nod [M_wx] = DEC2BIN [M_wx];
}

zero_lix = f_dec2_bin (endint_noV[vei_Pr[0][0]-1]);
if (zero_lix < 0){
    PutStr ("error in zero_lix");
}
for (M_wx = 0; M_wx < 5; M_wx++){
    end_nod [M_wx] = DEC2BIN [M_wx];
}

for (M_wx = 0; M_wx < Num_VeiT; M_wx++){
    curint_noV[M_wx] = staint_noV[vei_Pr[0][M_wx]-1];
}
        flag_vei = vei_Pr [0][0];
        flag_vpri= vei_Pr_id[0][vei_Pr[0][0] -1];
        Veic_id = flag_vei;

        /*****************************************************************/
        /*        actualizacao dos nos curr_nod e next_nod               */
        /*****************************************************************/
        for (M_wx = 0; M_wx<5; M_wx++){
            curr_nod [M_wx] = start_nod [M_wx];
        }
        sprintf(buf,"%ld \n",cont_casos);
        PutStr(buf);
        flag_inicio = 1;
    }
    if ((tpo_abs * Num_VeiT) > Num_Iter){
        fprintf (fich_001,"\nFIM DA SIMULACAO:");
        fprintf (fich_001,"%ld\t %ld\n", tpo_abs, cont_casos);
        for (M_wx = 0; M_wx < Num_MaxV; M_wx++){
            fprintf (fich_001,"Veic[%d]=%ld\t =%ld\n",M_wx+1,flag_DCont[M_wx],NotSolved[M_wx]);
            NotSolv = NotSolv + NotSolved[M_wx];
            DeadV_Cont = DeadV_Cont + flag_DCont [M_wx];
            Solv_Tot = Solv_Tot + Solved [M_wx];
            NotG_Tot = NotG_Tot + NotGood[M_wx];
        }
        fprintf (fich_001,"Totals = %ld\t %ld\t %ld\t %ld\n",Solv_Tot,DeadV_Cont,NotG_Tot,NotSolv);

        fprintf (fich_002,"\nFIM DA SIMULACAO:");
        fprintf (fich_002,"%ld\t %ld\n", tpo_abs, cont_casos);
        for (M_wx = 0; M_wx < Num_MaxV; M_wx++){
            fprintf (fich_002,"Veic[%d]=%ld\t =%ld\n",M_wx+1,flag_DCont[M_wx],NotSolved[M_wx]);
        }
        fprintf (fich_002,"Totals = %ld\t %ld\t %ld\t %ld\n",Solv_Tot,DeadV_Cont,NotG_Tot,NotSolv);

        fprintf (fich_003,"\nFIM DA SIMULACAO:");
        fprintf (fich_003,"%ld\t %ld\n", tpo_abs, cont_casos);
        for (M_wx = 0; M_wx < Num_MaxV; M_wx++){
            fprintf (fich_003,"Veic[%d]=%ld\t =%ld\n",M_wx+1,flag_DCont[M_wx],NotSolved[M_wx]);
        }
        fprintf (fich_003,"Totals = %ld\t %ld\t %ld\t %ld\n",Solv_Tot,DeadV_Cont,NotG_Tot,NotSolv);

        IORTNCDE = -1;
        goto Fim_test;
    }
    /*****************************************************************/
    /*****          fim da entrada dos dados iniciais                *******/
    /**** e comeco da actualizacao da entrada da rede (input_vec)**/
    /*****************************************************************/
    for (M_wx = 0; M_wx<5; M_wx++){
        next_nod [M_wx] = end_nod [M_wx];
    }
    mtra_RLUD();
    mat_RLUD();

    fnos_tliv();
    ftra_liv();
    fnos_tblo();
    fsaip_ini();

    /**** actualizar matrizes que representam o estado dos tramos ****/
    /*as matrizes MATTHE e MATTVE ja devem estar actualizadas neste ponto**/
    /*actualizar vector cursta_trk com base nos valores contidos em MATTEH/V*/
    /*****************************************************************/

    M_wx1 = 0;
    for (M_wx = 0; M_wx<3; M_wx++){
        for (M_wy = 0; M_wy<4; M_wy++){
            M_wx1 = MATTRH [M_wx][M_wy] -1;
            if (M_wx1 >= 24){
                PutStr ( "erro na actualizaŧao de cursta_trk ");
            }
            cursta_trk [M_wx1] = MATTHE[M_wx][M_wy] ;
        }
    }
    for (M_wx = 0; M_wx<4; M_wx++){
        for (M_wy = 0; M_wy<3; M_wy++){
            M_wx1 = MATTRV [M_wx][M_wy] -1;
            if (M_wx1 >= 24){
                PutStr ( "erro na actualizaŧao de inista_trk ");
            }
            cursta_trk [M_wx1] = MATTVE[M_wx][M_wy] ;
        }
    }
    /*****************************************************************/
```

```
    /*    atribuicao dos valores especificados ao vector de entrada     */
    /*e ponteiro respectivo e actualizacao das matrizes necessarias     */
    /*********************************************************************/
    fvec_inp (cursta_trk, curr_nod, next_nod);
    for (M_wx = 0; M_wx <IOCOUNT; M_wx++){
          IODATA [ M_wx] = inp_vec [ M_wx];
    }

    Veic_cont ++;

    Fim_test:         /* fim da simulacao */

    break;

case RQ_WRSTEP:
    break;

case RQ_WRITE:

    if (ALEATOR == 0){
          out_vec [0] = IODATA [0];
          out_vec [1] = IODATA [1];
    }
    else if (ALEATOR == 1){
          out_vec [0] = (float)rand()/MAX_F01;
          out_vec [1] = (float)rand()/MAX_F01;
    }
    else{
          PutStr ("error in ALEATOR");
          IORTNCDE = -1;
    }
    saida_x = conv1 (out_vec [0]);
    saida_y = conv1 (out_vec [1]);

    no_t1M = f_Int5_2dec (2);  /* 2: curr_nod */
    no_t2M = f_Int5_2dec (3); /* 3: end_nod */

    /** identificar o ultimo tramo que ainda nao se repetiu sabendo
    que um tramo usado mais do que uma vez sera bloqueado (= 1)
    (neste caso multi-veiculos nao se bloqueiam os tramos repetidos pelo
    que se escolhe o ultimo tramo livre, que devera ser o penultimo
    tramo usado)    ***/

    tr_usa_M = 0;
    if (tpo == 0){
          tr_usa_M = 0;
    }
    else{
          M_wx1 = tpo - 1;
          while (M_wx1 >= 0){
              if ((melT[M_wx1][Veic_id-1] > 0)&&(melT[M_wx1][Veic_id-1] <= 24)){
                    M_wy = M_wx1;
                    tr_usa_M = melT[M_wy][Veic_id-1];
                    M_wx1 = -10;
              }
              else if (melT[M_wx1][Veic_id-1] == 0){
                    tr_usa_M = 0;
              }
              else {
                    PutStr ("error in tr_usa_M:");
                    sprintf (buf,"%d",tr_usa_M);
                    PutStr (buf);
                    sp = GetStr ();
                    IORTNCDE = -1;
              }
              M_wx1 = M_wx1 -1;
          }
    }


    /* escrever informacao no ficheiro de resultados */
    fprintf (fich_002,"\n %ld\t %d %d\t",tpo_abs, Veic_id, vei_Pr_id[tpo][Veic_id-1]);
    fprintf (fich_002,"%d %d\t %d\t",no_t1M,no_t2M, tr_usa_M);
    if (tpo == 0){
          fprintf (fich_002,"%d\t", melT[tpo][Veic_id-1]);
    }
    else if (tpo > 0){
          fprintf (fich_002,"%d\t", melT[tpo-1][Veic_id-1]);
    }

    fprintf (fich_002,"%.6f  %.6f\t", saida_x, saida_y);

    if (tpo_lixo == 1000){
          sprintf (buf,"t:%ld\n",tpo_abs);
          PutStr (buf);
          tpo_lixo = 0;
    }

    if ((no_t1M <1)||(no_t1M >21)||(no_t2M <1)||(no_t2M >21)){
          PutStr ("error in no_t1/2M");
          sprintf (buf,"%d\t %d\n",no_t1M, no_t2M);
          PutStr (buf);
          sp = GetStr();
          IORTNCDE = -1;
    }

    eva_strat1 (no_t1M, no_t2M, tr_usa_M, saida_x, saida_y, 0.0000);

    fprintf (fich_002,"%d\t", tipo_sol);
    fprintf (fich_002,"%d\t %d\t %d\t %d\t
%d\n",registo[0],registo[1],registo[2],registo[3],registo[4]);


    /** curr_nod == no destino ??? **/
    flag_est [0] = 0;
    if(registo[0] == f_Int5_2dec (3)){
          flag_est[0] = 1;
    }
    else{
```

```
      flag_est[0] = 0;
}
/** verificar se indica paragem (S) ******/
flag_est [1] = 0;
if (registo [1] == 0){
      flag_est[1] = 1;
}
else{
      flag_est[1] = 0;
}
/* testar se o tramo seguinte esta bloqueado, nao existe
na grelha, nao e indicado, ou esta livre **/
flag_est[3] = 0;
if (registo[3] == 0){
      flag_est[3] = 0;       /* livre */
}
else if (registo [3] == 1){
      flag_est[3] = 1;       /* bloqueado */
}
else if (registo [3] == -1 || registo [3] == -2){
      flag_est[3] = 2;       /* nao existe na grelha */
}
else if (registo [3] == -3){
      flag_est[3] = 3;      /* nao e indicado porque para */
}
else {
      flag_est[3] = -1;     /* erro na variavel registo[]*/
}

/** actualizar a posicao do veiculo nos arrays melNO1,
melNO2, melT**/
tipo_solV [tpo][Veic_id-1] = tipo_sol;
melNO1 [tpo][Veic_id-1] = registo [0];

if ((flag_est[3]==0)&&(flag_est[1]==0)&&(flag_est[0]==0)){
      melNO2 [tpo][Veic_id-1] = registo[4];
      melT   [tpo][Veic_id-1] = registo[2];
      ftramo_val (registo[2],1);
}
else if ((flag_est[0] == 1) || (flag_est[1] == 1)||(flag_est[3] == 1)){
      melNO2 [tpo][Veic_id-1] = registo[0];
      melT   [tpo][Veic_id-1] = 0;
}
else {
      PutStr (" error in flag_est[]");
      for (ifM1 = 0; ifM1 < 5; ifM1 ++){
          sprintf (buf,"%d\t",registo[ifM1]);
          PutStr (buf);
          sp = GetStr();
      }

      IORTNCDE = -1;
}

if ( ESTRATE == 9){

      melNO1 [tpo + 1][Veic_id-1] = melNO2 [tpo][Veic_id-1];

      fmest_no_N (melNO1 [tpo][Veic_id-1],0);
      fmat_noV (melNO1 [tpo][Veic_id-1],0);
      fmest_no_N (melNO1 [tpo +1][Veic_id-1],1);
      fmat_noV (melNO1 [tpo +1][Veic_id-1],Veic_id);
}

/* ultimo veiculo a usar a rede ? */
/* SIM: analisar todas as situacoes de movimento,
actualizar percursos e alternar prioridades   */

ind_k1 = 0;
ind_k2 = 0;
ind_k3 = 0;
ind_k4 = 0;
if (Veic_cont == Num_VeiT){
      /* escrever informacao no ficheiro de resultados */
      fprintf (fich_001,"\n%ld\t",tpo_abs);
      for (M_wy = 0; M_wy < Num_VeiT; M_wy ++){
          fprintf (fich_001,"%2d %2d %2d",melNO1[tpo][M_wy],melNO2[tpo][M_wy],vei_Pr_id[tpo][M_wy]);
          fprintf (fich_001," %2d\t", tipo_solV[tpo][M_wy]);
      }
      fprintf (fich_001,"\n");

      if ((ESTRATE == 0)||(ESTRATE == 1)){
          f_st_ahead (ESTRATE);
      }
      else if (ESTRATE == 9){
          /* nao chama f_st_ahead()*/
      }
      else{
          PutStr ("error in ESTRATE");
          IORTNCDE = -1;
      }

      /* reset do elemento tpo dos arrays TRAMOS dos veiculos
      que param e contabilizar o tempo de espera no NO ACTUAL */

      for (ifM1 = 0; ifM1 < Num_VeiT; ifM1++){
          if (melNO1[tpo+1][ifM1] == melNO1[tpo][ifM1]){
                melT [tpo][ifM1] = 0;
                flag_esp [ifM1] = flag_esp [ifM1] +1;
          }
          else if (melNO1[tpo+1][ifM1] != melNO1[tpo][ifM1]){
                flag_esp [ifM1] = 0;
          }

          if (flag_esp [ifM1] > Lim_esp ){
                flag_esp [ifM1] = 0;
                for (M_wy = 0; M_wy < M_Array_Dim; M_wy ++){
                    melT [M_wy][ifM1] = 0;
                }
```

```
    }
}

/* verificar se ha dois veiculos num mesmo no,*/
/* ou se melNO1[t+1] < 1 ou > 21        */
for (ifM1 = 0; ifM1 < Num_VeiT; ifM1++){
    for (jfM1 = ifM1+1; jfM1 < Num_VeiT; jfM1++){
          if ((melNO1 [tpo+1][ifM1] > 0) && (melNO1 [tpo+1][ifM1] < 22)){
              if (melNO1[tpo+1][ifM1] == melNO1[tpo+1][jfM1]){
                    PutStr ("ERROR IN melNO1 [tpo +1]");
                    sp = GetStr();
                    sprintf (buf,"cont_casos=%\d\n",cont_casos);
                    PutStr (buf);
                    sp = GetStr ();
                    sprintf (buf,"melNO1[%d][%d]= %d\n",tpo+1,ifM1,melNO1[tpo+1][ifM1]);
                    PutStr (buf);
                    sp = GetStr ();
                    sprintf (buf,"melNO1[%d][%d]= %d\n",tpo+1,jfM1,melNO1[tpo+1][jfM1]);
                    PutStr (buf);
                    sp = GetStr ();

                    IORTNCDE = -1;
              }
          }
          else {
              PutStr ("ERROR: NEGATIVE melNO1 [][]");
              sp = GetStr();
              sprintf (buf,"cont_casos=%\d\n",cont_casos);
              PutStr (buf);
              sp = GetStr ();
              sprintf (buf,"melNO1[%d][%d]= %d\n",tpo+1,ifM1,melNO1[tpo+1][ifM1]);
              PutStr (buf);
              sp = GetStr ();
              sprintf (buf,"melNO1[%d][%d]= %d\n",tpo+1,jfM1,melNO1[tpo+1][jfM1]);
              PutStr (buf);
              sp = GetStr ();

              IORTNCDE = -1;
          }
    }
}

/* escrever informacao no ficheiro de resultados */
fprintf (fich_001,"%ld\t",tpo_abs);
for (M_wy = 0; M_wy < Num_VeiT; M_wy ++){
    fprintf (fich_001,"%2d %2d %2d\t",melNO1[tpo+1][M_wy],endint_noV[M_wy], flag_esp[M_wy]);
}
/* contabilizar comprimentos dos percursos/passos de cada veiculo */
/* distinguir entre tempo de movimento e tempo de espera/paragem ?*/
/* para analise posterior ? */


/* ha veiculos que param no destino ? Gerar aleatoriamente um
novo NO destino e actualizar variaveis correspondentes */
/* ha veiculos que ultrapassam o limite de tempo para atingir
o NO destino ? */

for (ifM1 = 0; ifM1 < Num_VeiT; ifM1++){

    flag_Vcont [ifM1] = flag_Vcont [ifM1] +1;

    if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_Vcont[ifM1]<Lock_step)){

          /* chegou ao destino em tempo <= 49 */
          cont_casos ++;
          staint_noV[ifM1] = endint_noV[ifM1];
          endint_noV[ifM1] = 0;
          flag_Vcont[ifM1] = 0;
          /*reset dos arrays TRAMOS*/
          for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
              melT [M_wy][ifM1] = 0;
          }
          Solved[ifM1] = Solved[ifM1] +1;

    }
    else if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_Vcont[ifM1]==Lock_step)){

          /* chegou ao destino em tempo == 50 */
          cont_casos ++;
          staint_noV[ifM1] = endint_noV[ifM1];
          endint_noV[ifM1] = 0;
          flag_Vcont[ifM1] = 0;
          /*reset dos arrays TRAMOS*/
          for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
              melT [M_wy][ifM1] = 0;
          }
          flag_DCont[ifM1] = flag_DCont[ifM1] +1;
    }
    else if
((melNO1[tpo+1][ifM1]==flag_NOS0[ifM1])&&(flag_NOS0[ifM1]>0)&&(flag_Vcont[ifM1]<100)){

          /* chegou(passou) ao destino original antes do no destino
          temporario em tempo <= 99      */
          cont_casos ++;
          staint_noV[ifM1] = flag_NOS0 [ifM1];
          endint_noV [ifM1] = 0;
          flag_NOS0[ifM1] = 0;
          flag_Vcont [ifM1] = 0;
          /*reset dos arrays TRAMOS*/
          for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
              melT [M_wy][ifM1] = 0;
          }
          flag_DCont[ifM1] = flag_DCont[ifM1] +1;
    }
    else if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_NOS0[ifM1]>0)){

          /* chegou ao no destino intermedio em tempo <= 99 */
          endint_noV[ifM1] = flag_NOS0[ifM1];
```

```
                        flag_NOS0[ifM1] = -1;
                    }
                    else if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_NOS0[ifM1]==-
1)&&(flag_Vcont[ifM1]<100)){

                        /* chegou ao no destino original depois de ter encontrado
                        o no destino temporario em tempo < 49 */
                        cont_casos ++;
                        staint_noV[ifM1] = endint_noV [ifM1];
                        endint_noV [ifM1] = 0;
                        flag_NOS0[ifM1] = 0;
                        flag_Vcont [ifM1] = 0;
                        /*reset dos arrays TRAMOS*/
                        for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
                            melT [M_wy][ifM1] = 0;
                        }
                        flag_DCont[ifM1] = flag_DCont[ifM1] +1;
                    }
                    else if
((melNO1[tpo+1][ifM1]==flag_NOS0[ifM1])&&(flag_NOS0[ifM1]>0)&&(flag_Vcont[ifM1]==100)){

                        /* chegou ao no destino em tempo == 100
                        sem ter encontrado o no temporario */
                        cont_casos ++;
                        staint_noV[ifM1] = flag_NOS0[ifM1];
                        endint_noV [ifM1] = 0;
                        flag_NOS0[ifM1] = 0;
                        flag_Vcont [ifM1] = 0;
                        /*reset dos arrays TRAMOS*/
                        for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
                            melT [M_wy][ifM1] = 0;
                        }
                        NotGood[ifM1] = NotGood[ifM1] +1;
                    }
                    else if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_NOS0[ifM1]==-
1)&&(flag_Vcont[ifM1]==100)){

                        /* chegou ao no destino em tempo == 100
                        depois de ter encontrado o no temporario */
                        cont_casos ++;
                        staint_noV[ifM1] = endint_noV[ifM1];
                        endint_noV [ifM1] = 0;
                        flag_NOS0[ifM1] = 0;
                        flag_Vcont [ifM1] = 0;
                        /*reset dos arrays TRAMOS*/
                        for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
                            melT [M_wy][ifM1] = 0;
                        }
                        NotGood[ifM1] = NotGood[ifM1] +1;
                    }
                    else if ((melNO1[tpo+1][ifM1]==endint_noV[ifM1])&&(flag_NOS0[ifM1]==-
2)&&(flag_Vcont[ifM1]<200)){

                        /* chegou ao no destino original em tempo <= 199 */
                        cont_casos ++;
                        staint_noV[ifM1] = endint_noV [ifM1];
                        endint_noV [ifM1] = 0;
                        flag_NOS0[ifM1] = 0;
                        flag_Vcont [ifM1] = 0;
                        /*reset dos arrays TRAMOS*/
                        for (M_wy = 0; M_wy < M_Array_Dim; M_wy++){
                            melT [M_wy][ifM1] = 0;
                        }
                        NotGood[ifM1] = NotGood[ifM1] +1;
                    }

                    /* casos nao terminados: controlo do tempo gasto na procura do NO destino
                    para atribuicao e actualizacao das variaveis necessarias*/

                    if ((flag_Vcont[ifM1]>(Lock_step-1))&&(flag_NOS0[ifM1]==0)){

                        /* nao encontrou o no destino antes de 50 iteracoes */
                        flag_NOS0[ifM1] = endint_noV[ifM1];
                        endint_noV[ifM1] = 0;
                    }
                    else if ((flag_Vcont[ifM1]>99)&&(flag_NOS0[ifM1]>0)){

                        /* se nao encontrou o no dest. temp. em <= 49
                        tenta voltar de novo ao no dest. original */
                        endint_noV[ifM1] = flag_NOS0[ifM1];
                        flag_NOS0[ifM1] = -2;
                    }
                    else if ((flag_Vcont[ifM1]>99)&&(flag_NOS0[ifM1]==-1)){

                        /* ou/e se ja encontrou o no destino temp. mas ultrapassou
                        os 49 passos sem encontrar o no destino */
                        flag_NOS0[ifM1] = -2;
                    }
                    else if (flag_Vcont[ifM1]>199){

                        cont_casos ++;
                        staint_noV [ifM1] = endint_noV [ifM1];
                        endint_noV [ifM1] = 0;
                        flag_NOS0 [ifM1] = 0;
                        flag_Vcont [ifM1] = 0;
                        NotSolved [ifM1] = NotSolved [ifM1] +1;
                    }
                }

                for (ifM1 = 0; ifM1 < Num_VeiT; ifM1++){

                    if ((endint_noV[ifM1]==0)&&(flag_NOS0[ifM1]!=0)){
                        /* escolher o no +proximo do no destino em deadlock
                        nao podendo ser nenhum dos ja escolhidos ou em lista
                        de espera.  Primeiro escolhe-se um dos que esteja a
                        um tramo do no destino, se nao for possivel escolhe-se
                        aleatoriamente      */

                        ind_k5 = indexi(flag_NOS0[ifM1]);
```

```
                                ind_k6 = indexj(flag_NOS0[ifM1]);
                                No_lixo = MATNOS [ind_k5+1][ind_k6];
                                lixoNV = 0;
                                if ((ind_k5 >= 0)&&(ind_k5 < 4)){
                                    if (No_lixo>0){
                                          for (zfM1 = 0; zfM1 < Num_VeiT; zfM1++){
                                              if
((No_lixo==endint_noV[zfM1])||(No_lixo==flag_NOS0[zfM1])||(No_lixo==melNO1[tpo+1][ifM1])){
                                                  lixoNV = 1;
                                              }
                                          }
                                    }
                                    else{
                                          lixoNV = 1;
                                    }
                                    endint_noV[ifM1] = No_lixo;
                                }

                                No_lixo = MATNOS [ind_k5-1][ind_k6];
                                if (lixoNV == 1){
                                    lixoNV = 0;
                                    if ((ind_k5 >= 1)&&(ind_k5 < 5)){
                                          if (No_lixo > 0){
                                              for (zfM1 = 0; zfM1 < Num_VeiT; zfM1++){
                                                  if
((No_lixo==endint_noV[zfM1])||(No_lixo==flag_NOS0[zfM1])||(No_lixo==melNO1[tpo+1][ifM1])){
                                                      lixoNV = 1;
                                                  }
                                              }
                                          }
                                          else{
                                              lixoNV = 1;
                                          }
                                          endint_noV[ifM1] = No_lixo;
                                    }
                                }

                                No_lixo = MATNOS [ind_k5][ind_k6+1];
                                if (lixoNV == 1){
                                    lixoNV = 0;
                                    if ((ind_k6 >= 0)&&(ind_k6 < 4)){
                                          if (No_lixo > 0){
                                              for (zfM1 = 0; zfM1 < Num_VeiT; zfM1++){
                                                  if
((No_lixo==endint_noV[zfM1])||(No_lixo==flag_NOS0[zfM1])||(No_lixo==melNO1[tpo+1][ifM1])){
                                                      lixoNV = 1;
                                                  }
                                              }
                                          }
                                          else{
                                              lixoNV = 1;
                                          }
                                          endint_noV[ifM1] = No_lixo;
                                    }
                                }

                                No_lixo = MATNOS [ind_k5][ind_k6-1];
                                if (lixoNV == 1){
                                    lixoNV = 0;
                                    if ((ind_k6 > 0)&&(ind_k6 < 5)){
                                          if (No_lixo > 0){
                                              for (zfM1 = 0; zfM1 < Num_VeiT; zfM1++){
                                                  if
((No_lixo==endint_noV[zfM1])||(No_lixo==flag_NOS0[zfM1])||(No_lixo==melNO1[tpo+1][ifM1])){
                                                      lixoNV = 1;
                                                  }
                                              }
                                          }
                                          else{
                                              lixoNV = 1;
                                          }
                                          endint_noV[ifM1] = No_lixo;
                                    }
                                }
                                if (lixoNV == 1){
                                    endint_noV[ifM1] = 0;
                                }

                                if ((endint_noV[ifM1]<0)||(endint_noV[ifM1]>21)){
                                    PutStr ("error in endint_noV[ifM1]");
                                    sp = GetStr();
                                    IORTNCDE = -1;
                                }

                                No_lixo = 0;
                                lixoNV = 0;
                                ind_k5 = 0;
                                ind_k6 = 0;
                            }

                            /* escolha aleatória para os veiculos ainda sem no destino */
                            if ((endint_noV[ifM1]==0)){
                                /* no ORIGEM dos 16 nos maq. se possivel */
                                lix_contF = 0;
                                for (f_R1 = 0; f_R1 < Num_MaxV;f_R1++){
                                    for (f_R2 = 0; f_R2 < 16; f_R2++){
                                          if ((endint_noV[f_R1]==MAQ_NOS [f_R2])||(flag_NOS0[f_R1]==MAQ_NOS[f_R2])){
                                              lix_contF++;
                                          }
                                    }
                                }

                                Salto_21:
                                if (lix_contF > 14){
                                    M_wx = f_randoes (1,21);
                                }
                                else{
                                    M_wx = MAQ_NOS [f_randoes (1,16)-1];
                                }
```

```
                                ifM3 = 0;
                                for (ifM2 = 0; ifM2 < Num_MaxV; ifM2 ++){
                                    if
((M_wx==flag_NOS0[ifM2])||(M_wx==endint_noV[ifM2])||(((M_wx==staint_noV[ifM2])||(M_wx==melNO1[tpo+1][ifM2]))&&(ifM
1==ifM2))){
                                            ifM3 = 1;
                                    }
                                }
                                if (ifM3 == 1){
                                    goto Salto_21;
                                }

                                endint_noV [ifM1] = M_wx;
                                lix_contF = 0;
                            }
                        }

                        tpo_abs ++;
                        tpo ++;
                        tpo_lixo ++;

                        /* rotatividade dos veiculos e prioridades: pri N:N+1 */
                        for (M_wx=1 ; M_wx <= Num_VeiT; M_wx ++){
                            if (M_wx == 1 ){
                                vei_Pr [tpo][M_wx-1] = vei_Pr [tpo-1][Num_VeiT-1];
                            }
                            else if ((M_wx>1) && (M_wx<= Num_VeiT)){
                                vei_Pr [tpo][M_wx-1] = vei_Pr [tpo-1][M_wx-2];
                            }
                        }
                        for (M_wx=0 ; M_wx < Num_VeiT; M_wx ++){
                            vei_Pr_id [tpo][vei_Pr [tpo][M_wx]-1] = M_wx+1;
                        }
                        Veic_id = vei_Pr[tpo][0];
                        if (vei_Pr_id [tpo][Veic_id-1] != 1){
                            PutStr ("error in vei_Pr_id\n");

                            IORTNCDE = -1;
                        }
                        flag_vei = Veic_id;
                        flag_vpri = 1;

                        /* reset de tpo e variaveis associadas */

                        if (tpo > (M_Array_Dim-2)){
                            for (M_wy = 0; M_wy < (M_Array_Dim-1); M_wy ++){
                                for (M_wx = 0; M_wx < Num_VeiT; M_wx ++){
                                    melNO1 [M_wy][M_wx] = melNO1 [M_wy+1][M_wx];
                                    melNO2 [M_wy][M_wx] = melNO2 [M_wy+1][M_wx];
                                    melT [M_wy][M_wx] = melT [M_wy+1][M_wx];
                                    vei_Pr [M_wy][M_wx] = vei_Pr [M_wy+1][M_wx];
                                    vei_Pr_id [M_wy][M_wx] = vei_Pr_id [M_wy+1][M_wx];
                                }
                            }
                            for (M_wx = 0; M_wx < Num_VeiT; M_wx ++){
                                melNO1 [M_Array_Dim-1][M_wx] = 0;
                                melNO2 [M_Array_Dim-1][M_wx] = 0;
                                melT   [M_Array_Dim-1][M_wx] = 0;
                                vei_Pr [M_Array_Dim-1][M_wx] = 0;
                                vei_Pr_id [M_Array_Dim-1][M_wx] = 0;
                            }
                            tpo = M_Array_Dim - 2;
                        }

                        /* escrever informacao no ficheiro de resultados */
                        fprintf (fich_001,"\n%d\t",tpo);
                        for (M_wy = 0; M_wy < Num_VeiT; M_wy ++){
                            fprintf (fich_001,"%2d %2d %2d\t",melNO1[tpo][M_wy],endint_noV[M_wy],flag_NOS0[M_wy]);
                        }
                        fprintf (fich_001,"\n");

                        Veic_cont = 0;
                        ftramo_zero ();

                    }
                    /* NAO: actualizar curr_nod,end_nod para o proximo veiculo */
                    else if (Veic_cont < Num_VeiT){
                        /* escolher o proximo veiculo */
                        if (Veic_id == Num_VeiT){
                            Veic_id = 1;
                        }
                        else {
                            Veic_id ++;
                        }
                        flag_vei = Veic_id ;
                        flag_vpri = vei_Pr_id [tpo][Veic_id -1];
                    }
                    no1_main = 0;
                    no1_main = melNO1 [tpo][Veic_id -1];
                    if ((no1_main < 1) || (no1_main > 21) ){
                        PutStr ("error in no1_main");
                        sp = GetStr();

                        IORTNCDE = -1;
                    }

                    /*** actualizar curr_nod  ***/
                    zero_lix = f_dec2_bin (no1_main);
                    if (zero_lix < 0){
                        PutStr ("error in zero_lix");
                    }
                    for (M_wx = 0; M_wx < 5; M_wx++){
                        curr_nod [M_wx] = DEC2BIN [M_wx];
                    }
                    /** actualizar end_nod    **/
                    if ((endint_noV [Veic_id-1] < 1) || (endint_noV [Veic_id-1] > 21)){
                        PutStr ("error in end_noV[Veic_id-1]");
                        sp = GetStr();
                        IORTNCDE = -1;
```

```
        }
        zero_lix = f_dec2_bin (endint_noV[Veic_id-1]);
        if (zero_lix < 0){
              PutStr ("error in zero_lix");
        }
        for (M_wx = 0; M_wx < 5; M_wx++){
              end_nod [M_wx] = DEC2BIN [M_wx];
        }

        /** actualizar e array_tra **/
        for (M_wx = 0; M_wx < M_Array_Dim; M_wx++){
              array_tra [M_wx] = melT [M_wx][Veic_id-1];
        }

        break;

case RQ_RCLTST:
        break;
case RQ_REND:
        break;
case RQ_TERM:
        /************ close any files which may be open *************/
        fclose (fich_001);
        fclose (fich_002);
        fclose (fich_003);
                /*****************************************************************/
                PutStr ( "bye bye\n");
                break;
        }
        return;
}
```

1.2 Code Listing of Functions and Variables Used in Main Program

```
/******   FUH_ALLR.c  23/08/96            *******/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
/* inicializacao de variaveis que caracterizam o estado estatico */

#define PI 3.14159265359

/*************************************************************************/
/*                                                                       */
/*   NAO USAR OS NOMES DE VARIAVEIS: record, x_temp, y_temp  */
/*                                                                       */
/*************************************************************************/


static int ESTRATE = (-1);

static int Num_MaxV = (11);


/*  matrizes que representam a numeracao dos elementos da grelha */
/*  matriz no's origem destino e nos cruzamento   */

static int MATNOS[5][5] = {
    {1,2,3,4,5},
    {6,0,7,0,8},
    {9,10,11,12,13},
    {14,0,15,0,16},
    {17,18,19,20,21}
};

/*  matriz que representa o estado livre(=0) ou ocupado(=1) de cada no*/

static int MEST_NO[5][5] = {
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0}
};
/*  matriz que identifica o veiculo que se encontar em cada no*/

static int MVEI_NO[5][5] = {
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0}
};

/*  matrizes que representam o numero de saidas possiveis, e as */
/*    que ja foram usadas, de cada no de acordo com o estado   */
/*  dos tramos adjacentes a cada no */

static int NOS_TSAI[5][5] = {
    {2,2,3,2,2},
    {2,0,2,0,2},
    {3,2,4,2,3},
    {2,0,2,0,2},
    {2,2,3,2,2}
};

static int NOS_SAIP[5][5] = {
    {2,2,3,2,2},
    {2,0,2,0,2},
    {3,2,4,2,3},
    {2,0,2,0,2},
    {2,2,3,2,2}
};

/* matriz para contabilizar o numero de intencoes de ocupacao
de nos pelos veiculoe */

static int NOS_OCUP[5][5] = {
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0}
};

/* matriz para contabilizar o numero de intencoes de ocupacao
de nos pelos veiculos e para ser usada em f_st_ahead        */

static int NOS_AHEAD [5][5] = {
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0},
    {0,-1,0,-1,0},
    {0,0,0,0,0}
};

/* matriz tramos horizontais  */

static int MATTRH[3][4] = {
    {1,2,3,4},
    {11,12,13,14},
    {21,22,23,24}

};
```

```
/* matriz tramos verticais   */

static int MATTRV[4][3] = {
    {5,6,7},
    {8,9,10},
    {15,16,17},
    {18,19,20}

};
/* matrizes estado ocupado (=1) ou livre (=0) dos tramos */
/* tramos horizontais (=0, todos livres por defeito) */

static int MATTHE[3][4] = {
          {0,0,0,0},
          {0,0,0,0},
          {0,0,0,0}

};
/* tramos verticais (=0, todos livres por defeito) */

static int MATTVE[4][3] = {
          {0,0,0},
          {0,0,0},
          {0,0,0},
          {0,0,0}
};
/* matrizes[5][5] que identificam o estado dos tramos (R,L,U,D)       */
/* que saiem de cada no      */
/* matriz R: num. tramo que sai do no pela direita */

static int MATR[5][5] = {
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1}
};
/* matriz L: num. tramo que sai do no pela esquerda */

static int MATL[5][5] = {
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0}
};
/* matriz U: num. tramo que sai do no para cima */

static int MATU[5][5] = {
          {-1,-1,-1,-1,-1},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0}
};
/* matriz D: num. tramo que sai do no para cima */

static int MATD[5][5] = {
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {-1,-1,-1,-1,-1}
};
/* matrizes[5][5] que identificam o numero de ordem de cada tramo (RLUD) */
/* que sai de cada no          */
/* matriz R: num. tramo que sai do no pela direita */

static int MATR_R[5][5] = {
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1}
};
/* matriz L: num. tramo que sai do no pela esquerda */

static int MATR_L[5][5] = {
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0}
};
/* matriz U: num. tramo que sai do no para cima */

static int MATR_U[5][5] = {
          {-1,-1,-1,-1,-1},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0}
};
/* matriz D: num. tramo que sai do no para cima */

static int MATR_D[5][5] = {
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {-1,-1,-1,-1,-1}
};

/* matrizes[5][5] que identificam os tramos (R,L,U,D)            */
/* disponiveis (nao usados) que saiem de cada no       */
/*      ( = 0 : ainda nao foi usado, > 0: num de vezes usado) */

/* matriz R: num. tramo que sai do no pela direita */
```

```
static int TRADIS_R[5][5] = {
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1},
          {-1,-2,-1,-2,-1},
          {0,0,0,0,-1}
};
/* matriz L: num. tramo que sai do no pela esquerda */

static int TRADIS_L[5][5] = {
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0},
          {-1,-2,-1,-2,-1},
          {-1,0,0,0,0}
};
/* matriz U: num. tramo que sai do no para cima */

static int TRADIS_U[5][5] = {
          {-1,-1,-1,-1,-1},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0}
};
/* matriz D: num. tramo que sai do no para cima */

static int TRADIS_D[5][5] = {
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {0,-1,0,-1,0},
          {0,-2,0,-2,0},
          {-1,-1,-1,-1,-1}
};

/*                                                                              */
/*  arrays representing:                                                        */
/*         (binary) - start, end, current and next nodes                        */
/*                    (start_nod [], end_nod [], curr_nod [], next_nod [])      */
/*         (binary) - initial, current and next track states                    */
/*                    (inista_trk [], cursta_trk [], nextsta_trk [])            */
/*  (input vector) - all 34 elements (track states, start and end nodes)        */
/*                    (inpvec_ini [ ]),apenas inicializar(0) todos os elementos */
/*  (output vector)- two elements representing x,y coordinates as they are      */
/*                           obtained from neural net output format             */
/*                    (out_vec [ ]), apenas inicializar (0) todos os elementos  */
/*                                                                              */
/*         (decimal) - current node, move direction, next track, track state,   */
/*                   and next node  (record [ ] )                               */
/*                                                                              */
/*         (decimal)    - numero de passos                                      */
/*                           (passo_sum)                                        */
/*                                                                              */
static int start_nod          [5] = (0,0,0,0,0);
static int end_nod  [5] = (0,0,0,0,0);
static int curr_nod [5] = (0,0,0,0,0);
static int next_nod [5] = (0,0,0,0,0);

static int cursta_trk         [24] =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

static float inp_vec [ 34]=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
static float out_vec [ 2] =(0,0);

static int registo [ 5] = (0,0,0,0,0);


static int array_nos [15];     /*  arrays para armazenar nos e tramos  */
static int array_tra [15];     /*   percorridos                        */
static int tipo_sol;               /*   ou regra usada para sair do no          */
                                   /* 1==ANN; 2==ANNX/Y; == ; == */

static long int cont_casos = (0);        /*  contagem de todos os casos lidos do fich. */


static int flag_est[5] = (0,0,0,0,0);
static int flag_ciclo = (0);
static int flag_exa = (0);

/**** variaveis adicionais para o caso de dois, ou mais veiculos (A e B) ***/
/*******  a usarem a mesma rede                                        **/
static int Veic_id, Veic_cont;
static int tipo_solV[15][11];
static int tpo;
static int melNO1[15][11], melNO2[15][11], melT[15][11];
static int vei_Pr [15][11]; /* para mais veiculos mudar [11] */
static int vei_Pr_id[15][11]; /* identifica a prioridade de cada veiculo em cada passo*/
static long int array_ord [5][2];       /* serve para guardar sequencias ordenados de 11? numeros */
static int temp_seq [5] ; /* serve para guardar as sequencias aleatorias dos numeros **s */
static int staint_noV[11], curint_noV[11], endint_noV[11]; /* nos origem/destino em decimal */

/*********************************************************/
static int flag_vei = (0);
static int flag_vpri = (0);
static int Num_VeiT = (0);
static int tip_lixo = (0);
static    int int_ordA [11][2]={
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
};
```

```
static int DEC2BIN [5]= {-1,-1,-1,-1,-1};

static int VEI [11]= {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};

/*********************************************************/

static char *splixo, buflixo[100];

void mtra_RLUD (void);
void tramtec ();
void tramoe (int);
void mat_RLUD (void);
double    conv1 (double  coor1);
double    modxy ( double      xcoor, double  ycoor);
double    angxy ( double      xcoor, double  ycoor);
int indexj ( int num_no1);
int indexi ( int num_no2);
void fvec_inp(int *trak_ptr, int *node1_ptr, int *node2_ptr);
void ftramo_val (int tra1_num, int tra1_est); /* pode substituir ftramo_upd*/
void ftramo_zero ();
void eva_strat1 (int node_num1,int no_fim,int pre_tra,double xcoor2,double ycoor2,double min_mod2);
int no_dist (int nodis1,int nodis2);    /** calcular a distancia entre 2 nos c/ grelha livre**/
int ftra_estid(int num_tra);  /** determinar o estado do tramo num_tra **/
void fnos_tblo();  /** act. matriz num. total sai. possiveis nos c/tra. act.*/
void fnos_tliv();  /** act. matriz num. total sai. possiveis nos c/tra. livres  */
void fsaip_ini (); /** actualizar NOS_SAIP [][] == NOS_TSAI [][]   ***/
void ftra_liv();  /** inicializacao das matrizes TRADIS_R/L/U/D [][]   ***/
void fmest_no_0 (); /* reset da matriz estado dos nos */
void fmest_no_N (int no_N, int val_est); /*** actualiza na matriz MEST_NO[][]
                              o valor do no indicado */
void fmat_noV0 (); /* reset da matriz veiculos ocupam nos (MVEI_NO[][])*/
void fmat_noV ( int no_ocumat, int vei_nmat); /*** actualiza na matriz MVEI_NO[][]
                              o valor do no indicado */
void fbtra_adj (int no_dest); /* bloquear tramos adjacentes a um no */

int f_randoes (int min_ele, int max_ele); /* gerar um numero pseudo-aleatorio entre
                                        os dois valores especficados */
void f_ordem (); /*definir as prioridades dos elementos array [5][0]
                        de acordo com os seus valores  */
void f_ordigu(); /*  trata dos casos em que ha mais do que uma solucao igual */
void f_aleaord (int num_aleas); /* ordenar aleatoriamente o conjunto de sol. iguais */

/* decidir qual o tramo que o veiculo pretende utilizar e actualiza
 o valor melNO2[][] respectivo que indica qual o no que pretende
atingir ao usar esse tramo: */

void f_N_ocup();
void f_N_ocpdo (int no_N1, int val_est1);

void f_Nos_ahead();
void f_st_ahead (int step_ind); /* analisa o NO2 do 1§veic. */

int f_NO2M_1 ();     /* analisa o NO2 do 2§veic. */
int f_NO2M_2 ();     /* analisa o NO2 do 3§veic. */
int f_NO2M_3 ();     /* analisa o NO2 do 4§veic. */
int f_NO2M_4 ();     /* analisa o NO2 do 5§veic. */
int f_NO2M_5 ();     /* analisa o NO2 do 6§veic. */
int f_NO2M_6 ();     /* analisa o NO2 do 7§veic. */
int f_NO2M_7 ();     /* analisa o NO2 do 8§veic. */
int f_NO2M_8 ();     /* analisa o NO2 do 9§veic. */
int f_NO2M_9 ();     /* analisa o NO2 do 10§veic. */
int f_NO2M_LAST ();   /* analisa o NO2 do 11§veic. ou ultimo veiculo a
                              considerar dependendo da amplitude do passo
                              escolhido */

int f_NO2M_1A ();     /* analisa apenas o NO2 do 2§veic.(look-ahead limitado)*/
int f_NO2M5 (int jdfM5);


int f_dec2_bin (int decinum);
int f_Int5_2dec (int sce_nod);



/*************************************************************************/

/*  funcao para converter um numero binario em decimal
    em que o binario sera 1:start_nod, 2:curr_nod, 3:end_nod
    e usando DEC2BIN[]        */
int f_Int5_2dec (int sce_nod )
{   int i_b2d, sum_b2d = 0, numdig = 5, flag_choose;

    flag_choose = sce_nod;
    for ( i_b2d = 0; i_b2d < numdig; i_b2d++){
        if (flag_choose == 1){
            DEC2BIN [i_b2d] = start_nod [i_b2d];
        }
        else if (flag_choose == 2){
            DEC2BIN [i_b2d] = curr_nod [i_b2d];
        }
        else if (flag_choose == 3){
            DEC2BIN [i_b2d] = end_nod [i_b2d];
        }
        else{
            return (-2);
        }

        if ((DEC2BIN [i_b2d] < 0)||(DEC2BIN [i_b2d] > 1)){
            return (-1);
        }
        sum_b2d = sum_b2d + (DEC2BIN [i_b2d] * (pow(2,(4 -i_b2d))));
    }
    return (sum_b2d);
}

/*  funcao para converter numero decimal em binario com 5 digitos e */
/*  usando uma variavel global (int DEC2BIN[5]) para guardar esse valor */
```

```
int f_dec2_bin ( int decinum)
{
    unsigned index_fdec;
    int  resultado, bin_lugar, valor, controlo;

    bin_lugar = 0;
    valor = decinum;
    controlo = decinum;
    for (index_fdec = 16; index_fdec > 0; index_fdec /= 2){
          DEC2BIN [bin_lugar] = -1;
          resultado = (int) ( valor / index_fdec);
          if (resultado == 1){
              DEC2BIN [bin_lugar] = 1;
          }
          else{
              DEC2BIN [bin_lugar] = 0;
          }
          if ((DEC2BIN [bin_lugar] < 0)||(DEC2BIN [bin_lugar] > 1)){
              return (-1);
          }
          bin_lugar = bin_lugar +1;
          valor %= index_fdec;
    }
    if ((controlo < 0)||(controlo > 21)){
          for (index_fdec = 0; index_fdec < 5; index_fdec++){
              DEC2BIN [index_fdec] = -1;
          }
          return (-2);
    }
    if ( bin_lugar == 5){
          return (1);
    }
    else{
          return (-3);
    }
}


/*  funcao para colocar os valores da matriz ocupacao dos nos (NOS_OCUP[5][5])
correspondentes a nos existentes todos livres     */
void f_N_ocup()
{   int i_ocup, j_ocup;

    for (i_ocup = 0; i_ocup < 5; i_ocup ++){
          for (j_ocup = 0; j_ocup < 5; j_ocup ++){
              if (MATNOS [i_ocup][j_ocup] >= 1 && MATNOS [i_ocup][j_ocup] <= 21){
                    NOS_OCUP [i_ocup][j_ocup] = 0;
              }
              else if (MATNOS [i_ocup][j_ocup] == 0){
                    NOS_OCUP [i_ocup][j_ocup] = -1;
              }
              else {
                    /* PutStr ("error in MATNOS "); */
              }
          }
    }
}
/*  funcao para colocar o valor indicado no elemento
correspondente da matriz ocupacao dos nos (NOS_OCUP[5][5])  ***/
void f_N_ocpdo (int no_N1, int val_est1)
{   int i_ocp, j_ocp, no_ocp, esta_ocp;

    no_ocp = no_N1;
    esta_ocp = val_est1;
    if (no_ocp >0 && no_ocp <= 21){
          i_ocp = indexi (no_ocp);
          j_ocp = indexj (no_ocp);
          NOS_OCUP [i_ocp][j_ocp] = esta_ocp;
    }
    else{
          /*PutStr ("f_N_ocupado,error in node number:");*/
    }
}

/*  funcao para colocar os valores da matriz estado dos nos (MEST_NO)
correspondentes a nos existentes todos livres     */
void fmest_no_0()
{   int i_mest, j_mest;

    for (i_mest = 0; i_mest < 5; i_mest ++){
          for (j_mest = 0; j_mest < 5; j_mest ++){
              if (MATNOS [i_mest][j_mest] >= 1 && MATNOS [i_mest][j_mest] <= 21){
                    MEST_NO [i_mest][j_mest] = 0;
              }
              else if (MATNOS [i_mest][j_mest] == 0){
                    MEST_NO [i_mest][j_mest] = -1;
              }
              else {
                    /*PutStr ("error in MATNOS "); */
              }
          }
    }
}
/*  funcao para colocar o valor indicado no elemento
correspondente da matriz estado dos nos (MEST_NO)  ***/
void fmest_no_N(int no_N, int val_est)
{   int i_mest, j_mest, no_mesN, esta_mesN;

    no_mesN = no_N;
    esta_mesN = val_est;
    if (no_mesN >0 && no_mesN <= 21){
          i_mest = indexi (no_mesN);
          j_mest = indexj (no_mesN);
          MEST_NO [i_mest][j_mest] = esta_mesN;
    }
    else{
          /*PutStr ("fmest_no_N,error in node number:");*/
/*        sprintf(
          buflixo,"%d \n",no_mesN);
```

```
            PutStr (buflixo);
            splixo = GetStr();*/
        }
}

/*************************************************************************/
/*  funcao para colocar os valores da matriz veiculos em nos (MVEI_NO)
correspondentes a nenhum veiculo, ou seja todos nos desocupados       */
void fmat_noV0()
{   int i_mat, j_mat;

    for (i_mat = 0; i_mat < 5; i_mat ++){
          for (j_mat = 0; j_mat < 5; j_mat ++){
              if (MATNOS [i_mat][j_mat] >= 1 && MATNOS [i_mat][j_mat] <= 21){
                    MVEI_NO [i_mat][j_mat] = 0;
              }
              else if (MATNOS [i_mat][j_mat] == 0){
                    MVEI_NO [i_mat][j_mat] = -1;
              }
              else {
                    /*PutStr ("error in MATNOS ");*/
              }
          }
    }
}

/*  funcao para colocar o valor indicado no elemento
correspondente da matriz veiculos em nos (MVEI_NO)          ***/
void fmat_noV ( int no_ocumat, int vei_nmat)
{   int i_matN, j_matN, no_matN, esta_matN;

    no_matN = no_ocumat;
    esta_matN = vei_nmat;
    if (no_matN >0 && no_matN <= 21){
          i_matN = indexi (no_matN);
          j_matN = indexj (no_matN);
          MVEI_NO [i_matN][j_matN] = esta_matN;
    }
    else{
          /*PutStr ("fmat_noV,error in node number:");*/
/*        sprintf(buflixo,"%d \n",no_numero);
          PutStr (buflixo);
          splixo = GetStr(); */
    }
}

/***  funcao para determinar o estado do tramo a partir do seu        ********/
/*****************  numero de ordem     ***********************************/

int ftra_estid(int num_tra)
{   int t_est_id, tramo1, i_cic, j_cic;

    tramo1 = num_tra;
    t_est_id = -9;

    for (i_cic = 0; i_cic <3; i_cic++){
    for (j_cic = 0; j_cic <4; j_cic++){
          if (MATTRH [i_cic][j_cic] == tramo1){
              t_est_id = MATTHE [i_cic][j_cic];
              goto fun_festid;
          }
    }
    }
    for (i_cic = 0; i_cic <4; i_cic++){
    for (j_cic = 0; j_cic <3; j_cic++){
          if (MATTRV [i_cic][j_cic] == tramo1){
              t_est_id = MATTVE [i_cic][j_cic];
              goto fun_festid;
          }
    }
    }
    fun_festid:

    return t_est_id;

}

/*************************************************************************/
/***  funcao para determinar a distancia entre dois nos        ********/

int no_dist (int nodis1,int nodis2)
{   double i_disx1, i_disx2, i_disy1, i_disy2, dist12, tr_Cdist;
   int no1_Ndis, no2_Ndis;

    tr_Cdist = 10.0000;
    no1_Ndis = nodis1;
    no2_Ndis = nodis2;

    i_disx1 = (double) indexi(no1_Ndis);
    i_disy1 = (double) indexj(no1_Ndis);
    i_disx2 = (double) indexi(no2_Ndis);
    i_disy2 = (double) indexj(no2_Ndis);

    dist12 = (fabs(i_disx2 - i_disx1) + fabs(i_disy2 - i_disy1)) * tr_Cdist;

    if ( i_disy1 == i_disy2 && i_disx1 != i_disx2 && i_disy1 != 0 && i_disy1 != 2 && i_disy1 != 4){

          dist12 = dist12 + 2*tr_Cdist;

    }
    if (i_disx1 == i_disx2 && i_disy1 != i_disy2 && i_disx1 != 0 && i_disx1 != 2 && i_disx1 != 4 && dist12 != 0){

          dist12 = dist12 + 2*tr_Cdist;

    }

    return (int) dist12;
}

/*************************************************************************/
```

```
/* funcao para fazer reset (a zero) dos valores das matrizes                */
/*    que representam o estado dos tramos (MATTHE e MATTVE)      */

void ftramo_zero ()
{  int ind_zero1, ind_zero2;

/* reset a zero dos valores das matrizes estado dos tramos   */

    for (ind_zero1 = 0; ind_zero1 < 3; ind_zero1++){
    for (ind_zero2 = 0; ind_zero2 < 4; ind_zero2++){
          MATTHE [ind_zero1][ind_zero2] = 0;
    }
    }
    for (ind_zero1 = 0; ind_zero1 < 4; ind_zero1++){
    for (ind_zero2 = 0; ind_zero2 < 3; ind_zero2++){
          MATTVE [ind_zero1][ind_zero2] = 0;
    }
    }
}

/* fun‡ao para actualizar o elemento da matriz estado dos tramos       */
/*    (MATTRH ou MATTRV) correspondente ao numero de ordem do tramo    */
/*     a que se pretende atribuir um dado estado (ex. 1=bloq.; 0=livre)          */

void ftramo_val (int tra1_num, int tra1_est)
{  int tramo_val, traest_val, ind1_val, ind2_val;

    tramo_val = tra1_num;
    traest_val = tra1_est;

    for (ind1_val = 0; ind1_val <3; ind1_val++){
    for (ind2_val = 0; ind2_val <4; ind2_val++){
          if (MATTRH [ind1_val][ind2_val] == tramo_val){
              MATTHE[ind1_val][ind2_val] = traest_val;
              goto fun_fval;
          }
    }
    }
    for (ind1_val = 0; ind1_val <4; ind1_val++){
    for (ind2_val = 0; ind2_val <3; ind2_val++){
          if (MATTRV [ind1_val][ind2_val] == tramo_val){
              MATTVE[ind1_val][ind2_val] = traest_val;
              goto fun_fval;
          }
    }
    }

    fun_fval:
    return;
}

/* funcao para bloquear os tramos adjacentes a um no */
void fbtra_adj (int no_dest)
{ int no_ocadj, i_adj, j_adj, tra_adj;

    no_ocadj = no_dest;
    i_adj = indexi (no_ocadj);
    j_adj = indexj (no_ocadj);
    tra_adj = MATR_R [i_adj][j_adj];

    if (tra_adj > 0 && tra_adj <= 24){
          ftramo_val (tra_adj,1);
    }
    tra_adj = MATR_L [i_adj][j_adj];
    if (tra_adj > 0 && tra_adj <= 24){
          ftramo_val (tra_adj,1);
    }
    tra_adj = MATR_U [i_adj][j_adj];
    if (tra_adj > 0 && tra_adj <= 24){
          ftramo_val (tra_adj,1);
    }
    tra_adj = MATR_D [i_adj][j_adj];
    if (tra_adj > 0 && tra_adj <= 24){
          ftramo_val (tra_adj,1);
    }

}


/* funcao para actualizar o valor do vector de entrada a partir dos  */
/*  dos arrays que representam o estado dos 24 tracks, o numero do   */
/*   no de origem e no destino em binario de 5 digitos                      */

void fvec_inp(int *trak_ptr, int *node1_ptr, int *node2_ptr)
{   int index_inp;

    for (index_inp = 0; index_inp < 34 ; index_inp++){
          if (index_inp < 24){
              inp_vec [index_inp] = * (trak_ptr + index_inp);}
          else if ((index_inp > 23) && (index_inp < 29)){
              inp_vec [index_inp] = *(node1_ptr + (index_inp - 24));}
          else if ((index_inp > 28) && (index_inp < 34)){
              inp_vec [index_inp] = *(node2_ptr + (index_inp - 29));}
          else if (index_inp > 34){
              printf ("error in fvec_inp");}
    }
}

/* actualizacao das matrizes MATR/L/U/D que representam o estado     */
/*    dos tramos que saiem de cada no para a direita, esquerda,      */
/*    para cima, ou para baixo (R,L,U,D)                                      */
/*         ocupado (=1), livre (=0), nao existe na grelha o no (=-2), */
/*         nao existe para aquele no o tramo (=-1)                           */

void mat_RLUD(void)
{
    int i_RLUD, j_RLUD;
```

```
    for (i_RLUD=0;i_RLUD<5;i_RLUD += 2){
    for (j_RLUD=0;j_RLUD<4;j_RLUD++){
          MATR[i_RLUD][j_RLUD] = MATTHE[i_RLUD/2][j_RLUD];
          MATL[i_RLUD][j_RLUD+1]=MATTHE[i_RLUD/2][j_RLUD];
    }
    }
    for (i_RLUD=0;i_RLUD<4;i_RLUD++){
    for (j_RLUD=0;j_RLUD<5;j_RLUD += 2){
          MATD[i_RLUD][j_RLUD] = MATTVE[i_RLUD][j_RLUD/2];
          MATU[i_RLUD+1][j_RLUD]=MATTVE[i_RLUD][j_RLUD/2];
    }
    }
}

/*  initializacao das matrizes que representam os tramos    */
/*   ligados a cada no (MATR_R, MATR_L, MATR_U, MATR_D)     */
void mtra_RLUD(void)
{
    int im_RLUD, jm_RLUD;

    for (im_RLUD=0;im_RLUD<5;im_RLUD += 2){
    for (jm_RLUD=0;jm_RLUD<4;jm_RLUD++){
          MATR_R[im_RLUD][jm_RLUD] = MATTRH[im_RLUD/2][jm_RLUD];
          MATR_L[im_RLUD][jm_RLUD+1]=MATTRH[im_RLUD/2][jm_RLUD];
    }
    }
    for (im_RLUD=0;im_RLUD<4;im_RLUD++){
    for (jm_RLUD=0;jm_RLUD<5;jm_RLUD += 2){
          MATR_D[im_RLUD][jm_RLUD] = MATTRV[im_RLUD][jm_RLUD/2];
          MATR_U[im_RLUD+1][jm_RLUD]=MATTRV[im_RLUD][jm_RLUD/2];
    }
    }
}

/** inicializacao das matrizes TRADIS_R/L/U/D [][]   ***/

void ftra_liv()
{   int i_liv, j_liv, imax_liv, jmax_liv;

    imax_liv = 5;
    jmax_liv = 5;

    for (i_liv      = 0;i_liv < imax_liv;i_liv += 2){
    for (j_liv = 0;j_liv < jmax_liv - 1;j_liv++){
          TRADIS_R [i_liv][j_liv]   = 0;
          TRADIS_L [i_liv][j_liv+1] = 0;
    }
    }
    for (i_liv = 0;i_liv < imax_liv -1;i_liv++){
    for (j_liv = 0;j_liv < jmax_liv;j_liv += 2){
          TRADIS_D [i_liv][j_liv]   = 0;
          TRADIS_U [i_liv+1][j_liv] = 0;
    }
    }
}

/**   actualizar  NOS_TSAI[5][5] com o numero possivel de saidas de **/
/**       cada no se todos os tramos na grelha estivessem livres      */
void fnos_tliv()
{   int i_tliv, j_tliv, imax_tliv, jmax_tliv;

    imax_tliv = 5;
    jmax_tliv = 5;

    for (i_tliv = 1; i_tliv < imax_tliv - 1; i_tliv += 2){
          for (j_tliv = 1; j_tliv < jmax_tliv -1; j_tliv += 2){
              NOS_TSAI [i_tliv][j_tliv] = 0;
          }
    }
    for (i_tliv = 1; i_tliv < imax_tliv - 1; i_tliv += 2){
          for (j_tliv = 0; j_tliv < jmax_tliv ; j_tliv += 2){
              NOS_TSAI [i_tliv][j_tliv] = 2;
          }
    }
    for (i_tliv = 0; i_tliv < imax_tliv ; i_tliv += 2){
          for (j_tliv = 1; j_tliv < jmax_tliv - 1; j_tliv += 2){
              NOS_TSAI [i_tliv][j_tliv] = 2;
          }
    }
    for (i_tliv = 2; i_tliv < imax_tliv - 2; i_tliv += 2){
          for (j_tliv = 2; j_tliv < jmax_tliv - 2 ; j_tliv += 2){
              NOS_TSAI [i_tliv][j_tliv] = 4;
          }
    }
    for (i_tliv = 0; i_tliv < imax_tliv; i_tliv += 2){
          if (i_tliv == 0 || i_tliv == imax_tliv -1){
              NOS_TSAI [i_tliv][0] = 2;
              NOS_TSAI [i_tliv][jmax_tliv - 1] = 2;
          }
          else {
              NOS_TSAI [i_tliv][0] = 3;
              NOS_TSAI [i_tliv][jmax_tliv - 1] = 3;
          }
    }

    for (j_tliv = 2; j_tliv < imax_tliv - 2; j_tliv += 2){
          NOS_TSAI [0][j_tliv] = 3;
          NOS_TSAI [imax_tliv][j_tliv] = 3;
    }
}

/**   actualizar  NOS_TSAI[5][5] com base no estado dos tramos ligados */
/**       a cada no   */

void fnos_tblo()
{   int i_tblo, j_tblo;

    mtra_RLUD ();
    fnos_tliv ();
```

```
    for (i_tblo = 0; i_tblo < 5; i_tblo++){
          for (j_tblo = 0; j_tblo < 5; j_tblo++){

              if (MATR [i_tblo][j_tblo] == 1){
                    NOS_TSAI [i_tblo][j_tblo] = NOS_TSAI [i_tblo][j_tblo] - 1;
              }
              if (MATL [i_tblo][j_tblo] == 1){
                    NOS_TSAI [i_tblo][j_tblo] = NOS_TSAI [i_tblo][j_tblo] - 1;
              }
              if (MATU [i_tblo][j_tblo] == 1){
                    NOS_TSAI [i_tblo][j_tblo] = NOS_TSAI [i_tblo][j_tblo] - 1;
              }
              if (MATD [i_tblo][j_tblo] == 1){
                    NOS_TSAI [i_tblo][j_tblo] = NOS_TSAI [i_tblo][j_tblo] - 1;
              }
          }
    }
}

/***  inicializar a matriz NOS_SAIP [5][5] com os valores da                    */
/******* matriz NOS_TSAI [5][5] , o que correspondera ao inicio do ciclo  */
void fsaip_ini ()
{   int i_sini, j_sini;

    for (i_sini = 0; i_sini < 5; i_sini++){
          for (j_sini = 0; j_sini < 5; j_sini++){
              NOS_SAIP [i_sini][j_sini] = NOS_TSAI [i_sini][j_sini];
          }
    }
}

/****  funcao para actualizar NOS_SAIP [][] com base no tramo usado   **/
/**                  saindo do no especificado     **/

void fsaip_upd (int no_dado, int tra_usado)
{   int no_supd, tra_supd, ni_supd, nj_supd;

    no_supd = no_dado;
    tra_supd = tra_usado;

    ni_supd = indexi (no_supd);
    nj_supd = indexj (no_supd);

    if (MATR_R [ni_supd][nj_supd] == tra_supd){
          TRADIS_R [ni_supd][nj_supd] = TRADIS_R [ni_supd][nj_supd] + 1;
    }
    if (MATR_L [ni_supd][nj_supd] == tra_supd){
          TRADIS_L [ni_supd][nj_supd] = TRADIS_L [ni_supd][nj_supd] + 1;
    }
    if (MATR_U [ni_supd][nj_supd] == tra_supd){
          TRADIS_U [ni_supd][nj_supd] = TRADIS_U [ni_supd][nj_supd] + 1;
    }
    if (MATR_D [ni_supd][nj_supd] == tra_supd){
          TRADIS_D [ni_supd][nj_supd] = TRADIS_D [ni_supd][nj_supd] + 1;
   }
    NOS_SAIP [ni_supd][nj_supd] = NOS_SAIP [ni_supd][nj_supd] - 1;

}

/* converter valores das coordenadas x ou y de [0,1] para [-1,1] */
double    conv1(double  coor1)
{
    double coor2_nv1;
    coor2_nv1 = (coor1-0.5)*2;
    return coor2_nv1;
}

/* determinar o modulo e angulo do vector associado as coordenadas x e y */
double    modxy(double  xcoor, double  ycoor)
{
    double  modu_dxy;
    modu_dxy = sqrt(xcoor*xcoor + ycoor*ycoor);
    return modu_dxy;

}
/*  para (x,y) = (0,0) a funcao nao esta definida, mas essa situacao  */
/*        devera ser prevista quando se calcular o modulo          */
double    angxy(double  xcoor, double  ycoor)
{
    double  angu_gxy;
    angu_gxy = atan2(ycoor,xcoor)*180.0/PI;
    return angu_gxy;
}

/* definir matrizes que caracterizam o estado dinamico                */
/*        (ex. ocupado/livre )   dos tramos da grelha )               */
/*                                                                    */
/* atribuicao dos tramos ocupados (MATTRH/V = 1)            */
void tramoe(int num_ord)
{
    int i_moe, j_moe;

    for (i_moe=0;i_moe<4;i_moe++){
    for (j_moe=0;j_moe<5;j_moe++){
          if (MATTRH[i_moe][j_moe] == num_ord){
          MATTHE[i_moe][j_moe] = 1;
          }
    }
    }
    for (i_moe=0;i_moe<5;i_moe++){
    for (j_moe=0;j_moe<4;j_moe++){
       if (MATTRV[i_moe][j_moe] == num_ord){
       MATTVE[i_moe][j_moe] = 1;
       }
    }
    }
}
```

```
/*  determinar os indices i,j da MATNOS a partir do numero do no  */
int indexj(int num_no1)
{
    int i_exj, j_exj, c_exj;

    for (i_exj=0;i_exj<5;i_exj++){
    for (j_exj=0;j_exj<5;j_exj++){
          if(MATNOS[i_exj][j_exj] == num_no1){
          c_exj = j_exj;
          }
    }
    }
    return c_exj;

}
int indexi(int num_no2)
{
    int i_exi, j_exi, c_exi;

    for (i_exi=0;i_exi<5;i_exi++){
    for (j_exi=0;j_exi<5;j_exi++){
          if(MATNOS[i_exi][j_exi] == num_no2){
          c_exi = i_exi;
          }
    }
    }
    return c_exi;

}


/***   funcao que implementa a estrategia de movimentacao na grelha
em que a saida da rede e reanalizada se indicar paragem ou movimento
para um tramo que nao existe ou esta bloqueado.  A situacao de paragem
nao e admitida excepto se o modulo do vector XY <= min_mod2.Neste caso a
rede escolhera o tramo disponivel que esteja mais perto.  Se indicar
tramo bloqueado ou inexistente so se considera a saida da rede que for
for possivel de implementar (X ou Y).       *******/


void eva_strat1 (int node_num1,int no_fim,int pre_tra,double xcoor2,double ycoor2,double min_mod2)
{   int n1_at1,n2_at1,tra0_at1,i1_at1,i2_at1,if_at1;
    int noR_at1 , noL_at1, noU_at1, noD_at1, cRLUD_at1, disR_at1, disL_at1, disU_at1, disD_at1;
    int tr_noat1 [4][2]= {
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0},
                                };
    int disde_at1[4][2]={
                                {0,0},
                                {0,0},
                                {0,0},
                                {0,0}
                                };
int cont1_at1 = 0, prdr_at1 = 0;
double vsaix_at1, vsaiy_at1, mod0_at1;
double angu1at1, angu2at1, angu3at1, angu4at1, mdxy2_at1, agxy2_at1;
char Mov_Sol2, CharDir ='A';

int noseg_at1, trseg_at1, trest_at1, ndir_at1 = -1;

tipo_sol = 0;
cRLUD_at1 = 0;
n1_at1 = node_num1;
n2_at1 = no_fim;
tra0_at1 = pre_tra;
vsaix_at1 = xcoor2;
vsaiy_at1 = ycoor2;
mod0_at1 = min_mod2;

if ((n1_at1 < 1)||(n2_at1 < 1)||(n1_at1 > 21)||(n2_at1 > 21)){
      n1_at1 = -9;
      noseg_at1 = -9;
      ndir_at1 = -9;
      trseg_at1 = -9;
      trest_at1 = -9;
      goto Fim_evastra;
}


i1_at1 = indexi (n1_at1);
i2_at1 = indexj (n1_at1);

    Mov_Sol2 = 'B';
    angu1at1 = 45.00001;
    angu2at1 = -44.99999;
    angu3at1 = 135.00001;
    angu4at1 = -134.99999;

    mdxy2_at1 = modxy (vsaix_at1,vsaiy_at1);
    agxy2_at1 = 0.00000;

/*******  antes de determinar "RLUDS" verifico os tramos do no1 ***/
/** verificando tambem qual dos tramos(direccao) pode corresponder**/
/************************      ao ultimo tramo usado **/

/* sem verificar o estado de ocupacao dos Nos correspondentes
aos tramos em consideracao (a usar com NNHANN77.c, NNHAN773.c, ou NNHEUAB7.c ...)*/

    if ((ESTRATE == 0)||(ESTRATE == 1)){

          for (if_at1=0;if_at1<4;if_at1++){
              if (if_at1==0){
                    tr_noat1[if_at1][0] = MATR_R [i1_at1][i2_at1];
                    tr_noat1[if_at1][1] = MATR [i1_at1][i2_at1];
                    if (tr_noat1[if_at1][0] == tra0_at1){
                        CharDir = 'R';
```

```
            prdr_at1 = 1;
        }
}
else if (if_at1 == 1){
        tr_noat1[if_at1][0] = MATR_L [i1_at1][i2_at1];
        tr_noat1[if_at1][1] = MATL [i1_at1][i2_at1];
        if (tr_noat1[if_at1][0] == tra0_at1){
            CharDir = 'L';
            prdr_at1 = 3;
        }
}
else if (if_at1 == 2){
        tr_noat1[if_at1][0] = MATR_U [i1_at1][i2_at1];
        tr_noat1[if_at1][1] = MATU [i1_at1][i2_at1];
        if (tr_noat1[if_at1][0] == tra0_at1){
            CharDir = 'U';
            prdr_at1 = 2;
        }
}
else if (if_at1 == 3){
        tr_noat1[if_at1][0] = MATR_D [i1_at1][i2_at1];
        tr_noat1[if_at1][1] = MATD [i1_at1][i2_at1];
        if (tr_noat1[if_at1][0] == tra0_at1){
            CharDir = 'D';
            prdr_at1 = 4;
                }
            }
        }
}
else if (ESTRATE == 9){
        /* verificando o estado de ocupacao dos Nos correspondentes
        aos tramos em consideracao (a usar com NNHANN78.c ) e o estado
        dos tramos para definir o estado dos tramos */

        for (if_at1=0;if_at1<4;if_at1++){
            if (if_at1==0){
                tr_noat1[if_at1][0] = MATR_R [i1_at1][i2_at1];
                tr_noat1[if_at1][1] = MATR [i1_at1][i2_at1];
                if (tr_noat1[if_at1][0] == tra0_at1){
                    CharDir = 'R';
                    prdr_at1 = 1;
                }
                if (tr_noat1 [if_at1][0] > 0 && tr_noat1 [if_at1][0] < 25){
                    if (MEST_NO [i1_at1][i2_at1 +1] != 0){
                        tr_noat1[if_at1][1] = 1;
                    }
                }
            }
            else if (if_at1 == 1){
                tr_noat1[if_at1][0] = MATR_L [i1_at1][i2_at1];
                tr_noat1[if_at1][1] = MATL [i1_at1][i2_at1];
                if (tr_noat1[if_at1][0] == tra0_at1){
                    CharDir = 'L';
                    prdr_at1 = 3;
                }
                if (tr_noat1 [if_at1][0] > 0 && tr_noat1 [if_at1][0] < 25){
                    if (MEST_NO [i1_at1][i2_at1 -1] != 0){
                        tr_noat1[if_at1][1] = 1;
                    }
                }
        }
        else if (if_at1 == 2){
                tr_noat1[if_at1][0] = MATR_U [i1_at1][i2_at1];
                tr_noat1[if_at1][1] = MATU [i1_at1][i2_at1];
                if (tr_noat1[if_at1][0] == tra0_at1){
                    CharDir = 'U';
                    prdr_at1 = 2;
                }
                if (tr_noat1 [if_at1][0] > 0 && tr_noat1 [if_at1][0] < 25){
                    if (MEST_NO [i1_at1-1][i2_at1] != 0){
                        tr_noat1[if_at1][1] = 1;
                    }
                }
        }
        else if (if_at1 == 3){
                tr_noat1[if_at1][0] = MATR_D [i1_at1][i2_at1];
                tr_noat1[if_at1][1] = MATD [i1_at1][i2_at1];
                if (tr_noat1[if_at1][0] == tra0_at1){
                    CharDir = 'D';
                    prdr_at1 = 4;
                }
                if (tr_noat1 [if_at1][0] > 0 && tr_noat1 [if_at1][0] < 25){
                    if (MEST_NO [i1_at1+1][i2_at1] != 0){
                        tr_noat1[if_at1][1] = 1;
                    }
                }
        }
    }
    else{
            /* PutStr ("error in ESTRATE");*/
    }

/*************************************************************************/

/**** determinar os nos mais proximos do n1_at1 e as suas distancias*/
/***       ao no destino (!!!mesmo que bloqueados:=1???? nao como esta!)****/

    if ( tr_noat1 [0][1] == 0 ){
            noR_at1 = MATNOS [i1_at1][i2_at1 + 1];
            disR_at1 = no_dist (noR_at1,n2_at1);
            cRLUD_at1++;
    }
    else{
            noR_at1 = 0;
            disR_at1 = -1;
    }
    if ( tr_noat1 [1][1] == 0 ){
            noL_at1 = MATNOS [i1_at1][i2_at1 - 1];
```

```
        disL_at1 = no_dist (noL_at1,n2_at1);
        cRLUD_at1++;
}
else{
        noL_at1 = 0;
        disL_at1 = -1;
}
if ( tr_noat1 [2][1] == 0 ){
        noU_at1 = MATNOS [i1_at1 - 1][i2_at1];
        disU_at1 = no_dist (noU_at1,n2_at1);
        cRLUD_at1++;
}
else{
        noU_at1 = 0;
        disU_at1 = -1;
}
if ( tr_noat1 [3][1] == 0 ){
        noD_at1 = MATNOS [i1_at1 + 1][i2_at1];
        disD_at1 = no_dist (noD_at1,n2_at1);
        cRLUD_at1++;
}
else{
        noD_at1 = 0;
        disD_at1 = -1;
}

/********* ordenar os nos por distancias ao no destino      ************/


    disde_at1[0][0] = disR_at1;
    disde_at1[0][1] = 1;        /*** 1=R, 3=L, 2=U, 4=D  ****/
    disde_at1[1][0] = disL_at1;
    disde_at1[1][1] = 3;
    disde_at1[2][0] = disU_at1;
    disde_at1[2][1] = 2;
    disde_at1[3][0] = disD_at1;
    disde_at1[3][1] = 4;


    for (if_at1 = 0; if_at1 < 4; if_at1 ++){
        if (disde_at1 [if_at1][0] == -1){
            array_ord [if_at1][0] = 1000;
        }
        else{
            array_ord [if_at1][0] = disde_at1 [if_at1][0];
        }
        array_ord [if_at1][1] = disde_at1 [if_at1][1];
    }
    array_ord [4][0] = 100000;
    array_ord [4][1] = 5;

    f_ordem ();
    f_ordigu ();

    for (if_at1 = 0; if_at1 < 4; if_at1 ++){
        if (array_ord [if_at1][0] == 1000){
            disde_at1 [if_at1][0] = -1;
        }
        else {
            disde_at1 [if_at1][0] = array_ord [if_at1][0];
        }
        disde_at1 [if_at1][1] = array_ord [if_at1][1];
    }

    cont1_at1 = 0;
    for (if_at1 = 0; if_at1 < 4; if_at1++){
        if (disde_at1[if_at1][0] == -1){
            cont1_at1++;
        }
    }

/****** determinacao da direccao de saida da rede ******************/
    if (mdxy2_at1 <= mod0_at1){
        Mov_Sol2 = 'S';
        tipo_sol = 10;
    }
    else if (mdxy2_at1 > mod0_at1){
        agxy2_at1 = angxy(vsaix_at1,vsaiy_at1);

        if ((agxy2_at1 <= angu1at1) && (agxy2_at1 > angu2at1)){
            Mov_Sol2 = 'R';
            tipo_sol = 1;
            if (tr_noat1[0][1] != 0 || Mov_Sol2 == CharDir || TRADIS_R[i1_at1][i2_at1] > 0){
                if (vsaiy_at1 < 0.000){
                    if (tr_noat1 [3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'D';
                        tipo_sol = 2;
    }
    else if (tr_noat1 [3][1] !=0 || CharDir == 'D' || TRADIS_D[i1_at1][i2_at1] > 0){
        if(tr_noat1[2][1] == 0 && CharDir != 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
            Mov_Sol2 = 'U';
            tipo_sol = 3;
        }
        else if (tr_noat1[1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
            Mov_Sol2 = 'L';
            tipo_sol = 4;
        }
        else {
            Mov_Sol2 = CharDir;    /**anda para tras **/
            tipo_sol = 5;
            if (Mov_Sol2 == 'R'){
                tipo_sol = 1;
            }
        }
    }
}
if (vsaiy_at1 > 0.000){
    if (tr_noat1 [2][1] == 0 && CharDir != 'U'&& TRADIS_U[i1_at1][i2_at1] == 0){
        Mov_Sol2 = 'U';
```

```
                    tipo_sol = 2;
                }
                else if (tr_noat1 [2][1] !=0 || CharDir == 'U' || TRADIS_U[i1_at1][i2_at1] > 0){
                    if(tr_noat1[3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'D';
                        tipo_sol = 3;
                    }
                    else if (tr_noat1[1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'L';
                        tipo_sol = 4;
                    }
                    else {
                        Mov_Sol2 = CharDir;   /**anda para tras **/
                        tipo_sol = 5;
                        if (Mov_Sol2 == 'R'){
                            tipo_sol = 1;
                        }
                    }
                }
            }
        }
    }
    else if ((agxy2_at1 <= angu3at1) && (agxy2_at1 > angu1at1)){
        Mov_Sol2 = 'U';
        tipo_sol = 1;
        if (tr_noat1[2][1] != 0 || Mov_Sol2 == CharDir || TRADIS_U[i1_at1][i2_at1] > 0){
            if (vsaix_at1 < 0.000){
                if (tr_noat1 [1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
                    Mov_Sol2 = 'L';
                    tipo_sol = 2;
                }
                else if (tr_noat1 [1][1] !=0 || CharDir == 'L' || TRADIS_L[i1_at1][i2_at1] > 0){
                    if(tr_noat1[0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'R';
                        tipo_sol = 3;
                    }
                    else if (tr_noat1[3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'D';
                        tipo_sol = 4;
                    }
                    else {
                        Mov_Sol2 = CharDir;   /**anda para tras **/
                        tipo_sol = 5;
                        if (Mov_Sol2 == 'U'){
                            tipo_sol = 1;
                        }
                    }
                }
            }
            if (vsaix_at1 > 0.000){
                if (tr_noat1 [0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                    Mov_Sol2 = 'R';
                    tipo_sol = 2;
                }
                else if (tr_noat1 [0][1] !=0 || CharDir == 'R' || TRADIS_R[i1_at1][i2_at1] > 0){
                    if(tr_noat1[1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'L';
                        tipo_sol = 3;
                    }
                    else if (tr_noat1[3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'D';
                        tipo_sol = 4;
                    }
                    else {
                        Mov_Sol2 = CharDir;   /**anda para tras **/
                        tipo_sol = 5;
                        if (Mov_Sol2 == 'U'){
                            tipo_sol = 1;
                        }
                    }
                }
            }
        }
    }
    else if ((agxy2_at1 > angu3at1) || (agxy2_at1 <= angu4at1)){
        Mov_Sol2 = 'L';
        tipo_sol = 1;
        if (tr_noat1[1][1] != 0 || Mov_Sol2 == CharDir || TRADIS_L[i1_at1][i2_at1] > 0){
            if (vsaiy_at1 < 0.000){
                if (tr_noat1 [3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                    Mov_Sol2 = 'D';
                    tipo_sol = 2;
                }
                else if (tr_noat1 [3][1] !=0 || CharDir == 'D' || TRADIS_D[i1_at1][i2_at1] > 0 ){
                    if(tr_noat1[2][1] == 0 && CharDir != 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'U';
                        tipo_sol = 3;
                    }
                    else if (tr_noat1[0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'R';
                        tipo_sol = 4;
                    }
                    else {
                        Mov_Sol2 = CharDir;   /**anda para tras **/
                        tipo_sol = 5;
                        if (Mov_Sol2 == 'L'){
                            tipo_sol = 1;
                        }
                    }
                }
            }
            if (vsaiy_at1 > 0.000){
                if (tr_noat1 [2][1] == 0 && CharDir != 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
                    Mov_Sol2 = 'U';
                    tipo_sol = 2;
                }
                else if (tr_noat1 [2][1] !=0 || CharDir == 'U' || TRADIS_U[i1_at1][i2_at1] > 0){
                    if(tr_noat1[3][1] == 0 && CharDir != 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'D';
```

```
                        tipo_sol = 3;
                    }
                    else if (tr_noat1[0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'R';
                        tipo_sol = 4;
                    }
                    else {
                        Mov_Sol2 = CharDir;   /**anda para tras **/
                        tipo_sol = 5;
                        if (Mov_Sol2 == 'L'){
                              tipo_sol = 1;
                        }
                    }
                }
            }
        }
    }
    else if ((agxy2_at1 > angu4at1) && (agxy2_at1 <= angu2at1)){
        Mov_Sol2 = 'D';
        tipo_sol = 1;
        if (tr_noat1[3][1] != 0 || Mov_Sol2 == CharDir || TRADIS_D[i1_at1][i2_at1] > 0){
              if (vsaix_at1 < 0.000){
                  if (tr_noat1 [1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'L';
                        tipo_sol = 2;
                  }
                  else if (tr_noat1 [1][1] !=0 || CharDir == 'L' || TRADIS_L[i1_at1][i2_at1] > 0){
                        if(tr_noat1[0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                            Mov_Sol2 = 'R';
                            tipo_sol = 3;
                        }
                        else if (tr_noat1[2][1] == 0 && CharDir != 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
                            Mov_Sol2 = 'U';
                            tipo_sol = 4;
                        }
                        else {
                            Mov_Sol2 = CharDir;   /**anda para tras **/
                            tipo_sol = 5;
                            if (Mov_Sol2 == 'D'){
                                  tipo_sol = 1;
                            }
                        }
                  }
              }
              if (vsaix_at1 > 0.000){
                  if (tr_noat1 [0][1] == 0 && CharDir != 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
                        Mov_Sol2 = 'R';
                        tipo_sol = 2;
                  }
                  else if (tr_noat1 [0][1] !=0 || CharDir == 'R' || TRADIS_R[i1_at1][i2_at1] > 0){
                        if(tr_noat1[1][1] == 0 && CharDir != 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
                            Mov_Sol2 = 'L';
                            tipo_sol = 3;
                        }
                        else if (tr_noat1[2][1] == 0 && CharDir != 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
                            Mov_Sol2 = 'U';
                            tipo_sol = 4;
                        }
                        else {
                            Mov_Sol2 = CharDir;   /**anda para tras **/
                            tipo_sol = 5;
                            if (Mov_Sol2 == 'D'){
                                  tipo_sol = 1;
                            }
                        }
                  }
              }
              else{
                  Mov_Sol2 = 'E';
                  tipo_sol = -1;
/*                PutStr ("error: Mov_Sol2 (E)");
                  GetStr();        */
              }
        }
        else {
              Mov_Sol2 ='e';
              tipo_sol = -2;
/*            PutStr ("error: Mov_Sol2 (e)");
              GetStr();    */
        }
/*****************************************************************/
/****  actualizacao da variavel global
  registo[no actual, dir.saida, num.tramo seg., est.tramo, no seguinte]**/
for (if_at1=0;if_at1<5;if_at1++){
      registo [if_at1] = 0;
}

if (Mov_Sol2 == 'R' && TRADIS_R[i1_at1][i2_at1] == 0){
      ndir_at1 = 1;
      trseg_at1 = tr_noat1 [0][0];
      trest_at1 = tr_noat1 [0][1];
      if (trest_at1 == 0){
          noseg_at1 = MATNOS[i1_at1][i2_at1 + 1];
          fsaip_upd (n1_at1, MATR_R [i1_at1][i2_at1]);
      }
      else{
          noseg_at1 = n1_at1;
      }
}
else if (Mov_Sol2 == 'L' && TRADIS_L[i1_at1][i2_at1] == 0){
      ndir_at1 = 3;
      trseg_at1 = tr_noat1 [1][0];
      trest_at1 = tr_noat1 [1][1];
      if (trest_at1 == 0){
          noseg_at1 = MATNOS[i1_at1][i2_at1 - 1];
          fsaip_upd (n1_at1, MATR_L [i1_at1][i2_at1]);
```

```
        }
        else{
            noseg_at1 = n1_at1;
        }
}
else if (Mov_Sol2 == 'U' && TRADIS_U[i1_at1][i2_at1] == 0){
        ndir_at1 = 2;
        trseg_at1 = tr_noat1 [2][0];
        trest_at1 = tr_noat1 [2][1];
        if (trest_at1 == 0){
            noseg_at1 = MATNOS[i1_at1 - 1][i2_at1];
            fsaip_upd (n1_at1, MATR_U [i1_at1][i2_at1]);

        }
        else{
            noseg_at1 = n1_at1;
        }
}
else if (Mov_Sol2 == 'D' && TRADIS_D[i1_at1][i2_at1] == 0){
        ndir_at1 = 4;
        trseg_at1 = tr_noat1 [3][0];
        trest_at1 = tr_noat1 [3][1];
        if (trest_at1 == 0){
            noseg_at1 = MATNOS[i1_at1 + 1][i2_at1];
            fsaip_upd (n1_at1, MATR_D [i1_at1][i2_at1]);

        }
        else{
            noseg_at1 = n1_at1;
        }
}
else if (Mov_Sol2 == 'S'){              /*** no sem saida *****/

        if (cont1_at1 == 4){
            ndir_at1 = 0;
            trseg_at1 = 0;
            trest_at1 = -3;
            noseg_at1 = n1_at1;
            tipo_sol = 10;
        }
        else if(cont1_at1 < 4){
                    ndir_at1 = disde_at1 [cont1_at1][1];
                    tipo_sol = 11;
                    if (disde_at1[cont1_at1][1] ==1 && TRADIS_R [i1_at1][i2_at1] == 0 && noR_at1 != 0){
                        trseg_at1 = tr_noat1 [0][0];
                        trest_at1 = tr_noat1 [0][1];
                        noseg_at1 = noR_at1;
                        fsaip_upd (n1_at1, MATR_R [i1_at1][i2_at1]);
                    }
                    else if (disde_at1[cont1_at1][1] == 3 && TRADIS_L [i1_at1][i2_at1] == 0 && noL_at1 != 0){
                        trseg_at1 = tr_noat1 [1][0];
                        trest_at1 = tr_noat1 [1][1];
                        noseg_at1 = noL_at1;
                        fsaip_upd (n1_at1, MATR_L [i1_at1][i2_at1]);
                    }
                    else if (disde_at1[cont1_at1][1] == 2 && TRADIS_U [i1_at1][i2_at1] == 0 && noU_at1 != 0){
                        trseg_at1 = tr_noat1 [2][0];
                        trest_at1 = tr_noat1 [2][1];
                        noseg_at1 = noU_at1;
                        fsaip_upd (n1_at1, MATR_U [i1_at1][i2_at1]);
                    }
                    else if (disde_at1[cont1_at1][1] == 4 && TRADIS_D [i1_at1][i2_at1] == 0 && noD_at1 != 0){
                        trseg_at1 = tr_noat1 [3][0];
                        trest_at1 = tr_noat1 [3][1];
                        noseg_at1 = noD_at1;
                        fsaip_upd (n1_at1, MATR_D [i1_at1][i2_at1]);
                    }
                    else {
                        if ( cont1_at1 + 1 < 4){
                            cont1_at1 = cont1_at1 + 1;
                            ndir_at1 = disde_at1 [cont1_at1][1];
                            tipo_sol = 12;
                            if (disde_at1[cont1_at1][1] ==1 && TRADIS_R [i1_at1][i2_at1] == 0 && noR_at1 != 0){
                                trseg_at1 = tr_noat1 [0][0];
                                trest_at1 = tr_noat1 [0][1];
                                noseg_at1 = noR_at1;
                                fsaip_upd (n1_at1, MATR_R [i1_at1][i2_at1]);
                            }
                            else if (disde_at1[cont1_at1][1] == 3 && TRADIS_L [i1_at1][i2_at1] == 0 && noL_at1
!= 0){
                                trseg_at1 = tr_noat1 [1][0];
                                trest_at1 = tr_noat1 [1][1];
                                noseg_at1 = noL_at1;
                                fsaip_upd (n1_at1, MATR_L [i1_at1][i2_at1]);
                            }
                            else if (disde_at1[cont1_at1][1] == 2 && TRADIS_U [i1_at1][i2_at1] == 0 && noU_at1
!= 0){
                                trseg_at1 = tr_noat1 [2][0];
                                trest_at1 = tr_noat1 [2][1];
                                noseg_at1 = noU_at1;
                                fsaip_upd (n1_at1, MATR_U [i1_at1][i2_at1]);
                            }
                            else if (disde_at1[cont1_at1][1] == 4 && TRADIS_D [i1_at1][i2_at1] == 0 && noD_at1
!= 0){
                                trseg_at1 = tr_noat1 [3][0];
                                trest_at1 = tr_noat1 [3][1];
                                noseg_at1 = noD_at1;
                                fsaip_upd (n1_at1, MATR_D [i1_at1][i2_at1]);

                            }
                            else{
                             if ( cont1_at1 + 1 < 4){
                                cont1_at1 = cont1_at1 + 1;
                                ndir_at1 = disde_at1 [cont1_at1][1];
                                tipo_sol = 13;
                                if (disde_at1[cont1_at1][1] ==1 && TRADIS_R [i1_at1][i2_at1] == 0 && noR_at1 !=
0){
                                    trseg_at1 = tr_noat1 [0][0];
                                    trest_at1 = tr_noat1 [0][1];
```

```
                                        noseg_at1 = noR_at1;
                                        fsaip_upd (n1_at1, MATR_R [i1_at1][i2_at1]);
                                    }
                                    else if (disde_at1[cont1_at1][1] == 3 && TRADIS_L [i1_at1][i2_at1] == 0 &&
noL_at1 != 0){
                                        trseg_at1 = tr_noat1 [1][0];
                                        trest_at1 = tr_noat1 [1][1];
                                        noseg_at1 = noL_at1;
                                        fsaip_upd (n1_at1, MATR_L [i1_at1][i2_at1]);
                                    }
                                    else if (disde_at1[cont1_at1][1] == 2 && TRADIS_U [i1_at1][i2_at1] == 0 &&
noU_at1 != 0){
                                        trseg_at1 = tr_noat1 [2][0];
                                        trest_at1 = tr_noat1 [2][1];
                                        noseg_at1 = noU_at1;
                                        fsaip_upd (n1_at1, MATR_U [i1_at1][i2_at1]);
                                    }
                                    else if (disde_at1[cont1_at1][1] == 4 && TRADIS_D [i1_at1][i2_at1] == 0 &&
noD_at1 != 0){
                                        trseg_at1 = tr_noat1 [3][0];
                                        trest_at1 = tr_noat1 [3][1];
                                        noseg_at1 = noD_at1;
                                        fsaip_upd (n1_at1, MATR_D [i1_at1][i2_at1]);
                                    }
                                    else{
                                    if ( cont1_at1 + 1 < 4){
                                        cont1_at1 = cont1_at1 + 1;
                                        ndir_at1 = disde_at1 [cont1_at1][1];
                                        tipo_sol = 14;
                                        if (disde_at1[cont1_at1][1] ==1 && TRADIS_R [i1_at1][i2_at1] == 0 &&
noR_at1 != 0){
                                            trseg_at1 = tr_noat1 [0][0];
                                            trest_at1 = tr_noat1 [0][1];
                                            noseg_at1 = noR_at1;
                                            fsaip_upd (n1_at1, MATR_R [i1_at1][i2_at1]);
                                        }
                                        else if (disde_at1[cont1_at1][1] == 3 && TRADIS_L [i1_at1][i2_at1] ==
0 && noL_at1 != 0){
                                            trseg_at1 = tr_noat1 [1][0];
                                            trest_at1 = tr_noat1 [1][1];
                                            noseg_at1 = noL_at1;
                                            fsaip_upd (n1_at1, MATR_L [i1_at1][i2_at1]);
                                        }
                                        else if (disde_at1[cont1_at1][1] == 2 && TRADIS_U [i1_at1][i2_at1] ==
0 && noU_at1 != 0){
                                            trseg_at1 = tr_noat1 [2][0];
                                            trest_at1 = tr_noat1 [2][1];
                                            noseg_at1 = noU_at1;
                                            fsaip_upd (n1_at1, MATR_U [i1_at1][i2_at1]);
                                        }
                                        else if (disde_at1[cont1_at1][1] == 4 && TRADIS_D [i1_at1][i2_at1] ==
0 && noD_at1 != 0){
                                            trseg_at1 = tr_noat1 [3][0];
                                            trest_at1 = tr_noat1 [3][1];
                                            noseg_at1 = noD_at1;
                                            fsaip_upd (n1_at1, MATR_D [i1_at1][i2_at1]);
                                        }
                                        else{
                                            trseg_at1 = 0;
                                            trest_at1 = -3;
                                            noseg_at1= n1_at1;
                                            ndir_at1 = 0;
                                            tipo_sol = 15;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
            else{
                tipo_sol = 16;
            }
    }
    else{
/*          PutStr ("No RLUDS possible");  */

            trseg_at1 = 0;
            trest_at1 = -3;
            noseg_at1= n1_at1;
            ndir_at1 = 0;
            tipo_sol = 17;
    }
    Fim_evastra :
    registo [0] = n1_at1;
    registo [1] = ndir_at1;  /** 0=S, 1=R, 3=L, 2=U, 4=D    ******/
    registo [2] = trseg_at1;
    registo [3] = trest_at1;
    registo [4] = noseg_at1;

/*    sprintf (buflixo,"reg[0...4]:%d\t %d\t %d\t %d\t %d\t
tipoSol=%d\n",registo[0],registo[1],registo[2],registo[3],registo[4],tipo_sol);
    PutStr (buflixo);
    splixo = GetStr();          */

}


/*********************************************************/
/* funcao para gerar um numero inteiro pseudo_aleatorio entre 2 numeros
especificados */

int f_randoes (int min_ele, int max_ele)
{   int ele_1, ele_2, randin;

    ele_1 = min_ele;
    ele_2 = max_ele;
```

```
    randin = rand()%(ele_2 +1 - ele_1) + ele_1;

    return (randin);

}

/*********************************************************/
/* funcao para ordenar por ordem crescente os elementos no array_ord[][]
de acordo com a grandeza do valor em array_ord[][0]  */
void f_ordem ()
{   int if_ord, jf_ord, ic1_ord;
    long int maxo_ord, lixo_ord [5][2];

    for (if_ord = 0; if_ord < 5; if_ord ++){
          for (jf_ord = 0; jf_ord < 2; jf_ord ++){
              lixo_ord [if_ord][jf_ord] = 0;
          }
    }

    ic1_ord = -1;
    maxo_ord = 99999;
    for (if_ord = 0; if_ord < 5; if_ord ++){
          if (array_ord [if_ord][0] < maxo_ord){
              maxo_ord = array_ord [if_ord][0];
              ic1_ord = if_ord;
          }
    }
    if (ic1_ord >= 0){
          lixo_ord [0][0] = array_ord [ic1_ord][0];
          lixo_ord [0][1] = array_ord [ic1_ord][1];
          array_ord [ic1_ord][0] = 100000;
    }
    ic1_ord = -1;
    maxo_ord = 99999;
    for (if_ord = 0; if_ord < 5; if_ord ++){
          if (array_ord [if_ord][0] < maxo_ord){
              maxo_ord = array_ord [if_ord][0];
              ic1_ord = if_ord;
          }
    }
    if (ic1_ord >= 0){
          lixo_ord [1][0] = array_ord [ic1_ord][0];
          lixo_ord [1][1] = array_ord [ic1_ord][1];
          array_ord [ic1_ord][0] = 100000;
    }
    ic1_ord = -1;
    maxo_ord = 99999;
    for (if_ord = 0; if_ord < 5; if_ord ++){
          if (array_ord [if_ord][0] < maxo_ord){
              maxo_ord = array_ord [if_ord][0];
              ic1_ord = if_ord;
          }
    }
    if (ic1_ord >= 0){
          lixo_ord [2][0] = array_ord [ic1_ord][0];
          lixo_ord [2][1] = array_ord [ic1_ord][1];
          array_ord [ic1_ord][0] = 100000;
    }
    ic1_ord = -1;
    maxo_ord = 99999;
    for (if_ord = 0; if_ord < 5; if_ord ++){
          if (array_ord [if_ord][0] < maxo_ord){
              maxo_ord = array_ord [if_ord][0];
              ic1_ord = if_ord;
          }
    }
    if (ic1_ord >= 0){
          lixo_ord [3][0] = array_ord [ic1_ord][0];
          lixo_ord [3][1] = array_ord [ic1_ord][1];
          array_ord [ic1_ord][0] = 100000;
    }
    ic1_ord = -1;
    maxo_ord = 99999;
    for (if_ord = 0; if_ord < 5; if_ord ++){
          if (array_ord [if_ord][0] < maxo_ord){
              maxo_ord = array_ord [if_ord][0];
              ic1_ord = if_ord;
          }
    }
    if (ic1_ord >= 0){
          lixo_ord [4][0] = array_ord [ic1_ord][0];
          lixo_ord [4][1] = array_ord [ic1_ord][1];
          array_ord [ic1_ord][0] = 100000;
    }
    ic1_ord = -1;

    for (if_ord = 0; if_ord < 5; if_ord++){
          for (jf_ord = 0; jf_ord < 2; jf_ord++){
              array_ord [if_ord][jf_ord] = lixo_ord [if_ord][jf_ord];
          }
    }

  /*   na funcao f_ordigu() sao resolvidas as situacoes de
 elementos com o mesmo valor  */

}

/* funcao para verificar e resolver casos em que haja mais do que
um elemento com o mesmo valor em array_ord [][0], ja com os seus elementos
ordenados por ordem crescente */
void f_ordigu ()
{   int iguCnt_1,iguCnt_2,iguCnt_3,iguCnt_4,iguCnt_5,if_igu,jf_igu;
    int i1_igu, i2_igu, i3_igu, i4_igu, ili_igu;
    long int ord2_igu [5][2], tli_igu;

    for (if_igu = 0; if_igu < 5; if_igu ++){
          for (jf_igu = 0; jf_igu < 2; jf_igu ++){
              ord2_igu[if_igu][jf_igu] = array_ord [if_igu][jf_igu];
```

```
        }
}

tli_igu = ord2_igu [0][0];
iguCnt_1 = 0;
for (if_igu = 0; if_igu < 5; if_igu ++){
        if (ord2_igu [if_igu][0] == tli_igu){
            iguCnt_1 ++;
        }
}
tli_igu = ord2_igu [1][0];
iguCnt_2 = 0;
if (iguCnt_1 > 1){
        i1_igu = 5;
}
else{
        i1_igu = iguCnt_1;
}
for (if_igu = i1_igu; if_igu < 5; if_igu ++){
        if (ord2_igu [if_igu][0] == tli_igu){
            iguCnt_2 ++;
        }
}
tli_igu = ord2_igu [2][0];
iguCnt_3 = 0;
if (iguCnt_1 > 2 || iguCnt_2 > 1){
        i2_igu = 5;
}
else{
        i2_igu = iguCnt_2+iguCnt_1;
}
for (if_igu = i2_igu; if_igu < 5; if_igu ++){
        if (ord2_igu [if_igu][0] == tli_igu){
            iguCnt_3 ++;
        }
}
tli_igu = ord2_igu [3][0];
iguCnt_4 = 0;
if (iguCnt_1 > 3 || iguCnt_2 > 2 || iguCnt_3 > 1){
        i3_igu = 5;
}
else{
        i3_igu = iguCnt_3 + iguCnt_2 + iguCnt_1;
}
for (if_igu = i3_igu; if_igu < 5; if_igu ++){
        if (ord2_igu [if_igu][0] == tli_igu){
            iguCnt_4 ++;
        }
}
tli_igu = ord2_igu [4][0];
iguCnt_5 = 0;
if (iguCnt_1 > 4 || iguCnt_2 > 3 || iguCnt_3 > 2 || iguCnt_4 > 1){
        i4_igu = 5;
}
else{
        i4_igu = iguCnt_4 + iguCnt_3 + iguCnt_2 + iguCnt_1;
}
for (if_igu = i4_igu; if_igu < 5; if_igu ++){
        if (ord2_igu [if_igu][0] == tli_igu){
            iguCnt_5 ++;
        }
}

/* todos os cinco elementos sÆo iguais */
if (iguCnt_1 == 5){
        f_aleaord (5);
        for (if_igu = 0; if_igu < 5; if_igu ++){
            array_ord [if_igu][0] = ord2_igu [temp_seq[if_igu]-1][0];
            array_ord [if_igu][1] = ord2_igu [temp_seq[if_igu]-1][1];
        }
}

/* quatro dos cinco elementos sao iguais */
if (iguCnt_1 == 4 || iguCnt_2 == 4){
        f_aleaord (4);
        if (iguCnt_1 == 4){
            ili_igu = 0;
        }
        else if (iguCnt_2 == 4){
            ili_igu = 1;
        }
        for (if_igu = ili_igu; if_igu < (ili_igu + 4); if_igu ++){
            array_ord [if_igu][0] = ord2_igu [ili_igu+temp_seq[if_igu - ili_igu]-1][0];
            array_ord [if_igu][1] = ord2_igu [ili_igu+temp_seq[if_igu - ili_igu]-1][1];
        }
}

/* tres dos cinco elementos sao iguais */
if (iguCnt_1 == 3 || iguCnt_2 == 3 || iguCnt_3 == 3){
        f_aleaord (3);
        if (iguCnt_1 == 3){
            ili_igu = 0;
        }
        else if (iguCnt_2 == 3){
            ili_igu = 1;
        }
        else if (iguCnt_3 == 3){
            ili_igu = 2;
        }
        for (if_igu = ili_igu; if_igu < (ili_igu + 3); if_igu ++){
            array_ord [if_igu][0] = ord2_igu [ili_igu+temp_seq[if_igu - ili_igu]-1][0];
            array_ord [if_igu][1] = ord2_igu [ili_igu+temp_seq[if_igu - ili_igu]-1][1];
        }
}

/* dois dos cinco elementos sao iguais */
if (iguCnt_1 == 2){
        f_aleaord (2);
        for (if_igu = 0; if_igu < 2; if_igu ++){
```

```
            array_ord [if_igu][0] = ord2_igu [temp_seq[if_igu]-1][0];
            array_ord [if_igu][1] = ord2_igu [temp_seq[if_igu]-1][1];
        }
}
if (iguCnt_2 == 2){
        f_aleaord (2);
        for (if_igu = 1; if_igu < 3; if_igu ++){
            array_ord [if_igu][0] = ord2_igu [1+temp_seq[if_igu-1]-1][0];
            array_ord [if_igu][1] = ord2_igu [1+temp_seq[if_igu-1]-1][1];
        }
}
if (iguCnt_3 == 2){
        f_aleaord (2);
        for (if_igu = 2; if_igu < 4; if_igu ++){
            array_ord [if_igu][0] = ord2_igu [2+temp_seq[if_igu-2]-1][0];
            array_ord [if_igu][1] = ord2_igu [2+temp_seq[if_igu-2]-1][1];
        }
}
if (iguCnt_4 == 2){
        f_aleaord (2);
        for (if_igu = 3; if_igu < 5; if_igu ++){
            array_ord [if_igu][0] = ord2_igu [3+temp_seq[if_igu-3]-1][0];
            array_ord [if_igu][1] = ord2_igu [3+temp_seq[if_igu-3]-1][1];
        }
}

}

/* funcao para atribuir conjuntos de numeros aleatorios todos !=s */
void f_aleaord (int num_aleas)
{   int nig_aord, alea_1, alea_2, alea_3, alea_4, alea_5, if_aord, ili_aord;

    nig_aord = num_aleas;

    for (if_aord = 0; if_aord < 5; if_aord++){
        if (if_aord < nig_aord){
            temp_seq [if_aord] = if_aord+1;
        }
        else{
            temp_seq [if_aord] = 0;
        }
    }

    if (nig_aord > 1){
        alea_1 = f_randoes(1,nig_aord);
        ili_aord = temp_seq [alea_1-1];
        for (if_aord = alea_1; if_aord < nig_aord; if_aord ++){
            temp_seq [if_aord-1] = temp_seq [if_aord];
        }
        temp_seq [nig_aord-1] = ili_aord;
    }
    if (nig_aord > 2){
        alea_2 = f_randoes(1,nig_aord-1);
        ili_aord = temp_seq [alea_2-1];
        for (if_aord = alea_2; if_aord < (nig_aord-1); if_aord ++){
            temp_seq [if_aord-1] = temp_seq [if_aord];
        }
        temp_seq [nig_aord-2] = ili_aord ;
    }
    if (nig_aord > 3){
        alea_3 = f_randoes(1,nig_aord-2);
        ili_aord = temp_seq [alea_3-1];
        for (if_aord = alea_3; if_aord < (nig_aord-2); if_aord ++){
            temp_seq [if_aord-1] = temp_seq [if_aord];
        }
        temp_seq [nig_aord-3] = ili_aord ;
    }
    if (nig_aord > 4){
        alea_4 = f_randoes(1,nig_aord-3);
        ili_aord = temp_seq [alea_4-1];
        for (if_aord = alea_4; if_aord < (nig_aord-3); if_aord ++){
            temp_seq [if_aord-1] = temp_seq [if_aord];
        }
        temp_seq [nig_aord-4] = ili_aord ;
    }

}

/* funcoes auxiliares para testar as implicacoes do movimento
de um veiculo! A usar na estrategia step-ahead. Os valores dos
argumentos jafM....jdfM cooresponden aos indices dos ciclos exteriores
ao ciclo em an lise */

/* funcao para testar NO2(v5) */
int f_NO2M5 (int jdfM5)
{   int jdM5M, i_5M, j_5M, p_5M, q_5M;
int mov3_5M, mov4_5M, for5M;

jdM5M = jdfM5;
mov4_5M = 0;
mov3_5M = 0;

i_5M = indexi(melNO2 [tpo][jdM5M]);
j_5M = indexj(melNO2 [tpo][jdM5M]);
/* existe um veiculo no NO2(v5) ? */
mov4_5M = 0;
for (for5M = 0; for5M<Num_VeiT; for5M++){
    if (for5M != jdM5M){
        if (((MATNOS[i_5M][j_5M]==melNO1[tpo][for5M])&&(melNO2[tpo][for5M]>0))||(NOS_AHEAD[i_5M][j_5M]<-1)){
            mov4_5M = 1;
        }
    }
}
if (mov4_5M == 0){
    for (for5M = 0; for5M < Num_VeiT; for5M++){
        if ((MATNOS[i_5M][j_5M]==melNO2[tpo][for5M])&&(for5M!=jdM5M)){
            p_5M = indexi (melNO1 [tpo][for5M]);
            q_5M = indexj (melNO1 [tpo][for5M]);
            melNO1 [tpo+1][for5M] = MATNOS [p_5M][q_5M];
```

```
                    melNO2 [tpo][for5M] = -112;
                    NOS_AHEAD [p_5M][q_5M] = -112;
                }
            }
            melNO1 [tpo+1][jdM5M] = MATNOS [i_5M][j_5M];
            melNO2 [tpo][jdM5M] = -112;
            NOS_AHEAD [i_5M][j_5M] = -112;
            mov3_5M = 0;
    }
    else if (mov4_5M == 1){
            mov3_5M = 1;
    }
    mov4_5M = 0;
    return mov3_5M;
}



/* funcao para testar NO2 em look-ahead limitado */
int f_NO2M_1A ()
{   int i_AM, j_AM, p_AM, q_AM, mov9_AM, movAM_AM, forAM, f1_AM, flag_AM;

    mov9_AM = 0;
    flag_AM = 0;

    i_AM = indexi(melNO2 [tpo][VEI[1]]);
    j_AM = indexj(melNO2 [tpo][VEI[1]]);
    /* existe um veiculo no NO2(vL) ? */
    movAM_AM = 0;
    for (forAM = 0; forAM<Num_VeiT; forAM++){
          if (forAM != VEI[1]){
              if (((MATNOS[i_AM][j_AM]==melNO1[tpo][forAM])&&(melNO2[tpo][forAM]>0))||(NOS_AHEAD[i_AM][j_AM]<-1)){
                    flag_AM = 0;
                    for (f1_AM = 0; f1_AM < Num_VeiT; f1_AM++){
                        if (forAM == VEI [f1_AM]){
                              flag_AM = 1;
                              if (f1_AM == 0){
                                  flag_AM = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_AM][j_AM]<-1) || (flag_AM == 1)){
                        movAM_AM = 1;
                    }
                    else {
                        movAM_AM = 0;
                    }
                    flag_AM = 0;
              }
          }
    }
    if (movAM_AM == 0){
          for (forAM = 0; forAM < Num_VeiT; forAM++){
              if (MATNOS[i_AM][j_AM]==melNO2[tpo][forAM]){
                    flag_AM = 0;
                    for (f1_AM = 0; f1_AM < Num_VeiT; f1_AM++){
                        if (forAM == VEI [f1_AM]){
                              flag_AM = 1;
                        }
                    }
                    if (flag_AM != 1){
                        p_AM = indexi (melNO1 [tpo][forAM]);
                        q_AM = indexj (melNO1 [tpo][forAM]);
                        melNO1 [tpo+1][forAM] = MATNOS [p_AM][q_AM];
                        melNO2 [tpo][forAM] = -112;
                        NOS_AHEAD [p_AM][q_AM] = -112;
                    }
                    flag_AM = 0;
              }
          }
          melNO1 [tpo+1][VEI[1]] = MATNOS [i_AM][j_AM];
          melNO2 [tpo][VEI[1]] = -112;
          NOS_AHEAD [i_AM][j_AM] = -112;
          mov9_AM = 0;
    }
    else if (movAM_AM == 1){
          mov9_AM = 1;
    }
    movAM_AM = 0;
    return mov9_AM;
}

/* funcao para testar NO2( [f_NO2M_10] ultimo veiculo, max. 11§) */
int f_NO2M_LAST ()
{   int i_LM, j_LM, p_LM, q_LM, mov9_LM, movLM_LM, forLM, f1_LM, flag_LM;

    mov9_LM = 0;
    flag_LM = 0;

    i_LM = indexi(melNO2 [tpo][VEI[10]]);
    j_LM = indexj(melNO2 [tpo][VEI[10]]);
    /* existe um veiculo no NO2(vL) ? */
    movLM_LM = 0;
    for (forLM = 0; forLM<Num_VeiT; forLM++){
          if (forLM != VEI[10]){
              if (((MATNOS[i_LM][j_LM]==melNO1[tpo][forLM])&&(melNO2[tpo][forLM]>0))||(NOS_AHEAD[i_LM][j_LM]<-1)){
                    flag_LM = 0;
                    for (f1_LM = 0; f1_LM < Num_VeiT; f1_LM++){
                        if (forLM == VEI [f1_LM]){
                              flag_LM = 1;
                              if (f1_LM == 0){
                                  flag_LM = -1;
                              }
                        }
                    }
```

```
                if ((NOS_AHEAD[i_LM][j_LM]<-1) || (flag_LM == 1)){
                    movLM_LM = 1;
                }
                else {
                    movLM_LM = 0;
                }
                flag_LM = 0;
            }
        }
}
if (movLM_LM == 0){
      for (forLM = 0; forLM < Num_VeiT; forLM++){
          if (MATNOS[i_LM][j_LM]==melNO2[tpo][forLM]){
                flag_LM = 0;
                for (f1_LM = 0; f1_LM < Num_VeiT; f1_LM++){
                    if (forLM == VEI [f1_LM]){
                          flag_LM = 1;
                    }
                }
                if (flag_LM != 1){
                    p_LM = indexi (melNO1 [tpo][forLM]);
                    q_LM = indexj (melNO1 [tpo][forLM]);
                    melNO1 [tpo+1][forLM] = MATNOS [p_LM][q_LM];
                    melNO2 [tpo][forLM] = -112;
                    NOS_AHEAD [p_LM][q_LM] = -112;
                }
                flag_LM = 0;
            }
          }
          melNO1 [tpo+1][VEI[10]] = MATNOS [i_LM][j_LM];
          melNO2 [tpo][VEI[10]] = -112;
          NOS_AHEAD [i_LM][j_LM] = -112;
          mov9_LM = 0;
    }
    else if (movLM_LM == 1){
          mov9_LM = 1;
    }
    movLM_LM = 0;
    return mov9_LM;
}

/* funcao para testar NO2 do 10§ veiculo */
int f_NO2M_9 ()
{   int i_9M, j_9M, p_9M, q_9M, mov9_9M, mov8_9M, for9M, f1_9M, flag_9M;

    mov8_9M = 0;
    flag_9M = 0;

    i_9M = indexi(melNO2 [tpo][VEI[9]]);
    j_9M = indexj(melNO2 [tpo][VEI[9]]);
    /* existe um veiculo no NO2(do veic.10) ? */
    mov9_9M = 0;
    for (for9M = 0; for9M < Num_VeiT; for9M ++){
          if (for9M != VEI[9]){
              if (((MATNOS[i_9M][j_9M]==melNO1[tpo][for9M])&&(melNO2[tpo][for9M]>0))||(NOS_AHEAD[i_9M][j_9M]<-1)){
                    flag_9M = 0;
                    for (f1_9M = 0; f1_9M < Num_VeiT; f1_9M++){
                        if (for9M == VEI [f1_9M]){
                              flag_9M = 1;
                              if (f1_9M == 0){
                                  flag_9M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_9M][j_9M]<-1) || (flag_9M == 1)){
                        mov9_9M = 1;
                    }
                    else if (flag_9M == -1){
                        mov9_9M = 0;
                    }
                    else {
                        VEI [10] = for9M;
                        mov9_9M = f_NO2M_LAST ();
                    }
                    flag_9M = 0;
              }
          }
}
if (mov9_9M == 0){
      for (for9M = 0; for9M < Num_VeiT; for9M++){
          if (MATNOS[i_9M][j_9M]==melNO2[tpo][for9M]){
                flag_9M = 0;
                for (f1_9M = 0; f1_9M < Num_VeiT; f1_9M++){
                    if (for9M == VEI [f1_9M]){
                          flag_9M = 1;
                    }
                }
                if (flag_9M != 1){
                    p_9M = indexi (melNO1 [tpo][for9M]);
                    q_9M = indexj (melNO1 [tpo][for9M]);
                    melNO1 [tpo+1][for9M] = MATNOS [p_9M][q_9M];
                    melNO2 [tpo][for9M] = -112;
                    NOS_AHEAD [p_9M][q_9M] = -112;
                }
                flag_9M = 0;
            }
          }
          melNO1 [tpo+1][VEI[9]] = MATNOS [i_9M][j_9M];
          melNO2 [tpo][VEI[9]] = -112;
          NOS_AHEAD [i_9M][j_9M] = -112;
          mov8_9M = 0;
    }
    else if (mov9_9M == 1){
          mov8_9M = 1;
    }
    mov9_9M = 0;
    return mov8_9M;
}
```

```
/* funcao para testar NO2 do 9$ veiculo */
int f_NO2M_8 ()
{   int i_8M, j_8M, p_8M, q_8M, mov8_8M, mov7_8M, for8M, f1_8M, flag_8M;

    mov7_8M = 0;
    flag_8M = 0;

    i_8M = indexi(melNO2 [tpo][VEI[8]]);
    j_8M = indexj(melNO2 [tpo][VEI[8]]);
    /* existe um veiculo no NO2(do veic.9) ? */
    mov8_8M = 0;
    for (for8M = 0; for8M < Num_VeiT; for8M ++){
          if (for8M != VEI[8]){
              if (((MATNOS[i_8M][j_8M]==melNO1[tpo][for8M])&&(melNO2[tpo][for8M]>0))||(NOS_AHEAD[i_8M][j_8M]<-1)){
                    flag_8M = 0;
                    for (f1_8M = 0; f1_8M < Num_VeiT; f1_8M++){
                        if (for8M == VEI [f1_8M]){
                              flag_8M = 1;
                              if (f1_8M == 0){
                                  flag_8M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_8M][j_8M]<-1) || (flag_8M == 1)){
                        mov8_8M = 1;
                    }
                    else if (flag_8M == -1){
                        mov8_8M = 0;
                    }
                    else {
                        VEI [9] = for8M;
                        mov8_8M = f_NO2M_9 ();
                    }
                    flag_8M = 0;
              }
          }
    }
    if (mov8_8M == 0){
          for (for8M = 0; for8M < Num_VeiT; for8M++){
              if (MATNOS[i_8M][j_8M]==melNO2[tpo][for8M]){
                    flag_8M = 0;
                    for (f1_8M = 0; f1_8M < Num_VeiT; f1_8M++){
                        if (for8M == VEI [f1_8M]){
                              flag_8M = 1;
                        }
                    }
                    if (flag_8M != 1){
                        p_8M = indexi (melNO1 [tpo][for8M]);
                        q_8M = indexj (melNO1 [tpo][for8M]);
                        melNO1 [tpo+1][for8M] = MATNOS [p_8M][q_8M];
                        melNO2 [tpo][for8M] = -112;
                        NOS_AHEAD [p_8M][q_8M] = -112;
                    }
                    flag_8M = 0;
              }
          }
          melNO1 [tpo+1][VEI[8]] = MATNOS [i_8M][j_8M];
          melNO2 [tpo][VEI[8]] = -112;
          NOS_AHEAD [i_8M][j_8M] = -112;
          mov7_8M = 0;
    }
    else if (mov8_8M == 1){
          mov7_8M = 1;
    }
    mov8_8M = 0;
    return mov7_8M;
}

/* funcao para testar NO2 do 8§ veiculo */
int f_NO2M_7 ()
{   int i_7M, j_7M, p_7M, q_7M, mov7_7M, mov6_7M, for7M, f1_7M, flag_7M;

    mov6_7M = 0;
    flag_7M = 0;

    i_7M = indexi(melNO2 [tpo][VEI[7]]);
    j_7M = indexj(melNO2 [tpo][VEI[7]]);
    /* existe um veiculo no NO2(do veic.8) ? */
    mov7_7M = 0;
    for (for7M = 0; for7M < Num_VeiT; for7M ++){
          if (for7M != VEI[7]){
              if (((MATNOS[i_7M][j_7M]==melNO1[tpo][for7M])&&(melNO2[tpo][for7M]>0))||(NOS_AHEAD[i_7M][j_7M]<-1)){
                    flag_7M = 0;
                    for (f1_7M = 0; f1_7M < Num_VeiT; f1_7M++){
                        if (for7M == VEI [f1_7M]){
                              flag_7M = 1;
                              if (f1_7M == 0){
                                  flag_7M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_7M][j_7M]<-1) || (flag_7M == 1)){
                        mov7_7M = 1;
                    }
                    else if (flag_7M == -1){
                        mov7_7M = 0;
                    }
                    else {
                        VEI [8] = for7M;
                        mov7_7M = f_NO2M_8 ();
                    }
                    flag_7M = 0;
              }
          }
    }
    if (mov7_7M == 0){
          for (for7M = 0; for7M < Num_VeiT; for7M++){
              if (MATNOS[i_7M][j_7M]==melNO2[tpo][for7M]){
                    flag_7M = 0;
```

```
                for (f1_7M = 0; f1_7M < Num_VeiT; f1_7M++){
                    if (for7M == VEI [f1_7M]){
                          flag_7M = 1;
                    }
                }
                if (flag_7M != 1){
                    p_7M = indexi (melNO1 [tpo][for7M]);
                    q_7M = indexj (melNO1 [tpo][for7M]);
                    melNO1 [tpo+1][for7M] = MATNOS [p_7M][q_7M];
                    melNO2 [tpo][for7M] = -112;
                    NOS_AHEAD [p_7M][q_7M] = -112;
                }
                flag_7M = 0;
            }
        }
        melNO1 [tpo+1][VEI[7]] = MATNOS [i_7M][j_7M];
        melNO2 [tpo][VEI[7]] = -112;
        NOS_AHEAD [i_7M][j_7M] = -112;
        mov6_7M = 0;
}
else if (mov7_7M == 1){
        mov6_7M = 1;
}
mov7_7M = 0;
return mov6_7M;
}

/* funcao para testar NO2 do 7§ veiculo */
int f_NO2M_6 ()
{   int i_6M, j_6M, p_6M, q_6M, mov6_6M, mov5_6M, for6M, f1_6M, flag_6M;

    mov5_6M = 0;
    flag_6M = 0;

    i_6M = indexi(melNO2 [tpo][VEI[6]]);
    j_6M = indexj(melNO2 [tpo][VEI[6]]);
    /* existe um veiculo no NO2(do veic.7) ? */
    mov6_6M = 0;
    for (for6M = 0; for6M < Num_VeiT; for6M ++){
          if (for6M != VEI[6]){
              if (((MATNOS[i_6M][j_6M]==melNO1[tpo][for6M])&&(melNO2[tpo][for6M]>0))||(NOS_AHEAD[i_6M][j_6M]<-1)){
                    flag_6M = 0;
                    for (f1_6M = 0; f1_6M < Num_VeiT; f1_6M++){
                        if (for6M == VEI [f1_6M]){
                              flag_6M = 1;
                              if (f1_6M == 0){
                                  flag_6M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_6M][j_6M]<-1) || (flag_6M == 1)){
                        mov6_6M = 1;
                    }
                    else if (flag_6M == -1){
                        mov6_6M = 0;
                    }
                    else {
                        VEI [7] = for6M;
                        mov6_6M = f_NO2M_7 ();
                    }
                    flag_6M = 0;
              }
          }
    }
    if (mov6_6M == 0){
          for (for6M = 0; for6M < Num_VeiT; for6M++){
              if (MATNOS[i_6M][j_6M]==melNO2[tpo][for6M]){
                    flag_6M = 0;
                    for (f1_6M = 0; f1_6M < Num_VeiT; f1_6M++){
                        if (for6M == VEI [f1_6M]){
                              flag_6M = 1;
                        }
                    }
                    if (flag_6M != 1){
                        p_6M = indexi (melNO1 [tpo][for6M]);
                        q_6M = indexj (melNO1 [tpo][for6M]);
                        melNO1 [tpo+1][for6M] = MATNOS [p_6M][q_6M];
                        melNO2 [tpo][for6M] = -112;
                        NOS_AHEAD [p_6M][q_6M] = -112;
                    }
                    flag_6M = 0;
              }
          }
          melNO1 [tpo+1][VEI[6]] = MATNOS [i_6M][j_6M];
          melNO2 [tpo][VEI[6]] = -112;
          NOS_AHEAD [i_6M][j_6M] = -112;
          mov5_6M = 0;
    }
    else if (mov6_6M == 1){
          mov5_6M = 1;
    }
    mov6_6M = 0;
    return mov5_6M;
}

/* funcao para testar NO2 do 6§ veiculo */
int f_NO2M_5 ()
{   int i_5M, j_5M, p_5M, q_5M, mov5_5M, mov4_5M, for5M, f1_5M, flag_5M;

    mov4_5M = 0;
    flag_5M = 0;

    i_5M = indexi(melNO2 [tpo][VEI[5]]);
    j_5M = indexj(melNO2 [tpo][VEI[5]]);
    /* existe um veiculo no NO2(do veic.6) ? */
    mov5_5M = 0;
    for (for5M = 0; for5M < Num_VeiT; for5M ++){
          if (for5M != VEI[5]){
              if (((MATNOS[i_5M][j_5M]==melNO1[tpo][for5M])&&(melNO2[tpo][for5M]>0))||(NOS_AHEAD[i_5M][j_5M]<-1)){
```

```
                    flag_5M = 0;
                    for (f1_5M = 0; f1_5M < Num_VeiT; f1_5M++){
                        if (for5M == VEI [f1_5M]){
                              flag_5M = 1;
                              if (f1_5M == 0){
                                  flag_5M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_5M][j_5M]<-1) || (flag_5M == 1)){
                        mov5_5M = 1;
                    }
                    else if (flag_5M == -1){
                        mov5_5M = 0;
                    }
                    else {
                        VEI [6] = for5M;
                        mov5_5M = f_NO2M_6 ();
                    }
                    flag_5M = 0;
               }
          }
}
if (mov5_5M == 0){
      for (for5M = 0; for5M < Num_VeiT; for5M++){
            if (MATNOS[i_5M][j_5M]==melNO2[tpo][for5M]){
                  flag_5M = 0;
                  for (f1_5M = 0; f1_5M < Num_VeiT; f1_5M++){
                      if (for5M == VEI [f1_5M]){
                            flag_5M = 1;
                      }
                  }
                  if (flag_5M != 1){
                      p_5M = indexi (melNO1 [tpo][for5M]);
                      q_5M = indexj (melNO1 [tpo][for5M]);
                      melNO1 [tpo+1][for5M] = MATNOS [p_5M][q_5M];
                      melNO2 [tpo][for5M] = -112;
                      NOS_AHEAD [p_5M][q_5M] = -112;
                  }
                  flag_5M = 0;
            }
      }
      melNO1 [tpo+1][VEI[5]] = MATNOS [i_5M][j_5M];
      melNO2 [tpo][VEI[5]] = -112;
      NOS_AHEAD [i_5M][j_5M] = -112;
      mov4_5M = 0;
}
else if (mov5_5M == 1){
      mov4_5M = 1;
}
mov5_5M = 0;
return mov4_5M;
}

/* funcao para testar NO2 do 5§ veiculo */
int f_NO2M_4 ()
{   int i_4M, j_4M, p_4M, q_4M, mov4_4M, mov3_4M, for4M, f1_4M, flag_4M;

    mov3_4M = 0;
    flag_4M = 0;

    i_4M = indexi(melNO2 [tpo][VEI[4]]);
    j_4M = indexj(melNO2 [tpo][VEI[4]]);
    /* existe um veiculo no NO2(do veic.5) ? */
    mov4_4M = 0;
    for (for4M = 0; for4M < Num_VeiT; for4M ++){
          if (for4M != VEI[4]){
              if (((MATNOS[i_4M][j_4M]==melNO1[tpo][for4M])&&(melNO2[tpo][for4M]>0))||(NOS_AHEAD[i_4M][j_4M]<-1)){
                    flag_4M = 0;
                    for (f1_4M = 0; f1_4M < Num_VeiT; f1_4M++){
                        if (for4M == VEI [f1_4M]){
                              flag_4M = 1;
                              if (f1_4M == 0){
                                  flag_4M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_4M][j_4M]<-1) || (flag_4M == 1)){
                        mov4_4M = 1;
                    }
                    else if (flag_4M == -1){
                        mov4_4M = 0;
                    }
                    else {
                        VEI [5] = for4M;
                        mov4_4M = f_NO2M_5 ();
                    }
                    flag_4M = 0;
               }
          }
}
if (mov4_4M == 0){
      for (for4M = 0; for4M < Num_VeiT; for4M++){
          if (MATNOS[i_4M][j_4M]==melNO2[tpo][for4M]){
                flag_4M = 0;
                for (f1_4M = 0; f1_4M < Num_VeiT; f1_4M++){
                    if (for4M == VEI [f1_4M]){
                          flag_4M = 1;
                    }
                }
                if (flag_4M != 1){
                    p_4M = indexi (melNO1 [tpo][for4M]);
                    q_4M = indexj (melNO1 [tpo][for4M]);
                    melNO1 [tpo+1][for4M] = MATNOS [p_4M][q_4M];
                    melNO2 [tpo][for4M] = -112;
                    NOS_AHEAD [p_4M][q_4M] = -112;
                }
                flag_4M = 0;
          }
```

```
            }
            melNO1 [tpo+1][VEI[4]] = MATNOS [i_4M][j_4M];
            melNO2 [tpo][VEI[4]] = -112;
            NOS_AHEAD [i_4M][j_4M] = -112;
            mov3_4M = 0;
        }
        else if (mov4_4M == 1){
            mov3_4M = 1;
        }
        mov4_4M = 0;
        return mov3_4M;
}

/* funcao para testar NO2 do 4§ veiculo */
int f_NO2M_3 ()
{   int i_3M, j_3M, p_3M, q_3M, mov3_3M, mov2_3M, for3M, f1_3M, flag_3M;

    mov2_3M = 0;
    flag_3M = 0;

    i_3M = indexi(melNO2 [tpo][VEI[3]]);
    j_3M = indexj(melNO2 [tpo][VEI[3]]);
    /* existe um veiculo no NO2(do veic.4) ? */
    mov3_3M = 0;
    for (for3M = 0; for3M < Num_VeiT; for3M ++){
        if (for3M != VEI[3]){
            if (((MATNOS[i_3M][j_3M]==melNO1[tpo][for3M])&&(melNO2[tpo][for3M]>0))||(NOS_AHEAD[i_3M][j_3M]<-1)){
                flag_3M = 0;
                for (f1_3M = 0; f1_3M < Num_VeiT; f1_3M++){
                    if (for3M == VEI [f1_3M]){
                        flag_3M = 1;
                        if (f1_3M == 0){
                            flag_3M = -1;
                        }
                    }
                }
                if ((NOS_AHEAD[i_3M][j_3M]<-1) || (flag_3M == 1)){
                    mov3_3M = 1;
                }
                else if (flag_3M == -1){
                    mov3_3M = 0;
                }
                else {
                    VEI [4] = for3M;
                    mov3_3M = f_NO2M_4 ();
                }
                flag_3M = 0;
            }
        }
}
if (mov3_3M == 0){
        for (for3M = 0; for3M < Num_VeiT; for3M++){
            if (MATNOS[i_3M][j_3M]==melNO2[tpo][for3M]){
                flag_3M = 0;
                for (f1_3M = 0; f1_3M < Num_VeiT; f1_3M++){
                    if (for3M == VEI [f1_3M]){
                        flag_3M = 1;
                    }
                }
                if (flag_3M != 1){
                    p_3M = indexi (melNO1 [tpo][for3M]);
                    q_3M = indexj (melNO1 [tpo][for3M]);
                    melNO1 [tpo+1][for3M] = MATNOS [p_3M][q_3M];
                    melNO2 [tpo][for3M] = -112;
                    NOS_AHEAD [p_3M][q_3M] = -112;
                }
                flag_3M = 0;
            }
        }
        melNO1 [tpo+1][VEI[3]] = MATNOS [i_3M][j_3M];
        melNO2 [tpo][VEI[3]] = -112;
        NOS_AHEAD [i_3M][j_3M] = -112;
        mov2_3M = 0;
    }
    else if (mov3_3M == 1){
        mov2_3M = 1;
    }
    mov3_3M = 0;
    return mov2_3M;
}

/* funcao para testar NO2 do 3§ veiculo */
int f_NO2M_2 ()
{   int i_2M, j_2M, p_2M, q_2M, mov2_2M, mov1_2M, for2M, f1_2M, flag_2M;

    mov1_2M = 0;
    flag_2M = 0;

    i_2M = indexi(melNO2 [tpo][VEI[2]]);
    j_2M = indexj(melNO2 [tpo][VEI[2]]);
    /* existe um veiculo no NO2(do veic.3) ? */
    mov2_2M = 0;
    for (for2M = 0; for2M < Num_VeiT; for2M ++){
        if (for2M != VEI[2]){
            if (((MATNOS[i_2M][j_2M]==melNO1[tpo][for2M])&&(melNO2[tpo][for2M]>0))||(NOS_AHEAD[i_2M][j_2M]<-1)){
                flag_2M = 0;
                for (f1_2M = 0; f1_2M < Num_VeiT; f1_2M++){
                    if (for2M == VEI [f1_2M]){
                        flag_2M = 1;
                        if (f1_2M == 0){
                            flag_2M = -1;
                        }
                    }
                }
                if ((NOS_AHEAD[i_2M][j_2M]<-1) || (flag_2M == 1)){
                    mov2_2M = 1;
                }
                else if (flag_2M == -1){
                    mov2_2M = 0;
```

```
                }
                else {
                    VEI [3] = for2M;
                    mov2_2M = f_NO2M_3 ();
                }
                flag_2M = 0;
            }
        }
}
if (mov2_2M == 0){
      for (for2M = 0; for2M < Num_VeiT; for2M++){
          if (MATNOS[i_2M][j_2M]==melNO2[tpo][for2M]){
                flag_2M = 0;
                for (f1_2M = 0; f1_2M < Num_VeiT; f1_2M++){
                    if (for2M == VEI [f1_2M]){
                          flag_2M = 1;
                    }
                }
                if (flag_2M != 1){
                    p_2M = indexi (melNO1 [tpo][for2M]);
                    q_2M = indexj (melNO1 [tpo][for2M]);
                    melNO1 [tpo+1][for2M] = MATNOS [p_2M][q_2M];
                    melNO2 [tpo][for2M] = -112;
                    NOS_AHEAD [p_2M][q_2M] = -112;
                }
                flag_2M = 0;
                }
          }
          melNO1 [tpo+1][VEI[2]] = MATNOS [i_2M][j_2M];
          melNO2 [tpo][VEI[2]] = -112;
          NOS_AHEAD [i_2M][j_2M] = -112;
          mov1_2M = 0;
    }
    else if (mov2_2M == 1){
          mov1_2M = 1;
    }
    mov2_2M = 0;
    return mov1_2M;
}

/* funcao para testar NO2 do 2§ veiculo */
int f_NO2M_1 ()
{   int i_1M, j_1M, p_1M, q_1M, mov1_1M, mov0_1M, for1M, f1_1M, flag_1M;

    mov0_1M = 0;
    flag_1M = 0;

    i_1M = indexi(melNO2 [tpo][VEI[1]]);
    j_1M = indexj(melNO2 [tpo][VEI[1]]);
    /* existe um veiculo no NO2(do veic.2) ? */
    mov1_1M = 0;
    for (for1M = 0; for1M < Num_VeiT; for1M ++){
          if (for1M != VEI[1]){
              if (((MATNOS[i_1M][j_1M]==melNO1[tpo][for1M])&&(melNO2[tpo][for1M]>0))||(NOS_AHEAD[i_1M][j_1M]<-1)){
                    flag_1M = 0;
                    for (f1_1M = 0; f1_1M < Num_VeiT; f1_1M++){
                        if (for1M == VEI [f1_1M]){
                              flag_1M = 1;
                              if (f1_1M == 0){
                                  flag_1M = -1;
                              }
                        }
                    }
                    if ((NOS_AHEAD[i_1M][j_1M]<-1) || (flag_1M == 1)){
                        mov1_1M = 1;
                    }
                    else if (flag_1M == -1){
                        mov1_1M = 0;
                    }
                    else {
                        VEI[2] = for1M;
                        mov1_1M = f_NO2M_2 ();
                    }
                    flag_1M = 0;
              }
          }
}
if (mov1_1M == 0){
      for (for1M = 0; for1M < Num_VeiT; for1M++){
          if (MATNOS[i_1M][j_1M]==melNO2[tpo][for1M]){
                flag_1M = 0;
                for (f1_1M = 0; f1_1M < Num_VeiT; f1_1M++){
                    if (for1M == VEI [f1_1M]){
                          flag_1M = 1;
                    }
                }
                if (flag_1M != 1){
                    p_1M = indexi (melNO1 [tpo][for1M]);
                    q_1M = indexj (melNO1 [tpo][for1M]);
                    melNO1 [tpo+1][for1M] = MATNOS [p_1M][q_1M];
                    melNO2 [tpo][for1M] = -112;
                    NOS_AHEAD [p_1M][q_1M] = -112;
                }
                flag_1M = 0;
              }
          }
          melNO1 [tpo+1][VEI[1]] = MATNOS [i_1M][j_1M];
          melNO2 [tpo][VEI[1]] = -112;
          NOS_AHEAD [i_1M][j_1M] = -112;
          mov0_1M = 0;
    }
    else if (mov1_1M == 1){
          mov0_1M = 1;
    }
    mov1_1M = 0;
    return mov0_1M;
}
```

```
/* funcao que implementa as regras heuristicas (look-ahead, prio.nos dest.)
comuns as varias estrategias (ANN, Greedy Pol., Benv. Pol.)
passo_ahead = 0: considera todas as implicacoes possiveis em todos os veiculos,
passo_ahead = 1: APENAS UM PASSO A FRENTE (se o segundo NO em analise estiver
ocupado PARA o primeiro veiculo)        */

void f_st_ahead (int step_ind)
{   int stM1,stM2,stz1,st_for,st_k1,st_k2,st_k3,st_k4,mst_NO2M0,step_flag;

    step_flag = step_ind;

    /* inicializar matriz NOS_AHEAD [5][5] */

    f_Nos_ahead ();

    for (stM1 = 0; stM1 < Num_MaxV; stM1 ++){
          VEI [stM1] = -1;
    }

    for (stM1 = 0; stM1 < Num_VeiT; stM1 ++){
          st_k1 = indexi(melNO1 [tpo][stM1]);
          st_k2 = indexj(melNO1 [tpo][stM1]);
          NOS_AHEAD [st_k1][st_k2] = NOS_AHEAD [st_k1][st_k2] + 1;
          if (melNO1 [tpo][stM1] != melNO2 [tpo][stM1]){
              st_k3 = indexi(melNO2 [tpo][stM1]);
              st_k4 = indexj(melNO2 [tpo][stM1]);
              NOS_AHEAD [st_k3][st_k4] = NOS_AHEAD [st_k3][st_k4] + 1;
          }
    }

/* quantos e quais os veiculos que tem o mesmo NO2 ? como decidir
quais os que irao para esses nos e quais os que esperam onde estao?
(a)- se um veiculo esta bloqueado entao tem prioridade sobre qq outro,
(b)- se um veiculo chega ao destino com esse no tem prioridade seguinte,
(c)- nos outros casos vale a prioridade que cada veiculo tem atribuida. */

/* quais os veiculos que estao bloqueados ? */
for (stM1 = 0; stM1 < Num_VeiT; stM1 ++){
      if (melNO2 [tpo][stM1] == melNO1 [tpo][stM1]){
          st_k1 = indexi (melNO1 [tpo][stM1]);
          st_k2 = indexj (melNO1 [tpo][stM1]);
          NOS_AHEAD [st_k1][st_k2] = -333;
          melNO1 [tpo+1][stM1] = melNO1 [tpo][stM1];
          melNO2 [tpo][stM1] = -333;
      }
}
/* quais os que pretendem ir para um NO destino ?
NOTA: o no destino nao pode ser o mesmo para varios veiculos
( ordenados pelas suas prioridades) */
for (stM1 = 0; stM1 < Num_MaxV; stM1 ++){
      int_ordA [stM1][0] = 0;
      int_ordA [stM1][1] = 0;
}
stz1 = 0;
for (stM1 = 0; stM1 < Num_VeiT; stM1 ++){
      if(melNO2 [tpo][vei_Pr[tpo][stM1]-1] == endint_noV [vei_Pr[tpo][stM1]-1]){
          int_ordA [stz1][0] = vei_Pr [tpo][stM1];
          int_ordA [stz1][1] = vei_Pr_id [tpo][vei_Pr [tpo][stM1]-1];
          stz1 = stz1 +1;
      }
}

/* comecar a atribuicao de lugares pelos mais prioritarios
e que pretendem que o proximo NO seja o NO destino */
stz1 = 0;
while (int_ordA [stz1][0] > 0){
      if ((melNO2[tpo][int_ordA[stz1][0]-1] > 0) && (melNO2[tpo][int_ordA[stz1][0]-1] < 22)){
          /* no destino livre ? (ou seja apenas e pretendido pelo veiculo em analise)*/
          st_k1 = indexi(melNO2 [tpo][int_ordA[stz1][0]-1]);
          st_k2 = indexj(melNO2 [tpo][int_ordA[stz1][0]-1]);
          if ( (NOS_AHEAD [st_k1][st_k2] == 0) || (NOS_AHEAD [st_k1][st_k2] == 1)){
                melNO1 [tpo+1][int_ordA[stz1][0]-1] = melNO2 [tpo][int_ordA[stz1][0]-1];
                NOS_AHEAD [st_k1][st_k2] = -444;
                st_k3 = indexi(melNO1 [tpo][int_ordA[stz1][0]-1]);
                st_k4 = indexj(melNO1 [tpo][int_ordA[stz1][0]-1]);
                NOS_AHEAD [st_k3][st_k4] = NOS_AHEAD[st_k3][st_k4] -1;
                melNO2 [tpo][int_ordA[stz1][0]-1] = -444;
          }
          /* no destino ja ocupado em definitivo? */
          else if (NOS_AHEAD [st_k1][st_k2] < -1){
                melNO1 [tpo+1][int_ordA[stz1][0]-1] = melNO1 [tpo][int_ordA[stz1][0]-1];
                st_k3 = indexi(melNO1 [tpo][int_ordA[stz1][0]-1]);
                st_k4 = indexj(melNO1 [tpo][int_ordA[stz1][0]-1]);
                NOS_AHEAD [st_k3][st_k4] = -555;
                melNO2 [tpo][int_ordA[stz1][0]-1] = -555;
          }
/* no destino ocupado por um veiculo que ainda nao
decidiu, ou no destino tambem para outro veiculo ?
(a decisao tomada de avancar tem que garantir que se
esse no estivesse ocupado por um veiculo ele tem que
sair e ocupar outro no, ou se esse no fosse um no
pretendido por outro veiculo esse outro veiculo tera
que ficar imobilizado onde se encontra). */
else if (NOS_AHEAD [st_k1][st_k2] > 1){

      /*registo do primeiro veiculo analisado */
      VEI [0] = int_ordA [stz1][0]-1;

      /* identificar o veiculo que ainda se encontra
      no NO pretendido */
      mst_NO2M0 = 0;
      for (st_for = 0; st_for < Num_VeiT; st_for++){
          if (st_for != VEI[0]){
                /* existe um veiculo no NO2 ? */
                if ((MATNOS[st_k1][st_k2]==melNO1[tpo][st_for])&&(melNO2[tpo][st_for]>0)){

                    VEI [1] = st_for;

                    if (step_flag == 0){
```

```
                        mst_NO2M0 = f_NO2M_1 ();
                    }
                    else if (step_flag == 1){
                        mst_NO2M0 = f_NO2M5 (st_for);
                    }
                    else{
                        /* PutStr ("error in step_flag"); */
                    }
                }
            }
        }
        if (mst_NO2M0 == 0){
            /* no inicial pode avancar e actualizar apenas os
            veiculos que pretendiam o mesmo no, uma vez que
            o que la estivesse ja foi actualizado atras*/
            for (st_for = 0; st_for < Num_VeiT; st_for++){
                if (st_for != VEI [0]){
                    /* existe um veiculo no NO2 ? */
                    if (MATNOS [st_k1][st_k2] == melNO2 [tpo][st_for]){
                        st_k3 = indexi (melNO1 [tpo][st_for]);
                        st_k4 = indexj (melNO1 [tpo][st_for]);
                        melNO1[tpo+1][st_for] = MATNOS [st_k3][st_k4];
                        melNO2[tpo][st_for] = -222;
                        NOS_AHEAD [st_k3][st_k4] = -222;
                    }
                    else{
                        }
                    }
                    melNO1 [tpo+1][VEI [0]] = MATNOS [st_k1][st_k2];
                    melNO2 [tpo][VEI [0]] = -222;
                    NOS_AHEAD [st_k1][st_k2] = -222;
                    st_k3 = indexi(melNO1 [tpo][VEI [0]]);
                    st_k4 = indexj(melNO1 [tpo][VEI [0]]);
                    NOS_AHEAD [st_k3][st_k4] = NOS_AHEAD [st_k3][st_k4] - 1;
                }
                else if (mst_NO2M0 == 1){
                    /* no inicial nao pode avancar */
                    st_k3 = indexi(melNO1 [tpo][VEI [0]]);
                    st_k4 = indexj(melNO1 [tpo][VEI [0]]);
                    melNO1 [tpo+1][VEI [0]] = MATNOS [st_k3][st_k4];
                    melNO2 [tpo][VEI [0]] = -222;
                    NOS_AHEAD [st_k3][st_k4] = -222;
                }
                mst_NO2M0 = 0;
                for (stM2 = 0; stM2 < Num_MaxV; stM2 ++){
                    VEI [stM2] = -1;
                }
            }
        }
        stz1 ++;
    }

    /* analise do movimento dos veiculos que nao foram afectados
    pela analise anterior (veiculos bloqueados ou veiculos que
    terminam o seu percurso com o no pretendido)
    (agora apenas temos que identificar quais os veiculos ainda em
    definicao de movimento e destes quais os que pretendem os mesmos
    nos). */

    st_k1 = 0;
    st_k2 = 0;
    st_k3 = 0;
    st_k4 = 0;
    for (stM1 = 0; stM1 < Num_VeiT; stM1++){
        if ((melNO2 [tpo][vei_Pr[tpo][stM1]-1] > 0) && (melNO2 [tpo][vei_Pr[tpo][stM1]-1] < 22)){
            st_k1 = indexi(melNO2 [tpo][vei_Pr[tpo][stM1]-1]);
            st_k2 = indexj(melNO2 [tpo][vei_Pr[tpo][stM1]-1]);
            /* verificar se o no pretendido esta livre (ou seja apenas e
            pretendido pelo veiculo em analise) ? */
            if ( (NOS_AHEAD [st_k1][st_k2] == 0) || (NOS_AHEAD [st_k1][st_k2] == 1)){
                melNO1 [tpo +1][vei_Pr[tpo][stM1]-1] =  MATNOS [st_k1][st_k2];
                melNO2 [tpo][vei_Pr[tpo][stM1]-1] = -777;
                st_k3 = indexi(melNO1 [tpo][vei_Pr[tpo][stM1]-1]);
                st_k4 = indexj(melNO1 [tpo][vei_Pr[tpo][stM1]-1]);
                NOS_AHEAD [st_k3][st_k4] = NOS_AHEAD [st_k3][st_k4] -1;
                NOS_AHEAD [st_k1][st_k2] = -777;
            }
            /* esta ocupado em definitivo ? */
            else if ( NOS_AHEAD [st_k1][st_k2] < -1){
                st_k3 = indexi(melNO1 [tpo][vei_Pr[tpo][stM1]-1]);
                st_k4 = indexj(melNO1 [tpo][vei_Pr[tpo][stM1]-1]);
                melNO1 [tpo +1][vei_Pr[tpo][stM1]-1] =  MATNOS[st_k3][st_k4];
                melNO2 [tpo][vei_Pr[tpo][stM1]-1] = -888;
                NOS_AHEAD [st_k3][st_k4] = -888;
            }
            /* ocupado por um veiculo que podera sair, ou com pretendentes ao no? */
            else if (NOS_AHEAD [st_k1][st_k2] > 1){

    /*registo do primeiro veiculo analisado */
    VEI [0] = vei_Pr[tpo][stM1]-1;

    /* identificar o veiculo que ainda se encontra
    no NO pretendido */
    mst_NO2M0 = 0;
    for (st_for = 0; st_for < Num_VeiT; st_for++){
        if (st_for != (VEI [0])){
            /* existe um veiculo no NO2(v1) ? */
            if ((MATNOS[st_k1][st_k2]==melNO1[tpo][st_for])&&(melNO2[tpo][st_for]>0)){

                VEI [1] = st_for;

                if (step_flag == 0){
                    mst_NO2M0 = f_NO2M_1 ();
                }
                else if (step_flag == 1){
                    mst_NO2M0 = f_NO2M5 (st_for);
                }
                else{
                    /* PutStr ("error in step_flag");*/
                }
```

```
                }
            }
        }
        if (mst_NO2M0 == 0){
            /* no inicial pode avancar e actualizar apenas os
            veiculos que pretendiam o mesmo no, uma vez que
            o que la estivesse ja foi actualizado atras*/
            for (st_for = 0; st_for < Num_VeiT; st_for++){
                if (st_for != (VEI [0])){
                    /* existe um veiculo para o NO2 ? */
                    if (MATNOS [st_k1][st_k2] == melNO2 [tpo][st_for]){
                        st_k3 = indexi (melNO1 [tpo][st_for]);
                        st_k4 = indexj (melNO1 [tpo][st_for]);
                        melNO1[tpo+1][st_for] = MATNOS [st_k3][st_k4];
                        melNO2[tpo][st_for] = -222;
                        NOS_AHEAD [st_k3][st_k4] = -222;
                    }
                }
            }
            melNO1 [tpo+1][VEI [0]] = MATNOS [st_k1][st_k2];
            melNO2 [tpo][VEI [0]] = -222;
            NOS_AHEAD [st_k1][st_k2] = -222;
            st_k3 = indexi(melNO1 [tpo][VEI [0]]);
            st_k4 = indexj(melNO1 [tpo][VEI [0]]);
            NOS_AHEAD [st_k3][st_k4] = NOS_AHEAD [st_k3][st_k4] - 1;
                    }
                    else if (mst_NO2M0 == 1){
                        /* no inicial nao pode avancar */
                        st_k3 = indexi(melNO1 [tpo][VEI [0]]);
                        st_k4 = indexj(melNO1 [tpo][VEI [0]]);
                        melNO1 [tpo+1][VEI [0]] = MATNOS [st_k3][st_k4];
                        melNO2 [tpo][VEI [0]] = -222;
                        NOS_AHEAD [st_k3][st_k4] = -222;
                    }
                    mst_NO2M0 = 0;
                    for (stM2 = 0; stM2 < Num_MaxV; stM2 ++){
                        VEI [stM2] = -1;
                    }
                }
            }
        }


}

/*  funcao para colocar os valores da matriz ocupacao dos nos (N_OCUP1[5][5])
correspondentes a nos existentes todos livres     */
void f_Nos_ahead()
{   int i_ahe, j_ahe;

    for (i_ahe = 0; i_ahe < 5; i_ahe ++){
        for (j_ahe = 0; j_ahe < 5; j_ahe ++){
            if (MATNOS [i_ahe][j_ahe] >= 1 && MATNOS [i_ahe][j_ahe] <= 21){
                NOS_AHEAD [i_ahe][j_ahe] = 0;
            }
            else if (MATNOS [i_ahe][j_ahe] == 0){
                NOS_AHEAD [i_ahe][j_ahe] = -1;
            }
            else {
                /* PutStr ("error in MATNOS "); */
            }
        }
    }
}
```

Appendix D

Program Developed to Analyse Results in the Multi-Vehicle Case

In this appendix the computer program that was used to analyse and quantify the results obtained from the tests done with the multi-vehicle case, is described. The same program was used to analyse all tests performed with the different strategies and discussed in Chapter 9. It was convenient to separate data generation from data analysis because it took longer to generate data, and the two processes being independent different analysis on the data could be performed without the need to repeat the experiments which generate data. The data analysed by this program consists of the details of the paths registered at each time instant of the duration of the test and using the program which implements each specific strategy, such as the interface program listed in Appendix C for the ANN based solutions. These details are stored in a particular format in a data file as shown by the example presented in section 1.1.1. The program used to analyse data consists of reading the data from that file and performing the required analysis. The results of this analysis are written to two files, one which contains the details of each path (example in section 1.1.2), and another which only contains a summary of the global results. The four measures of performance described in Chapter 9 were obtained using this program; the *'metric'* and *'number of bad paths'* are obtained directly, and the *'throughput'* and *'k-ratio'* are obtained by manipulating in a spreadsheet package (EXCEL, from Microsoft Corporation) the results file containing the details of each path (example in section 1.1.2).

In the next sections a description of the program is presented with the help of a flowchart and examples of the data files generated by the program and the files containing the information required by the program. The code was implemented using the Microsoft C package version 6.0 for MS-DOS (Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA), and a listing of the main program is also included in section 1.2.

1.1 Details of the program

The flowchart in Figure D- 1 represents the main phases of the program used to analyse the results obtained when testing each strategy in a multi-vehicle case. These phases are described as follows.

Phase A ⇒ In this initial phase the variables defined are reset to the initial default values, and the files used for reading data from and writing data to are open. All the information concerning the results of the test performed is read from a file. As is shown by the example presented in next section, section 1.1.1, this file starts with information relative to general aspects of the test performed which is followed by a set of lines with information associated with the status of the system at each time instant. In this first phase only the general information is read and it consists of (see section 1.1.1): the number of vehicles (10), the total number of iterations of all vehicles (100000), the seed number (5) used for the pseudo-random numbers generator, the maximum number of time units (50) allowed for a vehicle to find its destination before specific attention is given to it, the number of time units (3) a vehicle can be blocked at a node, before the last track used is no longer considered as a moving backwards alternative, and finally the type of "look-ahead" strategy used (i.e. 0, representing complete "look-ahead").

Phase B ⇒ This phase corresponds to the start of a loop that is repeated as many times as the number of time units of the duration of the test. In this phase three lines of the data file containing the results from the test are read in each iteration of the loop. Because the data file is read sequentially these three lines are each time the lines associated with each time instant of the test performed. Except for the first column, the information on this lines is associated with each vehicle in the system. The first element on the first line corresponds to the time instant and is followed by four types of information for each vehicle: the current node, the node wanted by the vehicle as a result of its decision on that time instant, the priority of the vehicle, and a number associated with the type of solution corresponding to that decision. The first element on the second line is also the time instant, and is followed by three more types of information for each vehicle: the node allocated to the vehicle, the destination node, and the number of time units the vehicle has been blocked at that node before

moving backwards is no longer considered as such. The first element on the third line represents the number of previous steps of each path that are still kept in memory. The other three elements of information for each vehicle on this line, indicate information relative to the next time instant: the current node, the destination node, and a number which is used to identify different path lengths. The specific value of this number is used in the identification of the following situations of each path at each time instant:

- the path length is less or equal to 50 (value = 0)
- the vehicle is looking for a temporary destination node (value = original node number)
- the temporary node number is found in time less or equal to 100 (value = -1)
- the path length is above 100 (value = -2).

Phase C ⇒ The information about the current path of each vehicle collected in the previous phase (Phase B) is now used to identify and classify each of these paths. Each path, or vehicle, is analyzed at each iteration of a loop with as many iterations as the number of vehicles involved. This analysis consists of identifying whether the path terminates or not, and updating the variables used to store the information required for data analysis. A path which does not terminates at the current time requires only to check and account for its movement or stopping situation. A path is considered as terminated if it reaches 200 time units, or finds its original destination node. The paths completed are further classified according to the path lengths:

- counted as a "very good solution", if path length is less than 50
- counted as "good solution", if path length is greater or equal to 50 and less than 100
- counted as "not good solution" if path length greater or equal to 100 and less than 200
- counted as "bad solution" if path length is 200

These results are stored individually for each vehicle and also for the total of all vehicles. In section 1.1.2 these values are represented as 'VGood', 'Good', 'NGood', 'Mas', and totals 'Som_VGo', 'Som_Goo', 'Som_NGo', 'Som_Mas', respectively; and 'cont_casos' as the total number of paths completed. When a path is completed the results file is updated with the following information: the time instant, the identification of the vehicle, the origin node number, the destination node number, the number of tracks used, the number of waiting times due to blockage at a node, and the minimum

geometric distance. These values are identified in section 1.1.2 in a line starting with a number (time instant), and then followed by 'AGV:', 'NOR:', 'NOD:', 'TRA', 'ESP', and 'OPT'. The totals of the values of these last three elements for each vehicle are also calculated ('STR', 'STE', 'MDIS' respectively in section 1.1.2).

In addition to these results the program also calculates the number of paths with time lengths (travel plus waiting) exceeding the respective minimum geometric distances by a specific number of times (i.e. 0 to 200) and which are represented by '[0...200]' 'HIS =' in the last set of elements in section 1.1.2.. The information for each of these classes of paths was also complemented with the average of the path lengths corresponding to the minimum geometric distances, and the maximum of the minimum geometric distances ('ACMP =', and 'MAX =' respectively in section 1.1.2). The last element calculated at this stage was the number of paths with a time duration equal to a specific value, ranging from the minimum (1) to the maximum (200) and represented in section 1.1.2 as 'CASOS ='.

Phase D ⇒ This last phase follows after all time instants of the duration of the test are analyzed in phases B and C. It consists of calculating and writing to the results files (i.e. section 1.1.2), first the total values for each individual vehicle and subsequently the values associated with the total number of paths performed by all vehicles. One of the results is the '*metric*' performance measure which can be identified as 'METRICA =' in section 1.1.2., and which is calculated using the total number of tracks used in all paths ('Soma_TT ='), the total number of waiting times ('Soma_TE ='), and the total number of tracks used if the paths were completed along the minimum geometric distance without traffic congestion ('Soma =').

The total values for each individual vehicle (i.e. number of paths completed by each vehicle, the total time of these paths) were used to calculate the 'throughput' performance (Chapter 9) using a spreadsheet package. The calculation of the 'k-ratio' performance measure (Chapter 9) was also performed using a spreadsheet package and the values of each individual path (i.e. the total time, time associated with the minimum geometric distance).

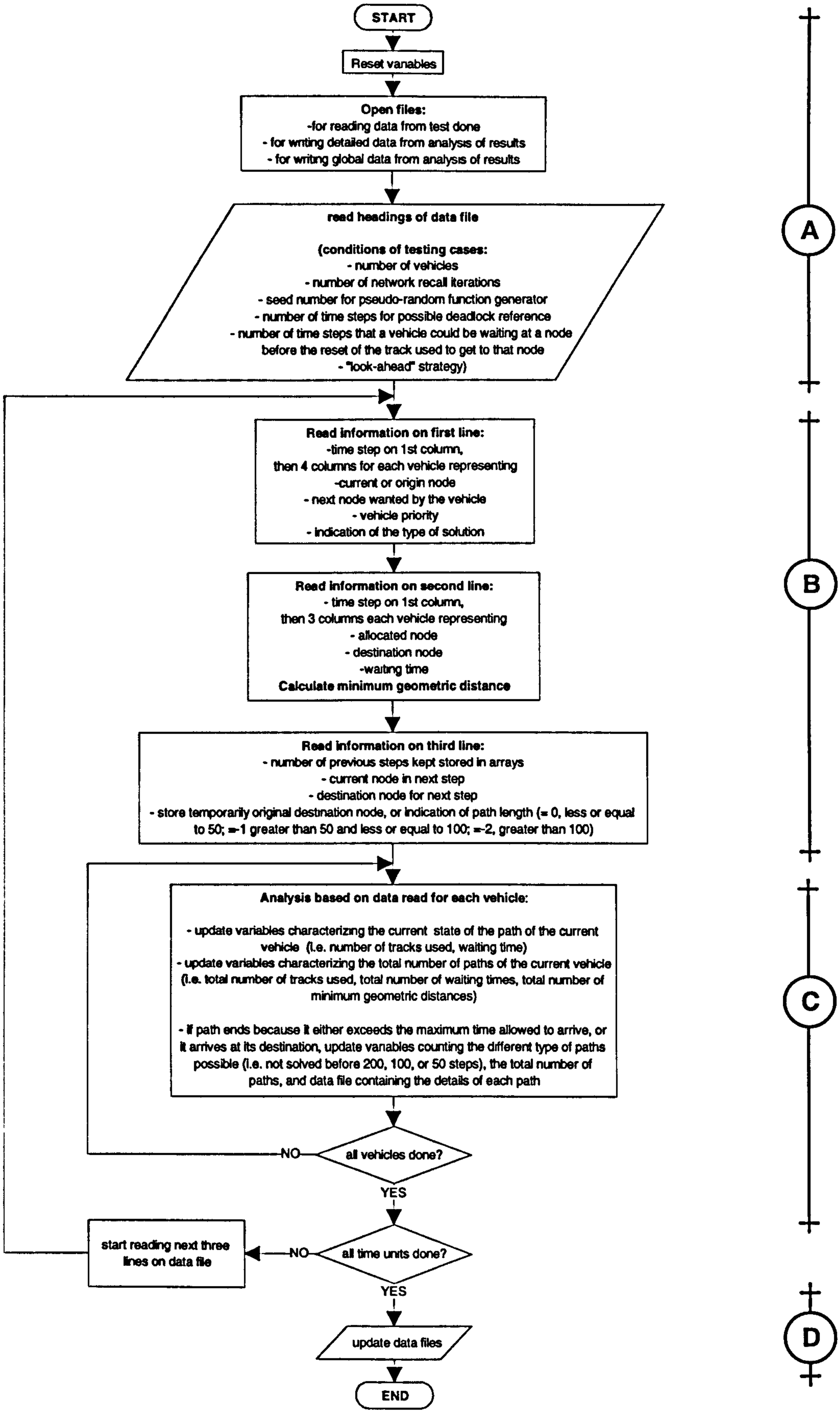


Figure D- 1: Flowchart representing the main steps of the program implemented to analyse data.

1.1.1 Example of Data File Containing Information From Tests

Num.Vehicles:10 Num. Iteracoes Rede NN:100000
Seed:5
Max. Iter. to Dealock:50

Reset dos tramos usados apos: 3

ESTRATEgia= 0

0	8 13 1 1	7 3 2 1	21 16 3 1	2 3 4 1	17 18 5 1	10 11 6 1	1 6 7 2	15 11 8 1	18 19 9 4	6 9 10 2
0	13 15 0	3 5 0	16 16 0	2 8 1	18 21 0	11 7 0	6 14 0	15 1 1	19 2 0	9 20 0
1	13 15 0	3 5 0	16 17 0	2 8 0	18 21 0	11 7 0	6 14 0	15 1 0	19 2 0	9 20 0
1	13 12 2 1	3 4 3 1	16 21 4 1	2 3 5 1	18 19 6 1	11 7 7 2	6 9 8 1	15 11 9 1	19 15 10 2	9 14 1 1
1	12 15 0	4 5 0	21 17 0	3 8 0	19 21 0	7 7 0	9 14 0	11 1 0	15 2 0	14 20 0
2	12 15 0	4 5 0	21 17 0	3 8 0	19 21 0	7 4 0	9 14 0	11 1 0	15 2 0	14 20 0
2	12 11 3 1	4 5 4 1	21 20 5 1	3 4 6 1	19 20 7 1	7 3 8 2	9 14 9 1	11 10 10 1	15 11 1 1	14 17 2 1
2	12 15 1	5 5 0	20 17 0	4 8 0	19 21 1	3 4 0	14 14 0	10 1 0	11 2 0	17 20 0
3	12 15 0	5 18 0	20 17 0	4 8 0	19 21 0	3 4 0	14 7 0	10 1 0	11 2 0	17 20 0
3	12 11 4 1	5 8 5 1	20 19 6 1	4 5 7 1	19 15 8 3	3 4 9 1	14 17 10 2	10 9 1 1	11 7 2 1	17 18 3 1
3	11 15 0	8 18 0	19 17 0	5 8 0	15 21 0	4 4 0	17 7 0	9 1 0	7 2 0	18 20 0
4	11 15 0	8 18 0	19 17 0	5 8 0	15 21 0	4 14 0	17 7 0	9 1 0	7 2 0	18 20 0
4	11 15 5 1	8 13 6 1	19 15 7 2	5 8 8 2	15 15 9 1	4 3 10 1	17 18 1 2	9 6 2 1	7 3 3 1	18 19 4 1
4	11 15 1	13 18 0	19 17 1	8 8 0	15 21 1	4 14 1	17 7 1	6 1 0	3 2 0	18 20 1
5	11 15 0	13 18 0	19 17 0	8 16 0	15 21 0	4 14 0	17 7 0	6 1 0	3 2 0	18 20 0
5	11 10 6 1	13 12 7 1	19 15 8 3	8 13 9 2	15 11 10 4	4 3 1 1	17 18 2 2	6 1 3 1	3 2 4 1	18 19 5 3
5	10 15 0	12 18 0	15 17 0	13 16 0	11 21 0	3 14 0	18 7 0	1 1 0	2 2 0	19 20 0
6	10 15 0	12 18 0	15 17 0	13 16 0	11 21 0	3 14 0	18 7 0	1 6 0	2 1 0	19 20 0
6	10 9 7 4	12 13 8 1	15 11 9 4	13 16 10 3	11 12 1 2	3 7 2 1	18 19 3 1	1 6 4 1	2 1 5 1	19 20 6 1
6	9 15 0	13 18 0	11 17 0	16 16 0	12 21 0	7 14 0	19 7 0	6 6 0	1 1 0	20 20 0
7	9 15 0	13 18 0	11 17 0	16 8 0	12 21 0	7 14 0	19 7 0	6 5 0	1 6 0	20 12 0
7	9 14 8 2	13 8 9 4	11 10 10 1	16 13 1 1	12 13 2 4	7 11 3 1	19 15 4 1	6 1 5 1	1 2 6 2	20 21 7 1
7	14 15 0	8 18 0	10 17 0	13 8 0	12 21 1	11 14 0	15 7 0	1 5 0	2 6 0	21 12 0
8	14 15 0	8 18 0	10 17 0	13 8 0	12 21 0	11 14 0	15 7 0	1 5 0	2 6 0	21 12 0
8	14 17 9 1	8 5 10 3	10 9 1 1	13 8 2 1	12 13 3 1	11 10 4 1	15 11 5 1	1 2 6 1	2 3 7 4	21 16 8 2
8	17 15 0	5 18 0	9 17 0	8 8 0	13 21 0	10 14 0	11 7 0	2 5 0	3 6 0	16 12 0
9	17 15 0	5 18 0	9 17 0	8 2 0	13 21 0	10 14 0	11 7 0	2 5 0	3 6 0	16 12 0
9	17 18 10 1	5 4 1 1	9 14 2 1	8 13 3 1	13 16 4 1	10 9 5 1	11 10 6 1	2 3 7 1	3 7 8 1	16 21 9 1
9	18 15 0	4 18 0	14 17 0	13 2 0	16 21 0	9 14 0	10 7 0	3 5 0	7 6 0	21 12 0
10	18 15 0	4 18 0	14 17 0	13 2 0	16 21 0	9 14 0	10 7 0	3 5 0	7 6 0	21 12 0
10	18 19 1 1	4 3 2 1	14 17 3 1	13 12 4 2	16 21 5 1	9 14 6 1	10 9 7 4	3 7 8 3	7 11 9 2	21 20 10 1
10	19 15 0	3 18 0	17 17 0	12 2 0	21 21 0	14 14 0	9 7 0	7 5 0	11 6 0	20 12 0
11	19 15 0	3 18 0	17 4 0	12 2 0	21 16 0	14 17 0	9 7 0	7 5 0	11 6 0	20 12 0
11	19 15 2 1	3 7 3 1	17 14 4 1	12 11 5 1	21 16 6 1	14 9 7 3	9 6 8 2	7 11 9 2	11 10 10 1	20 19 1 4
11	15 15 0	7 18 0	14 4 0	12 2 1	16 16 0	9 17 0	6 7 0	11 5 0	10 6 0	19 12 0
12	15 8 0	7 18 0	14 4 0	12 2 0	16 21 0	9 17 0	6 7 0	11 5 0	10 6 0	19 12 0
12	15 11 3 1	7 11 4 1	14 9 5 1	12 11 6 1	16 21 7 1	9 6 8 4	6 1 9 3	11 10 10 4	10 9 1 1	19 15 2 2
12	11 8 0	7 18 1	14 4 1	12 2 2	21 21 0	6 17 0	1 7 0	10 5 0	9 6 0	15 12 0
13	11 8 0	7 18 0	14 4 0	12 2 0	21 14 0	6 17 0	1 7 0	10 5 0	9 6 0	15 12 0
13	11 12 4 1	7 11 5 1	14 9 6 1	12 13 7 1	21 20 8 1	6 1 9 4	1 2 10 1	10 9 1 4	9 6 2 1	15 11 3 1
13	12 8 0	7 18 2	14 4 2	13 2 0	20 14 0	1 17 0	2 7 0	9 5 0	6 6 0	11 12 0
13	12 8 0	7 18 0	14 4 0	13 2 0	20 14 0	1 17 0	2 7 0	9 5 0	6 15 0	11 12 0
14	12 13 5 1	7 11 6 1	14 9 7 2	13 8 8 1	20 19 9 1	1 2 10 3	2 3 1 1	9 6 2 2	6 1 3 4	11 12 4 1
14	13 8 0	11 18 0	9 4 0	8 2 0	19 14 0	2 17 0	3 7 0	6 5 0	1 15 0	12 12 0
13	13 8 0	11 18 0	9 4 0	8 2 0	19 14 0	2 17 0	3 7 0	6 5 0	1 15 0	12 16 0
15	13 8 6 1	11 10 7 1	9 6 8 1	8 5 9 1	19 18 10 1	2 3 1 4	3 7 2 1	6 1 3 3	1 2 4 2	12 13 5 1
15	8 8 0	10 18 0	6 4 0	5 2 0	18 14 0	3 17 0	7 7 0	1 5 0	2 15 0	13 16 0
13	8 7 0	10 18 0	6 4 0	5 2 0	18 14 0	3 17 0	7 8 0	1 5 0	2 15 0	13 16 0
16	8 13 7 2	10 9 8 1	6 1 9 1	5 4 10 1	18 17 1 1	3 7 2 1	7 11 3 4	1 2 4 1	2 3 5 1	13 16 6 1
16	13 7 0	9 18 0	1 4 0	4 2 0	17 14 0	7 17 0	11 8 0	2 5 0	3 15 0	16 16 0
13	13 7 0	9 18 0	1 4 0	4 2 0	17 14 0	7 17 0	11 8 0	2 5 0	3 15 0	16 1 0
17	13 12 8 1	9 14 9 1	1 2 10 1	4 3 1 1	17 14 2 1	7 11 3 1	11 12 4 1	2 3 5 1	3 7 6 1	16 13 7 1
17	13 7 1	9 18 1	1 4 1	3 2 0	14 14 0	11 17 0	12 8 0	2 5 1	7 15 0	16 1 1
13	13 7 0	9 18 0	1 4 0	3 2 0	14 21 0	11 17 0	12 8 0	2 5 0	7 15 0	16 1 0
18	13 8 9 5	9 14 10 1	1 2 1 1	3 2 2 1	14 17 3 1	11 15 4 1	12 13 5 1	2 2 6 5	7 11 7 1	16 13 8 1
18	8 7 0	14 18 0	1 4 2	3 2 1	17 21 0	15 17 0	13 8 0	2 5 2	11 15 0	16 1 2
13	8 7 0	14 18 0	1 4 0	3 2 0	17 21 0	15 17 0	13 8 0	2 5 0	11 15 0	16 1 0
19	8 5 10 4	14 17 1 1	1 2 2 1	3 2 3 1	17 18 4 1	15 19 5 1	13 8 6 1	2 2 7 5	11 15 8 1	16 13 9 1
19	5 7 0	17 18 0	1 4 3	3 2 2	18 21 0	19 17 0	8 8 0	2 5 3	15 15 0	13 1 0
13	5 7 0	17 18 0	1 4 0	3 2 0	18 21 0	19 17 0	8 6 0	2 5 0	15 20 0	13 1 0
20	5 4 1 1	17 18 2 1	1 2 3 1	3 2 4 1	18 19 5 1	19 20 6 4	8 13 7 2	2 2 8 5	15 19 9 2	13 12 10 1
20	4 7 0	18 18 0	1 4 0	3 2 3	19 21 0	20 17 0	13 6 0	2 5 0	15 20 1	12 1 0
13	4 7 0	18 8 0	1 4 0	3 2 0	19 21 0	20 17 0	13 6 0	2 5 0	15 20 0	12 1 0
21	4 3 2 1	18 19 3 1	1 2 4 1	3 2 5 1	19 20 6 1	20 21 7 4	13 12 8 1	2 2 9 17	15 19 10 1	12 11 1 1
21	4 7 1	19 8 0	1 4 1	3 2 0	20 21 0	21 17 0	12 6 0	2 5 1	15 20 2	11 1 0

13	4 7 0	19 8 0	1 4 0	3 2 0	20 21 0	21 17 0	12 6 0	2 5 0	15 20 0	11 1 0
22	4 3 3 1	19 20 4 2	1 2 5 1	3 2 6 1	20 21 7 1	21 16 8 2	12 11 9 1	2 2 10 17	15 19 1 1	11 7 2 1
22	4 7 2	20 8 0	1 4 2	3 2 1	21 21 0	16 17 0	11 6 0	2 5 2	19 20 0	7 1 0
13	4 7 0	20 8 0	1 4 0	3 2 0	21 14 0	16 17 0	11 6 0	2 5 0	19 20 0	7 1 0
23	4 3 4 1	20 21 5 1	1 2 6 1	3 3 7 17	21 16 8 2	16 13 9 3	11 10 10 1	2 3 1 1	19 20 2 1	7 3 3 1
23	4 7 3	21 8 0	1 4 3	3 2 2	16 14 0	13 17 0	10 6 0	2 5 3	20 20 0	7 1 1
13	4 7 0	21 8 0	1 4 0	3 2 0	16 14 0	13 17 0	10 6 0	2 5 0	20 12 0	7 1 0
24	4 3 5 1	21 16 6 1	1 2 7 1	3 3 8 17	16 13 9 3	13 12 10 2	10 9 1 1	2 3 2 1	20 21 3 2	7 3 4 1
24	4 7 0	16 8 0	1 4 0	3 2 3	13 14 0	12 17 0	9 6 0	2 5 0	21 12 0	7 1 2
13	4 7 0	16 8 0	1 4 0	3 2 0	13 14 0	12 17 0	9 6 0	2 5 0	21 12 0	7 1 0
25	4 3 6 1	16 13 7 1	1 2 8 1	3 3 9 17	13 12 10 1	12 11 1 1	9 6 2 1	2 3 3 1	21 16 4 1	7 3 5 1
25	4 7 1	13 8 0	1 4 1	3 2 0	12 14 0	11 17 0	6 6 0	2 5 1	16 12 0	7 1 3
13	4 7 0	13 8 0	1 4 0	3 2 0	12 14 0	11 17 0	6 20 0	2 5 0	16 12 0	7 1 0
26	4 3 7 1	13 8 8 1	1 6 9 1	3 3 10 17	12 11 1 1	11 15 2 1	6 9 3 1	2 3 4 1	16 13 5 4	7 3 6 1
26	4 7 2	8 8 0	6 4 0	3 2 1	11 14 0	15 17 0	9 20 0	2 5 2	13 12 0	7 1 0
13	4 7 0	8 15 0	6 4 0	3 2 0	11 14 0	15 17 0	9 20 0	2 5 0	13 12 0	7 1 0
27	4 5 8 2	8 13 9 1	6 9 10 2	3 2 1 1	11 10 2 1	15 19 3 1	9 14 4 1	2 1 5 4	13 12 6 1	7 3 7 1
27	5 7 0	13 15 0	9 4 0	2 2 0	10 14 0	19 17 0	14 20 0	1 5 0	12 12 0	3 1 0
13	5 7 0	13 15 0	9 4 0	2 6 0	10 14 0	19 17 0	14 20 0	1 5 0	12 18 0	3 1 0
28	5 8 9 1	13 12 10 1	9 10 1 2	2 1 2 1	10 11 3 5	19 18 4 1	14 17 5 1	1 6 6 2	12 11 7 1	3 2 8 1
28	8 7 0	13 15 1	10 4 0	1 6 0	11 14 0	18 17 0	17 20 0	6 5 0	12 18 1	2 1 0
13	8 7 0	13 15 0	10 4 0	1 6 0	11 14 0	18 17 0	17 20 0	6 5 0	12 18 0	2 1 0
29	8 13 10 4	13 12 1 1	10 11 2 1	1 6 3 1	11 15 4 2	18 17 5 1	17 14 6 5	6 9 7 3	12 11 8 1	2 1 9 1
29	13 7 0	12 15 0	10 4 1	6 6 0	15 14 0	17 17 0	14 20 0	9 5 0	11 18 0	1 1 0
13	13 7 0	12 15 0	10 4 0	6 8 0	15 14 0	17 12 0	14 20 0	9 5 0	11 18 0	1 6 0
30	13 12 1 1	12 11 2 1	10 11 3 1	6 9 4 1	15 19 5 3	17 14 6 1	14 9 7 3	9 10 8 2	11 15 9 1	1 6 10 1
30	13 7 1	12 15 1	11 4 0	9 8 0	19 14 0	17 12 1	14 20 1	10 5 0	15 18 0	6 6 0
13	13 7 0	12 15 0	11 4 0	9 8 0	19 14 0	17 12 0	14 20 0	10 5 0	15 18 0	6 21 0
31	13 12 2 1	12 11 3 1	11 7 4 1	9 10 5 1	19 18 6 1	17 18 7 1	14 9 8 3	10 11 9 1	15 19 10 1	6 9 1 1
31	13 7 2	12 15 2	7 4 0	10 8 0	18 14 0	17 12 2	14 20 2	11 5 0	19 18 0	9 21 0
13	13 7 0	12 15 0	7 4 0	10 8 0	18 14 0	17 12 0	14 20 0	11 5 0	19 18 0	9 21 0
32	13 12 3 1	12 11 4 1	7 3 5 1	10 11 6 1	18 17 7 1	17 14 8 1	14 9 9 3	11 7 10 2	19 18 1 1	9 10 2 1
32	13 7 3	12 15 3	3 4 0	11 8 0	17 14 0	14 12 0	9 20 0	7 5 0	18 18 0	10 21 0
13	13 7 0	12 15 0	3 4 0	11 8 0	17 14 0	14 12 0	9 20 0	7 5 0	18 10 0	10 21 0
33	13 12 4 1	12 11 5 1	3 4 6 1	11 7 7 2	17 14 8 1	14 9 9 3	9 10 10 3	7 3 1 1	18 17 2 1	10 11 3 1
33	13 7 0	12 15 0	4 4 0	7 8 0	14 14 0	9 12 0	10 20 0	3 5 0	17 10 0	11 21 0
13	13 7 0	12 15 0	4 18 0	7 8 0	14 2 0	9 12 0	10 20 0	3 5 0	17 10 0	11 21 0
34	13 12 5 1	12 11 6 1	4 5 7 4	7 3 8 4	14 9 9 1	9 10 10 1	10 11 1 2	3 4 2 1	17 14 3 1	11 15 4 1
34	13 7 1	12 15 1	5 18 0	3 8 0	9 2 0	10 12 0	11 20 0	4 5 0	14 10 0	15 21 0
13	13 7 0	12 15 0	5 18 0	3 8 0	9 2 0	10 12 0	11 20 0	4 5 0	14 10 0	15 21 0
35	13 12 6 1	12 11 7 1	5 8 8 1	3 4 9 1	9 10 10 1	10 11 1 1	11 15 2 1	4 5 3 1	14 9 4 2	15 19 5 1
35	13 7 2	12 15 2	8 18 0	4 8 0	10 2 0	11 12 0	15 20 0	5 5 0	9 10 0	19 21 0
13	13 7 0	12 15 0	8 18 0	4 8 0	10 2 0	11 12 0	15 20 0	5 1 0	9 10 0	19 21 0
36	13 12 7 1	12 12 8 17	8 13 9 1	4 3 10 5	10 11 1 2	11 12 2 1	15 19 3 1	5 4 4 1	9 10 5 1	19 20 6 1
36	13 7 3	12 15 3	8 18 1	3 8 0	10 2 1	11 12 1	19 20 0	4 1 0	9 10 1	20 21 0
13	13 7 0	12 15 0	8 18 0	3 8 0	10 2 0	11 12 0	19 20 0	4 1 0	9 10 0	20 21 0
37	13 12 8 1	12 12 9 17	8 13 10 1	3 7 1 2	10 11 2 2	11 12 3 1	19 20 4 1	4 3 5 4	9 10 6 1	20 21 7 1
37	13 7 0	12 15 0	8 18 2	7 8 0	10 2 2	11 12 2	20 20 0	3 1 0	9 10 2	21 21 0
13	13 7 0	12 15 0	8 18 0	7 8 0	10 2 0	11 12 0	20 16 0	3 1 0	9 10 0	21 14 0
38	13 12 9 1	12 12 10 17	8 13 1 1	7 11 2 4	10 11 3 1	11 12 4 1	20 21 5 1	3 2 6 1	9 10 7 1	21 16 8 2
38	13 7 1	12 15 1	8 18 3	7 8 1	10 2 3	11 12 3	21 16 0	2 1 0	9 10 3	16 14 0
13	13 7 0	12 15 0	8 18 0	7 8 0	10 2 0	11 12 0	21 16 0	2 1 0	9 10 0	16 14 0
39	13 12 10 1	12 11 1 1	8 13 2 1	7 11 3 4	10 11 4 1	11 15 5 2	21 16 6 1	2 1 7 1	9 10 8 1	16 13 9 3
39	12 7 0	11 15 0	8 18 0	7 8 2	10 2 0	15 12 0	16 16 0	1 1 0	9 10 0	13 14 0
13	12 7 0	11 15 0	8 18 0	7 8 0	10 2 0	15 12 0	16 1 0	1 6 0	9 10 0	13 14 0
40	12 11 1 1	11 15 2 1	8 13 3 1	7 11 4 4	10 11 5 1	15 19 6 4	16 13 7 1	1 6 8 1	9 10 9 1	13 12 10 1
40	12 7 1	15 15 0	8 18 1	7 8 3	11 2 0	19 12 0	16 1 1	6 6 0	10 10 0	13 14 1
13	12 7 0	15 5 0	8 18 0	7 8 0	11 2 0	19 12 0	16 1 0	6 16 0	10 15 0	13 14 0
41	12 11 2 1	15 11 3 1	8 13 4 1	7 11 5 4	11 10 6 5	19 20 7 1	16 13 8 1	6 9 9 1	10 9 10 4	13 12 1 1
41	11 7 0	15 5 1	13 18 0	7 8 0	10 2 0	20 12 0	16 1 2	6 16 1	9 15 0	12 14 0
13	11 7 0	15 5 0	13 18 0	7 8 0	10 2 0	20 12 0	16 1 0	6 16 0	9 15 0	12 14 0
42	11 7 3 1	15 11 4 1	13 16 5 1	7 3 6 4	10 9 7 4	20 21 8 1	16 21 9 4	6 9 10 1	9 14 1 2	12 11 2 1
42	7 7 0	15 5 2	16 18 0	3 8 0	9 2 0	20 12 1	21 1 0	6 16 2	14 15 0	11 14 0
13	7 20 0	15 5 0	16 18 0	3 8 0	9 2 0	20 12 0	21 1 0	6 16 0	14 15 0	11 14 0
43	7 11 4 1	15 11 5 1	16 21 6 1	3 4 7 1	9 14 8 4	20 21 9 2	21 21 10 5	6 9 1 1	14 17 2 3	11 10 3 1
43	11 20 0	15 5 3	16 18 1	4 8 0	14 2 0	20 12 2	21 1 1	9 16 0	17 15 0	10 14 0
13	11 20 0	15 5 0	16 18 0	4 8 0	14 2 0	20 12 0	21 1 0	9 16 0	17 15 0	10 14 0
44	11 15 5 1	15 19 6 4	16 21 7 2	4 5 8 1	14 17 9 4	20 19 10 1	21 20 1 1	9 10 2 1	17 18 3 1	10 11 4 5
44	11 20 1	15 5 0	21 18 0	5 8 0	17 2 0	19 12 0	20 1 0	9 16 1	18 15 0	10 14 1
13	11 20 0	15 5 0	21 18 0	5 8 0	17 2 0	19 12 0	20 1 0	9 16 0	18 15 0	10 14 0
45	11 12 6 1	15 11 7 1	21 20 8 1	5 8 9 2	17 18 10 2	19 15 1 2	20 19 2 1	9 10 3 1	18 19 4 1	10 11 5 5
45	12 20 0	11 5 0	20 18 0	8 8 0	17 2 1	15 12 0	19 1 0	9 16 2	18 15 1	10 14 2
13	12 20 0	11 5 0	20 18 0	8 17 0	17 2 0	15 12 0	19 1 0	9 16 0	18 15 0	10 14 0

46	12 13 7 2	11 12 8 1	20 19 9 1	8 13 10 1	17 18 1 2	15 11 2 1	19 18 3 1	9 10 4 1	18 18 5 5	10 11 6 5
46	13 20 0	12 5 0	20 18 1	8 17 1	17 2 2	11 12 0	19 1 1	9 16 3	18 15 2	10 14 3
13	13 20 0	12 5 0	20 18 0	8 17 0	17 2 0	11 12 0	19 1 0	9 16 0	18 15 0	10 14 0
47	13 16 8 1	12 13 9 1	20 19 10 1	8 13 1 1	17 18 2 2	11 12 3 1	19 18 4 1	9 10 5 1	18 18 6 5	10 11 7 5
47	16 20 0	13 5 0	20 18 2	8 17 2	17 2 3	12 12 0	19 1 2	10 16 0	18 15 3	11 14 0
13	16 20 0	13 5 0	20 18 0	8 17 0	17 2 0	12 8 0	19 1 0	10 16 0	18 15 0	11 14 0
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
9984	15 11 5 1	2 3 6 1	14 17 7 2	5 8 8 1	17 17 9 17	8 13 10 1	18 17 1 1	19 18 2 1	11 7 3 1	12 13 4 1
9984	11 7 0	3 4 0	14 17 1	5 20 1	17 21 1	8 12 1	18 1 1	19 18 1	7 5 0	13 16 0
13	11 7 0	3 4 0	14 17 0	5 20 0	17 21 0	8 12 0	18 1 0	19 18 0	7 5 0	13 16 0
9985	11 7 6 1	3 4 7 1	14 17 8 2	5 8 9 1	17 17 10 17	8 13 1 1	18 17 2 1	19 18 3 1	7 3 4 1	13 16 5 1
9985	7 7 0	4 4 0	14 17 2	8 20 0	17 21 2	13 12 0	18 1 2	19 18 2	3 5 0	16 16 0
13	7 16 0	4 10 0	14 17 0	8 20 0	17 21 0	13 12 0	18 1 0	19 18 0	3 5 0	16 14 0
9986	7 11 7 1	4 5 8 4	14 17 9 1	8 13 10 1	17 18 1 1	13 12 2 1	18 19 3 4	19 15 4 2	3 4 5 1	16 13 6 2
9986	11 16 0	5 10 0	17 17 0	8 20 1	18 21 0	12 12 0	19 1 0	15 18 0	4 5 0	13 14 0
13	11 16 0	5 10 0	17 15 0	8 20 0	18 21 0	12 2 0	19 1 0	15 18 0	4 5 0	13 14 0
9987	11 10 8 4	5 8 9 1	17 18 10 1	8 13 1 1	18 19 2 1	12 11 3 1	19 15 4 2	15 11 5 4	4 5 6 1	13 12 7 1
9987	10 16 0	8 10 0	17 15 1	13 20 0	18 21 1	11 2 0	19 1 1	15 18 1	5 5 0	12 14 0
13	10 16 0	8 10 0	17 15 0	13 20 0	18 21 0	11 2 0	19 1 0	15 18 0	5 12 0	12 14 0
9988	10 9 9 4	8 13 10 1	17 18 1 1	13 16 2 1	18 19 3 1	11 7 4 1	19 15 5 2	15 11 6 2	5 8 7 1	12 11 8 1
9988	9 16 0	13 10 0	18 15 0	16 20 0	19 21 0	7 2 0	15 1 0	11 18 0	8 12 0	12 14 1
13	9 16 0	13 10 0	18 15 0	16 20 0	19 21 0	7 2 0	15 1 0	11 18 0	8 12 0	12 14 0
9989	9 14 10 2	13 12 1 1	18 19 2 1	16 21 3 1	19 20 4 1	7 3 5 1	15 11 6 1	11 10 7 1	8 13 8 1	12 11 9 1
9989	14 16 0	12 10 0	19 15 0	21 20 0	20 21 0	3 2 0	15 1 1	10 18 0	13 12 0	11 14 0
13	14 16 0	12 10 0	19 15 0	21 20 0	20 21 0	3 2 0	15 1 0	10 18 0	13 12 0	11 14 0
9990	14 17 1 1	12 11 2 1	19 15 3 1	21 20 4 1	20 19 5 5	3 2 6 1	15 11 7 1	10 9 8 1	13 12 9 1	11 10 10 1
9990	17 16 0	12 10 1	15 15 0	20 20 0	19 21 0	2 2 0	11 1 0	9 18 0	13 12 1	10 14 0
13	17 16 0	12 10 0	15 7 0	20 6 0	19 21 0	2 5 0	11 1 0	9 18 0	13 12 0	10 14 0
9991	17 18 2 1	12 11 3 1	15 11 4 1	20 19 5 1	19 15 6 2	2 3 7 1	11 10 8 1	9 14 9 1	13 12 10 1	10 9 1 1
9991	18 16 0	11 10 0	15 7 1	20 6 1	19 21 1	3 5 0	10 1 0	14 18 0	12 12 0	9 14 0
13	18 16 0	11 10 0	15 7 0	20 6 0	19 21 0	3 5 0	10 1 0	14 18 0	12 17 0	9 14 0
9992	18 19 3 1	11 10 4 1	15 11 5 1	20 19 6 1	19 15 7 2	3 4 8 1	10 9 9 1	14 17 10 2	12 11 1 1	9 14 2 1
9992	18 16 1	10 10 0	15 7 2	20 6 2	19 21 2	4 5 0	9 1 0	17 18 0	11 17 0	14 14 0
13	18 16 0	10 12 0	15 7 0	20 6 0	19 21 0	4 5 0	9 1 0	17 18 0	11 17 0	14 20 0
9993	18 19 4 1	10 11 5 1	15 19 6 4	20 19 7 1	19 19 8 1	4 5 9 1	9 6 10 1	17 18 1 1	11 15 2 1	14 17 3 2
9993	18 16 2	10 12 1	15 7 3	20 6 3	19 21 3	5 5 0	6 1 0	17 18 1	11 17 1	14 20 1
13	18 16 0	10 12 0	15 7 0	20 6 0	19 21 0	5 14 0	6 1 0	17 18 0	11 17 0	14 20 0
9994	18 19 5 1	10 9 6 4	15 11 7 1	20 19 8 1	19 15 9 2	5 8 10 1	6 1 1 1	17 18 2 1	11 10 3 1	14 17 4 1
9994	19 16 0	9 12 0	11 7 0	20 6 0	15 21 0	8 14 0	1 1 0	18 18 0	10 17 0	17 20 0
13	19 16 0	9 12 0	11 7 0	20 6 0	15 21 0	8 14 0	1 5 0	18 2 0	10 17 0	17 20 0
9995	19 20 6 1	9 6 7 2	11 7 8 1	20 21 9 4	15 11 10 4	8 13 1 1	1 2 2 1	18 19 3 1	10 9 4 1	17 18 5 1
9995	20 16 0	6 12 0	7 7 0	21 6 0	11 21 0	13 14 0	2 5 0	19 2 0	9 17 0	18 20 0
13	20 16 0	6 12 0	7 1 0	21 6 0	11 21 0	13 14 0	2 5 0	19 2 0	9 17 0	18 20 0
9996	20 21 7 1	6 1 8 4	7 3 9 2	21 16 10 2	11 12 1 2	13 12 2 1	2 3 3 1	19 15 4 2	9 14 5 1	18 19 6 1
9996	21 16 0	1 12 0	7 1 1	16 6 0	12 21 0	13 14 1	3 5 0	15 2 0	14 17 0	19 20 0
13	21 16 0	1 12 0	7 1 0	16 6 0	12 21 0	13 14 0	3 5 0	15 2 0	14 17 0	19 20 0
9997	21 16 8 1	1 2 9 2	7 3 10 1	16 13 1 1	12 13 2 4	13 8 3 5	3 4 4 1	15 11 5 1	14 17 6 1	19 20 7 1
9997	16 16 0	2 12 0	3 1 0	13 6 0	12 21 1	8 14 0	4 5 0	11 2 0	17 17 0	20 20 0
13	16 20 0	2 12 0	3 1 0	13 6 0	12 21 0	8 14 0	4 5 0	11 2 0	17 10 0	20 8 0
9998	16 21 9 1	2 1 10 5	3 2 1 1	13 12 2 1	12 11 3 5	8 5 4 4	4 5 5 1	11 10 6 1	17 14 7 1	20 19 8 1
9998	21 20 0	1 12 0	2 1 0	12 6 0	11 21 0	8 14 1	5 5 0	10 2 0	14 10 0	19 8 0
13	21 20 0	1 12 0	2 1 0	12 6 0	11 21 0	8 14 0	5 7 0	10 2 0	14 10 0	19 8 0
9999	21 20 10 1	1 6 1 2	2 1 2 1	12 11 3 1	11 15 4 1	8 5 5 4	5 4 6 1	10 9 7 4	14 9 8 4	19 15 9 2
9999	20 20 0	6 12 0	1 1 0	11 6 0	15 21 0	5 14 0	4 7 0	9 2 0	14 10 1	19 8 1
13	20 15 0	6 12 0	1 4 0	11 6 0	15 21 0	5 14 0	4 7 0	9 2 0	14 10 0	19 8 0
10000	20 19 1 1	6 9 2 1	1 2 3 1	11 10 4 1	15 19 5 1	5 4 6 2	4 3 7 1	9 14 8 3	14 17 9 5	19 18 10 4
10000	19 15 0	9 12 0	2 4 0	10 6 0	15 21 1	4 14 0	3 7 0	14 2 0	17 10 0	18 8 0
13	19 15 0	9 12 0	2 4 0	10 6 0	15 21 0	4 14 0	3 7 0	14 2 0	17 10 0	18 8 0

FIM DA SIMULACAO:10001 9574

Vesc[1]=10	=0		
Vesc[2]=10	=0		
Vesc[3]=7	=0		
Vesc[4]=12	=0		
Vesc[5]=9	=0		
Vesc[6]=6	=0		
Vesc[7]=6	=0		
Vesc[8]=8	=0		
Vesc[9]=10	=0		
Vesc[10]=7	=0		
Vesc[11]=0	=0		
Totals = 9485	85	4	0

1.1.2 Format of Data Files Containing Results of Data Analysis

Num.Vehicles:10 Num. Iteracoes Rede NN:100000
Seed:5
Max. Iter. to Dealock:50

Reset dos tramos usados apos: 3

ESTRATEgia= 0

0	AGV:3	NOR:21 NOD:16	TRA:1	ESP:0	OPT:1
1	AGV:6	NOR:10 NOD: 7	TRA:2	ESP:0	OPT:2
2	AGV:2	NOR: 7 NOD: 5	TRA:3	ESP:0	OPT:3
2	AGV:7	NOR: 1 NOD:14	TRA:3	ESP:0	OPT:3
3	AGV:6	NOR: 7 NOD: 4	TRA:2	ESP:0	OPT:2
4	AGV:4	NOR: 2 NOD: 8	TRA:4	ESP:1	OPT:4
5	AGV:8	NOR:15 NOD: 1	TRA:5	ESP:1	OPT:5
5	AGV:9	NOR:18 NOD: 2	TRA:6	ESP:0	OPT:6
6	AGV:4	NOR: 8 NOD:16	TRA:2	ESP:0	OPT:2
6	AGV:8	NOR: 1 NOD: 6	TRA:1	ESP:0	OPT:1
6	AGV:9	NOR: 2 NOD: 1	TRA:1	ESP:0	OPT:1
6	AGV:10	NOR: 6 NOD:20	TRA:6	ESP:1	OPT:6
8	AGV:4	NOR:16 NOD: 8	TRA:2	ESP:0	OPT:2
10	AGV:3	NOR:16 NOD:17	TRA:9	ESP:1	OPT:5
10	AGV:5	NOR:17 NOD:21	TRA:8	ESP:3	OPT:4
10	AGV:6	NOR: 4 NOD:14	TRA:6	ESP:1	OPT:6
11	AGV:1	NOR: 8 NOD:15	TRA:10	ESP:2	OPT:4
11	AGV:5	NOR:21 NOD:16	TRA:1	ESP:0	OPT:1
12	AGV:5	NOR:16 NOD:21	TRA:1	ESP:0	OPT:1
13	AGV:9	NOR: 1 NOD: 6	TRA:7	ESP:0	OPT:1
14	AGV:10	NOR:20 NOD:12	TRA:8	ESP:0	OPT:4
15	AGV:1	NOR:15 NOD: 8	TRA:4	ESP:0	OPT:4
15	AGV:7	NOR:14 NOD: 7	TRA:12	ESP:1	OPT:4
16	AGV:10	NOR:12 NOD:16	TRA:2	ESP:0	OPT:2
17	AGV:5	NOR:21 NOD:14	TRA:5	ESP:0	OPT:5
19	AGV:7	NOR: 7 NOD: 8	TRA:4	ESP:0	OPT:4
19	AGV:9	NOR: 6 NOD:15	TRA:6	ESP:0	OPT:4
20	AGV:2	NOR: 5 NOD:18	TRA:15	ESP:3	OPT:7
22	AGV:5	NOR:14 NOD:21	TRA:5	ESP:0	OPT:5
23	AGV:9	NOR:15 NOD:20	TRA:2	ESP:2	OPT:2
25	AGV:7	NOR: 8 NOD: 6	TRA:6	ESP:0	OPT:6
26	AGV:2	NOR:18 NOD: 8	TRA:6	ESP:0	OPT:6
27	AGV:4	NOR: 8 NOD: 2	TRA:8	ESP:11	OPT:4
27	AGV:9	NOR:20 NOD:12	TRA:4	ESP:0	OPT:4
29	AGV:4	NOR: 2 NOD: 6	TRA:2	ESP:0	OPT:2
29	AGV:6	NOR:14 NOD:17	TRA:19	ESP:0	OPT:1
29	AGV:10	NOR:16 NOD: 1	TRA:7	ESP:6	OPT:7
30	AGV:10	NOR: 1 NOD: 6	TRA:1	ESP:0	OPT:1
32	AGV:9	NOR:12 NOD:18	TRA:4	ESP:1	OPT:4
33	AGV:3	NOR:17 NOD: 4	TRA:11	ESP:12	OPT:7
33	AGV:5	NOR:21 NOD:14	TRA:11	ESP:0	OPT:5
35	AGV:8	NOR: 6 NOD: 5	TRA:19	ESP:10	OPT:5

37	AGV:7	NOR: 6 NOD:20	TRA:10	ESP:2	OPT:6
37	AGV:10	NOR: 6 NOD:21	TRA:7	ESP:0	OPT:7
39	AGV:7	NOR.20 NOD:16	TRA:2	ESP:0	OPT:2
39	AGV:8	NOR: 5 NOD: 1	TRA:4	ESP:0	OPT:4
40	AGV:2	NOR: 8 NOD:15	TRA:4	ESP:10	OPT:4
40	AGV:8	NOR: 1 NOD: 6	TRA:1	ESP:0	OPT:1
40	AGV:9	NOR:18 NOD:10	TRA:4	ESP:4	OPT:4
42	AGV:1	NOR: 8 NOD: 7	TRA:10	ESP:17	OPT:4
45	AGV:4	NOR: 6 NOD: 8	TRA:12	ESP:4	OPT:6
47	AGV:6	NOR:17 NOD:12	TRA:11	ESP:7	OPT:5
....	....	.....	.....	.....	.....
9983	AGV:1	NOR:16 NOD:15	TRA:4	ESP:2	OPT:4
9983	AGV:2	NOR: 6 NOD: 2	TRA:2	ESP:0	OPT:2
9983	AGV:3	NOR:12 NOD:14	TRA:4	ESP:0	OPT:4
9985	AGV:1	NOR:15 NOD: 7	TRA:2	ESP:0	OPT:2
9985	AGV:2	NOR: 2 NOD: 4	TRA:2	ESP:0	OPT:2
9985	AGV:10	NOR:18 NOD:16	TRA:8	ESP:0	OPT:4
9986	AGV:3	NOR:14 NOD:17	TRA:1	ESP:2	OPT:1
9986	AGV:6	NOR: 7 NOD:12	TRA:6	ESP:1	OPT:2
9987	AGV:9	NOR:20 NOD: 5	TRA:9	ESP:0	OPT:5
9990	AGV:3	NOR:17 NOD:15	TRA:3	ESP:1	OPT:3
9990	AGV:4	NOR: 4 NOD:20	TRA:6	ESP:2	OPT:6
9990	AGV:6	NOR:12 NOD: 2	TRA:4	ESP:0	OPT:4
9991	AGV:9	NOR: 5 NOD:12	TRA:3	ESP:1	OPT:3
9992	AGV:2	NOR: 4 NOD:10	TRA:6	ESP:1	OPT:4
9992	AGV:10	NOR:16 NOD:14	TRA:6	ESP:1	OPT:6
9993	AGV:6	NOR: 2 NOD: 5	TRA:3	ESP:0	OPT:3
9994	AGV:7	NOR:15 NOD: 1	TRA:13	ESP:10	OPT:5
9994	AGV:8	NOR: 5 NOD:18	TRA:13	ESP:4	OPT:7
9995	AGV:3	NOR:15 NOD: 7	TRA:2	ESP:3	OPT:2
9997	AGV:1	NOR: 7 NOD:16	TRA:10	ESP:2	OPT:4
9997	AGV:9	NOR:12 NOD:17	TRA:5	ESP:1	OPT:5
9997	AGV:10	NOR:14 NOD:20	TRA:4	ESP:1	OPT:4
9998	AGV:7	NOR: 1 NOD: 5	TRA:4	ESP:0	OPT:4
9999	AGV:1	NOR:16 NOD:20	TRA.2	ESP:0	OPT:2
9999	AGV:3	NOR: 7 NOD: 1	TRA.3	ESP:1	OPT:3

RESULTADOS DA ANALISE DOS DADOS
POR CADA VEICULO:

AGV[1]: NP=954 STR=6484 STE=3516 MDIS=3696
VGood=944 Good=10 NGood=0 Mas=0 Tipo_sol(1,2)= 491

AGV[2]: NP=946 STR=6372 STE=3621 MDIS=3722
VGood=936 Good=10 NGood=0 Mas=0 Tipo_sol(1,2)= 517

AGV[3]: NP=959 STR=6556 STE=3444 MDIS=3756
VGood=952 Good=7 NGood=0 Mas=0 Tipo_sol(1,2)= 493

AGV[4]: NP=909 STR=6352 STE=3639 MDIS=3570
VGood=897 Good=12 NGood=0 Mas=0 Tipo_sol(1,2)= 446

AGV[5]: NP=965 STR=6432 STE=3548 MDIS=3850
VGood=955 Good=9 NGood=1 Mas=0 Tipo_sol(1,2)= 507

AGV[6]: NP=979 STR=6447 STE=3547 MDIS=3733
VGood=972 Good=6 NGood=1 Mas=0 Tipo_sol(1,2)= 509

AGV[7]: NP=965 STR=6450 STE=3549 MDIS=3854
VGood=958 Good=6 NGood=1 Mas=0 Tipo_sol(1,2)= 514

AGV[8]: NP=967 STR=6476 STE=3519 MDIS=3698

VGood=959 Good=8 NGood=0 Mas=0 Tipo_sol(1,2)= 494

AGV[9]: NP=958 STR=6497 STE=3501 MDIS=3793
VGood=947 Good=10 NGood=1 Mas=0 Tipo_sol(1,2)= 499

AGV[10]: NP=972 STR=6474 STE=3524 MDIS=3736
VGood=965 Good=7 NGood=0 Mas=0 Tipo_sol(1,2)= 504

AGV[11]: NP=0 STR=0 STE=0 MDIS=0
VGood=0 Good=0 NGood=0 Mas=0 Tipo_sol(1,2)= 0

TOTAIS GLOBAIS:
cont_casos =9574
Soma_TT=64540 Soma_TE=35408 Soma=99948
Soma_MD=37408
METRICA=2.67183
Som_VGo=9485 Som_Goo=85 Som_NGo=4
Som_Mas=0
Som_Tipo12= 4974
ARRAY_HIS [0...200]:
[0]: HIS = 3151 ACMP = 3.015 MAX = 8 CASOS = 0
[1]: HIS = 725 ACMP = 4.050 MAX = 8 CASOS = 503
[2]: HIS = 648 ACMP = 4.255 MAX = 8 CASOS = 917
[3]: HIS = 505 ACMP = 4.642 MAX = 8 CASOS = 697
[4]: HIS = 501 ACMP = 4.387 MAX = 8 CASOS = 928
[5]: HIS = 394 ACMP = 4.520 MAX = 8 CASOS = 615
[6]: HIS = 415 ACMP = 4.123 MAX = 8 CASOS = 578
[7]: HIS = 356 ACMP = 4.295 MAX = 8 CASOS = 524
[8]: HIS = 323 ACMP = 4.248 MAX = 8 CASOS = 525
[9]: HIS = 291 ACMP = 4.326 MAX = 8 CASOS = 467
[10]: HIS = 256 ACMP = 4.344 MAX = 8 CASOS = 420
[11]: HIS = 199 ACMP = 4.266 MAX = 8 CASOS = 348
[12]: HIS = 171 ACMP = 4.561 MAX = 8 CASOS = 357
[13]: HIS = 174 ACMP = 4.511 MAX = 8 CASOS = 312
[14]: HIS = 143 ACMP = 4.406 MAX = 8 CASOS = 246
[15]: HIS = 136 ACMP = 4.654 MAX = 8 CASOS = 218
[16]: HIS = 116 ACMP = 4.388 MAX = 8 CASOS = 175
[17]: HIS = 102 ACMP = 4.284 MAX = 8 CASOS = 174
[18]: HIS = 101 ACMP = 4.208 MAX = 8 CASOS = 158
[19]: HIS = 75 ACMP = 4.013 MAX = 8 CASOS = 137
[20]: HIS = 64 ACMP = 4.484 MAX = 8 CASOS = 147
[21]: HIS = 59 ACMP = 4.542 MAX = 7 CASOS = 120
[22]: HIS = 56 ACMP = 4.393 MAX = 8 CASOS = 94
[23]: HIS = 55 ACMP = 4.618 MAX = 7 CASOS = 85
[24]: HIS = 61 ACMP = 4.656 MAX = 8 CASOS = 67
[25]: HIS = 44 ACMP = 4.523 MAX = 7 CASOS = 61
[26]: HIS = 43 ACMP = 4.698 MAX = 8 CASOS = 53
[27]: HIS = 40 ACMP = 4.300 MAX = 8 CASOS = 56
[28]: HIS = 32 ACMP = 4.625 MAX = 8 CASOS = 54
[29]: HIS = 32 ACMP = 4.250 MAX = 7 CASOS = 58
[30]: HIS = 19 ACMP = 4.158 MAX = 7 CASOS = 47
[31]: HIS = 23 ACMP = 4.435 MAX = 7 CASOS = 39
[32]: HIS = 28 ACMP = 4.786 MAX = 8 CASOS = 40
[33]: HIS = 25 ACMP = 4.240 MAX = 8 CASOS = 23
[34]: HIS = 19 ACMP = 4.526 MAX = 7 CASOS = 28
[35]: HIS = 25 ACMP = 4.640 MAX = 8 CASOS = 31
[36]: HIS = 13 ACMP = 4.231 MAX = 8 CASOS = 29
[37]: HIS = 18 ACMP = 4.111 MAX = 6 CASOS = 23
[38]: HIS = 6 ACMP = 3.833 MAX = 5 CASOS = 17
[39]: HIS = 7 ACMP = 4.000 MAX = 6 CASOS = 22
[40]: HIS = 5 ACMP = 4.400 MAX = 7 CASOS = 14
[41]: HIS = 5 ACMP = 2.600 MAX = 4 CASOS = 19
[42]: HIS = 11 ACMP = 3.909 MAX = 7 CASOS = 12
[43]: HIS = 5 ACMP = 3.600 MAX = 6 CASOS = 16
[44]: HIS = 5 ACMP = 4.600 MAX = 6 CASOS = 3
[45]: HIS = 7 ACMP = 5.571 MAX = 8 CASOS = 8
[46]: HIS = 6 ACMP = 5.000 MAX = 7 CASOS = 6
[47]: HIS = 3 ACMP = 3.333 MAX = 6 CASOS = 3
[48]: HIS = 2 ACMP = 3.000 MAX = 4 CASOS = 4
[49]: HIS = 1 ACMP = 3.000 MAX = 3 CASOS = 7
[50]: HIS = 2 ACMP = 4.500 MAX = 6 CASOS = 7
[51]: HIS = 5 ACMP = 4.600 MAX = 7 CASOS = 2
[52]: HIS = 1 ACMP = 7.000 MAX = 7 CASOS = 5
[53]: HIS = 3 ACMP = 5.000 MAX = 7 CASOS = 4
[54]: HIS = 6 ACMP = 4.667 MAX = 8 CASOS = 0
[55]: HIS = 4 ACMP = 4.250 MAX = 5 CASOS = 1
[56]: HIS = 1 ACMP = 6.000 MAX = 6 CASOS = 3
[57]: HIS = 3 ACMP = 6.000 MAX = 7 CASOS = 4
[58]: HIS = 4 ACMP = 5.250 MAX = 7 CASOS = 3
[59]: HIS = 1 ACMP = 7.000 MAX = 7 CASOS = 4
[60]: HIS = 3 ACMP = 5.667 MAX = 8 CASOS = 2
[61]: HIS = 2 ACMP = 5.500 MAX = 7 CASOS = 1
[62]: HIS = 1 ACMP = 8.000 MAX = 8 CASOS = 5
[63]: HIS = 1 ACMP = 4.000 MAX = 4 CASOS = 1
[64]: HIS = 2 ACMP = 5.500 MAX = 6 CASOS = 3
[65]: HIS = 2 ACMP = 7.000 MAX = 8 CASOS = 3
[66]: HIS = 1 ACMP = 5.000 MAX = 5 CASOS = 1
[67]: HIS = 1 ACMP = 6.000 MAX = 6 CASOS = 1
[68]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 2
[69]: HIS = 1 ACMP = 5.000 MAX = 5 CASOS = 1
[70]: HIS = 2 ACMP = 4.000 MAX = 4 CASOS = 2
[71]: HIS = 1 ACMP = 5.000 MAX = 5 CASOS = 2
[72]: HIS = 5 ACMP = 4.600 MAX = 6 CASOS = 0
[73]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 2
[74]: HIS = 1 ACMP = 7.000 MAX = 7 CASOS = 4
[75]: HIS = 2 ACMP = 5.000 MAX = 5 CASOS = 0
[76]: HIS = 1 ACMP = 4.000 MAX = 4 CASOS = 2
[77]: HIS = 2 ACMP = 4.500 MAX = 6 CASOS = 1

[78]: HIS = 2 ACMP = 3.500 MAX = 5 CASOS = 2
[79]: HIS = 2 ACMP = 5.000 MAX = 7 CASOS = 0
[80]: HIS = 1 ACMP = 4.000 MAX = 4 CASOS = 5
[81]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[82]: HIS = 2 ACMP = 3.500 MAX = 4 CASOS = 1
[83]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 2
[84]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[85]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[86]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 2
[87]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[88]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[89]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[90]: HIS = 2 ACMP = 4.000 MAX = 4 CASOS = 0
[91]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[92]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[93]: HIS = 1 ACMP = 2.000 MAX = 2 CASOS = 0
[94]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 2
[95]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[96]: HIS = 1 ACMP = 3.000 MAX = 3 CASOS = 0
[97]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[98]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[99]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[100]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[101]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[102]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[103]: HIS = 1 ACMP = 2.000 MAX = 2 CASOS = 0
[104]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[105]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[106]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[107]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[108]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[109]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[110]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[111]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[112]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[113]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[114]: HIS = 1 ACMP = 4.000 MAX = 4 CASOS = 0
[115]: HIS = 1 ACMP = 5.000 MAX = 5 CASOS = 0
[116]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[117]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[118]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[119]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[120]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[121]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0

....

[166]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[167]: HIS = 1 ACMP = 3.000 MAX = 3 CASOS = 0
[168]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[169]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[170]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 1
[171]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0

....

[199]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0
[200]: HIS = 0 ACMP = 0.000 MAX = 0 CASOS = 0

1.2 Code Listing

```
    /*****           DAT_AL10.c  5/09/96          *****/

/*********          NAO SEPARA CASOS NAO RESOLVIDOS (>=200)
                DOS CASOS RESOLVIDOS EM menos de 200 iteracoes *****/

#include "FUN_ALLR.c"

static int flag_inicio = (0); /* permite identificar a primeira (inicio)*/
                              /*  especificacao de dados              */
static char flag_veiculo = ('M');

static int InitFlag = (0);

static long int Num_Iter = (0);

static long int tpo_abs = (0);

static int Semente = (1);

static int Lock_step = (0);
static int DeadV2_Cont = (0);

static long int DeadV_Cont = (0);
static long int flag_DCont [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int     flag_D2Cont[11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int NotSolved  [11] =(0,0,0,0,0,0,0,0,0,0,0);
static long int NotSolv = (0);

static int flag_Vcont [11] = (0,0,0,0,0,0,0,0,0,0,0);
static int flag_NOS0 [11] = (0,0,0,0,0,0,0,0,0,0,0);
static int flag_esp [11] = (0,0,0,0,0,0,0,0,0,0,0);

static int Lim_esp = (0);

static int No_lixo = (0);

static int MAQ_NOS [16]= (1,2,4,5,6,7,8,10,12,14,15,16,17,18,20,21);

static long int ARRAY_HIS [201];    /* num. casos com dif. perc.real-opt */
static long int ARRAY_CPM [201];
static int ARRAY_MAX [201];
static long int ARRAY_CASOS [201]; /* num. casos resolvidos com comp.respectivos */


static float ARRAY_ACPM [201];

static int DIF_TRTEOP = (0);

main ()
{

    int No_um, No_dois, No_tres, No_quatr, No_cinc, No_seis, No_sete, tpo_lixo;
    int Ve_Pri, Tmp_esp, Tpo_sol, Tpo_acu, NVei;
    long int Tempo, Itera, TTempo, mai_NIt, mai_tempo, mai_j;

    int mai_i, mai_NV, mai_Seed, mai_DLok, mai_Lesp, mai_ESTRA;

    int NORIG [11], NODEST [11], NOSEG [11], NOPRET [11], NOTEMP [11], NOTARGET [11];
    int NOCURR [11], TPOSOL [11], C_PASSO [11], TRA_MIN [11];
    int NOSEG1 [11], NODEST1[11], FLAG_TSOL [11];
    long int PERCURSO [11], SOMA_TRA [11], SOMA_ESP [11], C_GOOD[11], C_NGOOD[11], C_MAS[11];
    long int C_VGOOD[11], TEMPO_ABS, TRA_USA [11], TEM_ESP [11], Soma_TT, Soma_TE, Soma;
    long int SOMA_NDIS[11], Soma_ND, Som_Mas, Som_NGo, Som_Goo, Som_VGo, Som_Tipo12;
    long int C_TPOSOL [11];

    FILE *ler_001; /*** FILE pointer a ser usado ficheiro de resultados**/
    FILE *esc_001; /*** FILE pointer a ser usado ficheiro de resultados**/
    FILE *esc_002; /*** FILE pointer a ser usado ficheiro de resultados**/

    tpo_lixo = 0;
    TEMPO_ABS = 0;
    Soma_TT   = 0;
    Soma_TE   = 0;
    Soma_ND   = 0;
    Soma      = 0;
    Som_Goo   = 0;
    Som_NGo   = 0;
    Som_VGo   = 0;
    Som_Mas   = 0;
    Som_Tipo12= 0;
    mai_NV    = 0;
    mai_Seed  = 0;
    mai_DLok  = 0;
    mai_Lesp  = 0;
    mai_NIt   = 0;
    mai_tempo = 0;
    DIF_TRTEOP = 0;
    DeadV_Cont = 0;
    DeadV2_Cont = 0;
    NotSolv = 0;

    for (mai_i = 0; mai_i < 11; mai_i ++){
          flag_DCont [mai_i] = 0;
          flag_D2Cont[mai_i] = 0;
          NotSolved  [mai_i] = 0;
          NORIG      [mai_i] = 0;
          NODEST     [mai_i] = 0;
          NOSEG      [mai_i] = 0;
          NOSEG1     [mai_i] = 0;
          NODEST1    [mai_i] = 0;
          NOPRET     [mai_i] = 0;
          NOTEMP     [mai_i] = 0;
```

```
        NOTARGET[mai_i] = 0;
        TEM_ESP [mai_i] = 0;
        NOCURR    [mai_i] = 0;
        TPOSOL    [mai_i] = 0;
        FLAG_TSOL[mai_i]= 0;
        TRA_USA [mai_i] = 0;
        TRA_MIN [mai_i] = 0;
        PERCURSO [mai_i]= 0;
        SOMA_TRA [mai_i]= 0;
        SOMA_ESP [mai_i]= 0;
        SOMA_MDIS[mai_i]= 0;
        C_TPOSOL [mai_i]= 0;
        C_PASSO [mai_i] = 0;
        C_GOOD    [mai_i] = 0;
        C_NGOOD [mai_i]     = 0;
        C_VGOOD [mai_i]     = 0;
        C_MAS     [mai_i] = 0;
}
for (mai_i =0; mai_i<201; mai_i ++){
        ARRAY_HIS [mai_i] = 0;
        ARRAY_CPM [mai_i] = 0;
        ARRAY_MAX [mai_i] = 0;
        ARRAY_ACPM [mai_i] = 0;
        ARRAY_CASOS [mai_i] = 0;
}

ler_001 = fopen ("filetes1.nnr", "r");
esc_001 = fopen ("res1.nnr", "a");
esc_002 = fopen ("res11.nnr", "a");
if ((ler_001 == NULL)||(esc_001 == NULL)||(esc_002 == NULL)){
          printf (" nao consegui abrir ficheiros ");
}

fscanf (ler_001,"\n Num.Vehicles:%d\t Num. Iteracoes Rede NN:%\ld", &NVei, &Itera);
mai_NV = NVei;
mai_NIt = Itera;

fprintf (esc_001,"\n Num.Vehicles:%d\t Num. Iteracoes Rede NN:%\ld", mai_NV, mai_NIt);

mai_tempo = mai_NIt/mai_NV;

fscanf (ler_001,"\n Seed:%d\n Max. Iter. to Dealock:%d\n", &mai_Seed, &mai_DLok);
fprintf (esc_001,"\n Seed:%d\n Max. Iter. to Dealock:%d\n", mai_Seed, mai_DLok);

fscanf (ler_001,"\n Reset dos tramos usados apos: %d\n", &mai_Lesp);
fprintf (esc_001,"\n Reset dos tramos usados apos: %d\n",mai_Lesp);

fscanf (ler_001,"\n ESTRATEgia= %d\n",&mai_ESTRA);
fprintf (esc_001,"\n ESTRATEgia= %d\n",mai_ESTRA);

/* ler dados do ficheiro */

for (mai_j = 0; mai_j <= mai_tempo; mai_j ++){

      fscanf (ler_001,"%ld\t", &Tempo);

TEMPO_ABS = Tempo;

if (tpo_lixo == 100){
    printf ("\n%ld\n", TEMPO_ABS);
    tpo_lixo = 0;
}
tpo_lixo ++;

for (mai_i = 0; mai_i < mai_NV; mai_i ++){

    fscanf (ler_001,"%2d %2d %2d", &No_um, &No_dois, &Ve_Pri);
    fscanf (ler_001," %2d\t", &Tpo_sol);
    if (C_PASSO [mai_i] == 0){
          C_TPOSOL [mai_i] = C_TPOSOL [mai_i] +1;
          NORIG [mai_i] = No_um;
    }
    NOCURR [mai_i] = No_um;
    NOPRET [mai_i] = No_dois;
    TPOSOL [mai_i] = Tpo_sol;
}

fscanf (ler_001,"%ld\t", &Tempo);

for (mai_i = 0; mai_i < mai_NV; mai_i ++){

    fscanf (ler_001,"%2d %2d %2d\t", &No_tres, &No_quatr, &Tmp_esp);

    NOSEG [mai_i] = No_tres;
    NODEST[mai_i] = No_quatr;
              if (C_PASSO [mai_i] == 0){
                    NOTARGET[mai_i] = NODEST [mai_i];
                    TRA_MIN [mai_i] = (no_dist (NORIG[mai_i],NOTARGET[mai_i]))/10;
                    if (TRA_MIN [mai_i] == 0){
                        printf ("erro na contagem do percurso minimo");
                        fprintf (esc_001,"erro na contagem do percurso minimo");
                        fprintf (esc_001,"\n%ld\t AGV:%d\t NOR:%2d NOD:%2d\t", TEMPO_ABS, mai_i+1, NORIG[mai_i],
NOTARGET[mai_i]);
                        fprintf (esc_001,"TRA:%ld\t ESP:%ld\t OPT:%d\n",
TRA_USA[mai_i],TEM_ESP[mai_i],TRA_MIN[mai_i]);
                    }
                    FLAG_TSOL[mai_i] = 0;
              }
          }

          fscanf (ler_001,"%d\t", &Tpo_acu);

          for (mai_i = 0; mai_i < mai_NV; mai_i ++){

              fscanf (ler_001,"%2d %2d %2d\t", &No_cinc, &No_seis, &No_sete);

              NOSEG1 [mai_i] = No_cinc;
              NODEST1 [mai_i]= No_seis;
```

```
    NOTEMP [mai_i] = No_sete;
    C_PASSO [mai_i] = C_PASSO [mai_i] + 1;
}


/* analisar dados do ficheiro */

for (mai_i = 0; mai_i < mai_NV; mai_i ++){

    /* chegou ao fim do ficheiro com percursos em aberto ?*/
    if ((TEMPO_ABS==mai_tempo)&&(NOSEG[mai_i]!=NOTARGET[mai_i])){
              TRA_USA [mai_i] = 0;
              TEM_ESP[mai_i] = 0;
              TRA_MIN [mai_i] = 0;
              C_PASSO [mai_i] = 0;
    }
    /* ultrapassou o tempo limite de 199 iteracoes ? */
    else if (C_PASSO[mai_i]>199){
          cont_casos ++;

          /* parou ? */
          if (NOCURR [mai_i] == NOSEG [mai_i]){
              TEM_ESP [mai_i] = TEM_ESP [mai_i] +1;
          }
          /* movimentou-se ?*/
          else if((NOSEG[mai_i]==NOPRET[mai_i])&&(NOSEG[mai_i]!=NOCURR[mai_i])){
              TRA_USA [mai_i] = TRA_USA [mai_i] +1;
          }

          fprintf (esc_001,"\n%ld\t AGV:%d\t NOR:%2d NOD:%2d\t", TEMPO_ABS, mai_i+1, NORIG[mai_i],
NOTARGET[mai_i]);
          fprintf (esc_001,"TRA:%ld\t ESP:%ld\t OPT:%d\t SOL.MA\n",
TRA_USA[mai_i],TEM_ESP[mai_i],TRA_MIN[mai_i]);

          PERCURSO [mai_i] = PERCURSO [mai_i] +1;
          SOMA_TRA [mai_i] = SOMA_TRA [mai_i] + TRA_USA [mai_i];
          SOMA_ESP [mai_i] = SOMA_ESP [mai_i] + TEM_ESP [mai_i];
          SOMA_MDIS[mai_i] = SOMA_MDIS [mai_i] + TRA_MIN [mai_i];

          DIF_TRTEOP = 200;

          ARRAY_HIS [DIF_TRTEOP] = ARRAY_HIS [DIF_TRTEOP] +1;
          ARRAY_CPM [DIF_TRTEOP] = ARRAY_CPM [DIF_TRTEOP] + TRA_MIN [mai_i];
          if (TRA_MIN[mai_i] > ARRAY_MAX [DIF_TRTEOP]){
              ARRAY_MAX [DIF_TRTEOP] = TRA_MIN [mai_i];
          }
          ARRAY_CASOS [DIF_TRTEOP] = ARRAY_CASOS [DIF_TRTEOP] +1;

          DIF_TRTEOP = 0;
          TRA_USA [mai_i] = 0;
          TEM_ESP [mai_i] = 0;
          TRA_MIN [mai_i] = 0;
          if (FLAG_TSOL [mai_i] == 1){
              C_TPOSOL [mai_i] = C_TPOSOL [mai_i] -1;
          }
          FLAG_TSOL [mai_i] = 0;

          C_MAS [mai_i] = C_MAS [mai_i] +1;
          C_PASSO [mai_i] = 0;
    }
    /* NAO ultrapassou o tempo limite de 199 iteracoes ? */
    else if (C_PASSO[mai_i]<200){

          /* parou ? */
          if (NOCURR [mai_i] == NOSEG [mai_i]){
              TEM_ESP [mai_i] = TEM_ESP [mai_i] +1;
          }
          /* movimentou-se ?*/
          else if((NOSEG[mai_i]==NOPRET[mai_i])&&(NOSEG[mai_i]!=NOCURR[mai_i])){
              TRA_USA [mai_i] = TRA_USA [mai_i] +1;
          }

          if ((TPOSOL [mai_i] != 1) && (TPOSOL [mai_i] != 2)){
              FLAG_TSOL [mai_i] = 1;
          }

          /* chegou ao no destino ? */
          if (NOSEG[mai_i]==NOTARGET[mai_i]){

              cont_casos ++;
              fprintf (esc_001,"\n%ld\t AGV:%d\t NOR:%2d NOD:%2d\t", TEMPO_ABS, mai_i+1, NORIG[mai_i],
NOTARGET[mai_i]);
              fprintf (esc_001,"TRA:%ld\t ESP:%ld\t
OPT:%d\n",TRA_USA[mai_i],TEM_ESP[mai_i],TRA_MIN[mai_i]);

              PERCURSO [mai_i] = PERCURSO [mai_i] +1;
              SOMA_TRA [mai_i] = SOMA_TRA [mai_i] + TRA_USA [mai_i];
              SOMA_ESP [mai_i] = SOMA_ESP [mai_i] + TEM_ESP [mai_i];
              SOMA_MDIS[mai_i] = SOMA_MDIS [mai_i] + TRA_MIN [mai_i];

              DIF_TRTEOP = (TRA_USA [mai_i] + TEM_ESP [mai_i]) - TRA_MIN [mai_i];

              ARRAY_HIS [DIF_TRTEOP] = ARRAY_HIS [DIF_TRTEOP] +1;
              ARRAY_CPM [DIF_TRTEOP] = ARRAY_CPM [DIF_TRTEOP] + TRA_MIN [mai_i];
              if (TRA_MIN[mai_i] > ARRAY_MAX [DIF_TRTEOP]){
                    ARRAY_MAX [DIF_TRTEOP] = TRA_MIN [mai_i];
              }
              ARRAY_CASOS [TRA_USA[mai_i]+TEM_ESP[mai_i]] = ARRAY_CASOS [TRA_USA[mai_i]+TEM_ESP[mai_i]]
+1;

              DIF_TRTEOP = 0;
              TRA_USA [mai_i] = 0;
              TEM_ESP [mai_i] = 0;
              TRA_MIN [mai_i] = 0;
              if (FLAG_TSOL [mai_i] == 1){
                    C_TPOSOL [mai_i] = C_TPOSOL [mai_i] -1;
              }
              FLAG_TSOL [mai_i] = 0;
```

```
                        /* em tempo < 50 */
                        if(C_PASSO[mai_i]<50){
                              C_VGOOD [mai_i] = C_VGOOD [mai_i] +1;
                        }
                        /* em tempo >= 50 e < 100 */
                        else if ((C_PASSO[mai_i] > 49)&&(C_PASSO[mai_i] < 100)){
                              C_GOOD [mai_i] = C_GOOD [mai_i] +1;
                        }
                        /* em tempo > 100 e < 200 */
                        else if ((C_PASSO [mai_i]>99)&&(C_PASSO [mai_i]<200)){
                              C_NGOOD [mai_i] = C_NGOOD [mai_i] +1;
                        }

                        C_PASSO [mai_i] = 0;

                  }
            }

      }
}

fprintf (esc_001,"\nRESULTADOS DA ANALISE DOS DADOS\n");
fprintf (esc_001,"POR CADA VEICULO:\n");
fprintf (esc_002,"\nRESULTADOS DA ANALISE DOS DADOS\n");
fprintf (esc_002,"POR CADA VEICULO:\n");
    printf ("\nRESULTADOS DA ANALISE DOS DADOS\n");
    printf ("POR CADA VEICULO:\n");

    for (mai_i = 0; mai_i < 11; mai_i ++){

          fprintf (esc_001,"\nAGV[%d]: NP=%ld\t STR=%ld\t STE=%ld\t MDIS=%ld\n",mai_i+1,
PERCURSO[mai_i],SOMA_TRA[mai_i], SOMA_ESP[mai_i],SOMA_MDIS[mai_i]);
          fprintf (esc_002,"\nAGV[%d]: NP=%ld\t STR=%ld\t STE=%ld\t MDIS=%ld\n",mai_i+1,
PERCURSO[mai_i],SOMA_TRA[mai_i], SOMA_ESP[mai_i],SOMA_MDIS[mai_i]);
          printf ("\nAGV[%d]: NP=%ld\t STR=%ld\t STE=%ld\t MDIS=%ld\n",mai_i+1, PERCURSO[mai_i],SOMA_TRA[mai_i],
SOMA_ESP[mai_i],SOMA_MDIS[mai_i]);
          fprintf (esc_001,"VGood=%ld\t Good=%ld\t NGood=%ld\t
Mas=%ld\t",C_VGOOD[mai_i],C_GOOD[mai_i],C_NGOOD[mai_i],C_MAS[mai_i]);
          fprintf (esc_002,"VGood=%ld\t Good=%ld\t NGood=%ld\t
Mas=%ld\t",C_VGOOD[mai_i],C_GOOD[mai_i],C_NGOOD[mai_i],C_MAS[mai_i]);
          printf ("VGood=%ld\t Good=%ld\t NGood=%ld\t
Mas=%ld\t",C_VGOOD[mai_i],C_GOOD[mai_i],C_NGOOD[mai_i],C_MAS[mai_i]);
          fprintf (esc_001,"Tipo_sol(1,2)= %ld\n",C_TPOSOL [mai_i]);
          fprintf (esc_002,"Tipo_sol(1,2)= %ld\n",C_TPOSOL [mai_i]);
          printf ("Tipo_sol(1,2)= %ld\n",C_TPOSOL [mai_i]);

          Soma_TT = Soma_TT + SOMA_TRA [mai_i];
          Soma_TE = Soma_TE + SOMA_ESP [mai_i];
          Soma_MD = Soma_MD + SOMA_MDIS[mai_i];
          Som_Mas = Som_Mas + C_MAS    [mai_i];
          Som_NGo = Som_NGo + C_NGOOD  [mai_i];
          Som_Goo = Som_Goo + C_GOOD   [mai_i];
          Som_VGo = Som_VGo + C_VGOOD  [mai_i];
          Som_Tipo12 = Som_Tipo12 + C_TPOSOL [mai_i];
    }

    Soma = Soma_TT + Soma_TE;

    fprintf (esc_001,"\nTOTAIS GLOBAIS:\n");
    fprintf (esc_001,"cont_casos =%ld\n",cont_casos);
    fprintf (esc_001,"Soma_TT=%ld\t Soma_TE=%ld\t Soma=%ld\n",Soma_TT, Soma_TE, Soma);
    fprintf (esc_001,"Soma_MD=%ld\nMETRICA=%.5f\n", Soma_MD, (float)(((float)Soma)/((float)Soma_MD)));
    fprintf (esc_002,"\nTOTAIS GLOBAIS:\n");
    fprintf (esc_002,"cont_casos =%ld\n",cont_casos);
    fprintf (esc_002,"Soma_TT=%ld\t Soma_TE=%ld\t Soma=%ld\n",Soma_TT, Soma_TE, Soma);
    fprintf (esc_002,"Soma_MD=%ld\nMETRICA=%.5f\n", Soma_MD, (float)(((float)Soma)/((float)Soma_MD)));
    printf ("\nTOTAIS GLOBAIS:\n");
    printf ("Soma_TT=%ld\t Soma_TE=%ld\t Soma=%ld\n",Soma_TT, Soma_TE, Soma);
    printf ("Soma_MD=%ld\nMETRICA=%.5f\n",Soma_MD, (float)(((float)Soma)/((float)Soma_MD)));

    fprintf (esc_001,"Som_VGo=%ld\t Som_Goo=%ld\t Som_NGo=%ld\n", Som_VGo, Som_Goo, Som_NGo);
    fprintf (esc_001,"Som_Mas=%ld\n",Som_Mas);

    fprintf (esc_002,"Som_VGo=%ld\t Som_Goo=%ld\t Som_NGo=%ld\n", Som_VGo, Som_Goo, Som_NGo);
    fprintf (esc_002,"Som_Mas=%ld\n",Som_Mas);

    printf ("Som_VGo=%ld\t Som_Goo=%ld\t Som_NGo=%ld\n", Som_VGo, Som_Goo, Som_NGo);
    printf ("Som_Mas=%ld\n",Som_Mas);

    fprintf (esc_001,"Som_Tipo12= %ld\n", Som_Tipo12);
    fprintf (esc_002,"Som_Tipo12= %ld\n", Som_Tipo12);
    printf ("Som_Tipo12= %ld\n", Som_Tipo12);

    fprintf (esc_001,"ARRAY_HIS [0...200]:\n");
    fprintf (esc_002,"ARRAY_HIS [0...200]:\n");

    for (mai_i = 0; mai_i<201; mai_i++){
          if (ARRAY_HIS [mai_i]>0){
              ARRAY_ACPM [mai_i] = (float)(((float)ARRAY_CPM[mai_i])/((float)ARRAY_HIS[mai_i]));
          }
          fprintf (esc_001,"[%d]: HIS = %ld\t ACMP = %.3f \t MAX = %d\t CASOS = %ld\n",mai_i,ARRAY_HIS
[mai_i],ARRAY_ACPM[mai_i],ARRAY_MAX[mai_i],ARRAY_CASOS[mai_i]);
          fprintf (esc_002,"[%d]: HIS = %ld\t ACMP = %.3f \t MAX = %d\t CASOS = %ld\n",mai_i,ARRAY_HIS
[mai_i],ARRAY_ACPM[mai_i],ARRAY_MAX[mai_i],ARRAY_CASOS[mai_i]);
    }

    fclose (esc_001);
    fclose (esc_002);
    fclose (ler_001);

}
```

Appendix E

Graphs of *'K'* ratio

- DISTRIBUTION of K-RATIO, represents the number of paths that can be included in one of the classes defined for the *'K'* ratio. The *'K'* ratio represents the value obtained when dividing the time required to execute a path, using one of the strategies tested, by the time of the path along the minimum geometric distance without traffic congestion. The classes considered for the *'K'* ratio are represented by the values ranging from 1.0 (minimum) to 200.0 (maximum), with increments of tenth (0.1). This distribution is represented as an accumulated (%) distribution. The *'K'* value for a path (i) is calculated by the following expression, and included in the nearest of the *'K'* classes considered:

$$k_i = \frac{TimePath_i}{T.Min.Path_i}$$

Where,

TimePath: is the total time (travel + waiting) to execute path *'i'*

T.Min.Path: is the time to travel the path along minimum geometric distance without traffic congestion, linking the origin and destination nodes as in path *'i'*.

The following figures represent the set of graphs containing the curves obtained when testing the various strategies with the number of vehicles varying from one to ten. First the graphs containing all strategies (Figure E- 1 to Figure E- 10) are presented, followed by the graphs with complete 'look-ahead' (Figure E-11 to Figure E-20), limited 'look-ahead' (Figure E-21 to Figure E-30), and no 'look-ahead' strategies (Figure E-31 to Figure E-40).

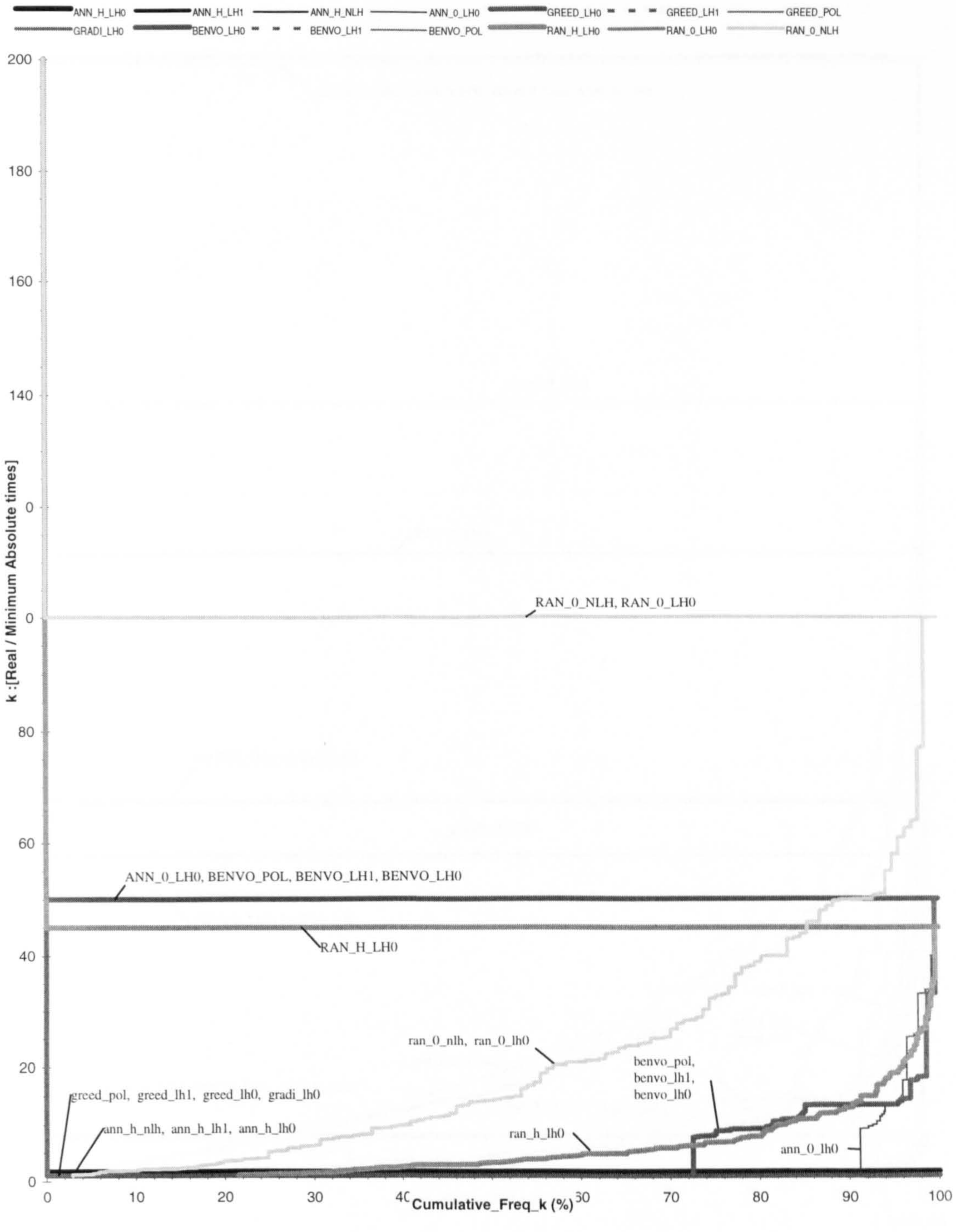


Figure E- 1: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, one vehicle.

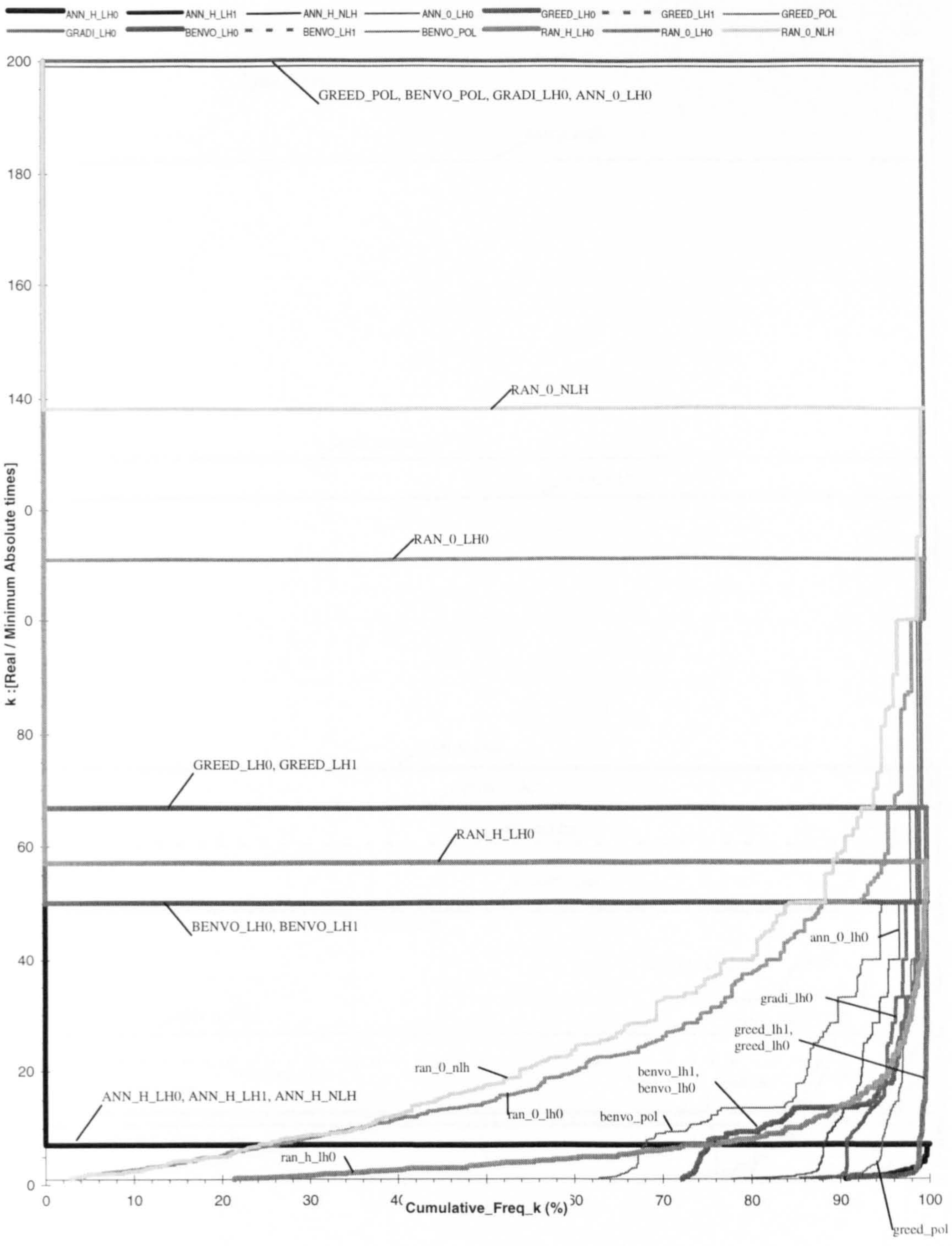


Figure E- 2: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, two vehicles.

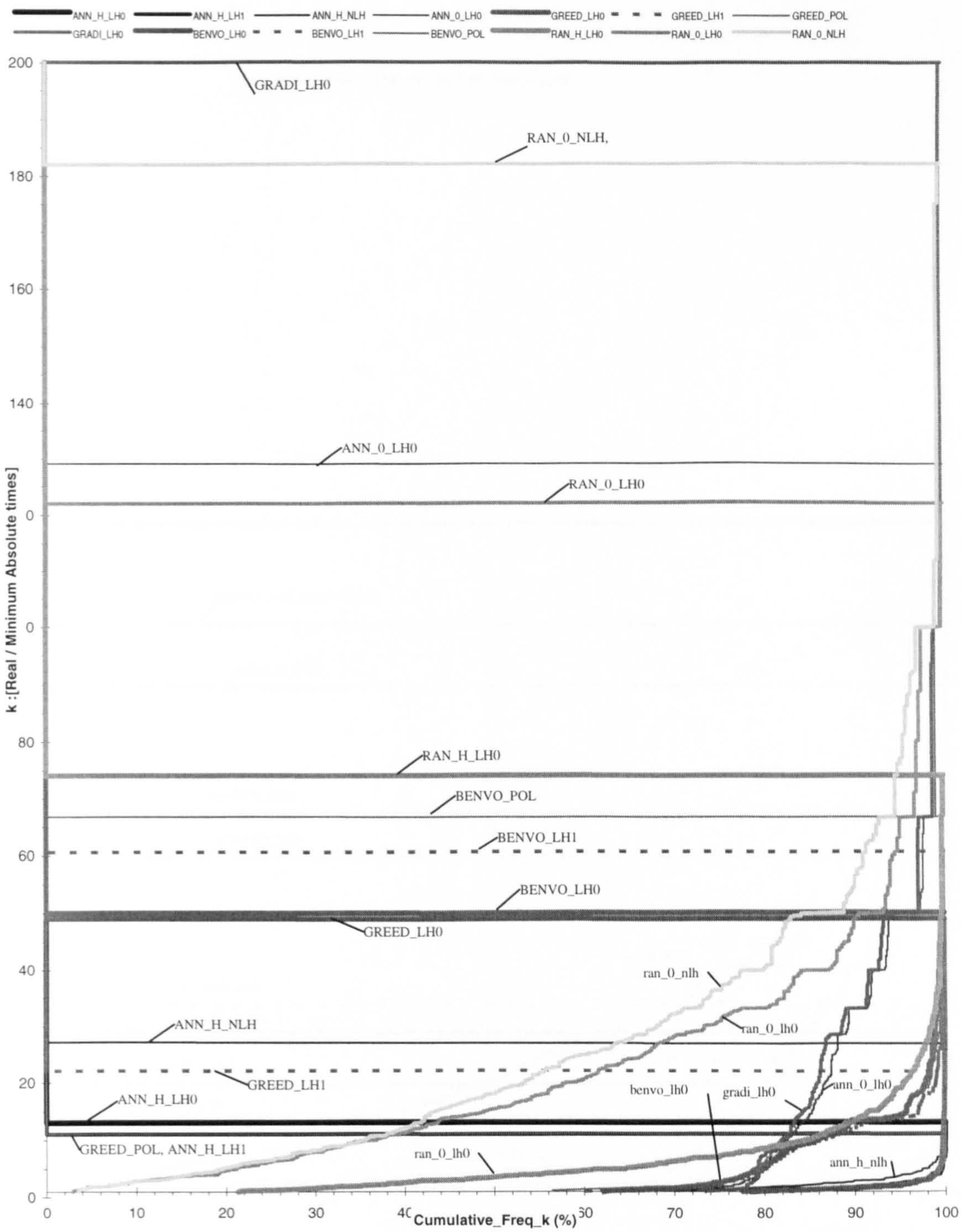


Figure E- 3: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, three vehicles.

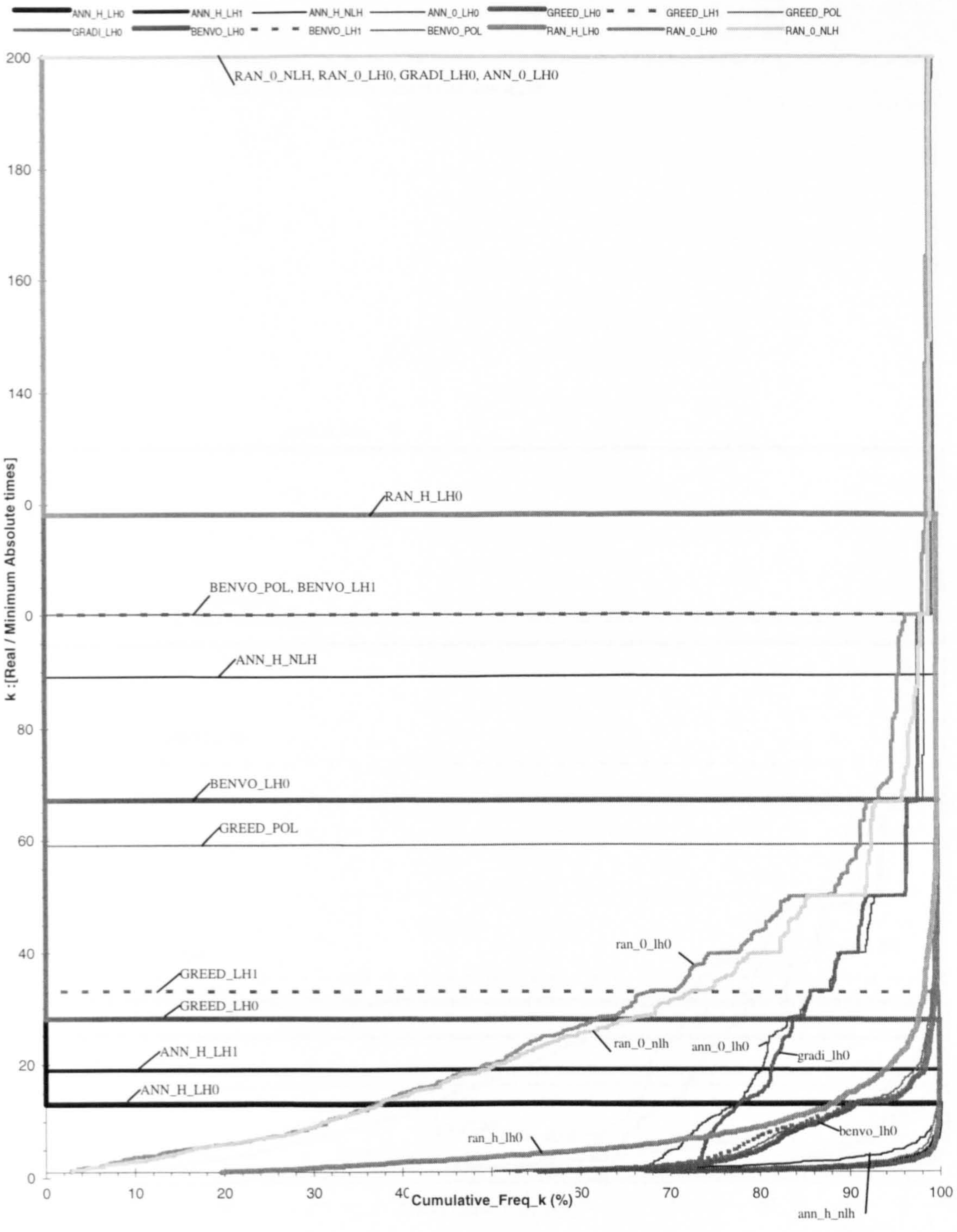


Figure E- 4: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, four vehicles.

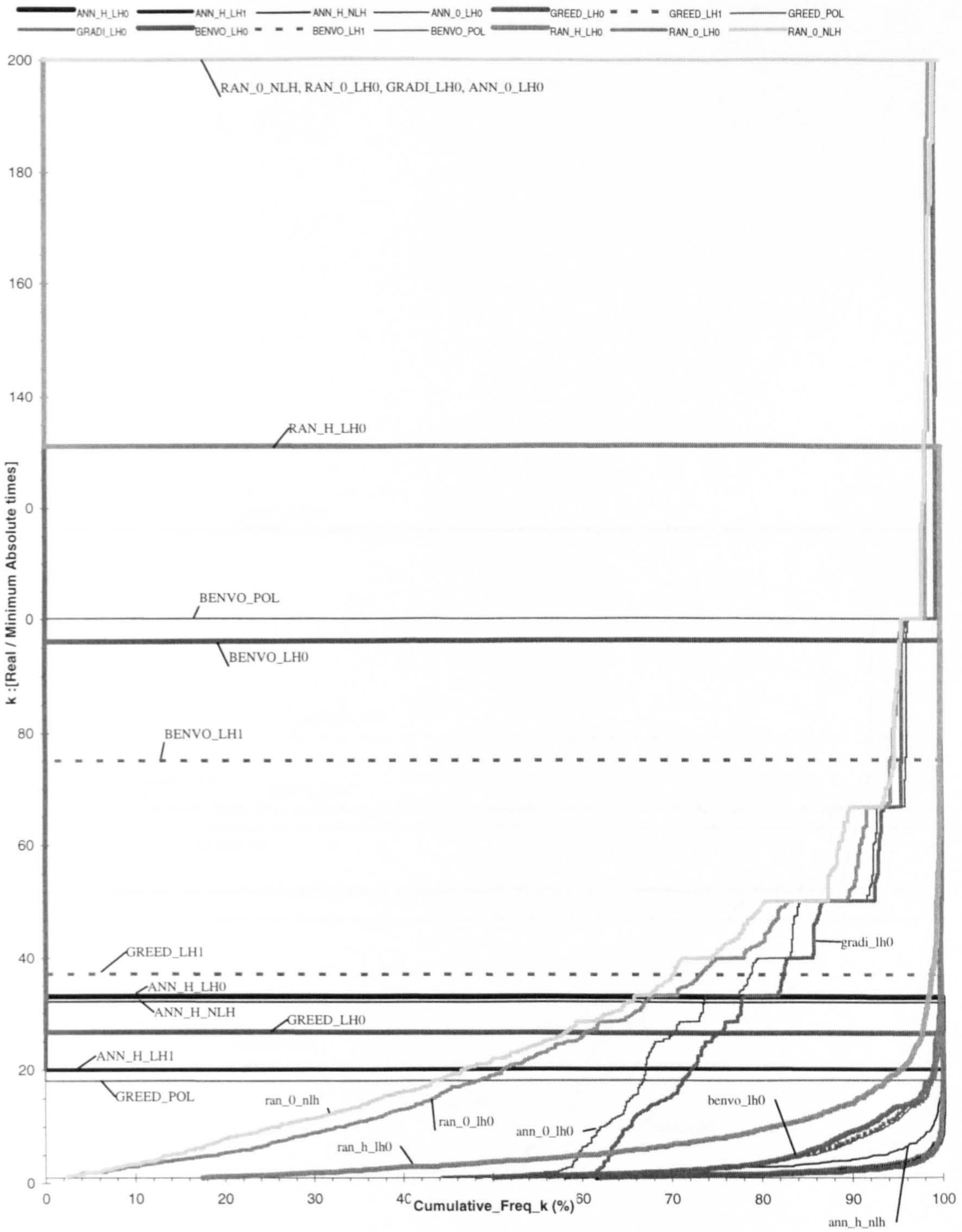


Figure E- 5: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, five vehicles.

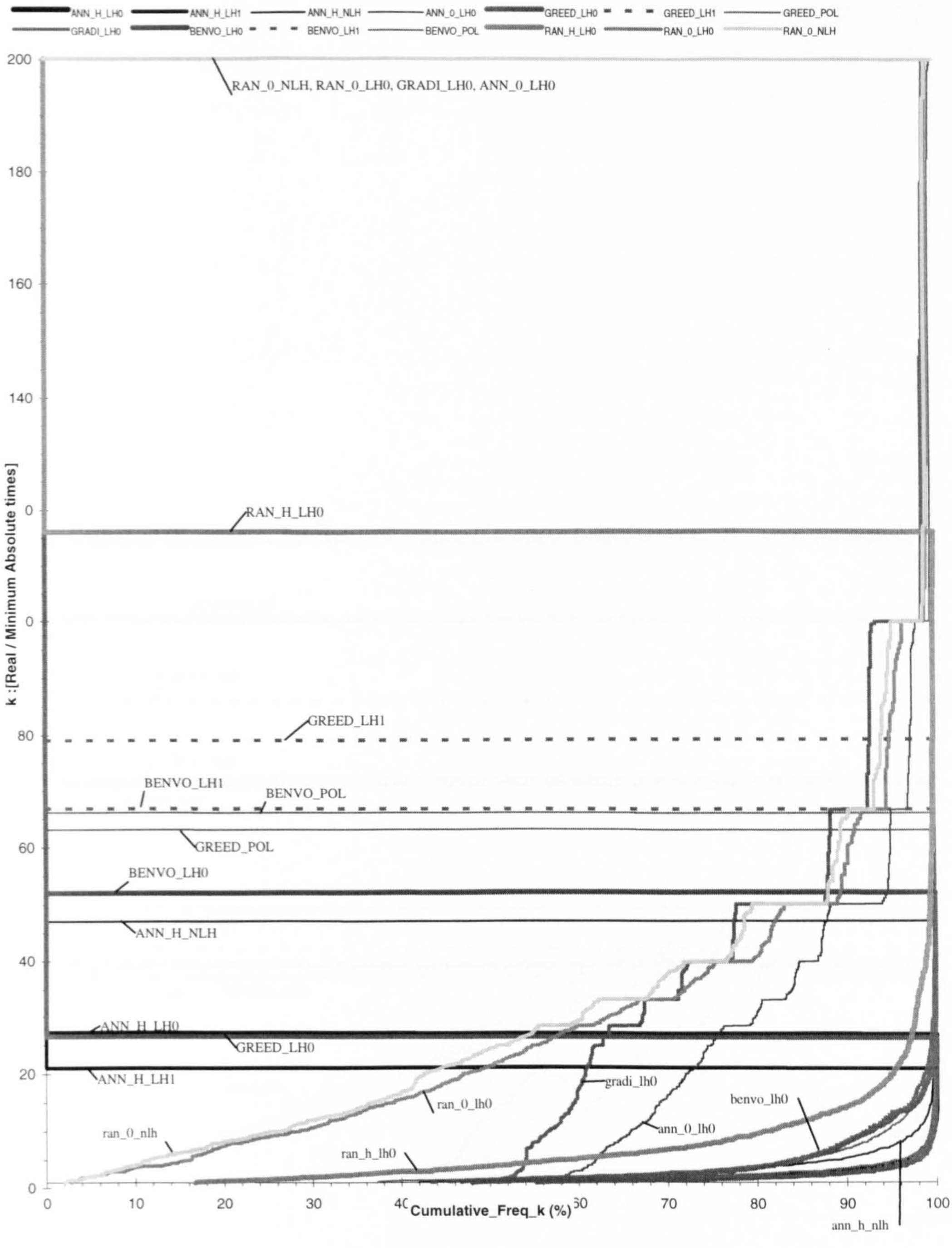


Figure E- 6: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, six vehicles.

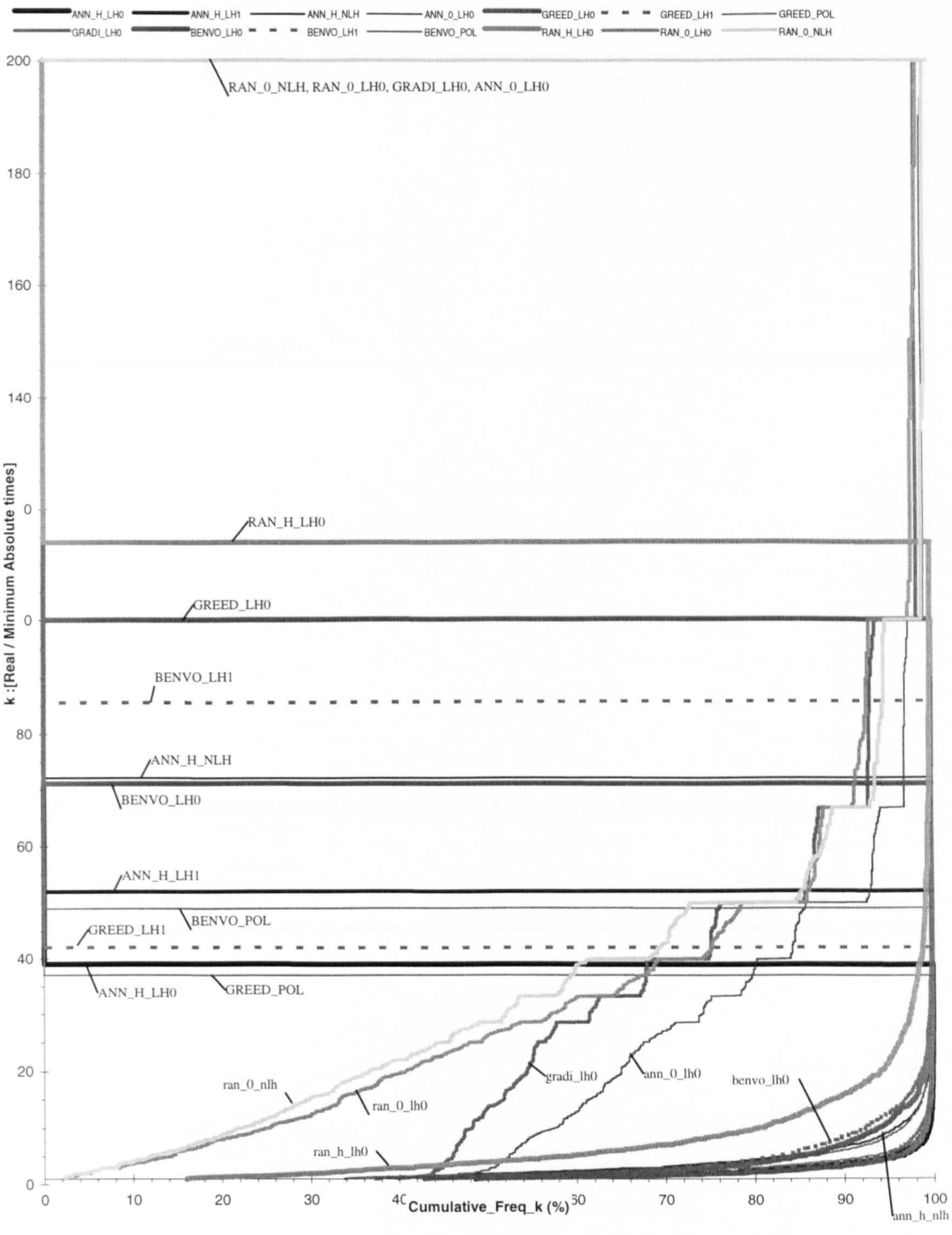


Figure E- 7: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, seven vehicles.

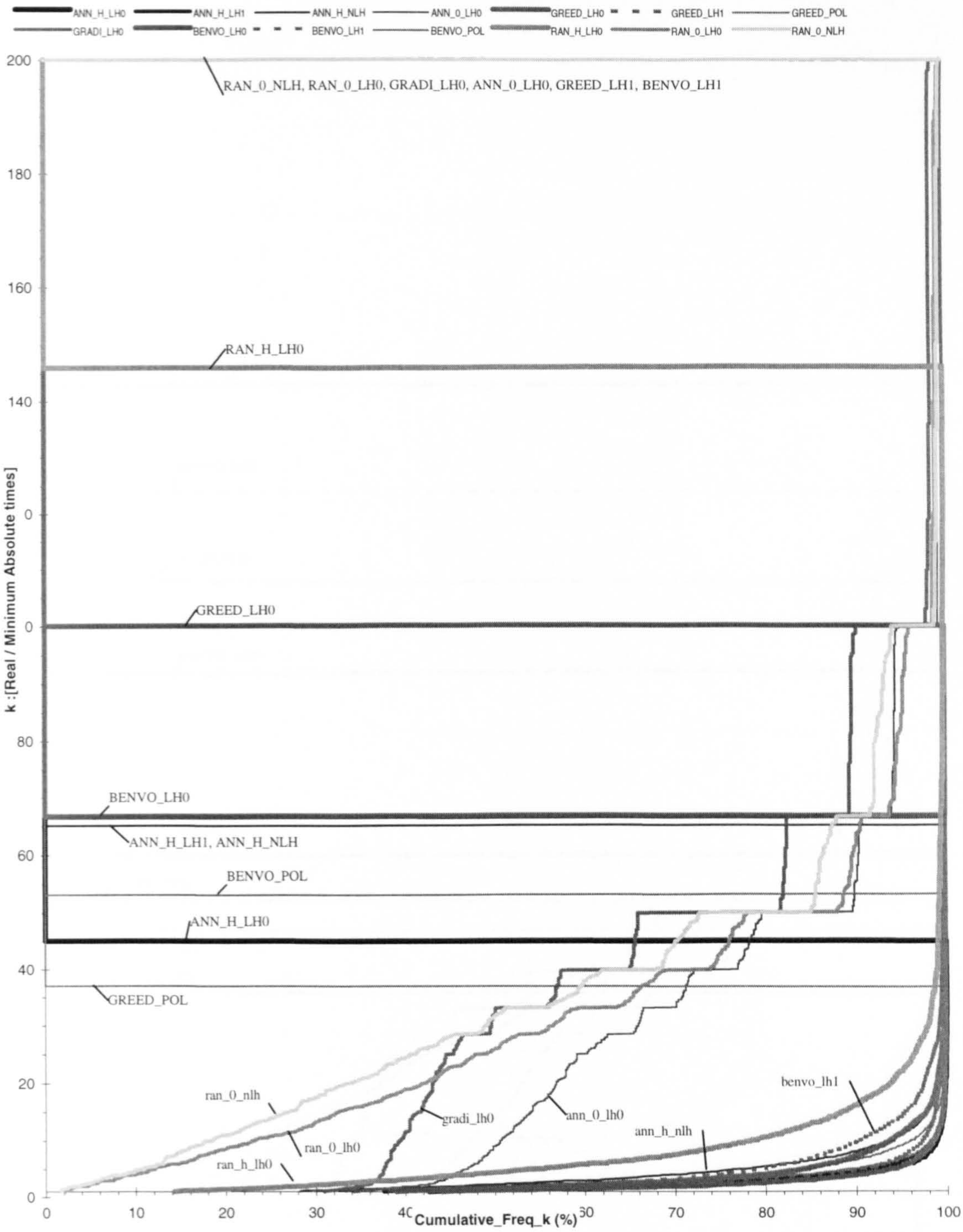


Figure E- 8: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, eight vehicles.

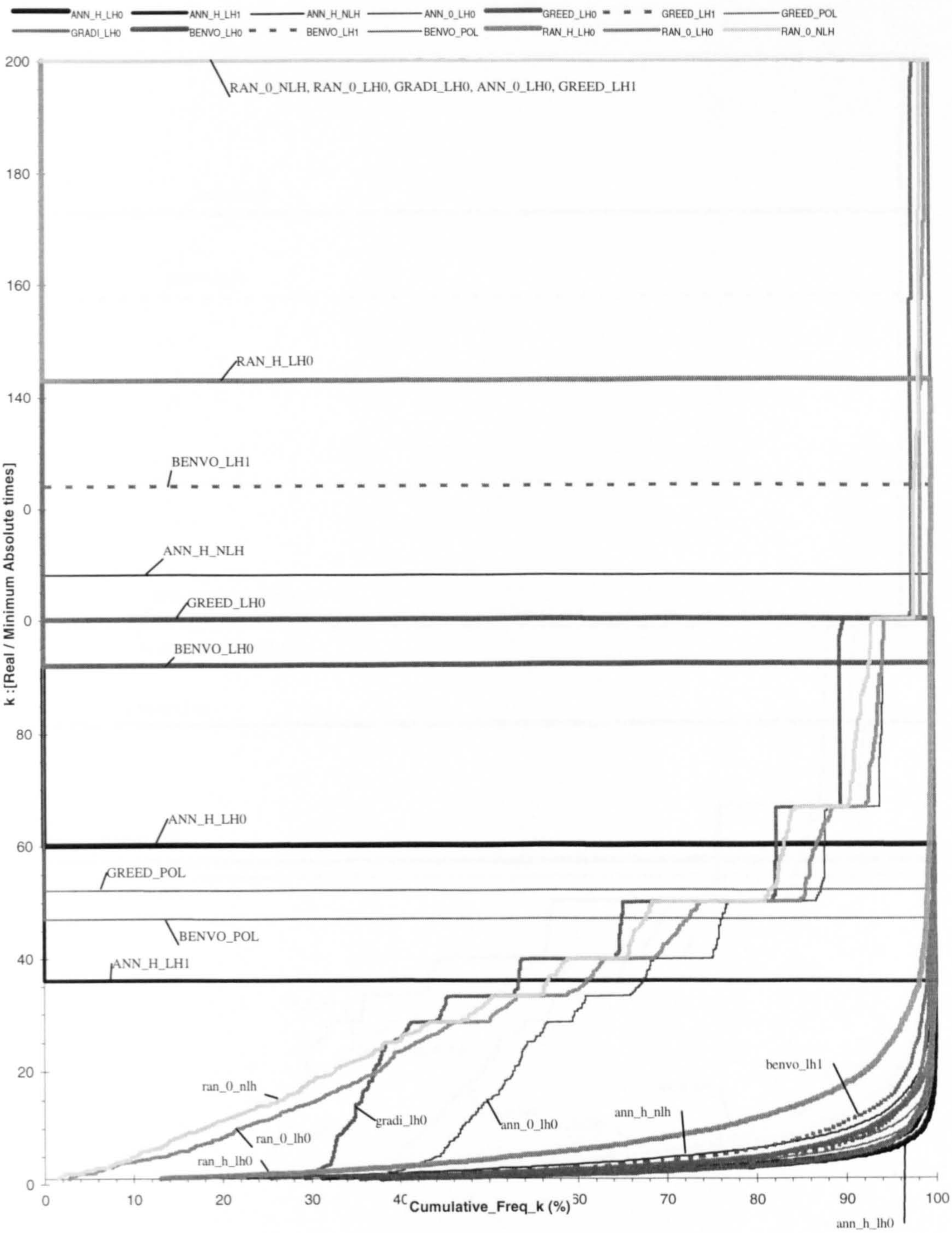


Figure E- 9: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, nine vehicles.

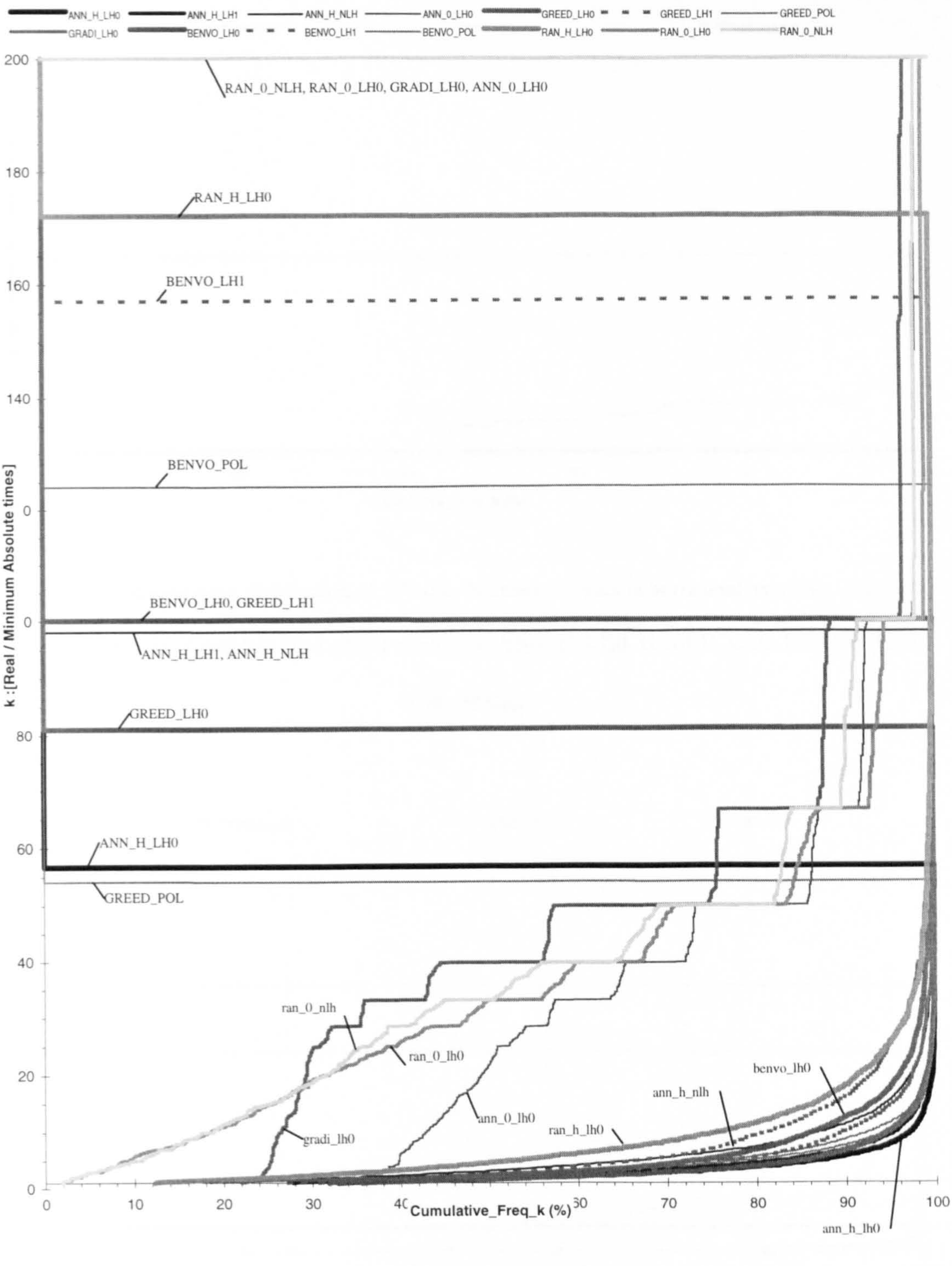


Figure E- 10: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. When *'K'* = 1 the paths were completed in the time required to complete the path along the minimum distance without stopping. All strategies, ten vehicles.

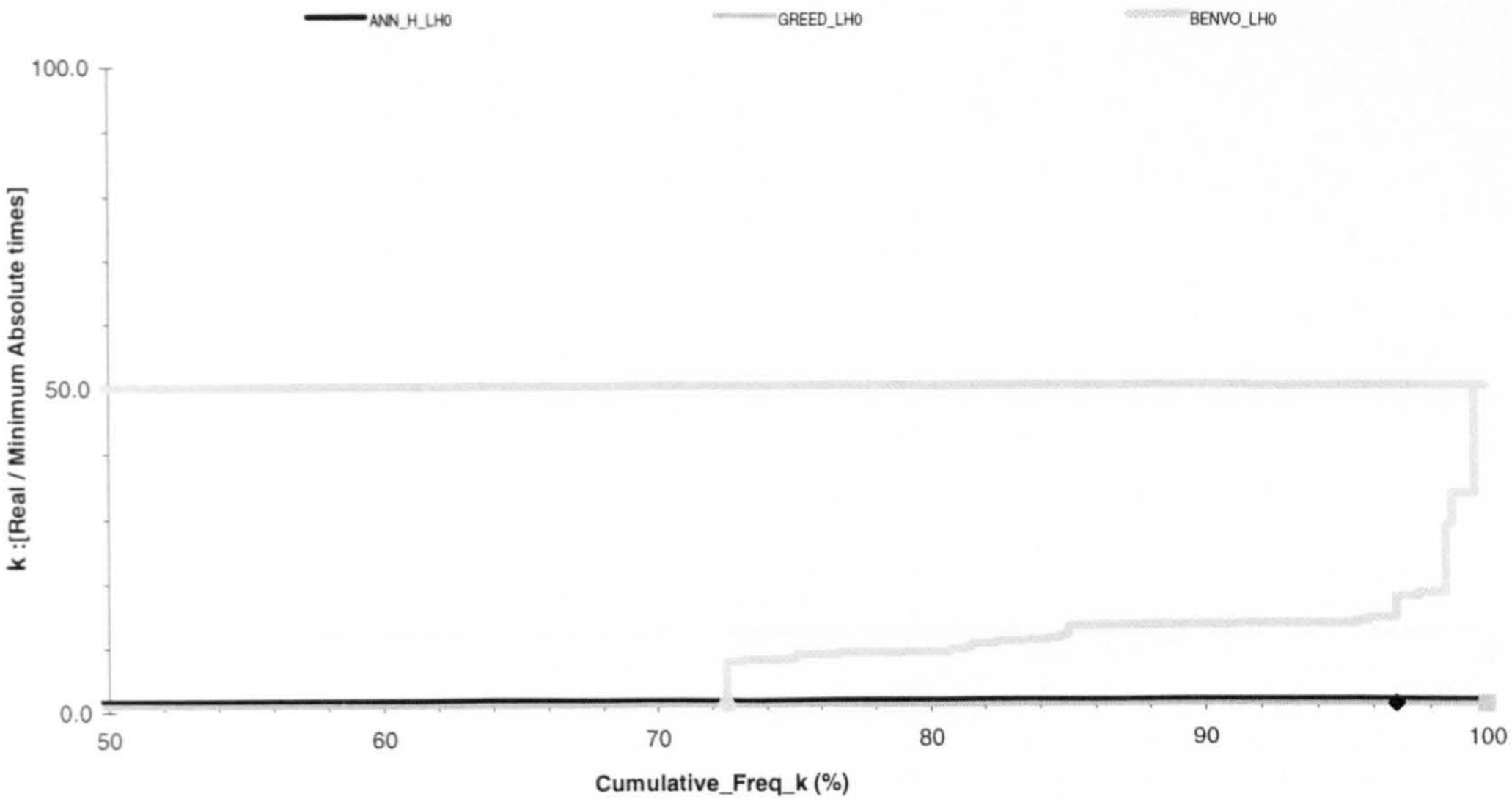


Figure E- 11: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

One vehicle.

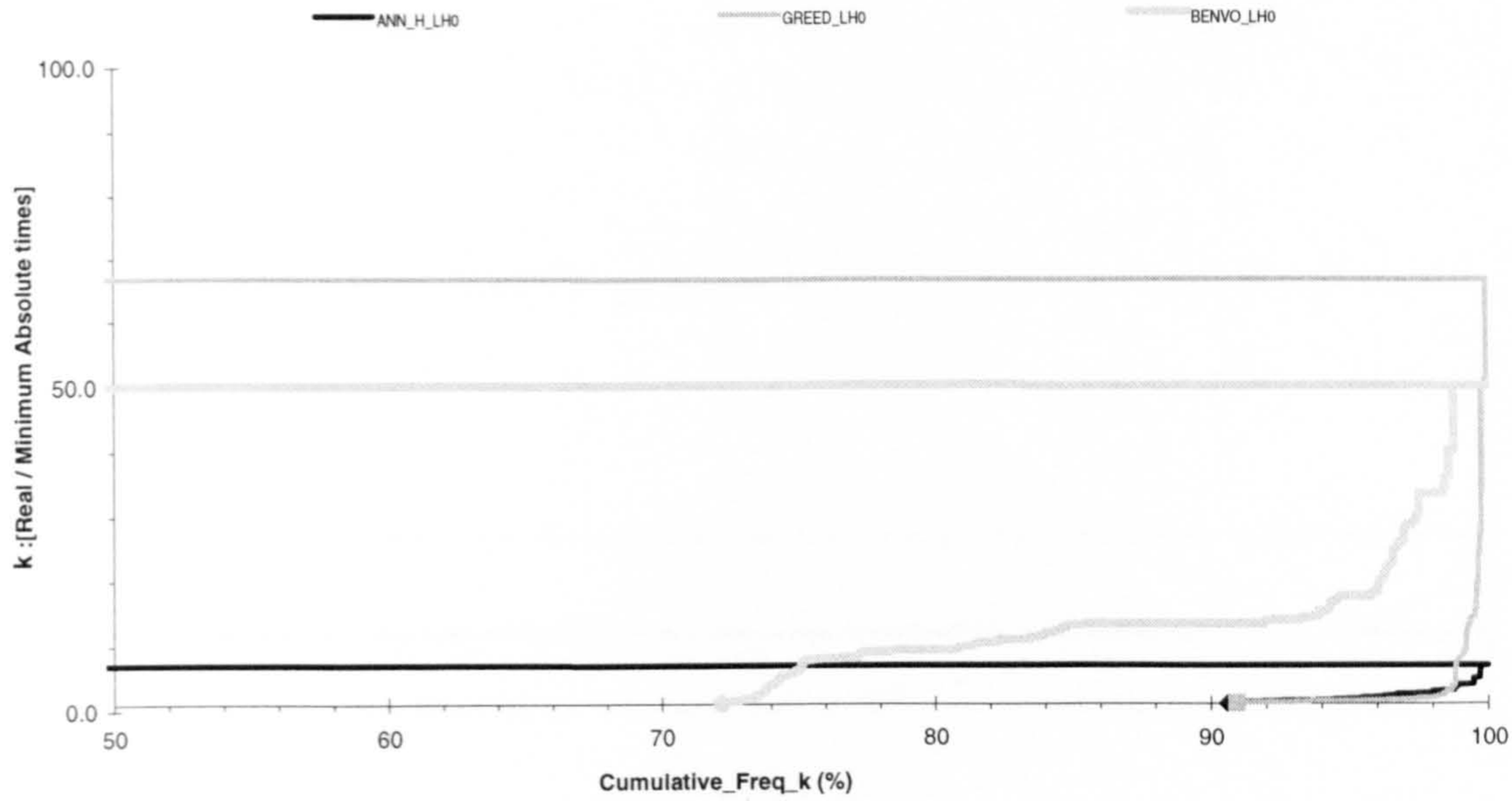


Figure E- 12: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Two vehicles.

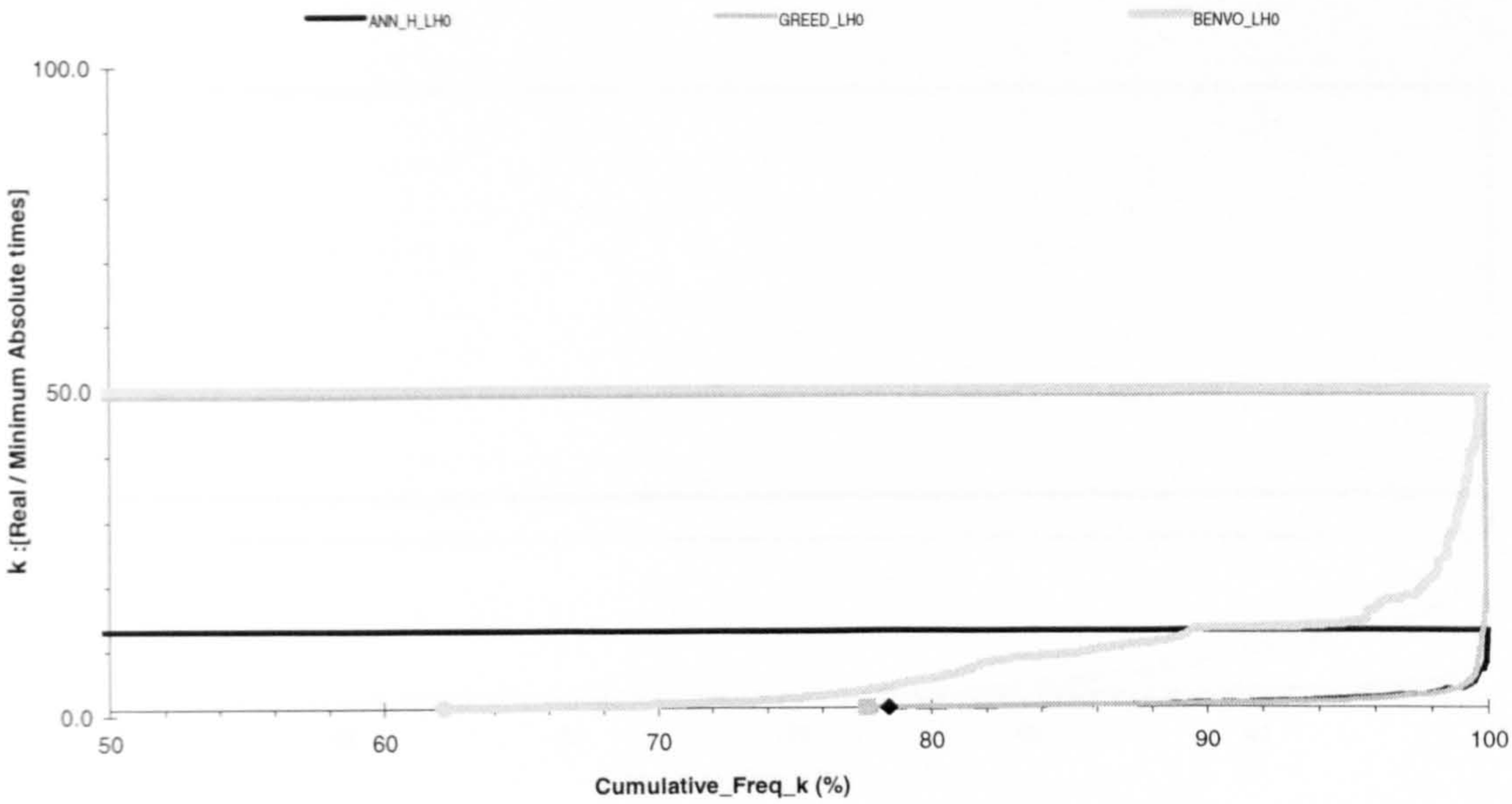


Figure E- 13: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Three vehicles.

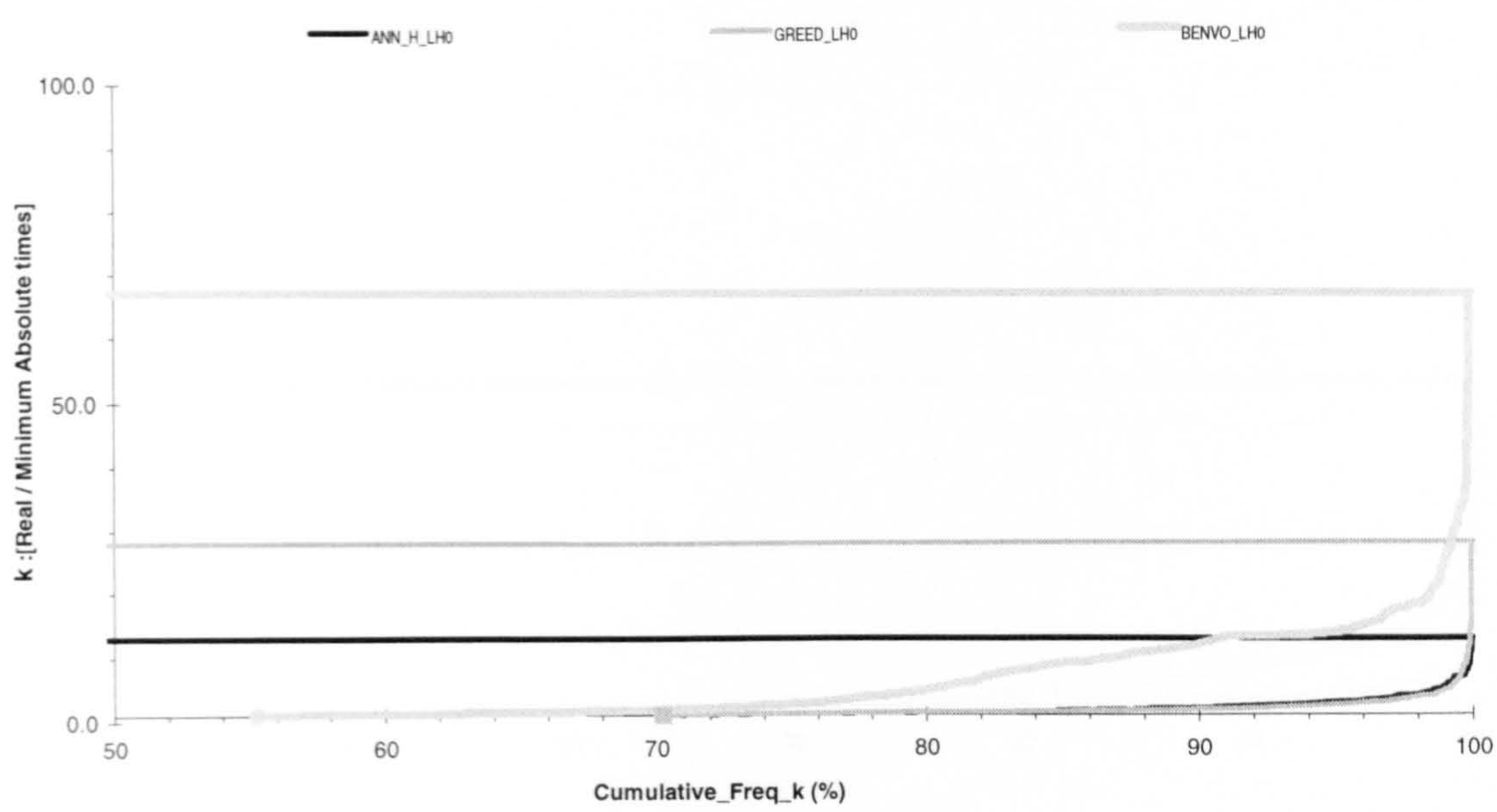


Figure E- 14: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Four vehicles.

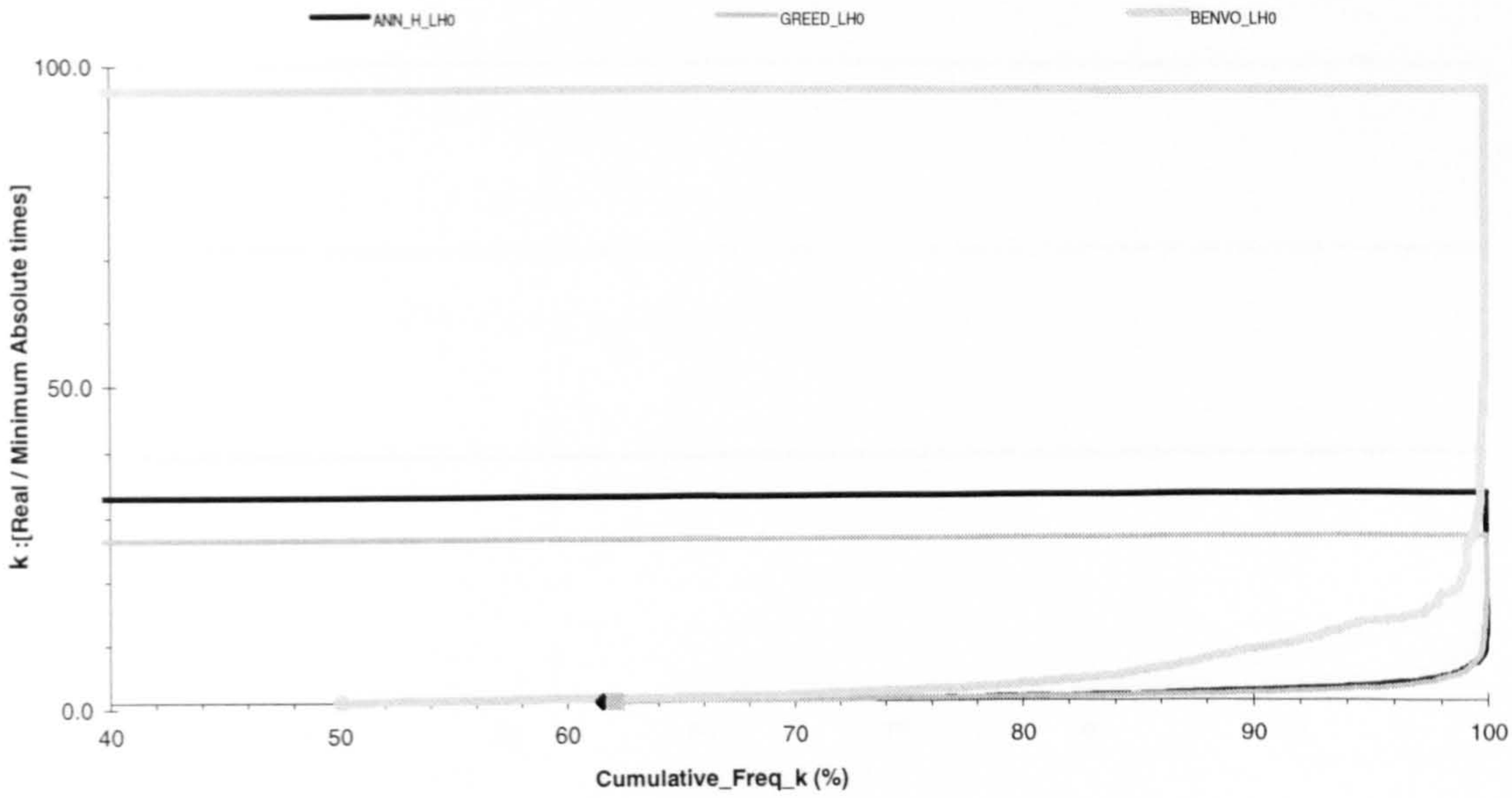


Figure E- 15: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Five vehicles.

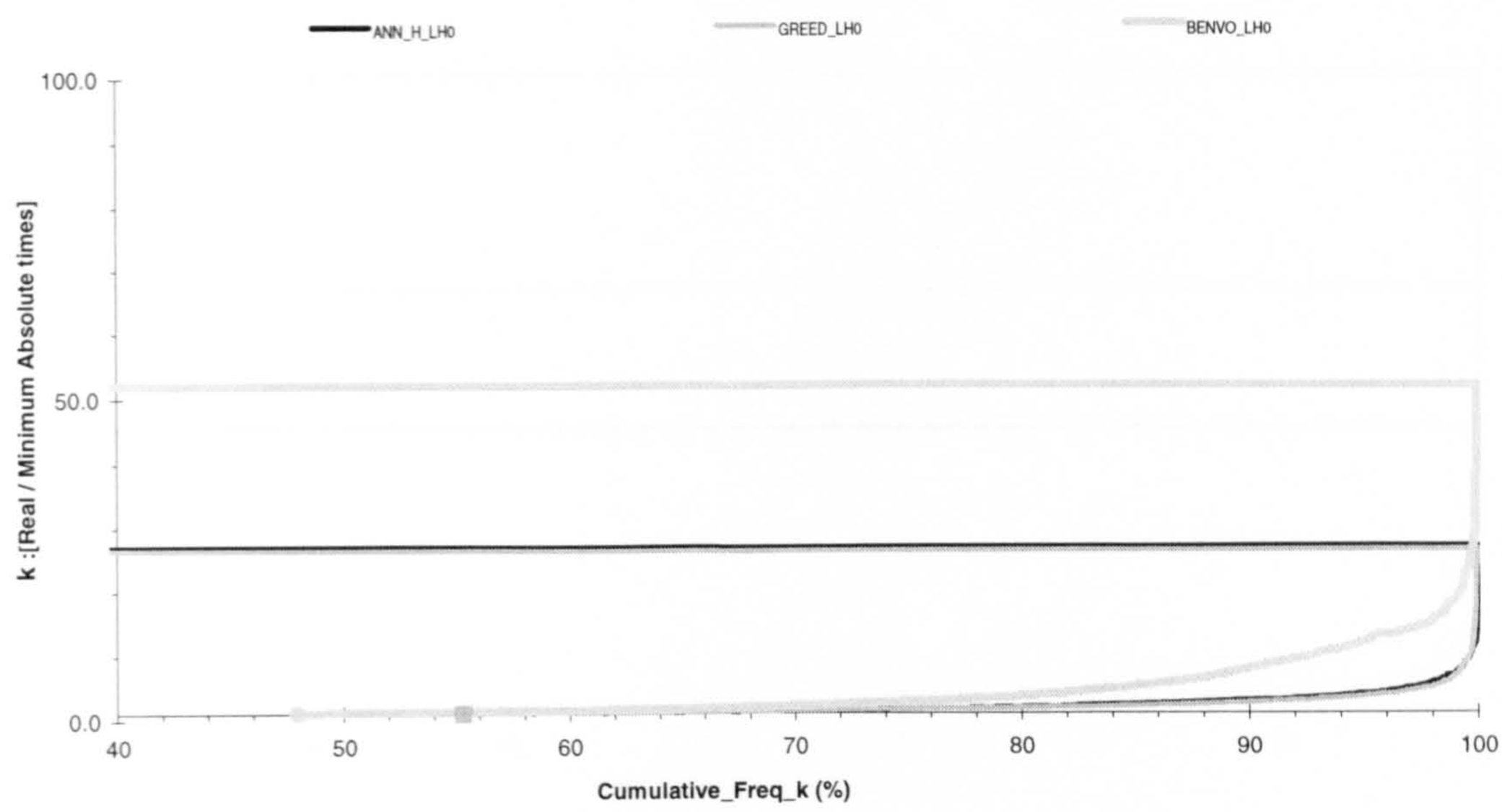


Figure E- 16: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Six vehicles.

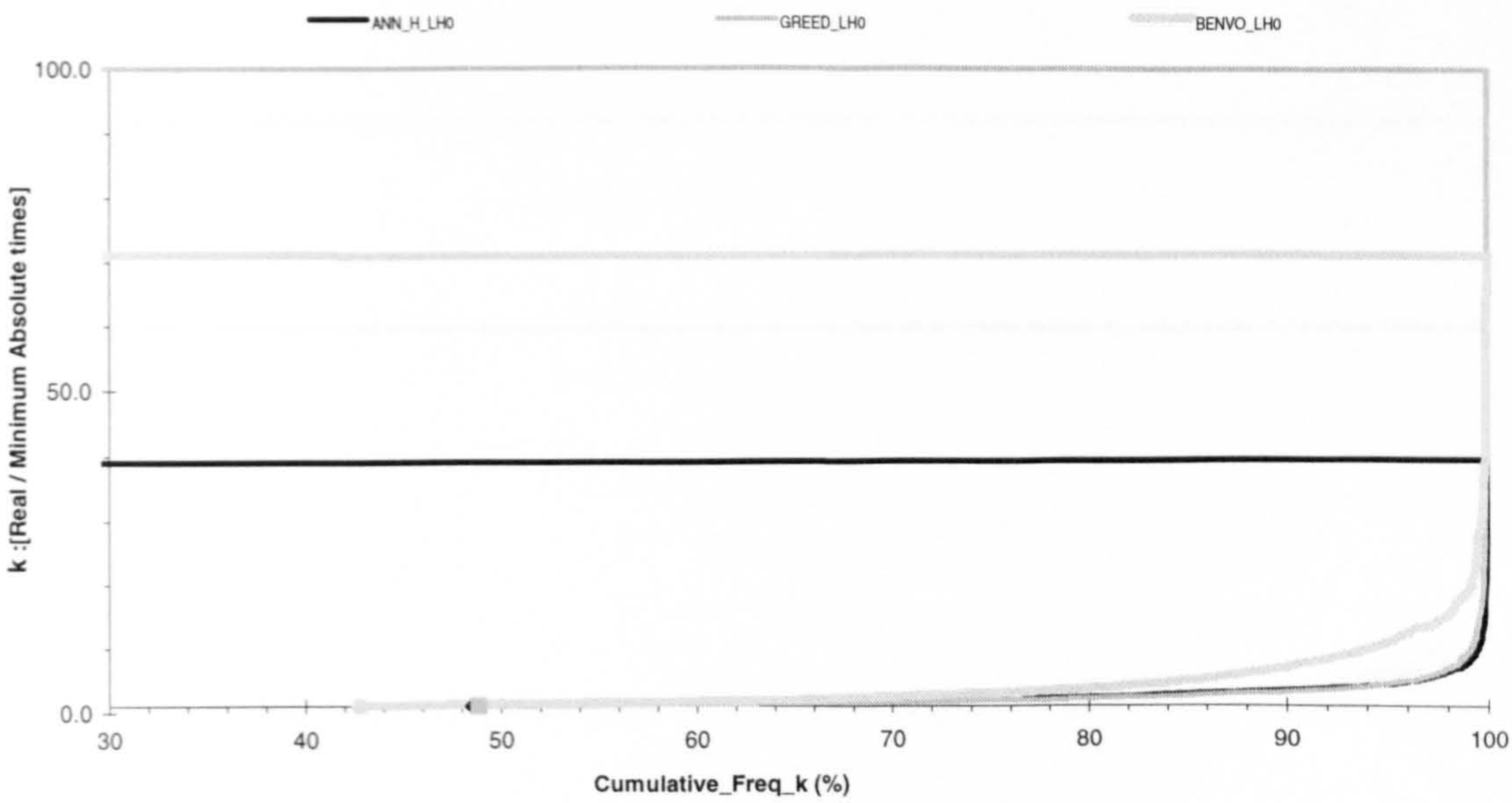


Figure E- 17: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Seven vehicles.

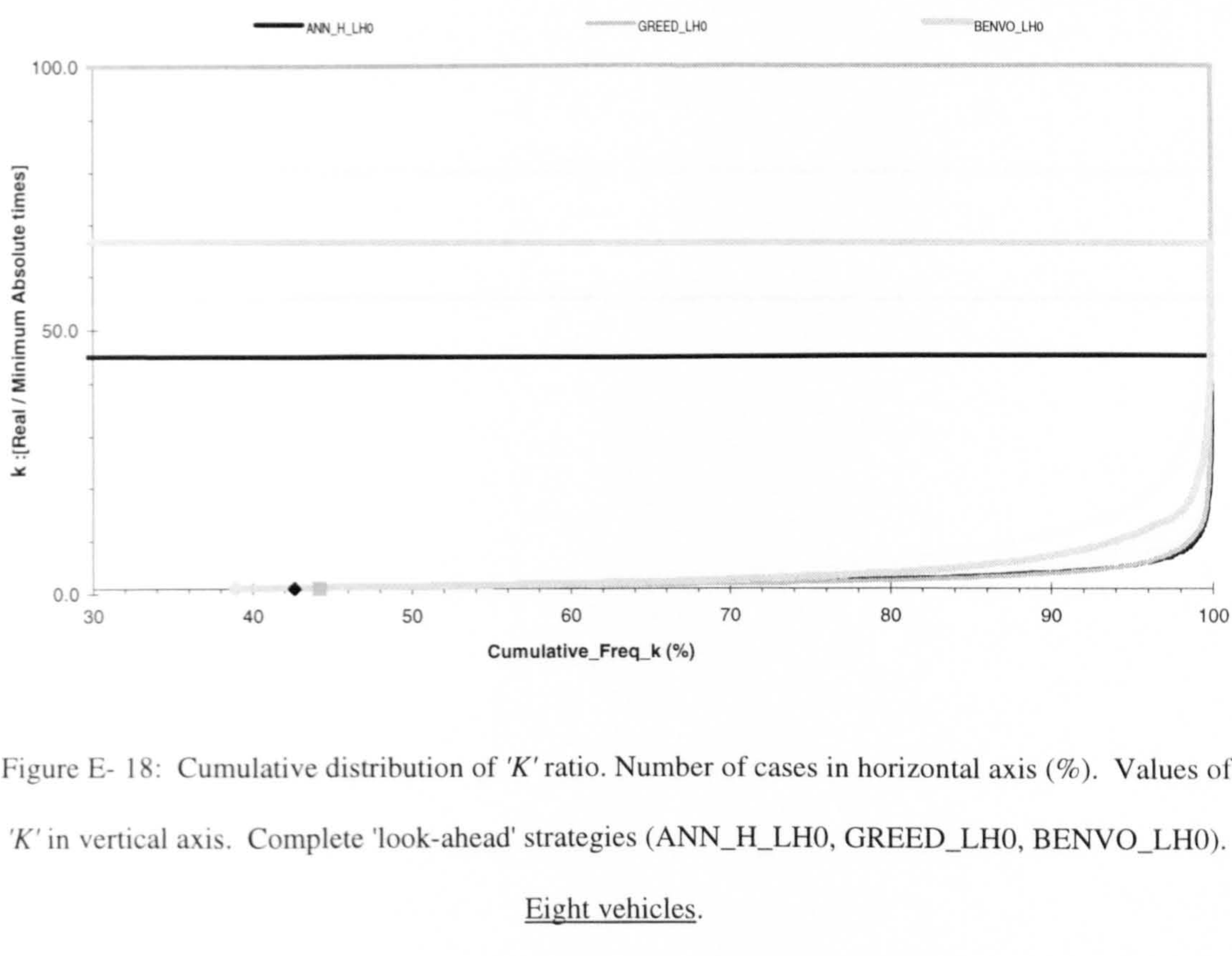


Figure E- 18: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Eight vehicles.

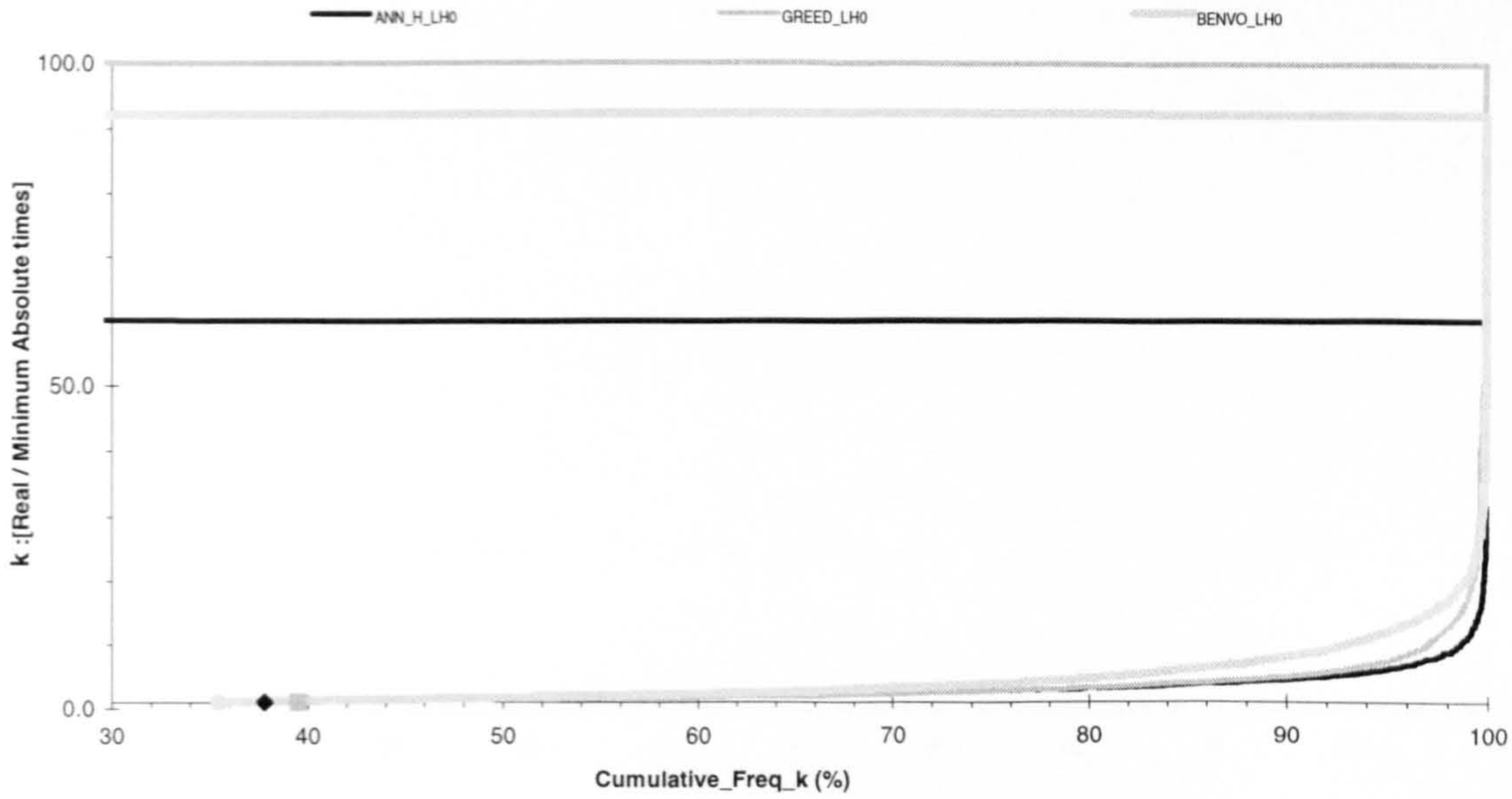


Figure E- 19: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Nine vehicles.

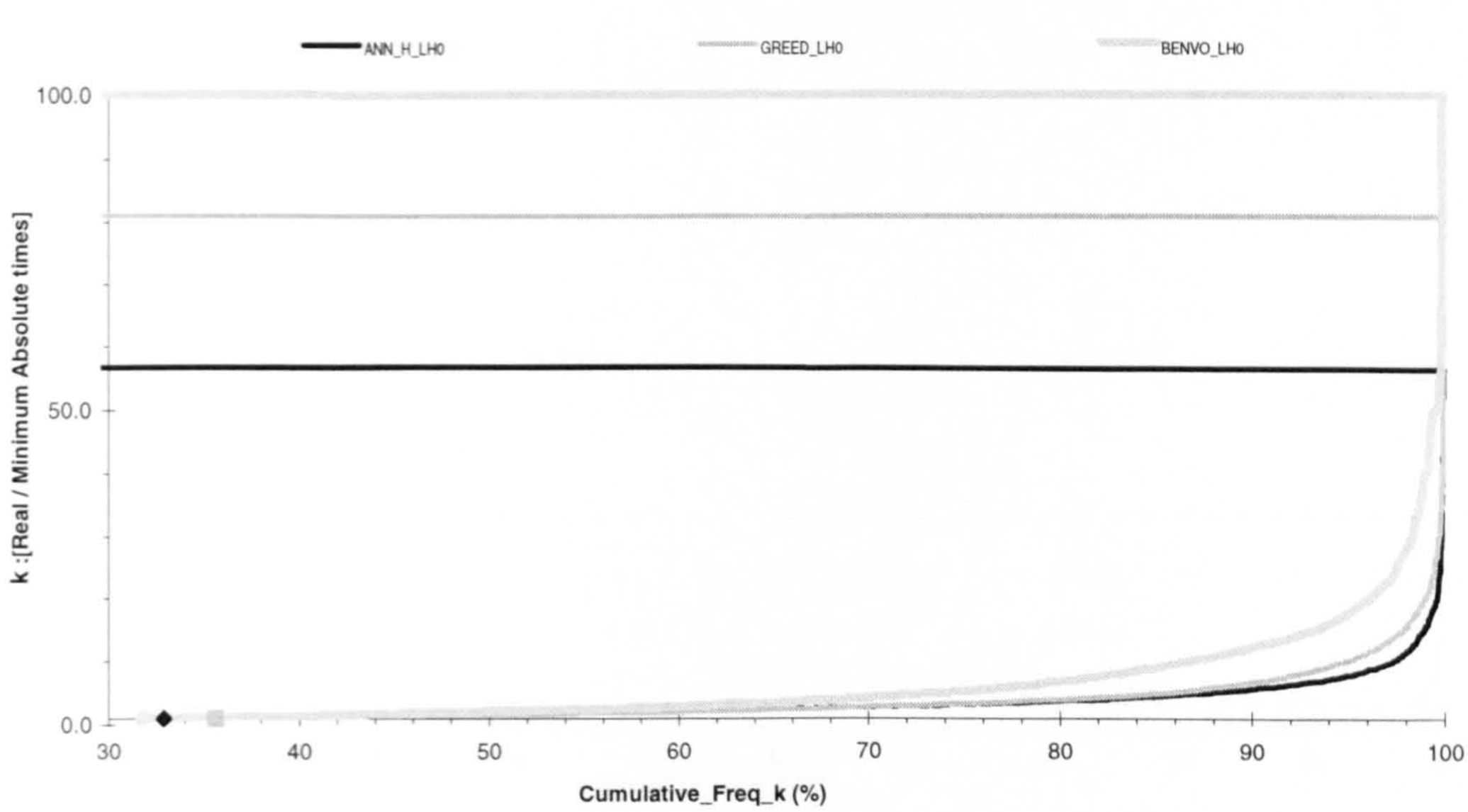


Figure E- 20: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Complete 'look-ahead' strategies (ANN_H_LH0, GREED_LH0, BENVO_LH0).

Ten vehicles.

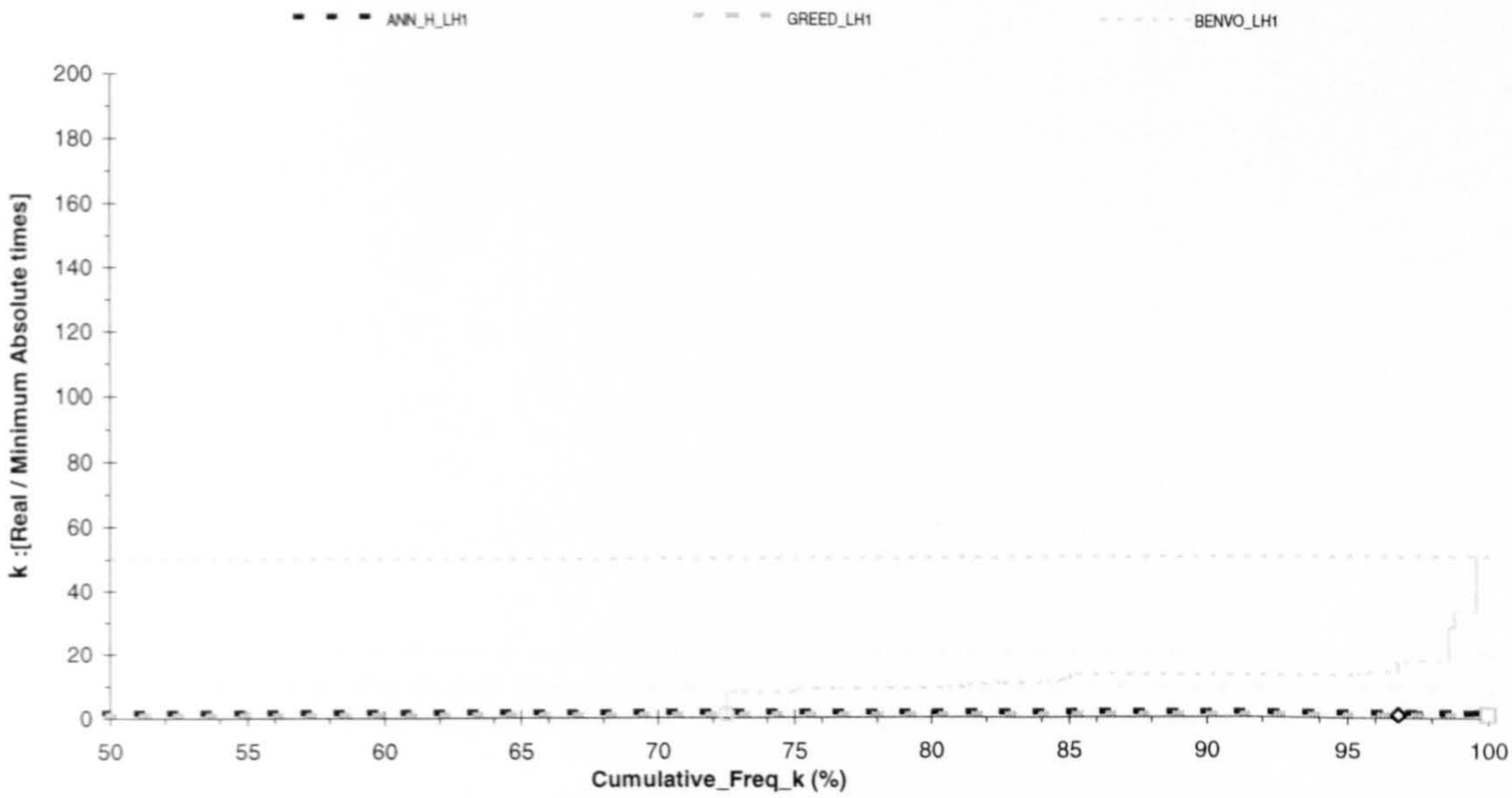


Figure E- 21: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

<u>One vehicle</u>.

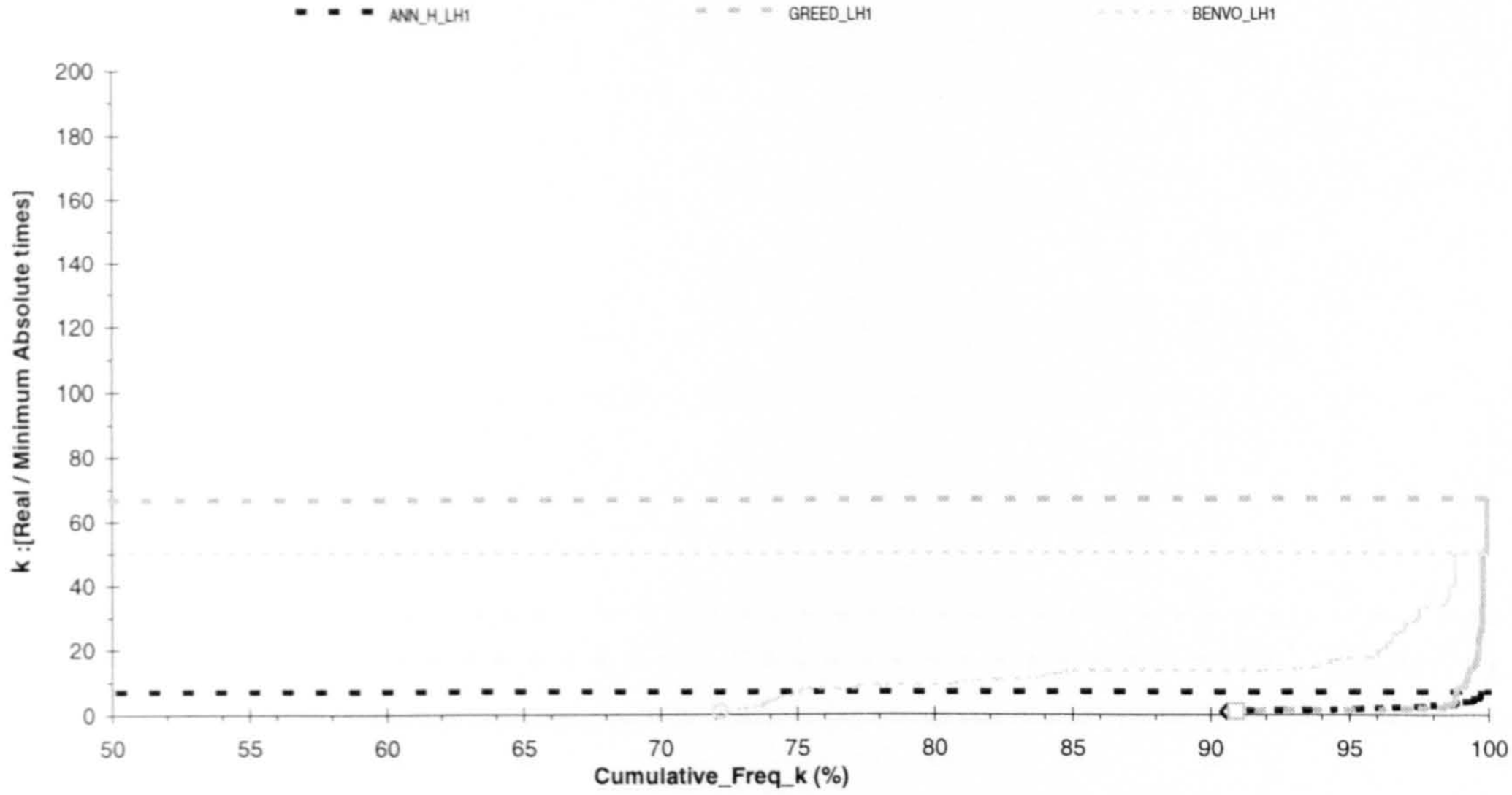


Figure E- 22: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

<u>Two vehicles</u>.

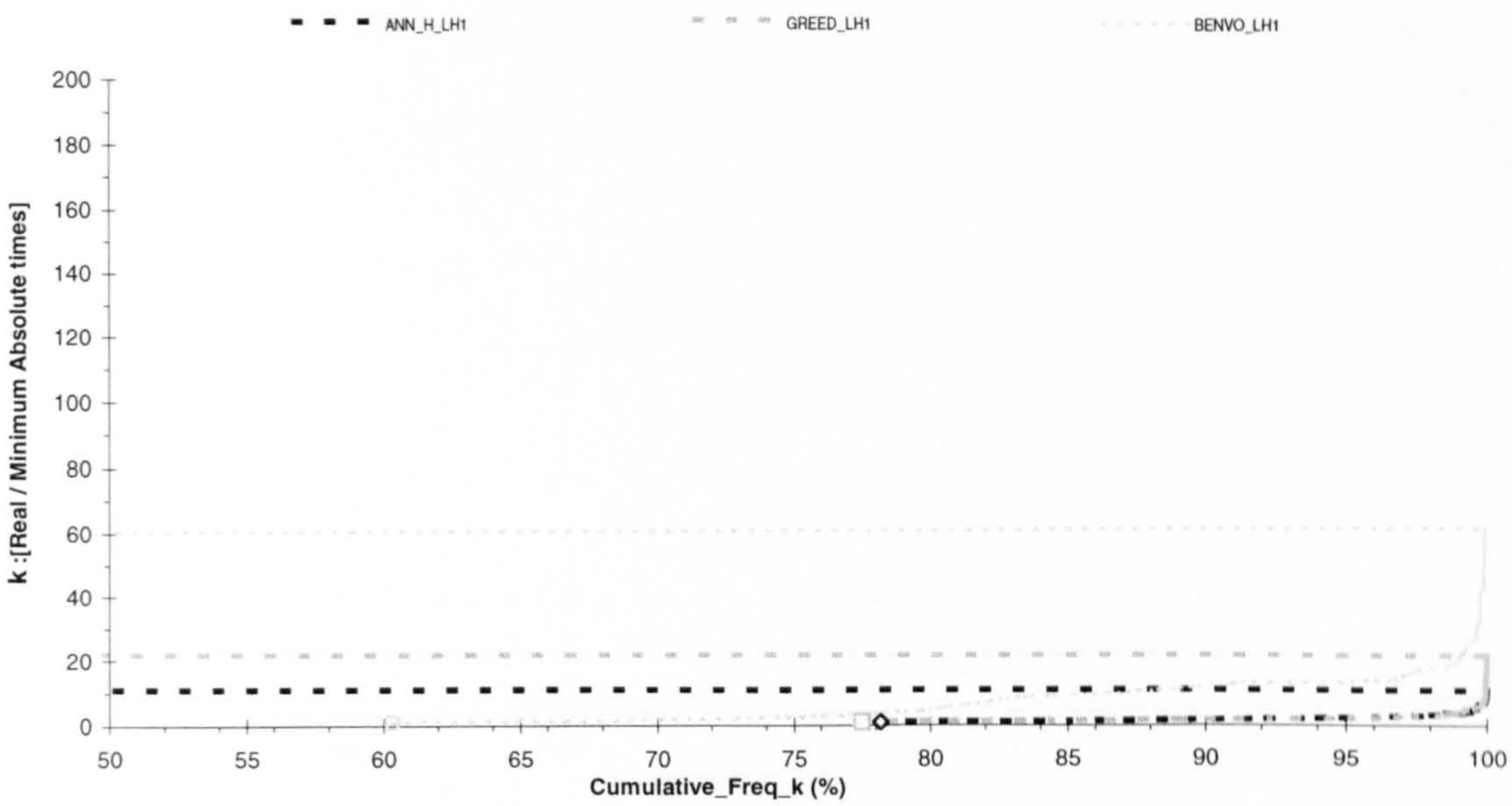


Figure E- 23: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Three vehicles.

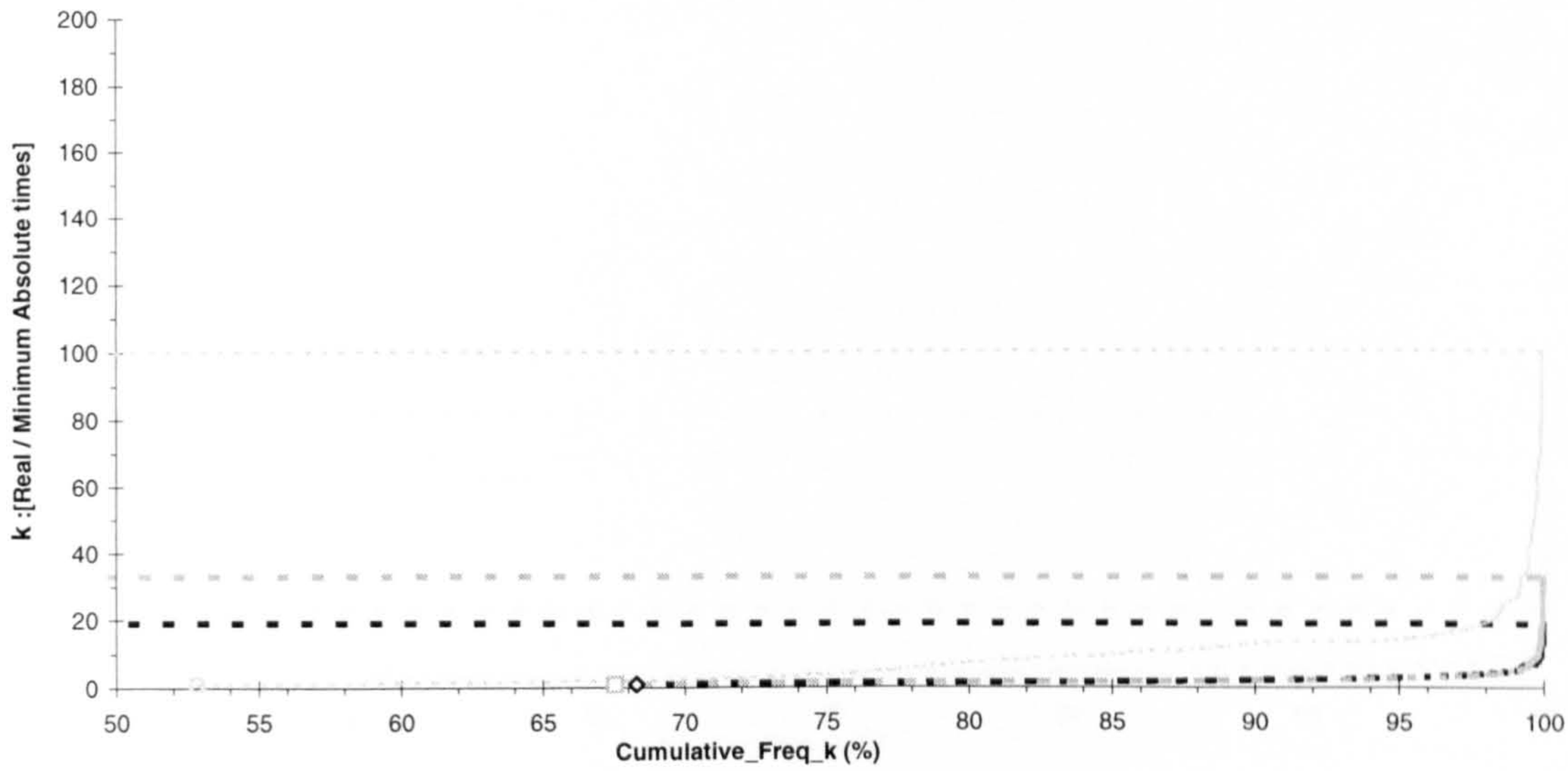


Figure E- 24: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Four vehicles.

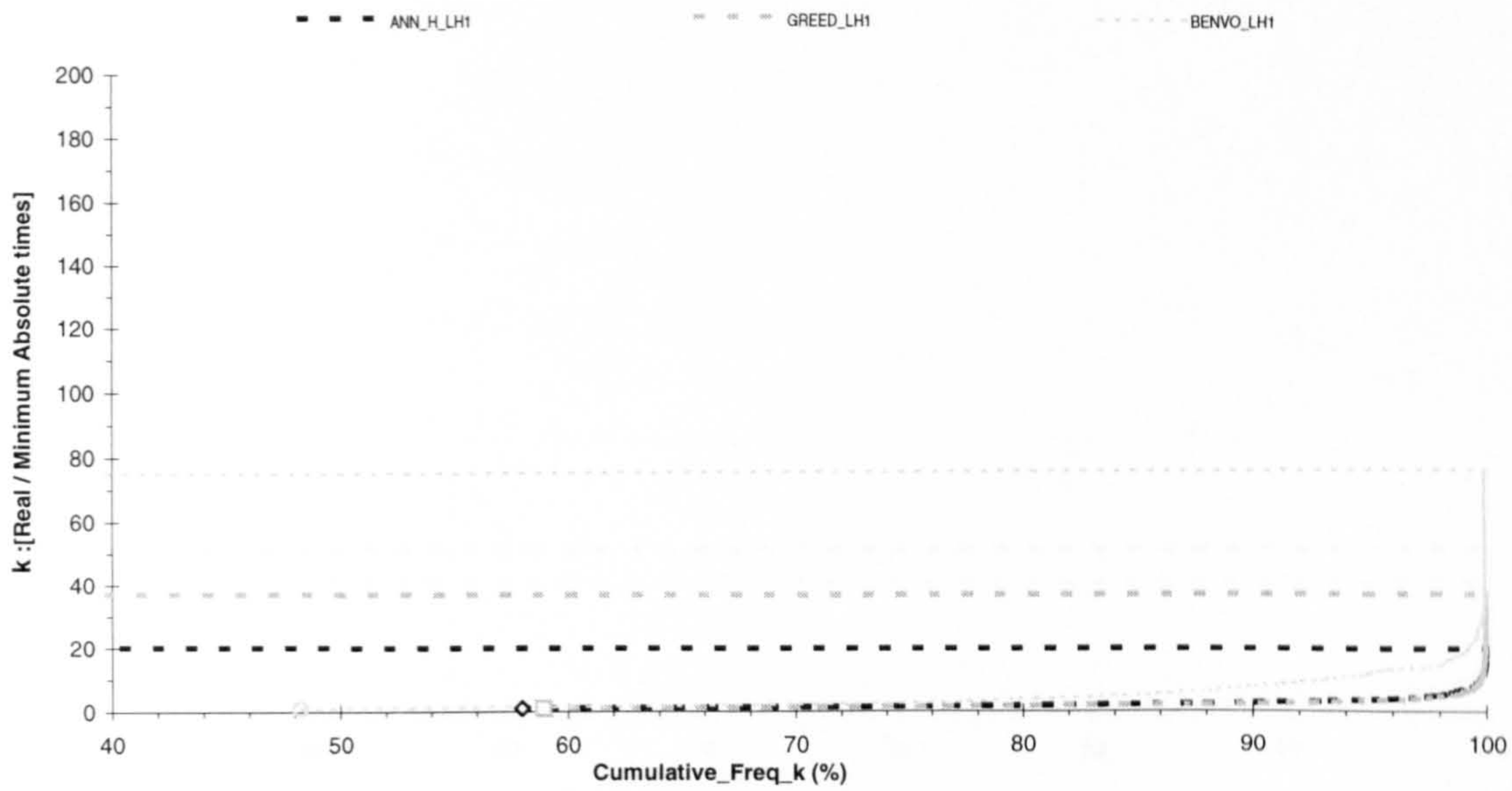


Figure E- 25: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Five vehicles.

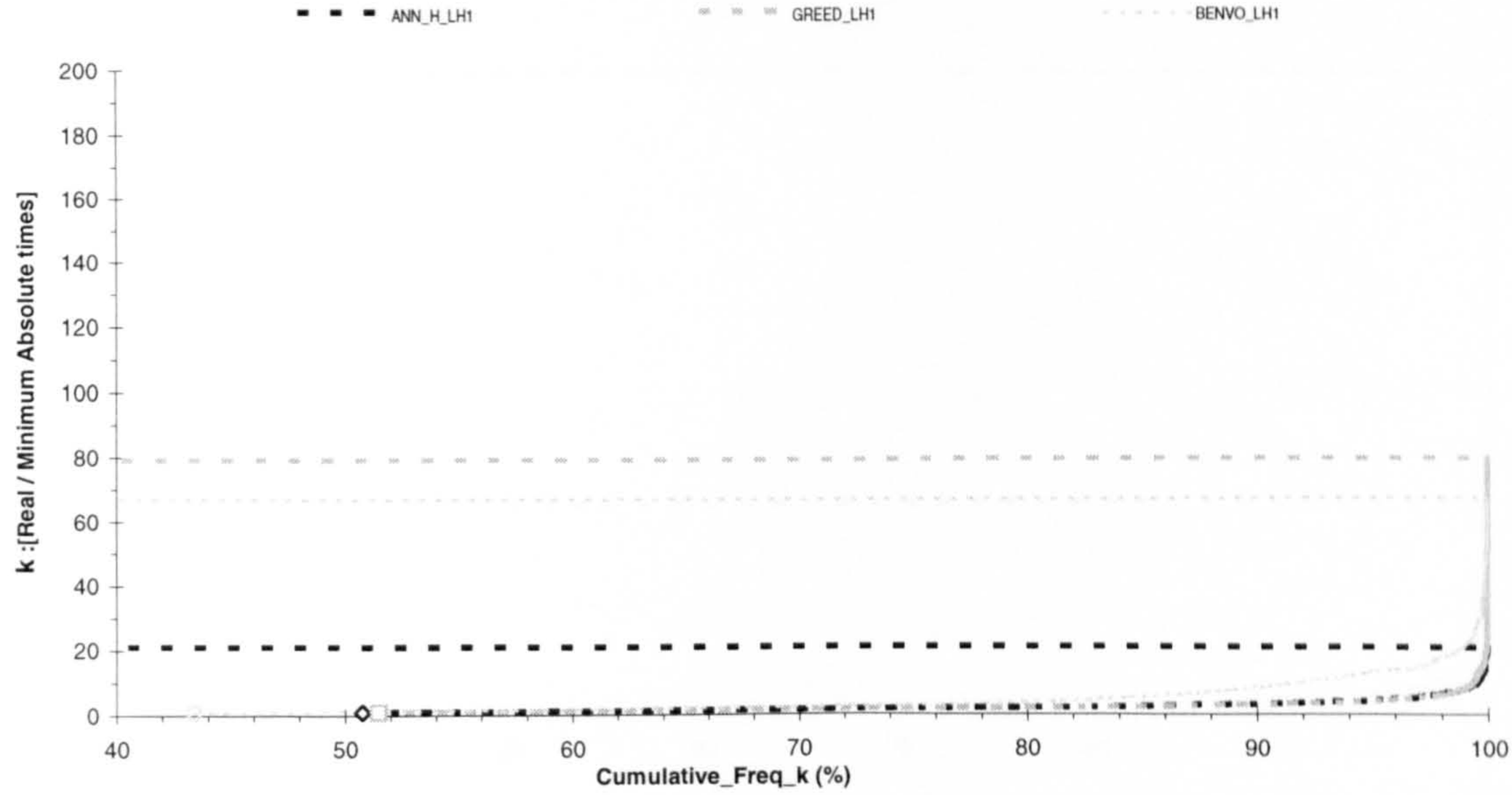


Figure E- 26: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Six vehicles.

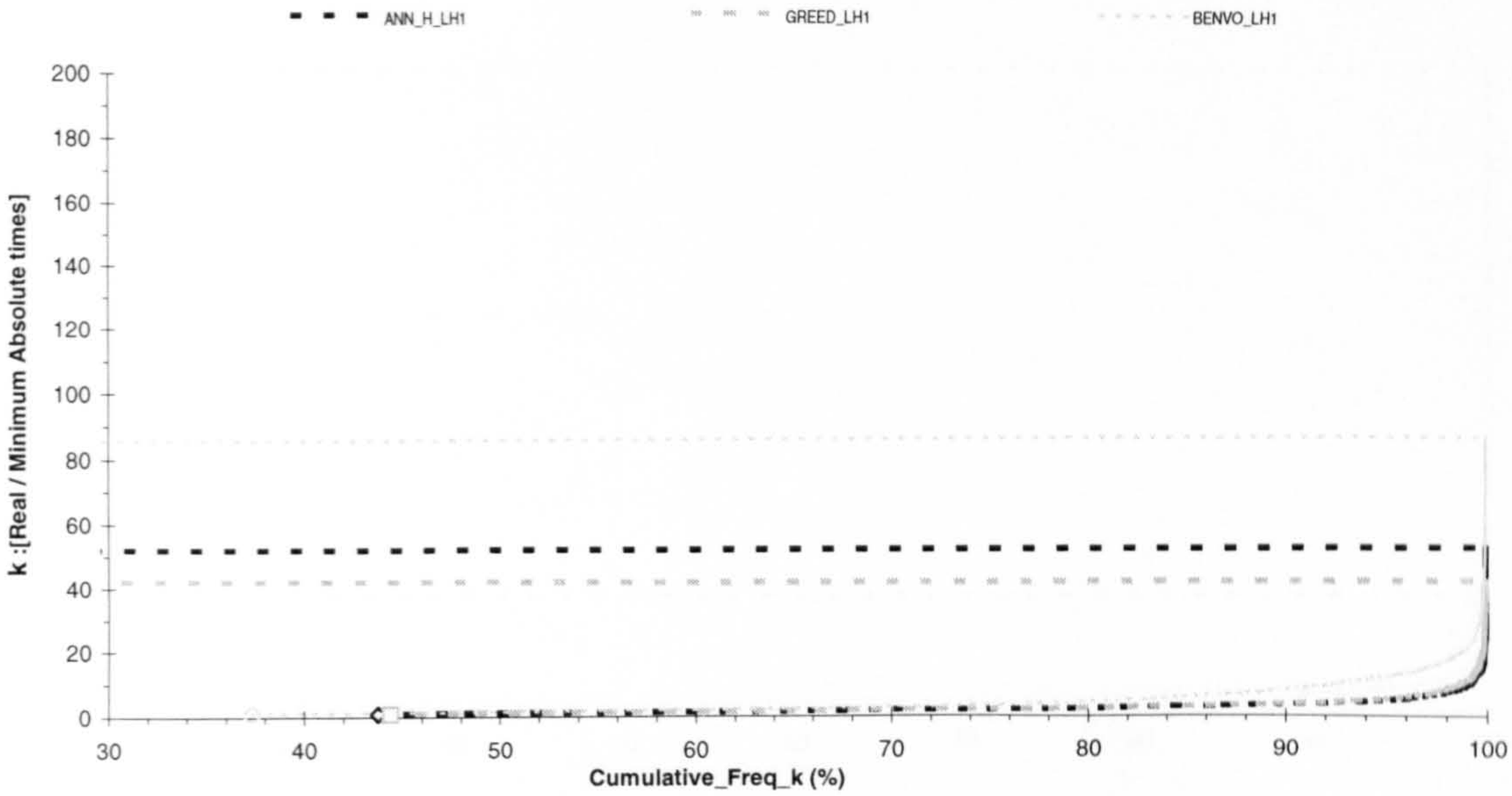


Figure E- 27: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Seven vehicles.

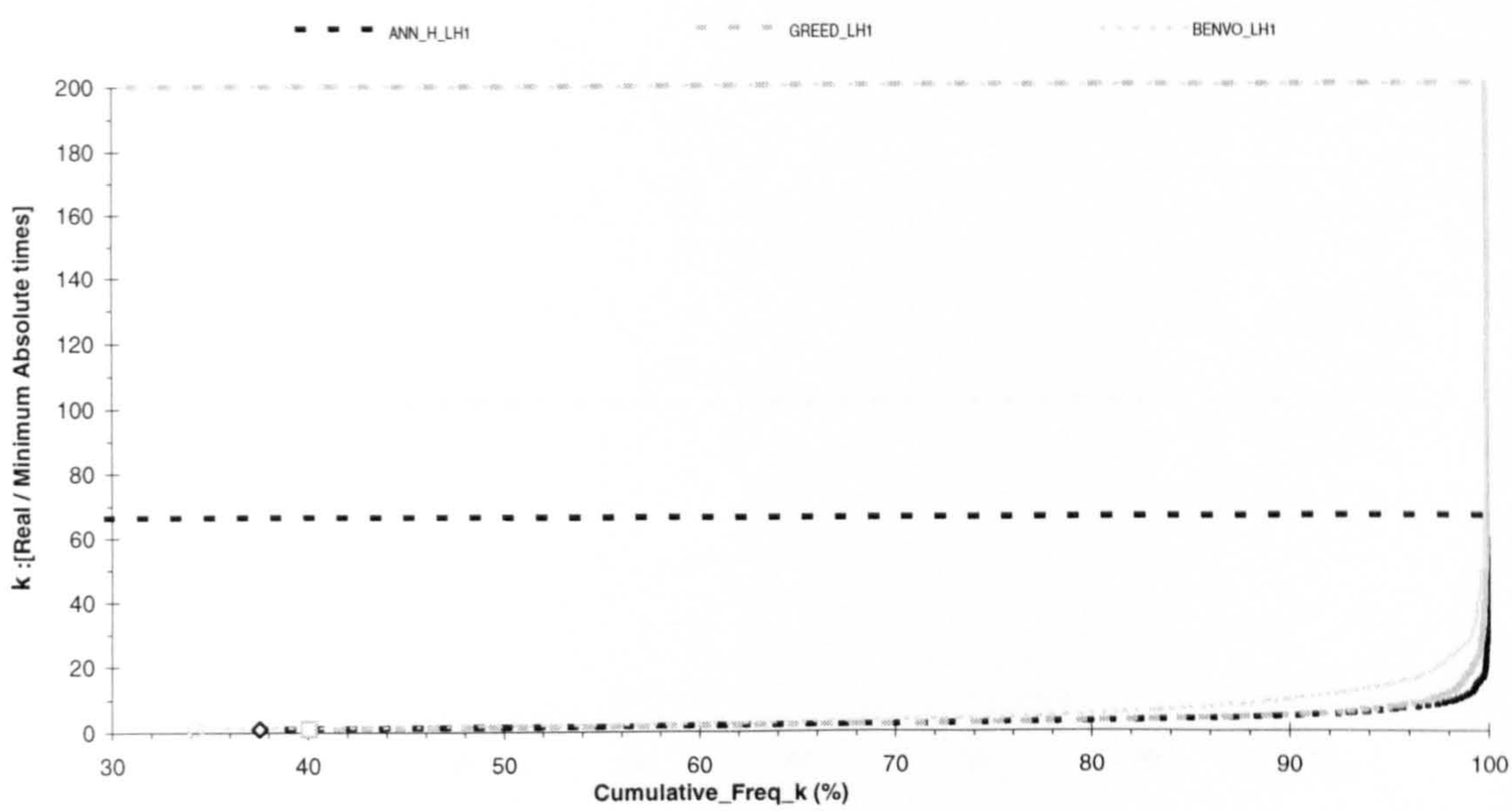


Figure E- 28: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Eight vehicles.

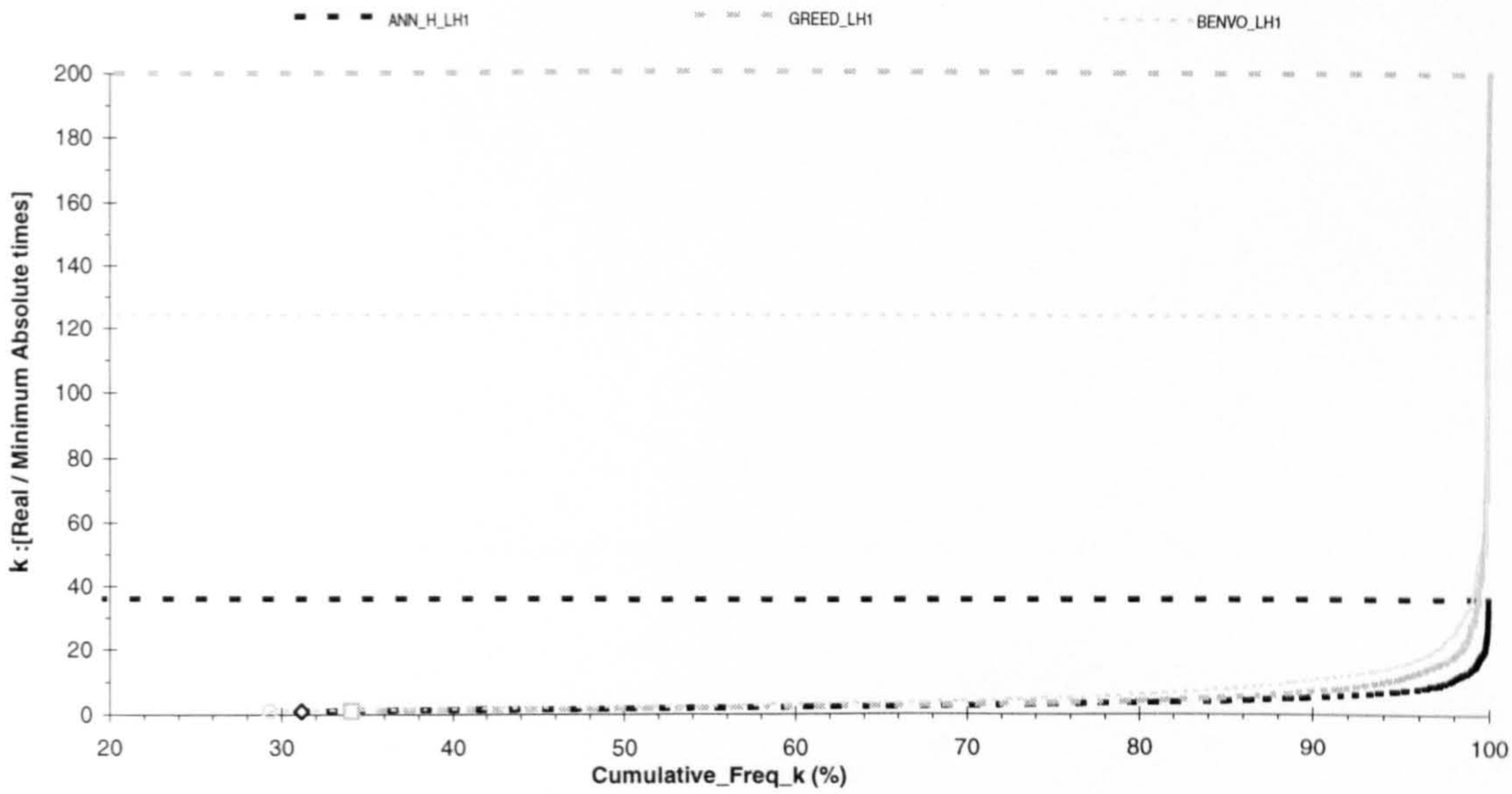


Figure E- 29: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Nine vehicles.

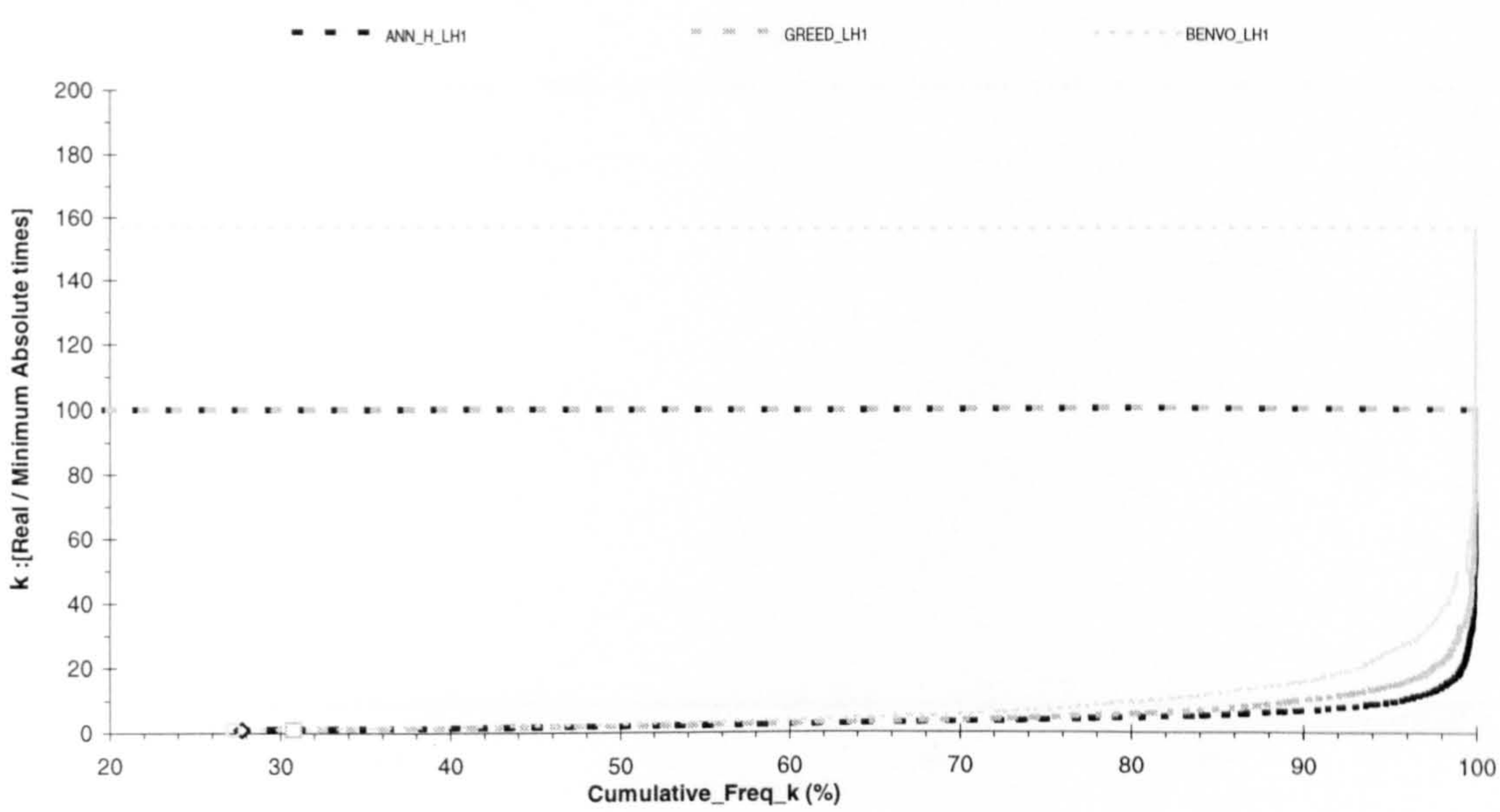


Figure E- 30: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. Limited 'look-ahead' strategies (ANN_H_LH1, GREED_LH1, BENVO_LH1).

Ten vehicles.

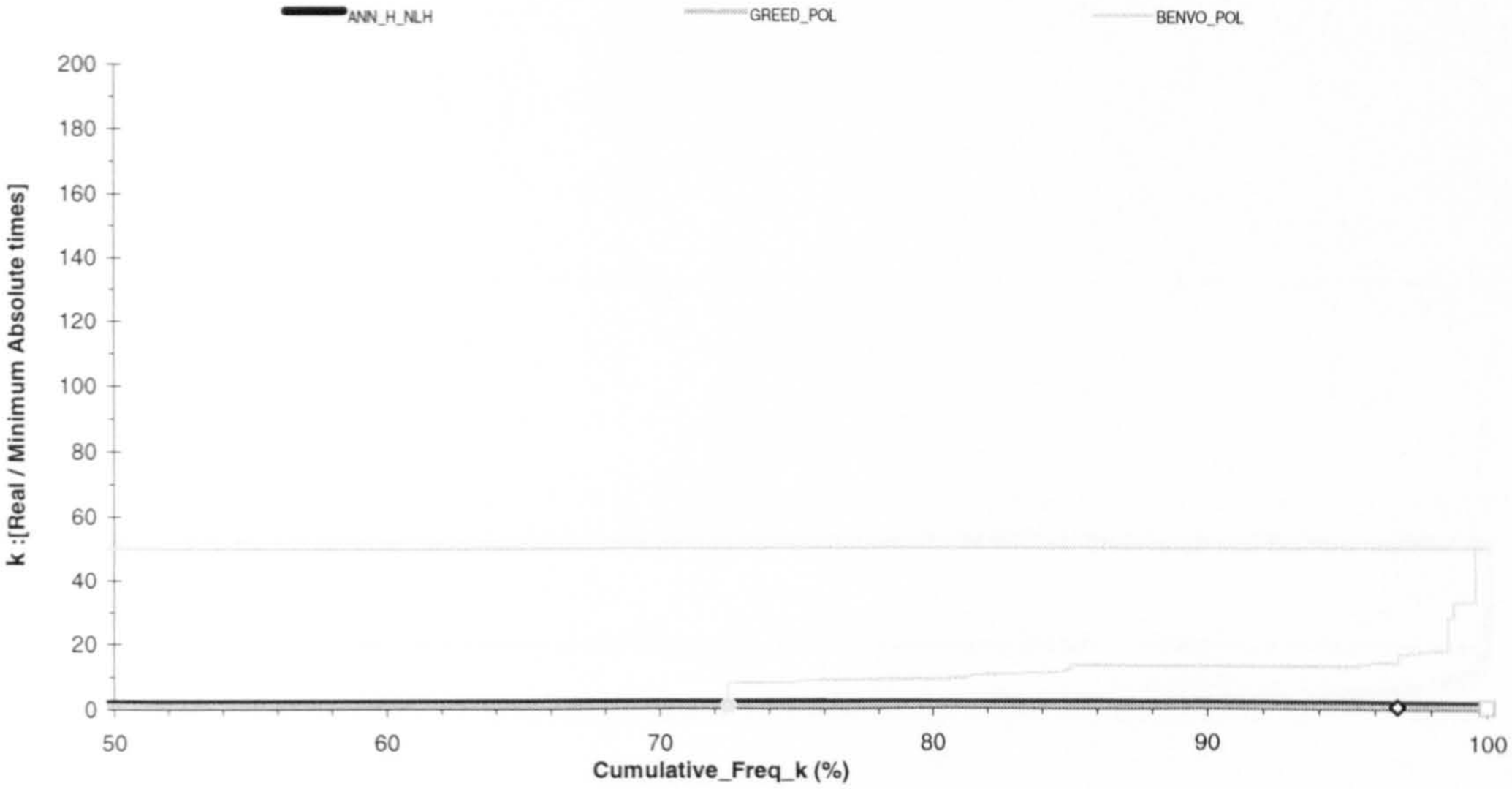


Figure E- 31: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

<u>One vehicle</u>.

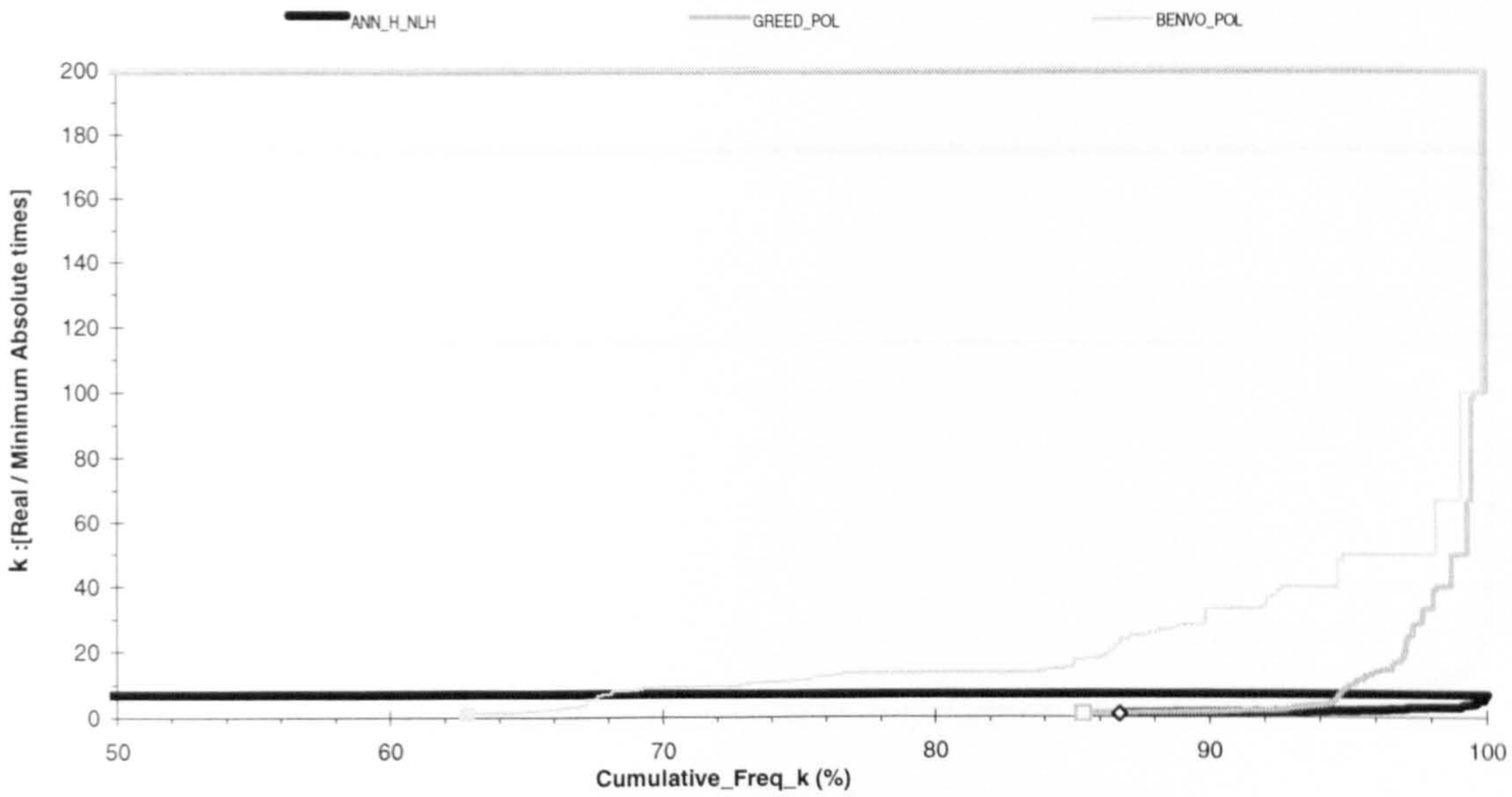


Figure E- 32: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

<u>Two vehicles</u>.

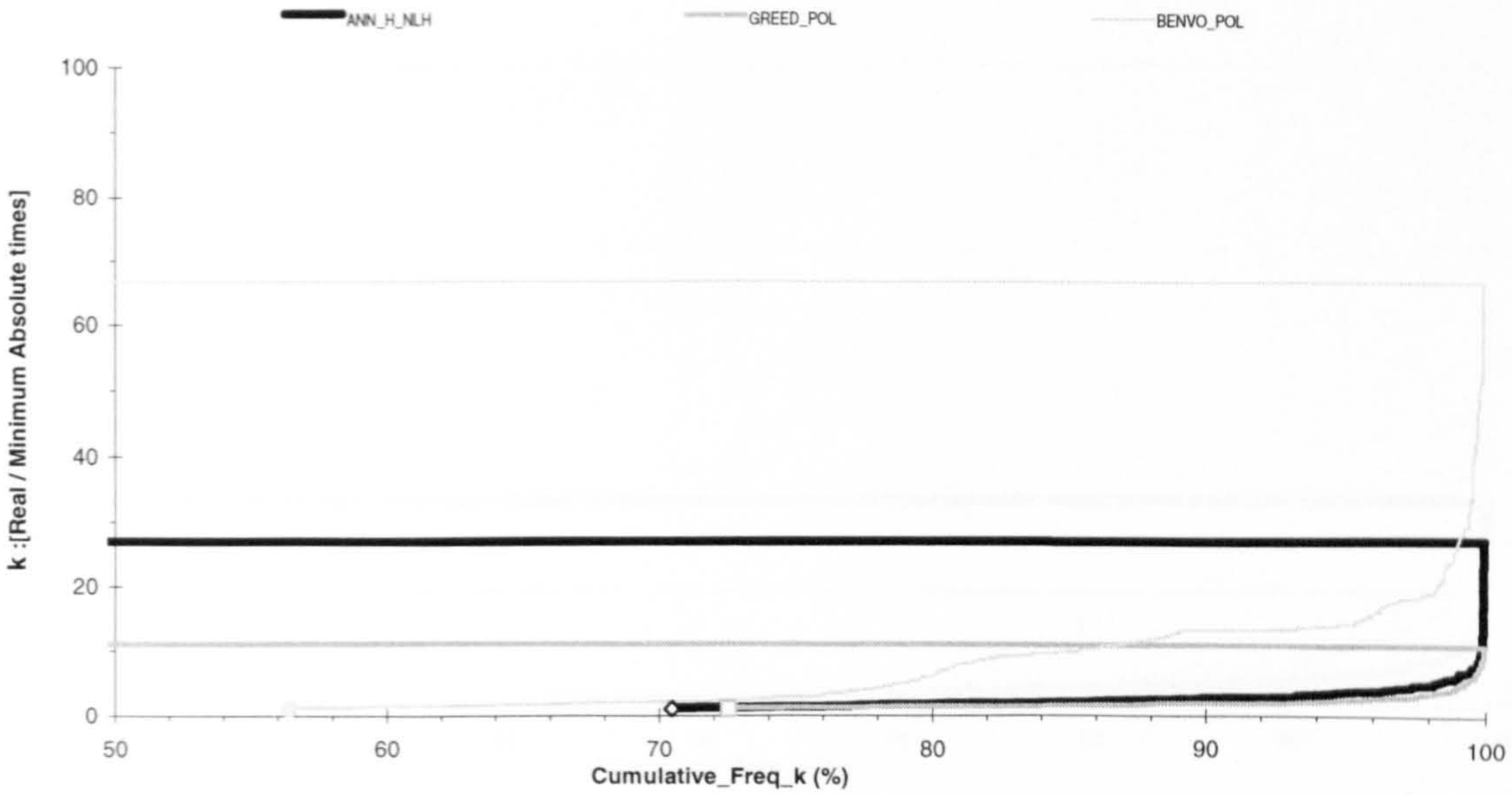


Figure E- 33: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Three vehicles.

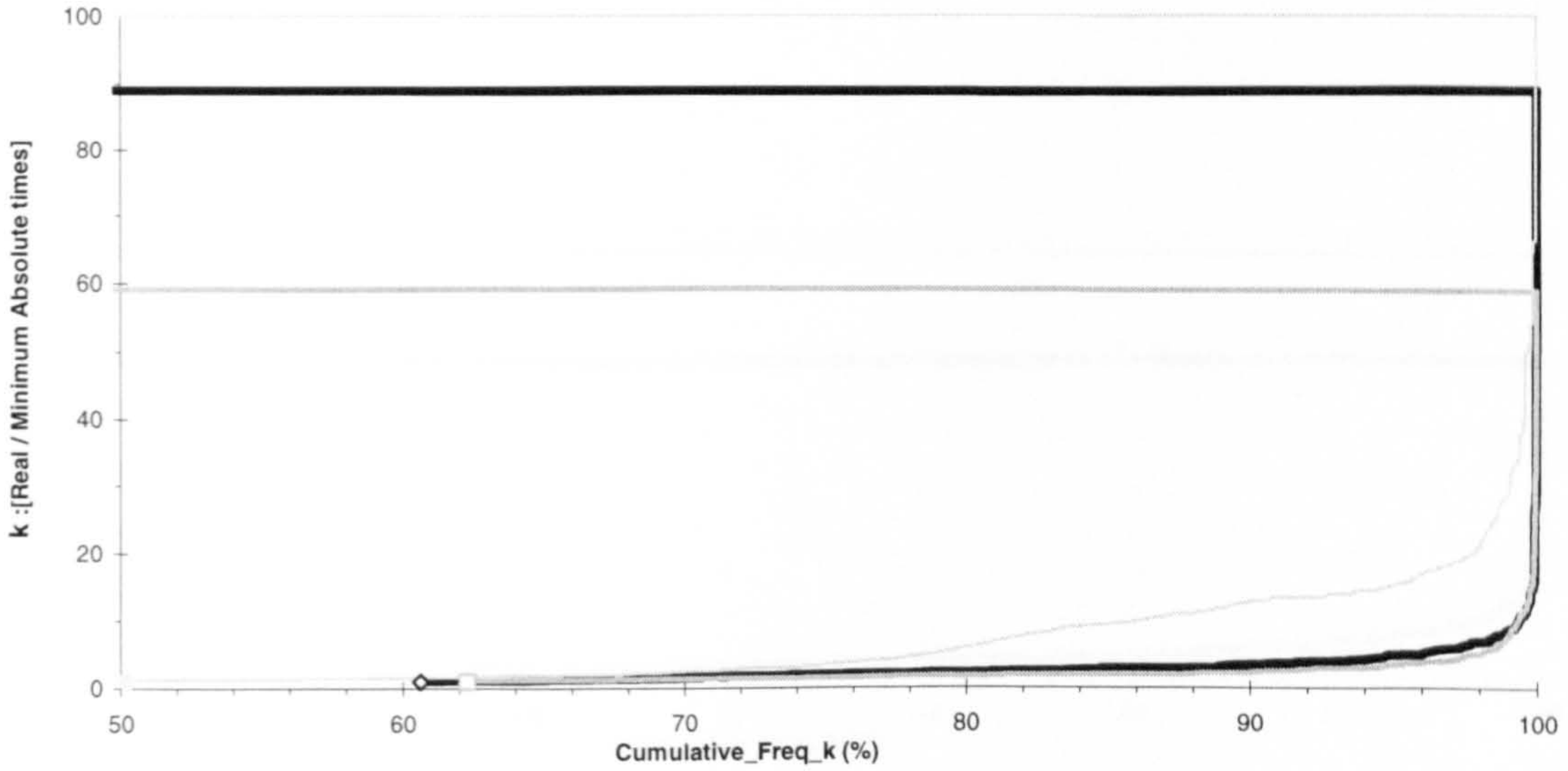


Figure E- 34: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Four vehicles.

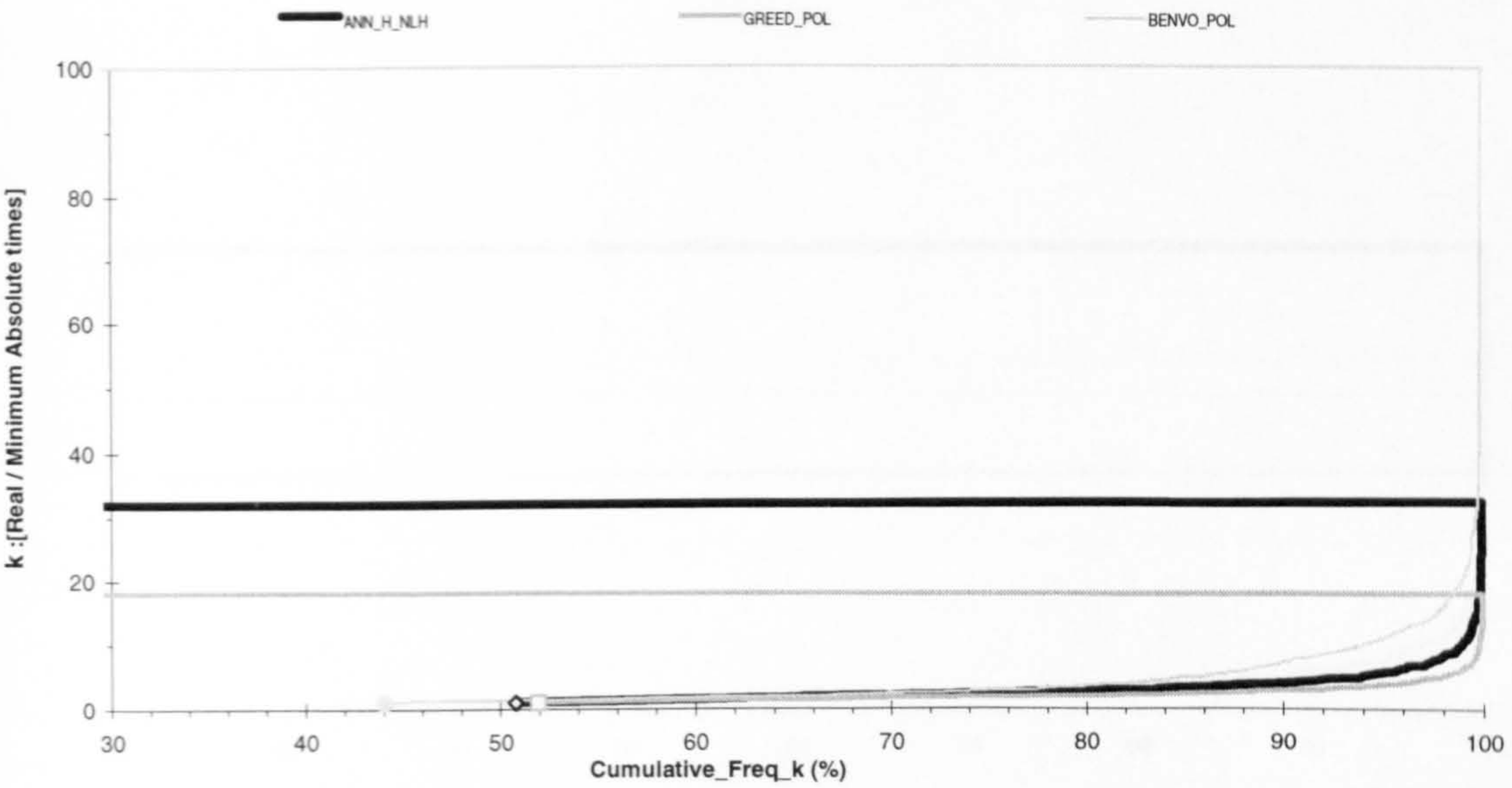


Figure E- 35: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Five vehicles.

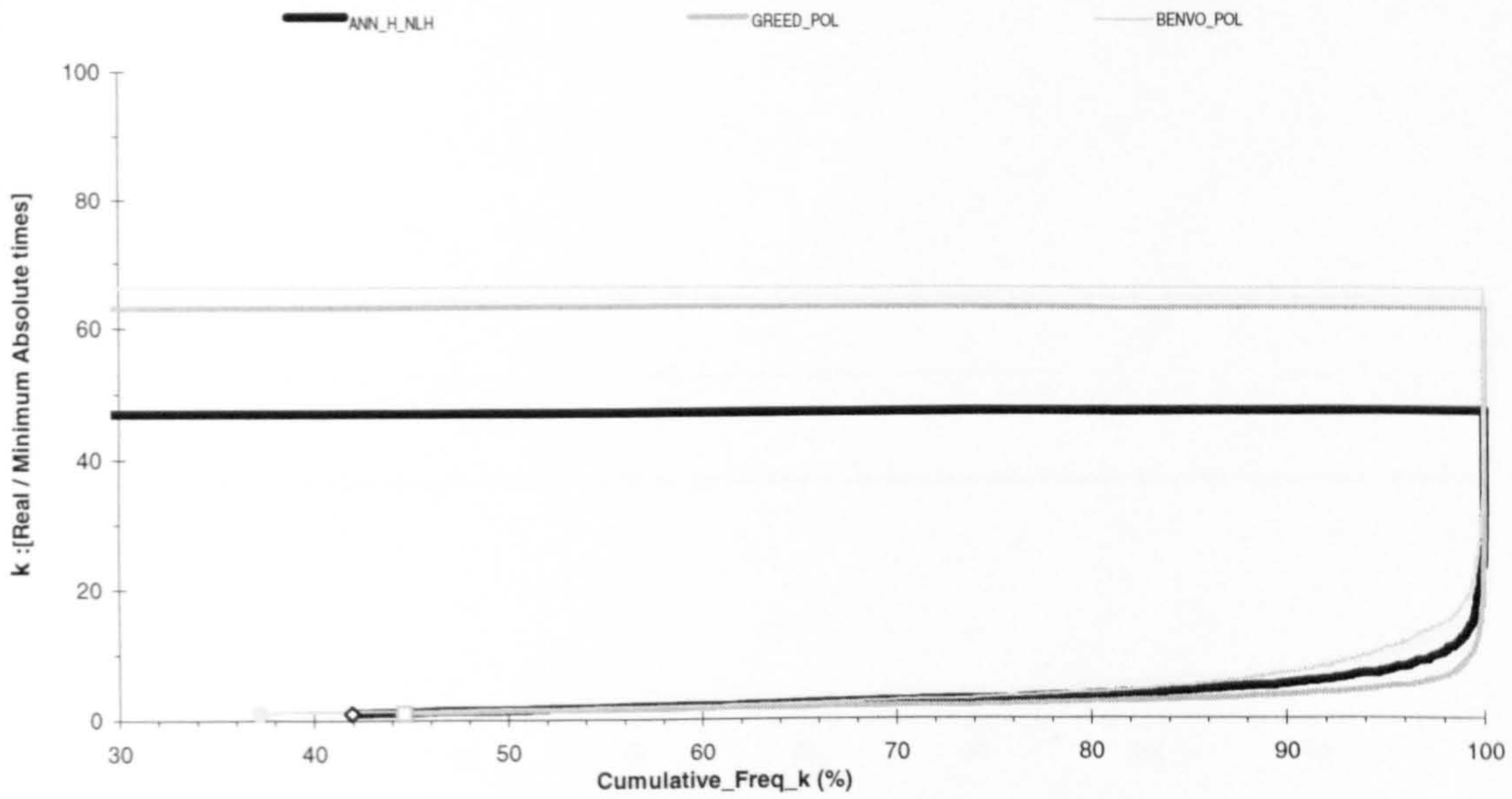


Figure E- 36: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Six vehicles.

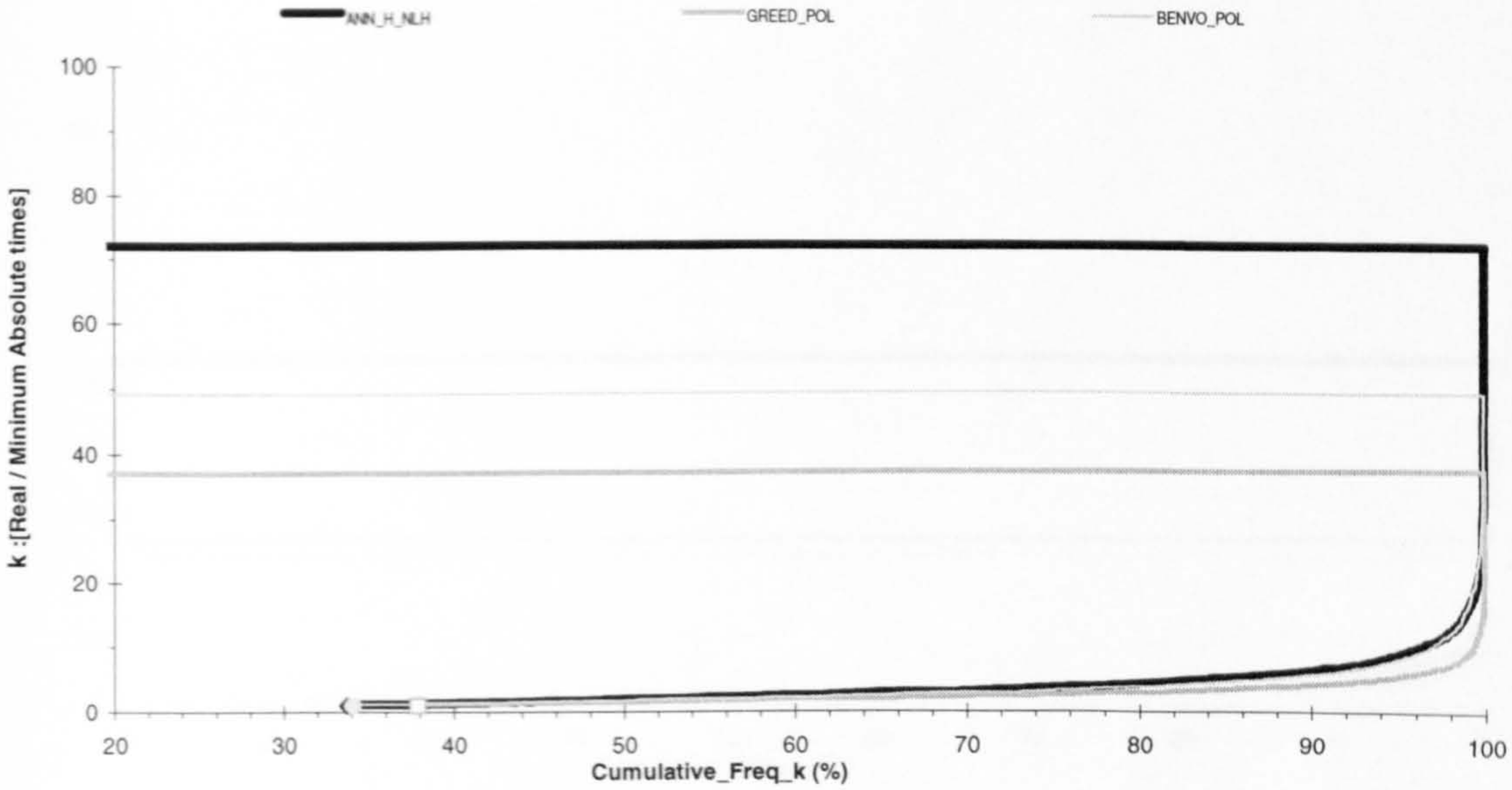


Figure E- 37: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Seven vehicles.

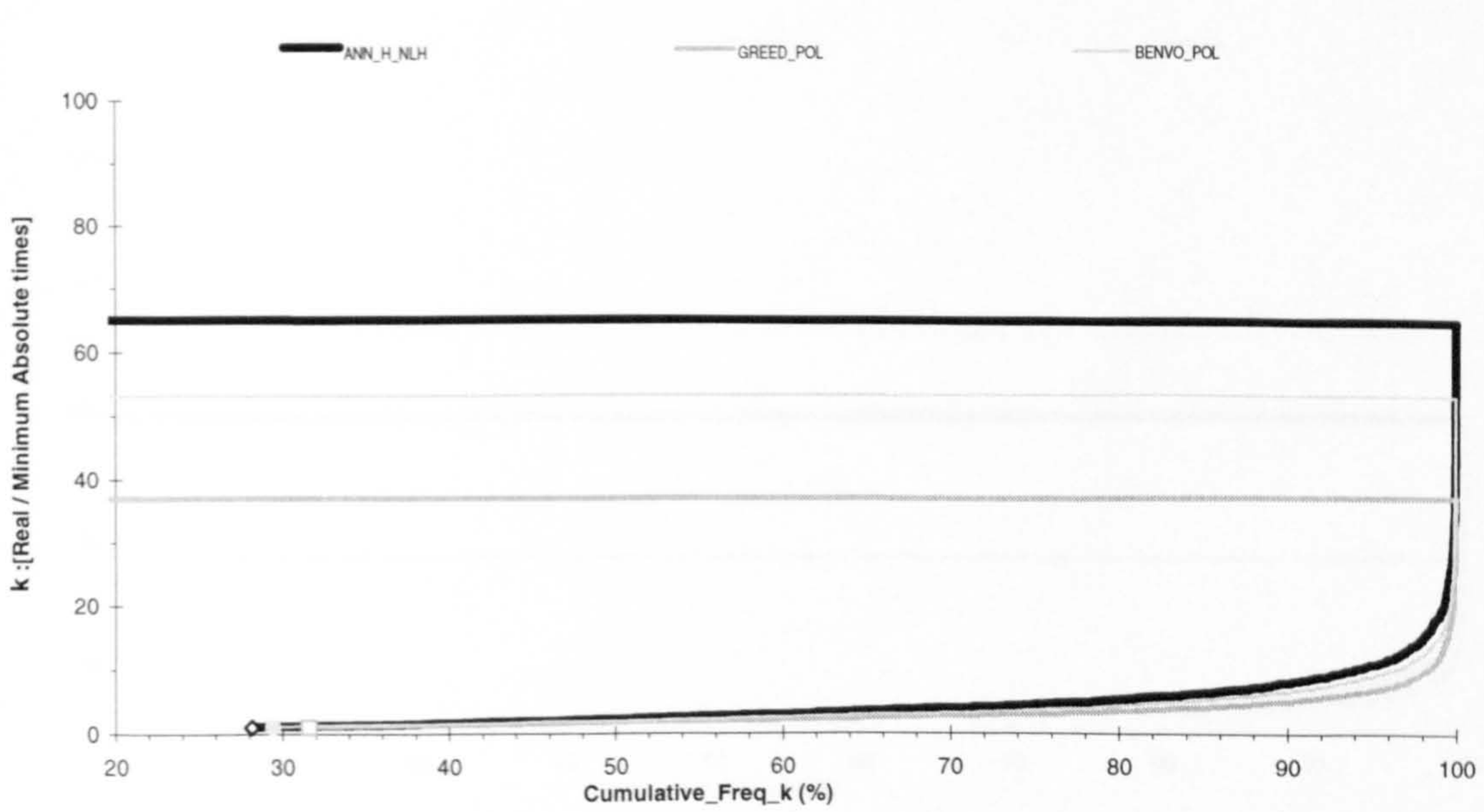


Figure E- 38: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Eight vehicles.

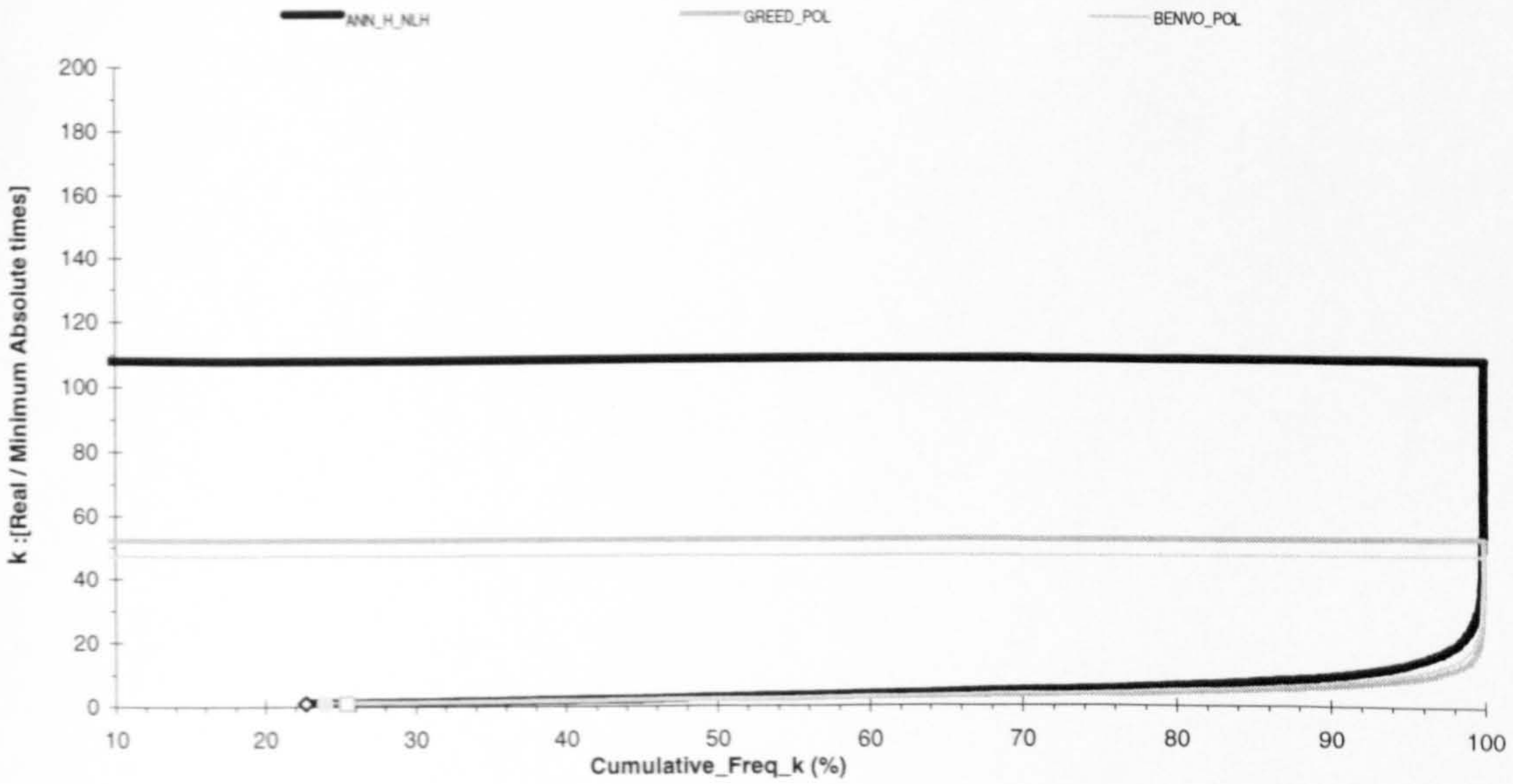


Figure E- 39: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Nine vehicles.

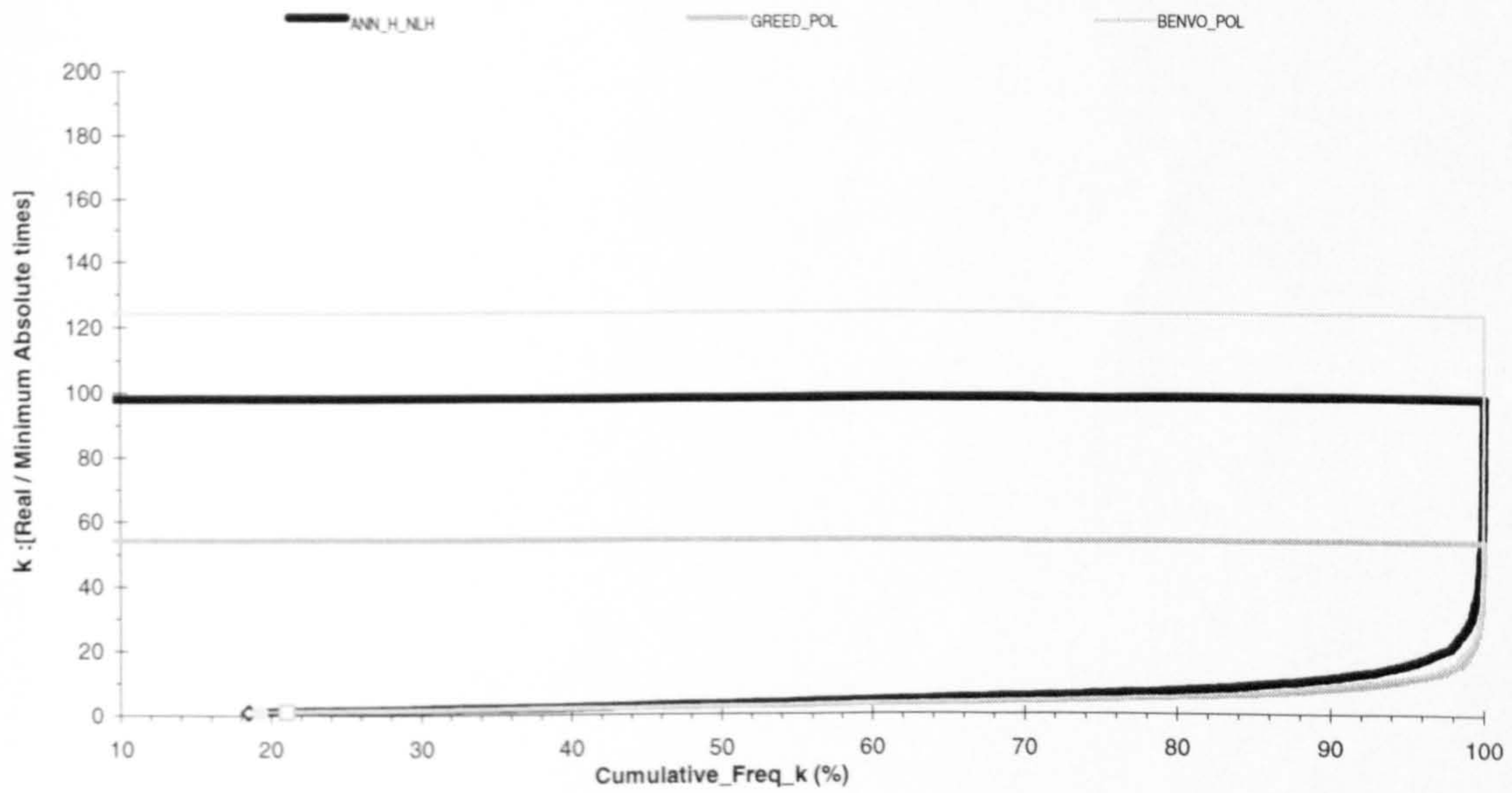


Figure E- 40: Cumulative distribution of *'K'* ratio. Number of cases in horizontal axis (%). Values of *'K'* in vertical axis. No 'look-ahead' strategies (ANN_H_NLH, GREED_POL, BENVO_POL).

Ten vehicles.