Utilitarian Resource Assignment*

Petra Berenbrink
School of Computing Science
Simon Fraser University

Leslie Ann Goldberg
Department of Computer Science
University of Warwick

Paul Goldberg
Department of Computer Science
University of Warwick

Russell Martin
Department of Computer Science
University of Warwick

Abstract

This paper studies a resource allocation problem introduced by Koutsoupias and Papadimitriou. The scenario is modelled as a multiple-player game in which each player selects one of a finite number of known resources. The cost to the player is the total weight of all players who choose that resource, multiplied by the "delay" of that resource. Recent papers have studied the Nash equilibria and social optima of this game in terms of the L_∞ cost metric, in which the social cost is taken to be the maximum cost to any player. We study the L_1 variant of this game, in which the social cost is taken to be the sum of the costs to the individual players, rather than the maximum of these costs. We give bounds on the size of the coordination ratio, which is the ratio between the social cost incurred by selfish behavior and the optimal social cost; we also study the algorithmic problem of finding optimal (lowest-cost) assignments and Nash Equilibria. Additionally, we obtain bounds on the ratio between alternative Nash equilibria for some special cases of the problem.

---

*Research Report CS-RR-405, Department of Computer Science, University of Warwick, October 2004.
This work was partially supported by the IST Program of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-1999-14036 (RAND-APX).

1 Introduction

This paper studies the resource allocation problem introduced by Koutsoupias and Papadimitriou [7]. In this problem, we are given a collection of resources such as computer servers, printers, or communication links, each of which is associated with a "delay"[1]. We are also given a collection of tasks, each of which is associated with a "weight" corresponding to its size. Each task chooses a resource. A given resource is shared between its tasks in such a way that each of these tasks incurs a cost corresponding to the time until the resource has completed its work. For example, the task might model a routing request and the resources might model parallel links of an network. If routing requests are broken into packets and these are sent in a round-robin fashion, each request will finish at (approximately) the time that the link finishes its work.

We assume that each task chooses its resource in a selfish manner, minimizing its own cost. Following [7] we are interested in determining the social cost of this selfish behavior. Previous work on this problem has measured "social cost" in terms of the L_∞ metric — that is, the longest delay incurred by any task. Our measure of social cost is the L_1 metric – that is, the average delay (over tasks). This is sometimes called the *utilitarian* interpretation of social welfare, and is a standard assumption in the multi-agent system literature, for example [3, 10, 13]. Furthermore, the model of [11] also uses the L_1 metric in the limit of infinitely many tasks.

We give bounds on the size of the coordination ratio, which is the ratio between the social cost incurred by selfish behavior and the optimal social cost [7]; we also study the algorithmic problem of finding optimal (lowest-cost) assignments. For the case of identical resources or identical tasks we obtain bounds on the ratio between alternative Nash equilibria.

Our results show that the L_1 metric behaves very different to the L_∞ metric. By an *assignment* we mean the set of choices of resource that are made by each task. In the case of the L_∞ metric, there always exists an optimal assignment that is also Nash, but the costs of different Nash assignments can differ a lot. In the case of the L_1 metric, the costs of any optimal assignment and the cost of the minimum-cost Nash assignment can be arbitrarily far away from each other, but in a lot of cases the costs of different Nash assignments can differ only by a constant factor.

1.1 The model

Here is the model from [7] (which is introduced in the context of networks, as mentioned above). We are given a set R of m resources with delays $d_1 \leq \ldots \leq d_m$. We are also given a set T of n tasks with weights $w_1, \ldots, w_n$. We assume that $w_i > 0$ for all i, and we let $W = \sum_{i=1}^n w_i$ denote the total task load. Each task will select one resource. Thus, an *assignment* is a vector $A = (A_1, \ldots, A_n)$ which assigns the ith task to resource $A_i \in R$. (In the language of game theory, an assignment associates each task with a "pure strategy".[2]) Let $\mathcal{A} = \{1, \ldots, m\}^n$ denote the set of all assignments. The *load* of resource ℓ in assignment A is defined to be

$$L(\ell, A) = d_\ell \sum_{i \in T : A_i = \ell} w_i.$$

[1] The delay is the reciprocal of the quantity commonly called the "speed" or "capacity" in related work. It is convenient to work in terms of the delay, as defined here, because this simplifies our results.

[2] [7] also considers mixed strategies. See Section 1.3.

The load of task i in assignment A is $L(A_i, A)$. Finally, the (social) *cost* of assignment A is given by

$$C(A) = \sum_{i \in T} L(A_i, A).$$

The notion of "selfish behavior" that we study comes from the game-theoretic notion of a Nash equilibrium. An assignment A is a *Nash equilibrium* if and only if no task can lower its own load by changing its choice of resource (keeping the rest of the assignment fixed). More formally, A is said to be a *Nash* assignment if, for every task i and every resource ℓ, we have $L(A_i, A) \leq L(\ell, A')$, where the assignment A' is derived from A by re-assigning task i to resource ℓ, and making no other change. We let $\mathcal{N}(T, R)$ denote the set of all Nash assignments for problem instance (T, R). When the problem instance is clear from the context, we refer to this as $\mathcal{N}$. For a given problem instance, we study the *coordination ratio* from [7] which is the ratio between the cost of the highest-cost Nash assignment and the cost of the lowest-cost assignment. That is

$$\frac{\max_{N \in \mathcal{N}} C(N)}{\min_{A \in \mathcal{A}} C(A)}.$$

This ratio measures the extent to which the social cost increases if we use a worst-case Nash equilibrium rather than an optimal assignment. We also study the ratio between the lowest cost of a Nash assignment and the lowest cost of an (arbitrary) assignment and also the ratio between the lowest cost of a Nash assignment and the highest cost of a Nash assignment.

Note that throughout the paper we study the average cost-per-task. The reader should not confuse this with the average cost-per-resource. The latter is trivial to optimize (it is achieved by assigning all tasks to the link with the lowest delay) but it is not natural.

1.2 Results

Thorem 2.5 in Section 2 bounds the coordination ratio in terms of the range over which the task weights vary. In particular, suppose that all task weights w_i lie in the range $[1, w_{\max}]$. Then

$$\frac{\max_{N \in \mathcal{N}} C(N)}{\min_{A \in \mathcal{A}} C(A)} \leq 4 w_{\max}.$$

Several of our results focus on the special cases in which the resource delays are identical (Section 3) or the task weights are identical (Section 4). The results are summarized as follows.

Section 3: Resources with Identical Delays

1. (Lemma 3.2) For every n, there is a problem instance with n tasks with weights in the range $[1, n^2]$ for which

   $$\frac{\min_{N \in \mathcal{N}} C(N)}{\min_{A \in \mathcal{A}} C(A)} \geq \frac{n}{5}.$$

   Note that this is the ratio of the best Nash cost to the optimal cost of an assignment, hence it gives a lower bound on the coordination ratio. This lower bound should be contrasted with Theorem 2.5 which gives an upper bound of $4n^2$.

2. (Theorem 3.3) Nash assignments satisfy the following relation:

$$\frac{\max_{N \in \mathcal{N}} C(N)}{\min_{N \in \mathcal{N}} C(N)} \leq 3.$$

3. (Lemma 3.4) For every $\epsilon > 0$, there is an instance satisfying

$$\frac{\max_{N \in \mathcal{N}} C(N)}{\min_{N \in \mathcal{N}} C(N)} \geq \frac{5}{3}(1 - \epsilon).$$

   The size of the problem instance depends upon ϵ.

Section 4: Tasks with Identical Weights

Theorem 2.5 gives an upper bound of 4 for the coordination ratio in the case of identical weights. We also have the following results.

1. (Lemma 4.6) For any $\epsilon > 0$ there is a problem instance for which

$$\frac{\min_{N \in \mathcal{N}} C(N)}{\min_{A \in \mathcal{A}} C(A)} \geq \frac{4}{3} - \epsilon.$$

2. (Theorem 4.7) The lowest-cost and highest-cost Nash assignments satisfy:

$$\frac{\max_{N \in \mathcal{N}} C(N)}{\min_{N \in \mathcal{N}} C(N)} \leq \frac{4}{3}$$

   which is an exact result; we show that $4/3$ is obtainable for some instance.

3. (Theorems 4.2 and 4.5) We give algorithms for finding a lowest-cost assignment and a lowest-cost Nash assignment. These algorithms run in time $\mathrm{O}(mn)$.

Finding social optima using dynamic programming

In Section 5 we show how dynamic programming can be used to find optimal assignments under the L_1 metric, in either the identical-tasks case, or the identical-resources case. The algorithms extend to the case where either the task sizes or the delays may take a limited set of values. This extension is used to give approximation schemes for the cases where instead of a limit on the number of distinct values, we have a limit on the ratio of largest to smallest values.

1.3 Alternative models and related work

There are two collections of work related to our paper. The first uses a similar model, but a different cost function. The second uses a similar cost function, but a different model.

The model that we study was introduced by Koutsoupias and Papadimitriou [7], who initiated the study of coordination ratios. They worked in the more general setting of *mixed strategies*. In a mixed strategy, instead of choosing a resource A_i, task i chooses a vector $(p_{i,1}, \ldots, p_{i,m})$ in which $p_{i,j}$ denotes the probability with which task i will use resource j. A collection of mixed strategies (one strategy for each task) is a Nash equilibrium if no task can reduce its expected cost by modifying its own probability vector. Unlike us, Koutsoupias and Papadimitriou measure social cost in terms of the L_∞ metric. Thus, the cost of a collection of strategies is the (expected) maximum load of a resource (maximized over all resources). Their coordination ratio is the ratio between the maximum cost (maximized over all Nash equilibria) divided by the cost of the optimal solution. Koutsoupias and Papadimitriou give bounds on the coordination ratio. These bounds are improved by Mavronicolas and Spirakis [9], and by Czumaj and Vöcking [1] who gave an asymptotically tight bound. Fotakis et al. [6] consider the same model. They study the following algorithmic problems: constructing a Nash equilibrium, constructing the worst Nash equilibrium, and computing the cost of a given Nash equilibrium. For our purposes, we note that the existence of at least one pure Nash assignment (as defined in Section 1.1) was also proven in [6]. Czumaj et al. [2] give further results for the model of [7] using the L_∞ metric for a wide class of so-called *simple* cost functions. They call a cost function simple if it depends only on the injected load of the resources. They also show that for some families of simple monotone cost functions, these results can be carried over to the L_1 metric. These are qualitative results relating the boundedness of the coordination ratio in terms of boundedness of the bicriteria ratio; in contrast here we are studying quantative bounds on the coordination ratio for a special case of non-simple cost functions.

In [5] Gairing et al. study the combinatorial structure and computational complexity of *extreme Nash equilibria*, i.e. equilibria that maximize or minimize the objective function. Their results provide substantial evidence for the *Fully Mixed Nash Equilibrium Conjecture*, which states that the worst case Nash equilibrium is the fully mixed Nash equilibrium where each user chooses each link with positive probability. They also develop some algorithms for *Nashification*, which is the problem of transforming an arbitrary pure strategy profile into a pure Nash equilibrium without increasing the social cost. In [4] Feldmann et al. give a polynomial time algorithm for Nashification and a polynomial time approximation scheme (PTAS) for computing a Nash equilibrium with minimum social cost. In [8] Lücking et al. continue to study the Fully Mixed Nash Equilibrium Conjecture and report substantial progress towards identifying the validity. Note that all these publications use the L_∞ metric to measure the social cost.

Roughgarden and Tardos [11] study coordination ratios in the setting of traffic routing. A problem instance specifies the rate of traffic between each pair of nodes in an arbitrary network. Each agent controls a small fraction of the overall traffic. Like us, Roughgarden and Tardos use an L_1 cost-measure. That is, the cost of a routing is the sum of the costs of the agents. The model of Roughgarden and Tardos is in one sense much more general than our model (from [7]) which corresponds to a two-node network with many parallel links. However, most work in the model of [11] relies on the simplifying assumption that each agent can split its traffic arbitrarily amongst different paths in the network. In our model, this assumption would correspond to allowing a task to split itself between the resources, dividing its weight into arbitrary proportions — a simplification which would make our problems trivial. In particular, this simplification forces all Nash assignments to have the same L_1 cost, which is not true in the unsplittable model that we study. In fact, in [11] it is demonstrated that if agents are not allowed to split their traffic arbitrarily but each chooses a single path on which to route their own traffic, then the cost of a Nash assignment can be arbitrarily larger than an optimal (lowest-cost) assignment. This is in contrast to their elegant coordination ratio [11] for the variant that they study. Even in our model, the splittable-task variant is useful as a proof device. In Section 2, we use the splittable-task setting to derive a lower bound on the cost of Nash assignments in our model. For other interesting results in the model of Roughgarden and Tardos, see [11] and [12].

2 Coordination Ratio in Terms of Task Weight Range

Suppose that the weights lie in the range $[1, w_{\max}]$. The purpose of this section is to prove Theorem 2.5, which shows that the coordination ratio is at most $4w_{\max}$.

Definition 2.1 *A* fractional assignment *A^F for an instance (T, R) is a collection of real numbers $h_t(\ell)$ for $t \in T, \ell \in R$, such that $0 \leq h_t(\ell) \leq 1$ and $\sum_{\ell \in R} h_t(\ell) = 1$ for all $t \in T$.*

If A^F is a fractional assignment, the load of resource ℓ is defined as $L(\ell, A^F) = d_\ell \sum_{i \in T} w_i h_i(\ell)$. The cost of task i is defined as $C_i(A^F) = \sum_{\ell \in R} h_i(\ell) L(\ell, A^F)$ and the cost of A^F is defined as $C(A^F) = \sum_{i \in T} C_i(A^F)$.

An integral assignment *is a fractional assignment where all the quantities $h_t(\ell)$ are equal to 0 or 1. Note that we reserve the notation A (or $A(T, R)$ to denote the sets of tasks and resources) strictly for integral assignments.*

Define the throughput *of resource set R to be $D = \sum_{\ell \in R} \frac{1}{d_\ell}$.*

We use Definition 2.1 to provide a lower bound on the cost of any integral assignment for a given instance (T, R). We start by giving a lower bound on the cost of a fractional assignment. The following lemma is essentially the same as Lemma 2.6 of [11].

Lemma 2.2 *If all tasks have weight 1, then the optimal fractional assignment $A^{F,opt}$ gives each resource a load of n/D and therefore any task t has $C_t(A^{F,opt}) = n/D$.*

Proof: Let $x_\ell = \sum_{i \in T} h_i(\ell)$. From Definition 2.1, the load of resource ℓ is $x_\ell d_\ell$.

We have:

$$\sum_{\ell \in R} x_\ell = n \tag{1}$$

$$C(A^F) = \sum_{i \in T} C_i(A^F) = \sum_{i \in T} \sum_{\ell \in R} h_i(\ell) L(\ell, A^F) = \sum_{i \in T} \sum_{\ell \in R} h_i(\ell) d_\ell \sum_{i \in T} h_i(\ell)$$

where we have used $w_i = 1$ in the expression for $L(\ell, A^F)$. Thus,

$$C(A^F) = \sum_{i \in T} \sum_{\ell \in R} h_i(\ell) x_\ell d_\ell = \sum_{\ell \in R} \sum_{i \in T} h_i(\ell) x_\ell d_\ell = \sum_{\ell \in R} x_\ell d_\ell \sum_{i \in T} h_i(\ell) = \sum_{\ell \in R} x_\ell^2 d_\ell.$$

Equation (1) gives a linear constraint on the x_ℓ values, and we have expressed $C(A^F)$ in terms of the x_ℓ values. To minimise $C(A^F)$ subject to (1) we use the well-known method of Lagrange multipliers. This means that the gradient of $C(A^F)$ and that of the function $\sum_{\ell \in R} x_\ell$ must have the same direction:

$$\exists \Lambda \in \mathbb{R} \text{ such that } \nabla(C(A^F)) = \Lambda \nabla\Big(\sum_{\ell \in R} x_\ell\Big)$$

$$\text{i.e.} \quad (2d_1 x_1, 2d_2 x_2, \ldots, 2d_m x_m) = (\Lambda, \Lambda, \ldots, \Lambda).$$

Hence, at the optimum we see that $x_\ell = \frac{\Lambda}{2d_\ell}$ for all ℓ. Using (1), we then find that $x_\ell = \frac{n}{Dd_\ell}$, and $L(\ell, A^{F,opt}) = x_\ell d_\ell = n/D$ for all $\ell \in R$.

Finally, for any task i

$$C_i(A^{F,opt}) = \sum_{\ell \in R} h_i(\ell) L(\ell, A^{F,opt}) = \sum_{\ell \in R} h_i(\ell) \frac{n}{D} = \frac{n}{D} \sum_{\ell \in R} h_i(\ell) = \frac{n}{D}.$$

□

The above result provides a useful lower bound on the cost of any integral assignment A. We make one refinement for the lower bound: note that if $m > n$, then any Nash or optimal assignment will only use n resources having smallest delays.[3] Hence an instance (T, R) with $m > n$ can be modified by removing the $m - n$ resources with largest delay. In what follows, we shall therefore make the assumption that $n \geq m$. We next proceed to give a bound on the coordination ratio for tasks having weights in the range $[1, w_{max}]$. We first give a definition and an observation that will be useful to us.

Definition 2.3 *Given a set R of m resources and a set of $n \geq m$ tasks, we say resource ℓ is* fast *provided that $d_\ell \leq 2n/D$, otherwise ℓ is* slow.

Given a set of tasks T, let T^* denote a set of tasks such that $|T^*| = |T|$ and each task $t \in T^*$ has unit weight. We first make an observation about the slow and fast resources for the optimal fractional assignment $A^{F,opt}(T^*, R)$.

Observation 2.4 *For any sets T, R, in the optimal fractional assignment for the instance (T^*, R) we have*

$$\sum_{\ell \in R; \ell \text{ fast}} \sum_{i \in T^*} h_i(\ell) \geq n/2.$$

Proof: Let $A^{F,opt}$ denote an optimal fractional assignment. First note that $\sum_{\ell \in R} \sum_{i \in T^*} h_i(\ell) = n$.

Using Lemma 2.2 (and the definition of a "slow resource") we find that in $A^{F,opt}(T^*, R)$ each slow resource ℓ satisfies $\sum_{i \in T^*} h_i(\ell) \leq 1/2$. Since we assume $n \geq m$, at most n resources are slow, so that $\sum_{\ell \in R; \ell \text{ slow}} \sum_{i \in T^*} h_i(\ell) \leq n/2$.

The result follows from $\sum_{\ell \in R; \ell \text{ fast}} \sum_{i \in T^*} h_i(\ell) = \sum_{\ell \in R} \sum_{i \in T^*} h_i(\ell) - \sum_{\ell \in R; \ell \text{ slow}} \sum_{i \in T^*} h_i(\ell)$. □

Here is our bound on the coordination ratio for tasks having weights in the range $[1, w_{max}]$.

Theorem 2.5 *Suppose (T, R) is a problem instance with n tasks having weights in the range $[1, w_{\max}]$ and m resources. Then*

$$\max_{N \in \mathcal{N}} C(N) \leq 4 w_{\max} \min_{A \in \mathcal{A}} C(A).$$

Proof: Following our comments preceeding Definition 2.3 we again assume that $n \geq m$.

Let $\mathcal{A}^F(T, R)$ denote the set of all fractional assignments for the instance (T, R). As before, we let T^* denote the set of unit-weight tasks, where $|T^*| = |T|$. We first note that

$$\min_{A \in \mathcal{A}(T,R)} C(A) \geq \min_{A^F \in \mathcal{A}^F(T,R)} C(A^F) \geq \min_{A^F \in \mathcal{A}^F(T^*,R)} C(A^F) = \frac{n^2}{D}. \quad (2)$$

[3] If the number of resources is allowed to be large by comparison with the number of tasks, then the optimal fractional assignment can be made artificially much lower than any integral assignment, by including a large number of resources with very large delays, thereby inflating the value of D.

The last equality is an application of Lemma 2.2 to the instance (T^*, R).

We show that in any integral Nash assignment N, all tasks i satisfy the inequality $L(N_i, N) \leq 4w_{max}\left(\frac{n}{D}\right)$. This would then imply that $\max_{N\in\mathcal{N}} C(N) = \sum_{i\in T} L(N_i, N) \leq 4w_{max}\left(\frac{n^2}{D}\right)$. This, together with (2), gives us the result.

Let N denote a Nash assignment. Suppose that under this assignment some resource j satisfies

$$L(j, N) > 4w_{\max}\left(\frac{n}{D}\right).$$

We prove that N is not Nash, by finding an assignment N' (obtained from N) by transferring one task from resource j to some j' such that

$$L(j', N') \leq 4w_{\max}\left(\frac{n}{D}\right).$$

We start by proving there exists a fast resource j' such that $L(j', N) \leq 2w_{\max}(\frac{n}{D})$. To prove this, suppose for a contradiction that all fast resources ℓ satisfy

$$L(\ell, N) > 2w_{\max}\left(\frac{n}{D}\right). \tag{3}$$

Let $A^{F,opt}$ denote an optimal fractional assignment for the instance (T^*, R). We recall from Lemma 2.2 that $L(\ell, A^{F,opt}) = \frac{n}{D}$ for all resources ℓ. Thus, if a fast resource ℓ satisfies (3), we must have $L(\ell, N)/d_\ell > 2w_{\max}L(\ell, A^{F,opt})/d_\ell$. This means that

$$\sum_{i\in T; N_i=\ell} w_i > 2w_{\max} \sum_{i\in T^*} h_i(\ell) \tag{4}$$

where $h_i(\ell)$ are the values for the optimal fractional assignment $A^{F,opt}$.

However, from Observation 2.4 we know that in $A^{F,opt}$

$$\sum_{\ell\in R;\ell \text{ fast}} \sum_{i\in T^*} h_i(\ell) \geq \frac{n}{2}$$

which, with Equation (4) implies

$$\sum_{\ell\in R;\ell \text{ fast}} \sum_{i\in T: N_i=\ell} w_i > \frac{n}{2}(2w_{\max}) = nw_{\max}.$$

This is a contradiction since the left hand side of this inequality (which is the sum of weights in the instance (T, R)) is at most nw_{max}.

Since we have a contradiction, we instead conclude there exists a fast resource j' where

$$L(j', N) \leq 2w_{\max}\left(\frac{n}{D}\right).$$

We now show how to construct N' from N, thereby proving that N was not a Nash assignment, a contradiction.

Recall since j' is a fast resource, $d_{j'} \leq \frac{2n}{D}$. We consider two cases for j'. Let $k = L(j', N)/d_{j'}$.

If $k \le w_{\max}$, then moving one task from resource j to resource j' (to get the new assignment N'), we find that

$$L(j', N') \le d_{j'}(k + w_{\max}) \le 2\left(\frac{n}{D}\right)(w_{\max} + w_{\max}) \le 4w_{\max}\left(\frac{n}{D}\right).$$

If instead $k > w_{\max}$, then moving one task from j to j' to get N', we find

$$L(j', N') = (k + w_{\max})d_{j'} \le 2kd_{j'} = 2L(j', N) \le 4w_{\max}\left(\frac{n}{D}\right).$$

In either case, we have shown that N is not a Nash assignment because we can move one task (currently having a load greater than $4w_{max}(\frac{n}{D})$ from resource j to resource j' where it has a lesser load.

Thus, we conclude that if N is a Nash assignment, then $L(j, N) \le 4w_{max}\left(\frac{n}{D}\right)$ for all resources j, as desired to prove the theorem. □

3 Resources with Identical Delay

In this section, we restrict our attention to problem instances with identical delays, i.e. $d_1 = d_2 = \cdots = d_m$. If we examine the cost function we are using, we see that if all of the delays are identical, we can factor this term from the cost. Therefore, without loss of generality, we can assume that for all i, $d_i = 1$.

Notation: Recall that $W = \sum_{t \in T} w_i$ denotes the total weight of tasks. Let L_{avg} be the average load on a resource, that is, $L_{\text{avg}} = \frac{1}{m}\sum_{\ell \in R} L(\ell, A) = W/m$. Note in the case of identical (unit) delays, L_{avg} is the same constant value for *all* assignments associated with a given problem instance (T, R).

The following observation will be useful.

Observation 3.1 *Suppose $N \in \mathcal{N}$. Every task i with $w_i > L_{\text{avg}}$ has its own resource (which is not shared) in N.*

Proof: Suppose to the contrary that task i shares a resource with task j. The load of task j is at least $w_j + w_i$. There must be some resource whose load is at most the average load L_{avg}, and task j would prefer to move to this resource, obtaining a new load of at most $w_j + L_{\text{avg}}$. □

The next lemma shows that in the case of identical resources, the ratio between the cost of the minimum (and, hence, any) Nash assignment and the lowest cost of any assignment can be arbitrarily large. In fact, our example needs just two resources to obtain this result.

Lemma 3.2 *For every n, there is an instance having identical resources, and n tasks with weights in the range $[1, n^2]$ for which the following holds:*

$$\min_{N \in \mathcal{N}} C(N) \ge \frac{n}{5} \min_{A \in \mathcal{A}} C(A).$$

Proof: For our problem instance we take $m = 2$, $d_1 = d_2 = 1$, $w_1 = w_2 = n^2$, and $w_3 = \cdots = w_n = 1$.

Any assignment in which tasks 1 and 2 use the same resource is in $\mathcal{A} - \mathcal{N}$ because one of these tasks could move to decrease its own load. Thus, any $N \in \mathcal{N}$ will have tasks 1 and 2 on different resources, which implies $C(N) \geq n^3$. On the other hand, $\min_{A \in \mathcal{A}} C(A) \leq C(A^*)$, where A^* is the assignment which assigns tasks 1 and 2 to resource 1 and the other tasks to resource 2. $C(A^*) = 4n^2 + (n-2)(n-2) \leq 5n^2$. Putting these facts together, for every $N \in \mathcal{N}$,

$$C(N) \geq \frac{n}{5} \min_{A \in \mathcal{A}} C(A).$$

□

Remark: The example from the lemma has $w_{\max} = n^2$ and $w_{\min} = 1$, showing that in this case $C(N) \geq \frac{\sqrt{w_{\max}}}{5} \min_{A \in \mathcal{A}} C(A)$. Thus, the bound of Theorem 2.5 needs to be some function of $w_{\max}$. The example in Section 5.3 of [11] gives an observation similar to Lemma 3.2 for the general-flow setting. The example is a four-node problem instance with two agents. The latency functions may be chosen so that there is a Nash equilibrium which is arbitrarily worse than the social optimum.

Lemma 3.2 shows that the cost of the best assignment and the cost of the best Nash assignment can be arbitrarily far apart. On the other hand, we can show that the costs of different Nash assignments are close to one another.

Theorem 3.3 *For every instance with identical resources we have*

$$\max_{N \in \mathcal{N}} C(N) \leq 3 \min_{N \in \mathcal{N}} C(N).$$

Proof: We first reduce the case in which T contains a task with $w_i > L_{\text{avg}}$ to the case in which T does not contain such a task. Let (T', R') be a problem instance derived from (T, R) by removing a task i with $w_i > L_{\text{avg}}$ and removing one resource. Then by Observation 3.1,

$$\max_{N \in \mathcal{N}(T,R)} C(N) = w_i + \max_{N \in \mathcal{N}(T',R')} C(N).$$

Similarly,

$$\min_{N \in \mathcal{N}(T,R)} C(N) = w_i + \min_{N \in \mathcal{N}(T',R')} C(N).$$

Thus, to prove the theorem, we only need to show

$$\max_{N \in \mathcal{N}(T,R)} C(N) \leq 3 \min_{N \in \mathcal{N}(T,R)} C(N)$$

for problem instances (T, R) in which every task has $w_i \leq L_{\text{avg}}$. Let (T, R) be such an instance.

Consider task i having weight w_i. In a Nash assignment A, the load of task i satisfies

$$L(A_i, A) \geq \max\{w_i, L_{\text{avg}}/2\} \tag{5}$$

since all resources must have load at least $L_{\text{avg}}/2$. (If a resource has load less than $L_{\text{avg}}/2$ then there must be a resource with load strictly larger than L_{avg} with at least 2 tasks on it, because of our assumption that $w_t \leq L_{\text{avg}}$ for all tasks t. Then one of the tasks on this heavily loaded resource would move to the other less loaded one.)

Since A is a Nash assignment, the load of task i satisfies

$$L(A_i, A) \leq L_{\text{avg}} + w_i. \tag{6}$$

The ratio of the upper bound from (6) and the lower bound from (5) is at most 3, attained when $w_i = L_{\text{avg}}/2$. Hence the ratio between total costs (which is the ratio between sums of individual task costs) is upper bounded by 3. □

The following lemma should be compared to Theorem 3.3.

Lemma 3.4 *For every $\epsilon > 0$, there is an instance with identical resources such that*

$$\min_{N\in\mathcal{N}} C(N) \leq \frac{3}{5}(1+\epsilon) \max_{N\in\mathcal{N}} C(N).$$

(The weights and number of tasks in this constructed instance are allowed to depend upon ϵ.)

Proof: The number of tasks n is equal to $6M+13$ where $M = \lceil \frac{2}{\epsilon} \rceil$. T will denote a set of tasks consisting of 6 tasks of weight $3M$, 6 tasks of weight $6M$, and $6M+1$ tasks of weight 1. In this case R consists of 6 resources. Let $N^{(1)}$ be the following Nash assignment:

Resource	Tasks/Resource	Cost/Resource
1	6M+1 tasks, each of weight 1	6M+1
2,3,4	2 tasks, each of weight 6M	12M
5,6	3 tasks, each of weight 3M	9M

Then $C(N^{(1)}) = (6M+1)\cdot(6M+1) + 6\cdot 12M + 6\cdot 9M = 36M^2 + 138M + 1$.
Let $N^{(2)}$ be the following Nash assignment:

Resource	Tasks/Resource	Cost/Resource
1,2,3,4,5	1 task of weight 6M; 1 task of weight 3M; M tasks of weight 1	10M
6	1 task of weight 6M; 1 task of weight 3M; M+1 tasks of weight 1	10M+1

In this case we have $C(N^{(2)}) \geq n \cdot 10M = (6M+13)10M$.

$$\frac{\min_{N\in\mathcal{N}} C(N)}{\max_{N\in\mathcal{N}} C(N)} \leq \frac{C(N^{(1)})}{C(N^{(2)})} \leq \frac{36M^2+138M+1}{10M(6M+13)} = \frac{3}{5}\Big(\frac{6M+23+\frac{1}{6M}}{6M+13}\Big)$$

$$= \frac{3}{5}\Big(1+\frac{10+\frac{1}{6M}}{6M+13}\Big) \leq \frac{3}{5}\Big(1+\frac{11}{6M+13}\Big) \leq \frac{3}{5}\Big(1+\frac{11}{\frac{12}{\epsilon}+13}\Big) = \frac{3}{5}\Big(1+\frac{11\epsilon}{12+13\epsilon}\Big) \leq \frac{3}{5}(1+\epsilon)$$

□

4 Tasks with Identical Weights

In this section, we turn our attention to instances in which the weights of the tasks are identical, but the delays may be diverse. Section 4.1 is algorithmic in nature. There, we present an algorithm that constructs a lowest-cost assignment and an algorithm for finding a Nash assignment with lowest possible cost. In Section 4.2, we compare the cost of Nash assignments to the cost of the best-possible assignment and we compare the cost of the best Nash assignment to the cost of the worst. The comparisons use structural observations arising from the algorithms in Section 4.1.

Definitions: Without loss of generality, we assume that each task has unit weight. Recall that $d_1 \leq d_2 \leq \cdots \leq d_m$. In this section, we use alternative notation to represent an assignment. In particular, an assignment will be denoted as $\overline{n} = \langle n_1, \ldots, n_m \rangle$, where n_ℓ is the *number* of tasks assigned to resource ℓ. Thus $L(\ell, \overline{n}) = n_\ell d_\ell$ and $C(\overline{n}) = \sum_\ell (n_\ell^2 d_\ell)$.

Note that an assignment $\overline{n}$ is a Nash assignment if and only if $n_i d_i \leq (n_j + 1) d_j$ for all i, j.

4.1 Algorithmic Results

We start with a structural observation about lowest-cost assignments.

Lemma 4.1 *Suppose that $\overline{n}$ is a lowest-cost assignment for problem instance (T, R). Let (T', R) be the problem instance derived from (T, R) by adding one task. Let k be any resource that minimizes the quantity $(2n_k + 1)d_k$. Let $\overline{\psi}$ be the assignment for (T', R) which agrees with $\overline{n}$ except that $\psi_k = n_k + 1$. Then $\overline{\psi}$ is a lowest-cost assignment for (T', R).*

Proof: We first argue that the problem instance (T', R) has a lowest-cost assignment $\overline{\nu}$ with $\nu_k \geq \psi_k$. To see this, suppose that $\overline{\sigma}$ is a lowest-cost assignment for (T', R) with $\sigma_k < \psi_k$. Let j be a resource with $\sigma_j > \psi_j$. Let $\overline{\nu}$ be the assignment for (T', R) that agrees with $\overline{\sigma}$ except that $\nu_k = \sigma_k + 1$ and $\nu_j = \sigma_j - 1$. Then

$$\begin{aligned}
C(\overline{\nu}) &= C(\overline{\sigma}) + ((\sigma_k + 1)^2 - \sigma_k^2)d_k + ((\sigma_j - 1)^2 - \sigma_j^2)d_j \\
&= C(\overline{\sigma}) + (2\sigma_k + 1)d_k - (2\sigma_j - 1)d_j \\
&\leq C(\overline{\sigma}) + (2n_k + 1)d_k - (2\sigma_j - 1)d_j \qquad (7) \\
&\leq C(\overline{\sigma}) + (2n_j + 1)d_j - (2\sigma_j - 1)d_j \qquad (8) \\
&\leq C(\overline{\sigma}) + (2n_j + 1)d_j - (2n_j + 1)d_j \qquad (9) \\
&= C(\overline{\sigma}),
\end{aligned}$$

where (7) follows from the upper bound on σ_k, (8) comes from the choice of k, and (9) comes from the choice of j. So by iterating the above argument, we can take $\overline{\nu}$ to be a lowest-cost assignment for (T', R) satisfying $\nu_k \geq \psi_k$.

Suppose now that $C(\overline{\nu}) < C(\overline{\psi})$. Let $\overline{y}$ be the assignment for (T, R) that agrees with ν on resources $\ell \neq k$ and has $y_k = \nu_k - 1$. Then

$$\begin{aligned}
C(\overline{\psi}) &= C(\overline{n}) + (\psi_k^2 - n_k^2)d_k \leq C(\overline{y}) + (\psi_k^2 - n_k^2)d_k \\
&= C(\overline{y}) + (2n_k + 1)d_k \leq C(\overline{y}) + (2\nu_k - 1)d_k = C(\overline{y}) + (\nu_k^2 - (\nu_k - 1)^2)d_k = C(\overline{\nu}),
\end{aligned}$$

where the first inequality follows from the optimality of $\overline{n}$, giving a contradiction to our assumption on the costs of $\overline{\nu}$ and $\overline{\psi}$. Therefore $\overline{\psi}$ is a lowest-cost assignment for (T', R). □

Theorem 4.2 follows directly from Lemma 4.1.

Theorem 4.2 *Let (T, R) be a problem instance with $n \geq 1$ tasks and m resources. Algorithm* **FindOpt** *(see Figure 1) constructs a lowest-cost assignment for (T, R) in $\mathrm{O}(nm)$ time.*

The following lemmas give information about the structure of Nash assignments.

FindOpt(T,R)

1. Set $n_i = 0$ for $i = 1, \dots, m$.
2. For $\tau = 1, \dots, n$
   (a) Choose a resource k so as to minimize $(2n_k + 1)d_k$.
   (b) Increment n_k.
3. Return $\overline{n}$, which is a lowest-cost assignment for (T, R).

Figure 1: An algorithm for constructing a lowest-cost assignment for a problem instance (T, R) with $n \geq 1$ tasks and m resources.

Lemma 4.3 *If $\overline{\nu} \in \mathcal{N}(T, R)$ and $\overline{\rho} \in \mathcal{N}(T, R)$ then, for any $j \in R$, $|\nu_j - \rho_j| \leq 1$.*

Proof: Suppose $\rho_\ell > \nu_\ell$. Let k be a resource such that $\rho_k < \nu_k$. Then since $\overline{\rho}$ and $\overline{\nu}$ are Nash assignments, $\rho_\ell d_\ell \leq (\rho_k + 1)d_k \leq \nu_k d_k \leq (\nu_\ell + 1)d_\ell$, so $\rho_\ell \leq \nu_\ell + 1$. $\square$

Lemma 4.4 *Suppose $\overline{n} \in \mathcal{N}(T, R)$. If $n_i > n_j$ then $d_i \leq d_j$.*

Proof: Suppose to the contrary that $n_i > n_j$ and $d_i > d_j$. Then $(n_j + 1)d_j < n_i d_i$, so $\overline{n}$ is not a Nash assignment. $\square$

FindOptNash(T,R)

1. Set $n_i = 0$ for $i = 1, \dots, m$.
2. For $\tau = 1, \dots, n$
   (a) Let K be the set of resources k that minimize $(n_k + 1)d_k$.
   (b) Choose $k \in K$ so as to minimize n_k.
   (c) Increment n_k.
3. Return $\overline{n}$, which is a lowest-cost assignment in $\mathcal{N}(T, R)$.

Figure 2: An algorithm for constructing a lowest-cost Nash assignment for a problem instance (T, R) with $n \geq 1$ tasks and m resources.

Theorem 4.5 *Let (T, R) be a problem instance with $n \geq 1$ tasks and m resources. Algorithm* **FindOpt-Nash** *(see Figure 2) constructs a lowest-cost assignment in $\mathcal{N}(T, R)$ in $\mathrm{O}(nm)$ time.*

Proof: First note that the algorithm maintains the invariant that the assignment for tasks $1,\ldots,j$ on resources in R is a Nash assignment. This follows from the fact that k is chosen so as to minimize $(n_k+1)d_k$. We prove by induction on n that the constructed assignment has lowest cost amongst Nash assignments. The base case is $n=1$. For the inductive step, let $\overline{n}$ be the (optimal) Nash assignment for a problem instance (T,R) with n tasks constructed by the algorithm. Derive (T',R) from (T,R) by adding one task. Let $\overline{\nu}$ be the assignment constructed by **FindOptNash(T',R)**. Let i be the resource such that $\nu_i = n_i + 1$. Suppose for contradiction that $\overline{\rho} \in \mathcal{N}(T',R)$ satisfies $C(\overline{\rho}) < C(\overline{\nu})$. By Lemma 4.3, there are three cases.

Case 1: $\rho_i = \nu_i = n_i + 1$

Since $C(\overline{\rho}) < C(\overline{\nu})$ there are resources j and ℓ in R such that $\rho_j = \nu_j - 1$ and $\rho_\ell = \nu_\ell + 1$ and

$$\rho_j^2 d_j + \rho_\ell^2 d_\ell < \nu_j^2 d_j + \nu_\ell^2 d_\ell = n_j^2 d_j + n_\ell^2 d_l. \tag{10}$$

Let $\overline{\psi}$ be the assignment constructed by the algorithm just before the n_jth task is assigned to resource j. Then

$$(\psi_\ell + 1)d_\ell \le (n_\ell + 1)d_\ell = \rho_\ell d_\ell \le (\rho_j + 1)d_j = n_j d_j = (\psi_j + 1)d_j, \tag{11}$$

where the second inequality follows from the fact that $\overline{\rho}$ is a Nash assignment. Because the algorithm chose resource j rather than resource ℓ, all of the inequalities in Equation (11) are equalities so

$$n_j d_j = (n_\ell + 1)d_\ell. \tag{12}$$

Furthermore, $\psi_j \le \psi_\ell$ so $n_j - 1 = \psi_j \le \psi_\ell \le n_\ell$ which, together with Equation (12) implies

$$d_j \ge d_\ell. \tag{13}$$

Finally, the following calculation contradicts Equation (10).

$$\begin{aligned}
\rho_j^2 d_j + \rho_\ell^2 d_\ell &= (n_j - 1)^2 d_j + (n_\ell + 1)^2 d_\ell \\
&= n_j^2 d_j + n_\ell^2 d_\ell + (2n_\ell + 1)d_\ell - (2n_j - 1)d_j \\
&= n_j^2 d_j + n_\ell^2 d_\ell + 2(n_\ell + 1)d_\ell - (2n_j - 1)d_j - d_\ell \\
&= n_j^2 d_j + n_\ell^2 d_\ell + 2n_j d_j - (2n_j - 1)d_j - d_\ell \\
&\ge n_j^2 d_j + n_\ell^2 d_\ell.
\end{aligned}$$

The final equality follows from (12) and the inequality follows from (13).

Case 2: $\rho_i = \nu_i - 1 = n_i$

We will construct an assignment $\overline{\sigma} \in \mathcal{N}(T',R)$ with $C(\overline{\sigma}) \le C(\overline{\rho})$ and $\sigma_i = \nu_i$. Case 1 then applies to $\overline{\sigma}$. Let j be a resource with $\rho_j > \nu_j$, so by Lemma 4.3 $\rho_j = \nu_j + 1$. Since $\overline{\nu}$ is a Nash assignment,

$$(n_i + 1)d_i = \nu_i d_i \le (\nu_j + 1)d_j = (n_j + 1)d_j. \tag{14}$$

Since $\overline{\rho}$ is a Nash assignment,

$$(n_j + 1)d_j = \rho_j d_j \le (\rho_i + 1)d_i = \nu_i d_i = (n_i + 1)d_i. \tag{15}$$

Inequalities (14) and (15) together imply

$$(n_i + 1)d_i = (n_j + 1)d_j \tag{16}$$

and

$$(\rho_i + 1)d_i = \rho_j d_j. \tag{17}$$

Since the algorithm chose to assign the last task in (T', R) to resource i rather to resource j, we have $n_i \le n_j$. Lemma 4.4 and Equation (16) imply that $d_i \ge d_j$.

Let $\overline{\sigma}$ be the assignment that agrees with $\overline{\rho}$ except $\sigma_i = \rho_i + 1$ and $\sigma_j = \rho_j - 1$. Equation (17) implies the following facts since $\overline{\rho}$ is a Nash assignment.

1. for $\ell \notin \{i, j\}$, $(\rho_i + 1)d_i = \rho_j d_j \le (\rho_\ell + 1)d_\ell$,
2. for $\ell \notin \{i, j\}$, $\rho_j d_j = (\rho_i + 1)d_i \ge \rho_\ell d_\ell$.

The first of these implies that $\sigma_i d_i \le (\sigma_\ell + 1)d_\ell$ and the second implies that $(\sigma_j + 1)d_j \ge \sigma_\ell d_\ell$. Thus, σ is a Nash assignment. The argument that $C(\overline{\sigma}) \le C(\overline{\rho})$ is exactly the same as the end of Case 1.

$$\begin{aligned} C(\overline{\sigma}) - C(\overline{\rho}) &= (\sigma_i^2 - \rho_i^2)d_i + (\sigma_j^2 - \rho_j^2)d_j \\ &= (2\rho_i + 1)d_i - (2\rho_j - 1)d_j \\ &= 2\rho_j d_j - d_i - (2\rho_j - 1)d_j \\ &= -d_i + d_j \\ &\le 0, \end{aligned}$$

where the second-to-last equality uses Equation (17).

Case 3: $\rho_i = \nu_i + 1 = n_i + 2$

As in Case 2, we construct an assignment $\overline{\sigma} \in \mathcal{N}(T', R)$ with $C(\overline{\sigma}) \le C(\overline{\rho})$ and $\sigma_i = \nu_i$. The argument is similar to Case 2, but is included for completeness.

Let j be a resource with $\rho_j < \nu_j$, so by Lemma 4.3 $\rho_j = \nu_j - 1$. Since $\overline{\nu}$ is a Nash assignment,

$$n_j d_j = \nu_j d_j \le (\nu_i + 1)d_i = (n_i + 2)d_i. \tag{18}$$

Since $\overline{\rho}$ is a Nash assignment,

$$(n_i + 2)d_i = \rho_i d_i \le (\rho_j + 1)d_j = n_j d_j. \tag{19}$$

Inequalities (18) and (19) together imply

$$(n_i + 2)d_i = \rho_i d_i = (\rho_j + 1)d_j = n_j d_j. \tag{20}$$

Let $\overline{\sigma}$ be the assignment that agrees with $\overline{\rho}$ except $\sigma_i = \rho_i - 1$ and $\sigma_j = \rho_j + 1$. Equation (20) implies the following facts since $\overline{\rho}$ is a Nash assignment.

1. for $\ell \notin \{i, j\}$, $(\rho_j + 1)d_j = \rho_i d_i \le (\rho_\ell + 1)d_\ell$,
2. for $\ell \notin \{i, j\}$, $\rho_i d_i = (\rho_j + 1)d_j \ge \rho_\ell d_\ell$.

The first of these implies that $\sigma_j d_j \leq (\sigma_\ell + 1)d_\ell$ and the second implies that $(\sigma_i + 1)d_i \geq \sigma_\ell d_\ell$. Thus, σ is a Nash assignment. Finally,

$$\begin{aligned} C(\overline{\sigma}) - C(\overline{\rho}) &= (\sigma_i^2 - \rho_i^2)d_i + (\sigma_j^2 - \rho_j^2)d_j \\ &= ((n_i+1)^2 - (n_i+2)^2)d_i + (n_j^2 - (n_j-1)^2)d_j \\ &= (2n_j - 1)d_j - (2n_i + 3)d_i \\ &= n_j d_j + (n_j - 1)d_j - (n_i + 2)d_i - (n_i + 1)d_i \\ &= (n_j - 1)d_j - (n_i + 1)d_i \\ &\leq n_j d_j - (n_i + 1)d_i \\ &\leq 0, \end{aligned}$$

since $\overline{n}$ is a Nash assignment. Note that we use Equation (20) in the last equality.

From the three cases together we see that the algorithm **FindOptNash** indeed finds an optimal Nash assignment. □

4.2 Comparison of Optimal and Nash Costs

Our first result shows that even for identical tasks the minimum cost of a Nash assignment can be larger then the optimal cost.

Lemma 4.6 *With identical task weights, for all $\epsilon > 0$ there is an instance for which*

$$\min_{N \in \mathcal{N}} C(N) \geq (\frac{4}{3} - \epsilon) \min_{A \in \mathcal{A}} C(A).$$

Proof: Consider the instance with $m = 2$, $d_1 = 1/2$, $d_2 = (1 + \frac{\epsilon}{4})$, $n = 2$, $w_1 = 1$ and $w_2 = 1$. There are three assignments. The assignment $\overline{n} = \langle 2, 0 \rangle$ has $L(1, \overline{n}) = 1$ and $C(\overline{n}) = 2$. This assignment is a Nash assignment, because moving one of the tasks to resource 2 would give it a new load of $1 + \frac{\epsilon}{4}$. The assignment $\overline{\rho} = \langle 1, 1 \rangle$ has $L(1, \overline{\rho}) = (1/2)$, $L(2, \overline{\rho}) = 1 + \frac{\epsilon}{4}$ and $C(\overline{\rho}) = 1.5 + \frac{\epsilon}{4}$. This assignment is not a Nash assignment, because the task on resource 2 could move to resource 1 for a new load of 1. Finally, the assignment $\overline{\psi} = \langle 0, 2 \rangle$ has $L(2, \overline{\psi}) = 2(1 + \frac{\epsilon}{4})$. It is not a Nash assignment, because either task could move to resource 1 for a new load of $1/2$. Thus, $\overline{n}$ is the only member of $\mathcal{N}$ and $C(\overline{n}) \geq (2/(1.5 + \frac{\epsilon}{4})) \min_{A \in \mathcal{A}} C(A) \geq (\frac{4}{3} - \epsilon) \min_{A \in \mathcal{A}} C(A)$. □

In the example from the proof of Lemma 4.6 there is only one Nash assignment, and its cost is almost $4/3$ times the cost of the best assignment. If we do the same construction with $\epsilon = 0$, we obtain an instance with two different Nash equilibria that differ in cost from each other by a factor $4/3$. The following theorem shows that $4/3$ is in fact the largest ratio obtainable between alternative Nash equilibria for any problem instance where task weights are identical.

Theorem 4.7 *Suppose the tasks weights are identical. For the ratio between the lowest-cost and the highest-cost Nash assignments we have*

$$\max_{N \in \mathcal{N}} C(N) \leq \frac{4}{3} \min_{N \in \mathcal{N}} C(N).$$

Proof: Suppose that $\overline{n}$ and $\overline{\rho}$ are distinct assignments in $\mathcal{N}(T,R)$. Suppose that ℓ is a resource for which $n_\ell > \rho_\ell$. By Lemma 4.3, $n_\ell = \rho_\ell + 1$. Also, there is a resource ℓ' for which $n_{\ell'} < \rho_{\ell'}$. Again, by Lemma 4.3, $n_{\ell'} + 1 = \rho_{\ell'}$. We will show that

$$\rho_\ell^2 d_\ell + \rho_{\ell'}^2 d_{\ell'} \leq \frac{4}{3}\left(n_\ell^2 d_\ell + n_{\ell'}^2 d_{\ell'}\right), \tag{21}$$

which proves the theorem since the resources on which $\overline{n}$ and $\overline{\rho}$ differ can be partitioned into pairs such as the pair ℓ, ℓ'.

Now

$$\rho_\ell^2 d_\ell + \rho_{\ell'}^2 d_{\ell'} = (n_\ell - 1)^2 d_\ell + (n_{\ell'} + 1)^2 d_{\ell'}. \tag{22}$$

Since $\overline{n}$ is a Nash assignment, $d_\ell n_\ell \leq d_{\ell'}(n_{\ell'} + 1) = d_{\ell'}\rho_{\ell'}$ and since $\overline{\rho}$ is a Nash assignment, $d_{\ell'}\rho_{\ell'} \leq d_\ell(\rho_\ell + 1) = d_\ell n_\ell$ so $d_\ell n_\ell = d_{\ell'}\rho_{\ell'}$. Now if $n_{\ell'} = 0$ then the right-hand side of (22) is

$$(n_\ell - 1)^2 d_\ell + (n_{\ell'} + 1) d_{\ell'} = (n_\ell - 1)^2 d_\ell + \rho_{\ell'} d_{\ell'} = (n_\ell - 1)^2 d_\ell + n_\ell d_\ell \leq n_\ell^2 d_\ell,$$

so (21) holds. So suppose that $n_{\ell'} \geq 1$. Note that, for any $A \geq 1$, the right-hand side of (22) is at most

$$A\left((n_\ell - 1) n_\ell d_\ell + \frac{n_{\ell'} + 1}{A}(n_{\ell'} + 1) d_{\ell'}.\right).$$

We will choose $A = (n_{\ell'}^2 + 2n_{\ell'} + 1)/(n_{\ell'}^2 + n_{\ell'} + 1)$ so $(n_{\ell'} + 1)/A = 1 + n_{\ell'}^2/(n_{\ell'} + 1)$. Plugging this in, we get that the right-hand side of (22) is at most

$$A\left((n_\ell - 1) n_\ell d_\ell + (n_{\ell'} + 1) d_{\ell'} + n_{\ell'}^2 d_{\ell'}\right) = A\left((n_\ell - 1) n_\ell d_\ell + n_\ell d_\ell + n_{\ell'}^2 d_{\ell'}\right) = A\left(n_\ell^2 d_\ell + + n_{\ell'}^2 d_{\ell'}\right).$$

Equation (21) follows from the observation that $A \leq 4/3$ for every $n_{\ell'} \geq 1$. $\square$

5 Finding Optima with Dynamic Programming

In [6], the authors present a polynomial time greedy algorithm for computing a Nash assignment for the L_∞ cost function. The algorithm works as follows. It considers each of the tasks in the order of non-increasing weights and assigns them to the resource that minimized their delay.

In this last section we give dynamic programming algorithms that find minimum-cost assignments for the various special cases that we have studied. These algorithms extend from the identical tasks (respectively, identical resources) case to the case where there are $O(1)$ distinct values that may be taken by the task weights (respectively, resource delays). The algorithms extend to give approximation schemes for the case where there is a $O(1)$ bound on the ratio between the largest and smallest task weights (respectively, largest to smallest delays), as studied in Theorem 2.5.

Lemma 5.1 *There exists an optimal assignment in which the set R of resources can be ordered in such a way that if $i \in R$ precedes $j \in R$, then all tasks assigned to i have weight less than or equal to all tasks assigned to j.*

Proof: Suppose that we have an assignment A where the resources cannot be ordered in this way. Then there exist two resources i and j, with two tasks assigned to i having weights w and w', and a task assigned to j with weight w'', such that $w < w'' < w'$.

Let n_i and n_j be the numbers of tasks assigned to i and j respectively, and let d_i and d_j be their delays. Let $W_i = L(i, A)/d_i$ and $W_j = L(j, A)/d_j$. The total cost of tasks assigned to i and j is $W_i n_i d_i + W_j n_j d_j$.

If $n_i d_i > n_j d_j$ then we may exchange the tasks with weights w'' and w' to reduce the social cost $C(A)$ (the operation reduces W_i by $w' - w''$ and increases W_j by $w' - w''$). If $n_i d_i < n_j d_j$ then we may exchange the tasks with weights w and w'' to reduce $C(A)$. In these cases A is suboptimal. If $n_i d_i = n_j d_j$ we may make either exchange and leave the social cost unchanged. □

Theorem 5.2 *Suppose that m resources have unit delay. Then an optimal assignment of n tasks with arbitrary weights to those resources may be found in time $O(n^2 m)$.*

Proof: We may order the task weights so that $w_1 \geq w_2 \geq \ldots w_n$. Let $C_{j,k}$ be the cost of an optimal assignment of tasks with weights $w_1, \ldots, w_j$ to resources $r_1, \ldots, r_k$. We want to compute the quantity $C_{n,m}$.

Lemma 5.1 guarantees an optimal assignment of the tasks to a set of resources that will assign the ℓ lowest-weight tasks to some resource, for some value of ℓ. $C_{j,k}$ may be found by, for each $\ell \in \{1, 2, \ldots, j\}$, assign tasks with weights $w_{j+1-\ell}, \ldots, w_j$ to resource r_k.

$$C_{j,k} = \min_{\ell \in \{0,1,2,\ldots,j\}} \Big(C_{j-\ell,k-1} + \ell \cdot (w_{j+1-\ell} + \ldots + w_j) \Big)$$

$C_{n,m}$ can be found using a dynamic programming table of size $O(nm)$ each of whose entries is computed in time $O(n)$. □

The above dynamic programming extends to the case where delays may belong to a set of $O(1)$ elements $\{d_1, \ldots, d_\alpha\}$ where α is a constant. Let m_ℓ be the number of resources with delay d_ℓ, so that $m = m_1 + \ldots + m_\alpha$.

Let $C_{j,k_1,k_2,\ldots k_\alpha}$ be the cost of an optimal assignment of tasks with weights $w_1, \ldots, w_j$ to a set of resources containing k_ℓ resources with delay d_ℓ, for $\ell = 1, 2, \ldots, \alpha$. Lemma 5.1 guarantees an optimal assignment that will (for some ℓ and ℓ') assign the ℓ lowest weight tasks to some resource with delay $d_{\ell'}$, provided $k_{\ell'} > 0$.

$$C_{j,k_1,k_2,\ldots k_\alpha} = \min_{\ell \in \{0,1,2,\ldots,j\}; \ell' \in \{1,2,\ldots,\alpha\}} \Big(C_{j-\ell,k_1,k_2,\ldots,k_{\ell'}-1,\ldots,k_\alpha} + \ell \cdot d_{\ell'} \cdot (w_{j-k} + \ldots + w_j) \Big)$$

The dynamic programming table has size $O(nm^\alpha)$ and each entry is computed in time $O(n)$.

The following theorem generalises the algorithm FindOpt to the case where there is an $O(1)$ bound on the number of distinct values taken by task weights.

Theorem 5.3 *Let weights $w_1, \ldots, w_n$ take values in $\{w'_1, \ldots, w'_\alpha\}$. Let n_ℓ be the number of tasks with weight w'_ℓ, so that $n = n_1 + \ldots + n_\alpha$. Given delays $d_1 \leq d_2 \leq \ldots d_m$, we may find an optimal assignment in time $O(mn^{2\alpha})$.*

Proof: Let $C_{k,j_1,\ldots,j_\alpha}$ be the cost of an optimal assignment to resources with delays $d_1,\ldots,d_k$ of a set of tasks containing j_ℓ tasks of weight w'_ℓ, for $\ell = 1,2,\ldots,\alpha$. For $x \in \mathbf{N}$, let $[x]$ denote the set $\{0,1,2,\ldots,x\}$.

$$C_{k,j_1,j_2,\ldots,j_\alpha} = \min_{j'_1\in[j_1];j'_2\in[j_2];\ldots;j'_\alpha\in[j_\alpha]} \Big(C_{k-1,j_1-j'_1,\ldots,j_\alpha-j'_\alpha} + (j'_1+\ldots+j'_\alpha)\cdot d_k\cdot(w'_1\cdot j'_1+\ldots+w'_\alpha\cdot j'_\alpha)\Big)$$

There are $O(mn^\alpha)$ entries in the dynamic programming table, and each entry is computed in time $O(n^\alpha)$. $\square$

The above algorithm can be used to obtain an approximation scheme for the case where there is a bound on the ratio of maximum to minimum weights, as studied in Theorem 2.5. Assume the weights are indexed in non-ascending order, $w_1 \geq w_2 \geq \ldots \geq w_n$ and the ratio w_1/w_n is upper-bounded by some pre-set limit α.

Let ϵ be the desired accuracy. Choose k such that $(w_1/w_n)^{1/k} \leq 1+\epsilon$. Take each weight and round it up to the nearest value of $w_n\cdot(w_1/w_n)^{t/k}$ where t is as small as possible in $\{0,\ldots,k\}$. The new weights take $k+1$ distinct values. An optimal assignment for the new weights has cost at most $1+\epsilon$ times the cost of an optimal assignment for the old weights, since each weight has increased by at most a factor $1+\epsilon$. In this special case of fixed ratio of largest to smallest task weight, k depends only on ϵ, and the resulting algorithm has run time $O(mn^{2k})$.

The other dynamic programming algorithm can be used in exactly the same way to obtain an approximation scheme subject to a fixed limit on the ratio of largest to smallest delay. The details are omitted.

References

[1] A. Czumaj and B. Vöcking. Tight Bounds for Worst-Case Equilibria. *Proc. 13th Annual Symposium on Discrete Algorithms*, SIAM, Philadelphia, PA (2002), pp. 413–420.

[2] A. Czumaj, P. Krysta and B. Vöcking. Selfish Traffic Allocation for Server Farms. *Proc. 34th Annual Symposium on Theory of Computing (STOC)*, Montreal, Canada (2002), pp. 287–296.

[3] U. Endriss, N. Maudet, F. Sadri and F. Toni. On Optimal Outcomes of Negotiations over Resources. *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Melbourne, Australia (2003), pp. 177–184.

[4] R. Feldmann, M. Gairing, T. Lücking, B. Monien and M. Rode. Nashification and the Coordination Ratio for a Selfish Routing Game. *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP)*, Eindhoven, Netherlands (2003), pp. 514–526.

[5] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien and P. Spirakis. Extreme Nash Equilibria. To appear in *Proc. 8th Italian Conference on Theoretical Computer Science (ICTCS)*, (2003).

[6] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas and P. Spirakis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. *Proc. 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, Malaga, Spain (2002), pp. 123–134.

[7] E. Koutsoupias and C.H. Papadimitriou. Worst-Case Equilibria. *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Trier, Germany (1999), pp. 404–413.

[8] T. Lücking, M. Mavronicolas, B. Monien, M. Rode, P. Spirakis and I. Vrto. Which is the Worst-case Nash Equilibrium? To appear in *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, (2003).

[9] M. Mavronicolas and P. Spirakis. The Price of Selfish Routing. *Proc. 33rd Annual Symposium on Theory of Computing (STOC)*, Crete, Greece (2001), pp. 510–519.

[10] P. McBurney, S. Parsons and M. Wooldridge. Desiderata for Argumentation Protocols. *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Bologna, Italy (2002), pp. 402–409.

[11] T. Roughgarden and É. Tardos. How Bad is Selfish Routing? *Journal of the ACM* **49(2)** (2002), pp. 236–259.

[12] T. Roughgarden. Many papers studying the cost of selfish routing in the flow-model are available at Tim Roughgarden's web page `http://www.cs.cornell.edu/timr/` (which also has information and summaries).

[13] T.W. Sandholm. Contract Types for Satisficing Task Allocation: I Theoretical Results. *AAAI Spring Symposium: Satisficing Models*, (1998).