

Original citation:

Kusetogullari, Huseyin, Sharif, Haidar, Leeson, Mark S. and Celik, Turgay. (2015) A reduced-uncertainty hybrid evolutionary algorithm for solving dynamic shortest-path routing problem. *Journal of Circuits, Systems and Computers*, 24 (5).

Permanent WRAP url:

<http://wrap.warwick.ac.uk/73771>

Copyright and reuse:

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:

Electronic version of an article published as *Journal of Circuits, Systems and Computers*, 24 (5). 10.1142/S021812661550067X © World Scientific Publishing Company <http://0-www.worldscientific.com.pugwash.lib.warwick.ac.uk/worldscinet/jcsc>

A note on versions:

The version presented here may differ from the published version or, version of record, if you wish to cite this item you are advised to consult the publisher's version. Please see the 'permanent WRAP url' above for details on accessing the published version and note that access may require a subscription. For more information, please contact the WRAP Team at: publications@warwick.ac.uk

warwick**publications**wrap

highlight your research

<http://wrap.warwick.ac.uk>

A reduced uncertainty based hybrid evolutionary algorithm for solving dynamic shortest-path routing problem

Huseyin Kusetogullari

Department of Computer Engineering, Gediz University, Seyrek, Izmir, Turkey

Md. Haidar Sharif*

Department of Computer Engineering, Gediz University, Seyrek, Izmir, Turkey

Mark S. Leeson

The School of Engineering, Warwick University, Coventry, UK

T. Celik

School of Computer Science, University of the Witwatersrand, Johannesburg, South Africa

Received (July 7, 2014)

Revised (December 17, 2014)

Accepted (Day Month Year)

The need of effective packet transmission to deliver advanced performance in wireless networks creates the need to find shortest network paths efficiently and quickly. This paper addresses a Reduced Uncertainty Based Hybrid Evolutionary Algorithm (RUBHEA) to solve Dynamic Shortest Path Routing Problem (DSPRP) effectively and rapidly. Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are integrated as a hybrid algorithm to find the best solution within the search space of dynamically changing networks. Both GA and PSO share context of individuals to reduce uncertainty in RUBHEA. Various regions of search space are explored and learned by RUBHEA. By employing a modified priority encoding method, each individual in both GA and PSO are represented as a potential solution for DSPRP. A Complete statistical analysis has been performed to compare the performance of RUBHEA with various state-of-the-art algorithms. It shows that RUBHEA is considerably superior (reducing the failure rate by up to 50%) to similar approaches with increasing number of nodes encountered in the networks.

Keywords: Shortest path, encoding, decoding, uncertainty reduction, genetic and hybrid algorithms.

1. Introduction

Dynamic Shortest Path (DSP) routing has become a critical and key element in routing protocols to improve the utilization of wireless communication networks. Routing data packets through DSP is normally deemed as an efficient approach to increase the performance of wireless network utilization since it minimizes cost or delay while maximizing

*Corresponding author

quality. Shortest-path routing is also a markedly notable technique for newly emerging technologies, chiefly video-conferencing and video on demand which require high bandwidth, low delay, and low delay jitter¹. DSP discovery in a more efficient and faster manner is a challenging problem in mobile ad hoc networks, wireless sensor networks, etc. Variations of DSPRP have to be solved to gain high throughput communication in a wide variety of network problems e.g., multi-shortest-path², constrained shortest-path³, multi-objective shortest-path⁴, DSP discovery⁵, and travelling salesman problems⁶. Most of these problems are non-polynomial hard and one way of solving them is to assign a cost metric (weight) for each link in the network. To date a range of deterministic algorithms e.g., Dijkstra's algorithm⁷, breadth-first search algorithm⁸, and Bellman-Ford algorithm⁹ have been developed to find the lowest cost or shortest-path routing from a source to a given destination through a network. However, they have limits such as being inflexible or lacking the adaptability needed to achieve shortest-path in rapidly changing network topologies^{10,11,12}. So it is eminent to employ proper new approaches for solving DSPRP rapidly and efficiently^{10,11,12}.

Evolutionary algorithms have drawn considerable attention as DSPRP solvers since they provide a more robust and efficient approach to solve such complex problems^{10,13,14,15,16}. GA¹⁰, genetic programming¹⁷, and evolutionary programming¹⁸ have been proposed for various network problems. The algorithm of Ji et al.¹⁹ integrated stochastic simulation and GA to solve the shortest path problem with stochastic arc length. Vijayalakshmi et al.²⁰ proposed a GA based hybrid algorithm for multi-cast routing in which the crossover and mutation operators have been refined for solving the multi-cast routing problem. An improved hybrid algorithm which applies in multi-cast routing to provide effective solutions for a complex quality of service problem has been proposed by Wang et al.²¹. El-Mihoub et al.²² showed the combined effect of the probability of local search and the learning strategy on the population size requirements of a hybrid algorithm. Incorporation of more than two algorithms was developed by Potter et al.²³ but the algorithmic complexity was greatly increased over a two element solution. Work employed a GA based on variable-length chromosomes to construct a routing table that is a population of strings with each one represents a route. An improved GA was investigated to optimize DSPRP by Yang et al.²⁴. In addition, neural networks have also got a great deal of attention in the area of network problems^{11,26,27}. PSO²⁸ is another efficient heuristic optimization algorithm which may be employed to search for the optimum solution based on social agent behaviour inspired swarm intelligence. Like GA the algorithm of PSO is first initialized in a set of randomly generated potential solutions (particles in PSO and chromosomes in GA), and then performs the search for the optimum solution iteratively. PSO has no evolution operators (e.g., crossover and mutation) but rather finds the optimum solution by updating the so called velocity and position of its particles. Also the population generation process of both GA and PSO distinguishes them from each other and makes them approach to the global solution from different perspectives. Key feature of using both GA and PSO is that they reach the optimal result from the updated population using specific operators and each algorithm singly employed may often become stuck in a local optimum value during searching for the global optima²⁹. In each algorithm, the sequence popula-

tion is generated from the current population which involves weak and strong candidates. The weak candidates undergo more corrections than the strong ones to overcome values at which the algorithms become stuck (local optima) or (occasionally) assist the strong ones to reach a global optimum. Numerous practical approaches can be carefully weighed to overcome stuck values or local minima in meta-heuristic algorithms such as increasing the number of iterations or using a larger population size, which increase both the memory requirements and the route computation time. Also these strategies are very basic and not largely effective in resolving a search space problem. Both GA and PSO have a solution from different perspectives since GA works based on the mechanism of natural selection as well as evolutionary genetics and PSO works based on social adaptation of knowledge. Hence, singly applied algorithms have not provided much opportunity for exploring the global optimum in the different regions of search space. Thus hybrid of GA and PSO plays a vital role to solve many real-world optimization problems.

A comparison of the PSO-based search algorithm and a GA for solving DSPRP is proposed by Mohemmed et al.²⁹. A priority-based indirect path-encoding scheme is used to widen the scope of the search space, and a heuristic operator is used to reduce the probability of invalid loop creations during the path-construction procedure. Authors²⁹ claimed that PSO-based algorithm was superior to that using GA for finding DSP in dynamically changed networks. Moreover, PSO has been applied inter alia to several applications in the fields of evolutionary computing and optimization^{30,31}. Juang¹² presented a Hybrid of GA and PSO (HGAPSO), an evolutionary learning algorithm, for recurrent fuzzy network design. In the algorithmic steps, GA and PSO work with the same population which means that the top 50% of individuals are evaluated and enhanced by using the operators of GA and PSO. Ji¹⁹ proposed a hybrid intelligent algorithm which was integrating stochastic simulation and GA to solve the shortest-path problem with stochastic arc length. Marinakis et al.⁶ addressed a Hybrid of Genetic Algorithm and Particle Swarm Optimization (HybGENPSO) for the vehicle routing problem. In the algorithmic steps, randomly created individuals are first placed in a population pool, and then the operators of GA and PSO are respectively applied to the individuals to produce the new generation. The authors⁶ claimed that by the effect of PSO the use of an intermediate phase between the two generations will give more efficient individuals and, henceforth, will improve the effectiveness of the algorithm. A hybrid algorithm based on ant colony optimization with scatter search was implemented for the vehicle routing problem in the field of distribution and logistics²⁵, delivering higher performance solutions relative to other competing heuristics.

In spite of different efforts the desired level of efficiency and rapidity in DSPRP algorithms did not come along. To solve DSPRP efficiently and rapidly a bit more, we have developed a new reduced uncertainty based hybrid evolutionary algorithm (RUBHEA) which is a hybrid of GA and PSO. From a given dynamically changing network model, RUBHEA finds a set of dynamic paths with their cost values which are then evaluated for obtaining the dynamic shortest-path as a final result quickly and efficiently.

A key advantage of RUBHEA is that it shares information and then learns and reduces the uncertainty during the algorithmic iterations. Primary presentations of the proposed RUBHEA are included to: (i) Optimum solution is obtained by exchanging information

between GA and PSO. Candidates in the sub-populations will be exchanged to compose the new sub-populations and the solution strategy will avoid the need to change the parameters of the operators in the algorithms for finding the global optimum. (ii) RUBHEA learns DSPRP and reduces the uncertainty in the problem. It adopts the modified search based priority encoding method to represent a path in an undirected network. Multiple comparisons without statistical tests and with statistical tests considering various scales of networks illustrate the effectiveness and the efficiency of RUBHEA by comparing with the existing state-of-art dynamic shortest-path routing algorithms based on GA (e.g., Ahn et al.¹⁰), PSO (e.g., Mohammed et al.²⁹), as well as hybrid of GA and PSO (e.g., Juang et al.¹² and Marinakis et al.⁶).

The rest of the paper is organized as follows. List of acronyms and symbols used in this paper will be demonstrated in section 2. DSPRP problem will be defined in section 3. Existing approaches for finding shortest-path will be explained in section 4. The proposed approach RUBHEA for finding shortest-path will be presented in section 5 and, thereafter, the modified version of encoding method for solving DSPRP will be discussed. Results and discussions will be provided in section 6, and finally the paper will be concluded in section 7.

2. Glossaries

A lot of acronyms and symbols have been used in this paper. In this section, we have demonstrated those acronyms and symbols so that readers can easily follow them throughout the paper.

Acronyms

DSPRP Dynamic Shortest Path Routing Problem. 1–5, 7, 9, 11, 12, 16, 19, 21, 25, 31, 32

DSP Dynamic Shortest Path. 1–3, 6, 7, 9, 16, 20, 21, 29

GA Genetic Algorithm. 1–4, 7–13, 15–18, 20–22, 26, 31, 32

HGAPSO Hybrid of GA and PSO. 3, 20, 26, 30, 31

HybGENPSO Hybrid of Genetic Algorithm and Particle Swarm Optimization. 3, 20, 21, 26, 30, 31

PSO Particle Swarm Optimization. 1–4, 7, 10–13, 15–17, 20–22, 26, 31, 32

RUBHEA Reduced Uncertainty Based Hybrid Evolutionary Algorithm. 1, 3–7, 11–26, 29–32

List of Symbols

$C(c_{k_1}^g)$ Fitness value of individual k_1 in the GA at iteration number g . 15

$C(c_{k_3}^g)$ Cost value of selected weaker chromosome k_3 at iteration number g . 15

$C(p_{k_2}^g)$ Fitness value of individual k_2 in the PSO at iteration number g . 15

$C(p_{k_3}^g)$ Fitness value of selected fitter particle k_3 at iteration number g . 15

D Destination node. 5, 9

$E(t)$ Set of time varying links at time t . 4, 5, 13

$G(t)$ Network model at time t . 4, 13

- $H(x)$ Entropy of x at particular instance of X . 6
- $H(y \setminus x)$ Relationship between past and future network topologies. 7
- IDs Identities of nodes. 7, 13, 16
- $I_{i,j}(t)$ Link connection indicator between nodes i and j at time t . 6, 9
- $I_{i,j}(t+1)$ Link connection indicator between nodes i and j at time $t+1$. 9
- $I_{j,i}(t+1)$ Link connection indicator between nodes j and i at time $t+1$. 9
- J Algorithm for describing behaviors of nodes and links versus time. 4
- $N(t)$ Set of time varying nodes at time t . 4, 5, 13
- N Total number of observations. 27, 28, 30–32
- $P(c_{k_1}^g)$ Probability of occurrence of chromosome k_1 at iteration number g . 15
- $P(p_{k_2}^g)$ Probability of occurrence of particle k_2 at iteration number g . 15
- $P(q_i^t)$ Probability of q_i^t in Q^t . 6, 7
- $P(q_i^{t+1} \setminus q_i^t)$ Probability of $q_i^{t+1} \setminus q_i^t$. 7
- Q^t State space at time t . 6, 7
- Q^{t+1} State space at time $t+1$. 7
- $R_{i,j}(t)$ Estimation learned about the least cost to reach destination node at time t . 9
- S Source node. 5, 9
- $W(t)$ Set of time varying cost values over links at time t . 4, 5, 13
- $W_{i,j}(t+1)$ Cost value associated with each link (i, j) at time $t+1$. 9
- X A random process. 6
- Y Next random process. 7
- ℓ Randomly selected point in the chromosome. 8
- ϵ Independent population data set. 15
- η Intersection value. 12–15, 17, 21–26, 29–32
- γ^g Number of discarded candidates at iteration g . 12
- μ_1^g The mean of cost values of population in the GA at iteration number g . 15
- μ_2^g The mean of cost values of population in the PSO at iteration number g . 15
- μ_3^g The mean of fitness values of population in the RUBHEA at iteration number g . 15
- ω Inertia weight. 10, 11, 13
- ρ^g Sub-population size at iteration g . 12, 15
- σ_1^g The standard deviation of cost values of population in the GA at iteration number g . 15
- σ_2^g The standard deviation of cost values of population in the PSO at iteration number g . 15
- σ_3^g The standard deviation of fitness values of population in the RUBHEA at iteration number g . 15
- τ_n^t n^{th} node in $N(t)$. 5
- \vec{v}_k Velocity vector of individual particle k . 10, 12, 13
- \vec{x}^{gbest} Global best position in the search space. 10
- \vec{x}_k Position vector of individual particle k . 10, 12, 13
- \vec{x}_k^{best} Personal best position of individual particle k in the search space. 10
- \vec{z}_a a^{th} chromosome. 8
- \vec{z}_b b^{th} chromosome. 8

- c_1, c_2 Two learning factors in the PSO. 10, 11, 13
- d Dimensional search space. 8, 10, 13
- e_m^t m^{th} link in $E(t)$. 5
- $f(t) : J$ Mapping function. 4, 5
- f^{gbest} Global best objective function's value. 10
- f_k Objective function value of particle k . 10
- f_k^{best} Personal best value of individual particle k . 10
- g Generation index. 10, 12, 13, 15
- i, j Pair of selected two nodes. 6, 9
- k_3 Individual particle and chromosome in the RUBHEA. 15
- k_c Number of chromosomes. 13, 15
- k_p Number of particles. 10, 13, 15
- k Particle number in the PSO. 10
- m^+ Convergent value. 15
- n Number of nodes in network. 6, 10, 13
- p_c Crossover rate. 8, 13, 20
- p_m Mutation rate. 8, 13, 20
- q_β^t β^{th} element in Q^t . 6
- r, s Randomly selected two crossover points. 8
- t Time or step number. 6, 7, 9
- u_1, u_2 Uniformly distributed two random variables. 11, 13
- x Particular instance of X . 6
- y Particular instance of Y . 7

3. Problem Definition in Dynamic Network Models

In this section, we have focussed on the explanation of dynamically changing networks followed by DSPRP as well as the measurements of uncertainty and entropy in dynamic network models.

3.1. Dynamically changing networks

A complete definition of network $G(t)$ must include both structural and behavioural information so the definition of dynamic network model is:

$$G(t) = \{N(t), E(t), W(t), f(t) : J\}, \quad (1)$$

where t is the time or step number, $N(t)$ is a set of time-varying nodes (also known as vertices), $E(t)$ is a set of time-varying links connected between pair of nodes (also known as edges), $W(t)$ is a set of time-varying cost values over links, $f(t) : N(t) \times N(t)$ is a time-varying mapping function that connects node pairs, yielding topology or network graph and J is the algorithm for describing behaviours of nodes and links versus time. The number and properties of $N(t)$, $E(t)$, and $f(t) : J$ are allowed to change

as time passes in a network. In this sense, a network is a dynamic. For instance, the shape of a network, as defined by behaviors and the mapping function $f(t) : J$, may change as links are dropped, added, or rewired (switched from one node to another) over time. It changes based on the movement and transmission range of the nodes in network topologies. Besides this, $W(t)$ has changed over time. Furthermore, a network may grow with the addition of nodes and links over time. Let $N(t) = \{\tau_1^t, \tau_2^t, \tau_3^t, \dots, \tau_n^t\}$ and $E(t) = \{e_1^t, e_2^t, e_3^t, \dots, e_m^t\} \subseteq N(t) \times N(t)$ be a set of nodes and a set of arcs, respectively. We assume that $N(t)$ and $E(t)$ are finite sets. Let S and D be two nodes of $N(t)$ such that $S \neq D$. Thus, one of the possible paths from source S to destination D may be represented as $\{S, \tau_1^t, e_1^t, \tau_2^t, e_2^t, \dots, D\}$, respectively. However, many different paths can be obtained from source node to destination node in the network topology so the uncertainty in network topologies should be reduced for finding the shortest path rapidly and efficiently.

3.2. DSPRP

A network graph is a collection of vertices and edges randomly connecting them pairwise. Network graphs can be static or dynamic. In the static graph the number of nodes and links, the shape of the mapping function and other properties of graphs remain the same whereas in the dynamic graph they may change over time³². Therefore, it is very hard to resolve dynamically changing graphs compared to the static network graphs. In dynamically change networks, finding shortest path is a very complex problem as it is necessary to compute the costs of all maintained paths between source and destination nodes. By applying the proposed method RUBHEA, we will prevent recomputing all paths once we obtain the dynamic shortest routing path. There are many research efforts reported in the literature of maintaining shortest paths on dynamic graphs. For examples, Dijkstra's algorithm⁷, breadth-first search algorithm⁸, and Bellman-Ford algorithm⁹ had been proposed and improved to obtain the shortest-path routing from a source node to a destination node in networks. These algorithms are very inefficient when the occurrence of link state changes in a network. Some other methods maintain shortest paths in Hypergraphs³³, in planar graphs^{32,34}, some require un-weighted graphs³⁵, and some permit only integer edge weights^{36,37}, and its variant³⁸. Many other shortest path problems have been solved by using different approaches^{39,40,40,41,42,43}. In this work, we have focused on developing a new method RUBHEA to find shortest-routing path efficiently and quickly without recomputing all paths in dynamically changing networks. In a generated dynamic network model, it was considered that each node has a fixed transmission range and an edge link is created if two nodes are within the transmission range of each other. For instance, node S is in the range of node C , and hence an edge can be constructed between each other but an edge cannot be created between nodes S and K because node S is not in the transmission range of node K as illustrated in Fig. 1. The edge construction process is used for all nodes in the dynamic network model. Fig. 1 illustrates a small dynamically changing network model including 12 nodes and 22 indirect links. In the network model, the server node (S) transmits data packets to the receiver node (D) through the shortest path for effective

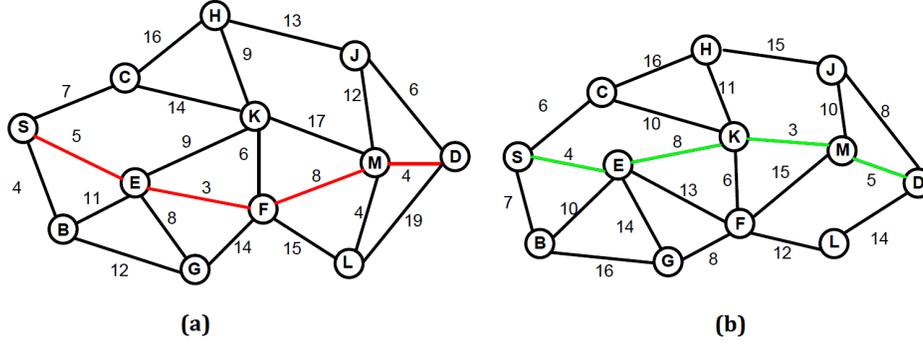


Fig. 1. Example of a small dynamically changing network model.

communication. To find the dynamic shortest path, each edge between the pair of nodes is assigned with an independent weight value as shown over the edges in Fig. 1. By using RUBHEA the dynamic shortest path (S-E-F-M-D) shown by the red line is obtained in Fig. 1 (a). After the weights of the edges are changed in the network model, the costs of all paths between source and destination nodes are recomputed and a new shortest path (S-E-K-M-D) is found as shown by the green line in the Fig. 1 (b).

3.3. Uncertainty and entropy measurements in dynamic networks

Uncertainty in dynamically changing network topology has the considerable effect on finding DSP. It should be as small as possible to obtain DSP or solve the optimization problem rapidly and efficiently. Otherwise, it will be extremely complex problem for efficient solution. For instance, finding DSP in a dynamically changing network is a very hard problem because of the movement of the nodes and variations of the connectivity between nodes. To offer a great solution for the complexity problem, uncertainty is reduced both in the dynamically changed network topologies and in the hybrid algorithms. A network of n nodes is defined by the set of indicator variables $I_{i,j}(t)$ for $i, j \in 1, 2, 3, \dots, n$. We note that a node cannot be adjacent to itself which means that a node cannot transmit a message to itself and it is described as $I_{i,j}(t) = 0$ for all $i = j$. An adjacent matrix represents the entire connectivity of the network topology at time t . The set of allowable configurations of that adjacent matrix forms a state space $Q^t = \{q_1^t, q_2^t, q_3^t, \dots, q_\beta^t\}$ where $\beta = 2^{n(n-1)/2}$. Let X be a random process denoting the trajectory of the networks topology through the state space Q^t and let x denote a particular instance of X , then the information entropy^{44,45} can be estimated as:

$$H(x) = - \sum_{q_i^t \in Q^t} P(q_i^t) \log_2 P(q_i^t) \quad (2)$$

where $\sum_{q_i^t \in Q^t} p(q_i^t) = 1$. If $P(q_i^t) = 0$ for some i , the value of the corresponding summand $0 \log_2 0$ is taken to be 0. $H(x) = 0$ signifies that there is no uncertainty and consequently, there is no information⁴⁵.

The main advantage of the proposed method is that it learns and reduces the uncertainty during iterations. On estimating DSP at time t , the connectivity between nodes in the network topology and adjacent matrix may change. The relationship between past and future network topologies is stated as follow:

$$H(y \setminus x) = - \sum_{q_i^t \in Q^t} P(q_i^t) \sum_{q_i^{t+1} \in Q^{t+1}} P(q_i^{t+1} \setminus q_i^t) \log_2 \frac{1}{P(q_i^{t+1} \setminus q_i^t)} \quad (3)$$

where Y is a random process denoting the trajectory of the new network topology through the state space Q^{t+1} and y denotes a particular instance of Y . In Eq. 3, the relationship between past and future network topologies can be clearly analysed. For instance, the entropy of the unknown result of DSP is minimum in the network topology at time $(t + 1)$ since $H(y \setminus x) = 0$. This outcome proves that the new network topology is completely same with the past network topology. On the other hand, the adjacent matrices of past and future network topologies are examined and compared to find the amount of the changes between the network topologies in the situation of maximum uncertainty as it is most difficult to predict the outcome of DSP. As a result, continuously changing problem becomes easier to be solved by decreasing the uncertainty from the known information and the proposed search space algorithm tries to find the new global optima only in the part of the network topology changed.

4. Existing Approaches for Finding Shortest-path

Ahn et al.¹⁰ and Mohammed et al.²⁹ demonstrated the effectiveness of using GA and PSO algorithms for finding shortest path in communication networks respectively. In this section, existing shortest path algorithms which are GA¹⁰ and PSO²⁹ to find shortest path will be clearly explained. In the next section, the proposed method which is reduced uncertainty based hybrid evolutionary algorithm RUBHEA will be presented to compute the shortest path in dynamically changing networks.

4.1. GA Approach

In the standard GA method¹⁰, a number of candidate solutions (chromosomes or population) are randomly generated with each one consisting of a fixed number of variables called genes or priority of node *IDs*. And then the chromosomes are examined to find the optimal solutions (fitness values) according to the given objective function. Priority values of genes are reflected in the node *IDs* of the chromosome which represents an order of nodes in a routing path. Each chromosome should not exceed the total number of nodes in the network. The gene of the first node ID is specified as a source node and other nodes of the chromosome are selected randomly or heuristically until the destination node is reached. Thus a routing path is constructed as a potential solution.

In the existing GA method to solve the DSPRP¹⁰, the best encoded chromosomes are selected based on the best fitness values to create the next generation by applying crossover and/or mutation. Crossover method is applied to two selected chromosomes (individuals)

known as parents and the results in new chromosomes known as offspring. Stochastic selection operators are effective approaches for choosing which (fitter) chromosomes will survive because they will have a higher chance of being selected than less fit ones. Nevertheless, even the weak chromosomes typically have a chance to become a parent or to survive. Sometimes individuals may rapidly come to dominate the population causing convergence to a local minimum. To avoid this, a larger population with a modest crossover rate p_c but a relatively high mutation rate p_m will cause rapid convergence to the known optimum solution and results in one new chromosome. Applying crossover and mutation leads to a set of new offspring chromosomes. Based on their fitness, these compete with the old chromosomes for a place in the next generation. This complete process can be iterated until an optimal fitness value is estimated or a previously set time limit is reached.

4.1.1. Offspring generation operators

Crossover and mutation operators are the major evolutionary and distinguishable operators of GA. Crossover operates on two chromosomes at any iteration number and creates offspring by combining genes of both chromosomes. Thus there is a transfer of information or genes between the candidates/individuals that created offspring/offsprings collect the beneficial information to find better result. The most known and used crossover operators are partial-mapped crossover, order crossover, cycle crossover, position-based crossover, heuristic crossover^{10,24}. In the crossover method, the most significant point is to define a proper crossover operator for the corresponding problem since it plays a major role in GA to achieve a better performance. Consider two different individuals \vec{z}_a and \vec{z}_b in d -dimensional search space of the given population, i.e.,

$$\begin{aligned}\vec{z}_a &= z_{a1}, z_{a2}, \dots, z_{ar}, \dots, z_{as}, \dots, z_{ad} \\ \vec{z}_b &= z_{b1}, z_{b2}, \dots, z_{br}, \dots, z_{bs}, \dots, z_{bd}\end{aligned}\quad (4)$$

where r, s denote two randomly selected crossover points denote two randomly selected crossover points, and $r < s$. After the crossover operator employed, the produced individuals or new candidate solutions become

$$\begin{aligned}\vec{z}_a &= z_{a1}, z_{a2}, \dots, z_{br}, \dots, z_{bs}, \dots, z_{ad} \\ \vec{z}_b &= z_{b1}, z_{b2}, \dots, z_{ar}, \dots, z_{as}, \dots, z_{bd}.\end{aligned}\quad (5)$$

Crossover probability determines the number of offspring individuals produced. Unlike the crossover, mutation is an operator which produces random changes of genes in various chromosomes. Mutation serves the crucial role of either replacing the genes lost from the population during the selection process so that they can be tried in a new context or providing the genes that were not present in the initial population. Let us suppose $\vec{z}_a = z_{a1}, z_{a2}, \dots, z_{a\ell}, \dots, z_{ad}$ where ℓ denotes the randomly selected point in the chromosome. Consequently, ℓ^{th} gene $z_{a\ell}$ of the individual can be changed or mutated to produce another individual. Mutation probability rate is defined as the percentage of the total number of mutational genes in the population. If it is too low, many genes that would have been useful are never tried out, while if it is too high, there will be much random perturbation,

the offspring will start to lose their resemblance to the ability to learn from the history of the search.

4.1.2. Selection operator and fitness function

Major role of applying the selection operator is to improve the quality of generated population. Roulette wheel selection operator is an effective approach to choose fitter chromosomes⁴⁶. Fitter chromosomes will have a higher chance of being selected than less fit ones⁴⁶. Nevertheless, even the weak chromosomes typically have a chance to become a parent or to survive. Superior individuals should have more opportunities to produce the next generation because they have a higher probability of reaching the feasible solution. In this work, Roulette wheel selection operator is used for choosing two fitter chromosomes from the population to create offspring. In a GA, the quality of a represented chromosome or potential solution is estimated by a fitness function, whose purpose is to map a chromosome representation into a scalar value or a cost value. For the DSP routing problem, the fitness function estimates a cost value for each chromosome that is based on the distance to the global optimum. Hence strong and weak candidates can be evaluated according to their fitness values and multiple offspring are produced based on them.

Let S and D denote source and destination nodes, respectively and $I_{i,j}(t)$ be the link connection indicator between nodes i and j which plays the role of a chromosome map providing information on whether the link from node i to node j is included in a routing path or not at time t , i.e.,

$$I_{i,j}(t) = \begin{cases} 1, & \text{if the link from node } i \text{ to node } j \\ & \text{exists in the routing path;} \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The classical fitness function^{10,29} has been modified to solve the DSPRP as:

$$\sum_{i=S}^D \sum_{\substack{i=S \\ j \neq i}}^D W_{i,j}(t+1) \cdot I_{i,j}(t+1) + R_{i,j}(t) \cdot I_{i,j}(t) \quad (7)$$

$$\sum_{\substack{i=S \\ j \neq i}}^D I_{i,j}(t+1) - \sum_{\substack{i=S \\ j \neq i}}^D I_{j,i}(t+1) = \begin{cases} 1, & \text{if } i = S; \\ -1, & \text{if } i = D; \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

$$\sum_{\substack{i=S \\ j \neq i}}^D I_{i,j}(t+1) = \begin{cases} \leq 1, & \text{if } i \neq D; \\ 0, & \text{if } i = D, \end{cases} \quad (9)$$

where $I_{i,j}(t+1) \in \{0, 1\}$, $\forall i, j$ and $R_{i,j}(t)$ is the estimation learned about the least cost to reach destination node from node i to node j . The value of $R_{i,j}(t)$ is 0 since $H(y \setminus x) = 0$, $\forall i, j$. The constraint Eq. 9 ensures that the computed result is indeed a path (without loops) between a source and a designated destination¹⁰.

4.2. PSO Approach

PSO algorithm is a relatively new optimization algorithm which inspired by social behavior of colony of animals in environment²⁸. Like GA, the PSO is a population-based optimization method that searches multiple solutions^{29,30,25}. However, PSO employs a different strategy unlike GA, which utilizes a competitive strategy. The search for an optimal position (solution) is performed by updating the particle velocities in each iteration/generation according to their fitness values, estimated by using the fitness function. If the fitness value does not provide the optimum solution, the position of the following properties of an individual particle k is updated in the following way in each iteration. In d -dimensional search space, current position of particle k is denoted by \vec{x}_k , the corresponding objective function value by f_k , the current velocity of particle k by \vec{v}_k , and the personal best position and global best position in search space by \vec{x}_k^{best} and \vec{x}^{gbest} , respectively. The personal best position, \vec{x}_k^{best} , corresponds to the position in search space where particle k had the smallest value f_k^{best} as determined by the objective function. Meanwhile, the global best position \vec{x}^{gbest} represents the position yielding the current global best objective function value f^{gbest} amongst the entire particles' best positions f_k^{best} . Eqs. 10 and 11 define how the velocities and locations of swarm particles are updated at iteration g , respectively. As a summary, the PSO²⁸ performs the following steps:

- (1) Initialize the number of particles k_p , generation index g , constants c_1, c_2 .
- (2) Randomly initialize particle positions $\vec{x}_k = x_{k1}, x_{k2}, \dots, x_{kd}$.
- (3) Randomly initialize particle velocities $\vec{v}_k = v_{k1}, v_{k2}, \dots, v_{kd}$, $0 \leq v_{ko} \leq v_{max}$ for $o = 1, 2, \dots, n$ where v_{max} is a constant.
- (4) Evaluate fitness values f_k :

$$\begin{aligned} &\text{if } f_k < f_k^{best} \text{ then } f_k^{best} = f_k, \vec{x}_k^{best} = \vec{x}_k \\ &\text{if } f_k < f^{gbest} \text{ then } f^{gbest} = f_k, \vec{x}^{gbest} = \vec{x}_k. \end{aligned}$$
- (5) Update all particle velocities \vec{v}_k using Eq. 10.
- (6) Update all particle positions \vec{x}_k using Eq. 11.
- (7) Set $g \leftarrow g + 1$, check termination criterion. If termination criterion satisfied stop and produce results else return to step 4.

PSO explores a d dimensional space, using a population of particles which are initially provided with random velocities and positions in the problem space. Each particle represents a candidate solution to the problem. The particles change their positions by moving around the search space until a relatively unchanging arrangement has been encountered, or the stop criterion is satisfied. Each particle changes its position according to its best velocity and all particles' best velocity or global best velocity. In this case, each particle's position will change to attain its own best result according to its best velocity and attempt to find the global optimum according to the best velocity of particles thus

$$\vec{v}_k \leftarrow \omega \vec{v}_k + c_1 u_1 (\vec{x}_k^{best} - \vec{x}_k) + c_2 u_2 (\vec{x}^{gbest} - \vec{x}_k) \quad (10)$$

$$\vec{x}_k \leftarrow \vec{x}_k + \vec{v}_k \quad (11)$$

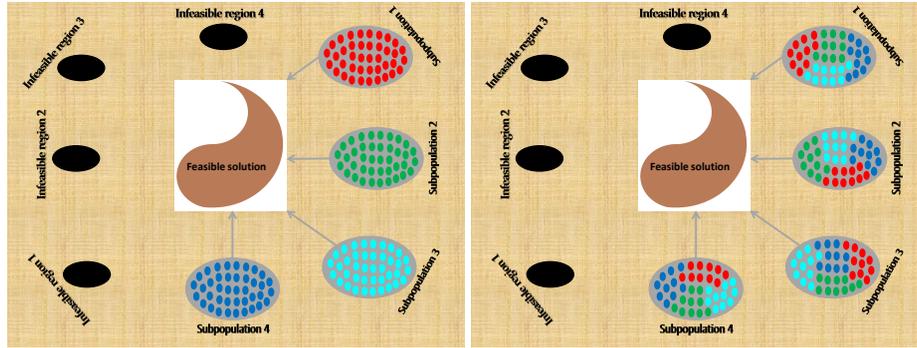


Fig. 2. Schematic diagram of different sub-population interaction strategy to approach to the feasible solution.

$\{u_1, u_2\} \in [0, 1]$ are uniformly distributed random variables and c_1, c_2 are two learning factors that controls the influence of personal best and global best in the search process. In PSO, both c_1 and c_2 are chosen between $[0, 4.0]$ to deliver good results²⁹. Inertia weight w is selected randomly between $[0, 1]$ in each generation.

5. The Proposed Approach RUBHEA for Finding Shortest-path

In this section, we have discussed our proposed approach RUBHEA for finding shortest-path from a continuously changing network model.

5.1. Algorithm of RUBHEA

RUBHEA consists of GA and PSO to solve DSPRP more efficiently and rapidly in a continuously changing network topology. It learns the estimations by reducing the uncertainty in the problem and in the algorithm itself, iteratively. Each node has estimation of the shortest path length (cost) to the destination node which should be learned and updated, and also the node learns the estimated cost of its edges to the neighbors. RUBHEA is employed by letting the operators of GA and PSO run simultaneously until an optimal fitness value is estimated or a previously set iteration limit is reached. During each iteration, RUBHEA shares information by each informing the other of their respective optimum values in their own sub-populations and consequently uncertainty is reduced. In the Fig. 2 for the first iteration the estimated shortest path lengths (costs) in the subpopulation 1, subpopulation 2, subpopulation 3, and subpopulation 4 are symbolized as \bullet , \bullet , \bullet , and \bullet , respectively. For the second iteration the estimated shortest path lengths (costs) in the subpopulation 1, subpopulation 2, subpopulation 3, and subpopulation 4 are obtained by sharing estimated \bullet , \bullet , \bullet , and \bullet to get a feasible solution.

To yield interaction between GA and PSO, a certain number of particles or chromosomes are chosen according to their fitness values. The chosen fitter individuals replace with the weaker ones in the population pool which contains both GA chromosomes and PSO particles, and the population size is kept constant. The sort of information that will

be shared depends on the current best optimum fitness value of all candidates. Assume that a chromosome reaches the best fitness value compared with all other chromosomes and particles. This shows that GA is closer to the global optimum than PSO and particles start to search around the current optimum result. As a result strong chromosomes are sent into the population of PSO and replaced with weak particles. And there will be a new population which is a combination of particles and strong chromosomes and they will be merged by using PSO operators to obtain better result in the next iteration. Henceforth, reducing uncertainty in two efficient meta-heuristic algorithms will make RUBHEA more efficient in resolving the DSPRP. To perform the RUBHEA, fittest members of populations from different meta-heuristic algorithms are evaluated and the populations with less fit members are modified by injecting fittest members of other populations resulted from different meta-heuristic algorithms. Meanwhile, less fit members of updated populations are discarded. Therefore, the following constraints between GA and PSO must be satisfied:

$$\rho^{(g+1)} = \rho^g - \gamma^g + \rho^g \times \eta = \rho^g(1 + \eta) - \gamma^g \quad (12)$$

with $\rho^g \times \eta \leq \gamma^g$ where $\eta \in [0, 1]$ as well as ρ^g and γ^g denote the sub-population size and the number of discarded candidates at iteration g , respectively.

In this paper, we have assumed that the number of discarded candidates is equal to number of join candidates. The intersection percentage of candidates η has been applied to keep the population size constant for each meta-heuristic algorithm and the candidates in each algorithm will be merging to approach to the global optimum effectively. Step-by-step procedure for RUBHEA has been stated in Algorithm 1.

5.2. Time complexity of RUBHEA

Typically, the computational complexity analysis is involved in the complexity of the problem over all possible instances, sizes, and algorithms. The number of computations required for a complete run of the PSO are the sum of the computations required to calculate the cost of a candidate solution based on Eq. 7 and the computations required to update each particle's \vec{x}_k and \vec{v}_k based on Eqs. 10 and 11. These estimations are directly proportional to g . Besides this, the execution time of the GA depends on g . Normally, the algorithm should stop executing when it is convergent to a solution. However, it is a challenging problem to give an answer whether the solution is global optimum. Therefore, in many cases, g is decided experimentally^{10,29}. In the GA, the computation time cost in an iteration depends on the evolutionary operations e.g., applying mutation and crossover operators, calculate fitness values of chromosomes, generate new solutions, etc. However, there is scarcely any guarantee that the GA and PSO will ever find a good enough solution. Consequently, it is extremely hard to discuss about their time complexities, so it is not taken into account the time complexities of the GA and PSO since they are based on search space algorithms. Instead, we have compared the execution time of the proposed method RUBHEA with the other existing algorithms. Moreover, it is analysed to understand the quality of the results in comparison to the benchmark methods, and about their convergence rate.

Algorithm 1 RUBHEA($G(t), \eta$). Algorithm of RUBHEA for solving dynamic shortest-path routing problem from a given network.

Input: $G(t) = (N(t), E(t), W(t)), \eta$ where $N(t) \Rightarrow$ set of time varying nodes in a given network model $G(t)$, $E(t) \Rightarrow$ set of time varying links in $G(t)$, and $W(t) \Rightarrow$ set of time varying links' weight values in $G(t)$.

Output: Best fitness value and shortest-path.

- 1: Initialize $k_p, k_c, p_c, p_m, \vec{x}_k, \vec{v}_k, \omega, u_1, u_2, c_1, c_2, g, d = n$.
 - 2: Set the source and destination node IDs .
 - 3: Randomly initialize the populations of particles and chromosomes.
 - 4: Use local search based encoding method on chromosomes and particles to represent valid paths.
 - 5: Evaluate the fitness values of all individuals by applying Eq. 7 .
 - 6: **if** the fittest value of all candidates is estimated from a particle **then**
 - 7: Apply Eqs. 7 and 12 for selecting the fitter particles and the weaker chromosomes.
 - 8: Reduce Uncertainty in sub-population of GA.
 - 9: **end if**
 - 10: **if** the fittest value of all candidates is estimated from a chromosome **then**
 - 11: Apply Eqs. 7 and 12 for selecting the fitter chromosomes and the weaker particles.
 - 12: Reduce Uncertainty in sub-population of PSO.
 - 13: **end if**
 - 14: Apply operators of GA and PSO for each chromosome and particle, respectively.
 - 15: **if** the optimal value is not estimated or a previously set iteration limit is not reached **then**
 - 16: Repeat from step 4.
 - 17: **end if**
 - 18: **return**
-

Fig. 3 depicts the comparison of the average execution time of the algorithms required to achieve the results as shown in Fig. 11, where the population size in the algorithms is 20. The implemented algorithms have been used to obtain the shortest-path in the given six different network models. For less than 80 nodes, the average execution time required for PSO proposed by Mohemmed et al.²⁹ is less than the other shortest-path algorithms. For the greater network models, the proposed method RUBHEA with $\eta = 0.05$ provides the best average execution time compared to the other algorithms and it takes average of 1.113, 1.183, 3.653 and 6.653 seconds for execution to find the shortest-path in 80, 100, 300, and 500 nodes, respectively.

We have calculated the speedup⁴⁷ of various algorithms with respect to the genetic algorithmic approach of Ahn et al.¹⁰, which solves the shortest path routing problem. Speedup has been defined as the ratio of the execution time of the algorithm of Ahn et al.¹⁰ and the execution time of the any other algorithm. We have chosen GA because it is a well established algorithm with many versions and applications. Fig. 4 illustrates the speedup of various algorithms with respect to the algorithm of Ahn et al.¹⁰ for six diverse network

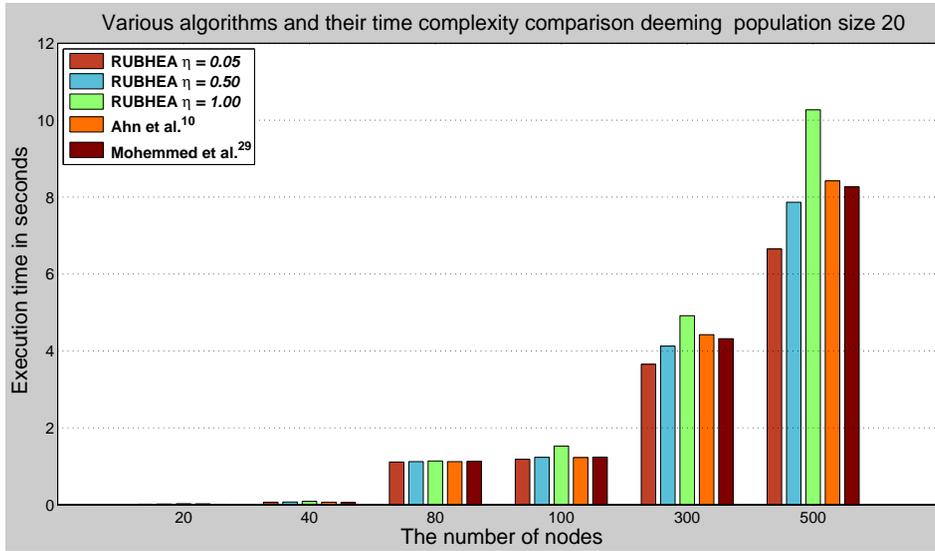


Fig. 3. Comparison of the execution time with various algorithms for six different network models.

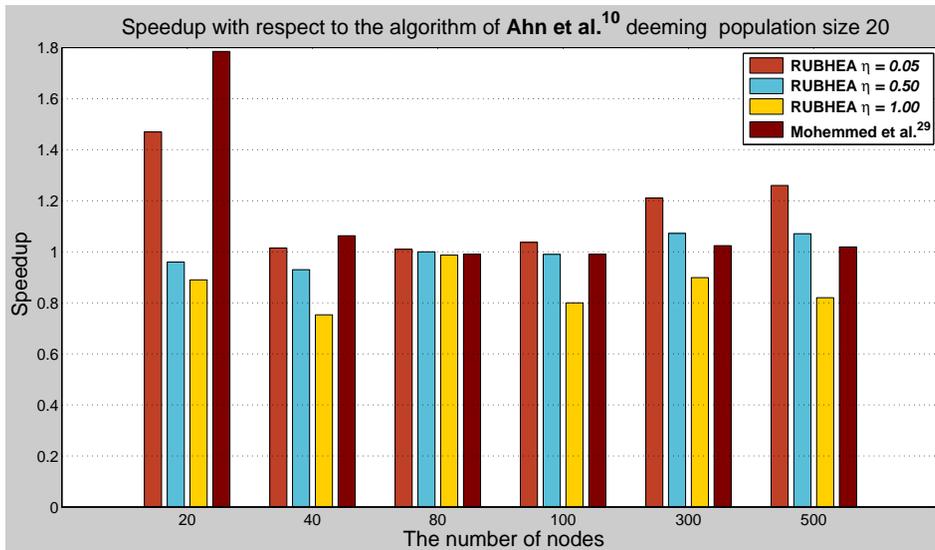


Fig. 4. Speedup with respect to the algorithm of Ahn et al.¹⁰.

models taking into account the population size 20. It is clear that speedup of RUBHEA with $\eta = 0.05$ algorithm is better than that of Mohammed et al.²⁹ if the number of nodes exceed 80. In addition, in section 6 (Fig. 12) we will see that from the experimental point of view the smallest number of iterations can be obtained by RUBHEA with $\eta = 0.05$.

5.3. Uncertainty reduction in RUBHEA

As it is known, the strong candidates have more chance than the weak candidates to reach to the global optimum according to the estimated cost values (fitness values)⁴⁸. Zhang et al.⁴⁶ discussed adaptively changing the probabilities of mutation and crossover in each generation based on the probability distribution of population. Let $P(c_{k_1}^g)$ and $P(p_{k_2}^g)$ be the probability of occurrence of chromosome $k_1 \in \{1, 2, 3, \dots, k_c\}$ and particle $k_2 \in \{1, 2, 3, \dots, k_p\}$ at generation g and $C(c_{k_1}^g)$ and $C(p_{k_2}^g)$ be the corresponding cost values (fitness values) in GA and PSO, respectively. The mean and standard deviation of cost values of populations denoted as μ_1^g and σ_1^g for GA as well as μ_2^g and σ_2^g for PSO are defined as follows:

$$\mu_1^g = \sum_{i=1}^{k_c} P(c_i^g) \lim_{\lambda \rightarrow m^+} (C(c_i^g) - \lambda) \quad \mu_2^g = \sum_{j=1}^{k_p} P(p_j^g) \lim_{\lambda \rightarrow m^+} (C(p_j^g) - \lambda) \quad (13)$$

$$\sigma_1^g = \sqrt{\sum_{i=1}^{k_c} P(c_i^g) [\mu_1^g - C(c_i^g)]^2} \quad \sigma_2^g = \sqrt{\sum_{j=1}^{k_p} P(p_j^g) [\mu_2^g - C(p_j^g)]^2} \quad (14)$$

where m^+ is defined as the convergent value. In RUBHEA, uncertainty reduction strategy is depending on the best fitness value of overall candidates in GA and PSO. If the best fitness value is found from a chromosomes, then a number of selected $\rho^g \times \eta$ chromosomes will replace with selected weak particles into the population of PSO. On the other hand, if the best fitness value is estimated from a particle then, a number of selected $\rho^g \times \eta$ particles will replace with the selected weak chromosomes into the population of GA.

Let $C(p_{k_3}^g)$ and $C(c_{k_3}^g)$ be the cost values of selected fitter particles and weaker chromosomes at generation g , $k_3 \in \{1, 2, \dots, n_3 = \rho^g \times \eta\}$, respectively. After this interaction, the μ_3^g and σ_3^g will be formulated in RUBHEA as follows:

$$\mu_3^g = \frac{1}{n_3} \left(k_c \mu_1^g - \sum_{i=1}^{k_c - \rho^g \eta} C(p_i^g) + k_p \mu_2^g - \sum_{j=1}^{\rho^g \eta} C(c_j^g) \right), \quad (15)$$

$$\sigma_3^g = \sqrt{\frac{1}{n_3} \sum_{i=1}^{n_3} (\mu_3^g - C(p_i^g))^2 + (\mu_3^g - C(c_i^g))^2}.$$

Generally, the interaction of n independent population data set $\epsilon = \epsilon_1, \epsilon_1, \dots, \epsilon_n$ will be formulated as follows:

$$\mu_\epsilon^g = \frac{\sum_i n_i \mu_i}{\sum_i n_i} \quad \sigma_\epsilon^g = \sqrt{\frac{\sum_i n_i (\sigma_i^2 + \mu_i^2)}{\sum_i n_i} - \mu_\epsilon^{g2}} \quad \epsilon_i \cap \epsilon_j = \emptyset \quad \forall i \neq j. \quad (16)$$

The means and standard deviations of all sub-populations are estimated in each generation/iteration and they will be used to estimate the variation of new population which is consisted of chromosomes and particles. By estimating μ_3 and σ_3 in RUBHEA, the probability of reaching to the global optimum will be increased because the σ_3 will be decreased and estimated less than both σ_1 and σ_2 . The aim which is to increase the probability of

candidates to find global optima or DSP routing in short time will be achieved after intersecting the sub-populations in two effective algorithms.

With regard to the development of an effective algorithm for solving search space problems, there are two possibilities that render the proposed approach remarkable and significant. One is to share information (priorities of node *ID*s) between two effective algorithms and the second is to apply the operators of algorithms to the mixed individuals information. The strategy of our proposed method and performance and influence of GA operators in RUBHEA are illustrated in an example^{10,29} of the undirected graph shown in Fig. 5 which has 20 nodes and 48 edges. The optimal paths shown in Fig. 5 with yellow and pink solid lines are estimated as optimal paths of GA (path \Rightarrow A-E-D-K-J-S-T and its total estimated cost $\Rightarrow 30+90+15+32+110+72 = 349$) and PSO (path \Rightarrow A-B-F-G-H-M-R-T and estimated cost $\Rightarrow 50+62+25+32+88+54+16 = 327$) by using local search based priority encoding method individually. Illustration of this method for a chromosome and particle and their decoded paths obtained from priority numbers of node *ID*s are shown in Figs. 6 and 7. According to the decoded paths, the PSO (with total path costs of 327) finds a better result or fitness value compared to GA (with total path costs of 349), and thus PSO shares its information with GA. Thereafter, the operators of GA apply to the mixed individuals that GA employs to find as well as or even better result than PSO. For instance, Fig. 6 shows the two cut-points crossover operator as it applies to the mixed individuals to produce the offspring to reach the global optima and Fig. 7 depicts the use of mutation operator on the shared information as the priority of node ID increases or decreases. From the produced offspring, the RUBHEA straightforwardly finds the optimal DSP as shown in Fig. 5 indicated by bold green line with optimal path A-C-H-N-T and its estimated cost is $65+35+20+22=142$. It is worth mentioning that the existing algorithms of Ahn et al.¹⁰ as well as Mohemmed et al.²⁹ also find feasible results similar to ours for this network model, but our new approach outperforms them in larger network models.

5.4. Encoding method for DSPRP

5.4.1. Population initialization

The population consisting of particles and chromosomes should be initialized depending on the problem. Each chromosome and particle is composed of a string of genes and location of positions thus representing a potential solution to the problem. The efficiency and complexity of a search algorithm greatly depends on the representation scheme. Classical optimization techniques use mathematical representations and different EAs use different representation schemes. For instance, the binary alphabet $\{0, 1\}$ is the most straightforward technique to represent the genes and positions of particles but it will be difficult to attain the global optimum by using this method^{48,49}. In the proposed RUBHEA, restricted ranges of real numbers are used to initialize the genes and positions where each one is randomly chosen between the real number (0, 100) and represents the priority number of the Node ID. In fact, almost any representation can be used that enables a solution to be encoded as a finite length string. Once a suitable representation has been decided, it is necessary to

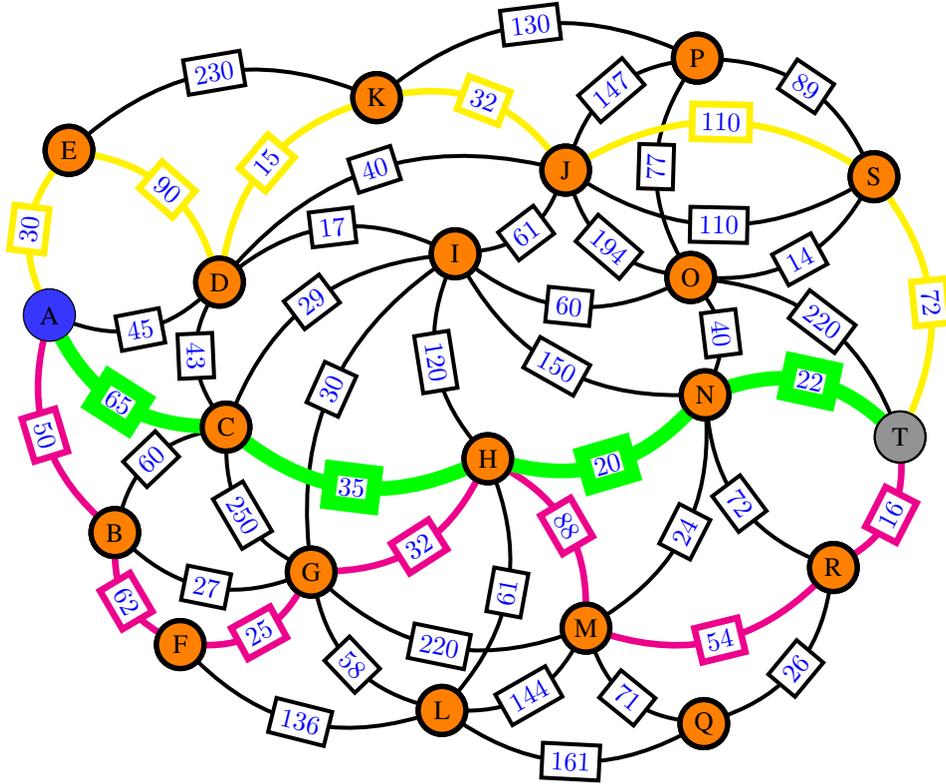


Fig. 5. An example of undirected graph for 20 nodes and 48 edges with optimal path A-C-H-N-T (its cost $\Rightarrow 65+35+20+22=142$) showed in bold green line obtained by Ahn et al.¹⁰, Moheemmed et al.²⁹, and the proposed RUBHEA with $\eta = 0.05$. The results of the GA (path \Rightarrow A-E-D-K-J-S-T and its total cost $\Rightarrow 30+90+15+32+110+72 = 349$) and the PSO (path \Rightarrow A-B-F-G-H-M-R-T and its total cost $\Rightarrow 50+62+25+32+88+54+16 = 327$) are indicated by yellow and pink solid lines, respectively.

create an initial population to serve as the starting point for the GA. This initial population is usually created randomly. From empirical studies, over a wide range of function optimization problems, a population size of between 20 and 100 is usually recommended^{10,29}. On the other hand, the processing time and memory requirement in a PC is increased when the larger population size is used to optimize a problem.

5.4.2. Local search based priority encoding method

Representing a valid path is a critical and typical complexity problem because of the many potential intermediate nodes that can be selected as a node ID in a solution path. If the chromosome and particle decoding strategy is not well employed, the path obtained may be invalid and thus not provide any benefit for solving the problem as it cannot be decoded to a solution. Priority-based encoding is a very effective method in representing a path,

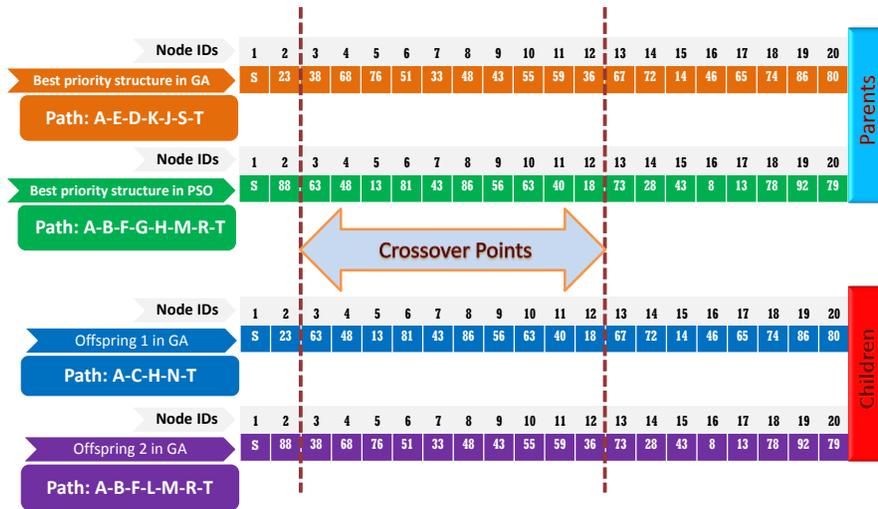


Fig. 6. Performance of GA operators in RUBHEA: Application of two-cut crossover method to the two optimal paths obtained in the example of undirected graph in Fig. 5.

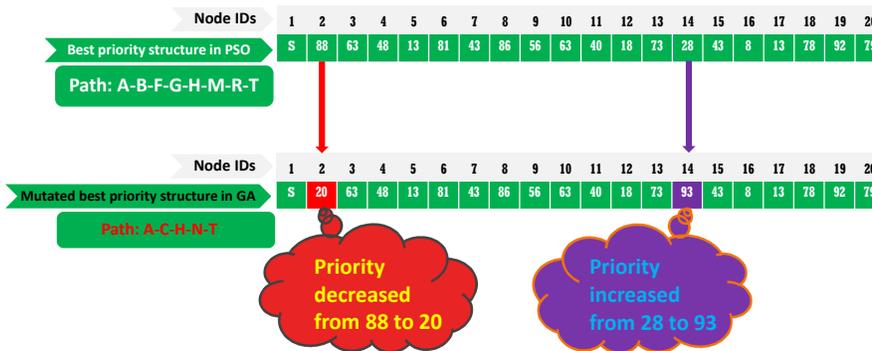


Fig. 7. Performance of GA operators in RUBHEA: Application of two-cut crossover method to the two optimal paths obtained in the example of undirected graph in Fig. 5.

first discussed by Gen et al.⁵⁰ in the context of solving the SP routing problem. The path is generated by a sequential node appending procedure beginning with the source node and terminating at the destination node. At each step of the path construction procedure, the node with the highest priority is added into the path and this process is repeated until the destination node is reached. To make an effective decoding scheme, a dynamic node adja-

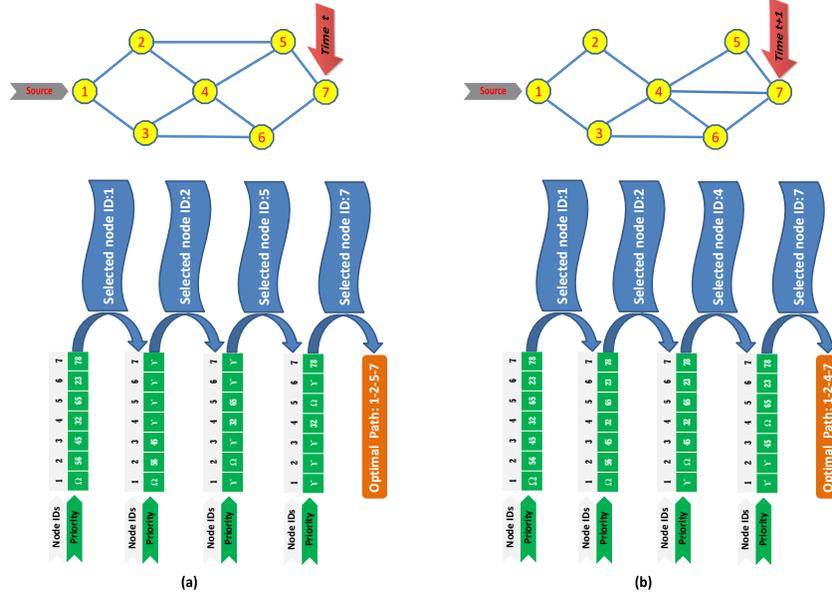


Fig. 8. Two examples of 7 nodes with 10 edges undirected graphs with path representation according to the modified search based priority encoding method. (a) Undirected graph at time t ; (b) Undirected graph at time $t+1$.

ency matrix is maintained in the implementation. This matrix is updated after selection of a node to be included in a path so that the selected node will not be a candidate for future selection. For many combinatorial optimization problems, problem specific encodings are used and these usually yield illegal offspring by a simple one-cut-point crossover operation. Repairing techniques are usually adopted to convert an illegal chromosome to a legal one. Inspired by the use of previous priority based method improvements for DSPRP²⁹, we modified priority based encoding by using a local search method. Characterizations of chromosomes and particles have been used to represent valid paths. A gene in a chromosome is characterized by two factors: the position of the gene within the structure of chromosome *locus*, i.e., the position of the gene located within the structure of chromosome, and *allele*, i.e., the value the gene takes. In RUBHEA encoding method, the position of a gene is used to represent path among candidate and a path can be uniquely determined from this encoding. Furthermore, a particle in the search space is also characterized by two factors: its *position*, i.e., represents the Node ID in a structure of particle, and the *velocity*, i.e., represents the distance to be travelled by this particle from its current position.

The symbol Ω in Fig. 8 indicates current node. Fig. 8 (a) shows an example undirected connection network to illustrate the modified search with priority based method and its decoded paths. Initially, the global priority is randomly generated and then an attempt is made to find a connected node ID to the source node 1. Nodes 2 and 3 are available for the next possible node and the choice between them can be easily determined according to the node adjacency relations. Node 2 has a priority of 56 compared to a value of 45 for node

3 and so is entered into the path. The possible nodes next to node 2 are nodes 1, 4, and 5 but the first must be removed and its priority set to any negative number Υ because it is the source and thus already in the path. Thus, node 5 has the largest priority value and is placed into the path. The formation of the set of nodes available for the next position results in the selection of node 7 to deliver the valid path (1-2-5-7). Fig. 8 (b) shows the modified search with priority based method employed in the undirected connection network where the nodes changed their position in the environment.

6. Results and Discussions

To benchmark the performance of RUBHEA, we compared it with GA-based and PSO-based dynamic shortest-path routing algorithms as implemented by Ahn et al.¹⁰, and Mohammed et al.²⁹, respectively. In addition, we investigated the performance of RUBHEA with various population intersection percentages; this is further discussed when comparisons are made with recently published hybrid algorithms^{12,6}. Following the established practice in the literature^{10,24,29}, randomly generated networks with different number of nodes, varying numbers of edges and randomly assigned link costs were used to assess the performance of the algorithms in finding the DSP in continuously changing network models. In addition, non-parametric tests have been performed to statistically show that our approach is better than its competitors proposed by Ahn et al.¹⁰, Juang¹², Marinakis et al.⁶, and Mohammed et al.²⁹.

6.1. Multiple comparisons without statistical tests

In this subsection, we have showed the impact of the individuals of GA and PSO in RUBHEA followed by the performance comparison of RUBHEA for random network topologies based on both GA (e.g., Ahn et al.¹⁰) and PSO (e.g., Mohammed et al.²⁹) as well as hybrid of GA and PSO algorithms (e.g., Juang (HGAPSO)¹² and Marinakis et al. (Hyb-GENPSO)⁶).

6.1.1. Used parameters

In the GA implementation, two point uniform crossover was employed with $p_c = 0.8$ and $p_m = 0.01$ to evolve the initial population. The parent selection was via the roulette wheel selection method and the fitness values were ascertained in each of the GA iterations. In the PSO implementation, the particle positions and velocities were initialized with random real numbers and the learning factors c_1 and c_2 were set to 2. To create dynamically changing network models, the network simulator Sinalgo⁵¹ was used. We randomly generated six different network models as case studies each having a different number of nodes (I: 20 nodes; II: 40 nodes; III: 80 nodes; IV: 100; V: 300 nodes; VI: 500 nodes). In addition, the number of edges is changing in dynamically changing network models at each time or step number t . We used two different population sizes (20 and 40) when running the algorithms, which were applied to 500 independent runs for case studies I, II, III, IV and 150 independent runs for case studies V, VI. Moreover, two different stopping criteria were

used, one being the achievement of the optimal fitness value and the other the reaching of a previously set iteration limit.

6.1.2. *Impact of the individuals of GA and PSO in RUBHEA*

In RUBHEA, strong PSO particles replaced weak chromosomes in the population of the GA or vice versa with the decision iteratively based on the sub-population fitness values evaluated until the algorithm terminated. Fig. 9 shows the impact of the GA individuals and PSO particles in RUBHEA with $\eta = 0.05$ in the achievement of the best fitness value. The lowest population intersection percentage was chosen to analyze and understand the impact of the populations in RUBHEA because this provided the best results with the least distortion of the populations. From the figure, PSO substantially outperforms the GA in small network models but advantage is modest for very large networks. For instance, when the network contains 20 nodes, the particles in PSO are reaching the global optimum 68% of the time compared to just 32% with the GA chromosomes, i.e., PSO performs $68/32 \approx 2.1$ times better. This kind of performance improvement of PSO also confirms the claims of Mohammed et al.²⁹ as well as Marinakis et al.⁶. Mohammed et al.²⁹ where a comparison of PSO-based search algorithm and a GA for solving the DSPRP was proposed and the authors claimed that the PSO-based algorithm was superior to that using GA for finding DSP in dynamically changed networks. Marinakis et al.⁶ proposed a hybrid of GA and PSO for the vehicle routing problem and claimed that by the effect of PSO the use of an intermediate phase between two generations gave more efficient individuals and would improve the effectiveness of their HybGENPSO algorithm. However, it should be borne in mind that the common test networks such as the Chinese National Network (CHNNET) and ARPANET contain relatively small numbers of nodes (15 and 20, respectively)⁵². Henceforth, RUBHEA will deliver substantial benefits for practical network design based on the superior PSO performance and will still perform well for very large networks.

6.1.3. *Route failure ratio performance for RUBHEA and others*

The route optimality was considered since this is the most common and widely-used definition for investigating the performance of algorithms. The definition of route optimality (or success rate) is the mean percentage of times that the algorithm finds the global optimum (shortest-path) over a large number of runs. Here, use is made of the route failure ratio, which is the fraction of the runs which result in non-optimal computed routes. Since the performance of RUBHEA will vary with the population intersection value η , three different values of η were employed, i.e., $\eta = \{0.05, 0.50, 1.00\}$. Fig. 10 illustrates the route failure ratio performance for various algorithms with different network models. The algorithm of Mohammed et al.²⁹ performed the best with population size 20 and 20 nodes. A network model having 100 nodes and a population size of 20, RUBHEA with $\eta = 0.05$ delivered a route failure ratio of just 0.09 (implying a route optimality of 91%) and the algorithm of Mohammed et al.²⁹ produced 0.17 (i.e., a route optimality of 83%) which was close to RUBHEA with $\eta = 0.50$. On the other hand, RUBHEA with $\eta = 1.00$ and

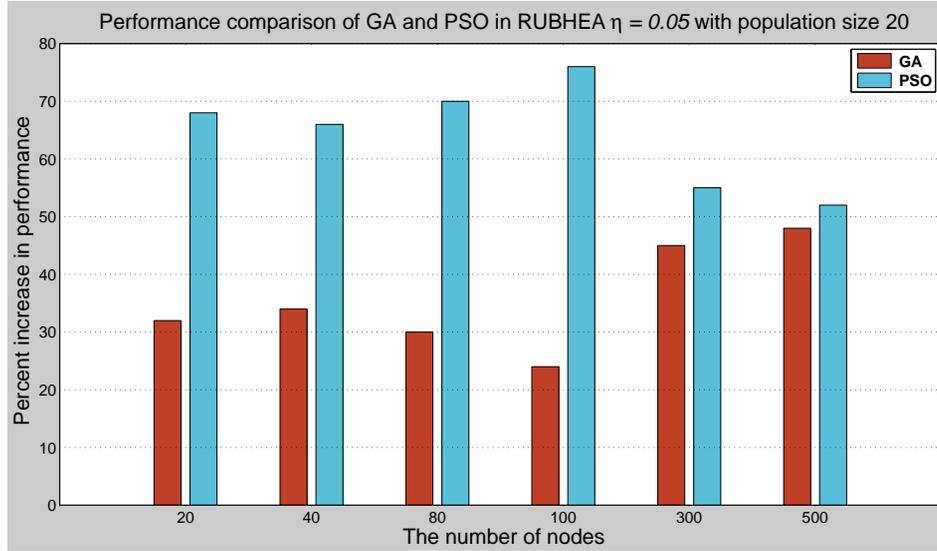


Fig. 9. Impact of the individuals of GA and PSO in RUBHEA $\eta = 5\%$ to achieve the best fitness value.

Ahn et al.¹⁰ resulted in route failure ratios of more than 0.30 (i.e., a route optimality up to 70%). With this simple statistic, RUBHEA with $\eta = 0.05$ is reducing the failure rate by up to $1 - 0.09/0.17 = 0.47$ or 47% as compared to the algorithm of Mohammed et al.²⁹ or RUBHEA with $\eta = 0.50$ as well as up to $1 - 0.09/0.30 = 0.70$ or 70% as compared to the algorithm of Ahn et al.¹⁰ or RUBHEA with $\eta = 1.00$, i.e., on the average up to 58.50%.

The algorithm of Ahn et al.¹⁰ lacked behind both Mohammed et al.²⁹ and RUBHEA with $\eta = 0.50$ in employed all network models. With the increasing of population size and/or nodes RUBHEA with $\eta = 0.50$ showed the upper hand over Mohammed et al.²⁹, RUBHEA with $\eta = 0.50$, Ahn et al.¹⁰, and RUBHEA with $\eta = 1.00$; i.e., RUBHEA with $\eta = 0.05$ provided the highest performance. Intrusively speaking, RUBHEA with $\eta = 0.05$ gave the best performance and this decreased with increased population intersection percentage. Furthermore, the quality of solution increased when a higher population size was employed as shown in Fig. 10 (b) because the increase population provided more chances for evolution. For instance, determined route failure ratio of RUBHEA with $\eta = 0.05$ for the case study of VI was 0.19 with a population size of 20 but decreased to 0.13 or $1 - 0.13/0.19 = 31.6\%$ when the population size was 40.

6.1.4. Average fitness function values for RUBHEA and others

The algorithm efficiencies were investigated by considering the average fitness function values for dynamically changed network models. Fig. 11 (a) illustrates that the performance of RUBHEA with $\eta = 0.05$ was better than the other algorithms with a population size of 20. For instance, in the case of 100 nodes, RUBHEA with $\eta = 0.05$ achieved an average fitness of 143 which was by far the lowest value of the algorithms consid-

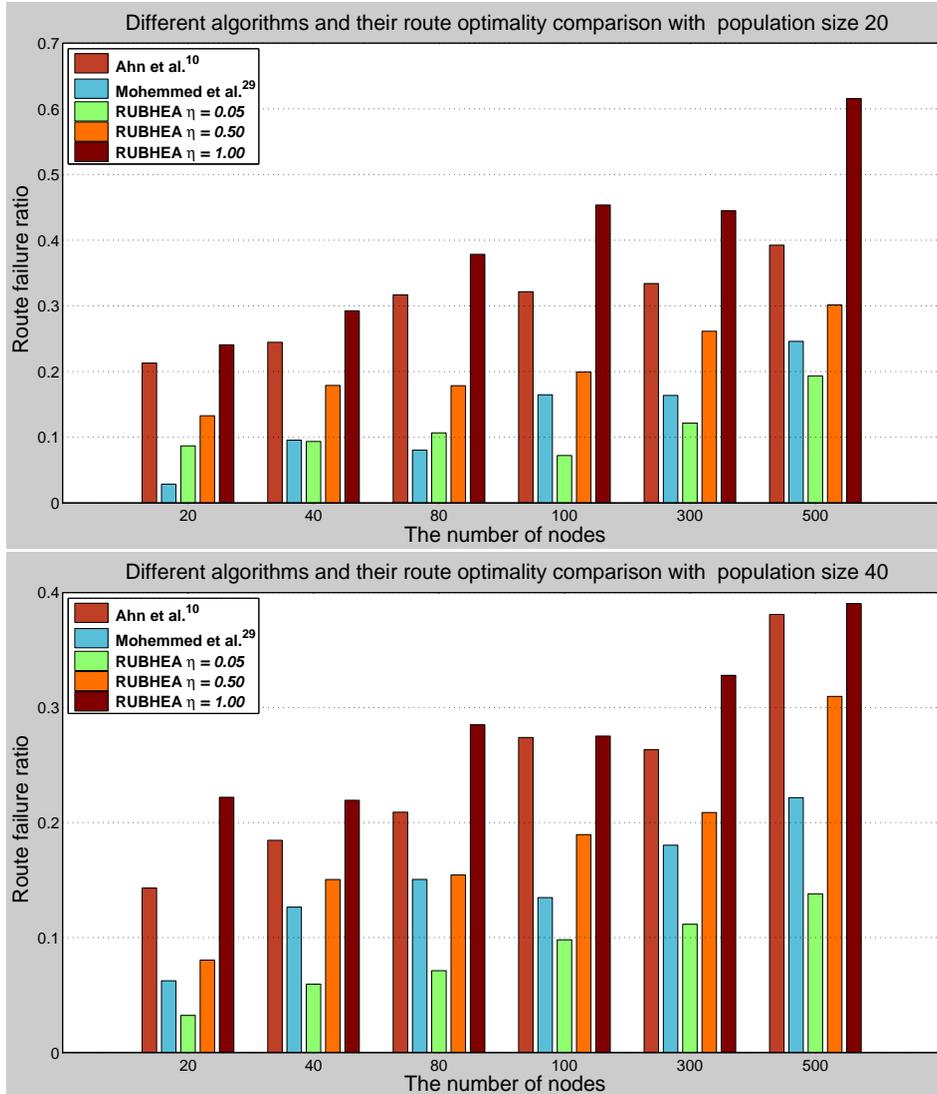


Fig. 10. Comparison of the route optimality with various algorithms for six different network models.

ered. Increased population intersection again decreased the performance but RUBHEA with $\eta = 0.50$ still delivered similar results to the algorithm of Mohammed et al.²⁹ and much better outcomes than the algorithm of Ahn et al.¹⁰. Increasing the intersection to $\eta = 1.00$ gave the worst performance because of information distortion by the process of the replacement of candidates in the populations of the algorithms. An exchange beyond a certain point will mean that relatively poor candidates will be swapped and the benefit of the process (strong replacing weak) will disappear. It is also interesting to note that there is no significant difference between the average of fitness values obtained by RUBHEA

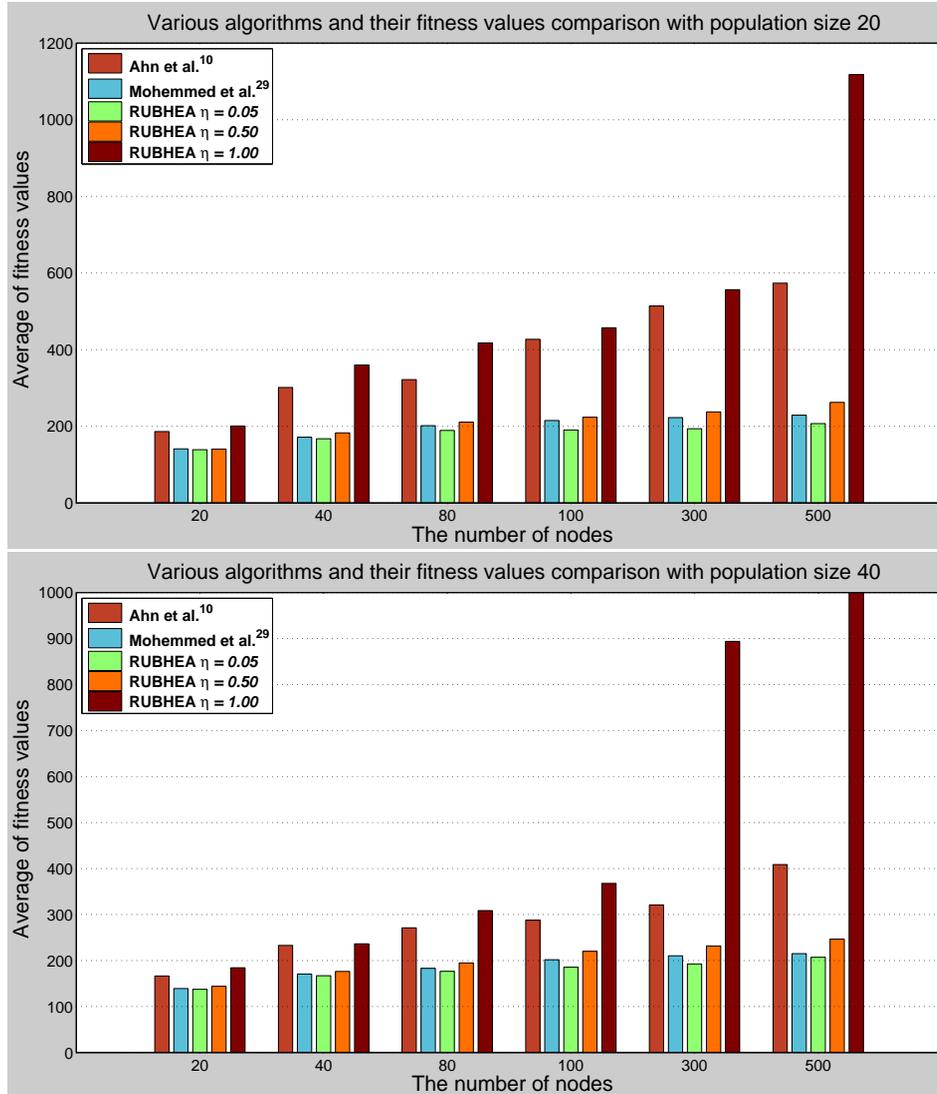


Fig. 11. Comparison of the estimated fitness values with various algorithms for six different network models.

with $\eta = 0.05$ using population sizes of 20 and 40, showing that the algorithm works well with a small population size. As a result, the fitness function in RUBHEA optimizes the dynamic shortest paths effectively as compared to other fitness functions shown in Fig 11.

Fig 12 depicts the average number of iterations required to achieve the feasible solution, showing that RUBHEA with $\eta = 0.05$ found the optimum route with the smallest number of iterations. For example, the algorithm of Mohammed et al.²⁹ required 9 iterations for 20 nodes using population size 40 and for the same the algorithm of Ahn et al.¹⁰ took some 16 iterations to converge. But for the same case RUBHEA with $\eta = 0.05$ needed only

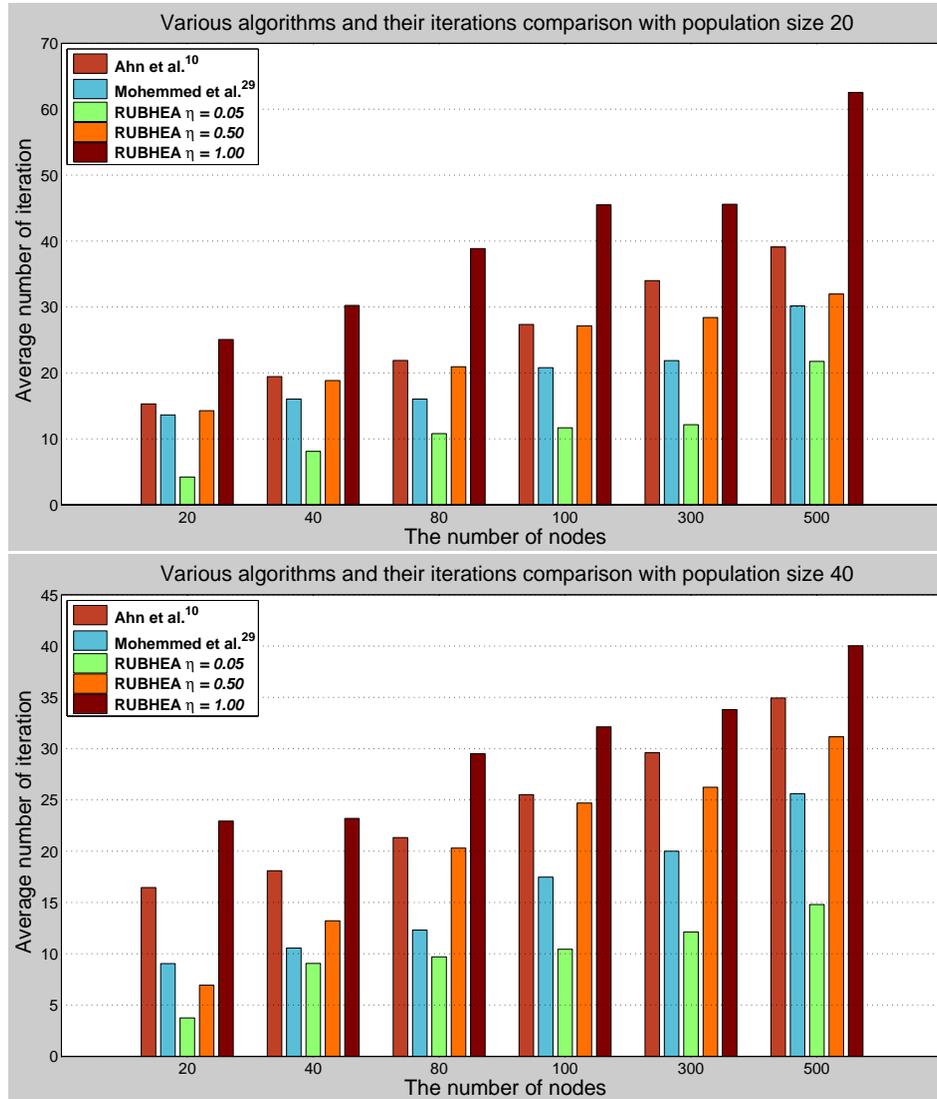


Fig. 12. Comparison of the average number of iterations with various algorithms for six different network models.

4 iterations, i.e., it has $1 - 4/9 \approx 56\%$ and $1 - 4/16 = 75\%$ better performance over Mohammed et al.²⁹ and Ahn et al.¹⁰, respectively. As one would expect, employing the algorithms with the higher population size gave better results as shown in Fig. 12 (b) but this caused a higher memory requirement in the computer.

However, without any shadow of doubt from Figs. 10, 11, and 12 one would conclude that irrespective of the population size RUBHEA with $\eta = 0.05$ solves the dynamic shortest-path routing problem (DSPRP) effectively and rapidly as compared to RUBHEA with $\eta = 1.00$, the algorithm of Ahn et al.¹⁰, RUBHEA with $\eta = 0.50$, and the algorithm

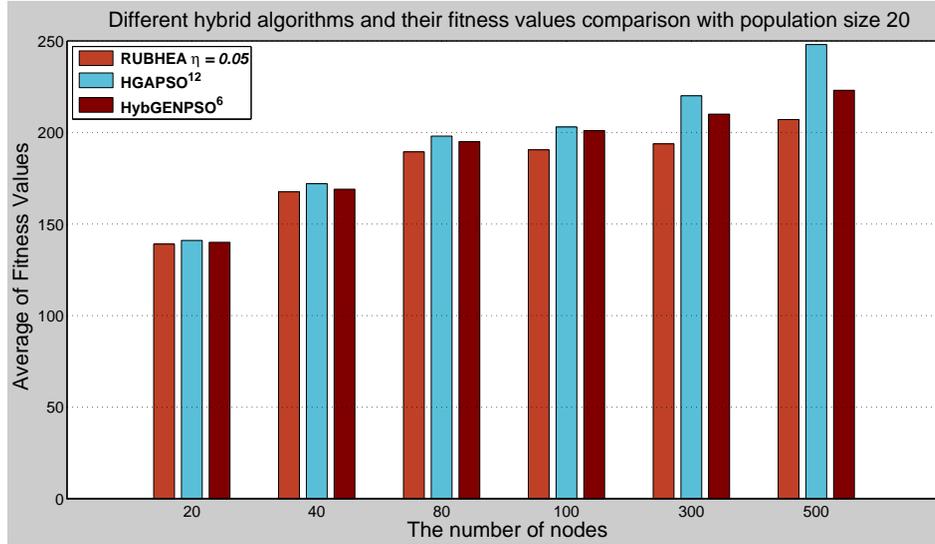


Fig. 13. Comparison of the average number of estimated fitness values for three hybrid algorithms.

of Mohammed et al.²⁹.

6.1.5. Comparing RUBHEA with other hybrid algorithms

The main conclusion from the tests described in the previous section is that the proposed method gives effective results as long as η is small. To further understand and analyze the performance and the capability of the RUBHEA approach here, it has been compared to recently used hybrid algorithms for recurrent network design (HGAPSO¹²) and vehicle routing (HybGENPSO⁶). In both the HGAPSO¹² and HybGENPSO⁶ algorithms, GA and PSO work with the same population. In the HGAPSO¹², the best 50% of individuals are evaluated and enhanced by using the operators of GA and PSO. In HybGENPSO⁶, the operators of GA and PSO are, respectively, applied to the all individuals to produce the new generation. Fig. 13 depicts the average number of estimated fitness values. The algorithm of HybGENPSO⁶ is slightly better than that of HGAPSO¹² based on finding the low average number of fitness values in all network models presented hereby. However, the proposed RUBHEA with $\eta = 0.05$ finds the lowest average number of estimated fitness values in all network models. Although the existing hybrid algorithms give feasible results similar to ours for small network models, our new approach outperforms them in larger network models. Fig. 14 shows that the algorithm of HGAPSO¹² is better than that of HybGENPSO⁶ in terms of the average number of iterations. It also demonstrates that the convergence rate of the RUBHEA with $\eta = 5\%$ is much better than HybGENPSO⁶ and slightly better than HGAPSO¹² when achieving the feasible solution.

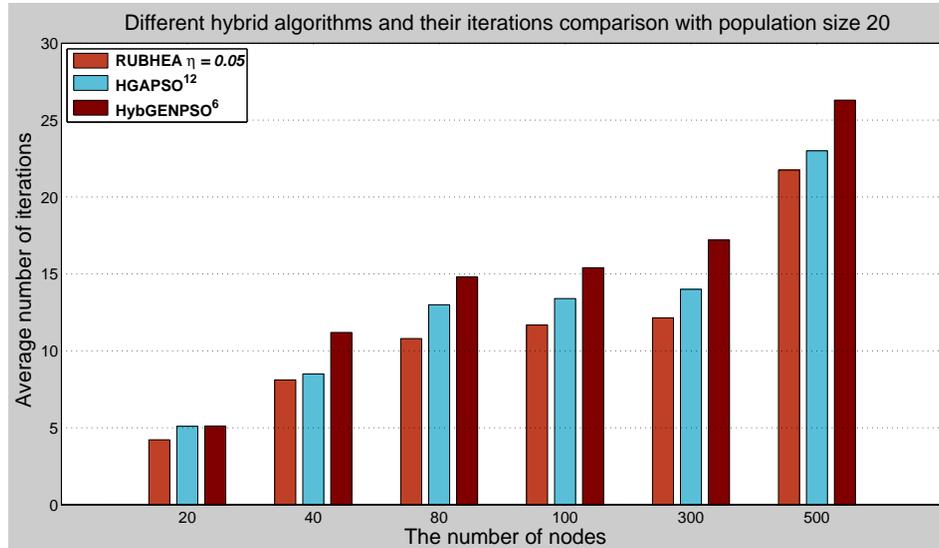


Fig. 14. Comparison of the average number of iterations to get the feasible solution for three hybrid algorithms.

6.2. Multiple comparisons with statistical tests

Multiple comparisons with a control algorithm have commonly been used to statistically show that one approach is better than its competitors in areas related to computer science. The main idea of using the non-parametric tests⁵⁴ is that they can deal with probabilistic and non-probabilistic methods without any limitation. In this subsection, non-parametric test results are presented and examined for comparing the proposed algorithms with the rest of algorithms. We have performed tests adequate to multiple comparisons together with a set of post-hoc procedures to compare a control algorithm with other algorithms ($1 \times N$ comparisons) or to perform all possible pairwise comparisons ($N \times N$ comparisons).

As we know, in conducting a test of significance or hypothesis test there are two important numbers namely p -value of the test statistic and the level of significance α . Both p -value and α could be easily confused because they are both numbers between zero and one, and are in fact probabilities. The number α tells us how extreme observed results must be in order to reject the null hypothesis of a significance test. The p -value of the test statistic is a way of saying how extreme that statistic is for our sample data. The smaller the p -value, the more unlikely the observed sample. In statistical significance testing, the p -value is the probability of obtaining a test statistic result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true⁵³. Critics of p -values point out that the criterion used to decide statistical significance is based on an arbitrary choice of level (often set at 0.05)⁵⁵. If significance testing is applied to hypotheses that are known to be false in advance, a non-significant result will simply reflect an insufficient sample size; a p -value depends only on the information obtained from a given experiment.

6.2.1. Various nonparametric tests

Friedman test⁵⁶ and its derivatives (e.g., Iman-Davenport test⁵⁷) are usually referred as one of the most important non-parametric tests for multiple comparisons. First of all, we have performed the Friedman test⁵⁶. A usable characteristics of this test is that it ranks the algorithms from the best performing to the poorest one. However, it can only inform the researcher about the presence of differences among all samples of results compared. We have also performed two more alternatives the Friedman Aligned Ranks⁵⁸ and the Quade test⁵⁹, which differ in the way of computing the rankings and may lead to better results depending on the features of the experimental study considered. After the null-hypotheses have been rejected, we have proceeded with the post-hoc procedures to find the particular pairs of algorithms which produce differences. The post-hoc procedures comprise Bonferroni-Dunn's⁶⁰, Holm's⁶¹, Hochberg's⁶², Hommel's^{63,64}, Holland's⁶⁵, Rom's⁶⁶, Finner's⁶⁷, and Li's⁶⁸, procedures in the case of $1 \times N$ comparisons, and Nemenyi's⁶⁹, Shaffer's⁷⁰, and Bergmann-Hommel's⁷¹ procedures in the case of $N \times N$ comparisons. The Bonferroni-Dunn's procedure⁶⁰ leads to the statement that the performance of two algorithms is significantly different if the corresponding average of rankings is at least as great as its critical difference. More powerful is Holms procedure⁶¹ which checks sequentially hypotheses ordered according to their p -values from the lowest to the highest. All hypotheses for which p -value is less than the significance level α divided by the number of algorithms minus the number of a successive step are rejected. All hypotheses with greater p -values are supported. Holland's⁶⁵ and Finner's⁶⁷ procedures, also adjust the value of α in a step-down manner as Holm's step down method⁶¹ does. The Hochberg's procedure⁶² operates in the opposite direction to the former, comparing the largest p -value with α , the next largest with $\alpha/2$, and so forth until it encounters a hypothesis it can reject. Rom⁶⁶ devised a modification to Hochberg's step-up procedure⁶² to increase its power. In turn, Li⁶⁸ proposed a two-step rejection procedure.

6.2.2. Tools used for nonparametric tests

Statistical analysis of the results of experiments was performed using the available software^a and the open source JAVA program calculates multiple comparison procedures: Friedman⁵⁶, Iman et al.⁵⁷, Bonferroni et al.⁶⁰, Holm⁶¹, Hochberg⁶², Holland⁶⁵, Rom⁶⁶, Finner⁶⁷, Li⁶⁸, Shaffer⁷⁰, and Bergamnn et al.⁷¹ tests as well as adjusted p -values.

When all possible pairwise comparisons need to be performed, the easiest is the Nemenyi's procedure⁶⁹. It assumes that the value of the significance level α is adjusted in a single step by dividing it merely by the number of comparisons performed. It is a very simple way but has little power. The Shaffer's static routine⁷⁰, in turn, follows the Holm's step down method⁶¹. At a given stage, it rejects a hypothesis if the p -value is less than α divided by the maximum number of hypotheses which can be true given that all previous hypotheses are false. The Bergmann et al.'s⁷¹ scheme is characterized by the best performance, but it is also the most sophisticated and therefore difficult to understand and computationally

^a<http://sci2s.ugr.es/sicidm>

expensive. It consists in finding all the possible exhaustive sets of hypotheses for a certain comparison and all elementary hypotheses which cannot be rejected. The details of the procedure are described in Bergmann et al.⁷¹, Garcia et al.⁷² and the rapid algorithm for conducting this test is presented in Hommel et al.⁶⁴.

6.2.3. Multiple comparison nonparametric tests

Table 1 depicts the average ranks computed using Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ non-parametric tests. To achieve the test results Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ non-parametric tests are applied to the average number of estimated cost fitness values. The purpose of using Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ non-parametric tests is to determine whether there are significant differences among the algorithms considered over given sets of data^{59,73}. These tests obtain the ranks of the algorithms for each individual data set, i.e., the best performing algorithm receives the rank of 1, the second best rank 2, etc. Here, we have not discussed the non-parametric methods, however, the mathematical equations and further explanation of the non-parametric procedures of Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ can be found in the literatures (e.g., Quade⁵⁹, Westfall et al.⁷³, etc.). Based on the obtained results in the Table 1, RUBHEA $\eta = 0.05$ is the best performing algorithm of the comparison, with average rank of 4.66, 16.00, and 0.99 for Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ tests, respectively. This shows that RUBHEA $\eta = 0.05$ provides great performance to find DSP in a given network model. The p -values computed through the statistics of each of the tests considered (7.399×10^{-6} , 0.525, and 1.395×10^{-8}). The Iman-Davenport⁵⁷ statistic and p -value are computed 76.29032 and 8.239×10^{-17} , respectively.

Various Approaches	Multiple Comparison Tests		
	Friedman ⁵⁶	Friedman Aligned Ranks ⁵⁸	Quade ⁵⁹
Ahn et al. ¹⁰	6.00	34.16	6.00
Juang ¹²	2.16	10.33	2.19
Mohammed et al. ²⁹	3.33	16.16	3.09
Marinakos et al. ⁶	3.83	18.16	3.81
RUBHEA $\eta = 1.00$	7.00	38.83	7.00
RUBHEA $\eta = 0.50$	4.66	26.83	4.90
RUBHEA $\eta = 0.05$	0.99	16.00	0.99
Statistics	33.785	5.14	17.45
p-value	7.399×10^{-6}	0.525	1.395×10^{-8}

Table 1. Average rankings of different algorithms using the non-parametric statistical procedures, statistics, and p -values.

6.2.4. *Post-hoc procedures for $1 \times N$ and $N \times N$ comparisons*

In these statistical analysis tests, multiple comparison post-hoc procedures considered for comparing the control algorithm RUBHEA $\eta = 0.05$ with the rest of algorithms. The results are shown by computing p -values for each comparison. Tables 2, 3, and 4 show the p -values obtained, using the ranks computed by the Friedman⁵⁶, Friedman Aligned Ranks⁵⁸, and Quade⁵⁹ non-parametric tests, respectively. Based on the computed results, all tests show significant improvements of the RUBHEA $\eta = 0.05$ over RUBHEA $\eta = 1.00$, RUBHEA $\eta = 0.50$, Marinakis et al. (HybGENPSO)⁶, and Juang (HGAPSO)¹² for all the post-hoc procedures considered. Besides this, the Li's⁶⁸ procedure performs the greatest performance, reaching the lowest p -values in the comparisons.

The post-hoc procedures comprise Bonferroni-Dunn's⁶⁰, Holm's⁶¹, Hochberg's⁶², Hommel's^{63,64}, Holland's⁶⁵, Rom's⁶⁶, Finner's⁶⁷, and Li's⁶⁸, procedures in the case of $1 \times N$ comparisons; and Nemenyi's⁶⁹, Shaffer's⁷⁰, as well as Bergmann-Hommel's⁷¹ procedures in the case of $N \times N$ comparisons.

Approaches	Unadjusted	$1 \times N$ post-hoc procedures					
		One/Two step procedures		Step-down procedures		Step-up procedures	
		Bonf. ⁶⁰	Li ⁶⁸	Holm ⁶¹ ,Hol. ⁶⁵	Finner ⁶⁷	Hoch. ⁶² ,Hom. ⁶³	Rom ⁶⁶
Ahn et al. ¹⁰	6.09×10^{-5}	3.65×10^{-4}	9.37×10^{-5}	3.04×10^{-4}	1.82×10^{-4}	3.04×10^{-4}	2.900×10^{-4}
Juang ¹²	0.3495	2.0974	0.3491	0.3495	0.3495	0.3495	0.3495
Mohammed et al. ²⁹	0.0613	0.3683	0.0462	0.1227	0.0731	0.1227	0.1227
Marinakis et al. ⁶	0.0231	0.1382	0.0343	0.0693	0.0344	0.0693	0.0693
RUBHEA $\eta = 1.00$	1.50×10^{-6}	9.02×10^{-6}	2.31×10^{-6}	9.02×10^{-6}	9.02×10^{-6}	9.02×10^{-6}	8.58×10^{-6}
RUBHEA $\eta = 0.50$	0.0032	0.0197	0.0050	0.0131	0.0065	0.0131	0.0125

Table 2. Adjusted p -values for Friedman test (RUBHEA $\eta = 0.05$ is the control method)

Approaches	Unadjusted	$1 \times N$ post-hoc procedures					
		One/Two step procedures		Step-down procedures		Step-up procedures	
		Bonf. ⁶⁰	Li ⁶⁸	Holm ⁶¹ ,Hol. ⁶⁵	Finner ⁶⁷	Hoch. ⁶² ,Hom. ⁶³	Rom ⁶⁶
Ahn et al. ¹⁰	6.98×10^{-5}	4.19×10^{-4}	1.52×10^{-4}	3.49×10^{-4}	2.09×10^{-4}	3.49×10^{-4}	3.32×10^{-4}
Juang ¹²	0.5406	3.2439	0.5401	0.5406	0.5406	0.5406	0.5406
Mohammed et al. ²⁹	0.1511	0.9070	0.1476	0.3023	0.1785	0.3023	0.3023
Marinakis et al. ⁶	0.0858	0.5150	0.1074	0.2575	0.1259	0.2575	0.2575
RUBHEA $\eta = 1.00$	3.55×10^{-6}	2.13×10^{-5}	7.74×10^{-6}	2.13×10^{-5}	2.13×10^{-5}	2.13×10^{-5}	2.03×10^{-5}
RUBHEA $\eta = 0.50$	0.0032	0.0196	0.0050	0.0130	0.0065	0.0130	0.0124

Table 3. Adjusted p -values for Friedman Aligned Ranks (RUBHEA $\eta = 0.05$ is the control method).

Approaches	Unadjusted	$1 \times N$ post-hoc procedures						
		One/Two step procedures		Step-down procedures			Step-up procedures	
		Bonf. ⁶⁰	Li ⁶⁸	Holm ⁶¹	Hol. ⁶⁵	Finner ⁶⁷	Hoch. ⁶² ,Hom. ⁶³	Rom ⁶⁶
Ahn et al. ¹⁰	0.0375	0.2250	0.0899	0.1875	0.1875	0.1083	0.1875	0.1783
Juang ¹²	0.6204	3.7224	0.6004	0.7667	0.6204	0.6204	0.6204	0.6204
Mohemmed et al. ²⁹	0.3833	2.3003	0.4024	0.7667	0.6204	0.4402	0.6204	0.6204
Marinakis et al. ⁶	0.2424	1.4548	0.3397	0.7274	0.6204	0.3406	0.6204	0.6204
RUBHEA $\eta = 1.00$	0.0125	0.0753	0.0320	0.0753	0.0753	0.0730	0.0753	0.0716
RUBHEA $\eta = 0.50$	0.1042	0.6256	0.1855	0.4170	0.4170	0.1976	0.4170	0.3977

Table 4. Adjusted p -values for Quade (RUBHEA $\eta = 0.05$ is the control method).

Table 5 presents 21 hypotheses of equality among the 6 different algorithms and the p -values achieved. Using level of significance $\alpha = 0.05$, first 4 hypotheses were rejected by the post-hoc procedures of Nemenyi⁶⁹, Holm⁶¹, Shaffer⁷⁰, and Bergmann⁷¹. For instance, let us compare RUBHEA $\eta = 0.01$ vs. RUBHEA $\eta = 1.00$ by using Nemenyi⁶⁹ and Holm⁶¹ procedures and estimated p -values of post-hoc procedures are less than $\alpha = 0.05$. Therefore, we reject the hypothesis of the RUBHEA $\eta = 0.01$ vs. RUBHEA $\eta = 1.00$. Shaffer⁷⁰ and Bergmann⁷¹ procedures reject two more hypotheses, thus confirming the improvement of Mohemmed et al.²⁹ over RUBHEA $\eta = 1.00$, RUBHEA $\eta = 0.01$ over RUBHEA $\eta = 0.50$. However, none of the remaining 15 hypotheses can be rejected using these procedures because computed p -values by using Nemenyi⁶⁹, Holm⁶¹, Shaffer⁷⁰, and Bergmann⁷¹ procedures are greater than $\alpha = 0.05$.

In summary, based on the results of aforementioned multiple comparisons without statistical tests and with statistical tests it would be easy to make an exclusive conclusion that RUBHEA with $\eta = 0.05$ outperforms over RUBHEA with $\eta = 0.05$, RUBHEA with $\eta = 1.00$, Juang (HGAPSO)¹², Marinakis et al. (HybGENPSO)⁶, and Mohemmed et al.²⁹. Intuitively speaking, it is observed that the performance of the proposed RUBHEA surpasses those of recently reported evolutionary hybrid algorithms based approaches for dynamic shortest-path routing problem.

7. Conclusion

We have presented a reduced uncertainty based hybrid evolutionary algorithm (RUBHEA) to solve DSPRP. GA and PSO were employed to find the global optimum within a set of dynamically changed network models. RUBHEA adopts the modified priority encoding method to represent paths in randomly generated undirected networks. The encoding method represented the valid paths between source and destination, thus improving efficiency in computation and decreasing memory usage. Results of RUBHEA are compared with the effective variations of state-of-art algorithms. Simulation results show that RUBHEA can provide promising results when η value is low. RUBHEA with $\eta = 0.05$ reduces the failure ratio by up to 50%. Yet a high η value causes the interaction between the

Index	Hypotheses	$N \times N$ post-hoc procedures				
		Unadjusted	Nemenyi ⁶⁹	Holm ⁶¹	Shaffer ⁷⁰	Bergmann ⁷¹
1	RUBHEA $\eta = 0.01$ vs. RUBHEA $\eta = 1.00$	1.50×10^{-5}	3.15×10^{-5}	3.15×10^{-5}	3.15×10^{-5}	3.15×10^{-5}
2	Ahn et al. ¹⁰ vs. RUBHEA $\eta = 0.01$	6.09×10^{-5}	0.0012	0.0012	9.14×10^{-4}	9.14×10^{-4}
3	RUBHEA $\eta = 1.00$ vs. Juang ¹²	1.06×10^{-4}	0.0022	0.0021	0.0015	0.0015
4	Ahn et al. ¹⁰ vs. Juang ¹²	0.0021	0.0444	0.0380	0.0317	0.0211
5	Mohammed et al. ²⁹ vs. RUBHEA $\eta = 1.00$	0.0032	0.0689	0.0558	0.0492	0.0361
6	RUBHEA $\eta = 0.01$ vs. RUBHEA $\eta = 0.50$	0.0032	0.0689	0.0558	0.0492	0.0361
7	RUBHEA $\eta = 1.00$ vs. Marinakis et al. ⁶	0.0111	0.2334	0.1667	0.1667	0.1001
8	RUBHEA $\eta = 0.01$ vs. Marinakis et al. ⁶	0.0231	0.4851	0.3234	0.2541	0.2079
9	Ahn et al. ¹⁰ vs. Mohammed et al. ²⁹	0.0325	0.6826	0.4226	0.3576	0.2275
10	RUBHEA $\eta = 0.50$ vs. Juang ¹²	0.0450	0.9454	0.5402	0.4952	0.3151
11	RUBHEA $\eta = 0.50$ vs. RUBHEA $\eta = 1.00$	0.0613	1.2887	0.6750	0.6750	0.3682
12	Mohammed et al. ²⁹ vs. RUBHEA $\eta = 0.01$	0.0613	1.2887	0.6750	0.6750	0.3682
13	Ahn et al. ¹⁰ vs. Marinakis et al. ⁶	0.0823	1.7293	0.7411	0.7411	0.3682
14	Marinakis et al. ⁶ vs. Juang ¹²	0.1814	3.8104	1.4515	1.2701	0.7257
15	Mohammed et al. ²⁹ vs. RUBHEA $\eta = 0.50$	0.2850	5.9860	1.9953	1.9953	1.4252
16	Ahn et al. ¹⁰ vs. RUBHEA $\eta = 0.50$	0.2850	5.9860	1.9953	1.9953	1.4252
17	Mohammed et al. ²⁹ vs. Juang ¹²	0.3495	7.3410	1.9953	1.9953	1.4252
18	RUBHEA $\eta = 0.01$ vs. Juang ¹²	0.3495	7.3410	1.9953	1.9953	1.4252
19	Ahn et al. ¹⁰ vs. RUBHEA $\eta = 1.00$	0.4226	8.8762	1.9953	1.9953	1.4252
20	RUBHEA $\eta = 0.50$ vs. Marinakis et al. ⁶	0.5040	10.5847	1.9953	1.9953	1.4252
21	Mohammed et al. ²⁹ vs. Marinakis et al. ⁶	0.6884	14.4584	1.9953	1.9953	1.4252

Table 5. Adjusted p -values for tests for multiple comparisons among the all methods.

two sub-populations to increase to such a degree that it causes high distortion in the sub-populations. Furthermore, it causes important information or strong candidates to be lost preventing RUBHEA from efficiently reaching the optimum value. Overall, it is apparent that combining the search abilities of GA and PSO with a population of solutions is an efficient approach to solve DSPRP. Consequently, RUBHEA can search the solution space effectively and speedily compared with other existing algorithms when η is kept low.

Acknowledgment

Authors acknowledge the anonymous reviewers for their appreciative and constructive comments on the draft of this paper.

References

1. Porter, S. and Mirmehdi, M. and Thomas, B.: A shortest path representation for video summarization, Proceedings of International Conference on Image Analysis and Processing, 2003, pp. 460–465.

2. Chen, Y.L. and Yang, H.H.: Finding the first K shortest paths in a time-window network, *Computers and Operations Research* 31, 2004, pp. 499–513.
3. Xiao, Y., Thulasiraman, K. and Xue, G.: Constrained shortest link-disjoint paths selection: a network programming based approach, *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53, 2006, pp. 1174–1187.
4. Sun, X., You, X. and Liu, S.: Multi-objective ant colony optimization algorithm for shortest route problem, *International Conference on Machine Vision and Human-Machine Interface*, 2010, pp. 796–798.
5. Davies, C. and Lingras, P.: Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. *European Journal of Operational Research*, 144, 2003, pp. 27–38.
6. Marinakis, Y. and Marinaki, M.: A hybrid genetic Particle Swarm Optimization Algorithm for the vehicle routing problem, *Expert Systems with Applications*, 37, 2010, pp. 1446–1455.
7. Dijkstra, E. W.: A note on two problems in connexion with graphs, *Numerische Mathematik*, 1959, pp. 269–271.
8. Awerbuch, B.: Distributed shortest paths algorithms, *Proceedings of 21st Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 490–500.
9. Bellman, R. E.: On a routing problem, *Quarterly of Applied Mathematics*, 1958, pp. 87–90.
10. Ahn, C. W. and Ramakrishna, R.S.: A genetic algorithm for shortest path routing problem and the sizing of populations, *IEEE Transactions on Evolutionary Computation*, 6, 2002, pp. 566–579.
11. Ahn, C.W., Ramakrishna, R.S., Kang, C.G. and Choi, I.C.: Shortest path routing algorithm using Hopfield neural network, *Electronics Letters*, 37, 2001, pp. 1176–1178.
12. Juang, C.-F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34, 2004, pp. 997–1006.
13. Isaacs, A. and Ray, T. and Smith, W.: A Hybrid Evolutionary Algorithm With Simplex Local Search, *IEEE Congress on Evolutionary Computation*, 2007, pp. 1701–1708.
14. Queiroz, L.M.O. and Lyra, C.: Adaptive Hybrid Genetic Algorithm for Technical Loss Reduction in Distribution Networks Under Variable Demands, *IEEE Transactions on Power Systems*, 24, 2009, pp. 326–335.
15. Ezeldin, A. S. and Soliman, A.: Hybrid time-cost optimization of nonserial repetitive construction projects, *Journal of Construction Engineering and Management*, 135, 2009, pp. 42–55.
16. Choi, S.S. and Moon, B.R.: A graph-based Lamarckian-Baldwinian hybrid for the sorting network problem, *IEEE Transactions on Evolutionary Computation*, 9, 2005, pp. 105–114.
17. Yan, G.: Design of qos anycast network cluster balance based on genetic algorithm, *Proceedings of International Conference on Signal Processing Systems*, 2009, pp. 610–614.
18. Arabas, J. and Kozdrowski, S.: Applying an evolutionary algorithm to telecommunication network design, *IEEE Transactions on Evolutionary Computation*, 5, 2001, pp. 309–322.
19. Ji, X.: Models and algorithm for stochastic shortest path problem, *Applied Mathematics and Computation*, 170, 2005, pp. 503–514.
20. Vijayalakshmi, K. and Radhakrishnan, S.: Artificial immune based hybrid GA for QoS based multicast routing in large scale networks (AISMR), *Computer Communications*, 31, 2005, pp. 3984–3994.
21. Wang, J. and Wang, M. and Huang, M.: A hybrid intelligent qos multicast routing algorithm in ngi, *International Conference on Communication Technology*, 2006, pp. 1–4.
22. El-Mihoub, T. and Hopgood, A. A. and Nolle, L. and Battersby, A., K.: Performance of hybrid genetic algorithms incorporating local search, *18th European Simulation Multiconference*, 2004, pp. 154–160.
23. Potter, C. and Venayagamoorthy, G.K. and Kosbar, K.: MIMO beam-forming with neural network channel prediction trained by a novel PSO-EA-DEPSO algorithm, *IEEE International Joint Conference on Neural Networks*, 2008, pp. 3338–3344.
24. Yang, S., Cheng, H. and Wang, F.: Genetic Algorithms With Immigrants and Memory Schemes

- for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40, 2010, pp. 52–63.
25. Zhang, X. and Tang, L.: A new hybrid ant colony optimization algorithm for the vehicle routing problem, *Pattern Recognition Letters*, 30, 2009, pp. 848–855.
 26. Ali, M.K.M. and Kamoun, F.: Genetic Algorithms With Immigrants and Memory Schemes for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks, *IEEE Transactions on Neural Networks*, 4, 1993, pp. 941–954.
 27. Park, D.C. and Choi, S.E.: A neural network based multi-destination routing algorithm for communication network, *International Joint Conference on Neural Networks*, 1998, pp. 1673–1678.
 28. Kennedy, J. and Eberhart, R.: Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.
 29. Mohemmed, A. W. and Sahoo, N. C. and Geok, T. K.: Solving shortest path problem using particle swarm optimization, *Applied Soft Computing*, 8, 2008, pp. 1643–1653.
 30. Das, S. and Abraham, A. and Chakraborty, U.K. and Konar, A.: Differential Evolution Using a Neighborhood-Based Mutation Operator, *IEEE Transactions on Evolutionary Computation*, 13, 2009, pp. 526–553.
 31. Zhang, J. and Sanderson, A.C.: JADE: Adaptive Differential Evolution With Optional External Archive, *IEEE Transactions on Evolutionary Computation*, 13, 2009, pp. 945–958.
 32. Fakcharoenphol, J. and Rao, S.: Planar Graphs, Negative Weight Edges, Shortest Paths, and Near Linear Time, *Proc. 42nd IEEE Ann. Symp. Foundations of Computer Science (FOCS'01)*, 2001, pp. 232–241.
 33. Gao et al.: Dynamic Shortest Path Algorithms for Hypergraphs, *10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2012, pp. 238–245.
 34. Klein, P. and Rao, S. and Rauch, M. and Subramanian, S.: Faster Shortest Path Algorithms for Planar Graphs, *Proc. 26th Ann. ACM Symp. Theory of Computing (STOC'94)*, 1994, pp. 27–37.
 35. Baswana, S. and Hariharan, R. and Sen, S.: Improved Decremental Algorithms for Maintaining Transitive Closure and All-Pairs Shortest Paths, *Proc. 34th Ann. ACM Symp. Theory of Computing (STOC'02)*, 2002 pp. 113–127.
 36. Ausiello, G. and Italiano, G.F. and Marchetti-Spaccamela, A. and Nanni, U.: Incremental Algorithms for Minimal Length Paths, *J. Algorithms*, 12, 1991, pp. 615–638.
 37. King, V.: Fully Dynamic Algorithms for Maintaining All-Pairs Shortest Paths and Transitive Closure in Digraphs, *Proc. 40th IEEE Symp. Foundations of Computer Science (FOCS'99)*, 1999, pp. 81–91.
 38. Kusetogullari, H. and Leeson, M. S. and Kole, B. and Hines, E. L.: Meta-heuristic algorithms for optimized network flow wavelet-based image coding, *Applied Soft Computing*, 14, 2014, pp. 536–553.
 39. Buriol, L.S. and Resende, M.G.C. and Thorup, M.: Speeding Up Dynamic Shortest Path Algorithms, *TR TD-5RJ8B, AT&T Labs Research*, Sept. 2003.
 40. Xiao, B. and Cao, J. and Shao, Z. and Sha, E.H.M.: An Efficient Algorithm for Dynamic Shortest Path Tree Update in Network Routing, *Journal of Communications and Networks*, 9, 2007, pp. 499–510.
 41. Demetrescu, C. and Frigioni, D. and Marchetti-Spaccamela, A. and Nanni, U.: Maintaining Shortest Paths in Digraphs with Arbitrary Arc Weights: An Experimental Study, *Proc. 4th Int. Workshop Algorithm Eng. (WAE'01)*, 2001, pp. 218–229.
 42. Frigioni, D. and Ioffreda, M. and Nanni, U. and Pasquale, G.: Experimental Analysis of Dynamic Algorithms for the Single-Source Shortest Path Problem, *ACM J. Experimental Algorithms*, 3, 1998, pp. 5.
 43. Kusetogullari, H. and Leeson, M. S. and Ren, W. and Hines, E. L.: K- Shortest Path Network Problem Solution with a Hybrid Genetic Algorithm - Particle Swarm Optimization Algorithm, *13th IEEE International Conference on Transparent Optical Networks (ICTON'11)*,

- 2011, pp. 1–6.
44. Jaynes, E.T.: Information theory and statistical mechanics, *Physical Review*, 106, 1957, pp. 620–630.
 45. Sharif, M.H. and Djeraba, C.: An entropy approach for abnormal activities detection in video streams, *Pattern Recognition*, 45, 2012, pp. 2543–2561.
 46. Zhang, J. and Chung, H. S.H. and Lo, W.L.: Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms, *IEEE Transactions on Evolutionary Computation*, 22, 2007, pp. 326–335.
 47. Sharif, M.H.: High-performance mathematical functions for single-core architectures, *Journal of Circuits, Systems and Computers*, 23, 2014, pp. 1450051.
 48. Michalewicz, Z.: *Genetic Algorithms+Data Structure=Evolution Programs*, IEEE Transactions on Evolutionary Computation, Springer-Verlag, Berlin, Germany, 1996.
 49. Sayoud, H. and Takahashi, K. and Vaillant, B.: Designing communication network topologies using steady-state genetic algorithms, *IEEE Communications Letters*, 5, 2001, pp. 113–115.
 50. Gen, M. and Cheng, R. and Wang, D.: Genetic algorithms for solving shortest path problems, *IEEE International Conference on Evolutionary Computation*, 1997, pp. 401–406.
 51. Sinalgo: a simulation framework for manets. Available on : dcg.ethz.ch/projects/sinalgo.
 52. Leesutthipornchai, P. and Charnsripinyo, C. and Wattanapongsakorn, N.: Solving multi-objective routing and wavelength assignment in WDM network using hybrid evolutionary computation approach, *Computer Communications*, 33, 2010, pp. 2246 – 2259.
 53. Goodman, S.N.: Toward Evidence-Based Medical Statistics. 1: The P Value Fallacy, *Annals of Internal Medicine*, 130, 1999, pp. 995–1004.
 54. Trawinski, B. and Smetek, M. and Telec, Z. and Lasota, T.: Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms, *International Journal of Applied Mathematics and Computer Science*, 22, 2012, pp. 867-881.
 55. Sellke, T. and Bayarri, M.J. and Berger, J.O.: Calibration of p Values for Testing Precise Null Hypotheses, *The American Statistician*, 55, 2001, pp. 6271.
 56. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *J. of the American Statistical Assoc.*, 32:200, 1937, pp. 675-701.
 57. Iman, R.L. and Davenport, J.M.: Approximations of the critical region of the Friedman statistic, *Communications in Statistics*, 18, 1980, pp. 571-595.
 58. Hodges, J.L. and Lehmann, E.L: Ranks methods for combination of independent experiments in analysis of variance, *Annals of Mathematical Statistics*, 33, 1962, pp. 482-497.
 59. Quade D.: Using weighted rankings in the analysis of complete blocks with additive block effects, *Journal of the American Statistical Association*, 74, 1979, pp. 680–683.
 60. Dunn, O.: Multiple comparisons among means, *Journal of the American Statistical Association*, 56, 1961, pp. 52–64.
 61. Holm, S.: A simple sequentially rejective multiple test procedure, *Scandinavian Journal of Statistics*, 6, 1979, pp. 65–70.
 62. Hochberg, Y.: A sharper Bonferroni procedure for multiple tests of significance, *Biometrika*, 75, 1988, pp. 800–803.
 63. Hommel, G.: A stagewise rejective multiple test procedure based on a modified Bonferroni test, *Biometrika*, 75, 1988, pp. 383–386.
 64. Hommel, G., Bernhard, G.: A rapid algorithm and a computer program for multiple test procedures using procedures using logical structures of hypotheses. *Computer Methods and Programs in Biomedicine*, 1994, 43, pp. 213-216.
 65. Holland, M.C.B.S.: An improved sequentially rejective Bonferroni test procedure, *Biometrics*, 43, 1987, pp. 417–423.
 66. Rom, D.: A sequentially rejective test procedure based on a modified Bonferroni inequality, *Biometrika*, 77, 1990, pp. 663–665.
 67. Finner, H.: On a monotonicity problem in step-down multiple test procedures, *Journal of the*

- American Statistical Association, 88, 1993, pp. 920–923.
68. Li, J.: A two-step rejection procedure for testing multiple hypotheses, *Journal of Statistical Planning and Inference*, 138, 2008, pp. 1521–1527.
 69. Nemenyi, P.B.: *Distribution-free Multiple comparisons*, PhD thesis, Princeton University, 1963.
 70. Shaffer, J.: Modified sequentially rejective multiple test procedures, *Journal of American Statistical Association*, 81, 1986, pp. 826–831.
 71. Bergmann, G. and Hommel, G.: Improvements of general multiple test procedures for redundant systems of hypotheses, in: P. Bauer, G. Hommel, E. Sonnemann (Eds.), *Multiple Hypotheses Testing*, Springer, 1988, pp. 100–115.
 72. Garca, S. and Herrera, F.: An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, *Journal of Machine Learning Research*, 2008, 9, pp. 2677–2694.
 73. Westfall, P.H. and Young, S.S.: *Resampling-based Multiple Testing: Examples and Methods for p-Value Adjustment*, John Wiley and Sons, 2004.