Alcom-FT Technical Report Series
ALCOMFT-TR-03-147

Nashification and the Coordination Ratio for a Selfish Routing Game ⋆

R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode⋆⋆

Department of Computer Science, Electrical Engineering and Mathematics
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany
{obelix,gairing,luck,bm,rode}@uni-paderborn.de

Abstract. We study the problem of n users selfishly routing traffic through a network consisting of m parallel related links. Users route their traffic by choosing private probability distributions over the links with the aim of minimizing their private latency. In such an environment Nash equilibria represent stable states of the system: no user can improve its private latency by unilaterally changing its strategy.
Nashification is the problem of converting any given non-equilibrium routing into a Nash equilibrium without increasing the social cost. Our first result is an $O(nm^2)$ time algorithm for Nashification. This algorithm can be used in combination with any approximation algorithm for the routing problem to compute a Nash equilibrium of the same quality. In particular, this approach yields a PTAS for the computation of a best Nash equilibrium. Furthermore, we prove a lower bound of $\Omega(2^{\sqrt{n}})$ and an upper bound of $O(2^n)$ for the number of greedy selfish steps for identical link capacities in the worst case.
In the second part of the paper we introduce a new structural parameter which allows us to slightly improve the upper bound on the coordination ratio for pure Nash equilibria in [3]. The new bound holds for the individual coordination ratio and is asymptotically tight. Additionally, we prove that the known upper bound of $\frac{1+\sqrt{4m-3}}{2}$ on the coordination ratio for pure Nash equilibria also holds for the individual coordination ratio in case of mixed Nash equilibria, and we determine the range of m for which this bound is tight.

1 Introduction

Motivation-Framework. We study a routing problem in a communication network where n sources of traffic, called users, are going to route their traffic through a shared network. Traffic is routed through links of the network at a certain rate depending on the link, and different users may have different objectives, e.g. speed, quality of service, etc. The users choose routing strategies in order to minimize their private costs in terms of their private objectives

⋆ Partly supported by the DFG-SFB 376 and by the IST Program of the EU under contract numbers IST-1999-14186 (ALCOM-FT), and IST-2001-33116 (FLAGS).
⋆⋆ International Graduate School of Dynamic Intelligent Systems

without cooperating with other users. Such networks are called non-cooperative networks [10]. A famous example of such a network is the internet. Motivated by non-cooperative systems like the internet, combining ideas from game theory and computer science has become increasingly important [4, 9, 15, 16, 18].

Such an environment, which lacks a central control unit due to its size or operational mode, can be modeled as a non-cooperative game [17]. Users selfishly choose their private strategies, which in our environment correspond to probability distributions over the paths from their sources to their destinations. When routing their traffic according to the probability distributions chosen, the users will experience an expected latency caused by the traffic of all users sharing edges. Each user tries to minimize its expected individual latency without taking the global performance of the whole network into account. The theory of Nash equilibria [14] provides us with an important solution concept for environments of this kind: a Nash equilibrium is a state of the system such that no user can decrease its individual cost by unilaterally changing its strategy.

The concept of Nash equilibria has become an important mathematical tool in analyzing the behavior of selfish users in non-cooperative systems [18]. Many algorithms have been developed to compute a Nash equilibrium in a general game (see [13] for an overview). The computational complexity of computing a Nash equilibrium in general games is open [18]. The problem becomes even more challenging when global objective functions have to be optimized over the set of all Nash equilibria.

In this work, we concentrate on a special non-cooperative network consisting of a single source and a single destination which are connected by m parallel related links of capacities $c_1, \ldots, c_m$. Users $1, \ldots, n$ are going to selfishly route their traffics $w_1, \ldots, w_n$ from the source to the destination. This model has been introduced by Koutsoupias and Papadimitriou [11]. The individual cost of a user is defined as the maximum expected latency of any link it has chosen with positive probability. Depending on how the latency of a link is defined we distinguish between three variations of the model: In the identical link model all links have equal capacity. In the model of related links the latency for a link j is defined to be the quotient of the sum of the traffics through j and the capacity c_j. In the general case of unrelated links a traffic i induces load w_{ij} on link j. In this work we concentrate on the models of related and identical links.

In our model the social cost is defined to be the expected maximum latency on a link, where the expectation is taken over all random choices of the users. It is well known that, due to the lack of coordination, the users may get to a solution, i.e. a Nash equilibrium, that is suboptimal in terms of the social cost. Koutsoupias and Papadimitriou [11] defined the coordination ratio as the ratio of the social cost of a worst Nash equilibrium and the social cost of the global optimal solution. Results on the coordination ratio depend on the definition of the individual cost and the social cost. A model which uses the sum of the edge latencies as a cost function was considered by Roughgarden and Tardos [19].

In the case that the users are not allowed to randomize their strategies, the set of solutions of the routing problem consists of the set of all pure Nash equilibria. When restricted to pure strategies, the problem of computing a routing (not necessarily an equilibrium one) with minimum social cost is equivalent to the problem of scheduling n independent jobs on m related parallel machines with minimum makespan [7]. In this environment the problem of Nashification becomes important. The problem of Nashification is to compute an equilibrium routing from a given non-equilibrium one without increasing the social cost. An efficient algorithm for the Nashification problem allows to compute a Nash equilibrium with low social cost by first computing an appropriate non-equilibrium routing with known algorithms for the scheduling problem and then converting this routing into a Nash equilibrium. Here, the intention to centrally nashify a non-equilibrium solution is to provide a routing from which no user has an incentive to deviate.

One way to nashify an assignment is to perform a sequence of greedy selfish steps. A greedy selfish step is a user's change of its current pure strategy to its best pure strategy with respect to the current strategies of all other users. Any sequence of greedy selfish steps leads to a pure Nash equilibrium. However, the length of such a sequence may be exponential in n.

Related work. The selfish routing problem considered in this paper was first introduced by Koutsoupias and Papadimitriou in [11]. The problem was later studied by Mavronicolas and Spirakis [12], who introduced and analyzed fully mixed equilibria of the problem. These works were aimed at analyzing the coordination ratio of the routing game. Czumaj and Vöcking [3] gave two upper bounds of $\Gamma^{-1}(m)+1 = O(\frac{\log m}{\log\log m})$ and $O(\log \frac{c_{max}}{c_{min}})$, respectively, for the coordination ratio when restricted to pure Nash equilibria and showed that these bounds are tight up to a constant factor. For mixed Nash equilibria they showed an upper bound of $O(\frac{\log m}{\log\log\log m})$ for the coordination ratio.

It has been shown by Fotakis et al. [6] that in our model a pure Nash equilibrium can be computed in polynomial time. In the same work it was proved that the problem of computing a pure Nash equilibrium with minimum (or maximum, respectively) social cost is NP-hard.

In Gairing et al. [7] it was shown that it is NP-hard to decide whether a given routing can be transformed into an equilibrium in k greedy selfish steps, even if the number of links is 2. In the same work a polynomial time algorithm was given which, in the case of identical capacities, nashifies any non-equilibrium assignment. For identical link capacities it was shown that a PTAS exists for approximating the best social cost of a Nash equilibrium within a factor of $1+\varepsilon$.

The routing problem considered in this paper is equivalent to the multiprocessor scheduling problem. Here, pure Nash equilibria and Nashification translate to local optima and sequences of local improvements. A schedule is said to be *jump optimal* if no job on a processor with maximum load can improve by moving to another processor [20]. Obviously, the set of pure Nash equilibria is a subset of the set of jump optimal schedules. Thus, the strict upper bound of $\frac{1+\sqrt{4m-3}}{2}$ on the ratio between best and worst makespan of jump optimal schedules [2, 20] also holds for pure Nash equilibria.

In the model of identical processors every jump optimal schedule can be transformed into a pure Nash equilibrium without changing the makespan. Algorithms for computing a jump optimal schedule on identical processors from any given schedule have been proposed in [1, 5, 20]. The fastest algorithm is given by Schuurman and Vredeveld [20]. However, in all algorithms the resulting jump optimal schedule is not necessarily a Nash equilibrium.

Results. In the first part of this work we study the problem of Nashification. Given any pure routing, the goal is to compute a Nash equilibrium with less or equal social cost. We present an $O(nm^2)$ time algorithm which nashifies any pure routing in the model of related link capacities, generalizing the result of Gairing et al. [7]. The routing problem considered here is equivalent to the scheduling problem for related machines. As an immediate consequence of our result, we get a PTAS for computing a Nash equilibrium with minimum social cost by applying the PTAS of Hochbaum and Shmoys [8] to the scheduling problem and nashifying the schedule. Moreover, our algorithm efficiently computes a jump optimal schedule in the model of related processors.

One approach to nashify a routing would be to let the users, in some order, make greedy selfish steps until a Nash equilibrium is reached. We prove that for our routing problem there exists an instance of size polynomial in n such that the maximum length of a sequence of greedy selfish steps is at least $\Omega(2^{\sqrt{n}})$. This result is followed by an $O(2^n)$ upper bound for the length of any sequence of greedy selfish steps in the model of identical capacities. As a consequence we have shown that nashifying a solution using the above mentioned naive approach may take time exponential in n.

Czumaj and Vöcking [3] consider upper bounds on the maximum expected load Λ of any mixed Nash equilibrium in order to get bounds on the coordination ratio. Their two bounds on Λ depend on the number of links m and the fraction of the largest and the smallest link capacity, respectively. However, not only the capacities, but the relation between the sizes of the traffics and capacities determine the individual coordination ratio. We introduce a new structural parameter p that considers the relation between the largest traffic of a user and the capacities of the links. We denote by p the fraction of the sum of all link capacities of links, to which the largest traffic can be assigned causing latency at most the maximum latency $\mathsf{OPT}(w)$ of an optimal assignment.

In the last part of the paper, using the parameter p and similar techniques as in [3], we show the upper bound $\Gamma^{-1}(\frac{1}{p})$ on the coordination ratio for pure Nash equilibria, which is asymptotically tight for all p. Here, Γ^{-1} is the inverse of the Gamma function. Since $p \geq \frac{1}{m}$, our result also shows an asymptotically tight upper bound of $\Gamma^{-1}(m)$ for the coordination ratio, which is a slight improvement of the result in [3]. We prove our results for the individual coordination ratio, that is, the ratio between the maximum expected individual cost $\mathsf{IC}(w, P)$ and the social cost of a globally optimal solution $\mathsf{OPT}(w)$. For every Nash equilibrium P, $\mathsf{IC}(w, P)$ is at most the social cost $\mathsf{SC}(w, P)$, which is defined to be the expected maximum latency. $\mathsf{SC}(w, P)$ equals $\mathsf{IC}(w, P)$ if P is a pure Nash equilibrium. Additionally, we prove an upper bound of $\mathsf{IC}(w, P) \leq \frac{1+\sqrt{4m-3}}{2} \cdot \mathsf{OPT}(w)$. For small m, namely $m \leq 19$, this bound improves on the $\Gamma^{-1}(\frac{1}{p})$ bound, and for $m \leq 5$ it is tight.

2 Notation

Mathematical Preliminaries. For an integer $i \geq 1$, denote $[i] = \{1, \ldots, i\}$. Denote Γ the *Gamma function*; that is, for any natural number i, $\Gamma(i+1) = i!$, while for any arbitrary real number $x > 0$, $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. We will use the fact that $\Gamma(x+1) = x \cdot \Gamma(x)$. The Gamma function is invertible; both Γ and its inverse Γ^{-1} are increasing.

General. We consider a *network* consisting of a set of m parallel *links* $1, 2, \ldots, m$ from a *source* node to a *destination* node. Each of n *network users* $1, 2, \ldots, n$, or *users* for short, wishes to route a particular amount of traffic along a (non-fixed) link from source to destination. Denote w_i the *traffic* of user $i \in [n]$. Define the $n \times 1$ *traffic vector* w in the natural way. Assume, without loss of generality, that $w_1 \geq w_2 \geq \ldots \geq w_n$, and denote $W = \sum_{i=1}^{n} w_i$ the *total traffic*. A *pure strategy* for user $i \in [n]$ is some specific link. A *mixed strategy* for user $i \in [n]$ is a probability distribution over pure strategies; thus, a mixed strategy is a probability distribution over the set of links. A *pure strategy profile* L is represented by an n-tuple $\langle l_1, l_2, \ldots, l_n \rangle \in [m]^n$; a *mixed strategy profile* P is represented by an $n \times m$ *probability matrix* of nm probabilities p_{ij}, $i \in [n]$ and $j \in [m]$, where p_{ij} is the probability that user i chooses link j. The *support* of the mixed strategy for user $i \in [n]$, denoted *support*(i), is the set of those pure strategies (links) to which i assigns positive probability; so, *support*$(i) = \{j \in [m] \mid p_{ij} > 0\}$. For pure strategies we denote $link(i) = l_i$.

System, Models and Cost Measures. Denote $c_j > 0$ the *capacity* of link $j \in [m]$, representing the rate at which the link processes traffic. In the model of *identical capacities,* all link capacities are equal. Link capacities may vary arbitrarily in the model of *related capacities.* Without loss of generality assume $c_1 \geq \ldots \geq c_m$, and denote $C = \sum_{j=1}^{m} c_j$ the *total capacity.* So, the *latency* for traffic w_i through link j equals $\frac{w_i}{c_j}$. Let P be an arbitrary mixed strategy profile. The *expected latency* of user i on link j is $\lambda_{ij} = \frac{w_i + \sum_{k \in [n], k \neq i} p_{kj} w_k}{c_j}$. The *minimum expected latency* of user i is $\lambda_i = \min_{j \in [m]} \lambda_{ij}$. Denote $\mathsf{IC}(w, P)$ the *maximum expected individual latency*, that is, the maximum, over all users, of the minimum expected latency. Thus, $\mathsf{IC}(w, P) = \max_{i \in [n]} \lambda_i$. The *expected traffic* on link j is defined by $\delta_j = \sum_{i \in [n]} p_{ij} w_i$. We denote the *expected traffic on link* j *without user* i by $\tau_{ij} = \sum_{k \in [n], k \neq i} p_{kj} w_k = \delta_j - p_{ij} w_i$. The *expected load* Λ_j on link j is the ratio between the expected traffic on link j and the capacity of link j. Thus, $\Lambda_j = \frac{\delta_j}{c_j}$. The *maximum expected load* $\Lambda = \max_{j \in [m]} \Lambda_j$ is the maximum (over all links) of the expected load Λ_j on a link j. Associated with a traffic vector w and a mixed strategy profile P is the *social cost* [11, Section 2], denoted $\mathsf{SC}(w, P)$, which is the expected maximum latency on a link, where the expectation is taken over all random choices of the users. Thus,

$$\mathsf{SC}(w, P) = \sum_{\langle l_1, l_2, \ldots, l_n \rangle \in [m]^n} \left(\prod_{k=1}^{n} p_{k l_k} \cdot \max_{j \in [m]} \frac{\sum_{k: l_k = j} w_k}{c_j} \right).$$

Note that $\mathsf{SC}(w, P)$ reduces to the maximum latency through a link in the case of pure strategies. On the other hand, the *social optimum* [11, Section 2] associated with a traffic vector w, denoted $\mathsf{OPT}(w)$, is the *least possible* maximum (over all links) latency through a link; thus,

$$\mathsf{OPT}(w) = \min_{\langle l_1, l_2, \ldots, l_n \rangle \in [m]^n} \max_{j \in [m]} \frac{\sum_{k: l_k = j} w_k}{c_j}.$$

Nash Equilibria and Coordination Ratio. Say that a user $i \in [n]$ is *satisfied for the probability matrix* P if $\lambda_{ij} = \lambda_i$ for all links $j \in support(i)$, and $\lambda_{ij} \geq \lambda_i$ for all $j \notin support(i)$. Otherwise, user i is *unsatisfied.* Thus, a satisfied user has no incentive to unilaterally deviate from its mixed strategy. P is a *Nash equilibrium* [11, Section 2] iff all users $i \in [n]$ are satisfied for P.

Fix any traffic vector w. A *best (worst) Nash equilibrium* is a Nash equilibrium that minimizes (maximizes) $\mathsf{SC}(w, P)$. The *best social cost* is the social cost of a best Nash equilibrium and equals $\mathsf{OPT}(w)$. The *worst social cost* is the social cost of a worst Nash equilibrium and is denoted by $\mathsf{WC}(w)$. Fotakis et al. [6, Theorem 1] consider sequences of selfish steps starting from any arbitrary pure strategy profile. In a *selfish step,* exactly one unsatisfied user is allowed to change its pure strategy. A selfish step is a *greedy selfish step* if the user chooses its best strategy. Selfish steps do not increase the social cost of the initial pure strategy profile. The *coordination ratio* [11] is the *maximum* of $\mathsf{WC}(w)/\mathsf{OPT}(w)$, over all traffic vectors w. Correspondingly, we denote the maximum of $\mathsf{IC}(w, P)/\mathsf{OPT}(w)$ the *individual coordination ratio.*

3 Nashification

In this section, we consider the problem of converting a given pure strategy profile on related links into a Nash equilibrium without increasing the social cost. Every sequence of (greedy) selfish steps yields a Nash equilibrium eventually. However, in Section 3.2 we show that this approach can lead to an exponential number of steps, even on identical links. We present an algorithm which nashifies any pure routing by performing a polynomial number of (not necessarily selfish) moves without increasing the maximum latency.

3.1 A polynomial time algorithm for Nashification

Figure 1 shows the algorithm Nashify which converts a pure strategy profile into a Nash equilibrium. A crucial observation for proving the correctness of the algorithm is stated in the following lemma:

Lemma 1. *If user i with traffic w_i performs a greedy selfish step from link j to link k with $c_j \leq c_k$, then no user s with traffic $w_s \geq w_i$ becomes unsatisfied.*

Proof. Let user s be located on link $q = link(s)$. Since only the loads on link j and k change due to the greedy selfish step of user i we have to show that user s cannot improve by moving to link j. Also we have to show that, if user s is located on link k, s does not become unsatisfied due to the arrival of user

i. Assume first, $q \neq k$. As user s is satisfied, $\frac{\delta_k + w_s}{c_k} \geq \frac{\delta_q}{c_q}$. User i improves by moving to link k, thus $\frac{\delta_j}{c_j} > \frac{\delta_k + w_i}{c_k}$, and we can estimate

$$\frac{\delta_j - w_i + w_s}{c_j} > \frac{\delta_k + w_i}{c_k} + \frac{w_s - w_i}{c_j} \geq \frac{\delta_q}{c_q} - \frac{w_s}{c_k} + \frac{w_i}{c_k} + \frac{w_s - w_i}{c_j}$$
$$\geq \frac{\delta_q}{c_q} + (w_s - w_i)(\frac{1}{c_j} - \frac{1}{c_k}) \geq \frac{\delta_q}{c_q}.$$

The last inequality holds since $c_k \geq c_j$ and $w_s \geq w_i$. Thus, s cannot improve by moving to link j after i moved. It remains to prove that user s cannot become unsatisfied, if $q = k$. Because of $\frac{\delta_j - w_i + w_s}{c_j} \geq \frac{\delta_j}{c_j} > \frac{\delta_k + w_i}{c_k}$, user s cannot improve by moving to link j. Since user i performed a greedy selfish step, we have

$$\frac{\delta_r + w_s}{c_r} \geq \frac{\delta_r + w_i}{c_r} \geq \frac{\delta_k + w_i}{c_k} \; \forall r \in [m] \setminus \{j\},$$

and therefore user s cannot improve by moving to any link $r \neq j$. □

For identical links Lemma 1 implies that by moving a user to its best link, no user with larger or equal traffic can become unsatisfied. Thus, by successively moving each user to its best link in order of non-increasing traffic sizes, we end up in a Nash equilibrium without increasing the social cost of the initial routing. This algorithm is described in [7].

```
Nashify()
Input: n users with traffics w_1 ≥ ··· ≥ w_n
       m links with capacities c_1 ≥ ··· ≥ c_m
       Assignment of users to links
Output: Assignment of users to links with less
        or equal maximum latency, which is a NE
{
    // phase 1:
    i := n;  S := {n};
    while i ≥ 1 {
        move user i to link with highest possible index
            without increasing overall maximum latency;
        if i was moved or i ∈ S or link(i) ≤ link(i + 1)
        then S := S ∪ {i};  i := i − 1;
        else {
            move user i to link with smallest possible index
                without increasing overall maximum latency;
            if i was moved then S := S ∪ {i};  i := n;
            else break;
        }
    }
    // phase 2:
    while ∃i ∈ S {
        make greedy selfish step for user i = min(S);
        S := S\{i};
    }
}
```

Fig. 1. Algorithm Nashify

With algorithm Nashify in Figure 1 we generalize this idea to non-identical links. The algorithm works in two phases. At every time $link(i)$ denotes the link user i is currently assigned to. The main idea is to fill up slow links with users with small traffic as close to the maximum latency as possible in the first phase (but without increasing the maximum latency) and to perform greedy selfish steps for unsatisfied users in the second phase. During the first phase, the set S is used to collect all those users with small traffics, who have been used to fill up slow links. Throughout the whole algorithm, each user in S is located on a link with non-greater index than any smaller user in S. In other words, the smaller the traffic of a user in S, the slower the link it is assigned to. We may start with $S = \{n\}$, because the above property is trivially fulfilled if S contains only one user. When no further user is added to S by the algorithm, the first phase terminates. In the second phase we successively perform greedy selfish steps for all unsatisfied users, starting with the largest one. That is, we move each user that can improve by changing its link to its best link. Because of the special conditions that have been established by phase 1, and by Lemma 1, these greedy selfish steps do not cause other users with larger traffic to become unsatisfied.

Lemma 2. *After phase 1 the following holds:*

(1) All unsatisfied users are in S.
(2) $S = \{n, (n-1), \ldots, (n+1-|S|)\}$, that is, S contains the $|S|$ users with smallest traffics.
(3) $i, i+1 \in S \Rightarrow link(i) \leq link(i+1)$.
(4) Every user $i \in S$ can only improve by moving to a link with smaller index.

Proof. The while-loop in phase 1 can only be terminated if either i becomes 0 (the while condition does not hold) or some user $i \notin S$ on a link $link(i) > link(i+1)$ cannot be moved to any other link $j < link(i)$ (the break-command is executed). In the first case all users are in S, which implies (1). In the second case we know that user $(i+1)$ does not fit on any link $j > link(i+1)$, as user $(i+1)$ was put on the link with maximal index without increasing the maximum latency in the previous run of the loop. In particular, as $link(i+1) < link(i)$, user $(i+1)$ cannot be moved to any link $j \geq link(i)$. Thus, no user $k \notin S$ would fit on any link $j \neq link(k)$, as $w_k \geq w_i \geq w_{i+1}$, $\forall k \notin S$. This again implies (1).

To see that (2) holds at any time, notice first that a user which is included in S will never be removed from S. Second, whenever a user is added, it is the user with smallest traffic which is not contained in S so far. So S is always a consecutive set of the users with smallest traffics.

(3) is an invariant which holds before and after every run of the while-loop in phase 1. Before the first run it holds because $S = \{n\}$. Whenever a user $i \in S$ is moved, it is moved to a link $j > link(i)$ with capacity $c_j \leq c_{link(i)}$. As the traffic of user $(i+1)$ is not larger than the traffic of user i, it would fit on link j, too. But user $(i+1)$ has been considered before user i in the previous run of the while-loop. Thus, user $(i+1)$ is located on some link $link(i+1) \geq j$, because otherwise it would have been moved to link j. Therefore, $link(i+1) \geq j$ and (3) remains true after moving user i to link j.

To show (4), consider the last $|S|$ runs of the while-loop, not counting the run which executes the break command (in which no user is moved). (2) implies, that these runs establish a sweep over all users in S, beginning with user n and ending up with the user having the smallest index in S. Each user $i \in S$ is moved to the link with highest index it fits on (without increasing the maximum latency). After user i is assigned, only users with larger or equal traffics are considered. They are located on links $j \leq link(i)$, which follows from (3). Thus, by moving the remaining users, the maximum latency on any link $j > link(i)$ is not decreased, which implies that user i cannot be moved to a link $j > link(i)$ after the sweep either. As this holds for all users $i \in S$, (4) is valid. □

Theorem 1. *Given any pure strategy profile, algorithm Nashify computes a Nash equilibrium with non-increased social cost, performing at most $(m+1)n$ moves in sequential running time $\mathcal{O}(m^2 n)$.*

Proof. We first prove the correctness of the algorithm Nashify. After phase 1 the conditions from Lemma 2 hold. We now show that these conditions still hold after each run of the while-loop in phase 2.
Consider any run of the while-loop and assume that the conditions of Lemma 2 hold. Let i be the user with smallest index in S, and suppose it is moved from link $j = link(i)$ to its best link k. Because of (2), we have $i = n + 1 - |S|$. (4) implies $k < link(i)$ and therefore, $c_k \geq c_j$.
Now let $s \notin S$ be any user on some link $q = link(s)$.

Due to Lemma 1, user s is satisfied after user i has been moved. Thus, (1) still holds after moving i. As i is removed from S and $i = n + 1 - |S|$, (2) still holds. As i was the user with largest traffic in S, (3) still holds. (3) and the fact that i was moved to a link $j \leq link(i)$ imply that (4) remains true after the run. At the end of the algorithm, because S is empty and condition (1) still holds, there are no unsatisfied users, i.e., we have a Nash equilibrium. As in no step of the algorithm the overall maximum latency is increased, the algorithm correctly computes a Nash equilibrium with non-increased social cost.

Now we show the bound on the running time. In phase 1, each user is shifted at most once to a link with smaller index. Afterwards it can be shifted at most $m-1$ times to a link with higher index. So we have at most m moves per user. In phase 2 we have at most $|S| \leq n$ moves. Thus, at most $mn+n$ moves are required altogether. Using appropriate data structures in phase 1, it takes time $\mathcal{O}(1)$ to determine whether a user has to be moved or not. One possibility to do this is to maintain two arrays (x_j) and (y_j) during phase 1, both containing one entry for each link. x_j is the maximal size of a user on link j that would fit on some link $k > j$ without increasing the overall maximum latency. Analogously, y_j is the maximal size of a user on link j that would fit on some link $k < j$. Certainly, $x_m = 0$ and $y_1 = 0$. For each move the algorithm may have to consider m links to find an appropriate target link. It then must update the data structures. Finding the target link and updating the data structures can be done in time $\mathcal{O}(m)$. This yields time complexity $\mathcal{O}(m^2 n)$ for phase 1. Phase 2 requires time $\mathcal{O}(mn)$. □

Combining any approximation algorithm for the computation of good routings with the Nashify algorithm yields a method for approximating the best Nash equilibrium. Particularly, using the PTAS for the Scheduling Problem from Hochbaum and Shmoys [8], we get:

Corollary 1. *There is a PTAS for computing a best pure Nash equilibrium.*

This is optimal in the sense that the development of an FPTAS is not possible since the exact computation of the best Nash equilibrium is $\mathcal{NP}$-complete in the strong sense [6].

Remark 1. Apart from the routing model with related links as considered here, the algorithm can also cope with a slightly relaxed setting. All we need is an order of the users and links, such that $\forall i \in [n-1], j \in [m] : w_{ij} \geq w_{i+1,j}$ and $\forall i \in [n-1], j \in [m-1] : w_{i,j+1} - w_{ij} \geq w_{i+1,j+1} - w_{i+1,j}$. Recall that w_{ij} denotes the contribution of user i to the latency on link j in the model of unrelated links. In the related link model we have the special case $w_{ij} = \frac{w_i}{c_j}$.

3.2 Sequences of greedy selfish steps

Performing greedy selfish steps will eventually convert any routing into a pure Nash equilibrium. However, this may take exponential time even if the links have identical capacities, as shown in the following two theorems. Due to lack of space the proofs are omitted here.

Theorem 2. *There exists an instance of n users with traffics whose bitlength is polynomial in n on $m = \sqrt{n+7} - 1$ identical links for which the maximum length of a sequence of greedy selfish steps is at least $2^{\sqrt{n+7}-3}$.*

Theorem 3. *For any instance with n users on identical links, the length of any sequence of greedy selfish steps is at most $2^n - 1$.*

4 Coordination ratio

In this section we introduce a structural parameter p. We denote $\mathcal{M}_1 = \{j \in [m] \mid w_1 \leq c_j \cdot \mathsf{OPT}(w)\}$ and $p = \sum_{j \in \mathcal{M}_1} c_j / C$. In other words, p is the ratio between the sum of link capacities of links to which the largest traffic can be assigned causing latency at most $\mathsf{OPT}(w)$ and the sum of all link capacities. With the help of p we are able to prove an upper bound on the individual coordination ratio.

Theorem 4. *For any mixed Nash equilibrium P the ratio between the maximum expected individual latency $\mathsf{IC}(w,P) = \max_{i \in [n]} \lambda_i$ and $\mathsf{OPT}(w)$ is bounded by*

$$\frac{\mathsf{IC}(w,P)}{\mathsf{OPT}(w)} < \begin{cases} \frac{3}{2} + \sqrt{\frac{1}{p} - \frac{3}{4}} & \text{if } \frac{1}{3} \leq p \leq 1, \\ 2 + \sqrt[3]{\frac{1}{p} - 2} & \text{if } \frac{1}{37} \leq p < \frac{1}{3}, \\ \Gamma^{-1}\left(\frac{1}{p}\right) & \text{if } p < \frac{1}{37}. \end{cases}$$

Since $\frac{w_1}{c_1} \leq \mathsf{OPT}(w)$, we have $p \geq \frac{c_1}{C} \geq \frac{1}{m}$. Furthermore, $\mathsf{IC}(w,P) \geq \Lambda$ holds for every assignment. Thus, from Theorem 4 we get the following corollaries:

Corollary 2. *The maximum expected load Λ is bounded from above by $\Lambda \leq \Gamma^{-1}\left(\frac{1}{p}\right) \mathsf{OPT}(w) \leq \Gamma^{-1}(m)\, \mathsf{OPT}(w)$.*

Corollary 3. *The individual coordination ratio $\mathsf{IC}(w,P)$ is bounded from above by $\mathsf{IC}(w,P) \leq \Gamma^{-1}(m) \mathsf{OPT}(w)$.*

Corollary 2 shows that the generalized upper bound is an improvement of the upper bound $\Gamma^{-1}(m) + 1$ on the maximum expected load Λ in [3]. This leads to an improvement of the upper bound on the coordination ratio [3, Lemma 2.1].

We now introduce a pure Nash equilibrium in Example 1. This can be used to prove that the upper bounds of $\Gamma^{-1}(\frac{1}{p})$ and $\Gamma^{-1}(m)$ are tight.

Example 1 *Let $k \in \mathbb{N}$, and consider the following instance with k different classes of users:*

- *Class U_1:* $|U_1| = k$ *users with traffic* 2^{k-1}
- *Class U_i:* $|U_i| = 2^{i-1} \cdot (k-1) \prod_{j=1,\ldots,i-1}(k-j)$ *users with traffic* 2^{k-i} *for all* $2 \leq i \leq k$.

In the same way we define $k+1$ different classes of links:

- *Class P_0:* *One link with capacity* 2^{k-1}.
- *Class P_1:* $|P_1| = |U_1| - 1$ *links with capacity* 2^{k-1}.
- *Class P_i:* $|P_i| = |U_i|$ *links with capacity* 2^{k-i} *for all* $2 \leq i \leq k$.

Consider the following assignment:

- *Class P_0:* *All users in U_1 are assigned to this link.*
- *Class P_i:* *On each link in P_i there are $2(k-i)$ users from U_{i+1}, respectively, for all $1 \leq i \leq k-1$.*
- *Class P_k:* *The links from P_k remain empty.*

The above assignment is a pure Nash equilibrium L with social cost $\mathsf{SC}(w, L) = k$ and $\mathsf{OPT}(w) = 1$.

Lemma 3. *For each $k \in \mathbb{N}$ there exists an instance with a pure Nash equilibrium L with*

$$k = \frac{\mathsf{SC}(w, L)}{\mathsf{OPT}(w)} \geq \Gamma^{-1}\left(\frac{1}{3p}\right).$$

Lemma 4. *For each $k \in \mathbb{N}$ there exists an instance with a pure Nash equilibrium L with*

$$k = \frac{\mathsf{SC}(w, L)}{\mathsf{OPT}(w)} \geq \Gamma^{-1}(m) \cdot (1 + o(1)).$$

Note that we can prove $k \geq \Gamma^{-1}(\frac{1}{p}) - 1$ in a similar way as in Lemma 3. This shows that the generalized upper bound is tight up to an additive constant for **all** m whereas due to Lemma 4, $\Gamma^{-1}(m)$ is tight only for **large** m.

We conclude this section by giving an upper bound on the maximal expected individual latency of a mixed Nash equilibrium, which depends on the number of links m. The same bound also applies to the social cost of a pure Nash equilibrium. This bound improves on Corollary 3 for small m.

Theorem 5. *For any mixed Nash equilibrium P on m links, $\mathsf{IC}(w, P)$ is bounded by $\mathsf{IC}(w, P) \leq \frac{1+\sqrt{4m-3}}{2}\mathsf{OPT}(w)$. This bound is not tight if $m \geq 6$. For $m \geq 4$, there is no pure Nash equilibrium matching the bound. For $m \geq 2$, there is no fully mixed Nash equilibrium matching the bound.*

Lemma 5. *The bound from Theorem 5 is tight for up to five links. For pure Nash equilibria, the bound is tight for up to three links.*

Theorem 5 slightly extends this result to the maximum expected individual latency $\mathsf{IC}(w, P)$ of mixed Nash equilibria. Furthermore, we have shown that the bound on $\mathsf{IC}(w, P)$ is tight if and only if $1 \leq m \leq 5$ for mixed Nash equilibria, and if and only if $1 \leq m \leq 3$ for pure Nash equilibria. The bound of $\Gamma^{-1}(m)$ from Corollary 3 is asymptotically tight, but for small numbers of links ($m \leq 19$) the bound from Theorem 5 is better.

References

1. P. Brucker, J. Hurink, and F. Werner. Improving local search heuristics for some scheduling problems. part ii. *Discrete Applied Mathematics*, 72:47–69, 1997.
2. Y. Cho and S. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.
3. A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proc. of SODA 2002*, pp 413–420, 2002.
4. J. Feigenbaum, C. Papdimitriou, and S. Shenker. Sharing the cost of multicast transmissions. In *Proc. of STOC 2000*, pp 218–227, 2000.
5. G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979.
6. D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. In *Proc. of ICALP 2002*, pp 123–134, 2002.
7. M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and P. Spirakis. Extreme nash equilibria. Technical report, FLAGS-TR-03-10, 2002.
8. D.S. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.
9. K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proc. of STOC 2001*, pp 364–372, 2001.
10. Y.A. Korilis, A.A. Lazar, and A. Orda. Architecting noncooperative networks. *IEEE Journal on Selected Areas in Communications*, 13(7):1241–1251, 1995.
11. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc. of STACS 1999*, pp 404–413, 1999.
12. M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. of STOC 2001*, pp 510–519, 2001.
13. R.D. McKelvey and A. McLennan. Computation of equilibria in finite games. In H. Amman, D. Kendrick, and J. Rust, editors, *Handbook of Computational Economics*, 1996.
14. J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.
15. N. Nisan. Algorithms for selfish agents. In *Proc. of STACS 1999*, pp 1–15, 1999.
16. N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. of STOC 1999*, pp 129–140, 1999.
17. M.J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.
18. C.H. Papadimitriou. Algorithms, games, and the internet. In *Proc. of STOC 2001*, pp 749–753, 2001.
19. T. Roughgarden and E. Tardos. How bad is selfish routing? In *Proc. of FOCS 2000*, pp 93–102, 2000.
20. P. Schuurman and T. Vredeveld. Performance guarantees of load search for multiprocessor scheduling. In *Proc. of IPCO 2001*, pp 370–382, 2001.