Analyzing Models for Scheduling and Routing

Dissertation

von

Thomas Lücking

Schriftliche Arbeit zur Erlangung des Grades
Doktor der Naturwissenschaften

an der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn.

Paderborn, 22. März 2005

Für meine Familie

Insofern sich die Sätze der Mathematik auf die Wirklichkeit beziehen,
sind sie nicht sicher,
und insofern sie sicher sind,
beziehen sie sich nicht auf die Wirklichkeit.

Albert Einstein (1879–1955)

Contents

— 1 —

Introduction

Der Anfang ist die Hälfte des Ganzen.

Aristoteles (384–322 BC)

1.1 Outline

Scheduling and *routing* are classical fields of theoretical computer science which in the last decades gained a lot of flourish attention as tools for improving the performance of large-scale computer systems. The aim of both scheduling and routing is to ensure an efficient use of such a computer system. The objective of scheduling is to determine an *optimum assignment* of *jobs* to (one or more) *processors* with respect to some performance measure, whereas the goal of routing is to efficiently ship (splittable or unsplittable) *packets* through a common processor network.

The theory of scheduling and routing is characterized by a virtually unlimited number of problem types. In this thesis, we analyze three of them. Since they are independent, we extensively study them in three self-contained chapters and only briefly introduce them here.

1.1.1 Scheduling Identical Malleable Jobs

Multiprocessor scheduling is a well-known scheduling problem which has been studied extensively in many different variations. If the number of processors on which a specific job has to be executed is part of the input, then the jobs are called *non-malleable*. Otherwise they are called *malleable*. If the jobs may be interrupted while being executed, then the resulting schedule is called *preemptive*, otherwise it is called *non-preemptive*. Furthermore, the definition of the multiprocessor scheduling problem depends on *precedence constraints* between jobs, and on the *objective*.

In Chapter 2, we consider the problem of finding a *non-preemptive* schedule for *independent malleable identical jobs* on *identical processors* with *minimum total completion time*, the

so-called *makespan*. Since the jobs are identical, the execution time on any certain number of processors is the same for all jobs. We assume that the execution of the jobs achieves some speed-up, but no super-linear speed-up. Our research is motivated by the fact that this scheduling problem arises while using the *Descartes method* to isolate real roots in parallel.

Up to now, it is not clear whether a schedule with minimum makespan in this setting can be computed in polynomial time. There only exists an algorithm which computes a schedule with minimum makespan using time polynomial in the number of jobs if the number of processors is constant. In order to approximate an optimum schedule, we introduce *phase-by-phase* schedules, consisting of phases in which each job uses the same number of processors. A new phase can not be started until the last phase has finished. We illustrate with the help of an example that the quotient of the makespan of an optimum phase-by-phase schedule and the makespan of an optimum schedule can be $\frac{5}{4}$. Furthermore, we give a *constant-time* algorithm which only uses certain phase-by-phase schedules providing an approximation factor of $\frac{5}{4}$.

1.1.2 Flow Scheduling

Load balancing is an essential task for the efficient use of parallel computer systems. In many parallel applications, the work loads have dynamic behavior and may change dramatically during runtime. In order to efficiently use the parallel computer system, the work load has to be balanced among the processors during runtime. Clearly, the balancing scheme is required to be highly efficient itself in order to ensure an overall benefit.

In Chapter 3, we consider *synchronous distributed processor networks*. In each round, a processor of the network can send and receive messages to/from all its neighbors simultaneously. Furthermore, the situation is *static*, i.e., no load is generated or consumed during the balancing process, and the network does not change. We assume that the load on the processors consists of independent load units, called *tokens*. In order to balance the network, it is necessary to migrate parts of the processors' loads during runtime. We migrate the load according to a given *balancing flow*. The goal is to use the minimum number of rounds to reach the balanced state.

We show that for every distributed scheduling strategy there exists a flow graph on which this strategy requires at least $\frac{3}{2}$ times the minimum number of rounds. Then, we present a distributed algorithm for flows in tree networks. In contrast to the known local greedy algorithms, this algorithm investigates the structure of the flow graph before sending tokens. We prove that this algorithm requires at most twice the minimum number of rounds, and we show that this bound is tight. To the best of our knowledge, this is the first distributed flow scheduling algorithm (even though for a *restricted* class of balancing flows) which is optimum up to a constant factor.

1.1.3 Selfish Routing in Non-Cooperative Networks

Large-scale traffic and communication networks, like e.g. the internet, telephone networks, or road traffic systems often lack a central regulation for several reasons: The size of the network may be too large, the networks may be dynamically evolving over time, or the users of the network may be free to act according to their private interest, without regard to the overall performance of the system. Besides the lack of central regulation even cooperation of the

users among themselves may be impossible due to the fact that the users may not even know each other. Motivated by such non-cooperative systems, combining ideas from game theory and theoretical computer science has become increasingly important. Here, the concept of Nash equilibrium has become an important mathematical tool for analyzing the behavior of selfish users.

In Chapter 4, we consider a routing game, widely known as the *KP-model*. In this model, non-cooperative users wish to route their unsplittable *traffics* through a very simple network of *parallel links* with *capacities* from source to destination. Each user is allowed to route its traffic along links from its *strategy set* and employs a mixed strategy, trying to minimize its *expected latency*. A stable state in which no user has an incentive to unilaterally change its strategy is called a *Nash equilibrium*. There is also a global objective function called *social cost*. However, users do not attend to it. The ratio of the maximum social cost of a Nash equilibrium over the minimum social cost of an assignment is called *price of anarchy* or *coordination ratio*.

We consider two different definitions of social cost, namely *makespan social cost*, defined as the *maximum expected latency*, and *polynomial social cost*, defined as the expectation of the weighted sum of a *polynomial cost function*, evaluated at the incurred link loads. We prove a multitude of interesting results on the various algorithmic, combinatorial, structural and optimality properties of Nash equilibria in the KP-model and its variations. In order to simplify the evaluation of these results, we integrate them in a thorough survey.

1.2 Publications

The results described in this thesis are published in parts as joint work in the Proceedings of the *International Colloquium on Automata, Languages, and Programming (ICALP)* [46, 58], the Proceedings of the *Italian Conference on Theoretical Computer Science (ICTCS)* [59], the Proceedings of the *International Symposium on Mathematical Foundations of Computer Science (MFCS)* [47, 57, 103, 104], the Proceedings of the *International Symposium on Theoretical Aspects of Computer Science (STACS)* [102], the Proceedings of the *Annual ACM Symposium on Theory of Computing (STOC)* [56], and the Proceedings of the *International Workshop on Approximation and Online Algorithms (WAOA)* [34].

1.3 Acknowledgments

First of all, I would like to thank Prof. Dr. Burkhard Monien for his great support. I would also like to thank Prof. Dr. Marios Mavronicolas for making my three months' stay at Cyprus possible. Furthermore, I would like to express my thanks to those people with whom I had a great collaboration. To name them in alphabetic order, these are: Robert Elsässer, Rainer Feldmann, Martin Gairing and Manuel Rode. I would also like to thank all members of the research group Monien for the good cooperation and the nice atmosphere.

Besides my colleagues, I am very grateful to my family and friends whose continuous support greatly helped me to finish this thesis.

— 2 —

Scheduling Identical Malleable Jobs

Entscheide lieber ungefähr richtig, als genau falsch.

Johann Wolfgang von Goethe (1749–1832)

2.1 Introduction

2.1.1 Motivation and Framework

The *multiprocessor scheduling problem* for identical processors is well-known and has been studied extensively in many different variations. If the number of processors on which a specific job has to be executed is part of the input, then the jobs are called *non-malleable*. Otherwise they are called *malleable*. If the jobs may be interrupted while being executed, then the resulting schedule is called *preemptive*, otherwise it is called *non-preemptive*. Furthermore, the definition of the multiprocessor scheduling problem depends on precedence constraints between jobs, and on the objective. For an overview of a multitude of models and works, we recommend the book of Brucker [16] and the paper of Veltman *et al.* [143].

We now consider the problem of finding a non-preemptive schedule for *n independent malleable identical jobs* on *m identical processors* with *minimum total completion time*, the so-called *makespan*. We assume that the same properties for the execution time as in [13] hold. This implies that the execution of the jobs achieves some speed-up, but no super-linear speed-up. Since the jobs are identical, the time function is equal for all jobs, that is, the execution time on any certain number of processors is the same for all jobs. Figure 2.1 illustrates a possible schedule for $n = 11$ jobs and $m = 10$ processors.

Using the branch & bound or the divide & conquer strategy to solve a problem, the problem is split into smaller subproblems which have to be solved, that is, jobs which have to be executed. In many cases, the parallelism given by the branch & bound tree or by the divide & conquer tree is sufficient to yield a good speed-up. However, in many other cases this is not true and we have to parallelize the computations performed at the tree nodes. All these

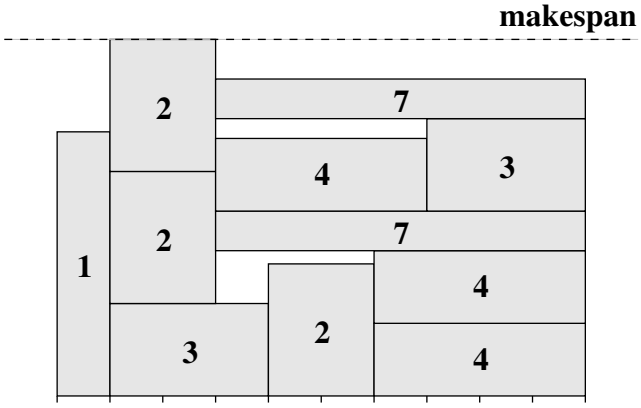


Figure 2.1: Schedule for $n = 11$ jobs and $m = 10$ processors. The vertical axis shows the time whereas the horizontal axis shows the processors (ordered by their numbers, that is, processor 1 on the left-most position and processor 10 on the right-most position). Every rectangle represents the execution of a job, where the width corresponds to the *set* of used processors, the number within the rectangle corresponds to the *number* of used processors, and the height corresponds to the used *execution time*.

computations are of the same type, so we may assume that they are identical. In this case, the scheduling problem we consider applies. Our motivation for carrying out research on this problem is that this scheduling problem arises while using the *Descartes method* to isolate real roots in parallel [33]. Here, the time function can be computed by analyzing the parallel algorithm in the LogP-*model* (see e.g. [25]).

2.1.2 Contribution

As a matter of course, reading the input and returning the output by an algorithm takes time. However, in the following, we only give time bounds on the execution time in order to simplify the readability of our results. Moreover, we assume that the manipulation of numbers can be done in constant time.

Up to now, it is not clear whether a schedule with minimum makespan in the setting which we consider here can be computed in polynomial time. Decker *et al.* [34] gave an algorithm which computes a schedule with minimum makespan with execution time exponential in the number m of processors. Though this yields an algorithm polynomial in the number n of jobs if m is constant, the algorithm is not suitable for practical purposes. In order to approximate an optimum schedule, we introduce *phase-by-phase schedules*. Here, schedules have a simple structure. They consist of phases in which each job uses the same number of processors. A new phase can not be started until the last phase has finished. Trystram [141] illustrated with the help of an example that the quotient of the makespan of an optimum phase-by-phase schedule and the makespan of an optimum schedule can be $\frac{5}{4}$. We obtain the following results:

- We give a *constant-time* algorithm (Algorithm 1, page 12) which only uses certain phase-by-phase schedules providing an approximation factor of $\frac{5}{4}$ (Theorem 2.6, page 11).

- In order to prove this approximation factor, we prove two technical lemmas, providing good lower bounds on the makespan of an optimum schedule as well as restricting the set of schedules which have to be considered (Lemmas 2.8 and 2.9, page 14).

Since there exists an ε-approximation algorithm in the case that the speed-up is optimum up to a constant factor, this indicates that approximating an optimum schedule is easy when the speed-up is near-optimum. In general, we do not know which class of instances is easy to optimize.

2.1.3 Related Work

Du and Leung [39] showed that both decision problems corresponding to the problems of scheduling non-malleable and malleable jobs on identical processors with minimum makespan are $\mathcal{NP}$-hard. So, researchers are interested in approximation algorithms. If the jobs are non-malleable, then the scheduling problem is a special case of the resource constraint scheduling problem. Hence, an optimum schedule can be approximated up to a factor of 2 using list scheduling [60].

Krishnamurti and Ma [94] were the first to study approximation algorithms for the scheduling problem with malleable jobs. Belkhale and Banerjee [13] introduced an algorithm with approximation factor of $\frac{2}{1-\frac{1}{m}}$, assuming that execution time decreases with the number of processors while the computational work increases. Turek *et al.* [142] improved this result, using no assumptions, and showed an approximation factor of 2. Using the same assumptions for the execution time as Belkhale and Banerjee [13], Blazewicz *et al.* [14] gave an approximation algorithm with performance guarantee 2, starting from the continuous version of the problem and using rounding techniques. The latest result is from Mounié *et al.* [113]. They proved that an optimum schedule for malleable jobs can be approximated up to the factor of $\sqrt{3}$, obtained by a direct constructing method. A proof for the factor of $\frac{3}{2}$ has been submitted for publication [114].

The best known approximation algorithm for schedules where jobs are only allowed to be executed on processors with successive indices is from Steinberg [140]. He showed an approximation factor of 2. This scheduling problem is closely related to the orthogonal packing problem of rectangles, first investigated by Baker *et al.* [10]. See [9, 40, 80] for a typology of cutting and packing problems, and results on approximation algorithms.

Jansen and Porkolab [77] invented the first polynomial time approximation algorithm for a constant number of processors using linear programming. This problem is related to orthogonal strip packing of rectangles [31, 79]. Furthermore, Jansen [76] proposed an asymptotic fully polynomial time approximation scheme if m is part of the input, that is, for any fixed $\varepsilon > 0$, the approximation algorithm computes a schedule with makespan at most $(1+\varepsilon)$ times the optimum (plus an additive term), using time polynomial in n, m and $\frac{1}{\varepsilon}$.

2.1.4 Organization

The rest of this chapter is organized as follows. After a formal definition of our model in Section 2.2, we introduce an algorithm to compute an optimum schedule in Section 2.3. In Section 2.4, we show how phase-by-phase schedules can be used to approximate an optimum schedule within factor $\frac{5}{4}$ using constant execution time. The proof of this result is given in

Sections 2.5 - 2.7. In Section 2.8, we give an ε-approximation algorithm in the case that the speed-up is optimum up to a constant factor. We close, in Section 2.9, with a discussion of our results and some open problems.

2.2 Model

For all $k \in \mathbb{N}$, we denote $[k] = \{1, \ldots, k\}$.

2.2.1 Instance

For the malleable scheduling problem, the input is an **instance** (n, m, t), where n is the number of **identical jobs** which can be executed on a different number of processors in parallel, m is the number of **identical processors** on which the jobs are to be scheduled, and t is the **time function** given by $t : [m] \longrightarrow \mathbb{R}^+$. Here, $t(j)$ is the running time needed to compute a job on j processors. We choose $\mathbb{R}^+$ for our theoretical analysis since it is the most general possible image. As a matter of course, all results also hold if the image of t is $\mathbb{Q}^+$. In this case, t can be encoded. For all $j_1, j_2 \in [m]$ with $j_1 \leq j_2$, the time function t must have the following properties:

- **monotonicity**: $t(j_2) \leq t(j_1)$
- **speed-up property**: $j_1 \cdot t(j_1) \leq j_2 \cdot t(j_2)$

These properties imply that the execution of the jobs may achieve some speed-up, but no super-linear speed-up.

2.2.2 Schedule

A **schedule** $\mathbf{S} = \{(\sigma_i, \tau_i) \mid i \in [n]\}$ assigns a set σ_i of processors and a starting time τ_i to every job i such that all jobs are executed and every processor executes at most one job at any time. We call (σ_i, τ_i) the **plan** for job i. Associated with an instance (n, m, t) and a schedule $\mathbf{S}$ is the **makespan**, defined by

$$T(n, m, t, \mathbf{S}) \quad = \quad \max\{\tau_i + t(|\sigma_i|) \mid (\sigma_i, \tau_i) \in \mathbf{S}\} .$$

The **optimum makespan** associated with an instance (n, m, t), denoted $T_{opt}(n, m, t)$, is the *least possible* makespan among all schedules. A schedule with optimum makespan is called an **optimum schedule**.

2.3 Optimum Schedules

Decker *et al.* [34] proposed an algorithm to compute an optimum schedule. In order to prove the correctness of this algorithm, they showed that for each schedule there exists another schedule with at most the same makespan which also exhibits additional properties. One of these properties is that each job in a schedule starts either at time $\tau = 0$ or directly subsequent to another job. We now define this property formally.

Fix any instance (n,m,t) and associated schedule $\mathbf{S}$. Denote $\Lambda_j(\mathbf{S})$ the **latency** on a processor $j \in [m]$, that is,

$$\Lambda_j(\mathbf{S}) \quad = \quad \max\{\tau_i + t(|\sigma_i|) \mid (\sigma_i,\tau_i) \in \mathbf{S} \text{ and } j \in \sigma_i\} .$$

Define the $m \times 1$ **latency vector** $\Lambda(\mathbf{S})$ in the natural way, and define the $m \times 1$ **sorted latency vector** $\widetilde{\Lambda}(\mathbf{S})$ by $\widetilde{\Lambda}_j(\mathbf{S}) = \Lambda_{\pi(j)}(\mathbf{S})$ for all $j \in [m]$, where π is a permutation with $\Lambda_{\pi(j_1)}(\mathbf{S}) \leq \Lambda_{\pi(j_2)}(\mathbf{S})$ for all $j_1, j_2 \in [m]$ with $\pi(j_1) \leq \pi(j_2)$. Note that

$$T(n,m,t,S) \quad = \quad \widetilde{\Lambda}_m(\mathbf{S}) .$$

The schedule $\mathbf{S}$ is **packed** if for all plans $(\sigma,\tau) \in \mathbf{S}$ either $\tau = 0$ holds, or there exists another pair $(\sigma',\tau') \in \mathbf{S}$ with $\sigma \cap \sigma' \neq \emptyset$ and $\tau' + t(|\sigma'|) = \tau$. The schedules $\mathbf{S}_1, \ldots, \mathbf{S}_{n-1}$ are called **intermediary schedules of** $\mathbf{S}$ if $\mathbf{S}_1 \subseteq \ldots \subseteq \mathbf{S}_{n-1} \subseteq \mathbf{S}$ and $|\mathbf{S}_i| = i$ for all $i \in [n-1]$.

Lemma 2.1 (Decker *et al.* [34]) *For any instance (n,m,t) and associated schedule $\mathbf{S}$, there exists a schedule $\mathbf{S}_n$ and intermediary schedules $\mathbf{S}_1, \ldots, \mathbf{S}_{n-1}$ of $\mathbf{S}_n$ with the following properties:*

(1.) The makespan of $\mathbf{S}_n$ is bounded by the makespan of $\mathbf{S}$, that is,

$$T(n,m,t,\mathbf{S}_n) \quad \leq \quad T(n,m,t,\mathbf{S}) .$$

(2.) All (intermediary) schedules $\mathbf{S}_i$, $i \in [n]$, are packed.

(3.) For all (intermediary) schedules $\mathbf{S}_i$, $i \in [n]$, the latencies on the processors differ by at most $t(1)$, that is,

$$\widetilde{\Lambda}_m(\mathbf{S}_i) - \widetilde{\Lambda}_1(\mathbf{S}_i) \quad \leq \quad t(1) .$$

(4.) The finishing time of each job in $\mathbf{S}_n$ is bounded by its finishing time in $\mathbf{S}$.

In general, we can compute a schedule for $i \geq 2$ jobs by using a schedule for $i-1$ jobs and assigning a set of free processors to the ith job. This fact can be used to give an algorithm. Every step leads to a set of intermediary schedules. We get optimum schedules by computing intermediary schedules of optimum schedules iteratively. By Lemma 2.1, we can restrict our search to packed schedules $\mathbf{S}$ with a sorted vector for which $\widetilde{\Lambda}_m(\mathbf{S}) - \widetilde{\Lambda}_1(\mathbf{S}) \leq t(1)$ holds. By definition, packed schedules have the following property:

Proposition 2.2 (Decker *et al.* [34]) *Consider an arbitrary instance (n,m,t) and associated schedule $\mathbf{S}$. If $\mathbf{S}$ is packed, then, for all $j \in [m]$,*

$$\widetilde{\Lambda}_j(\mathbf{S}) \quad \in \quad \{r_1 \cdot t(1) + \cdots + r_m \cdot t(m) \mid r_1, \cdots, r_m \in [n] \cup \{0\}\} .$$

This observation leads to the following result:

Theorem 2.3 (Decker *et al.* [34])

(1.) If the range of the time function t is $\mathbb{R}^+$, then there exists an algorithm which computes an optimum schedule using $O(n(n+1)^{m^2}2^m)$ time.

(2.) If the range of the time function t is $\mathbb{N}$, then there exist algorithms which compute an optimum schedule using $O(\log n \cdot t(1)^{3m})$ or $O(n \cdot t(1)^m \cdot 2^m)$ time.

2.4 Phase-By-Phase Schedules

We have seen that it is possible to compute an optimum schedule with execution time exponential in the number m of processors, yielding an algorithm polynomial in the number n of jobs if m is constant. We now use **phase-by-phase schedules** to *approximate* an optimum schedule. Here, a schedule consists of phases in which each job uses the same number of processors. A new phase can not be started until the last phase has finished. Hence, we do not have to store plans for all jobs but may write a phase-by-phase schedule with k phases as $\mathbf{P} = (m_1, \ldots, m_k)$, where m_j denotes the number of processors used in phase $j \in [k]$. Clearly, the makespan is

$$T(n,m,t,\mathbf{P}) \quad = \quad \sum_{j\in[k]} t(m_j)\ .$$

Decker and Krandick [33] introduced an algorithm which computes an optimum phase-by-phase schedule using $O(n^2)$ time. The execution time of this algorithm can be improved to $O(n \cdot \min\{n,m\})$. We now show that there exists an algorithm which computes an optimum phase-by-phase schedule using $O(m^3)$ time.

Theorem 2.4 *There exists an algorithm which computes an optimum phase-by-phase schedule using $O(m^3)$ time.*

Proof: Fix any instance (n,m,t) and associated optimum phase-by-phase schedule $\mathbf{P}$, and denote r_j the number of phases using $j \in [m]$ processors in $\mathbf{P}$. Due to the speed-up property, $r_j \leq j-1$ holds for all $j \geq 2$. Furthermore, at most $\lfloor \frac{m}{j} \rfloor$ jobs are executed during such a phase. Therefore, the total number of jobs executed in phases using more than one processor is at most

$$\sum_{2\leq j\leq m} r_j \left\lfloor \frac{m}{j} \right\rfloor \quad \leq \quad \sum_{2\leq j\leq m} (j-1) \left\lfloor \frac{m}{j} \right\rfloor \quad \leq \quad (m-1)m\ .$$

The algorithm works as follows: If $n > (m-1)m$, then compute the minimum $k \in \mathbb{N}$ such that $n - k \cdot m \leq (m-1)m$. This can be done in constant time. The schedule starts with k sequential phases. Then, the improved algorithm from [33] is used to compute the schedule for the remaining $n - k \cdot m$ jobs. This needs $O(n \min\{n,m\}) = O(m^3)$ time. ■

Decker [32] showed that the makespan of an optimum phase-by-phase schedule is at most twice as large as the makespan of an optimum schedule. Trystram [141] gave the following example illustrating that computing an optimum phase-by-phase schedule can not lead to an approximation factor lower than $\frac{5}{4}$.

Example 2.5 (Trystram [141]) *Consider the following instance $(3,5,t)$: We have $n = 3$ jobs, $m = 5$ processors, and the time function t defined by $t(1) = 1$, $t(2) = \frac{1}{2}$ and $t(i) = \frac{1}{3}$ for all $i \in [5] \setminus [2]$. The optimum phase-by-phase schedule $\mathbf{P} = (2,5)$ has makespan*

$$T(3,5,t,\mathbf{P}) \quad = \quad t(2)+t(5) \quad = \quad \frac{5}{6}$$

whereas the optimum schedule $\{(\{1,2,3\},0),(\{1,2,3\},\frac{1}{3}),(\{4,5\},0)\}$ *has makespan*

$$T_{opt}(3,5,t) \quad = \quad 2t(3) \quad = \quad \frac{2}{3}$$

(see Figure 2.2). So, the approximation factor via phase-by-phase schedules is $\frac{5}{4}$*. Note that this example can be extended to* $n = 2k+1$ *jobs and* $m = 3k+2$ *processors for all* $k \in \mathbb{N}$*.*

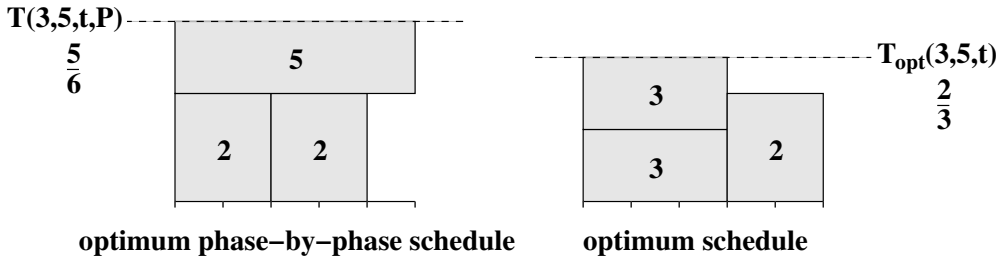


Figure 2.2: Optimum phase-by-phase schedule $\mathbf{P} = (2,5)$ (left hand side) and optimum schedule $\{(\{1,2,3\},0),(\{1,2,3\},\frac{1}{3}),(\{4,5\},0)\}$ (right hand side) for the instance in Example 2.5 (page 10).

We now present the algorithm PPS, stated as Algorithm 1 (page 12), using the following idea: Since we do not know anything about the time function (up to the monotonicity and the speed-up property), we compute a phase-by-phase schedule depending only on the relation between n and m. For the sake of readability, we only return the makespan of the phase-by-phase schedule which we choose. Note that the phase-by-phase schedule is (implicitly) returned. In the following, we prove that PPS always computes a phase-by-phase schedule which approximates an optimum schedule up to a factor of $\frac{5}{4}$ if $n \leq m$, and up to a factor of $\frac{6}{5}$ if $n > m$ (Theorem 2.6). Clearly, Example 2.5 (page 10) implies that the factor for the first case is tight. For the latter case, we think that by using a more complex algorithm we could get a better result. However, the approximation factor of PPS is at least $\frac{8}{7}$ (Example 2.7).

Theorem 2.6 PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor of* $\frac{5}{4}$ *if* $n \leq m$*, and up to a factor of* $\frac{6}{5}$ *if* $n > m$*, using constant time.*

Proof: In Section 2.5, we prove lower bounds on the makespan of an optimum schedule. Moreover, we show that we only have to consider a rather small set of phase-by-phase schedules to prove an approximation factor. By case analysis, we then prove the claim for the case $m < n$ in Section 2.6, and for the case $m \geq n$ in Section 2.7. ■

Example 2.7 *Consider the following instance* $(12,11,t)$*: We have* $n = 12$ *jobs,* $m = 11$ *processors, and the time function* t *defined by* $t(1) = 1$*,* $t(2) = \frac{1}{2}$ *and* $t(i) = \frac{1}{3}$ *for all* $i \in [11] \setminus [2]$*. The optimum phase-by-phase schedule* $\mathbf{P} = (1,3)$ *has makespan*

$$T(12,11,t,\mathbf{P}) \quad = \quad t(1)+t(3) \quad = \quad \frac{4}{3}$$

Algorithm 1 (PPS)

Input: an instance (n,m,t)
Output: a phase-by-phase schedule **P**

(1) **begin**
(2) **if** $n > m$ **then**
(3) **if** $m < n \leq \lfloor \frac{3}{2}m \rfloor$ **then**
(4) $a = \lfloor \frac{m}{n-m} \rfloor$
(5) **return** $t(1)+t(a)$
(6) **if** $\lfloor \frac{3}{2}m \rfloor < n \leq 2m$ **then**
(7) $a = \left\lfloor \frac{m}{n-\lfloor \frac{3}{2}m \rfloor} \right\rfloor$
(8) **return** $\min\{t(1)+t(2)+t(a), 2t(1)\}$
(9) **if** $2m < n \leq \lfloor \frac{5}{2}m \rfloor$ **then**
(10) $a = \lfloor \frac{m}{n-2m} \rfloor$
(11) **return** $2t(1)+t(a)$
(12) **if** $\lfloor \frac{5}{2}m \rfloor < n \leq 3m$ **then**
(13) $a = \left\lfloor \frac{m}{n-\lfloor \frac{5}{2}m \rfloor} \right\rfloor$
(14) **return** $\min\{2t(1)+t(2)+t(a), 3t(1)\}$
(15) **else**
(16) $a = \left\lfloor \frac{m}{n-\lfloor \frac{n}{m} \rfloor \cdot m} \right\rfloor$
(17) **return** $\lfloor \frac{n}{m} \rfloor \cdot t(1)+t(a)$
(18) **else**
(19) **if** $\lfloor \frac{m}{2} \rfloor < n \leq m$ **then**
(20) $a = \left\lfloor \frac{m}{n-\lfloor \frac{m}{2} \rfloor} \right\rfloor$
(21) **if** $2\lfloor \frac{m}{3} \rfloor < n \leq m$ **then**
(22) **return** $\min\{t(1), t(2)+t(a)\}$
(23) **if** $\lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor < n \leq 2\lfloor \frac{m}{3} \rfloor$ **then**
(24) **return** $\min\{t(1), t(2)+t(a), 2t(3)\}$
(25) **else**
(26) **return** $\min\{t(1), t(2)+t(a), t(3)+t(4)\}$
(27) **if** $\lfloor \frac{m}{3} \rfloor < n \leq \lfloor \frac{m}{2} \rfloor$ **then**
(28) $a = \left\lfloor \frac{m}{n-\lfloor \frac{m}{3} \rfloor} \right\rfloor$
(29) **if** $\frac{2}{5}m \leq n \leq \lfloor \frac{m}{2} \rfloor$ **then**
(30) **return** $\min\{t(2), t(3)+t(a)\}$
(31) **if** $2\lfloor \frac{m}{5} \rfloor < n < \frac{2}{5}m$ **then**
(32) **if** $m \leq 17$ **then**
(33) **return** $\min\{t(2), t(3)+t(a), t(4)+t(6)\}$
(34) **else**
(35) **return** $\min\{t(2), t(3)+t(a), t(4)+t(5)\}$
(36) **else**
(37) **return** $\min\{t(2), t(3)+t(a), 2t(5)\}$
(38) **if** $\lfloor \frac{m}{4} \rfloor < n \leq \lfloor \frac{m}{3} \rfloor$ **then**
(39) $a = \left\lfloor \frac{m}{n-\lfloor \frac{m}{4} \rfloor} \right\rfloor$
(40) **if** $\frac{2}{7}m \leq n \leq \lfloor \frac{m}{3} \rfloor$ **then**
(41) **return** $\min\{t(3), t(4)+t(a)\}$
(42) **if** $2\lfloor \frac{m}{7} \rfloor < n < \frac{2}{7}m$ **then**
(43) **if** $\mathrm{rem}(m,7) = 5$ **and** $m \leq 36$ **then**
(44) **return** $\min\{t(3), t(4)+t(a), t(5)+t(9), t(6)+t(7)\}$
(45) **if** $\mathrm{rem}(m,7) = 6$ **and** $m \leq 37$ **then**
(46) **return** $\min\{t(3), t(4)+t(a), t(5)+t(11), t(6)+t(7)\}$
(47) **else**
(48) **return** $\min\{t(3), t(4)+t(a), t(6)+t(7)\}$
(49) **else**
(50) **return** $\min\{t(3), t(4)+t(a), 2t(7)\}$
(51) **if** $\lfloor \frac{m}{k} \rfloor < n \leq \lfloor \frac{m}{k-1} \rfloor$ **and** $k \geq 5$ **then**
(52) $a = \left\lfloor \frac{m}{n-\lfloor \frac{m}{k} \rfloor} \right\rfloor$
(53) **return** $\min\{t(k-1), t(k)+t(a)\}$
(54) **end**

whereas the optimum schedule $\{(\{1,2,3\},0), (\{4,5,6\},0), (\{7,8,9\},0), (\{10\},0), (\{11\},0), (\{1,2,3\},\frac{1}{3}), (\{4,5,6\},\frac{1}{3}), (\{7,8,9\},\frac{1}{3}), (\{1,2\},\frac{2}{3}), (\{3,4\},\frac{2}{3}), (\{5,6\},\frac{2}{3}), (\{7,8\},\frac{2}{3})\}$ *has makespan*

$$T_{opt}(12,11,t) \quad = \quad t(2)+2t(3) \quad = \quad \frac{7}{6}$$

(see Figure 2.3). So, the approximation factor via phase-by-phase schedules is $\frac{8}{7}$.

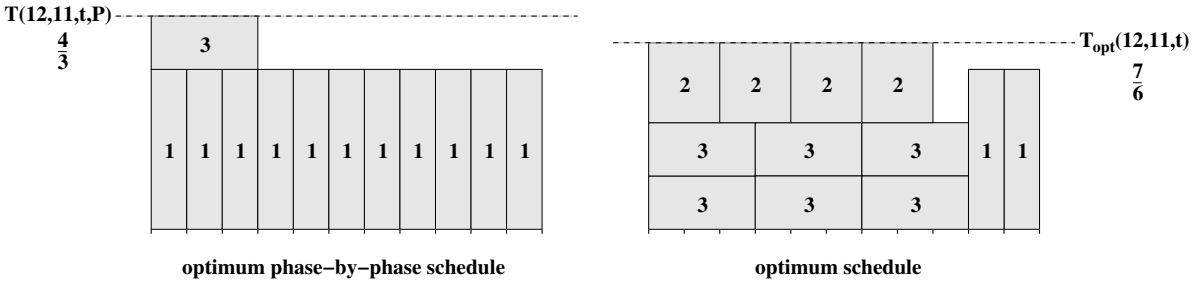


Figure 2.3: Optimum phase-by-phase schedule $\mathbf{P} = (1,3)$ (left hand side) and optimum schedule $\{(\{1,2,3\},0), (\{4,5,6\},0), (\{7,8,9\},0), (\{10\},0), (\{11\},0), (\{1,2,3\},\frac{1}{3}), (\{4,5,6\},\frac{1}{3}), (\{7,8,9\},\frac{1}{3}), (\{1,2\},\frac{2}{3}), (\{3,4\},\frac{2}{3}), (\{5,6\},\frac{2}{3}), (\{7,8\},\frac{2}{3})\}$ (right hand side) for the instance in Example 2.7 (page 11).

2.5 Technical Lemmas

In order to prove Theorem 2.6 (page 11), we mainly consider the quotient of the upper bound provided by PPS and a lower bound for $T_{opt}(n,m,t)$. We first prove lower bounds on $T_{opt}(n,m,t)$ (Lemmas 2.8 and 2.9, page 14). Unfortunately, these lower bounds do not always suffice to prove the factor $\frac{5}{4}$. In these cases, we have to consider all makespans of possible schedules. Since the number of such makespans can become very large, we introduce the following definition which helps us to restrict the number of makespans which have to be taken into account.

Consider two packed schedules $\mathbf{S}$ and $\widetilde{\mathbf{S}}$ for n jobs and m processors. Then, $\mathbf{S}$ **dominates** $\widetilde{\mathbf{S}}$ if for all valid time functions t, that is, time functions for which the monotonicity and the speed-up property hold, it is $T(n,m,t,\mathbf{S}) \leq T(n,m,t,\widetilde{\mathbf{S}})$. We show that we only have to consider the makespans of dominating schedules for which $\sum_{j\in[m]} r_j \cdot \frac{m}{j} \geq n$ (Lemma 2.8, page 14). In order to find another way to get a lower bound on $T_{opt}(n,m,t)$, we use the speed-up property as follows: If each job in the schedule uses at least j processors, then each job needs at least $j \cdot t(j)$ area. This leads to the lower bound

$$T_{opt}(n,m,t) \quad \geq \quad \frac{n}{m} \cdot (j \cdot t(j)) \,.$$

This technique will be beneficial.

Lemma 2.8 *Consider an arbitrary instance (n,m,t), and let $k \in \mathbb{R}^+$ with $k < n$. Then*

$$T_{opt}(n,m,t) \;\geq\; \min\left\{ \sum_{j\in[m]} r_j \cdot t(j) \;\middle|\; \sum_{j\in[m]} r_j \cdot \frac{m}{j} > k \right\}.$$

Proof: Fix any instance (n,m,t) with optimum makespan

$$T_{opt}(n,m,t) \;=\; \sum_{j\in[m]} r_j \cdot t(j)\,.$$

Assume, by way of contradiction, that $T_{opt}(n,m,t) \leq k$. Clearly, at most

$$\sum_{j\in[m]} r_j \cdot \frac{m}{j} \;\leq\; k < n$$

jobs can be executed in an optimum schedule, a contradiction to solvability. ■

Lemma 2.9 *Consider an arbitrary instance (n,m,t) and associated optimum schedule* **S** *with makespan*

$$T(n,m,t,\mathbf{S}) \;=\; T_{opt}(n,m,t) \;=\; \sum_{j\in[m]} r_j \cdot t(j)$$

and $\sum_{j\in[m]} r_j \geq 2$. Let $u_1,u_2 \in [m]$ with $u_1 \leq u_2$ such that $\frac{m}{u_1} + \frac{m}{u_2} \leq n$.

(1.) If $r_{u_1} \geq 1$, then $T_{opt}(n,m,t) \geq t(u_1)+t(u_2)$.
(2.) If $u_2 \leq u_1+1$ and $r_j = 0$ for all $j \in [u_1-1]$, then $T_{opt}(n,m,t) \geq t(u_1)+t(u_2)$.

Proof:

(1.) Assume that $r_{u_1} \geq 1$. If $r_{u_1} \geq 2$ or $r_j \geq 1$ for some $j \in [u_2-1]$, then monotonicity implies that the claim holds. Otherwise,

$$\begin{aligned}
T_{opt}(n,m,t) &\overset{r_{u_1}=1}{=} t(u_1) + \sum_{u_2\leq j\leq m} r_j \cdot t(j) \\
&\overset{\textit{speed-up property}}{\geq} t(u_1) + \sum_{u_2\leq j\leq m} r_j \cdot \frac{u_2}{j} \cdot t(u_2) \\
&= t(u_1) + \frac{u_2}{m} \cdot t(u_2) \sum_{u_2\leq j\leq m} r_j \cdot \frac{m}{j} \\
&\overset{\textit{Lemma 2.8}}{\geq} t(u_1) + \frac{u_2}{m} \cdot t(u_2) \cdot \left(n - \frac{m}{u_1}\right) \\
&\overset{n\geq \frac{m}{u_1}+\frac{m}{u_2}}{\geq} t(u_1) + t(u_2)\,,
\end{aligned}$$

as needed.

(2.) Assume that $u_2 \leq u_1 + 1$ and $r_j = 0$ for all $j \in [u_1 - 1]$. If $r_{u_1} \geq 1$, then we are done by (1.). Otherwise, we have $r_j = 0$ for all $j \in [u_1]$, and we get

$$\begin{aligned}
T_{opt}(n,m,t) &= \sum_{j\in[m]} r_j \cdot t(j) \\
&= \sum_{u_1+1\leq j\leq m} r_j \cdot t_j \\
&\overset{speed-up\ property}{\geq} \sum_{u_1+1\leq j\leq m} r_j \cdot \frac{u_2}{j} \cdot t(u_2) \\
&= \frac{u_2}{m} \cdot t(u_2) \cdot \sum_{u_1+1\leq j\leq m} r_j \cdot \frac{m}{j} \\
&\overset{Lemma\ 2.8,\ page\ 14}{\geq} \frac{u_2}{m} \cdot t(u_2) \cdot n \\
&\overset{n\geq \frac{m}{u_1}+\frac{m}{u_2}}{\geq} \frac{u_2}{m} \cdot t(u_2) \cdot \left(\frac{m}{u_1} + \frac{m}{u_2}\right) \\
&= t(u_2) + \frac{u_2}{u_1} \cdot t(u_2) \\
&\overset{speed-up\ property}{\geq} t(u_1) + t(u_2)\ ,
\end{aligned}$$

as needed. ■

2.6 Proof for the Case $m < n$

We prove Theorem 2.6 (page 11) by case analysis in Lemmas 2.10 - 2.14. The dominating schedules used in the proofs are listed in Table 2.1.

Case	$m < n \leq \lfloor \frac{3}{2}m \rfloor$ $a = \lfloor \frac{m}{n-m} \rfloor$	$\lfloor \frac{3}{2}m \rfloor < n \leq 2m$ $a = \lfloor \frac{m}{n-\lfloor \frac{3}{2}m \rfloor} \rfloor$	$2m < n \leq \lfloor \frac{5}{2}m \rfloor$ $a = \lfloor \frac{m}{n-2m} \rfloor$
Makespan	$t(1)+t(a)$	$2t(1)$	
	$t(2)+t(3)+t(5)$	$t(1)+t(2)+t(a)$ $t(1)+t(3)+t(5)$	$t(1)+t(1)+t(a)$
	$t(2)+t(3)+t(7)+t(41)$ $t(2)+t(3)+t(8)+t(23)$ $t(2)+t(3)+t(9)+t(17)$ $t(2)+t(3)+t(10)+t(14)$ $t(2)+t(3)+t(11)+t(13)$ $t(2)+t(4)+t(5)+t(19)$ $t(2)+t(4)+t(6)+t(11)$ $t(2)+t(4)+t(7)+t(9)$ $t(2)+t(5)+t(5)+t(9)$ $t(2)+t(5)+t(6)+t(7)$ $t(3)+t(3)+t(4)+t(11)$ $t(3)+t(3)+t(5)+t(7)$	$t(1)+t(3)+t(7)+t(41)$ $t(1)+t(3)+t(8)+t(23)$ $t(1)+t(3)+t(9)+t(17)$ $t(1)+t(3)+t(10)+t(14)$ $t(1)+t(3)+t(11)+t(13)$ $t(1)+t(4)+t(5)+t(19)$ $t(1)+t(4)+t(6)+t(11)$ $t(1)+t(4)+t(7)+t(9)$ $t(1)+t(5)+t(5)+t(10)$ $t(1)+t(5)+t(6)+t(7)$	$t(1)+t(2)+t(3)+t(5)$

Table 2.1: Makespan of dominating schedules for the case $m < n \leq \lfloor \frac{5}{2}m \rfloor$.

Lemma 2.10 *Let $m < n \leq \lfloor \frac{3}{2}m \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{6}{5}$.*

Proof: In this interval, PPS returns the phase-by-phase schedule $\mathbf{P} = (1, a)$, where $a = \lfloor \frac{m}{n-m} \rfloor$. Since $n \leq \lfloor \frac{3}{2} m \rfloor$, we have $a \geq 2$.
If $t(a) \leq \frac{1}{5} t(1)$, then the trivial lower bound $t(1)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(a)}{t(1)} \\ &\overset{t(a)\leq \frac{1}{5}t(1)}{\leq} \frac{t(1)+\frac{1}{5}t(1)}{t(1)} \\ &= \frac{6}{5}\,. \end{aligned}$$

So let $t(a) > \frac{1}{5} t(1)$, and consider the makespan of an optimum schedule.
First case: There exist no more than m jobs each being executed on at most a processors. We have

$$\begin{aligned} T_{opt}(n,m,t) &\overset{speed-up\ property}{\geq} \frac{1}{m}(m \cdot t(1) + (n-m) \cdot (a+1) \cdot t(a+1)) \\ &= t(1) + \frac{n-m}{m} \cdot (a+1) \cdot t(a+1) \\ &= t(1) + \frac{n-m}{m} \left(\left\lfloor \frac{m}{n-m} \right\rfloor + 1 \right) t(a+1) \\ &> t(1) + t(a+1) \\ &\overset{speed-up\ property}{\geq} t(1) + \frac{a}{a+1} t(a)\,. \end{aligned}$$

Due to monotonicity of the time function, we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(a)}{t(1)+\frac{a}{a+1}t(a)} \\ &\overset{a\geq 2}{\leq} \frac{t(1)+t(a)}{t(1)+\frac{2}{3}t(a)} \\ &\overset{t(a)\leq t(1)}{\leq} \frac{t(1)+t(1)}{t(1)+\frac{2}{3}t(1)} \\ &= \frac{6}{5}\,. \end{aligned}$$

Second case: There exist at least $m+1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{5} t(1)$, we have $T_{opt}(n,m,t) \geq 6t(a) > t(1) + t(a)$ for all makespans of a schedule with more than 5 addends. If we have 5 addends, then

$T_{opt}(n,m,t) \geq 5t(a) > \frac{4}{5}t(1)+t(a)$, and we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &< \frac{t(1)+t(a)}{\frac{4}{5}t(1)+t(a)} \\ &\overset{t(a)>\frac{1}{5}t(1)}{<} \frac{t(1)+\frac{1}{5}t(1)}{\frac{4}{5}t(1)+\frac{1}{5}t(1)} \\ &= \frac{6}{5}\,.\end{aligned}$$

Hence, we only have to consider makespans with at most 4 addends. The dominating schedules are those corresponding to the makespans $t(1)+t(a)$, $t(2)+t(3)+t(a)$, $t(2)+t(5)+2t(a)$ for $a \geq 4$, and $2t(3)+2t(a)$ for $a \geq 3$ (see Table 2.1, page 15).
$\underline{t(1)+t(a)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(1)+t(a)}{t(1)+t(a)} = 1\,.$$

$\underline{t(2)+t(3)+t(a)}$: For $a = 1$, we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)}{t(1)+t(2)+t(3)} \\ &\overset{speed-up\ property}{\leq} \frac{2t(1)}{(1+\frac{1}{2}+\frac{1}{3})t(1)} \\ &= \frac{12}{11}\,.\end{aligned}$$

For $a = 2$, we have

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)}{t(2)+t(2)+t(3)} \\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(2)}{t(1)+t(3)} \\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(2)}{t(1)+\frac{2}{3}t(2)} \\ &\overset{t(2)\leq t(1)}{\leq} \frac{2t(1)}{\frac{5}{3}t(1)} \\ &= \frac{6}{5}\,.\end{aligned}$$

For $a = 3$, we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(3)}{t(2)+2t(3)}\\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(3)}{\frac{1}{2}t(1)+2t(3)}\\ &\overset{speed-up\ property}{\leq} \frac{t(1)+\frac{1}{3}t(1)}{\frac{1}{2}t(1)+\frac{2}{3}t(1)}\\ &= \frac{8}{7}.\end{aligned}$$

For $a \geq 4$, we have

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(a)}{t(2)+t(3)+t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(a)}{(\frac{1}{2}+\frac{1}{3})t(1)+t(a)}\\ &\overset{t(a)>\frac{1}{5}t(1)}{<} \frac{t(1)+\frac{1}{5}t(1)}{\frac{5}{6}t(1)+\frac{1}{5}t(1)}\\ &= \frac{6}{5}\cdot\frac{30}{31}.\end{aligned}$$

$\underline{t(2)+t(5)+2t(a)}$: We have

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(a)}{t(2)+t(5)+2t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(a)}{(\frac{1}{2}+\frac{1}{5})t(1)+2t(a)}\\ &\overset{t(a)>\frac{1}{5}t(1)}{<} \frac{t(1)+\frac{1}{5}t(1)}{\frac{7}{10}t(1)+\frac{2}{5}t(1)}\\ &= \frac{6}{5}\cdot\frac{10}{11}.\end{aligned}$$

$\underline{2t(3)+2t(a)}$: It follows that

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(a)}{2t(3)+2t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(1)+t(a)}{\frac{2}{3}t(1)+2t(a)}\\ &\overset{t(a)>\frac{1}{5}t(1)}{<} \frac{t(1)+\frac{1}{5}t(1)}{\frac{2}{3}t(1)+\frac{2}{5}t(1)}\\ &= \frac{6}{5}\cdot\frac{15}{16}.\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.11 *Let $\lfloor \frac{3}{2}m \rfloor < n \leq 2m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{6}{5}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \quad = \quad \min\{2t(1), t(1)+t(2)+t(a)\} \,,$$

where $a = \lfloor \frac{m}{n-\lfloor \frac{3}{2}m \rfloor} \rfloor$. If $a = 1$, then $n = 2m$ and thus $T_{opt}(n,m,t) \geq 2t(1)$, proving the claim. So let $a \geq 2$.
If $t(a) \leq \frac{1}{4}t(1)$, then the trivial lower bound $t(1)+t(2)$ on the makespan of an optimum schedule implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(1)+t(2)} \\
&\overset{t(a)\leq\frac{1}{4}t(1)}{\leq} \frac{\frac{5}{4}t(1)+t(2)}{t(1)+t(2)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{\frac{5}{4}t(1)+\frac{1}{2}t(1)}{t(1)+\frac{1}{2}t(1)} \\
&= \frac{7}{6} \,.
\end{aligned}$$

So let $t(a) > \frac{1}{4}t(1)$, and consider the makespan of an optimum schedule.
First case: There exist no more than $\lfloor \frac{3}{2}m \rfloor$ jobs each being executed on at most a processors. If more than m jobs are executed by one processor, then $T_{opt}(n,m,t) \geq 2t(1)$, and we are done. So, assume that at most m jobs are executed by one processor. If $2 \mid m$, then

$$\begin{aligned}
T_{opt}(n,m,t) &\overset{\textit{speed-up property}}{\geq} \frac{1}{m}\left(m \cdot t(1) + \frac{m}{2} \cdot 2t(2) + \left(n - \frac{3}{2}m\right) \cdot (a+1) \cdot t(a+1)\right) \\
&\overset{\textit{speed-up property}}{\geq} t(1)+t(2)+\frac{a}{a+1}t(a) \\
&\overset{a\geq 2}{\geq} t(1)+t(2)+\frac{2}{3}t(a) \,.
\end{aligned}$$

Thus, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(1)+t(2)+\frac{2}{3}t(a)} \\
&\overset{a\geq 2,\ t(a)\leq t(2)}{\leq} \frac{t(1)+2t(2)}{t(1)+\frac{5}{3}t(2)} \\
&\overset{t(2)\leq t(1)}{\leq} \frac{3t(1)}{\frac{8}{3}t(1)} \\
&= \frac{9}{8} \,.
\end{aligned}$$

So, assume $2 \nmid m$. If $m = 3$, then we only have to consider phase-by-phase schedules to find an optimum schedule. If $n = 6$, then $T_{opt}(n,m,t) = 2t(1)$, and we are done. So, assume $n = 5$. Then $a = 3$, and we get $T_{opt}(n,m,t) \geq t(1) + 2t(3)$. Thus,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(3)}{t(1)+2t(3)} \\
&\overset{\text{speed-up property}}{\leq} \frac{t(1)+\frac{5}{2}t(3)}{t(1)+2t(3)} \\
&\overset{t(3)\leq t(1)}{\leq} \frac{\frac{7}{2}t(1)}{3t(1)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

If $m \geq 5$, it follows that

$$\begin{aligned}
&T_{opt}(n,m,t) \\
&\overset{\text{speed-up property}}{\geq} \frac{1}{m}\left(m\cdot t(1) + \left\lfloor \frac{m}{2} \right\rfloor \cdot 2t(2) + \left(n - \left\lfloor \frac{3}{2}m \right\rfloor\right)\cdot(a+1)\cdot t(a+1)\right) \\
&= t(1) + \left(1-\frac{1}{m}\right)t(2) + \frac{n-\left\lfloor \frac{3}{2}m \right\rfloor}{m}\cdot\left(\left\lfloor \frac{m}{n-\left\lfloor \frac{3}{2}m \right\rfloor} \right\rfloor + 1\right)\cdot t(a+1)) \\
&\overset{\text{speed-up property}}{\geq} t(1) + \left(1-\frac{1}{m}\right)t(2) + \left(1+\frac{1}{m}\right)\frac{a}{a+1}\cdot t(a) \\
&\overset{a\geq 2,\ m\geq 5}{\geq} t(1) + \frac{4}{5}t(2) + \frac{6}{5}\cdot\frac{2}{3}t(a) \\
&= t(1) + \frac{4}{5}\left(t(2)+t(a)\right)\,.
\end{aligned}$$

We get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(1)+\frac{4}{5}(t(2)+t(a))} \\
&\overset{t(2)\leq t(1),\ t(a)\leq t(1)}{\leq} \frac{t(1)+2t(1)}{t(1)+\frac{4}{5}\cdot 2t(1)} \\
&= \frac{15}{13} \\
&< \frac{6}{5}\,.
\end{aligned}$$

Second case: There exist at least $\lfloor \frac{3}{2}m \rfloor + 1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{4}t(1)$, we have $T_{opt}(n,m,t) \geq 7t(a) > \frac{7}{4}t(1)$ for all makespans of a schedule with more than 6 addends. So,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)}{\frac{7}{4}t(1)} \\
&= \frac{8}{7}\,.
\end{aligned}$$

If we have 6 addends, then at least one addend is $t(i)$, $i \leq 3$, due to feasibility. This yields

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(3)+5t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{t(1)+t(2)+t(a)}{t(1)+t(3)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(1)+t(2)+t(a)}{t(1)+\frac{2}{3}t(2)+t(a)} \\
&\overset{t(2)\leq t(1)}{\leq} \frac{2t(1)+t(a)}{\frac{5}{3}t(1)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{2t(1)+\frac{1}{4}t(1)}{\frac{5}{3}t(1)+\frac{1}{4}t(1)} \\
&= \frac{27}{23} \\
&< \frac{6}{5}.
\end{aligned}$$

If we have 5 addends, then at least one addend is $t(i)$, $i \leq 2$, or $3t(3)$ is an addend. In the first case, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(2)+4t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{t(1)+t(2)+t(a)}{\frac{3}{4}t(1)+t(2)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{\frac{3}{2}t(1)+t(a)}{\frac{5}{4}t(1)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{\frac{3}{2}t(1)+\frac{1}{4}t(1)}{\frac{5}{4}t(1)+\frac{1}{4}t(1)} \\
&= \frac{7}{6}.
\end{aligned}$$

In the second case,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{3t(3)+2t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(1)+t(2)+t(a)}{\frac{1}{2}t(1)+t(2)+2t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{t(1)+t(2)+t(a)}{\frac{3}{4}t(1)+t(2)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{\frac{3}{2}t(1)+t(a)}{\frac{5}{4}t(1)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{\frac{3}{2}t(1)+\frac{1}{4}t(1)}{\frac{5}{4}t(1)+\frac{1}{4}t(1)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

Hence, we only have to consider makespans with at most 4 addends. The dominating schedules are those corresponding to the makespans $2t(1)$, $t(1)+t(3)+t(a)$ for $a \geq 3$, and $t(1)+t(5)+2t(a)$ for $a \geq 3$ (see Table 2.1, page 15).
$\underline{2t(1)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{2t(1)}{2t(1)} = 1\,.$$

$\underline{t(1)+t(3)+t(a)}$: For $a=3$, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(3)}{t(1)+2t(3)} \\
&\overset{speed-up\ property}{\leq} \frac{t(1)+\frac{5}{2}t(3)}{t(1)+2t(3)} \\
&\overset{t(3)\leq t(1)}{\leq} \frac{\frac{7}{2}t(1)}{3t(1)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

For $a \geq 4$, we have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(1)+t(2)+t(a)}{t(1)+t(3)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(1)+t(2)+t(a)}{t(1)+\frac{2}{3}t(2)+t(a)} \\
&\overset{t(2)\leq t(1)}{\leq} \frac{2t(1)+t(a)}{\frac{5}{3}t(1)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{2t(1)+\frac{1}{4}t(1)}{\frac{5}{3}t(1)+\frac{1}{4}t(1)} \\
&= \frac{27}{23} \\
&< \frac{6}{5}.
\end{aligned}$$

$\underline{t(1)+t(5)+2t(a)}$: It follows that

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)}{t(1)+t(5)+2t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{2t(1)}{(1+\frac{1}{5})t(1)+2t(a)} \\
&\overset{t(a)>\frac{1}{4}t(1)}{<} \frac{2t(1)}{(1+\frac{1}{5}+\frac{2}{4})t(1)} \\
&= \frac{20}{17} \\
&< \frac{6}{5}.
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.12 *Let $2m < n \leq \lfloor \frac{5}{2}m \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{6}{5}$.*

Proof: In this interval, PPS returns the phase-by-phase schedule $\mathbf{P} = (1,1,a)$, where $a = \lfloor \frac{m}{n-2m} \rfloor$. Since $n \leq \lfloor \frac{5}{2}m \rfloor$, we have $a \geq 2$.
If $t(a) \leq \frac{1}{3}t(1)$, then the trivial lower bound $2t(1)$ on the makespan of an optimum schedule implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)+t(a)}{2t(1)} \\
&\overset{t(a)\leq\frac{1}{3}t(1)}{<} \frac{2t(1)+\frac{1}{3}t(1)}{2t(1)} \\
&= \frac{7}{6}.
\end{aligned}$$

So let $t(a) > \frac{1}{3}t(1)$, and consider the makespan of an optimum schedule.
First case: There exist no more than $2m$ jobs each being executed on at most a processors. We have

$$\begin{aligned} T_{opt}(n,m,t) &\overset{\text{speed-up property}}{\geq} \frac{1}{m}(2m \cdot t(1) + (n-2m) \cdot (a+1) \cdot t(a+1)) \\ &= 2 \cdot t(1) + \frac{n-2m}{m} \cdot (a+1) \cdot t(a+1) \\ &= 2 \cdot t(1) + \frac{n-2m}{m} \cdot \left(\left\lfloor \frac{m}{n-2m} \right\rfloor + 1 \right) t(a+1) \\ &\overset{\text{speed-up property}}{\geq} 2 \cdot t(1) + \frac{a}{a+1} t(a) \,. \end{aligned}$$

We get

$$\begin{aligned} T_{opt}(n,m,t) &\leq \frac{2t(1)+t(a)}{2t(1)+\frac{a}{a+1}t(a)} \\ &\overset{a \geq 1,\ t(a) \leq t(1)}{\leq} \frac{2t(1)+t(1)}{2t(1)+\frac{1}{2}t(1)} \\ &= \frac{6}{5} \,. \end{aligned}$$

Second case: There exist at least $2m+1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{3}t(1)$, we have $T_{opt}(n,m,t) \geq 6t(a) > \frac{5}{3}t(1) + t(a)$ for all makespans of a schedule with more than 5 addends. So,

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &< \frac{2t(1)+t(a)}{\frac{5}{3}t(1)+t(a)} \\ &\overset{t(a) > \frac{1}{3}t(1)}{<} \frac{2t(1)+\frac{1}{3}t(1)}{2t(1)} \\ &= \frac{7}{6} \,. \end{aligned}$$

If we have 5 addends, then the addend $t(1)$ yields $T_{opt}(n,m,t) \geq t(1) + 4t(a) > 2t(1) + t(a)$. So,

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{2t(1)+t(a)}{2t(1)+t(a)} = 1 \,.$$

Otherwise, at least two addends are $t(2)$ due to feasibility. We get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)+t(a)}{2t(2)+3t(a)} \\ &\overset{\text{speed-up property}}{\leq} \frac{2t(1)+t(a)}{t(1)+3t(a)} \\ &\overset{t(a) > \frac{1}{3}t(1)}{<} \frac{2t(1)+\frac{1}{3}t(1)}{t(1)+t(1)} \\ &= \frac{7}{6} \,. \end{aligned}$$

Hence, we only have to consider makespans with at most 4 addends. The dominating schedules are those corresponding to the makespans $2t(1)+t(a)$ and $t(1)+t(2)+2t(a)$ for $a \geq 3$ (see Table 2.1, page 15).
$\underline{2t(1)+t(a)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{2t(1)+t(a)}{2t(1)+t(a)} = 1 .$$

$\underline{t(1)+t(2)+2t(a)}$: We have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(1)+t(a)}{t(1)+t(2)+2t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{2t(1)+t(a)}{\frac{3}{2}t(1)+2t(a)} \\
&\overset{t(a)>\frac{1}{3}t(1)}{<} \frac{2t(1)+\frac{1}{3}t(1)}{\frac{3}{2}t(1)+\frac{2}{3}t(1)} \\
&= \frac{14}{13} .
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.13 *Let $\lfloor \frac{5}{2}m \rfloor < n \leq 3m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{6}{5}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule $\mathbf{P}$ with

$$T(n,m,t,\mathbf{P}) = \min\{3t(1), 2t(1)+t(2)+t(a)\} ,$$

where $a = \lfloor \frac{m}{n-\lfloor \frac{5}{2}m \rfloor} \rfloor$. If $a = 1$, then $n = 3m$ and thus $T_{opt}(n,m,t) \geq 3t(1)$, proving the claim. So, let $a \geq 2$. The trivial lower bound $2t(1)+t(2)$ on the makespan of an optimum schedule implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{3t(1)}{2t(1)+t(2)} \\
&\overset{speed-up\ property}{\leq} \frac{3t(1)}{2t(1)+\frac{1}{2}t(1)} \\
&= \frac{6}{5} .
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.14 *Let $km < n \leq (k+1)m$ with $k \in \mathbb{N}$, $k \geq 3$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{6}{5}$.*

Proof: In this interval, PPS returns the phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \quad = \quad \left\lfloor \frac{n}{m} \right\rfloor \cdot t(1) + t(a) \; ,$$

where $a = \lfloor \frac{m}{n-km} \rfloor$.
If $t(a) \leq \frac{1}{2}t(1)$, then the trivial lower bound $k \cdot t(1)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} & \leq & \frac{k \cdot t(1) + t(a)}{k \cdot t(1)} \\ & \overset{t(a) \leq \frac{1}{2}t(1)}{\leq} & \frac{kt(1) + \frac{1}{2}t(1)}{kt(1)} \\ & \overset{k \geq 3}{\leq} & \frac{3t(1) + \frac{1}{2}t(1)}{3t(1)} \\ & = & \frac{7}{6} \; . \end{aligned}$$

So let $t(a) > \frac{1}{2}t(1)$, and consider the makespan of an optimum schedule.
First case: There exist no more than km jobs each being executed on at most a processors.
We have

$$\begin{aligned} T_{opt}(n,m,t) & \overset{\textit{speed-up property}}{\geq} & \frac{1}{m}(km \cdot t(1) + (n-km) \cdot (a+1) \cdot t(a+1)) \\ & = & k \cdot t(1) + \frac{n-km}{m} \cdot (a+1) \cdot t(a+1) \\ & = & k \cdot t(1) + \frac{n-km}{m} \cdot \left(\left\lfloor \frac{m}{n-km} \right\rfloor + 1 \right) t(a+1) \\ & \overset{\textit{speed-up property}}{\geq} & k \cdot t(1) + \frac{a}{a+1} t(a) \; . \end{aligned}$$

We get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} & \leq & \frac{k \cdot t(1) + t(a)}{k \cdot t(1) + \frac{a}{a+1} t(a)} \\ & \overset{a \geq 1,\; t(a) \leq t(1)}{\leq} & \frac{(k+1)t(1)}{(k+\frac{1}{2})t(1)} \\ & \overset{k \geq 3}{\leq} & \frac{8}{7} \; . \end{aligned}$$

Second case: There exist at least $km+1$ jobs each being executed on at most a processors.
Since $t(a) > \frac{1}{2}t(1)$, $(k+\frac{1}{2}) \cdot t(1)$ is a trivial lower bound on the makespan of an optimum schedule, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} & \leq & \frac{(k+1) \cdot t(1)}{(k+\frac{1}{2}) \cdot t(1)} \\ & \overset{k \geq 3}{\leq} & \frac{8}{7} \; . \end{aligned}$$

This completes the proof of the claim. ■

2.7 Proof for the Case $m \geq n$

Unfortunately, the proof of this case is more technical. In order to simplify the reading, we investigate some smaller sub-intervals separately. We first restrict n and m which have to be considered (Observations 2.15 and 2.16, page 28). In addition, we prove a helpful lower bound on the makespan of an optimum schedule (Lemma 2.17, page 28). We then prove the approximation factor by case analysis (Lemmas 2.18 - 2.30). The dominating schedules used in the proofs are listed in Table 2.2.

Case	$\lfloor \frac{m}{2} \rfloor < n \leq m$ $a = \lfloor \frac{m}{n-\lfloor \frac{m}{2} \rfloor} \rfloor$	$\lfloor \frac{m}{3} \rfloor < n \leq \lfloor \frac{m}{2} \rfloor$ $a = \lfloor \frac{m}{n-\lfloor \frac{m}{3} \rfloor} \rfloor$	$\lfloor \frac{m}{4} \rfloor < n \leq \lfloor \frac{m}{3} \rfloor$ $a = \lfloor \frac{m}{n-\lfloor \frac{m}{4} \rfloor} \rfloor$	
Makespan	$t(1)$	$t(2)$	$t(3)$	
	$t(2)+t(a)$ $t(3)+t(5)$	$t(3)+t(a)$ $t(4)+t(11)$ $t(5)+t(7)$	$t(4)+t(a)$ $t(5)+t(19)$ $t(6)+t(11)$ $t(7)+t(9)$	
	$t(3)+t(7)+t(41)$	$t(4)+t(13)+t(155)$	$t(5)+t(21)+t(419)$	$t(7)+t(10)+t(39)$
	$t(3)+t(8)+t(23)$	$t(4)+t(14)+t(83)$	$t(5)+t(22)+t(219)$	$t(7)+t(11)+t(61)$
	$t(3)+t(9)+t(17)$	$t(4)+t(15)+t(59)$	$t(5)+t(23)+t(153)$	$t(7)+t(12)+t(41)$
	$t(3)+t(10)+t(14)$	$t(4)+t(16)+t(47)$	$t(5)+t(24)+t(119)$	$t(7)+t(13)+t(33)$
	$t(3)+t(11)+t(13)$	$t(4)+t(17)+t(40)$	$t(5)+t(25)+t(99)$	$t(7)+t(14)+t(27)$
	$t(4)+t(5)+t(19)$	$t(4)+t(18)+t(35)$	$t(5)+t(26)+t(86)$	$t(7)+t(15)+t(24)$
	$t(4)+t(6)+t(11)$	$t(4)+t(19)+t(32)$	$t(5)+t(27)+t(77)$	$t(7)+t(16)+t(22)$
	$t(4)+t(7)+t(9)$	$t(4)+t(20)+t(29)$	$t(5)+t(28)+t(69)$	$t(7)+t(17)+t(20)$
	$t(5)+t(5)+t(9)$	$t(4)+t(21)+t(27)$	$t(5)+t(29)+t(64)$	$t(7)+t(18)+t(19)$
	$t(5)+t(6)+t(7)$	$t(4)+t(22)+t(26)$	$t(5)+t(30)+t(59)$	$t(8)+t(9)+t(71)$
		$t(4)+t(23)+t(25)$	$t(5)+t(31)+t(56)$	$t(8)+t(10)+t(39)$
		$t(5)+t(9)+t(44)$	$t(5)+t(32)+t(53)$	$t(8)+t(11)+t(29)$
		$t(5)+t(10)+t(29)$	$t(5)+t(33)+t(50)$	$t(8)+t(12)+t(23)$
		$t(5)+t(11)+t(23)$	$t(5)+t(34)+t(48)$	$t(8)+t(13)+t(20)$
		$t(5)+t(12)+t(19)$	$t(5)+t(35)+t(46)$	$t(8)+t(14)+t(18)$
		$t(5)+t(13)+t(17)$	$t(5)+t(36)+t(44)$	$t(8)+t(15)+t(17)$
		$t(5)+t(14)+t(16)$	$t(5)+t(37)+t(43)$	$t(9)+t(9)+t(35)$
		$t(6)+t(7)+t(41)$	$t(5)+t(38)+t(42)$	$t(9)+t(10)+t(25)$
		$t(6)+t(8)+t(23)$	$t(5)+t(39)+t(41)$	$t(9)+t(11)+t(20)$
		$t(6)+t(9)+t(17)$	$t(6)+t(13)+t(155)$	$t(9)+t(12)+t(17)$
		$t(6)+t(10)+t(14)$	$t(6)+t(14)+t(83)$	$t(9)+t(13)+t(16)$
		$t(6)+t(11)+t(13)$	$t(6)+t(15)+t(59)$	$t(9)+t(14)+t(14)$
		$t(7)+t(7)+t(20)$	$t(6)+t(16)+t(47)$	$t(10)+t(10)+t(19)$
		$t(7)+t(8)+t(15)$	$t(6)+t(17)+t(40)$	$t(10)+t(11)+t(16)$
		$t(7)+t(9)+t(12)$	$t(6)+t(18)+t(35)$	$t(10)+t(12)+t(14)$
		$t(7)+t(10)+t(11)$	$t(6)+t(19)+t(32)$	$t(10)+t(13)+t(13)$
		$t(8)+t(8)+t(11)$	$t(6)+t(20)+t(29)$	$t(11)+t(11)+t(14)$
		$t(8)+t(9)+t(10)$	$t(6)+t(21)+t(27)$	$t(11)+t(12)+t(13)$
			$t(6)+t(22)+t(26)$	
			$t(6)+t(23)+t(25)$	

Table 2.2: Makespan of dominating schedules for the case $\lfloor \frac{m}{4} \rfloor < n \leq m$.

Observation 2.15

(1.) If $m = 1$, then $T_{opt}(n,m,t) = t(1)$.

(2.) If $m = 2$, then

$$T_{opt}(n,m,t) = \begin{cases} t(2) & \text{if } n = 1, \\ t(1) & \text{if } n = 2. \end{cases}$$

(3.) If $m = 3$, then

$$T_{opt}(n,m,t) = \begin{cases} t(3) & \text{if } n = 1, \\ \min\{t(1), 2t(3)\} & \text{if } n = 2, \\ t(1) & \text{if } n = 3. \end{cases}$$

(4.) If $m = 4$, then

$$T_{opt}(n,m,t) = \begin{cases} t(4) & \text{if } n = 1, \\ t(2) & \text{if } n = 2, \\ \min\{t(1), t(2)+t(4)\} & \text{if } n = 3, \\ t(1) & \text{if } n = 4. \end{cases}$$

In all cases, PPS *computes a phase-by-phase schedule with the same makespan. Thus, we only have to consider $m \geq 5$.*

Observation 2.16

(1.) If $n = 1$, then $T_{opt}(n,m,t) = t(m)$, and PPS *returns a phase-by-phase schedule with the same makespan.*

(2.) If $n = 2$, then the makespan of the phase-by-phase schedule $\mathbf{P}$ computed by PPS *is at most $t(\lfloor \frac{m}{2} \rfloor)$. On the other hand, each job is executed by at least $\lfloor \frac{m}{2} \rfloor$ processors in an optimum schedule. Thus,*

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(\lfloor \frac{m}{2} \rfloor)}{\frac{n}{m}\lfloor \frac{m}{2} \rfloor t(\lfloor \frac{m}{2} \rfloor)} \\ &\overset{n=2}{=} \frac{1}{\frac{2}{m}\lfloor \frac{m}{2} \rfloor} \\ &\leq \frac{1}{\frac{2}{m}\frac{m-1}{2}} \\ &= \frac{m}{m-1} \\ &\overset{m \geq 5 \text{ (Observation 2.15, page 27)}}{\leq} \frac{5}{4}. \end{aligned}$$

Thus, we only have to consider $n \geq 3$.

Lemma 2.17 *Let $k \in \mathbb{N}$, $k \geq 2$, with $\lfloor \frac{m}{k} \rfloor < n \leq \lfloor \frac{m}{k-1} \rfloor$, let $a = \lfloor \frac{m}{n - \lfloor \frac{m}{k} \rfloor} \rfloor$, and let $r = \mathrm{rem}(m,k)$. If there exist no more than $\lfloor \frac{m}{k} \rfloor$ jobs each being executed on at least k processors and at most a processors, then*

$$T_{opt}(n,m,t) \geq \left(1 - \frac{r}{m}\right) \cdot t(k) + \frac{a}{a+1} \cdot \frac{m+1}{m} \cdot t(a).$$

Proof: We have

$$\begin{aligned}
T_{opt}(n,m,t) &\overset{\text{speed-up property}}{\geq} \frac{1}{m}\left(\left\lfloor\frac{m}{k}\right\rfloor\cdot k\cdot t(k)+\left(n-\left\lfloor\frac{m}{k}\right\rfloor\right)\left(\left\lfloor\frac{m}{n-\lfloor\frac{m}{k}\rfloor}\right\rfloor+1\right)\cdot t(a+1)\right)\\
&\geq \left(1-\frac{r}{m}\right)\cdot t(k)+\frac{n-\lfloor\frac{m}{k}\rfloor}{m}\cdot\frac{m+1}{n-\lfloor\frac{m}{k}\rfloor}\cdot t(a+1)\\
&= \left(1-\frac{r}{m}\right)\cdot t(k)+\frac{m+1}{m}\cdot t(a+1)\\
&\overset{\text{speed-up property}}{\geq} \left(1-\frac{r}{m}\right)\cdot t(k)+\left(\frac{m+1}{m}\right)\cdot\left(\frac{a}{a+1}\right)\cdot t(a)\,.
\end{aligned}$$

This completes the proof of the claim. ■

2.7.1 The Case $\left\lfloor\frac{m}{2}\right\rfloor < n \leq m$

In the following, let $a = \lfloor\frac{m}{n-\lfloor\frac{m}{2}\rfloor}\rfloor$. Before we start to prove Theorem 2.6 (page 11) in this interval, we show that we can assume $a \geq 4$.

Lemma 2.18 *Let $\lfloor\frac{m}{2}\rfloor < n \leq m$. If $a \leq 3$, then* PPS *computes a phase-by-phase schedule which is an optimum schedule up to a factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \;\leq\; \min\{t(1),t(2)+t(a)\}\,.$$

$\underline{a \leq 2}$: This implies $n > \lfloor\frac{m}{2}\rfloor + \frac{m}{3}$, and we get

$$\begin{aligned}
n &\geq \left\lfloor\frac{m}{2}\right\rfloor+\left\lfloor\frac{m}{3}\right\rfloor+1\\
&\geq \frac{5}{6}m-\frac{1}{2}-\frac{2}{3}+1\\
&= \frac{5}{6}m-\frac{1}{6}\,.
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(1)}{\frac{n}{m}t(1)}\\
&\overset{n\geq\frac{5}{6}m-\frac{1}{6}}{\leq} \frac{1}{\frac{5}{6}-\frac{1}{6m}}\\
&\overset{m\geq 5\ (\textit{Observation 2.15, page 27})}{\leq} \frac{1}{\frac{5}{6}-\frac{1}{30}}\\
&= \frac{5}{4}\,,
\end{aligned}$$

proving the claim for this case.
$\underline{a=3}$: This implies $n > \lfloor \frac{m}{2} \rfloor + \frac{m}{4}$, and we get

$$\begin{aligned} n &\geq \left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{4} \right\rfloor + 1 \\ &\geq \frac{3}{4}m - \frac{1}{2} - \frac{3}{4} + 1 \\ &= \frac{3}{4}m - \frac{1}{4}\,. \end{aligned}$$

If more than $\frac{m}{2}$ jobs are executed on 2 processors, then $T_{opt}(n,m,t) \geq t(1)$, and we are done. Otherwise,

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(2)+t(3)}{\frac{1}{m}\left(\frac{m}{2}\cdot 2t(2) + \left(n-\frac{m}{2}\right)3t(3)\right)} \\ &\overset{n \geq \frac{3}{4}m-\frac{1}{4}}{\leq} \frac{t(2)+t(3)}{t(2)+\frac{\frac{3}{4}m-\frac{1}{4}-\frac{m}{2}}{m}\cdot 3t(3)} \\ &= \frac{t(2)+t(3)}{t(2)+\frac{3}{4}\left(1-\frac{1}{m}\right)t(3)} \\ &\overset{m \geq 5 \text{ (Observation 2.15, page 27)}}{\leq} \frac{t(2)+t(3)}{t(2)+\frac{3}{4}\cdot\frac{4}{5}t(3)} \\ &= \frac{t(2)+t(3)}{t(2)+\frac{3}{5}t(3)} \\ &\overset{t(3)\leq t(2)}{\leq} \frac{t(2)+t(2)}{t(2)+\frac{3}{5}t(2)} \\ &= \frac{5}{4}\,. \end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.19 *Let $\lfloor \frac{m}{2} \rfloor < n \leq \lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) = \min\{t(1), t(2)+t(a), t(3)+t(4)\}\,.$$

If $t(a) \leq \frac{1}{4}t(2)$, then the trivial lower bound $t(2)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{t(2)} \\ &\overset{t(a)\leq\frac{1}{4}t(2)}{\leq} \frac{t(2)+\frac{1}{4}t(2)}{t(2)} \\ &= \frac{5}{4}\,. \end{aligned}$$

So let $t(a) > \frac{1}{4}t(2)$, and consider the makespan of an optimum schedule.
First case: There exist no more than m jobs each being executed on at most a processors.
If there exists a job which is executed by 1 processor, then $T_{opt}(n,m,t) \geq t(1)$, and we are done. So, assume that all jobs are executed by at least 2 processors. Clearly, $n \leq \frac{7}{12}m$ and $n \geq 3$ implies $m \geq 6$. Thus,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.17, page 28}}{\leq} \frac{t(2)+t(a)}{\left(1-\frac{1}{m}\right)t(2)+\left(1+\frac{1}{m}\right)\cdot\frac{a}{a+1}\cdot t(a)} \\
&\overset{a\geq 4 \text{ (Lemma 2.18, page 29)}}{\leq} \frac{t(2)+t(a)}{\left(1-\frac{1}{m}\right)t(2)+\left(1+\frac{1}{m}\right)\cdot\frac{4}{5}\cdot t(a)} \\
&\overset{m\geq 6}{\leq} \frac{t(2)+t(a)}{\left(1-\frac{1}{6}\right)t(2)+\left(1+\frac{1}{6}\right)\cdot\frac{4}{5}\cdot t(a)} \\
&= \frac{t(2)+t(a)}{\frac{5}{6}t(2)+\frac{14}{15}t(a)} \\
&< \frac{5}{4}\,.
\end{aligned}$$

Second case: There exist at least $\lfloor \frac{m}{2} \rfloor + 1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{4}t(2)$, we have $T_{opt}(n,m,t) \geq 4t(a) > \frac{3}{4}t(2) + t(a)$ for all makespan of a schedule with more than 3 addends. This yields

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{\frac{3}{4}t(2)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(2)}{<} \frac{t(2)+\frac{1}{4}t(2)}{\frac{3}{4}t(2)+\frac{1}{4}t(2)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

Hence, we only have to consider makespans with at most 3 addends. The dominating schedules are those corresponding to the makespans $t(1)$, $t(2)+t(a)$, $t(3)+t(a)$ for $a \leq 5$, $t(4)+2t(a)$, $2t(5)+t(a)$, and $t(5)+t(6)+t(a)$ for $a \leq 7$ (see Table 2.2, page 27).
$\underline{t(1)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(1)}{t(1)} = 1\,.$$

$\underline{t(2)+t(a)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(2)+t(a)}{t(2)+t(a)} = 1\,.$$

$\underline{t(3)+t(a)}$: Since $a \leq 5$ we have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(4)}{t(3)+t(a)} \\
&\overset{a\leq 5,\ t(a)\geq t(5)}{\leq} \frac{t(3)+t(4)}{t(3)+t(5)} \\
&\overset{speed-up\ property}{\leq} \frac{t(3)+t(4)}{t(3)+\frac{4}{5}t(4)} \\
&\overset{t(4)\leq t(3)}{\leq} \frac{t(3)+t(3)}{t(3)+\frac{4}{5}t(3)} \\
&= \frac{10}{9}.
\end{aligned}$$

$\underline{t(4)+2t(a)}$: We get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{t(4)+2t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(2)+t(a)}{\frac{1}{2}t(2)+2t(a)} \\
&\overset{t(a)>\frac{1}{4}t(2)}{<} \frac{t(2)+\frac{1}{4}t(2)}{\frac{1}{2}t(2)+\frac{1}{2}t(2)} \\
&= \frac{5}{4}.
\end{aligned}$$

$\underline{2t(5)+t(a)}$: We have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{2t(5)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(2)+t(a)}{\frac{4}{5}t(2)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(2)}{<} \frac{t(2)+\frac{1}{4}t(2)}{\frac{4}{5}t(2)+\frac{1}{4}t(2)} \\
&= \frac{5}{4}\cdot\frac{20}{21}.
\end{aligned}$$

$\underline{t(5)+t(6)+t(a)}$: Since $a \leq 7$ we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(4)}{t(5)+t(6)+t(a)} \\
&\overset{a\leq 7,\ t(a)\geq t(7)}{\leq} \frac{t(3)+t(4)}{t(5)+t(6)+t(7)} \\
&\overset{speed-up\ property}{\leq} \frac{\frac{4}{3}t(4)+t(4)}{\frac{4}{5}t(4)+\frac{4}{6}t(4)+\frac{4}{7}t(4)} \\
&= \frac{245}{214} \\
&< \frac{6}{5}\,.
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.20 *Let $\lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor < n < \frac{7}{12}m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) = \min\{t(1), t(2)+t(a), 2t(3)\}\,.$$

There exists an n with $\lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor < n < \frac{7}{12}m$ if and only if $\frac{1}{3}\cdot \mathrm{rem}(m,3) + \frac{1}{4}\cdot \mathrm{rem}(m,4) > 1$. This implies that $n \geq \frac{7}{12}m - \frac{5}{12}$. Furthermore, we have $m \in \{7\} \cup \mathbb{N}_{\geq 9}$. Consider the makespan of an optimum schedule.

- $\underline{t(1) \text{ is the largest addend in the makespan:}}$ In this case, we are done.
- $\underline{t(2) \text{ is the largest addend in the makespan:}}$ Let $b = \min\{m, \lceil \frac{m}{n-\frac{m}{2}} \rceil\}$. Since $\frac{m}{2} + \frac{m}{b} \leq n$, we get

$$T_{opt}(n,m,t) \overset{Lemma\ 2.9,\ page\ 14}{\geq} t(2)+t(b)\,.$$

(1.) If $\mathrm{rem}(m,3) = 2$ and $\mathrm{rem}(m,4) = 2$, then $2 \mid m$. Thus, $b \leq a+1$, and we get

$$\frac{b}{a} \leq \frac{a+1}{a} \overset{a\geq 4\ (Lemma\ 2.18,\ page\ 29)}{\leq} \frac{5}{4}\,.$$

(2.) If $\mathrm{rem}(m,3) = 2$ and $\mathrm{rem}(m,4) = 3$, then $2 \nmid m$ and $n = \lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor + 1 = \frac{7}{12}m - \frac{5}{12}$. Furthermore, we can write $m = 12q+11$, $q \geq 0$. Thus,

$$\begin{aligned}
a &= \left\lfloor \frac{m}{n-\frac{m}{2}+\frac{1}{2}} \right\rfloor = \left\lfloor \frac{m}{\frac{m}{12}+\frac{1}{12}} \right\rfloor = \left\lfloor \frac{12q+11}{q+1} \right\rfloor, \\
b &= \min\left\{ m, \left\lceil \frac{m}{\frac{m}{12}-\frac{5}{12}} \right\rceil \right\} = \min\left\{ 12q+11, \left\lceil \frac{24q+22}{2q+1} \right\rceil \right\}.
\end{aligned}$$

If $q = 0$, then $a = b = 11$. Otherwise, $q \geq 1$ implies

$$\begin{aligned} a &= \left\lfloor \frac{12q+11}{q+1} \right\rfloor = \left\lfloor 12 - \frac{1}{q+1} \right\rfloor \geq \left\lfloor 12 - \frac{1}{2} \right\rfloor = 11\,, \\ b &\leq \left\lceil \frac{24q+22}{2q+1} \right\rceil = \left\lceil 12 + \frac{10}{2q+1} \right\rceil \leq \left\lceil 12 + \frac{10}{3} \right\rceil = 16\,. \end{aligned}$$

(3.) If $\mathrm{rem}(m,3) = 1$ and $\mathrm{rem}(m,4) = 3$, then $2 \nmid m$ and $n = \lfloor \frac{m}{3} \rfloor + \lfloor \frac{m}{4} \rfloor + 1 = \frac{7}{12}m - \frac{1}{12}$. Furthermore, we can write $m = 12q+7$, $q \geq 0$. Thus,

$$\begin{aligned} a &= \left\lfloor \frac{m}{n - \frac{m}{2} + \frac{1}{2}} \right\rfloor = \left\lfloor \frac{m}{\frac{m}{12} + \frac{5}{12}} \right\rfloor = \left\lfloor \frac{12q+7}{q+1} \right\rfloor, \\ b &= \min\left\{ m, \left\lceil \frac{m}{\frac{m}{12} - \frac{1}{12}} \right\rceil \right\} = \min\left\{ 12q+7, \left\lceil \frac{24q+14}{2q+1} \right\rceil \right\}. \end{aligned}$$

If $q = 0$, then $a = b = 7$. Otherwise, $q \geq 1$ implies

$$\begin{aligned} a &= \left\lfloor \frac{12q+7}{q+1} \right\rfloor = \left\lfloor 12 - \frac{5}{q+1} \right\rfloor \geq \left\lfloor 12 - \frac{5}{2} \right\rfloor = 9\,, \\ b &\leq \left\lceil \frac{24q+14}{2q+1} \right\rceil = \left\lceil 12 + \frac{2}{2q+1} \right\rceil \leq \left\lceil 12 + \frac{2}{3} \right\rceil = 13\,. \end{aligned}$$

Thus, $\frac{b}{a} \leq \frac{16}{11}$ in all cases, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{t(2)+t(b)} \\ &\overset{\text{speed-up property}}{\leq} \frac{t(2)+t(a)}{t(2)+\frac{11}{16}t(a)} \\ &\overset{a\geq 2,\ t(a)\leq t(2)}{\leq} \frac{2t(2)}{\frac{27}{16}t(2)} \\ &= \frac{32}{27} \\ &< \frac{5}{4}\,. \end{aligned}$$

- $t(3)$ is the largest addend in the makespan: If $m = 7$, then $n = 4$, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{\frac{n}{m}\cdot 3t(3)} \\ &\overset{m=7,n=4}{=} \frac{2t(3)}{\frac{12}{7}t(3)} \\ &= \frac{7}{6}\,. \end{aligned}$$

Otherwise, $m \geq 9$ implies $\frac{m}{3} + \frac{m}{5} \leq \frac{7}{12}m - \frac{5}{12} \leq n$. Thus,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.9, page 14}}{\leq} \frac{2t(3)}{t(3)+t(5)} \\
&\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{t(3)+\frac{3}{5}t(3)} \\
&= \frac{5}{4}.
\end{aligned}$$

- $t(i)$, $i \geq 4$, is the largest addend in the makespan: If $m = 7$, then $n = 4$, and we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{\frac{n}{m} \cdot 4t(4)} \\
&\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{\frac{n}{m} \cdot 3t(3)} \\
&\overset{m=7,\ n=4}{=} \frac{2t(3)}{\frac{12}{7}t(3)} \\
&= \frac{7}{6}.
\end{aligned}$$

Otherwise,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{\frac{n}{m} \cdot 4t(4)} \\
&\overset{n \geq \frac{7}{12}m - \frac{5}{12}}{\leq} \frac{2t(3)}{\frac{\frac{7}{12}m - \frac{5}{12}}{m} \cdot 4t(4)} \\
&= \frac{2t(3)}{\left(\frac{7}{3} - \frac{5}{3m}\right) t(4)} \\
&\overset{m \geq 9}{\leq} \frac{2t(3)}{\frac{58}{27}t(4)} \\
&\overset{\text{speed-up property}}{\leq} \frac{2t(3)}{\frac{29}{18}t(3)} \\
&= \frac{36}{29} \\
&< \frac{5}{4}.
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.21 *Let $\frac{7}{12}m \leq n \leq 2\lfloor \frac{m}{3} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \quad = \quad \min\{t(1), t(2)+t(a), 2t(3)\} \; .$$

Consider the makespan of an optimum schedule.

- $t(1)$ is the largest addend in the makespan: In this case, we are done.
- $t(2)$ is the largest addend in the makespan: Let $b = \min\{m, \lceil \frac{m}{n-\frac{m}{2}} \rceil\}$. Since $\frac{m}{2} + \frac{m}{b} \leq n$, we get

$$T_{opt}(n,m,t) \overset{\text{Lemma 2.9, page 14}}{\geq} t(2)+t(b) \; .$$

We have

$$\begin{aligned}
\frac{b}{a} &\leq \frac{\left\lceil \frac{m}{n-\frac{m}{2}} \right\rceil}{\left\lfloor \frac{m}{n-\frac{m}{2}+\frac{1}{2}} \right\rfloor} \\
&\leq \frac{\frac{m+(n-\frac{m}{2})-\frac{1}{2}}{n-\frac{m}{2}}}{\frac{m-(n-\frac{m}{2}+\frac{1}{2})+\frac{1}{2}}{n-\frac{m}{2}+\frac{1}{2}}} \\
&= \frac{\left(n+(\frac{m}{2}-\frac{1}{2})\right)\left(n-(\frac{m}{2}-\frac{1}{2})\right)}{\left(n-\frac{m}{2}\right)\left(\frac{3}{2}m-n\right)} \\
&< \frac{n^2-\frac{m^2}{4}}{\left(n-\frac{m}{2}\right)\left(\frac{3}{2}m-n\right)} \\
&= f(n) \; .
\end{aligned}$$

The function $f(n)$ is strictly increasing in n since its first derivative is

$$f'(n) \quad = \quad \frac{8m}{(2n-3m)^2} \quad \overset{\frac{2}{3}m \geq n > \frac{7}{12}m}{>} \quad 0 \; .$$

Hence,

$$\begin{aligned}
f(n) &\overset{n \leq \frac{2}{3}m}{\leq} f(\frac{2}{3}m) \\
&= \frac{\left(\frac{2}{3}m\right)^2-\frac{m^2}{4}}{\left(\frac{2}{3}m-\frac{m}{2}\right)\left(\frac{3}{2}m-\frac{2}{3}m\right)} \\
&= \frac{7}{5} \; .
\end{aligned}$$

Thus, $\frac{b}{a} \leq \frac{7}{5}$, and we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{t(2)+t(b)} \\
&\overset{speed-up\ property}{\leq} \frac{t(2)+t(a)}{t(2)+\frac{5}{7}t(a)} \\
&\overset{a\geq 2,\ t(a)\leq t(2)}{\leq} \frac{2t(2)}{\frac{12}{7}t(2)} \\
&= \frac{7}{6}\ .
\end{aligned}$$

- $t(i)$, $i \geq 3$, is the largest addend in the makespan: Since $n \geq \frac{7}{12}m = \frac{m}{3} + \frac{m}{4}$, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{Lemma\ 2.9,\ page\ 14}{\leq} \frac{2t(3)}{t(3)+t(4)} \\
&\overset{speed-up\ property}{\leq} \frac{2t(3)}{\frac{7}{4}t(3)} \\
&= \frac{8}{7}\ .
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.22 *Let $2\lfloor\frac{m}{3}\rfloor < n < \frac{2}{3}m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule $\mathbf{P}$ with

$$T(n,m,t,\mathbf{P}) = \min\{t(1), t(2)+t(a)\}\ .$$

There exists an n with $2\lfloor\frac{m}{3}\rfloor < n < \frac{2}{3}m$ if and only if $\mathrm{rem}(m,3) = 2$. This implies $n = \frac{2}{3}m - \frac{1}{3}$. Furthermore, we can write $m = 3q+2$, $q \geq 1$. Consider the makespan of an optimum schedule.

- $t(1)$ is the largest addend in the makespan: In this case, we are done.
- $t(2)$ is the largest addend in the makespan: Let $b = \min\{m, \lceil\frac{m}{n-\frac{m}{2}}\rceil\}$. Since $\frac{m}{2} + \frac{m}{b} \leq n$, we get

$$T_{opt}(n,m,t) \overset{Lemma\ 2.9,\ page\ 14}{\geq} t(2)+t(b)\ .$$

We have

$$\begin{aligned}
a &\geq \left\lfloor \frac{m}{n-\frac{m}{2}+\frac{1}{2}} \right\rfloor = \left\lfloor \frac{m}{\frac{m}{6}+\frac{1}{6}} \right\rfloor = \left\lfloor \frac{6q+4}{q+1} \right\rfloor\ , \\
b &= \min\left\{ m, \left\lceil \frac{m}{\frac{m}{6}-\frac{1}{3}} \right\rceil \right\} = \min\left\{ 3q+2, \left\lceil \frac{6q+4}{q} \right\rceil \right\}\ .
\end{aligned}$$

If $q = 1$, then $a \geq b = m = 5$. If $q = 2$, then $2 \mid m$. Thus, $b \leq a+1$, and we get

$$\frac{b}{a} \leq \frac{a+1}{a} \overset{a\geq 4 \text{ (Lemma 2.18, page 29)}}{\leq} \frac{5}{4}.$$

So, assume $q \geq 3$. Then

$$\begin{aligned} a &\geq \left\lfloor \frac{6q+4}{q+1} \right\rfloor = \left\lfloor 6 - \frac{2}{q+1} \right\rfloor \geq \left\lfloor 6 - \frac{1}{2} \right\rfloor = 5, \\ b &\leq \left\lceil \frac{6q+4}{q} \right\rceil = \left\lceil 6 + \frac{4}{q} \right\rceil \leq \left\lceil 6 \cdot \frac{4}{3} \right\rceil = 8. \end{aligned}$$

Thus, $\frac{b}{a} \leq \frac{8}{5}$ in all cases, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{t(2)+t(b)} \\ &\overset{\text{speed-up property}}{\leq} \frac{t(2)+t(a)}{t(2)+\frac{5}{8}t(a)} \\ &\overset{a\geq 2,\ t(a)\leq t(2)}{\leq} \frac{2t(2)}{\frac{13}{8}t(2)} \\ &= \frac{16}{13} \\ &< \frac{5}{4}. \end{aligned}$$

- $t(i)$, $i \geq 3$, is the largest addend in the makespan: If more than $\frac{m}{3}$ jobs are processed by 3 processors, then $T_{opt}(n,m,t) \geq 2t(3)$, and we have

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{2t(3)} \\ &\overset{a\geq 3,\ t(a)\leq t(3)}{\leq} \frac{t(2)+t(3)}{2t(3)} \\ &\overset{\text{speed-up property}}{\leq} \frac{\frac{5}{3}t(2)}{\frac{4}{3}t(2)} \\ &= \frac{5}{4}. \end{aligned}$$

Otherwise,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\textit{speed-up property}}{\leq} \frac{t(2)+t(a)}{\frac{1}{m}\left(\frac{m}{3}\cdot 3t(3)+\left(n-\frac{m}{3}\right)\cdot 4t(4)\right)} \\
&\overset{n=\frac{2}{3}m-\frac{1}{3}}{=} \frac{t(2)+t(a)}{t(3)+\left(\frac{4}{3}-\frac{4}{3m}\right)t(4)} \\
&\overset{m\geq 5 \text{ (}\textit{Observation } 2.15\text{, }\textit{page } 27\text{)}}{\leq} \frac{t(2)+t(a)}{t(3)+\frac{16}{15}t(4)} \\
&\overset{a\geq 4,\ t(a)\leq t(4)}{\leq} \frac{t(2)+t(4)}{t(3)+\frac{16}{15}t(4)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{\frac{3}{2}t(2)}{(\frac{2}{3}+\frac{8}{15})t(2)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.23 *Let $\frac{2}{3}m \leq n \leq m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule $\mathbf{P}$ with

$$T(n,m,t,\mathbf{P}) \;=\; \min\{t(1),t(2)+t(a)\}\,.$$

Consider the makespan of an optimum schedule.

- $t(1)$ is the largest addend in the makespan: In this case, we are done.
- $t(2)$ is the largest addend in the makespan: If more than $\frac{m}{2}$ jobs are executed by 2 or 3 processors, then $T_{opt}(n,m,t) \geq 2t(3)$, and we have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)+t(a)}{2t(3)} \\
&\overset{a\geq 3,\ t(a)\leq t(3)}{\leq} \frac{t(2)+t(3)}{2t(3)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{\frac{5}{3}t(2)}{\frac{4}{3}t(2)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

Otherwise,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(2)+t(a)}{\frac{1}{m}\left(\frac{m}{2}\cdot 2t(2)+\left(n-\frac{m}{2}\right)\cdot 4t(4)\right)} \\
&\overset{n\geq\frac{2}{3}m}{\leq} \frac{t(2)+t(a)}{\frac{1}{m}\left(\frac{m}{2}\cdot 2t(2)+\left(\frac{2}{3}m-\frac{m}{2}\right)\cdot 4t(4)\right)} \\
&= \frac{t(2)+t(a)}{t(2)+\frac{2}{3}t(4)} \\
&\overset{a\geq 4,\ t(a)\leq t(4)}{\leq} \frac{t(2)+t(4)}{t(2)+\frac{2}{3}t(4)} \\
&\overset{t(4)\leq t(2)}{\leq} \frac{2t(2)}{\frac{5}{3}t(2)} \\
&= \frac{6}{5}\,.
\end{aligned}$$

- $t(i)$, $i \geq 3$, is in the makespan: Since $n \geq \frac{2}{3}m$, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.9, page 14}}{\leq} \frac{t(2)+t(a)}{2t(3)} \\
&\overset{a\geq 3,\ t(a)\leq t(3)}{\leq} \frac{t(2)+t(3)}{2t(3)} \\
&\overset{\text{speed-up property}}{\leq} \frac{\frac{5}{3}t(2)}{\frac{4}{3}t(2)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

This completes the proof of the claim. ■

2.7.2 The Case $\lfloor \frac{m}{3} \rfloor < n \leq \lfloor \frac{m}{2} \rfloor$.

In the following, let $a = \lfloor \frac{m}{n-\lfloor \frac{m}{3} \rfloor} \rfloor$. We have

$$\begin{aligned}
a &= \left\lfloor \frac{m}{n-\lfloor \frac{m}{3} \rfloor} \right\rfloor \\
&\overset{n\leq\frac{m}{2}}{\geq} \left\lfloor \frac{m}{\frac{m}{2}-\lfloor \frac{m}{3} \rfloor} \right\rfloor \\
&\geq \left\lfloor \frac{m}{\frac{m}{6}+\frac{2}{3}} \right\rfloor \\
&\overset{m\geq 5\ (\text{Observation 2.15, page 27})}{\geq} \left\lfloor \frac{5}{\frac{5}{6}+\frac{2}{3}} \right\rfloor \\
&= 4\,.
\end{aligned}$$

Lemma 2.24 *Let $\lfloor \frac{m}{3} \rfloor < n \leq 2\lfloor \frac{m}{5} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \quad = \quad \min\{t(2), t(3)+t(a), 2t(5)\} .$$

If $t(a) \leq \frac{1}{4}t(3)$, then the trivial lower bound $t(3)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(3)} \\ &\overset{t(a)\leq\frac{1}{4}t(3)}{\leq} \frac{t(3)+\frac{1}{4}t(3)}{t(3)} \\ &= \frac{5}{4} . \end{aligned}$$

So let $t(a) > \frac{1}{4}t(3)$, and consider the makespan of an optimum schedule.
First case: There exist no more than $\lfloor \frac{m}{3} \rfloor$ jobs each being executed on at most a processors. Clearly, $n \leq 2\lfloor \frac{m}{5} \rfloor$ and $n \geq 3$ implies $m \geq 10$. Thus,

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.17, page 28}}{\leq} \frac{t(3)+t(a)}{\left(1-\frac{2}{m}\right)t(3)+\left(1+\frac{1}{m}\right)\cdot\frac{a}{a+1}\cdot t(a)} \\ &\overset{a\geq 4}{\leq} \frac{t(3)+t(a)}{\left(1-\frac{2}{m}\right)t(3)+\left(1+\frac{1}{m}\right)\cdot\frac{4}{5}\cdot t(a)} \\ &\overset{m\geq 10}{\leq} \frac{t(3)+t(a)}{\left(1-\frac{1}{5}\right)t(3)+\left(1+\frac{1}{10}\right)\cdot\frac{4}{5}\cdot t(a)} \\ &< \frac{5}{4} . \end{aligned}$$

Second case: There exist at least $\lfloor \frac{m}{3} \rfloor + 1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{4}t(3)$, we have $T_{opt}(n,m,t) \geq 4t(a) > \frac{3}{4}t(3) + t(a)$ for all makespans of a schedule with more than 3 addends. This yields

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{\frac{3}{4}t(3)+t(a)} \\ &\overset{t(a)>\frac{1}{4}t(3)}{<} \frac{t(3)+\frac{1}{4}t(3)}{\frac{3}{4}t(3)+\frac{1}{4}t(3)} \\ &= \frac{5}{4} . \end{aligned}$$

Hence, we only have to consider makespans with at most 3 addends. The dominating schedules are those corresponding to the makespans $t(2)$, $t(3)+t(a)$, $t(4)+t(a)$ for $a \leq 11$, $t(5)+t(a)$ for $a \leq 7$, $t(6)+2t(a)$, $t(7)+t(9)+t(a)$, $t(7)+t(10)+t(a)$ for $a \leq 11$, and $t(8)+t(9)+$

$t(a)$ for $a \leq 10$ (see Table 2.2, page 27).
$\underline{t(2)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(2)}{t(2)} = 1\,.$$

$\underline{t(3)+t(a)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(3)+t(a)}{t(3)+t(a)} = 1\,.$$

$\underline{t(4)+t(a)}$: Since $a \leq 11$ we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(4)+t(a)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{t(3)+t(a)}{\frac{3}{4}t(3)+t(a)} \\
&\overset{a\leq 11,\ t(a)\geq t(11)}{\leq} \frac{t(3)+t(11)}{\frac{3}{4}t(3)+t(11)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{t(3)+\frac{3}{11}t(3)}{\frac{3}{4}t(3)+\frac{3}{11}t(3)} \\
&= \frac{56}{45} \\
&< \frac{5}{4}\,.
\end{aligned}$$

$\underline{t(5)+t(a)}$: The fact that $a \leq 7$ implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(5)}{t(5)+t(a)} \\
&\overset{a\leq 7,\ t(a)\geq t(7)}{\leq} \frac{2t(5)}{t(5)+t(7)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{2t(5)}{t(5)+\frac{5}{7}t(5)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

$\underline{t(6)+2t(a)}$: We have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(6)+2t(a)} \\
&\overset{\textit{speed-up property}}{\leq} \frac{t(3)+t(a)}{\frac{1}{2}t(3)+2t(a)} \\
&\overset{t(a)>\frac{1}{4}t(3)}{<} \frac{t(3)+\frac{1}{4}t(3)}{\frac{1}{2}t(3)+\frac{1}{2}t(3)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

$\underline{t(7)+t(9)+t(a)}$: We have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(7)+t(9)+t(a)} \\
&\overset{speed-up\ property}{\leq} \frac{t(3)+t(a)}{(\frac{3}{7}+\frac{3}{9})t(3)+t(a)} \\
&= \frac{t(3)+t(a)}{\frac{16}{21}t(3)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(3)}{<} \frac{t(3)+\frac{1}{4}t(3)}{\frac{16}{21}t(3)+\frac{1}{4}t(3)} \\
&= \frac{5}{4}\cdot\frac{84}{85}.
\end{aligned}$$

$\underline{t(7)+t(10)+t(a)}$: Since $a \leq 11$ we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(5)}{t(7)+t(10)+t(a)} \\
&\overset{a\leq 11,\ t(a)\geq t(11)}{\leq} \frac{2t(5)}{t(7)+t(10)+t(11)} \\
&\overset{speed-up\ property}{\leq} \frac{2t(5)}{(\frac{5}{7}+\frac{5}{10}+\frac{5}{11})t(5)} \\
&= \frac{308}{257} \\
&< \frac{6}{5}.
\end{aligned}$$

$\underline{t(8)+t(9)+t(a)}$: Since $a \leq 10$ we have

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(5)}{t(8)+t(9)+t(a)} \\
&\overset{a\leq 10,\ t(a)\geq t(10)}{\leq} \frac{2t(5)}{t(8)+t(9)+t(10)} \\
&\overset{speed-up\ property}{\leq} \frac{2t(5)}{(\frac{5}{8}+\frac{5}{9}+\frac{5}{10})t(5)} \\
&= \frac{144}{121} \\
&< \frac{6}{5}.
\end{aligned}$$

This completes the proof the claim. ■

Lemma 2.25 *Let $2\lfloor\frac{m}{5}\rfloor < n < \frac{2}{5}m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule with

$$T(n,m,t,\mathbf{P}) = \begin{cases} \min\{t(2), t(3)+t(a), t(4)+t(6)\} & \text{if } m \leq 17, \\ \min\{t(2), t(3)+t(a), t(4)+t(5)\} & \text{otherwise.} \end{cases}$$

There exists an n with $2\lfloor \frac{m}{5} \rfloor < n < \frac{2}{5}m$ if and only if $\text{rem}(m,5) \geq 3$. This implies $n \geq \frac{2}{5}m - \frac{3}{5}$. Furthermore, we have $m \in \{8,13,14\} \cup \mathbb{N}_{\geq 18}$ due to $n \geq 3$. Consider the makespan of an optimum schedule.

- $t(2)$ is the largest addend in the makespan: In this case, we are done.
- $t(3)$ is the largest addend in the makespan: Let $b = \min\{m, \lceil \frac{m}{n-\frac{m}{3}} \rceil\}$. If $m = 8$, then $T_{opt}(n,m,t) \geq t(3)+t(8)$, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(2)}{t(3)+t(8)} \\ &\overset{speed-up\ property}{\leq} \frac{t(2)}{(\frac{2}{3}+\frac{2}{8})t(2)} \\ &= \frac{12}{11}. \end{aligned}$$

Now let $m \geq 13$.

(1.) If $\text{rem}(m,5) = 3$, then $n = \frac{2}{5}m - \frac{1}{5}$. Furthermore, we can write $m = 5q+3$, $q \geq 2$. Thus,

$$\begin{aligned} a &\geq \left\lfloor \frac{m}{n-\frac{m}{3}+\frac{2}{3}} \right\rfloor = \left\lfloor \frac{m}{\frac{m}{15}+\frac{7}{15}} \right\rfloor = \left\lfloor \frac{15q+9}{q+2} \right\rfloor, \\ b &= \min\left\{ m, \left\lceil \frac{m}{\frac{m}{15}-\frac{1}{5}} \right\rceil \right\} = \min\left\{ 5q+3, \left\lceil \frac{15q+9}{q} \right\rceil \right\}. \end{aligned}$$

If $q = 2$, then $b = 13$ and $a \geq 9$. If $q = 3$, then $3 \mid m$, and we have $b \leq a+1$, $a \geq 11$. Otherwise, $q \geq 4$ implies

$$\begin{aligned} a &\geq \left\lfloor \frac{15q+9}{q+2} \right\rfloor = \left\lfloor 15 - \frac{21}{q+2} \right\rfloor \geq \left\lfloor 15 - \frac{21}{6} \right\rfloor = 11, \\ b &\leq \left\lceil \frac{15q+9}{q} \right\rceil = \left\lceil 15 + \frac{9}{q} \right\rceil \leq \left\lceil 15 + \frac{9}{4} \right\rceil = 18. \end{aligned}$$

(2.) If $\text{rem}(m,5) = 4$, then $n = \frac{2}{5}m - \frac{3}{5}$. Furthermore, we can write $m = 5q+4$, $q \geq 2$. Thus,

$$\begin{aligned} a &\geq \left\lfloor \frac{m}{n-\frac{m}{3}+\frac{2}{3}} \right\rfloor = \left\lfloor \frac{m}{\frac{m}{15}+\frac{1}{15}} \right\rfloor = \left\lfloor \frac{15q+12}{q+1} \right\rfloor, \\ b &= \min\left\{ m, \left\lceil \frac{m}{\frac{m}{15}-\frac{3}{5}} \right\rceil \right\} = \min\left\{ 5q+4, \left\lceil \frac{15q+12}{q-1} \right\rceil \right\}. \end{aligned}$$

If $q = 2$, then $a \geq b = 14$. If $q = 3$, then $b = 19$ and $a \geq 14$. If $q = 4$, then $3 \mid m$, and we have $b \leq a+1$, $a \geq 14$. Otherwise, $q \geq 5$ implies

$$\begin{array}{rcccccccc} a & \geq & \left\lfloor \dfrac{15q+12}{q+1} \right\rfloor & = & \left\lfloor 15 - \dfrac{3}{q+1} \right\rfloor & \geq & \left\lfloor 15 - \dfrac{1}{2} \right\rfloor & = & 14\,, \\ b & \leq & \left\lceil \dfrac{15q+12}{q-1} \right\rceil & = & \left\lceil 15 + \dfrac{27}{q-1} \right\rceil & \leq & \left\lceil 15 + \dfrac{27}{4} \right\rceil & = & 22\,. \end{array}$$

Thus, $\frac{b}{a} \leq \frac{18}{11}$ in all cases, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(3)+t(b)} \\ &\overset{speed-up\ property}{\leq} \frac{t(3)+t(a)}{t(3)+\frac{11}{18}t(a)} \\ &\overset{a\geq 3,\ t(a)\leq t(3)}{\leq} \frac{2t(3)}{\frac{29}{18}t(3)} \\ &= \frac{36}{29} \\ &< \frac{5}{4}\,. \end{aligned}$$

- $t(4)$ is the largest addend in the makespan: If $m = 8$, then we have $T_{opt}(n,m,t) \geq t(4) + t(8)$, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(5)}{t(4)+t(8)} \\ &\overset{speed-up\ property}{\leq} \frac{t(4)+t(5)}{t(4)+\frac{5}{8}t(5)} \\ &\overset{t(5)\leq t(4)}{\leq} \frac{t(4)+t(4)}{t(4)+\frac{5}{8}t(4)} \\ &= \frac{16}{13} \\ &< \frac{5}{4}\,. \end{aligned}$$

Now let $m \geq 13$. Since this implies $\frac{m}{4} + \frac{m}{10} \leq \frac{2}{5}m - \frac{3}{5} \leq n$, we get

$$T_{opt}(n,m,t) \overset{Lemma\ 2.9,\ page\ 14}{\geq} t(4) + t(10)\,.$$

If $m \in \{13,14\}$, then $a = m$. Otherwise,

$$\begin{aligned} a &= \left\lfloor \frac{m}{n - \lfloor \frac{m}{3} \rfloor} \right\rfloor \\ &\overset{n \le \frac{2}{5}m - \frac{1}{5}}{\geq} \left\lfloor \frac{m}{\frac{2}{5}m - \frac{1}{5} - \frac{m}{3} + \frac{2}{3}} \right\rfloor \\ &= \left\lfloor \frac{15m}{m+7} \right\rfloor \\ &\overset{m \geq 18}{\geq} \left\lfloor \frac{270}{25} \right\rfloor \\ &= 10\,. \end{aligned}$$

Thus, $a \geq 10$ in all cases, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)+t(a)}{t(4)+t(10)} \\ &\overset{a \geq 10,\ t(a) \leq t(10)}{\leq} \frac{t(3)+t(10)}{t(4)+t(10)} \\ &\overset{speed-up\ property}{\leq} \frac{t(3)+t(10)}{\frac{3}{4}t(3)+t(10)} \\ &\overset{speed-up\ property}{\leq} \frac{\frac{13}{10}t(3)}{\frac{21}{20}t(3)} \\ &= \frac{26}{21} \\ &< \frac{5}{4}\,. \end{aligned}$$

- $t(i)$, $i \geq 5$, is the largest addend in the makespan: If $m = 8$, then $n = 3$, and we get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{speed-up\ property}{\leq} \frac{t(4)+t(5)}{\frac{n}{m} \cdot 5t(5)} \\ &\overset{n=3,\ m=8}{=} \frac{t(4)+t(5)}{\frac{15}{8} \cdot t(5)} \\ &\overset{speed-up\ property}{\leq} \frac{t(4)+\frac{4}{5}t(4)}{\frac{15}{8} \cdot \frac{4}{5}t(4)} \\ &= \frac{6}{5}\,. \end{aligned}$$

If $m \in \{13,14\}$, then $n = 5$. If more than 2 users are executed by 5 processors, then

$T_{opt}(n,m,t) \geq 2t(5)$, and we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(5)}{2t(5)} \\ &\overset{speed-up\ property}{\leq} \frac{t(4)+\frac{4}{5}t(4)}{\frac{8}{5}t(4)} \\ &= \frac{9}{8}\,.\end{aligned}$$

Otherwise, we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{speed-up\ property}{\leq} \frac{t(4)+t(6)}{\frac{1}{14}(2\cdot 5t(5)+3\cdot 6t(6))} \\ &= \frac{t(4)+t(6)}{\frac{5}{7}t(5)+\frac{9}{7}t(6)} \\ &\overset{speed-up\ property}{\leq} \frac{t(4)+\frac{2}{3}t(4)}{\frac{4}{7}t(4)+\frac{6}{7}t(4)} \\ &= \frac{7}{6}\,.\end{aligned}$$

If $m \geq 18$, then $\frac{m}{5}+\frac{m}{6} \leq \frac{2}{5}m-\frac{3}{5}$. Thus,

$$T_{opt}(n,m,t) \overset{\text{Lemma 2.9, page 14}}{\geq} t(5)+t(6)\,,$$

and we get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(5)}{t(5)+t(6)} \\ &\overset{speed-up\ property}{\leq} \frac{t(4)+t(5)}{\frac{2}{3}t(4)+t(5)} \\ &\overset{speed-up\ property}{\leq} \frac{\frac{9}{5}t(4)}{\frac{22}{15}t(4)} \\ &= \frac{27}{22} \\ &< \frac{5}{4}\,.\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.26 *Let $\frac{2}{5}m \leq n \leq \lfloor \frac{m}{2} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t) = \min\{t(2), t(3)+t(a)\} .$$

Since $n \geq \frac{2}{5}m$,

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(2)}{\frac{\frac{2}{5}m}{m} \cdot 2t(2)} \\ &= \frac{5}{4} . \end{aligned}$$

This completes the proof of the claim. ■

2.7.3 The Case $\lfloor \frac{m}{4} \rfloor < n \leq \lfloor \frac{m}{3} \rfloor$.

In the following, let $a = \lfloor \frac{m}{n - \lfloor \frac{m}{4} \rfloor} \rfloor$. We have

$$\begin{aligned} a &= \left\lfloor \frac{m}{n - \lfloor \frac{m}{4} \rfloor} \right\rfloor \\ &\overset{n \leq \frac{m}{3}}{\geq} \left\lfloor \frac{m}{\frac{m}{3} - \lfloor \frac{m}{4} \rfloor} \right\rfloor \\ &\geq \left\lfloor \frac{m}{\frac{m}{3} - \frac{m}{4} + \frac{3}{4}} \right\rfloor \\ &= \left\lfloor \frac{m}{\frac{m}{12} + \frac{3}{4}} \right\rfloor \\ &\overset{m \geq 5 \text{ (Observation 2.15, page 27)}}{\geq} \left\lfloor \frac{5}{\frac{5}{12} + \frac{3}{4}} \right\rfloor \\ &= 4 . \end{aligned}$$

Lemma 2.27 *Let $\lfloor \frac{m}{4} \rfloor < n \leq 2\lfloor \frac{m}{7} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) = \min\{t(3), t(4)+t(a), 2t(7)\} .$$

If $t(a) \leq \frac{1}{4}t(4)$, then the trivial lower bound $t(4)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(a)}{t(4)} \\ &\overset{t(a) \leq \frac{1}{4}t(4)}{\leq} \frac{t(4)+\frac{1}{4}t(4)}{t(4)} \\ &= \frac{5}{4} . \end{aligned}$$

So let $t(a) > \frac{1}{4}t(4)$, and consider the makespan of an optimum schedule.
First case: There exist no more than $\lfloor \frac{m}{4} \rfloor$ jobs each being executed on at most a processors. Since $n \leq 2\lfloor \frac{m}{7} \rfloor$ and $n \geq 3$, we have $m \geq 14$. If $m = 14$, then

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.17, page 28}}{\leq} \frac{t(4)+t(a)}{\left(1-\frac{2}{14}\right)t(4)+\left(1+\frac{1}{14}\right)\cdot\frac{a}{a+1}\cdot t(a)} \\
&\overset{a\geq 4}{\leq} \frac{t(4)+t(a)}{\left(1-\frac{1}{7}\right)t(4)+\left(1+\frac{1}{14}\right)\cdot\frac{4}{5}\cdot t(a)} \\
&= \frac{t(4)+t(a)}{\frac{6}{7}t(4)+\frac{6}{7}t(a)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

Otherwise, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.17, page 28}}{\leq} \frac{t(4)+t(a)}{\left(1-\frac{3}{m}\right)t(4)+\left(1+\frac{1}{m}\right)\cdot\frac{a}{a+1}\cdot t(a)} \\
&\overset{m\geq 15}{\leq} \frac{t(4)+t(a)}{\left(1-\frac{1}{5}\right)t(4)+\left(1+\frac{1}{15}\right)\cdot\frac{a}{a+1}\cdot t(a)} \\
&\overset{a\geq 4}{\leq} \frac{t(4)+t(a)}{\left(1-\frac{1}{5}\right)t(4)+\left(1+\frac{1}{15}\right)\cdot\frac{4}{5}\cdot t(a)} \\
&= \frac{t(4)+t(a)}{\frac{4}{5}t(4)+\frac{16}{15}\cdot\frac{4}{5}\cdot t(a)} \\
&< \frac{5}{4}\,.
\end{aligned}$$

Second case: There exist at least $\lfloor \frac{m}{4} \rfloor + 1$ jobs each being executed on at most a processors. Clearly, we only have to consider dominating schedules for which $t(a)$ is the smallest addend which appears in the makespan. Since $t(a) > \frac{1}{4}t(4)$, we have $T_{opt}(n,m,t) \geq 4t(a) > \frac{3}{4}t(4) + t(a)$ for all makespans of a schedule with more than 3 addends. This yields

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(a)}{\frac{3}{4}t(4)+t(a)} \\
&\overset{t(a)>\frac{1}{4}t(4)}{<} \frac{t(4)+\frac{1}{4}t(4)}{\frac{3}{4}t(4)+\frac{1}{4}t(4)} \\
&= \frac{5}{4}\,.
\end{aligned}$$

Hence, we only have to consider makespans with at most 3 addends. The dominating schedules are those corresponding to the makespans $t(3)$, $t(4)+t(a)$, $t(5)+t(a)$ for $a \leq 9$, $t(6)+t(a)$ for $a \leq 11$, $t(7)+t(a)$ for $a \leq 9$, $t(8)+2t(a)$, $t(9)+t(13)+t(a)$, and $t(11)+t(13)+t(16)$ (see Table 2.2, page 27).
$t(3)$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(3)}{t(3)} = 1\,.$$

$\underline{t(4)+t(a)}$: We immediately get

$$\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} \leq \frac{t(4)+t(a)}{t(4)+t(a)} = 1\,.$$

$\underline{t(5)+t(a)}$: We get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(a)}{t(5)+t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(4)+t(a)}{\frac{4}{5}t(4)+t(a)}\\ &< \frac{5}{4}\,.\end{aligned}$$

$\underline{t(6)+t(a)}$: The fact that $a \leq 11$ yields

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(7)}{t(6)+t(a)}\\ &\overset{t(6)\geq t(7)}{\leq} \frac{2t(7)}{t(7)+t(a)}\\ &\overset{a\leq 11,\ t(a)\geq t(11)}{\leq} \frac{2t(7)}{t(7)+t(11)}\\ &\overset{speed-up\ property}{\leq} \frac{2t(7)}{t(7)+\frac{7}{11}t(7)}\\ &= \frac{11}{9}\\ &< \frac{5}{4}\,.\end{aligned}$$

$\underline{t(7)+t(a)}$: Since $a \leq 9$ we have

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(7)}{t(7)+t(a)}\\ &\overset{a\leq 9,\ t(a)\geq t(9)}{\leq} \frac{2t(7)}{t(7)+t(9)}\\ &\overset{speed-up\ property}{\leq} \frac{2t(7)}{t(7)+\frac{7}{9}t(7)}\\ &= \frac{9}{8}\,.\end{aligned}$$

$\underline{t(8)+2t(a)}$: We get

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(a)}{t(8)+2t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(4)+t(a)}{\frac{1}{2}t(4)+2t(a)}\\ &\overset{t(a)>\frac{1}{4}t(4)}{<} \frac{t(4)+\frac{1}{4}t(4)}{\frac{1}{2}t(4)+\frac{1}{2}t(4)}\\ &= \frac{5}{4}\,.\end{aligned}$$

$\underline{t(9)+t(13)+t(a)}$: We have

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(4)+t(a)}{t(9)+t(13)+t(a)}\\ &\overset{speed-up\ property}{\leq} \frac{t(4)+t(a)}{(\frac{4}{9}+\frac{4}{13})t(4)+t(a)}\\ &\overset{t(a)>\frac{1}{4}t(4)}{<} \frac{t(4)+\frac{1}{4}t(4)}{\frac{88}{117}t(4)+\frac{1}{4}t(4)}\\ &= \frac{5}{4}\cdot\frac{468}{469}\,.\end{aligned}$$

$\underline{t(11)+t(13)+t(16)}$: It follows that

$$\begin{aligned}\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{2t(7)}{t(11)+t(13)+t(16)}\\ &\overset{speed-up\ property}{\leq} \frac{2t(7)}{(\frac{7}{11}+\frac{7}{13}+\frac{7}{16})t(7)}\\ &= \frac{4576}{3689}\\ &< \frac{5}{4}\,.\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.28 *Let $2\lfloor\frac{m}{7}\rfloor < n < \frac{2}{7}m$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) \;\leq\; \min\{t(3),t(4)+t(a),t(6)+t(7)\}\,.$$

There exists an n with $2\lfloor\frac{m}{7}\rfloor < n < \frac{2}{7}m$ if and only if $\mathrm{rem}(m,7) \geq 4$. By Observation 2.15 (page 27), we only have to consider $m \geq 5$.

If $\text{rem}(m,7) = 4$, then we have $m \geq 11$, and we can write $n = 2\lfloor \frac{m}{7} \rfloor + 1 = \frac{2}{7}m - \frac{1}{7}$. If there exists a job being executed on 3 processors, then we are done. Otherwise, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(3)}{\frac{n}{m} \cdot 4t(4)} \\
&\overset{n=\frac{2}{7}m-\frac{1}{7}}{=} \frac{t(3)}{\frac{\frac{2}{7}m-\frac{1}{7}}{m} \cdot 4t(4)} \\
&= \frac{t(3)}{\left(\frac{8}{7} - \frac{4}{7m}\right) t(4)} \\
&\overset{m\geq 11}{\leq} \frac{t(3)}{\frac{12}{11} t(4)} \\
&\overset{\text{speed-up property}}{\leq} \frac{t(3)}{\frac{9}{11} t(3)} \\
&= \frac{11}{9} \\
&< \frac{5}{4}.
\end{aligned}$$

If $\text{rem}(m,7) = 5$, then we can write $n = 2\lfloor \frac{m}{7} \rfloor + 1 = \frac{2}{7}m - \frac{3}{7}$. If $m \geq 37$, then this implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(3)}{\frac{n}{m} \cdot 3t(3)} \\
&\overset{n=\frac{2}{7}m-\frac{3}{7}}{=} \frac{t(3)}{\frac{\frac{2}{7}m-\frac{3}{7}}{m} \cdot 3t(3)} \\
&= \frac{t(3)}{\left(\frac{6}{7} - \frac{9}{7m}\right) t(3)} \\
&\overset{m\geq 37}{\leq} \frac{t(3)}{\frac{213}{259} t(3)} \\
&< \frac{5}{4}.
\end{aligned}$$

If $\text{rem}(m,7) = 6$, then we can write $n = 2\lfloor \frac{m}{7} \rfloor + 1 = \frac{2}{7}m - \frac{5}{7}$. If $m \geq 38$, then this implies

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{speed-up property}}{\leq} \frac{t(3)}{\frac{n}{m} \cdot 3t(3)} \\
&\overset{n=\frac{2}{7}m-\frac{5}{7}}{=} \frac{t(3)}{\frac{\frac{2}{7}m-\frac{5}{7}}{m} \cdot 3t(3)} \\
&= \frac{t(3)}{\left(\frac{6}{7} - \frac{15}{7m}\right) t(3)} \\
&\overset{m\geq 38}{\leq} \frac{t(3)}{\frac{213}{266} t(3)} \\
&< \frac{5}{4}.
\end{aligned}$$

So, we only have to prove the claim for $m \leq 36$ and $\text{rem}(m,7) = 5$, and for $m \leq 37$ and $\text{rem}(m,7) = 6$, that is, $m \in \{19,26,33\} \cup \{27,34\}$. Consider the makespan of an optimum schedule.

- $t(3)$ is the largest addend in the makespan: In this case, we are done.
- $t(4)$ is the largest addend in the makespan: Let $b = \min\{m, \lceil \frac{m}{n-\frac{m}{4}} \rceil\}$.
  Since $\frac{m}{4} + \frac{m}{b} \leq n$, Lemma 2.9 (page 14) implies $T_{opt}(n,m,t) \geq t(4) + t(b)$. Note that $m \in \{19,26,33\} \cup \{27,34\}$ yields $a = b = m$. Thus, we are done.
- $t(5)$ is the largest addend in the makespan: Let $\text{rem}(m,7) = 5$, and let $m = 19$. In this case, $n = 5$. If the makespan consists only of one addend, then we are done. If the makespan consists of at least 3 addends, then

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)}{t(5)+2t(19)} \\ &\overset{\textit{speed-up property}}{\leq} \frac{t(3)}{\frac{87}{95}t(3)} \\ &< \frac{5}{4}. \end{aligned}$$

  So, assume that the number of addends in the makespan is two, that is, $T_{opt}(n,m,t) = t(5) + t(c)$. Due to feasibility $\frac{m}{5} + \frac{m}{c} \geq n$. This yields $c \leq \lfloor \frac{m}{n-\frac{m}{5}} \rfloor = \lfloor \frac{19}{5-\frac{19}{5}} \rfloor = 15$. We get

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(3)}{t(5)+t(15)} \\ &\overset{\textit{speed-up property}}{\leq} \frac{t(3)}{\frac{3}{5}t(3)+\frac{1}{5}t(3)} \\ &= \frac{5}{4}. \end{aligned}$$

  If $m \geq 26$, then $\frac{m}{5} + \frac{m}{15} \leq \frac{2}{7}m - \frac{3}{7} = n$. Thus,

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\textit{Lemma 2.9, page 14}}{\leq} \frac{t(5)+t(9)}{t(5)+t(15)} \\ &\overset{\textit{speed-up property}}{\leq} \frac{t(5)+t(9)}{t(5)+\frac{9}{15}t(9)} \\ &\overset{t(9)\leq t(5)}{\leq} \frac{2t(5)}{\frac{24}{15}t(5)} \\ &= \frac{5}{4}. \end{aligned}$$

  Let $\text{rem}(m,7) = 6$. Then we have $m \in \{27,34\}$. Since $m \geq 27$, we have $\frac{m}{5} + \frac{m}{17} \leq$

$\frac{2}{7}m - \frac{5}{7} = n$. We get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.9, page 14}}{\leq} \frac{t(5)+t(11)}{t(5)+t(17)} \\
&\overset{\text{speed-up property}}{\leq} \frac{t(5)+t(11)}{t(5)+\frac{11}{17}t(11)} \\
&\overset{t(11)\leq t(5)}{\leq} \frac{2t(5)}{\frac{28}{17}t(5)} \\
&= \frac{17}{14} \\
&< \frac{5}{4}.
\end{aligned}$$

- $t(6)$ is the largest addend in the makespan: If $\text{rem}(m,7) = 5$, then $m \geq 19$. If $\text{rem}(m,7) = 6$, then $m \geq 27$. In both cases, $n \geq \frac{m}{6} + \frac{m}{11}$, and we get,

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{\text{Lemma 2.9, page 14}}{\leq} \frac{t(6)+t(7)}{t(6)+t(11)} \\
&\overset{\text{speed-up property}}{\leq} \frac{t(6)+t(7)}{t(6)+\frac{7}{11}t(7)} \\
&\overset{t(7)\leq t(6)}{\leq} \frac{2t(6)}{\frac{18}{11}t(6)} \\
&= \frac{11}{9} \\
&< \frac{5}{4}.
\end{aligned}$$

- $t(i)$, $i \geq 7$, is the largest addend in the makespan: If more than $\frac{m}{7}$ jobs are executed on 7 processors, then $T_{opt}(n,m,t) \geq 2t(7)$, and we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(6)+t(7)}{2t(7)} \\
&\overset{\text{speed-up property}}{\leq} \frac{t(6)+\frac{6}{7}t(6)}{\frac{12}{7}t(6)} \\
&= \frac{13}{12}.
\end{aligned}$$

Otherwise, we get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{speed-up\ property}{\leq} \frac{t(6)+t(7)}{\frac{1}{m}\left(\frac{m}{7}\cdot 7t(7)+\left(\frac{2}{7}m-\frac{5}{7}-\frac{m}{7}\right)\cdot 8t(8)\right)} \\
&= \frac{t(6)+t(7)}{t(7)+\left(\frac{8}{7}-\frac{40}{7m}\right)\cdot t(8)} \\
&\overset{m\geq 19}{\leq} \frac{t(6)+t(7)}{t(7)+\frac{16}{19}t(8)} \\
&\overset{speed-up\ property}{\leq} \frac{t(6)+t(7)}{\frac{12}{19}t(6)+t(7)} \\
&\overset{speed-up\ property}{\leq} \frac{t(6)+\frac{6}{7}t(6)}{\frac{12}{19}t(6)+\frac{6}{7}t(6)} \\
&= \frac{247}{198} \\
&< \frac{5}{4}\,.
\end{aligned}$$

This completes the proof of the claim. ■

Lemma 2.29 *Let $\frac{2}{7}m \leq n \leq \lfloor \frac{m}{3} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule **P** with

$$T(n,m,t,\mathbf{P}) = \min\{t(3), t(4)+t(a)\}\,.$$

We get

$$\begin{aligned}
\frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\overset{speed-up\ property}{\leq} \frac{t(3)}{\frac{n}{m}\cdot 3t(3)} \\
&\overset{n\geq \frac{2}{7}m}{\leq} \frac{t(3)}{\frac{\frac{2}{7}m}{m}\cdot 3t(3)} \\
&= \frac{7}{6}\,.
\end{aligned}$$

This completes the proof of the claim. ■

2.7.4 The Case $n \leq \lfloor \frac{m}{4} \rfloor$.

In the following, consider some $k \in \mathbb{N}$, $k \geq 5$, and let $a = \left\lfloor \frac{m}{n - \lfloor \frac{m}{k} \rfloor} \right\rfloor$.

Lemma 2.30 *Let $\lfloor \frac{m}{k} \rfloor < n \leq \lfloor \frac{m}{k-1} \rfloor$. Then,* PPS *computes a phase-by-phase schedule which is an optimum schedule up to the factor $\frac{5}{4}$.*

Proof: In this interval, PPS returns a phase-by-phase schedule $\mathbf{P}$ with

$$T(n,m,t,\mathbf{P}) \quad = \quad \min\{t(k-1), t(k)+t(a)\} .$$

The trivial lower bound $t(k)$ on the makespan of an optimum schedule implies

$$\begin{aligned} \frac{T(n,m,t,\mathbf{P})}{T_{opt}(n,m,t)} &\leq \frac{t(k-1)}{t(k)} \\ &\overset{speed-up\ property}{\leq} \frac{\frac{k}{k-1}\cdot t(k)}{t(k)} \\ &\overset{k\geq 5}{\leq} \frac{5}{4} . \end{aligned}$$

This completes the proof of the claim. ■

2.8 An ε-Approximation Algorithm

In the previous sections, we proved that there exists a constant-time algorithm with an approximation factor of $\frac{5}{4}$. Decker *et al.* [34] showed that if the speed-up is good enough and m is large enough, then an optimum phase-by-phase schedule (which can be computed in $O(m^3)$ time as seen in Theorem 2.4, page 10) approximates an optimum schedule up to a factor of $1+\varepsilon$ for any $\varepsilon > 0$. They proceeded as follows: Let

$$T_k \quad = \quad \min\left\{ \sum_{j\in[k]} r_j \cdot t(j) \;\middle|\; \sum_{j\in[k]} r_j \cdot \frac{m}{j} \geq n \right\} .$$

Note that T_m is a lower bound on the makespan of an optimum schedule (see Lemma 2.8, page 14). For all $k \in [m]$, it is $T_m \geq T_k - t(k)$ (Lemma 2.31). If the speed-up is optimum up to a constant factor, then this yields an approximation algorithm (Theorem 2.32). Combining Theorem 2.3 (page 9), Theorem 2.4 (page 10) and Theorem 2.32 leads to an ε-approximation algorithm: If $m \geq 4 \cdot \frac{c^3}{\varepsilon^6}$, then use the optimum phase-by-phase algorithm, otherwise use the optimum algorithm from Theorem 2.3 (page 9). Theorem 2.32 can be generalized to other time functions fulfilling $t(j) \to 0$ for $j \to \infty$.

Lemma 2.31 (Decker *et al.*[34]) *$T_m \geq T_k - t(k)$ for all $k \in [m]$.*

Theorem 2.32 (Decker *et al.*[34]) *For all $j_1, j_2 \in [m]$, let $t(j_1 \cdot j_2) \leq \frac{c}{j_1} \cdot t(j_2)$ for some constant c. Then, the optimum phase-by-phase schedule is optimum up to a factor of $1 + \sqrt[6]{\frac{4c^3}{m}}$.*

2.9 Conclusion and Directions for Further Research

We considered the problem of finding a non-preemptive schedule for independent malleable identical jobs on identical processors with minimum makespan. We assumed that the same

properties for the execution time as in [13] hold. This implies that the execution of the jobs achieves some speed-up, but no super-linear speed-up.

We have seen that we can compute an optimum schedule with execution time exponential in the number of processors. This yields to an algorithm polynomial in the number of jobs if the number of processors is constant. In order to approximate an optimum schedule, we introduced phase-by-phase schedules. We illustrated with help of an example that the quotient of the makespan of an optimum phase-by-phase schedule and the makespan of an optimum schedule can be $\frac{5}{4}$. Furthermore, we gave a constant time approximation algorithm which only uses certain phase-by-phase schedules matching this bound. Finally, we observed that there exists an ε-approximation algorithm in the case that the speed-up is optimum up to a constant factor.

Though we gave a thorough analysis of the considered scheduling problem, some of the fundamental problems still remain tantalizingly open:

- Is it possible to significantly improve the approximation factor by adding a constant number of non-phase-by-phase schedules?
- Which class of instances is easy to optimize? Clearly, the existence of an ε-approximation algorithm in case of time functions with near-optimum speed-up indicates that approximating an optimum schedule is easy for this class of instances. However, in general we can not give an answer to this question.
- What is the time complexity of computing an optimum schedule? Is it possible to compute an optimum schedule in polynomial time, or is there no such algorithm?

— 3 —

Flow Scheduling

Für den Unwissenden ist alles möglich.

Christoph Martin Wieland (1733–1813)

3.1 Introduction

3.1.1 Motivation and Framework

Load balancing is an essential task for the efficient use of parallel computer systems. In many parallel applications, the work loads have dynamic behavior and may change dramatically during runtime. To achieve an efficient use of the parallel computer system, the work load has to be balanced among the processors during runtime. Clearly, the balancing scheme is required to be highly efficient itself in order to ensure an overall benefit.

One major field of application are parallel adaptive finite element simulations where a geometric space, discretized using a mesh, is partitioned into sub-regions. The computation proceeds on the mesh elements in each sub-region independently [52]. As the computation is carried out, the mesh refines and coarsens, depending on the problem characteristics such as turbulence or shocks (in the case of fluid dynamics simulations, for example). Thus, the size of the sub-regions (in terms of the numbers of elements) has to be balanced. The problem of parallel finite element simulation has been extensively studied – see the text book [52] for an excellent selection of applications, case studies and references.

Much work has been done on the topic of *load balancing*. The approaches depend on the model used to describe the interprocessor communication. We consider synchronous distributed processor networks. In each round, a processor of the network can send and receive messages to/from all its neighbors simultaneously. Furthermore, we assume that the situation is static, i.e., no load is generated or consumed during the balancing process, and the network does not change. In order to balance the network, it is necessary to migrate parts of the processors' loads during runtime. We assume that the load consists of independent load units,

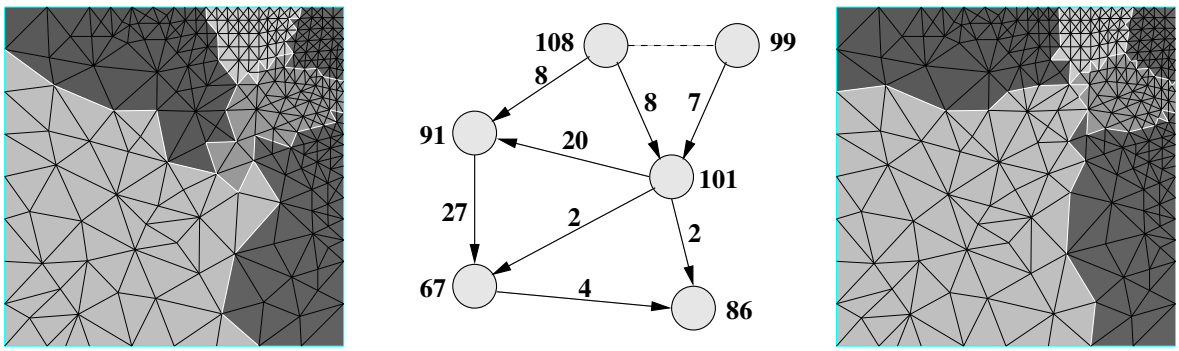


Figure 3.1: An unbalanced partition of a mesh (left hand side), the balancing flow graph (center), and the obtained balanced partition (right hand side).

called *tokens*. One possible approach to balance the network is based on a two-step approach (illustrated with help of an example in Figure 3.1):

(1.) First, a balancing *flow graph* is calculated, putting the network into a balanced state where all processors keep the same load up to one token. The flow graph is represented by a directed acyclic graph $G = (V,E)$ with $|V|$ vertices and $|E|$ edges, combined with a *load function* δ and a *flow function* f.

(2.) Second, the load items are *migrated* according to this flow graph. The goal is to use the minimum number of rounds to reach the balanced state.

In this chapter, we consider step (2.) of this approach.

3.1.2 Contribution

We obtain through a thorough analysis the following results:

- For every distributed scheduling strategy, there exists a flow graph on which this strategy requires at least $\frac{3}{2}$ times the minimum number of rounds (Theorem 3.1, page 63).
- We present a distributed algorithm for flow graphs in tree networks. In contrast to the known local greedy algorithms, this algorithm investigates the structure of the flow graph before sending tokens. We show that the algorithm requires at most twice the minimum number of rounds, and we show that this bound is tight (Theorem 3.4, page 69). To the best of our knowledge, this is the first distributed flow scheduling algorithm (even though for a *restricted* class of flow graphs) which is optimum up to a constant factor.

3.1.3 Related Work

Computing a balancing flow graph can be done efficiently with help of *diffusion* (see [26] for detailed description), yielding a balancing flow graph optimum with respect to the l_2-norm [35, 41].

The migration of items according to a balancing flow has been considered in [35, 36, 144]. The goal to use the minimum number r of rounds to reach the balanced state trivially leads

to a formulation as a linear program with $r(|E|+|V|)$ unknowns and $|E|+2r|V|$ equations. Such a system is solvable in $O(r^5(|E|+|V|)^5)$ rounds using Karmarkar's algorithm [137]. Diekmann *et. al.* [35] showed that the $(r-1)$-commodity flow problem in bipartite graphs can be reduced to the problem of deciding whether there exists a schedule for a given flow graph requiring at most r rounds.

Diekmann *et al.* [35] introduced a special class of distributed algorithms, called *local greedy flow scheduling algorithms*. Here, the algorithm determines for each vertex and each round how much of the available load to send to which of the outgoing edges. Diekmann *et al.* [35] defined local greedy heuristics as follows:

(1.) The scheduling only depends on local information about the flow and the available load.

(2.) If in a certain round a vertex contains enough load to fulfill all outgoing edges, then it immediately saturates all of them.

(3.) If a vertex does not contain enough load, then it distributes all available load to its outgoing edges according to some tie-breaking.

Moreover, Diekmann *et al.* [35] introduced two memory-less local greedy algorithms where the decision only depends on the current situation and not on the history: ROUND-ROBIN GREEDY chooses a subset of edges which are filled up to saturation (one edge of the subset might not be saturated completely); PROPORTIONAL GREEDY sends load via all edges of a vertex in parallel, and the amount of load is chosen proportional to the current demand of the edges. Diekmann *et al.* [35] showed that ROUND-ROBIN GREEDY is $\Theta(\sqrt{|V|})$-optimum whereas the factor of PROPORTIONAL GREEDY lies in between $\Omega(\log|V|)$ and $O(\sqrt{|V|})$.

3.1.4 Organization

The rest of this chapter is organized as follows. After a formal introduction of our model in Section 3.2, we show that for every distributed scheduling strategy there exists a flow graph on which this strategy requires at least $\frac{3}{2}$ times the minimum number of rounds in Section 3.3. In Section 3.4, we introduce and analyze a distributed algorithm for flow graphs in tree networks. We close, in Section 3.5, with a discussion of our results and some open problems.

3.2 Model

For all $k \in \mathbb{N}$, we denote $[k] = \{1,\dots,k\}$.

3.2.1 Network

We consider a *synchronous processor network*. In order to describe this network formally, we use the model defined in [105]. The **network** $N = (V,C)$ consists of $|V|$ **processors** and $|C|$ undirected **channels**. Execution of the entire system begins with all processors in arbitrary start states, and all channels empty. Then, the processors, in lock-step, repeatedly perform the following two steps:

(1.) Generate the messages to be sent to the neighbors. Put these messages in the appropriate channels.

(2.) Compute the new state from the current state and the incoming messages. Remove all messages from the channels.

A **round** is the combination of these two steps.

For our distributed algorithm in Section 3.4, we restrict our discussion to **tree networks**. Furthermore, we assume that the network is **rooted**, that is, exactly one processor is assigned to be the root of the tree, and each other processor v knows its **parent** with respect to this root, denoted $\mathsf{parent}(v)$ (if the processor network is not rooted, then parents can be determined successively, starting at the leaves; the number of required rounds is bounded by half the diameter of N).

3.2.2 Flow Graph

The load situation in N is given by the **load function** $\delta : V \to \mathbb{N}_0$, representing the number of unit sized **tokens** on the processors. A **flow network** is a directed acyclic subgraph $G = (V,E)$ of N with $|E|$ **edges**. Denote $s : E \to V$ and $t : E \to V$ functions, defining source and target of each edge. The flow network G is a **directed tree** if each edge of G is directed away from the root of N, i.e., $s(e) = \mathsf{parent}(t(e))$ for each $e \in E$. For every $v \in V$, denote

$$\begin{aligned} \text{in}(v) &= \{e \in E \mid t(e) = v\}, \\ \text{out}(v) &= \{e \in E \mid s(e) = v\} \end{aligned}$$

the **set of incoming edges** and the **set of outgoing edges** of v, respectively. A **flow** is a function $f : E \to \mathbb{N}$. For every $v \in V$, the **flow property** holds, that is,

$$\delta(v) + \sum_{e \in \text{in}(v)} f(e) \;\geq\; \sum_{e \in \text{out}(v)} f(e)\,.$$

A **flow graph F** is a triple (G, δ, f).

3.2.3 Schedule

A **schedule** $\mathbf{S} = (G, \delta, f)$ for a flow graph $\mathbf{F}$ is a decomposition of the flow f into flows f^i, determining the flow in round $i \in [r_{\mathbf{S}}(\mathbf{F})]$, where $r_{\mathbf{S}}(\mathbf{F})$ is the number of rounds required by $\mathbf{S}$. Thus, $\mathbf{S} = (f^1, \ldots, f^{r_{\mathbf{S}}(\mathbf{F})})$ with

$$f(e) \;=\; \sum_{i \in [r_{\mathbf{S}}(\mathbf{F})]} f^i(e)$$

for all $e \in E$, and

$$\delta(v) + \sum_{j \in [i-1]} \sum_{e \in \text{in}(v)} f^j(e) \;\geq\; \sum_{j \in [i]} \sum_{e \in \text{out}(v)} f^j(e) \tag{3.1}$$

for all $v \in V$ and $i \in [r_{\mathbf{S}}(\mathbf{F})]$. Inequality (3.1) ensures that every processor has sufficient load in each round $i \in [r_{\mathbf{S}}(\mathbf{F})]$ to fulfill f^i on its outgoing edges.

Given a flow graph $\mathbf{F}$, each processor initially knows its parent, its load, and the flow on its ingoing and outgoing edges, respectively. The problem of (distributedly) finding a schedule $\mathbf{S}$ requiring the minimum number of rounds $r_{opt}(\mathbf{F})$ is called **flow scheduling problem**.

3.3 General Distributed Scheduling Strategies

We now show that for every distributed scheduling strategy there exists a flow graph on which this strategy requires at least $\frac{3}{2}$ times the minimum number of rounds. Note that this result is a lower bound on the worst-case performance of *all* distributed scheduling strategies. For a *certain* strategy, there might exist a flow graph on which it performs much worse.

Theorem 3.1 *Let $d \in \mathbb{N}$ with $d \geq 2$. Then, for every distributed scheduling strategy, there exists a flow graph $\mathbf{F} = (G, \delta, f)$ with directed tree G, $\deg(G) = d+1$, on which this strategy requires at least $\frac{3}{2} \cdot r_{opt}(\mathbf{F})$ rounds.*

Proof: Fix any $d \in \mathbb{N}$ with $d \geq 2$, and let k be a power of d. Consider the following balancing flow graph $\mathbf{F} = (G, \delta, f)$ (illustrated in Figure 3.2):

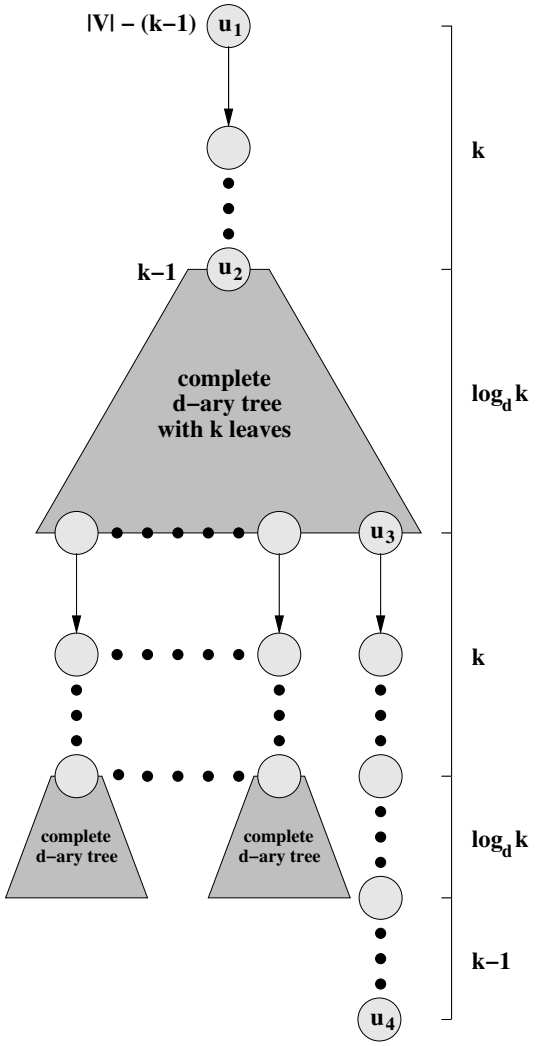


Figure 3.2: Flow graph $\mathbf{F}$ used in the proof of Theorem 3.1. The terms on the left of vertex u_1 and vertex u_2 give their initial loads. The expressions on the right hand side of the graph denote the lengths of the paths or depths of the trees.

- The graph G consists of a path of length k from vertex u_1 to vertex u_2, followed by a complete d-ary tree of depth $\log_d k$ with root u_2 and k leaves. To each of the first $k-1$ leaves, a path of length k is attached whose last vertex is the root of a complete d-ary tree of depth $\log_d k$. To the remaining leaf u_3, a path of length $2k+\log_d k-1$ is attached. Clearly, G is a directed tree with $\deg(G) = d+1$.

- The load on the processors is

$$\delta(v) \quad = \quad \begin{cases} |V|-(k-1) & \text{if } v = u_1, \\ (k-1) & \text{if } v = u_2, \\ 0 & \text{otherwise.} \end{cases}$$

  So, only vertices u_1 and u_2 have non-zero load. Clearly, the total number of tokens is $|V|$. Thus, in a balanced state, all vertices have load 1.

- The flow f is uniquely determined and therefore also l_2-optimum since the network is a tree.

We proceed as follows: We first derive an optimum schedule for the given flow graph **F**. We then prove a lower bound on the number of rounds, required in the worst-case by any distributed scheduling strategy.

(1.) Clearly, the minimum number of migration rounds is attained if the initial load of vertex u_2 is sent toward the path below u_3. At the same time u_1's load is sent downward. After $2k+2\log_d k$ rounds each processor holds one token, and the migration phase is finished. Thus,

$$r_{opt}(\mathbf{F}) \quad = \quad 2k+2\log_d k\,.$$

(2.) When using a distributed scheduling strategy, each vertex initially only knows its own load and the flow on its incident edges. Among all vertices with distance at most $k+\log_d k$ to u_2, the incoming edges of those being at the same distance to u_2 have equal flow. Hence, all subtrees below each inner vertex of the complete tree with root u_2 are equal up to a depth of k. Thus, during the first k rounds these vertices cannot gather information that helps to decide in which direction to send the load they might get.

As no helpful information is available during the first k rounds, we can assume that u_3 is the leaf to which the least amount of load has been sent. The only source of load in a distance of at most k from the k leaves is u_2 with load $k-1$. Hence, the least amount of load cannot exceed $\frac{k-1}{k} < 1$. Thus, in the worst-case none of the tokens reaches the path at u_3 during the first k rounds of any distributed scheduling algorithm, and the path has to be filled up by tokens from u_1. One of these tokens has to travel a distance of $3k+2\log_d k-1$ (from u_1 to u_4, the last vertex of the path). Hence, the number of rounds required by a worst-case schedule **S** is at least

$$r_{\mathbf{S}}(\mathbf{F}) \quad \geq \quad 3k+2\log_d k-1\,.$$

Combining these bounds, we get

$$\frac{r_{\mathbf{S}}(\mathbf{F})}{r_{opt}(\mathbf{F})} \quad \geq \quad \frac{3k+2\log_d k-1}{2k+2\log_d k}\,,$$

and this ratio converges to $\frac{3}{2}$ as k increases. ■

3.4 The Distributed Algorithm

We proceed by giving a distributed scheduling algorithm for flow graphs in tree networks. We start with a distributed algorithm for flow graphs with directed tree G in Subsection 3.4.1. In Subsection 3.4.2 (page 69), we then use this result to give a distributed algorithm for arbitrary trees G.

3.4.1 Directed Trees

Consider a flow graph $\mathbf{F} = (G, \delta, f)$ with directed tree G. For every $e \in E$, we recursively define for all $i \in \mathbb{N}_0$

$$d_i(e) = \begin{cases} f(e) & \text{if } i = 0, \\ \max\left\{ \sum_{\tilde{e} \in \text{out}(t(e))} d_{i-1}(\tilde{e}) - \delta(t(e)), 0 \right\} & \text{otherwise.} \end{cases} \tag{3.2}$$

Note that $d_i(e)$ is the number of tokens that have to be sent along edge e to vertices at distance of at least i from vertex $t(e)$.

Theorem 3.2 *Let $\mathbf{F} = (G, \delta, f)$ be a flow graph with directed tree G, and let $k \in \mathbb{N}_0$ be minimum such that $d_k(e) = 0$ for all $e \in E$. Then, $r_{opt}(\mathbf{F}) = k$.*

Proof: We proceed as follows. We first show that there exists a schedule using k rounds. We then prove that any schedule requires at least k rounds.

(1.) There exists a schedule using k rounds.
To prove the existence of a schedule using k rounds, we construct a schedule $\mathbf{S} = (f^1, \ldots, f^k)$ as follows. In round $i \in [k]$, every vertex $v \in V$ sends

$$f^i(\tilde{e}) = d_{k-i}(\tilde{e}) - d_{k-i+1}(\tilde{e}) \tag{3.3}$$

tokens along each edge $\tilde{e} \in \text{out}(v)$. After i rounds,

$$\begin{aligned} \sum_{j \in [i]} f^j(\tilde{e}) &= \sum_{j \in [i]} \left(d_{k-j}(\tilde{e}) - d_{k-j+1}(\tilde{e}) \right) \\ &= d_{k-i}(\tilde{e}) \end{aligned}$$

tokens have been sent along edge $\tilde{e}$ for all $\tilde{e} \in E$. Denote $\delta^i(v)$ the load on vertex v before round i, and denote e the (unique) incoming edge of v. For all $i \in [k]$, we have

$$\begin{aligned} \delta_i(v) &= \delta(v) + d_{k-(i-1)}(e) - \sum_{\tilde{e} \in \text{out}(v)} d_{k-(i-1)}(\tilde{e}) \\ &\overset{(3.2)}{=} \delta(v) + \max\left\{ \sum_{\tilde{e} \in \text{out}(v)} d_{k-(i-1)-1}(\tilde{e}) - \delta(v), 0 \right\} - \sum_{\tilde{e} \in \text{out}(v)} d_{k-(i-1)}(\tilde{e}) \\ &\geq \sum_{\tilde{e} \in \text{out}(v)} \left(d_{k-(i-1)-1}(\tilde{e}) - d_{k-(i-1)}(\tilde{e}) \right) \\ &= \sum_{\tilde{e} \in \text{out}(v)} \left(d_{k-i}(\tilde{e}) - d_{k-i+1}(\tilde{e}) \right) \\ &\overset{(3.3)}{=} \sum_{\tilde{e} \in \text{out}(v)} f^i(\tilde{e}) \, . \end{aligned}$$

This shows that in every round $i \in [k]$, all vertices have enough load to fulfill the flow f^i on their outgoing edges. Hence, the described schedule is feasible. Clearly, it requires k rounds.

(2.) Any schedule requires at least k rounds.
We show by induction on $r_{opt}(\mathbf{F})$, $r_{opt}(\mathbf{F}) \geq 1$, that $d_{r_{opt}(\mathbf{F})}(e) = 0$ for all $e \in E$. By definition of k, this implies $r_{opt}(\mathbf{F}) \geq k$. As our basis case, let $r_{opt}(\mathbf{F}) = 1$. Clearly, every vertex has sufficient load to fulfill all its outgoing edges at once. This implies that $d_1(e) = 0$ for all $e \in E$, showing that the claim holds for the basis case.

For the induction step, let $r_{opt}(\mathbf{F}) \geq 2$, and assume that the claim holds for all flow graphs $\mathbf{F}'$ with $r_{opt}(\mathbf{F}') < r_{opt}(\mathbf{F})$. Let $\mathbf{S} = (f^1, \ldots, f^{r_\mathbf{S}(\mathbf{F})})$ be a schedule with $r_\mathbf{S}(\mathbf{F}) = r_{opt}(\mathbf{F})$, and let $f' = f^1$ and

$$f'' = \sum_{2 \leq i \leq r_{opt}(\mathbf{F})} f^i .$$

Clearly, $f(e) = f'(e) + f''(e)$ for all $e \in E$. For all $v \in V$, denote

$$\delta'(v) = \delta(v) + \sum_{\tilde{e} \in \text{in}(v)} f'(\tilde{e}) - \sum_{\tilde{e} \in \text{out}(v)} f'(\tilde{e}) \tag{3.4}$$

the load on v after the first round, and let $\mathbf{F}' = (G, \delta, f')$ and $\mathbf{F}'' = (G, \delta', f'')$. By induction hypothesis, applying the schedule construction to the flow graphs $\mathbf{F}'$ and $\mathbf{F}''$ yields schedules $\mathbf{S}'$ and $\mathbf{S}''$ and, for all $e \in E$, values $d'_i(e)$ and $d''_i(e)$, $i \geq 0$, with

$$d'_1(e) = 0 , \tag{3.5}$$
$$d''_{r_{opt}(\mathbf{F})-1}(e) = 0 . \tag{3.6}$$

We now show by induction on $i \in [r_{opt}(\mathbf{F})] \cup \{0\}$ that applying the schedule construction to $\mathbf{F}$ yields

$$d_i(e) \leq \begin{cases} d''_i(e) + f'(e) & \text{if } 0 \leq i \leq r_{opt}(\mathbf{F}) - 1, \\ 0 & \text{otherwise,} \end{cases}$$

for all $e \in E$. As our basis case, let $i = 0$. We have

$$\begin{aligned} d_0(e) &= f(e) \\ &= f'(e) + f''(e) \\ &= f'(e) + d''_0(e) , \end{aligned}$$

which proves that the claim holds for the basis case. For the induction step, let $i \in$

$[r_{opt}(\mathbf{F})]$, and assume that the claim holds for $(i-1)$. If $i \leq r_{opt}(\mathbf{F}) - 1$, then we get

$$\begin{aligned}
d_i(e) &= \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} d_{i-1}(\tilde{e}) - \delta(t(e)), 0\right\} \\
&\overset{\textit{Induction}}{\leq} \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} \left(d''_{i-1}(\tilde{e}) + f'(\tilde{e})\right) - \delta(t(e)), 0\right\} \\
&\overset{(3.4),\ \textit{page}\ 66}{=} \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} \left(d''_{i-1}(\tilde{e}) + f'(\tilde{e})\right) \right. \\
&\qquad\qquad \left. - \left(\delta'(t(e)) - f'(e) + \sum_{\tilde{e}\in\text{out}(t(e))} f'(\tilde{e})\right), 0\right\} \\
&= \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} d''_{i-1}(\tilde{e}) - \delta'(t(e)) + f'(e), 0\right\} \\
&\overset{(3.2),\ \textit{page}\ 65}{\leq} \max\left\{d''_i(e) + f'(e), 0\right\} \\
&= d''_i(e) + f'(e)
\end{aligned}$$

for all $e \in E$. Otherwise,

$$\begin{aligned}
d_{r_{opt}(\mathbf{F})}(e) &= \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} d_{r_{opt}(\mathbf{F})-1}(\tilde{e}) - \delta(t(e)), 0\right\} \\
&\overset{\textit{Induction}}{\leq} \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} \left(d''_{r_{opt}(\mathbf{F})-1}(\tilde{e}) + f'(\tilde{e})\right) - \delta(t(e)), 0\right\} \\
&\overset{(3.6),\ \textit{page}\ 66}{=} \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} f'(\tilde{e}) - \delta(t(e)), 0\right\} \\
&= \max\left\{\sum_{\tilde{e}\in\text{out}(t(e))} d'_0(\tilde{e}) - \delta(t(e)), 0\right\} \\
&\overset{(3.2),\ \textit{page}\ 65}{=} d'_1(e) \\
&\overset{(3.5),\ \textit{page}\ 66}{=} 0\,.
\end{aligned}$$

for all $e \in E$. Since $d_{r_{opt}(\mathbf{F})} \geq 0$, this implies $d_{r_{opt}(\mathbf{F})} = 0$, proving the inductive claim. ■

We now use Theorem 3.2 (page 65) to give a distributed algorithm SCHEDULEDIRECTEDTREES, stated as Algorithm 2 (page 68), which (implicitly) computes a schedule for flow graphs $\mathbf{F} = (G, \delta, f)$ with directed tree G. The algorithm works in two phase:

(1.) In the first phase, every vertex $v \in V$ keeps a variable i initialized with 0. Then, v successively sends the values $d_j(e)$, $j \in [k_e]$, along its incoming edge $e = (\mathsf{parent}(v), v)$ to its parent $\mathsf{parent}(v)$, where k_e is minimum with $d_{k_e}(e) = 0$. Clearly, $d_1(e)$ can be computed from the flows on the outgoing edges, and $d_j(e)$, $j \in [k_e] \setminus \{1\}$, can be computed

Algorithm 2 (SCHEDULEDIRECTEDTREES on vertex $v \in V$)

Input: a flow graph $\mathbf{F} = (G, \delta, f)$ with directed tree G

```
(1)   begin
(2)   e ← (parent(v), v);
      // phase 1
(3)   j ← 0;
(4)   d_0(e) ← f(e);
(5)   while d_j(e) ≠ 0 do
(6)         j ← j + 1;
(7)         d_j(e) ← max{ Σ_{ẽ∈out(t(e))} d_{j−1}(ẽ) − δ(t(e)), 0 };
(8)         send d_j(e) along e;
(9)         receive d_j(ẽ) from all ẽ ∈ out(v);
(10)  k_e ← j;
      // phase 2
(11)  i ← 0;
(12)  while i ≤ k_e do
(13)        if δ(v) ≥ Σ_{ẽ∈out(v)} (d_{k_e−i−1}(ẽ) − d_{k_e−i}(ẽ)) then
(14)              send d_{k_e−i−1}(ẽ) − d_{k_e−i}(ẽ) tokens along each ẽ ∈ out(v);
(15)              i ← i + 1;
(16)        receive tokens from all ẽ ∈ out(v);
(17)        update δ(v);
(18)  end
```

in round j from the values received in the previous round $(j-1)$. The value $d_j(\tilde{e})$ of an outgoing edge $\tilde{e}$ is assumed to be 0 if no information has been sent along $\tilde{e}$. By Theorem 3.2 (page 65), we have $k_e \leq r_{opt}(\mathbf{F})$ for all $e \in E$.

(2.) In the second phase, v sends $d_{k_e-i-1}(\tilde{e}) - d_{k_e-i}(\tilde{e})$ tokens via each of its outgoing edges $\tilde{e} \in \text{out}(v)$ and increments i if the number of tokens on v in the current round is at least

$$\sum_{\tilde{e} \in out(v)} \left(d_{k_e-i-1}(\tilde{e}) - d_{k_e-i}(\tilde{e})\right) .$$

Otherwise, no tokens are sent. This step is repeated until $i = k_e$.

We now prove that the schedule (implicitly) computed by SCHEDULEDIRECTEDTREES requires at most $2r_{opt}(\mathbf{F})$ rounds.

Corollary 3.3 *Consider a flow graph $\mathbf{F} = (G, \delta, f)$ with directed tree G. Then,* SCHEDULEDIRECTEDTREES *(implicitly) computes a schedule for $\mathbf{F}$ requiring at most $2r_{opt}(\mathbf{F})$ rounds.*

Proof: Let $k = \max_{e \in E} k_e$. Fix an arbitrary $v \in V$, and denote i_r the value of i after r rounds of SCHEDULEDIRECTEDTREES on v. Clearly, v has sent an overall amount of $d_{k_e-i_r}(\tilde{e})$ tokens along each edge $\tilde{e} \in \text{out}(v)$ after r rounds.

We now show by induction on r, $r \geq 0$, that $k_e - i_{k+r} \leq k - r$. Since $d_j(\tilde{e})$, $j \in [k_e] \cup \{0\}$, is monotonically decreasing in j for all $\tilde{e} \in E$, this implies $d_{k_e-i_{k+r}}(\tilde{e}) \geq d_{k-r}(\tilde{e})$, showing

that SCHEDULEDIRECTEDTREES delays the optimum schedule by at most k rounds. Since Theorem 3.2 (page 65) implies $k \leq r_{opt}(\mathbf{F})$, this proves the claim.

As our basis case, let $r = 0$. In this case, no load has been sent, showing that the claim holds for the basis case. For the induction step, let $r \geq 0$ and assume that the claim holds for r. If $k_e - i_{k+r} < k - r$, then we immediately get $k_e - i_{k+(r+1)} \leq k - (r+1)$, proving the claim. So, assume $k_e - i_{k+r} = k - r$. We proceed by showing that in this case $i_{k+(r+1)} = i_{k+r} + 1$. Clearly, the load $\delta'(v)$ kept by vertex v after $k+r$ rounds is

$$\begin{aligned}
\delta'(v) &= \delta(v) + d_{k_e - i_{k+r}}(e) - \sum_{\tilde{e} \in out(v)} d_{k_e - i_{k+r}}(\tilde{e}) \\
&\overset{(3.2),\ page\ 65}{=} \max\left\{ \sum_{\tilde{e} \in out(v)} d_{k_e - i_{k+r} - 1}(\tilde{e}), \delta(v) \right\} - \sum_{\tilde{e} \in out(v)} d_{k_e - i_{k+r}}(\tilde{e}) \\
&\geq \sum_{\tilde{e} \in out(v)} d_{k_e - i_{k+r} - 1}(\tilde{e}) - \sum_{\tilde{e} \in out(v)} d_{k_e - i_{k+r}}(\tilde{e}) \\
&= \sum_{\tilde{e} \in out(v)} \left(d_{k_e - i_{k+r} - 1}(\tilde{e}) - d_{k_e - i_{k+r}}(\tilde{e}) \right) .
\end{aligned}$$

Thus, vertex v has sufficient load to send $d_{k_e - i_{k+r} - 1}(\tilde{e}) - d_{k_e - i_{k+r}}(\tilde{e})$ tokens via each of its outgoing edges $\tilde{e} \in \text{out}(v)$. Therefore, $i_{k+(r+1)} = i_{k+r} + 1$, as needed. ■

3.4.2 Arbitrary Trees

We now show how SCHEDULEDIRECTEDTREES can be used to give a distributed algorithm for flow graphs $\mathbf{F} = (G, \delta, f)$ with arbitrary tree G.

Theorem 3.4 *Let $\mathbf{F} = (G, \delta, f)$ be a flow graph with arbitrary tree G. Then, there exists a distributed algorithm which (implicitly) computes a schedule for $\mathbf{F}$ requiring at most $2r_{opt}(\mathbf{F})$ rounds.*

Proof: In order to prove the claim, we construct an algorithm SCHEDULEARBITRARY-TREES. In this algorithm, we distinguish between **upward edges** and **downward edges**. An edge $e \in E$ is called upward if $\mathsf{parent}(s(e)) = t(e)$ holds, otherwise it is called downward. Since the underlying network is a tree, every vertex has at most one outgoing edge which is upward. SCHEDULEARBITRARYTREES works as follows:

(1.) In each round, every vertex $v \in V$ sends all available load via its outgoing upward edge until it is saturated. Note that a vertex with an outgoing upward edge receives load via all its incoming edges while sending load only via its sole outgoing upward edge. As a consequence, all upward edges are saturated after at most $r_{opt}(\mathbf{F})$ rounds.

(2.) Simultaneously, as in SCHEDULEDIRECTEDTREES, every vertex $v \in V$ connected with its parent by a downward edge e sends the values $d_j(e)$, $j \in [k_e]$, along e, where k_e is minimum with $d_{k_e} = 0$. Here, $d_0(e), \ldots, d_{k_e}(e)$ are calculated according to the remaining flow with saturated upward edges. Therefore, v has to know the load $\delta'(v)$ it will have when all upward edges are saturated. This load can easily be computed by

$$\delta'(v) = \delta(v) + \sum_{\substack{e \in \text{in}(v) \\ v = \mathsf{parent}(s(e))}} f(e) - \sum_{\substack{e \in \text{out}(v) \\ t(e) = \mathsf{parent}(v)}} f(e) .$$

(3.) After a vertex $v \in V$ has received $d_j(\tilde{e})$, $j \in [k_{\tilde{e}}]$, for all outgoing downward edges $\tilde{e}$, and all incoming upward edges are saturated, which is the case after at most $r_{opt}(\mathbf{F})$ rounds, v sends tokens along its outgoing edges in the same way as in SCHEDULEDIRECTEDTREES.

Similar to the proof of Corollary 3.3 (page 68), it can be proved that the total number of rounds required by SCHEDULEARBITRARYTREES is at most $2r_{opt}(\mathbf{F})$ rounds, as needed. ■

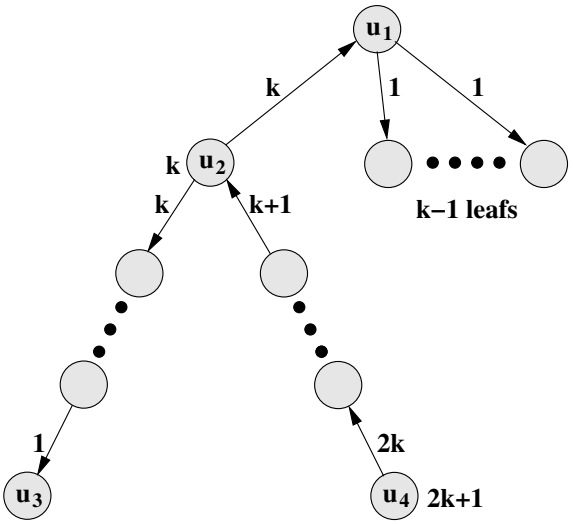


Figure 3.3: Flow graph for which SCHEDULEARBITRARYTREES needs twice the minimum number of rounds.

Note that the proof of Theorem 3.4 (page 69) does not depend on the choice of the root of the tree network. The following example shows that the bound proved in Theorem 3.4 (page 69) is tight.

Example 3.5 *Consider the flow graph $\mathbf{F} = (G, \delta, f)$ with root u_1 (illustrated in Figure 3.3). Applying* SCHEDULEARBITRARYTREES*, vertex u_2 sends its k tokens to its parent u_1. As a result, one of the tokens from u_4 has to be sent along the path from u_4 to u_3 of length $2k$. However, sending the k tokens of u_2 toward u_3 leads to $k+2$ rounds. So, the ratio between the number of rounds used by* SCHEDULEARBITRARYTREES *and the minimum number of rounds is $\frac{2k}{k+2}$, and this ratio converges to 2 as k increases.*

3.5 Conclusion and Directions for Further Research

We considered the problem of scheduling items in synchronous processor networks according to a given flow graph, trying to minimize the required number of rounds. We first analyzed general distributed scheduling strategy. In particular, we showed that for every distributed scheduling strategy there exists a flow on which this strategy requires at least $\frac{3}{2}$ times the minimum number of rounds. Furthermore, we presented a distributed algorithm for flows in tree networks. In contrast to the known local greedy algorithms, this algorithm investigates the structure of the flow graph before sending tokens. We showed that the algorithm requires

at most twice the minimum number of rounds. To the best of our knowledge, this is the first distributed flow scheduling algorithm (even though for a *restricted* class of flow graphs) which is optimum up to a constant factor.

Though we gave a thorough analysis of the considered flow scheduling problem on tree networks, some of the fundamental problems still remain open:

- Close the gap between the lower bound $\frac{3}{2}$ and the upper bound 2 on tree networks.
- Can the proposed approach be adapted to more general networks (like e.g. networks with bounded treewidth)?
- What is the best possible approximation factor for general networks?

— 4 —

Selfish Routing in Non-Cooperative Networks

Aller Eigensinn beruht darauf, daß der Wille
sich an die Stelle der Erkenntnis gedrängt hat.

Arthur Schopenhauer (1788–1860)

4.1 Introduction

4.1.1 Motivation and Framework

Large-scale traffic and communication networks, like e.g. the internet, telephone networks, or road traffic systems often lack a central regulation for several reasons: The size of the network may be too large, the network may be dynamically evolving over time, or the users of the network may be free to act according to their private interest, without regard to the overall performance of the system. Besides the lack of central regulation even cooperation of the users among themselves may be impossible due to the fact that the users may not even know each other. Networks with non-cooperative users have already been studied in the early 1950s in the context of road traffic systems [11, 30, 145]. Recently, motivated by non-cooperative systems like the internet, combining ideas from game theory and theoretical computer science has become increasingly important [45, 75, 119, 120, 124].

An environment, which lacks a central control unit due to its size or operational mode, can be modeled as a non-cooperative game [123]. Users selfishly choose their private strategies, which in our environment correspond to paths (or probability distributions over the paths) from their *sources* to their *destinations*. When routing their *traffics* according to the strategies chosen, the users will experience an *expected latency* caused by the traffics of all users sharing edges. Each user tries to minimize its *private cost*, expressed in terms of its expected latency. This often contradicts the goal of optimizing the *social cost* which measures the global performance of the whole network. Such networks are called *non-cooperative networks* [88] (see

Figure 4.1 for an illustration of such a network). The degradation of the global performance due to the selfish behavior of its users is often termed *price of anarchy* [124, 130] and measured in terms of the *coordination ratio*. The theory of Nash equilibria [95, 117, 118] provides us with an important tool for environments of this kind: A *Nash equilibrium* is a state of the system such that no user can decrease its private cost by unilaterally changing its strategy.

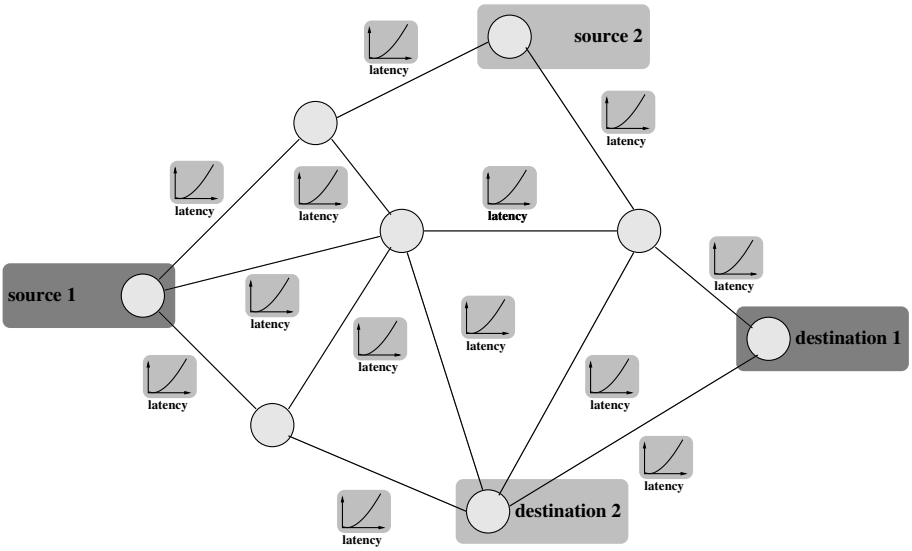


Figure 4.1: A non-cooperative network.

The concept of Nash equilibrium [95, 117, 118] has become an important mathematical tool for analyzing the behavior of selfish users in non-cooperative systems. It has been shown by Nash that a Nash equilibrium exists under fairly broad circumstances. Many algorithms have been developed to compute a Nash equilibrium in a general game (see [108] for an overview). Although the celebrated result of Nash [95, 117, 118] guarantees the existence of a Nash equilibrium for any finite strategic game, the complexity of computing a Nash equilibrium in general games is wide open even if only two users are involved. This problem has been advocated as one of the most important open problems in theoretical computer science today [124].

In this chapter, we consider a routing game introduced by Koutsoupias and Papadimitriou [93], widely known as the *KP-model*. In this model, n non-cooperative users wish to route their unsplittable *traffics* $w_1, \ldots, w_n$ through a very simple network of *parallel links* with *capacities* $c_1, \ldots, c_m$ from source to destination. In the model of *identical users*, all users have equal traffic whereas the traffics may be different in the model of *arbitrary users*. Depending on how the latency of a link is defined we distinguish between three variations of the model. In the model of *identical links*, 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 most general model of *unrelated links*, there exists neither an ordering on the traffics nor on the capacities, that is, user i induces load w_{ij} on link j.

Each user is allowed to route its traffic along links from its *strategy set*. If the strategy sets of all users contain all links, then we have *unrestricted strategy sets*, otherwise *restricted strategy sets*. A *pure strategy* for a user corresponds to some specific link in its strategy set. A

mixed strategy for a user is a probability distribution over pure strategies. Each user employs a (mixed) strategy, trying to minimize its *expected latency*. A *pure assignment* is an n-tuple of pure strategies. A *mixed assignment* is an $n \times m$ probability matrix. A mixed assignment is *fully mixed* if every user chooses each link with non-zero probability.

There is also a global objective function called *social cost*. We investigate the KP-model with respect to two different definitions of social cost. The first one, called *makespan social cost*, is defined as the *maximum expected latency* [93] over all links, whereas the second one, called *polynomial social cost*, is defined as the expectation of the weighted sum of a *polynomial cost function* of degree $d \geq 1$, evaluated at the incurred link loads [57, 102]. However, users do not attend to social cost. The ratio of the maximum social cost of a Nash equilibrium over the minimum social cost of an assignment is called *price of anarchy* or *coordination ratio*.

4.1.2 Contribution

In this chapter, we prove a multitude of results on the KP-model and its variations. In order to simplify the evaluation of these results, we integrate them in a thorough survey. We continue to state our *main* findings here.

4.1.2.1 Makespan Social Cost

Computation of Pure Nash Equilibria. It is easy to see that, for any given pure assignment, the *lexicographical ordering* of the vector of link latencies decreases if allowing exactly one user at a time to decrease its private cost by changing its strategy. Such an improvement step is called *selfish step*. Thus, starting with any pure assignment, every sequence of selfish steps eventually ends in a pure Nash equilibrium, and we can use any such sequence to compute a pure Nash equilibrium. However, a priori it is not clear how many selfish steps are necessary to reach a pure Nash equilibrium. For identical links, we obtain through a thorough analysis the following results:

- The length of a sequence of selfish steps can be $2^{\sqrt{n+7}-3}$, that is, exponential in the number of users, before reaching a pure Nash equilibrium (Theorem 4.13, page 99).
- The length of a sequence of selfish steps is at most $2^n - 1$ if the users always choose their best link (Theorem 4.16, page 101).
- There exists an algorithm, called NASHIFY-IDENTICAL (Algorithm 4, page 102), using selfish steps to compute a Nash equilibrium in $O(n \log n)$ time. NASHIFY-IDENTICAL first orders the users according to their traffics and then one after another reassigns the users to their best link, starting with the user with largest traffic. The length of the used sequence of selfish steps is at most n (Theorem 4.21, page 102).
- Clearly, selfish steps do not increase makespan social cost. Thus, combining the PTAS of Hochbaum and Shmoys [70] for scheduling *n jobs* on m identical *machines* with NASHIFY-IDENTICAL yields a PTAS for computing a pure Nash equilibrium with minimum social cost (Theorem 4.25, page 106).

Price of Anarchy. For pure Nash equilibria, we prove an extensive collection of bounds on the price of anarchy. In particular, we show:

- For the model of arbitrary users and identical links, the price of anarchy is $2 - \frac{2}{m+1}$ (Theorem 4.28, page 106).
- For the model of arbitrary users and related links, the price of anarchy is $\Gamma^{-1}(m)$ up to an additive constant (Corollary 4.63, page 131, and Proposition 4.65, page 131), where Γ^{-1} is the inverse of the well-known *Gamma function* (see Subsection 4.2.2).
- For the model of arbitrary users with restricted strategy sets and identical links, the price of anarchy is $\Gamma^{-1}(m)$ up to an additive constant (Theorem 4.91, page 147, and Theorem 4.92, page 148).
- For the model of identical users with restricted strategy sets and related links, the price of anarchy is bounded from above by $\Gamma^{-1}(n)+1$ (Theorem 4.96, page 152). This bound is tight up to an additive constant if $n = m$ (Theorem 4.91, page 147).
- For the model of arbitrary users with restricted strategy sets and related links, the price of anarchy lies in between $m-1$ and m (Theorem 4.98, page 155).
- For the model of unrelated links, the price of anarchy is $\frac{\max_{i\in[n],j\in[m]} w_{ij}}{\min_{i\in[n],j\in[m]} w_{ij}}$ if $w_{ij} < \infty$ for all $i \in [n], j \in [m]$ (Theorem 4.107, page 162).

Complexity Results. We also give a comprehensive collection of complexity results. In particular, we prove:

- For the model of arbitrary users and identical links, it is $\mathcal{NP}$-complete to decide for a given instance and associated pure Nash equilibrium whether there exists another pure Nash equilibrium with less social cost (Theorem 4.24, page 104).
- For the model of arbitrary users with restricted strategy sets and identical links, a pure Nash equilibrium with minimum social cost is not $(\frac{3}{2}-\varepsilon)$-approximable for any ε with $0 < \varepsilon \leq \frac{1}{2}$ unless $\mathcal{P} = \mathcal{NP}$ (Theorem 4.89, page 145).
- For the model of arbitrary users and identical links, a pure Nash equilibrium with maximum social cost is not $(2 - \frac{2}{m+1} - \varepsilon)$-approximable for any ε with $0 < \varepsilon \leq 1 - \frac{2}{m+1}$ unless $\mathcal{P} = \mathcal{NP}$ (Theorem 4.29, page 108).
- For the model of arbitrary users with restricted strategy sets and related links, a pure Nash equilibrium with maximum social cost is not $(m-2-\varepsilon)$-approximable for any ε with $0 < \varepsilon \leq m-3$ unless $\mathcal{P} = \mathcal{NP}$ (Theorem 4.99, page 157).

4.1.2.2 Polynomial Social Cost

Computation of Pure Nash Equilibria. We consider the computation of pure Nash equilibria under the assumption that the polynomial cost function is the dth power. We prove:

- For the model of arbitrary users and identical links, selfish steps do not increase social cost (Proposition 4.114, page 166). Thus, combining the PTAS of Alon *et al.* [3] for scheduling *jobs* on identical *machines* with respect to the d-norm with NASHIFY-IDENTICAL yields a PTAS for computing a pure Nash equilibrium with minimum social cost (Theorem 4.120, page 168).

- For the model of identical users and related links, a pure Nash equilibrium with minimum/maximum social cost can be computed in $O(m \log n \log m + m)$ time if $d \geq 2$ (Theorem 4.146, page 194, and Theorem 4.149, page 198).

Price of Anarchy. We prove an extensive collection of bounds on the price of anarchy. In particular, we show:

- For the model of arbitrary users and identical links, restricted to pure Nash equilibria, the price of anarchy is $\frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}} \left(\frac{d-1}{d}\right)^d$ if the polynomial cost function is the dth power, $d \geq 2$ (Theorem 4.126, page 171).

- For the model of identical users and two identical links, we prove that the fully mixed Nash equilibrium has maximum social cost (Theorem 4.134, page 178). Equipped with this result, we prove that the price of anarchy in this model is $\frac{1}{2}\left(2^{d-1}+1\right)$ if the polynomial cost function is the dth power (Theorem 4.136, page 182). This implies that the price of anarchy is bounded from above by $\frac{1}{2}\left(2^d + d - 1\right)$ for general polynomial cost functions (Corollary 4.137, page 183).

- For the model of identical users and identical links, we prove that the fully mixed Nash equilibrium has maximum social cost up to factor $\left(1+\frac{1}{n-1}\right)^d$ (Theorem 4.138, page 184). Equipped with this result, we prove that the price of anarchy in this model is bounded from above by $\left(1+\frac{1}{n-1}\right)^d \cdot B_d$ if the polynomial cost function is the dth power, where B_d is the dth *Bell number* (Theorem 4.140, page 189). This implies that the price of anarchy is at most $\sum_{t\in[d]} \left(1+\frac{1}{n-1}\right)^t \cdot B_t$ for general polynomial cost functions (Corollary 4.137, page 183).

- For the model of identical users and related links, the price of anarchy is bounded by $\Omega(m^{d-2})$ if the polynomial cost function is the dth power, $d \geq 2$ (Proposition 4.151, page 198). Thus, the price of anarchy is polynomial in m whereas it is independent of m in the previous cases.

4.1.3 Related Work and Comparison

We now give a brief survey on areas of game theory closely related to the KP-model. For a general introduction to game theory, we recommend [106, 115, 122, 123].

4.1.3.1 Congestion Games

Congestion games were introduced by Rosenthal [127, 128]. In such a game, a finite number of users chooses a non-empty subset of a finite set of resources. The private cost of a player is the sum of the cost of the resources it uses. Here, the cost of a resource only depends on the

number of players sharing it. By constructing an *potential function* for such congestion games, the existence of pure Nash equilibria can be established. Moreover, Monderer and Shapley [110] showed that every *(finite) potential game* is isomorphic to a congestion game. In case of identical users, the KP-model is a special case of congestion games in which each link corresponds to a resource. Otherwise, the KP-model is a special case of *weighted congestion games* [109].

4.1.3.2 Selfish Unsplittable Routing

In the remainder of this chapter, we present our results on the KP-model and its variants, included in a thorough survey. Other surveys are in [27, 47, 91].

Libman and Orda [99, 100], Czumaj *et al.* [28] and Gairing *et al.* [58] studied the network of parallel links with *general latency functions*. Fotakis *et al.* [51] considered selfish unsplittable routing in general networks. They showed that there exist single-commodity (weighted) network congestion games without pure Nash equilibrium. Moreover, for *layered networks*, they showed that for delays equal to the congestions, there always exists a pure Nash equilibrium, and it can be computed in pseudo-polynomial time. The price of anarchy for this type of game is $\Theta(\frac{\log m}{\log\log m})$.

Selfish unsplittable routing is closely related to unsplittable flows. The first constant-factor approximation algorithms for the *single-source* unsplittable flow problem were already obtained by Kleinberg [81]. Further polynomial time approximation algorithms based on rounding techniques were presented by Kolliopoulos and Stein [86]. Dinitz *et al.* [37] showed how to turn a splittable flow into an unsplittable flow in polynomial time. This yielded an approximation factor of 2. For other publications on unsplittable routing, we refer to [8, 18, 82, 83, 84, 85, 87, 139].

4.1.3.3 Selfish Splittable Routing

The earliest model for non-cooperative networks, denoted *Wardrop-model*, was already studied in the 1950's [11, 30, 145] in the context of road traffic systems. Here, traffics are *splittable* into arbitrary pieces. Wardrop [145] introduced the concept of equilibrium to describe user behavior in this kind of traffic networks. For a survey of the early work on this model, see [12]. In this environment, unregulated traffic is modeled as *network flow*. Given an arbitrary network with edge latency functions, equilibrium flows have been classified as flows with all flow paths used between a given source-destination pair having equal latency. Equilibrium flows are optimum solutions to a convex program if the edge latencies are given by convex functions.

A lot of subsequent work (see [135, Section 1.2] for a brief survey) on this model was motivated by Braess's Paradox [15]. An equilibrium in this model can be interpreted as a Nash equilibrium in a game with infinitely many users, each carrying an infinitesimal amount of traffic from a source to a destination. The private cost of a user is defined to be the sum of the edge latencies on a path from the user's source to its destination, the social cost is defined to be the sum over all edge latencies in the network. In contrast to the KP-model, the social costs of all Nash equilibria are equal.

Orda *et al.* [121] investigated equilibria when restricting to a network of parallel links. Inspired by the new interest in the coordination ratio, the Wardrop-model was re-investigated

[20, 101, 129, 130, 131, 132, 134, 135, 136]. Recently, the influence of taxes to the selfish behavior of the users were considered [22, 23, 49]. The practical relevance of the Wardrop-model is underpinned by its use by traffic engineers, who utilized equilibria in route-guidance systems to prescribe user behavior. Recent analysis of this framework have been done by Correa *et al.* [24, 112] for capacitated networks. Algorithms and experimental benchmarking on real-world problems were given by Jahn *et al.* [74].

4.1.3.4 Stackelberg Games

In a *Stackelberg game*, one user acts as a *leader*, and the remaining users as *followers*. The problem is then to compute a strategy for the leader, a so-called *Stackelberg strategy*, that induces the followers to react in a way that they attend to social cost.

Korilis *et al.* [89, 90] started considering Stackelberg strategies for the Wardrop-model restricted to parallel links. Roughgarden [133] showed that computing an optimum Stackelberg strategy is $\mathcal{NP}$-hard, and he introduced an algorithm, computing a Stackelberg strategy that leads to a Wardrop equilibrium with social cost at most a constant factor away from the optimum. Moreover, Kumar and Marathe [96] gave a FPTAS to compute an optimum Stackelberg strategy.

4.1.3.5 Network Design

Pigou's example [125] and the well known Braess's Paradox [15] show that there exist networks such that strict sub-networks perform better when users are selfish. If the goal is to construct a network where the coordination ratio of the network is small, then an interesting network design problem arises: Given a network and the corresponding routing tasks, determine a set of edges which should be removed from the network to obtain a best possible routing at Nash equilibrium. Such network design problems arise e.g. in the routing of road traffic, when traffic engineers want to determine roads that should be closed, or changed to one-way roads, in order to obtain an optimum traffic flow. See [4, 5, 43, 67, 88, 129] for recent publications on this topic.

4.1.4 Organization

The rest of this chapter is organized as follows. After some preliminary remarks in Section 4.2, we formally introduce the KP-model and its variants in Section 4.3, and investigate these variants in Sections 4.4 - 4.9. We conclude, in Section 4.10, with a discussion of our results and some open problems.

4.2 Preliminaries

After the introduction of some basic notations in Subsection 4.2.1, we define the *Gamma function*, a generalization of factorials to non-integer values, and state some of its basic properties in Subsection 4.2.2. In Subsection 4.2.3, we introduce *falling factorials*, *Stirling numbers of the second kind* and *Bell numbers*, and we show how they are related. We then define and investigate a *binomial cost function* in Subsection 4.2.4. We close, in Subsection 4.2.5, by giving a list of decision problems.

4.2.1 Notation

For all integers $k \geq 0$, we denote $[k] = \{1, \ldots, k\}$. For a random variable X with associated probability distribution $\mathbf{P}$, denote $\mathbb{E}_{\mathbf{P}}(X)$ the **expectation** of X.

4.2.2 The Gamma Function

For any integer $k \geq 1$, the **Gamma function** Γ is defined by

$$\Gamma(k+1) \quad = \quad k! \,, \tag{4.1}$$

while for any arbitrary real number $x > 0$,

$$\Gamma(x) \quad = \quad \int_0^\infty t^{x-1} e^{-t} dt \,. \tag{4.2}$$

We notice that

$$\Gamma(x+1) \quad = \quad x \cdot \Gamma(x) \,. \tag{4.3}$$

The Gamma function is invertible, and both Γ and its inverse Γ^{-1} are monotonic increasing. Moreover, it is well known (see e.g. [62]) that for any integer $k \geq 1$, it is

$$\Gamma^{-1}(k) \quad = \quad \frac{\log k}{\log\log k} \cdot (1 + o(1)) \quad = \quad \Theta\left(\frac{\log k}{\log\log k}\right) \,. \tag{4.4}$$

For a textbook introduction to the Gamma function, see [66, 68].

4.2.3 Falling Factorials, Stirling Numbers and Bell Numbers

For any pair of integers $k \geq 1$ and $i \geq 0$, denote $k^{\underline{i}}$ the ith **falling factorial** given by

$$k^{\underline{i}} = k \cdot (k-1) \cdots (k-i+1) \,.$$

For any pair of integers $d \geq 1$ and $i \in [d] \cup \{0\}$, the **Stirling number of the second kind** $S(d,i)$ counts the number of partitions of a set with d elements into exactly i *blocks* (non-empty subsets). In particular, $S(d,0)$ is taken to be 0, while

$$S(d,1) \quad = \quad S(d,d) \quad = \quad 1 \,. \tag{4.5}$$

Also, for all $d \geq 2$,

$$S(d,2) \quad = \quad 2^{d-1} - 1 \,. \tag{4.6}$$

Furthermore, Stirling numbers of the second kind for $d \geq 2$ and $i \in [d-1]$ satisfy the recurrence

$$S(d,i) \quad = \quad S(d-1,i-1) + i \cdot S(d-1,i) \,. \tag{4.7}$$

It is known that for all integers $d \geq 1$, it holds that

$$k^d \quad = \quad \sum_{i \in [d]} S(d,i) \cdot k^{\underline{i}} \,. \tag{4.8}$$

This implies that the Stirling numbers of the second kind are the *connecting coefficients* between the sequence of powers and the sequence of falling factorials. For any integer $d \geq 1$, the dth **Bell number** B_d counts the number of partitions of a set with d elements into blocks. Thus, we have

$$B_d \quad = \quad \sum_{i \in [d]} S(d,i) \; . \tag{4.9}$$

For a textbook introduction to falling factorials, Stirling numbers of the second kind and Bell numbers, see [2, Chapters II & III].

4.2.4 Binomial Cost Functions

We now introduce the *binomial cost function* $H(\mathbf{p},g)$, where $\mathbf{p} = (p_1,\ldots,p_r)$ is a probability vector and $g : \mathbb{R} \to \mathbb{R}$ is a function. Gairing *et al.* [58] proved that in case that g is convex, the binomial cost function increases by replacing all probabilities by the average probability (Lemma 4.2). We prove how binomial cost functions, Stirling numbers of the second kind, and falling factorials are related if g is just the dth power (Proposition 4.3).

Definition 4.1 *For any integer $r \in [n]$, consider a probability vector $\mathbf{p} = (p_1,\ldots,p_r)$, and fix a function $g : \mathbb{R} \to \mathbb{R}$. The* **binomial cost function** *$H(\mathbf{p},g)$ is defined by*

$$H(\mathbf{p},g) \quad = \quad \sum_{A \subseteq [r]} \prod_{k \in A} p_k \prod_{k \notin A} (1-p_k) \cdot g(|A|) \; .$$

If all probabilities have the same value p, then we write

$$H(p,r,g) \quad = \quad \sum_{0 \leq k \leq r} \binom{r}{k} p^k (1-p)^{r-k} \cdot g(k) \; . \tag{4.10}$$

Lemma 4.2 (Gairing et al. [58]) *For a probability vector $\mathbf{p} = (p_1,\ldots,p_r)$, let $\widetilde{p} = \frac{\sum_{i \in [r]} p_i}{r}$. If the function g is convex, then $H(\mathbf{p},g) \leq H(\widetilde{p},r,g)$.*

Proposition 4.3 *For the binomial cost function $H(p,r,x^d)$, $d \geq 1$, we have*

$$H(p,r,x^d) \quad = \quad \sum_{i \in [d]} p^i \cdot S(d,i) \cdot r^{\underline{i}} \; .$$

Proof: Clearly,

$$\begin{aligned}
H(p,r,x^d) &\overset{(4.10),\ page\ 81}{=} \sum_{0\le k\le r}\binom{r}{k}p^k(1-p)^{r-k}k^d \\
&\overset{d\ge 1}{=} \sum_{k\in[r]}\binom{r}{k}p^k(1-p)^{r-k}k^d \\
&= p\cdot r\sum_{k\in[r]}\binom{r-1}{k-1}p^{k-1}(1-p)^{r-k}k^{d-1} \\
&= p\cdot r\sum_{0\le k\le r-1}\binom{r-1}{k}p^k(1-p)^{r-1-k}(k+1)^{d-1} \\
&= p\cdot r\sum_{0\le t\le d-1}\binom{d-1}{t}\sum_{0\le k\le r-1}\binom{r-1}{k}p^k(1-p)^{r-1-k}k^t \\
&= p\cdot r\sum_{0\le t\le d-1}\binom{d-1}{t}H(p,r-1,x^t)\,. \qquad (4.11)
\end{aligned}$$

Resolving this recurrence leads to

$$H(p,r,x^d) = \sum_{i\in[d]} p^i\cdot\alpha(d,i)\cdot r^{\underline{i}} \qquad (4.12)$$

for some $\alpha(d,i)>0$ with

$$\begin{aligned}
\alpha(d,1) &= S(d,1) = 1\,,\\
\alpha(d,d) &= S(d,d) = 1\,.
\end{aligned}$$

We proceed to show that $\alpha(d,i)=S(d,i)$ for all $i\in[d-1]\setminus\{1\}$. We have

$$\begin{aligned}
H(p,r,x^d) &\overset{(4.11)}{=} p\cdot r\sum_{0\le t\le d-1}\binom{d-1}{t}H(p,r-1,x^t) \\
&= p\cdot r\cdot H(p,r-1,1)+p\cdot r\sum_{t\in[d-1]}\binom{d-1}{t}H(p,r-1,x^t) \\
&\overset{H(p,r-1,1)=1}{=} p\cdot r+p\cdot r\sum_{t\in[d-1]}\binom{d-1}{t}H(p,r-1,x^t) \\
&\overset{(4.12)}{=} p\cdot r+p\cdot r\sum_{t\in[d-1]}\binom{d-1}{t}\sum_{i\in[t]}p^i\cdot\alpha(t,i)\cdot(r-1)^{\underline{i}} \\
&= p\cdot r+p\cdot r\sum_{t\in[d-1]}\sum_{i\in[t]}\binom{d-1}{t}p^i\cdot\alpha(t,i)\cdot(r-1)^{\underline{i}} \\
&= p\cdot r+\sum_{t\in[d-1]}\sum_{i\in[t]}\binom{d-1}{t}p^{i+1}\cdot\alpha(t,i)\cdot r^{\underline{i+1}} \\
&= p\cdot r+\sum_{i\in[d-1]}p^{i+1}\cdot r^{\underline{i+1}}\sum_{i\le t\le d-1}\binom{d-1}{t}\alpha(t,i) \\
&= p\cdot r+\sum_{2\le i\le d}p^i\cdot r^{\underline{i}}\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\,.
\end{aligned}$$

Thus, comparison with Equation (4.12) (page 82) implies

$$\alpha(d,i) \;=\; \sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\,.$$

We proceed to show by induction on d, $d\ge 2$, that $\alpha(d,i)=\alpha(d-1,i-1)+i\cdot\alpha(d-1,i)$ for all $i\in[d-1]$. As our basis case, let $d=2$. We only have to consider $i=1$. Since $\alpha(1,1)=\alpha(2,1)=1$ and $\alpha(1,0)=0$, we have $\alpha(2,1)=\alpha(1,0)+1\cdot\alpha(1,1)=1$, proving that the claim holds for the basis case. For the induction step, let $d\ge 2$, and assume that the claim holds for $(d-1)$. On the one hand,

$$\begin{aligned}
&\alpha(d,i-1)+i\cdot\alpha(d,i)\\
&\quad= \sum_{i-1\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-2)+i\cdot\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\\
&\quad= \binom{d-1}{i-2}\alpha(i-2,i-2)+\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-2)\\
&\qquad+i\cdot\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\\
&\quad= \binom{d-1}{i-2}\alpha(i-2,i-2)+\sum_{i\le t\le d}\binom{d-1}{t-1}\left[\alpha(t-1,i-2)+(i-1)\alpha(t-1,i-1)\right]\\
&\qquad+\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\\
&\overset{\textit{Induction}}{=} \binom{d-1}{i-2}+\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t,i-1)+\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t-1,i-1)\\
&\quad= \binom{d-1}{i-2}+\sum_{i\le t\le d}\binom{d-1}{t-1}\alpha(t,i-1)+\sum_{i-1\le t\le d-1}\binom{d-1}{t}\alpha(t,i-1)\\
&\quad= \binom{d-1}{i-2}+\sum_{i\le t\le d-1}\binom{d-1}{t-1}\alpha(t,i-1)+\sum_{i\le t\le d-1}\binom{d-1}{t}\alpha(t,i-1)\\
&\qquad+\binom{d-1}{d-1}\alpha(d,i-1)+\binom{d-1}{i-1}\alpha(i-1,i-1)\\
&\quad= \binom{d-1}{i-2}+\binom{d-1}{i-1}+\alpha(d,i-1)+\sum_{i\le t\le d-1}\left[\binom{d-1}{t-1}+\binom{d-1}{t}\right]\alpha(t,i-1)\\
&\quad= \binom{d-1}{i-2}+\binom{d-1}{i-1}+\alpha(d,i-1)+\sum_{i\le t\le d-1}\binom{d}{t}\alpha(t,i-1)\\
&\quad= \binom{d}{i-1}+\alpha(d,i-1)+\sum_{i\le t\le d-1}\binom{d}{t}\alpha(t,i-1)\,.
\end{aligned}$$

On the other hand,

$$\begin{aligned}\alpha(d+1,i) &= \sum_{i\le t\le d+1}\binom{d}{t-1}\alpha(t-1,i-1)\\ &= \sum_{i-1\le t\le d}\binom{d}{t}\alpha(t,i-1)\\ &= \binom{d}{i-1}+\alpha(d,i-1)+\sum_{i\le t\le d-1}\binom{d}{t}\alpha(t,i-1)\,.\end{aligned}$$

This proves the inductive claim. Thus, $\alpha(d,i)$ is defined in the same way as $S(d,i)$ (see Equation (4.7), page 80), as needed. ■

4.2.5 List of Decision Problems

We conclude this section with a list some decision problems used in the remainder of this chapter. The definitions are given in the style of Garey and Johnson [61].

3-DIMENSIONAL MATCHING

INSTANCE:	A finite set $\mathcal{T} \subseteq X \times Y \times Z$, where X, Y, and Z are disjoint sets with $\lvert X\rvert = \lvert Y\rvert = \lvert Z\rvert = q$ elements.
QUESTION:	Does $\mathcal{T}$ contain a matching, i.e., a subset $\mathcal{T}' \subseteq \mathcal{T}$ such that $\lvert\mathcal{T}'\rvert = q$ and no two elements of $\mathcal{T}'$ agree in any coordinate?

BIN PACKING

INSTANCE:	A finite set $\mathcal{U}$ of items, and a size $s(u_i) \in \mathbb{N}$ for each item $u_i \in \mathcal{U}$, $i \in [\lvert\mathcal{U}\rvert]$, a number K of bins, and a positive integer capacity B.
SOLUTION:	Is there a partition of $\mathcal{U}$ into disjoints sets $\mathcal{U}_1,\ldots,\mathcal{U}_K$ such that the sum of the sizes of the items in each $\mathcal{U}_j$, $j \in [K]$, is B or less?

MULTIPROCESSOR SCHEDULING

INSTANCE:	A finite set $\mathcal{T}$ of tasks, a number K of processors, a length $l(t,j) \in \mathbb{N}$ for each $t \in \mathcal{T}$ on machine $j \in [K]$, and a positive integer deadline D.
QUESTION:	Is there a K-processor schedule for $\mathcal{T}$ that meets the overall deadline?

PARTITION

INSTANCE:	A finite set $\mathcal{U}$ of items, a size $s(u_i) \in \mathbb{N}$ for each item $u_i \in \mathcal{U}$, $i \in [\lvert\mathcal{U}\rvert]$, and a number K.
QUESTION:	Is there a partition of $\mathcal{U}$ into disjoint sets $\mathcal{U}_1,\ldots,\mathcal{U}_K$ such that $\sum_{u\in\mathcal{U}_i} s(u) = \sum_{u\in\mathcal{U}_j} s(u)$ for all $i,j \in [K]$?

If K is not part of the input, then the problem is called K-PARTITION.

4.3 KP-Model

We now formally introduce the **KP-model** and its variants considered in the remainder of this chapter. The definitions are patterned after those in [46, Section 2], [50, Section 2], [57, Section 2], [59, Section 2], [103, Section 2], and [107, Section 2], which, in turn, were based on those in [93, Sections 1 & 2].

4.3.1 Instance

We consider a simple **network** consisting of a set of m parallel **links** $1, 2, \ldots, m$ from a **source** node to a **destination** node. Each of n **users** $1, 2, \ldots, n$ wishes to route a particular amount of traffic along a (non-fixed) link from source to destination. Assume throughout that $m \geq 2$ and $n \geq 2$. Denote $w_i > 0$ the **traffic** of user $i \in [n]$. Define the $n \times 1$ **traffic vector w** in the natural way. Without loss of generality, assume that $w_1 \geq w_2 \geq \ldots \geq w_n$. Let $W = \sum_{i \in [n]} w_i$. In the model of **identical users**, all user traffics are equal to 1, whereas traffics may vary arbitrarily in the model of **arbitrary users**.

Denote $c_j > 0$ the **capacity** of link $j \in [m]$, representing the rate at which the link processes traffic. Define the $m \times 1$ **capacity vector c** in the natural way. Assume throughout, without loss of generality, that $c_1 \geq \ldots \geq c_m$. Let $C = \sum_{j \in [m]} c_j$. In the model of **identical links**, all link capacities are equal to 1, whereas link capacities may vary arbitrarily in the model of **related links**. In the model of **unrelated links**, there exists neither an ordering on the traffics nor on the capacities, and we denote $w_{ij} > 0$ the **traffic** of user $i \in [n]$ on link $j \in [m]$. We set $w_{ij} = \infty$ if user i is not allowed to choose link j. Define the $n \times m$ **traffic matrix w** in the natural way. Clearly, link capacities are not necessary as a problem input when links are unrelated. However, to obtain common expressions in the following definitions, we assume that the link capacities are equal to 1. An **instance** is a pair $(\mathbf{w}, \mathbf{c})$. If the users are identical, then we replace **w** by n. Similarly, we replace **c** by m if the links are identical. An illustration of an instance with arbitrary users and related links is given in Figure 4.2.

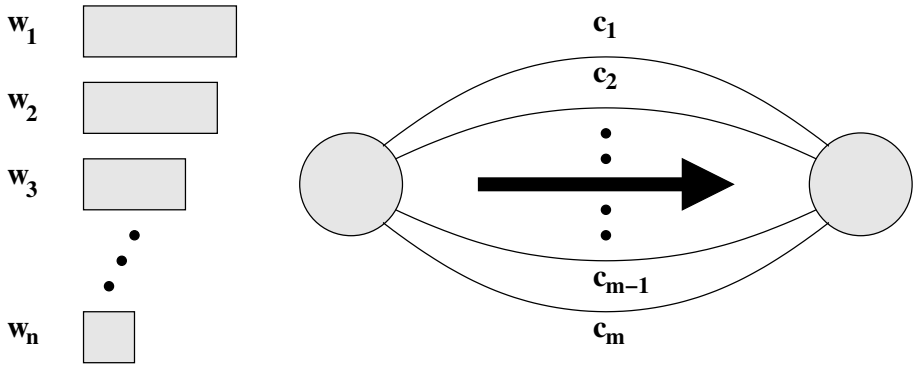


Figure 4.2: The KP-model with arbitrary users and related links.

4.3.2 Strategy and Assignment

Each user $i \in [n]$ is allowed to route its traffic along links from its **strategy set** $R_i \subseteq [m]$. Define the $n \times 1$ **strategy set vector R** in the natural way. Let $R = \sum_{i\in[n]} |R_i|$. If $R_i = [m]$ for all users $i \in [n]$, then we have **unrestricted strategy sets**, otherwise **restricted strategy sets**. Note that restricted strategy sets can be interpreted as a special case of the unrelated links model where, for all users $i \in [n]$, $w_{ij} = w_i$ for all $j \in R_i$ and $w_{ij} = \infty$ otherwise. Thus, it is not necessary to explicitly state **R**, and we will omit it in the sequel.

A **pure strategy** for user $i \in [n]$ corresponds to some specific link. A **mixed strategy** for user $i \in [n]$ is a probability distribution over pure strategies. Thus, it is a probability distribution over the set of links. We define **indicator variables** $I_{ij} \in \{0,1\}$, $i \in [n]$ and $j \in [m]$, such that $I_{ij} = 1$ if and only if $p_{ij} > 0$.

A **pure assignment L** is an n-tuple $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle \in [m]^n$. A user $i \in [n]$ is **solo** in **L** if no other user is assigned to link ℓ_i. A **mixed assignment** is an $n \times m$ **probability matrix** $\mathbf{P} = (p_{ij})$ of nm probabilities p_{ij}, $i \in [n]$ and $j \in [m]$, where p_{ij} is the probability that user i chooses link j. Throughout, we will cast a pure assignment as a special case of a mixed assignment in which all strategies are pure. The **support** of the mixed strategy for user $i \in [n]$, denoted $\mathsf{support}_{\mathbf{P}}(i)$, is the set of those pure strategies (links) to which user i assigns positive probability:

$$\begin{aligned} \mathsf{support}_{\mathbf{P}}(i) &= \{j \in [m] \mid p_{ij} > 0\} \\ &= \{j \in [m] \mid I_{ij} = 1\} . \end{aligned}$$

For each link $j \in [m]$, define the **view** of link j, denoted $\mathsf{view}_{\mathbf{P}}(j)$, as the set of users $i \in [n]$ that potentially assign their traffics to link j:

$$\mathsf{view}_{\mathbf{P}}(j) = \{i \in [n] \mid p_{ij} > 0\} .$$

A mixed assignment $\mathbf{F} = (f_{ij})$ is **fully mixed** [107, Section 2.2] if $f_{ij} > 0$ for all users $i \in [n]$ and links $j \in [m]$. It is **generalized fully mixed** [50, Section 2] if there exists a subset $S \subseteq [m]$ such that $f_{ij} > 0$ for all $i \in [n]$ and $j \in S$, and $f_{ij} = 0$ otherwise. Thus, the fully mixed assignment is a special generalized fully mixed assignment where $S = [m]$.

4.3.3 Load and Latency

Fix now a mixed assignment **P**. The **expected load** $\delta_j(\mathbf{P})$ on a link $j \in [m]$ is defined by

$$\delta_j(\mathbf{P}) = \sum_{i\in[n]} p_{ij} w_{ij} .$$

The **expected latency** $\Lambda_j(\mathbf{P})$ on a link $j \in [m]$ is the ratio between the expected load on link j and the capacity of link j. Thus,

$$\Lambda_j(\mathbf{P}) = \frac{\delta_j(\mathbf{P})}{c_j} = \frac{\sum_{i\in[n]} p_{ij} w_{ij}}{c_j} .$$

The **maximum expected latency** $\Lambda(\mathbf{P})$ is the maximum, over all links, of the expected latency $\Lambda_j(\mathbf{P})$ on a link $j \in [m]$, that is,

$$\Lambda(\mathbf{P}) = \max_{j\in[m]} \Lambda_j(\mathbf{P}) .$$

4.3.4 Individual Cost

The **expected individual cost** for user $i \in [n]$ on link $j \in [m]$, denoted $\lambda_{ij}(\mathbf{P})$, is the expectation of the latency for user i given that its traffic is assigned to link j. Thus,

$$\begin{aligned} \lambda_{ij}(\mathbf{P}) &= \frac{w_{ij} + \sum_{k \in [n], k \neq i} p_{kj} w_{kj}}{c_j} \\ &= \frac{\delta_j(\mathbf{P}) + (1 - p_{ij}) w_{ij}}{c_j} . \end{aligned}$$

For each user $i \in [n]$, the **minimum expected individual cost**, denoted $\lambda_i(\mathbf{P})$, is the minimum, over all links $j \in [m]$, of the expected individual cost for user i on link j. Thus,

$$\lambda_i(\mathbf{P}) = \min_{j \in [m]} \lambda_{ij}(\mathbf{P}) .$$

Denote $\mathsf{IC}(\mathbf{w}, \mathbf{c}, \mathbf{P})$ the **maximum expected individual cost**, defined as the maximum, over all users, of the minimum expected individual cost, that is,

$$\mathsf{IC}(\mathbf{w}, \mathbf{c}, \mathbf{P}) = \max_{i \in [n]} \lambda_i(\mathbf{P}) .$$

4.3.5 Social Cost Measures

4.3.5.1 Makespan Social Cost

In their seminal work, Koutsoupias and Papadimitriou [93] introduced the following measurement for the social welfare. Associated with an instance $(\mathbf{w}, \mathbf{c})$ and a mixed assignment $\mathbf{P}$ is the **makespan social cost**, or **social cost** [93, Section 2] for short, denoted $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{P})$, which is defined as the expected maximum latency on a link, where the expectation is taken over all random choices of the users. Thus,

$$\begin{aligned} \mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{P}) &= \mathbb{E}_{\mathbf{P}} \left(\max_{j \in [m]} \frac{\sum_{i \in [n] : \ell_i = j} w_{ij}}{c_j} \right) \\ &= \sum_{\langle \ell_1, \ell_2, \ldots, \ell_n \rangle \in [m]^n} \left(\prod_{k \in [n]} p_{k\ell_k} \cdot \max_{j \in [m]} \frac{\sum_{i \in [n] : \ell_i = j} w_{ij}}{c_j} \right) . \end{aligned}$$

Note that $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{P})$ reduces to the maximum latency through a link in the case of pure strategies. Moreover, by definition of the social cost, there always exists a pure assignment with minimum social cost. So, the **optimum** [93, Section 2] associated with an instance $(\mathbf{w}, \mathbf{c})$, denoted $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$, is the *least possible* maximum (over all links) latency through a link, that is,

$$\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = \min_{\langle \ell_1, \ell_2, \ldots, \ell_n \rangle \in [m]^n} \max_{j \in [m]} \frac{\sum_{i \in [n] : \ell_i = j} w_{ij}}{c_j} .$$

Note that $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$ refers to an *optimum* pure assignment.

4.3.5.2 Polynomial Social Cost

We also consider a measurement for global welfare introduced by Gairing *et al.* [57]. Let

$$\pi^d(x) \;=\; \sum_{0\le t\le d} a_t x^t$$

be a polynomial of degree $d \ge 1$ with non-negative coefficients, that is, $a_t \ge 0$ for all $t \in [d]\cup\{0\}$. Associated with an instance $(\mathbf{w},\mathbf{c})$, a **polynomial cost function** $\pi^d(x)$, and a mixed assignment $\mathbf{P}$ is the **polynomial social cost** [57, Section 2], or **social cost** for short, denoted $\mathsf{SC}_{\pi^d(x)}(\mathbf{w},\mathbf{c},\mathbf{P})$, which is the expectation of the weighted sum of the polynomial $\pi^d(x)$ evaluated at the incurred link loads. Thus,

$$\begin{aligned}
\mathsf{SC}_{\pi^d(x)}(\mathbf{w},\mathbf{c},\mathbf{P}) &= \mathbb{E}_{\mathbf{P}}\left(\sum_{j\in[m]}\frac{\pi^d\left(\sum_{i\in[n]:\ell_i=j} w_{ij}\right)}{c_j}\right)\\
&= \sum_{j\in[m]}\mathbb{E}_{\mathbf{P}}\left(\frac{\pi^d\left(\sum_{i\in[n]:\ell_i=j} w_{ij}\right)}{c_j}\right)\\
&= \sum_{j\in[m]}\sum_{A\subseteq[n]}\left(\prod_{i\in A}p_{ij}\right)\left(\prod_{i\notin A}(1-p_{ij})\right)\frac{\pi^d\left(\sum_{i\in A} w_{ij}\right)}{c_j}\,.
\end{aligned}$$

If we restrict to the polynomial cost function $\pi^d(x) = x^d$, then we write $\mathsf{SC}_{x^d}(\mathbf{w},\mathbf{P})$. Note that

$$\mathsf{SC}_{\pi^d(x)}(\mathbf{w},\mathbf{c},\mathbf{P}) \;=\; \sum_{0\le t\le d} a_t\cdot\mathsf{SC}_{x^t}(\mathbf{w},\mathbf{P})\,. \tag{4.13}$$

Moreover, if we restrict to identical users, then the formula for social cost reduces to

$$\begin{aligned}
\mathsf{SC}_{\pi^d(x)}(n,\mathbf{c},\mathbf{P}) &= \sum_{j\in[m]}\sum_{A\subseteq[n]}\left(\prod_{i\in A}p_{ij}\right)\left(\prod_{i\notin A}(1-p_{ij})\right)\frac{\pi^d(|A|)}{c_j}\\
&\overset{\text{Definition 4.1 (page 81)}}{=} \sum_{j\in[m]} H((p_{1j},\ldots,p_{nj}),\frac{\pi^d(x)}{c_j})\,.
\end{aligned} \tag{4.14}$$

For the polynomial cost function $\pi^2(x) = x^2$, Rode [126] showed that social cost reduces to

$$\mathsf{SC}_{x^2}(\mathbf{w},\mathbf{c},\mathbf{P}) \;=\; \sum_{i\in[n]} w_i \sum_{j\in[m]} p_{ij}\lambda_{ij}(\mathbf{P})\,. \tag{4.15}$$

The **optimum** [57, Section 2] associated with an instance $(\mathbf{w},\mathbf{c})$ and a polynomial cost function $\pi^d(x)$, denoted $\mathsf{OPT}_{\pi^d(x)}(\mathbf{w},\mathbf{c})$, is the *least possible* weighted sum of the polynomial $\pi^d(x)$ evaluated at the incurred link latencies, that is,

$$\mathsf{OPT}_{\pi^d(x)}(\mathbf{w},\mathbf{c}) \;=\; \min_{\langle\ell_1,\ell_2,\ldots,\ell_n\rangle\in[m]^n}\sum_{j\in[m]}\frac{\pi^d\left(\sum_{i\in[n]:\ell_i=j} w_{ij}\right)}{c_j}\,.$$

Note again that $\mathsf{OPT}_{\pi^d(x)}(\mathbf{w},\mathbf{c})$ refers to an *optimum* pure assignment.

4.3.6 Nash Equilibria

We are interested in a special class of (mixed) assignments called Nash equilibria [117, 118] that we describe here. Given an instance $(\mathbf{w}, \mathbf{c})$ and associated mixed assignment $\mathbf{P}$, a user $i \in [n]$ is **satisfied** if $\lambda_{ij}(\mathbf{P}) = \lambda_i(\mathbf{P})$ for all links $j \in \mathsf{support}_{\mathbf{P}}(i)$, and $\lambda_{ij}(\mathbf{P}) \geq \lambda_i(\mathbf{P})$ for all $j \notin \mathsf{support}_{\mathbf{P}}(i)$. Otherwise, user i is **unsatisfied**. The mixed assignment $\mathbf{P}$ is a **Nash equilibrium** [93, Section 2] if and only if all users $i \in [n]$ are satisfied. Thus, each user assigns its traffic with positive probability only to links (possibly more than one of them) for which its expected individual cost is minimized. This implies that there is no incentive for a user to unilaterally deviate from its mixed strategy in order to avoid links on which its expected individual cost is higher than necessary. Depending on the type of assignment, we differ between **pure**, **mixed** and (generalized) **fully mixed** Nash equilibria.

Note that the definition of Nash equilibria is *independent* of the definition of social cost. Let $\star \in \{\infty, \pi^d(x)\}$. A priori, it is not clear how to efficiently compute the social cost of a given Nash equilibrium:

NASH EQUILIBRIUM SOCIAL COST

INSTANCE:	A problem instance $(\mathbf{w}, \mathbf{c})$ and an associated Nash equilibrium $\mathbf{P}$.
OUTPUT:	The social cost $\mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{P})$.

4.3.7 Price of Anarchy

Fix an instance $(\mathbf{w}, \mathbf{c})$. A **best (worst) Nash equilibrium** is a Nash equilibrium $\mathbf{P}$ that minimizes (maximizes) $\mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{P})$. The **best social cost** is the social cost of a best Nash equilibrium and is denoted by $\mathsf{BC}_\star(\mathbf{w}, \mathbf{c})$. The **worst social cost** is the social cost of a worst Nash equilibrium and is denoted by $\mathsf{WC}_\star(\mathbf{w}, \mathbf{c})$. In the sequel, we (among others) consider the following three decision problems, given in the style of Garey and Johnson [61]:

BEST PURE NE

INSTANCE:	An instance $(\mathbf{w}, \mathbf{c})$, and a positive integer B.
QUESTION:	Is there a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{L}) < B$?

If m is constant, then the problem is called m-BEST PURE NE.

BETTER PURE NE

INSTANCE:	An instance $(\mathbf{w}, \mathbf{c})$, and an associated pure Nash equilibrium $\mathbf{L}$.
QUESTION:	Is there a pure Nash equilibrium $\mathbf{L}'$ with $\mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{L}') < \mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{L})$?

WORST PURE NE

INSTANCE:	An instance $(\mathbf{w}, \mathbf{c})$, and a positive integer B.
QUESTION:	Is there a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\star(\mathbf{w}, \mathbf{c}, \mathbf{L}) > B$?

If m is constant, then the problem is called m-WORST PURE NE.

The **individual price of anarchy**, also called **individual coordination ratio**, is the worst-case ratio

$$\frac{\mathsf{IC}(\mathbf{w},\mathbf{c},\mathbf{P})}{\mathsf{OPT}_{\star}(\mathbf{w},\mathbf{c})} ,$$

over all instances $(\mathbf{w},\mathbf{c})$ and associated Nash equilibria $\mathbf{P}$. The **price of anarchy**, denoted PoA and also called **coordination ratio** [93, Section 2], is the worst-case ratio

$$\frac{\mathsf{WC}_{\star}(\mathbf{w},\mathbf{c})}{\mathsf{OPT}_{\star}(\mathbf{w},\mathbf{c})} ,$$

over all instances $(\mathbf{w},\mathbf{c})$.

4.3.8 Fully Mixed Nash Equilibrium Conjecture

A natural goal is to identify a Nash equilibrium with worst social cost for a given instance. For the model of related links, Gairing *et al.* [59] conjectured that, in case of its existence, the *fully mixed Nash equilibrium*, which is unique (see Theorem 4.67, page 133), is the worst Nash equilibrium with respect to makespan social cost. Recently, this conjecture was also consider with respect to polynomial social cost [57, 102]. We will investigate the conjecture for the most general model of unrelated links and with respect to both definitions of social cost.

Conjecture 4.4 (Fully Mixed Nash Equilibrium Conjecture (Gairing *et al.* [59]))
Consider the model of unrelated links. Then, for any instance $(\mathbf{w},\mathbf{c})$ *such that a fully mixed Nash equilibrium* $\mathbf{F}$ *exists, and for any associated Nash equilibrium* $\mathbf{P}$*,*

$$\mathsf{SC}_{\star}(\mathbf{w},\mathbf{c},\mathbf{P}) \;\leq\; \mathsf{SC}_{\star}(\mathbf{w},\mathbf{c},\mathbf{F}) .$$

In the following, we denote this conjecture as **FMNE Conjecture**. The FMNE Conjecture is a simultaneously intuitive and natural conjecture:

- To support intuition, observe that the fully mixed Nash equilibrium favors collisions between different users (since each user assigns its traffic with positive probability to every link). This increased probability of collisions favors an increase to social cost.
- To support significance, note that the FMNE Conjecture identifies the actual *worst-case* Nash equilibrium for the selfish routing game we consider. We stress that, in sharp contrast, the worst-case measure of price of anarchy only determines how far the worst-case Nash equilibrium is from optimum performance. Since it does not identify the worst-case Nash equilibrium, it fails to provide a basis of comparison between different Nash equilibria on the basis of their social costs.

The ultimate settlement of the FMNE Conjecture may also reveal an interesting complexity-theoretic contrast between the worst-case pure and the worst-case mixed Nash equilibria. On one hand, identifying a worst-case pure Nash equilibrium is $\mathcal{NP}$-complete (see Theorem 4.26, page 106, and Theorem 4.121, page 169). On the other hand, if the FMNE Conjecture is valid, identification of the worst-case mixed Nash equilibrium is immediate in the cases where the fully mixed Nash equilibrium exists. In addition, the characterization of the fully mixed Nash equilibrium shown in Theorem 4.67 (page 133) implies that such existence can be checked in polynomial time.

4.3.9 Sequence of Greedy Selfish Steps

In contrast to general strategic games, there always exists a pure Nash equilibrium in the KP-model. A proof can be given with help of **sequences of greedy selfish steps**.

In a **selfish step**, exactly one unsatisfied user is allowed to improve by changing its pure strategy. A selfish step is a **greedy selfish step** if the user chooses a best pure strategy. Even-Dar *et al.* [42] considered different **rules** for sequences of selfish steps, listed in Table 4.1.

RANDOM	:	Choose each unsatisfied user with the same probability
FIFO	:	Choose user who is unsatisfied for the longest time
MAX WEIGHT JOB	:	Choose unsatisfied user with maximum traffic
MIN WEIGHT JOB	:	Choose unsatisfied user with minimum traffic
MAX LOAD MACHINE	:	Choose unsatisfied user on a link with maximum latency

Table 4.1: Rules for Sequences of Greedy Selfish Steps

It is easy to see that the *lexicographical ordering* of the latency vector decreases in each selfish step. Thus, starting with any pure assignment, every sequence of selfish steps eventually ends in a pure Nash equilibrium.

Theorem 4.5 (Fotakis *et al.* [50]) *Consider the model of unrelated links. Then, for any instance, there exists at least one pure Nash equilibrium.*

Though the existence of a pure Nash equilibrium can be proved with help of selfish steps, it is not clear how many selfish steps are necessary to reach a pure Nash equilibrium. We address this question in the following decision problem:

NASHIFY

INSTANCE:	A problem instance $(\mathbf{w}, \mathbf{c})$, an associated pure assignment $\mathbf{L}$, and a positive integer k.
QUESTION:	Is there a sequence of at most k selfish steps that transforms $\mathbf{L}$ into a pure Nash equilibrium?

If k is not part of the input, then the problem is called k-NASHIFY.

4.3.9.1 Makespan Social Cost

In case of makespan social cost, selfish steps do *not increase* the social cost of the initial pure assignment. Thus, selfish steps can be used for **nashification** [46], that is, to compute a pure Nash equilibrium from any given pure assignment without altering the social cost. Starting with any pure assignment with minimum social cost, this also shows that there always exists a pure Nash equilibrium with optimum social cost.

Theorem 4.6 (Fotakis *et al.* [50]) *Consider the model of unrelated links. Then, for any problem instance $(\mathbf{w}, \mathbf{c})$, there exists a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L}) = \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$.*

4.3.9.2 Polynomial Social Cost

We will see later that, in contrast to makespan social cost, we can not (at least directly) use selfish steps to **nashify** a given pure assignment since a selfish step can *increase* polynomial social cost. Moreover, though there always exists a pure Nash equilibrium with optimum social cost in the model of identical links, this is not always the case in the model of related links.

4.3.10 Relation to Multiprocessor Scheduling

Due to the simple structure of its routing network, the KP-model is closely related to **multiprocessor scheduling**. Here, n **jobs** (users) have to be scheduled on m **machines** (links), and the quality of a **schedule** (assignment) is measured in terms of **makespan** (makespan social cost). Note that the corresponding decision problem MULTIPROCESSOR SCHEDULING is $\mathcal{NP}$-complete in the strong sense [61]. We now give some results on multiprocessor scheduling that will turn out to be useful in the remainder of this chapter.

4.3.10.1 Identical and Related Machines

There exists a large number of algorithms to approximate an optimum schedule on identical and related machines. Some of them are listed in Table 4.2, together with their performance guarantees.

	Identical Machines		**Related Machines**	
Algorithm	Upper Bound	Lower Bound	Upper Bound	Lower Bound
LIST SCHEDULING	$2-\frac{1}{m}$ [64]	$2-\frac{1}{m}$ [64]	**unknown**	
LPT	$\frac{4}{3}-\frac{1}{3m}$ [65]	$\frac{4}{3}-\frac{1}{3m}$ [65]	$\frac{5}{3}$ [54]	$\frac{5}{3}$ [54]
MULTIFIT	1.2 [53]	$\frac{13}{11}$ [53]	1.4 [55]	1.341 [55]
PTAS	$1+\varepsilon$ [70]	—	$1+\varepsilon$ [71]	—

Table 4.2: Performance guarantees of algorithms for multiprocessor scheduling.

For our purposes, the LPT-algorithm, introduced by Graham [65] and further studied in [38, 54, 63, 116], and the PTAS, introduced by Hochbaum and Shmoys [70, 71] are the most important algorithms. The LPT-algorithm is stated as Algorithm 3 (page 93).

Another way to approximate an optimum schedule is through local search heuristics [17, 21, 48, 138]. The simplest form of local search is **iterative improvement**, also called **local improvement** or, in case of minimization problems, **descent algorithms**. This method iteratively chooses a better solution in the **neighborhood** of the current solution. It stops when no better solution is found. We say that the current solution is a **local optimum**.

In the literature, many different definitions of neighborhood can be found. The **jump neighborhood** is closely related to selfish steps. Here, we move a job from a machine with maximum latency to another machine. We say that we are in a **jump optimal solution**, if no jump can decrease the makespan or the number of machines with maximum latency without

Algorithm 3 (LPT)
Input: n jobs with sizes $w_1, \ldots, w_n$, and m related machines with speeds $c_1, \ldots, c_m$
Output: a schedule **L**

(1) **begin**
(2) sort the job sizes in non-increasing order so that $w_1 \geq \ldots \geq w_n$;
(3) **for each** job $i \leftarrow 1$ to n **do**
(4) assign job i to the machine where it causes minimum latency;
(5) **return** the resulting schedule **L**;
(6) **end**

increasing the makespan. A list of performance guarantees of jump optimal solutions is given in Table 4.3. Since each pure Nash equilibrium is also jump optimal, the bounds also hold for pure Nash equilibria. For an overview on local search heuristics, we recommend the paper by Schuurman and Vredefeld [138].

	Identical Machines	**Related Machines**
Upper Bound	$2 - \frac{2}{m+1}$ [48]	$\frac{1+\sqrt{4m-3}}{2}$ [21]

Table 4.3: Performance guarantees for jump optimal solutions.

4.3.10.2 Unrelated Machines

Computing an optimum schedule on unrelated links was first considered by Horowitz and Sahni [73]. They presented an exponential time, dynamic programming algorithm. They also gave an FPTAS to approximate the optimum schedule for the case that m is constant. For general m, Lenstra *et al.* [97] proved that an optimum schedule is not $(\frac{3}{2} - \varepsilon)$-approximable for any ε with $0 < \varepsilon \leq \frac{1}{2}$ unless $\mathcal{P} = \mathcal{NP}$. This holds even if all processing times are taken from $\{1, 2, \infty\}$. In contrast, an optimum assignment can be computed in polynomial time if all processing times are taken from $\{1, 2\}$. Lenstra *et al.* [97] also presented a polynomial time approximation algorithm with approximation factor 2, which is based on linear programming.

4.3.11 Tabular Overview

Before we start to give a thorough survey on the KP-model, we illustrate some of the results on pure Nash equilibria in table form.

		Identical Users		**Arbitrary Users**	
		Upper Bound	Lower Bound	Upper Bound	Lower Bound
Unrestricted	**Identical Links**	1 (Proposition 4.8, page 96)		$2-\frac{2}{m+1}$ (Theorem 4.28, page 106)	
	Related Links	1 (Proposition 4.42, page 113)		$\Gamma^{-1}\left(\frac{1}{\rho}\right)$ (Corollary 4.64, page 131)	$\Gamma^{-1}\left(\frac{1}{\rho}\right)-3$ (Proposition 4.65, page 131)
Restricted	**Identical Links**	$\Gamma^{-1}(m)$ (Theorem 4.92, page 148)	$\Gamma^{-1}(m)-2$ (Theorem 4.91, page 147)	$\Gamma^{-1}(m)$ (Theorem 4.92, page 148)	$\Gamma^{-1}(m)-2$ (Theorem 4.91, page 147)
	Related Links	$\Gamma^{-1}(n)+1$ (Theorem 4.96, page 152)	$\Gamma^{-1}(m)-2$ (Theorem 4.91, page 147)	$m-1$ (Theorem 4.98, page 155)	m (Theorem 4.98, page 155)

	Upper Bound	Lower Bound
Unrelated Links	$\Theta\left(s+\frac{\log m}{\log(1+\frac{\log m}{s})}\right)$ (Theorem 4.106, page 162)	

Table 4.4: Bounds on the price of anarchy for pure Nash equilibria in the KP-model with makespan social cost.

		Identical Users		**Arbitrary Users**	
		Best Pure	Worst Pure	Best Pure	Worst Pure
Unrestricted	**Identical Links**	$O(n)$ (Proposition 4.8, page 96)		PTAS (Theorem 4.25, page 106)	$2-\frac{2}{m+1}-\varepsilon$ (Theorem 4.29, page 108)
	Related Links	$O(m\log n\log m)$ (Proposition 4.42, page 113, & Theorem 4.43, page 114)		PTAS (Theorem 4.52, page 120)	$2-\frac{2}{m+1}-\varepsilon$ (Theorem 4.29, page 108)
Restricted	**Identical Links**	$O(R\sqrt{n})$ (Theorem 4.72, page 134)	**unknown**	$\frac{3}{2}-\varepsilon$ (Theorem 4.89, page 145)	$2-\frac{2}{m+1}-\varepsilon$ (Theorem 4.29, page 108)
	Related Links	$O(nm\log m)$ (Theorem 4.73, page 134)	**unknown**	$\frac{3}{2}-\varepsilon$ (Theorem 4.89, page 145)	$m-2-\varepsilon$ (Theorem 4.99, page 157)

	Best Pure	Worst Pure
Unrelated Links	$\frac{3}{2}-\varepsilon$ (Theorem 4.89, page 145)	**unknown**

Table 4.5: Computational complexity of best and worst pure Nash equilibria in the KP-model with makespan social cost. If there is an entry in O-notation, then this means that a best/worst pure Nash equilibrium can be computed in the given running time. The entry PTAS states that the problem admits a PTAS. Terms in ε indicate that the problem is inapproximable within the given bound.

		Identical Users		Arbitrary Users	
		Upper Bound	Lower Bound	Upper Bound	Lower Bound
Identical Links	$\pi^2(x) = x^2$	1 (Proposition 4.115, page 167)		$\frac{9}{8}$ (Corollary 4.127, page 176)	
	$\pi^d(x) = x^d$ $d \geq 2$	1 (Proposition 4.115, page 167)		$\frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}} \left(\frac{d-1}{d}\right)^d$ (Theorem 4.126, page 171)	
Related Links	$\pi^2(x) = x^2$	$\frac{4}{3}$ (Theorem 4.150, page 198)		**unknown**	$\frac{4}{3}$ (Theorem 4.150, page 198)
	$\pi^d(x) = x^d$ $d \geq 2$	**unknown**	$\Omega\left(m^{d-2}\right)$ (Proposition 4.151, page 198)	**unknown**	$\Omega\left(m^{d-2}\right)$ (Proposition 4.151, page 198)

Table 4.6: Bounds on the price of anarchy for pure Nash equilibria in the KP-model with polynomial social cost.

		Identical Users		Arbitrary Users	
		Best Pure	Worst Pure	Best Pure	Worst Pure
Identical Links	$\pi^2(x) = x^2$	$O(n)$ Proposition 4.115, page 167)		PTAS (Theorem 4.120, page 168)	**unknown**
	$\pi^d(x) = x^d$ $d \geq 2$	$O(n)$ Proposition 4.115, page 167)		PTAS (Theorem 4.120, page 168)	**unknown**
Related Links	$\pi^2(x) = x^2$	$O(m \log n \log m)$ (Theorem 4.146, page 194, & Theorem 4.149, page 198)		**unknown**	
	$\pi^d(x) = x^d$ $d \geq 2$	$O(m \log n \log m)$ (Theorem 4.146, page 194, & Theorem 4.149, page 198)			

Table 4.7: Computational complexity of best and worst pure Nash equilibria in the KP-model with polynomial social cost. If there is an entry in O-notation, then this means that a best/worst pure Nash equilibrium can be computed in the given running time. The entry PTAS states that the problem admits a PTAS.

4.4 Makespan Social Cost and Identical Links

In this section, we consider the KP-model with makespan social cost and identical links. Clearly, for any instance $(\mathbf{w}, m)$ and associated assignment $\mathbf{P}$, the expected latency on a link is equal to its expected load, that is, $\Lambda_j(\mathbf{P}) = \delta_j(\mathbf{P})$ for all $j \in [m]$. Subsection 4.4.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.4.2 hold for general (i.e. mixed) Nash equilibria. Subsection 4.4.3 concentrates on the fully mixed Nash equilibrium. In order to illustrate the results, we use the following instance from [65, 69].

Example 4.7 *Consider the following instance* $(\mathbf{w}, 3)$*: We have* $n = 7$ *arbitrary users and* $m = 3$ *identical links. There are two users with traffics* $w_1 = w_2 = 5$*, two users with traffics* $w_3 = w_4 = 4$*, and three users with traffics* $w_5 = w_6 = w_7 = 3$*.*

4.4.1 Pure Nash Equilibria

4.4.1.1 Computation of Nash Equilibria

We first turn our attention to the problem of computing a pure Nash equilibrium. Basically, two different approaches can be found in the literature. The first approach is to directly compute a pure Nash equilibrium. The second one is to **nashify** a given pure assignment, that is, to convert a given pure assignment into a Nash equilibrium without increasing the social cost. Since selfish steps do not increase the social cost and any sequence of selfish steps eventually reaches a pure Nash equilibrium, selfish steps seem to be suitable for nashification. However, we will see that we have to use them carefully since the length of some sequences of selfish steps is exponential in the number of users before reaching a pure Nash equilibrium.

Identical Users. We first consider the model of identical users. For any instance (n, m) and associated pure assignment $\mathbf{L}$, we can write $\delta_j(\mathbf{L}) = n_j(\mathbf{L})$ for all $j \in [m]$, where $n_j(\mathbf{L})$ is the number of users assigned to link j. This is possible since all users are identical, that is, $w_i = 1$ for all $i \in [n]$. In this model, all pure Nash equilibria have optimum social cost, and such a pure Nash equilibrium can be computed in $O(n)$ time (Proposition 4.8). Moreover, we can use sequences of (not necessarily greedy) selfish steps to convert any given pure assignment into a Nash equilibrium. The length of such a sequence may be $\Omega(\min\{nm, n\log n, \frac{\log m}{\log\log n}\})$ (Theorem 4.9, page 97), but not more than $\frac{n^2}{2}$ (Proposition 4.10, page 97).

Proposition 4.8 *Consider the model of identical users and identical links. Then, for any instance* (n, m) *and associated pure Nash equilibrium* $\mathbf{L}$*, it is* $\mathsf{SC}_\infty(n, m, \mathbf{L}) = \mathsf{OPT}_\infty(n, m)$*, and such a pure Nash equilibrium* $\mathbf{L}$ *can be computed in* $O(n)$ *time.*

Proof: Fix any instance (n, m). Clearly, the users are evenly distributed to the links in an optimum assignment, that is,

$$\mathsf{OPT}_\infty(n, m) \quad = \quad \left\lceil \frac{n}{m} \right\rceil .$$

Assume, by way of contradiction, that there exists a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\infty(n, m, \mathbf{L})$ $> \mathsf{OPT}_\infty(n, m)$. Then, there exist links $j_1, j_2 \in [m]$ with $n_{j_1}(\mathbf{L}) > \lceil \frac{n}{m} \rceil$ and $n_{j_2}(\mathbf{L}) < \lceil \frac{n}{m} \rceil$, respectively. Thus,

$$n_{j_1}(\mathbf{L}) \quad \geq \quad \left\lceil \frac{n}{m} \right\rceil + 1 \quad > \quad n_{j_2}(\mathbf{L}) + 1 \, ,$$

showing that all users on link j_1 are unsatisfied, a contradiction to the assumption that $\mathbf{L}$ is a Nash equilibrium.

In order to compute a pure Nash equilibrium, (1.) assign $\lfloor \frac{n}{m} \rfloor$ users to each of the links, and then (2.) evenly distribute the remaining users to the links. This takes at most $O(n)$ time, as needed. ■

Theorem 4.9 (Even-Dar *et al.* [42]) *Consider the model of identical users and identical links. Then, there exists an instance (n,m) and associated pure assignment for which the maximum length of a sequence of (not necessarily greedy) selfish steps is at least*

$$\Omega\left(\min\left\{nm, n\log n, \frac{\log m}{\log\log n}\right\}\right)$$

before reaching a Nash equilibrium.

Proposition 4.10 *Consider the model of identical users and identical links. Then, for any instance (n,m) and associated pure assignment, the length of a sequence of (not necessarily greedy) selfish steps is at most $\frac{n^2}{2}$ before reaching a Nash equilibrium.*

Proof: Fix any instance (n,m) and associated pure assignment $\mathbf{L}$, and consider the potential function

$$\Pi(\mathbf{L}) \quad = \quad \sum_{j\in[m]} n_j(\mathbf{L})^2 \,.$$

Clearly, n^2 is a trivial upper bound and 0 is a trivial lower bound on $\Pi(\mathbf{L})$. In order to prove the claim, it suffices to show that a selfish step decreases $\Pi(\mathbf{L})$ by at least 2. Consider a selfish step of a user from link $j_1 \in [m]$ to link $j_2 \in [m]$. Clearly, $n_{j_1}(\mathbf{L}) > n_{j_2}(\mathbf{L}) + 1$, and thus

$$n_{j_1}(\mathbf{L}) \quad \geq \quad n_{j_2}(\mathbf{L}) + 2 \tag{4.16}$$

since $n_{j_1}(\mathbf{L})$ and $n_{j_2}(\mathbf{L})$ are integers. We get

$$\begin{aligned}(n_{j_1}(\mathbf{L})-1)^2 + (n_{j_2}(\mathbf{L})+1)^2 \quad &= \quad n_{j_1}(\mathbf{L})^2 + n_{j_2}(\mathbf{L})^2 + 2(n_{j_2}(\mathbf{L}) - n_{j_1}(\mathbf{L})) + 2 \\ &\overset{(4.16)}{\leq} \quad n_{j_1}(\mathbf{L})^2 + n_{j_2}(\mathbf{L})^2 - 2 \,.\end{aligned}$$

Thus, $\Pi(\mathbf{L})$ decreases by at least 2 in every selfish step, as needed. ■

Arbitrary Users. For the model of arbitrary users, things are more complicated. The first approach to directly compute a pure Nash equilibrium was given by Fotakis *et al.* [50]. They showed that the LPT-algorithm (see Algorithm 3, page 93) yields a pure Nash equilibrium. Graham [65] proved that the computed assignment approximates an optimum assignment within factor $\frac{4}{3} - \frac{1}{3m}$ (see Table 4.2, page 92). We use the instance in Example 4.7 (page 96) to show tightness of this upper bound for $m = 3$. Note that this instance can be generalized to show tightness for all $m \geq 3$ (see e.g. Hochbaum [69]).

Theorem 4.11 (Fotakis *et al.* [50], Graham [65]) *Consider the model of arbitrary users and identical links. Then, for any instance* $(\mathbf{w}, m)$*,* LPT *computes a pure Nash equilibrium* $\mathbf{L}$ *with*

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}) \leq \left(\frac{4}{3} - \frac{1}{3m}\right) \mathsf{OPT}_\infty(\mathbf{w}, m) ,$$

using $O((n+m)\log m)$ *time.*

Example 4.7 (continued) *For the given instance,* LPT *returns a pure Nash equilibrium* $\mathbf{L} = \langle 1,2,3,3,1,2,1\rangle$ *with social cost* 11 *whereas the optimum assignment* $\langle 1,2,1,2,3,3,3\rangle$ *has social cost* 9 *(see Figure 4.3). Thus,*

$$\frac{\mathsf{SC}_\infty(\mathbf{w}, 3, \mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w}, 3)} = \frac{11}{9} \overset{m=3}{=} \frac{4}{3} - \frac{1}{3m} .$$

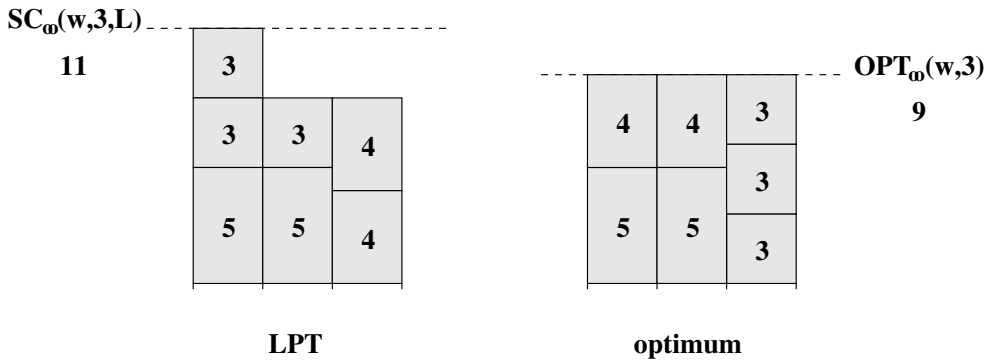


Figure 4.3: Pure Nash equilibrium $\mathbf{L} = \langle 1,2,3,3,1,2,1\rangle$ returned by LPT applied to the instance in Example 4.7 (page 96) (left hand side), and an optimum assignment $\langle 1,2,1,2,3,3,3\rangle$ of this instance (right hand side).

Since every sequence of (not necessarily greedy) selfish steps eventually ends in a pure Nash equilibrium, we can use any such sequence to compute a pure Nash equilibrium. Unfortunately, the length of such a sequence can be exponential in the number of users before reaching a pure Nash equilibrium (Theorem 4.12, page 99, and Theorem 4.13, page 99). Note that these results were found independently. The number of links of the instance used to prove Theorem 4.13 is $m = \sqrt{n+7} - 1$. For this case, the lower bound in Theorem 4.12 is strictly larger than the lower bound in Theorem 4.13.

On the other hand, if all traffics are integers, then the length of a sequence of *greedy* selfish steps is at most $W + n$ (Theorem 4.14, page 101), and in general at most $2^n - 1$ (Theorem 4.16, page 101). The proof of the latter result is based on the observation that a greedy selfish step of a user with traffic w makes no users with traffic at most w unsatisfied (Lemma 4.15, page 101). This observation was independently made by Even-Dar *et al.*[42].

Instead of the maximum length we may ask about the minimum length of a sequence of greedy selfish steps. In particular, we may ask whether a given pure assignment can be transformed into a pure Nash equilibrium using at most k selfish steps. This problem, called NASHIFY (see Subsection 4.3.9, page 91), is $\mathcal{NP}$-complete (Theorem 4.17, page 102).

The proof relies on a reduction from 2-PARTITION. This reduction implies that NASHIFY is $\mathcal{NP}$-complete in the strong sense if m is part of the input [61]. Thus, there is no pseudo-polynomial-time algorithm for NASHIFY (unless $\mathcal{P} = \mathcal{NP}$). In contrast, there is a natural pseudo-polynomial-time algorithm for k-NASHIFY which exhaustively searches all sequences of k selfish steps. Since a selfish step involves an (unsatisfied) user and a link for a total of mn choices, the running time of such an algorithm is $O((mn)^k)$ (Proposition 4.18, page 102).

Though there exist sequences of greedy selfish steps of exponential length, and though NASHIFY is $\mathcal{NP}$-complete, it is possible to use greedy selfish steps to compute a Nash equilibrium in polynomial time (Theorem 4.19, page 102, and Theorem 4.20, page 102). In particular, NASHIFY-IDENTICAL, stated as Algorithm 4 (page 102) and independently found by Even-Dar *et al.* [42], solves NASHIFY when n selfish steps are allowed. NASHIFY-IDENTICAL is based on the rule MAX WEIGHT JOB. First, it sorts the user traffics in non-increasing order so that $w_1 \geq \ldots \geq w_n$. Then, for each user $i \leftarrow 1$ to n, it moves user i to its best link if user i is unsatisfied. With help of Lemma 4.15 (page 101), it is possible to show that this approach yields a pure Nash equilibrium, using at most n greedy selfish steps and $O(n \log n)$ time (Theorem 4.21, page 102).

Theorem 4.12 (Even-Dar *et al.* [42]) *Consider the model of arbitrary users and identical links. Then, there exists an instance* $(\mathbf{w}, m)$ *and associated pure assignment for which the maximum length of a sequence of greedy selfish steps is at least*

$$\frac{\left(\frac{n}{m-1}\right)^{m-1}}{2(m-1)!} .$$

Theorem 4.13 *Consider the model of arbitrary users and identical links. Then, there exists an instance* $(\mathbf{w}, m)$ *and associated pure assignment for which the maximum length of a sequence of greedy selfish steps is at least* $2^{\sqrt{n+7}-3}$.

Proof: We construct an instance with r different classes of traffics.

- Class $\mathcal{U}_1$: $|\mathcal{U}_1| = 2$ users with traffic $x_1 = 1$
- Class $\mathcal{U}_i$: $|\mathcal{U}_i| = 2i+3$ users with traffic

  $$x_i \;=\; (|\mathcal{U}_{i-1}|+1) \cdot x_{i-1} \;=\; 2(i+1)x_{i-1}$$

  for all $i \in [r] \setminus \{1\}$.

The number of users is

$$\begin{aligned} n &= |\mathcal{U}_1| + \cdots + |\mathcal{U}_r| \\ &= 2 + \sum_{2 \leq i \leq r} (2i+3) \\ &= r^2 + 4r - 3 . \end{aligned}$$

Furthermore, we consider $m = r+1$ links. In the following, we denote $t(i)$ the maximum length of a sequence of greedy selfish steps on $i+1$ links, starting with all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_i$ on the same link and the same **load offset**, that is, the same additional load on all $i+1$ links. We proceed as follows: In part (1.), we show a lower bound on the traffic size x_i of a user in $\mathcal{U}_i$ for all $i \in [r] \setminus \{1\}$. In part (2.), we then use this result to prove a lower bound on $t(i)$.

(1.) We first show by induction on $i \in [r] \setminus \{1\}$ that

$$x_i \;>\; \sum_{j\in[i-1]} |\mathcal{U}_j| \cdot x_j \,.$$

As our basis case, let $i = 2$. Since

$$x_2 \;=\; (|\mathcal{U}_1|+1)\cdot x_1 \;>\; |\mathcal{U}_1|\cdot x_1 \,,$$

this proves that the claim holds for the basis case. For the induction step, let $i \geq 3$, and assume that the claim holds for $(i-1)$. We get

$$\begin{aligned} x_i &= (|\mathcal{U}_{i-1}|+1)\cdot x_{i-1} \\ &\overset{Induction}{>} |\mathcal{U}_{i-1}|\cdot x_{i-1} + \sum_{j\in[i-2]} |\mathcal{U}_j|\cdot x_j \\ &= \sum_{j\in[i-1]} |\mathcal{U}_j|\cdot x_j \,, \end{aligned}$$

proving the inductive claim.

(2.) We now use this property to prove $t(i) \geq 2^{i-1}$ by induction on $i \in [r]$. As our basis case, let $i = 1$. Starting with both users in $\mathcal{U}_1$ on the same link and the same load offset on both links, we can do one greedy selfish step. Therefore, $t(1) = 1$, proving that the claim holds for the basis case. For the induction step, let $i \geq 2$, and assume that the claim holds for $(i-1)$. Consider the assignment in which all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_{i-1}$ are on the same link, and all $i+1$ links have the same load offset. We construct a sequence of at least 2^{i-1} greedy selfish steps in the following way:

- Move all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_{i-1}$ to the other i links using greedy selfish steps. This is possible since $x_i > \sum_{j\in[i-1]} |\mathcal{U}_j| \cdot x_j$. Every user with traffic less than x_i who shares the link with a user in $\mathcal{U}_i$ can improve by moving to a link to which no user in $\mathcal{U}_i$ is assigned.
- For each of $i-1$ users in $\mathcal{U}_i$, do one greedy selfish step. Since $x_i > \sum_{j\in[i-1]} |\mathcal{U}_j| \cdot x_j$, all $i-1$ users with traffic x_i choose different links. Then, we move all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_{i-1}$ to the remaining link to which no user in $\mathcal{U}_i$ is assigned. Finally, moving a user in $\mathcal{U}_i$ to this link, we get the following assignment: On each of i links there is exactly one user with traffic x_i. Additionally, all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_{i-1}$ are on one of these i links. On the $(i+1)$th link there are $i+3$ users with traffic $x_i > \sum_{j\in[i-1]} |\mathcal{U}_j| \cdot x_j$. This shows that, by only moving users with traffic less than x_i, no user with traffic less than x_i wants to move to the $(i+1)$th link. So, we exactly have the starting assignment for doing $t(i-1)$ greedy selfish steps.
- After $t(i-1)$ greedy selfish steps have been carried out, we collect all users in $\mathcal{U}_1 \cup \cdots \cup \mathcal{U}_{i-1}$, again, using i users in $\mathcal{U}_i$. This can be done for the following reason: A user with traffic x_i is unsatisfied as long as at least 3 more users with traffic x_i are on the same link. Since initially $2i+3$ users in $\mathcal{U}_i$ are on the same link, this is the case for $2i$ users. As in the previous step, by moving only users with traffic less than x_i, no user with traffic less than x_i wants to move to the $(i+1)$th link. Then, we restart the sequence of $t(i-1)$ greedy selfish steps.

All in all we get $t(i) \geq 2t(i-1)$. By induction hypothesis, this shows that $t(i) \geq 2^{i-1}$, proving the inductive claim.

Since $n = r^2 + 4r - 3$, we have $r = \sqrt{n+7} - 2$. Thus, starting with the pure assignment where all users are assigned to the same link, we get

$$t(r) \;\geq\; 2^{r-1} \;=\; 2^{\sqrt{n+7}-3}\,.$$

It remains to show that the traffics are polynomial in n. The bitlength of the largest traffic x_r is

$$\begin{aligned}\log x_r &\leq \log\left(2(r+1)^r\right)\\ &\leq r\cdot\log\left(2(r+1)\right)\\ &= \left(\sqrt{n+7}-2\right)\cdot\log\left(2\sqrt{n+7}-2\right),\end{aligned}$$

as needed. ■

Theorem 4.14 (Even-Dar *et al.* [42]) *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$ with $w_i \in \mathbb{N}$ for all $i \in [n]$ and associated pure assignment, the length of a sequence of greedy selfish steps is at most $W + n$ before reaching a Nash equilibrium.*

Lemma 4.15 *Consider the model of arbitrary users and identical links. Then, a greedy selfish step of an unsatisfied user $i_1 \in [n]$ with traffic w_{i_1} makes no satisfied user $i_2 \in [n]$ with traffic $w_{i_2} \geq w_{i_1}$ unsatisfied.*

Proof: See Lemma 4.48 (page 117) for a generalization of this result. ■

Theorem 4.16 *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$ and associated pure assignment, the length of a sequence of greedy selfish steps is at most $2^n - 1$ before reaching a Nash equilibrium.*

Proof: Fix any instance $(\mathbf{w}, m)$ and associated pure assignment. We prove by induction on $i \in [n]$ that user i can make at most 2^{i-1} greedy selfish steps. As our basis case, let $i = 1$. Since w_1 is the largest of all traffics, Lemma 4.15 implies that user 1 can make at most one greedy selfish step. This proves that the claim holds for the basis case. For the induction step, let $i \geq 2$, and assume that the claim holds for $(i-1)$. By Lemma 4.15, user i can only become unsatisfied by a move of a user with larger traffic. By induction hypothesis, the number of greedy selfish steps made by users in $[i-1]$ is at most

$$\sum_{k\in[i-1]} 2^{k-1} \;=\; 2^{i-1} - 1\,.$$

This shows that user i can become unsatisfied at most $2^{i-1} - 1$ times. Since user i can be unsatisfied in the initial pure assignment, user i can make at most 2^{i-1} greedy selfish steps, proving the inductive claim. Summing up over all users, the total number of greedy selfish steps is at most

$$\sum_{i\in[n]} 2^{i-1} \;=\; 2^n - 1\,,$$

as needed. ■

Theorem 4.17 (Gairing *et al.* [59]) *Consider the model of arbitrary users and identical links. Then,* NASHIFY *is $\mathcal{NP}$-complete even if $m = 2$.*

Proposition 4.18 (Gairing *et al.* [59]) *Consider the model of arbitrary users and identical links. Then, there exists a pseudo-polynomial algorithm for k-*NASHIFY.

Theorem 4.19 (Even-Dar *et al.* [42]) *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$ and associated pure assignment, the length of a sequence of greedy selfish steps using the rule* FIFO *is at most $\frac{n(n+1)}{2}$ before reaching a Nash equilibrium.*

Theorem 4.20 (Even-Dar *et al.* [42]) *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$ and associated pure assignment, the expected length of a sequence of greedy selfish steps using the rule* RANDOM *is at most $\frac{n(n+1)}{2}$ before reaching a Nash equilibrium.*

Algorithm 4 (NASHIFY-IDENTICAL)

Input: an instance $(\mathbf{w}, m)$ and associated pure assignment $\mathbf{L}$
Output: a pure assignment $\mathbf{L}'$

```
(1)  begin
(2)  sort the user traffics in non-increasing order so that w_1 ≥ ... ≥ w_n;
(3)  for each user i ← 1 to n do
(4)      if user i is unsatisfied then
(5)          let user i carry out a greedy selfish step;
(6)  return the resulting pure assignment L';
(7)  end
```

Theorem 4.21 *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$ and associated pure assignment $\mathbf{L}$, algorithm* NASHIFY-IDENTICAL *computes a Nash equilibrium $\mathbf{L}'$ from $\mathbf{L}$ with $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \leq \mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})$ using at most n greedy selfish steps and $O(n \log n)$ time.*

Proof: Clearly, $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \leq \mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})$ since greedy selfish steps do not increase social cost. Moreover, in every iteration $i \in [n]$, user i becomes satisfied and stays satisfied in the subsequent iterations by Lemma 4.15 (page 101). Thus, $\mathbf{L}'$ is a Nash equilibrium.

NASHIFY-IDENTICAL needs $O(n \log n)$ time for sorting the n user traffics and $O(m \log m)$ time for the construction of a heap containing all loads $\delta_j(\mathbf{L})$, $j \in [m]$. Moreover, in each iteration NASHIFY-IDENTICAL needs $O(\log m)$ time find the minimum element in the heap, and to update the heap after the greedy selfish step. Thus, the total running time is $O(n \log n + m \log m + n \log m)$. The interesting case is when $m \leq n$ (since otherwise, a single user can be assigned to each link, achieving an optimum Nash equilibrium). Thus, in the interesting case, the total running time of NASHIFY-IDENTICAL is $O(n \log n)$. ■

Example 4.7 (continued) *For the given instance and associated pure assignment $\mathbf{L} = \langle 1, 1, 1, 1, 1, 1, 1 \rangle$ where all users are assigned to the first link,* NASHIFY-IDENTICAL *needs* 4 *greedy selfish steps before reaching a pure Nash equilibrium (see Figure 4.4).*

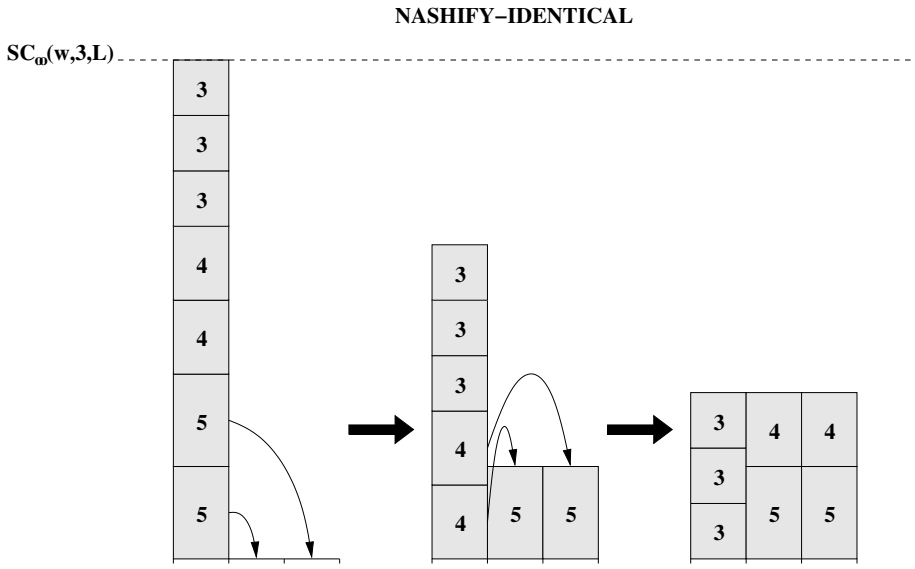


Figure 4.4: NASHIFY-IDENTICAL, applied to the pure assignment $\mathbf{L} = \langle 1,1,1,1,1,1,1 \rangle$ of the instance in Example 4.7 (page 96). Each small arrow corresponds to a greedy selfish step.

4.4.1.2 Computation of Best Nash Equilibria

Now that we have seen that a pure Nash equilibrium can be computed in polynomial time, we might ask for a polynomial time algorithm to compute a pure Nash equilibrium with *minimum* social cost. For the model of identical users, this problem is solvable in polynomial time (Proposition 4.8, page 96). For arbitrary users, however, Fotakis *et al.* [50] showed by reduction from BIN PACKING that BEST PURE NE is $\mathcal{NP}$-hard. Since this problem can be formulated as an integer program, it follows that it is $\mathcal{NP}$-complete (Theorem 4.22, page 104). Since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense [50], there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.23, page 104). Moreover, it is even $\mathcal{NP}$-complete to decide for a given instance and associated pure Nash equilibrium whether there exists another pure Nash equilibrium with less social cost even on two identical links (Theorem 4.24, page 104). However, the algorithm NASHIFY-IDENTICAL enables us to use any approximation algorithm for scheduling *n jobs* on *m* identical *machines* (see Table 4.2, page 92, for a list of such approximation algorithms) to get an approximation algorithm for BEST PURE NE. For example, MULTIFIT combined with NASHIFY-IDENTICAL yields an approximation factor of $\frac{6}{5}$. Moreover, we can give a PTAS for BEST PURE NE (Theorem 4.25, page 106) by proceeding as follows:

(1.) First run the PTAS of Hochbaum and Shmoys [70] for scheduling *n jobs* on *m* identical *machines*. This yields a pure assignment $\mathbf{L}$ such that

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}) \;\leq\; (1+\varepsilon) \cdot \mathsf{OPT}_\infty(\mathbf{w}, m) \; .$$

(2.) Then, apply the algorithm NASHIFY-IDENTICAL on $\mathbf{L}$. This yields a Nash equilibrium $\mathbf{L}'$ such that $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \leq \mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})$. Thus,

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \;\leq\; (1+\varepsilon) \cdot \mathsf{OPT}_\infty(\mathbf{w}, m) \; .$$

We cannot expect to find an FPTAS since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense [50].

Theorem 4.22 (Fotakis *et al.* [50]) *Consider the model of arbitrary users and identical links. Then,* BEST PURE NE *is $\mathcal{NP}$-complete even if $m = 2$.*

Theorem 4.23 *Consider the model of arbitrary users and identical links. Then, there exists a pseudo-polynomial-time algorithm for m-*BEST PURE NE.

Proof: See Theorem 4.104 (page 161) for a generalization of this result. ■

Theorem 4.24 *Consider the model of arbitrary users and identical links. Then,* BETTER PURE NE *is $\mathcal{NP}$-complete even for $m = 2$.*

Proof: Clearly, BETTER PURE NE $\in \mathcal{NP}$. We now prove $\mathcal{NP}$-hardness by reduction from 2-PARTITION, that is, we employ a polynomial time transformation from 2-PARTITION to BETTER PURE NE. Consider an arbitrary instance of 2-PARTITION with at least 2 items (otherwise, the instance of 2-PARTITION is a trivial *no* instance), and let $S = \sum_{u_i \in \mathcal{U}} s(u_i)$. Clearly, $S > 1$ since $s(u_i) \in \mathbb{N}$ for all $u_i \in \mathcal{U}$. From this instance, we construct an instance of BETTER PURE NE as follows:

- There are $n = |\mathcal{U}| + 2$ users with

$$w_i \;=\; \begin{cases} s(u_i) & \text{if } i \in [|\mathcal{U}|], \\ \frac{S+1}{2} & \text{if } i \in \{n-1, n\}. \end{cases}$$

- There are $m = 2$ identical links.

- The pure assignment $\mathbf{L}$ is defined as follows: All users $i \in [|\mathcal{U}|]$ are assigned to link 1, and users $n-1$ and n are assigned to link 2. Clearly, $\delta_1(\mathbf{L}) = \sum_{u_i \in \mathcal{U}} s(u_i) = S$ and $\delta_2(\mathbf{L}) = w_{n-1} + w_n = S + 1$. Since $\delta_1(\mathbf{L}) < \delta_2(\mathbf{L})$, all users $i \in [|\mathcal{U}|]$ are satisfied. Moreover, since

$$\begin{aligned} \delta_1(\mathbf{L}) + w_{n-1} &= \delta_1(\mathbf{L}) + w_n \\ &= S + \frac{S+1}{2} \\ &\overset{S>1}{>} S + 1 \\ &= \delta_2(\mathbf{L}) \,, \end{aligned}$$

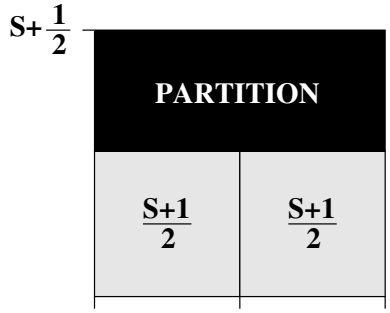
  users $n-1$ and n are satisfied. So, $\mathbf{L}$ is a pure Nash equilibrium, and

$$\mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L}) \;=\; S + 1 \,.$$

Clearly, this is a polynomial time transformation. We prove that this is a transformation from 2-PARTITION to BETTER PURE NE.

(1.) The instance of 2-PARTITION is positive:
Consider a partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1, \mathcal{U}_2$ such that $\sum_{u_i \in \mathcal{U}_1} s(u_i) = \sum_{u_i \in \mathcal{U}_2} s(u_i)$. Use this partition to define a pure assignment $\mathbf{L}'$ for the instance of BETTER PURE NE as follows:

- For each item u_i in $\mathcal{U}_1$, user i is assigned to link 1. For each item $u_i \in \mathcal{U}_2$, user i is assigned to link 2.
- Users $n-1$ and n are assigned to link 1 and 2, respectively.



Clearly,

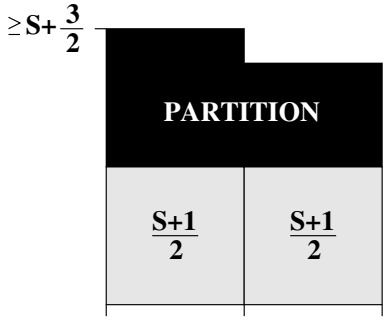
$$\begin{aligned}\delta_1(\mathbf{L}') &= \sum_{u_i \in \mathcal{U}_1} s(u_i) + w_{n-1} \\ &= \frac{S}{2} + \frac{S+1}{2} \\ &= S + \frac{1}{2}\,,\end{aligned}$$

and similarly $\delta_2(\mathbf{L}') = S + \frac{1}{2}$. So, $\mathbf{L}'$ is a pure Nash equilibrium. Moreover,

$$\begin{aligned}\mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L}') &= S + \frac{1}{2} \\ &< S + 1 \\ &= \mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L})\,,\end{aligned}$$

as needed.

(2.) The instance of 2-PARTITION is negative:
For every partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1, \mathcal{U}_2$, it is $\sum_{u_i \in \mathcal{U}_1} s(u_i) \neq \sum_{u_i \in \mathcal{U}_2} s(u_i)$. It follows that either $\sum_{u_i \in \mathcal{U}_1} s(u_i) < \frac{S}{2}$ or $\sum_{u_i \in \mathcal{U}_2} s(u_i) < \frac{S}{2}$.
Consider any pure Nash equilibrium $\mathbf{L}'$ for the instance of BETTER PURE NE. If users $n-1$ and n are assigned to the same link, then the social cost of $\mathbf{L}'$ is at least $w_{n-1} + w_n = S + 1 = \mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L})$, as needed. So, assume that users $n-1$ and n are assigned to different links, say link 1 and 2, respectively.



Assume, without loss of generality, that $\sum_{u_i \in \mathcal{U}_1} s(u_i) < \frac{S}{2}$. Since $s(u_i) \in \mathbb{N}$ for all $u_i \in \mathcal{U}$,

it follows that $\sum_{u_i \in \mathcal{U}_2} s(u_i) \geq \frac{S}{2} + \frac{1}{2}$. So,

$$\begin{aligned} \mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L}') &= \delta_2(\mathbf{L}') \\ &= \sum_{u_i \in \mathcal{U}_2} s(u_i) + w_n \\ &\geq \frac{S}{2} + \frac{1}{2} + \frac{S+1}{2} \\ &= S+1 \\ &= \mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L}) \,, \end{aligned}$$

as needed. ■

Theorem 4.25 *Consider the model of arbitrary users and identical links. Then, there exists a* PTAS *for* BEST PURE NE.

4.4.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

We now turn our attention to worst pure Nash equilibria. For identical users, such a Nash equilibrium can be computed in polynomial time (see Proposition 4.8, page 96). For arbitrary users, Fotakis *et al.* [50] proved by reduction from BIN PACKING that WORST PURE NE is $\mathcal{NP}$-hard. Since this problem can be formulated as an integer program, it follows that it is $\mathcal{NP}$-complete (Theorem 4.26). Since WORST PURE NE is $\mathcal{NP}$-complete in the strong sense [50], there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.27).

For identical users, the price of anarchy is 1 (see Proposition 4.8, page 96). For arbitrary users, the price of anarchy is bounded from above by $2 - \frac{2}{m+1}$, and this bound is *tight* (Theorem 4.28). It is worth noticing that Finn and Horowitz [48] proved the same upper bound for *jump optimal schedules*. Schuurman and Vredeveld [138] showed that this bound is tight (see Table 4.3, page 93). Since every pure Nash equilibrium is also jump optimal, the upper bound follows directly. Greedy selfish steps on identical links can only increase the minimum load over all links. Thus, we can transform every jump optimal schedule into a Nash equilibrium without altering the makespan, proving tightness. Since the social cost of any Nash equilibrium is at most the factor $2 - \frac{2}{m+1}$ away from the social cost of an optimum Nash equilibrium, this also implies that every Nash equilibrium approximates the social cost of the worst Nash equilibrium within this factor. We establish that we can not do better unless $\mathcal{P} = \mathcal{NP}$ (Theorem 4.29, page 108).

Theorem 4.26 (Fotakis *et al.* [50]) *Consider the model of arbitrary users and identical links. Then,* WORST PURE NE *is $\mathcal{NP}$-complete even for $m = 3$.*

Theorem 4.27 (Gairing *et al.* [59]) *Consider the model of arbitrary users and identical links. Then, there exists a pseudo-polynomial-time algorithm for m-*WORST PURE NE.

Proof: See Theorem 4.105 (page 161) for a generalization of this result. ■

Theorem 4.28 *Consider the model of arbitrary users and identical links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} = 2 - \frac{2}{m+1} \,.$$

Proof:
Upper bound: Fix any instance $(\mathbf{w},m)$ and associated pure Nash equilibrium $\mathbf{L}$, and let $j \in [m]$ be a link with $\delta_j(\mathbf{L}) = \mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})$. Clearly, if there is only one user on link j, then $\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) = \mathsf{OPT}_\infty(\mathbf{w},m)$. So, assume that there are at least two users on link j, and denote w_{min} the minimum traffic of these users. By definition of Nash equilibrium,

$$\delta_j(\mathbf{L}) \;\leq\; \delta_\ell(\mathbf{L}) + w_{min}$$

for all $\ell \in [m]$. So,

$$W \;\geq\; \delta_j(\mathbf{L}) + (m-1)(\delta_j(\mathbf{L}) - w_{min})\,,$$

and we get

$$\begin{aligned}\mathsf{OPT}_\infty(\mathbf{w},m) &\geq \frac{W}{m} \\ &\geq \frac{\delta_j(\mathbf{L}) + (m-1)(\delta_j(\mathbf{L}) - w_{min})}{m}\,.\end{aligned}$$

This implies

$$\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} \;\leq\; \frac{m\delta_j(\mathbf{L})}{m\delta_j(\mathbf{L}) - (m-1)w_{min}}\,.$$

Clearly, this expression is strictly increasing in w_{min}. Since $w_{min} \leq \frac{\delta_j(\mathbf{L})}{2}$, we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} &\leq \frac{m\delta_j(\mathbf{L})}{m\delta_j(\mathbf{L}) - (m-1)\cdot\frac{\delta_j(\mathbf{L})}{2}} \\ &= \frac{m}{m - \frac{m-1}{2}} \\ &= 2 - \frac{2}{m+1}\,,\end{aligned}$$

proving the upper bound.
Lower bound: To establish the lower bound, consider the following instance: There are m identical links and $n = m(m-1)+2$ users with traffics $w_1 = w_2 = 1$ and $w_i = \frac{1}{m}$ for $i \in [n] \setminus [2]$.

(1.) Consider the pure assignment $\mathbf{L}$ that assigns user 1 to link 1, user 2 to link 2, $m(m-2)$ users to the remaining $m-2$ links in a uniform way, and each of the remaining m users to links $1,2,\ldots,m$, respectively. Clearly, the load on every link $j \in [m]$ is $\delta_j(\mathbf{L}) = 1 + \frac{1}{m}$. Thus,

$$\begin{aligned}\mathsf{OPT}_\infty(\mathbf{w},m) &\leq \mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) \\ &= 1 + \frac{1}{m}\,.\end{aligned} \tag{4.17}$$

(2.) Consider now the pure assignment $\mathbf{L}'$ that assigns users 1 and 2 to link 1, and the remaining $m(m-1)$ users to the remaining $m-1$ links in a uniform way. The load on link 1 is $\delta_1(\mathbf{L}') = 2$ whereas the load on each remaining link $j \in [m] \setminus \{1\}$ is

$$\delta_j(\mathbf{L}') \;=\; \frac{m(m-1)}{m-1}\cdot\frac{1}{m} \;=\; 1\,.$$

Clearly, $\mathbf{L}'$ is a Nash equilibrium with

$$\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}') \;=\; 2\,. \tag{4.18}$$

Combining Equation (4.17) (page 107) and Equation (4.18), we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}')}{\mathsf{OPT}_\infty(\mathbf{w},m)} &\geq \frac{2}{1+\frac{1}{m}}\\ &= 2-\frac{2}{m+1}\,,\end{aligned}$$

as needed. ■

Theorem 4.29 *Consider the model of arbitrary users and identical links. If, for any* ε *with* $0<\varepsilon\leq 1-\frac{2}{m+1}$, WORST PURE NE *is* $(2-\frac{2}{m+1}-\varepsilon)$*-approximable, then* $\mathcal{P}=\mathcal{NP}$.

Proof: We prove the claim by reduction from PARTITION, that is, we employ a polynomial time transformation from PARTITION to WORST PURE NE. For any ε with $0<\varepsilon\leq 1-\frac{2}{m+1}$, given an instance of PARTITION, we construct an instance of WORST PURE NE such that if we had a polynomial-time $(2-\frac{2}{m+1}-\varepsilon)$-approximation algorithm for WORST PURE NE, then we could decide whether an instance of PARTITION is positive in polynomial time. From this construction the theorem then follows. Consider an arbitrary instance of PARTITION with

$$s(u_i) \;\leq\; \xi \;<\; \frac{S}{K}\cdot\min\left\{\frac{2}{2-\frac{2}{m+1}-\varepsilon}-\frac{m+1}{m}\,,\;\frac{m+1}{m}\cdot\left(1-\frac{2}{m+1}-\varepsilon\right)\right\}$$

for all $u_i\in\mathcal{U}$, where $S=\sum_{u_i\in\mathcal{U}}s(u_i)$. Clearly, we can make this property hold by adding a multiple of K of items of *sufficiently large* size. From this instance we construct an instance for the stated problem as follows:

- There are $n=|\mathcal{U}|+2$ users with

$$w_i \;=\; \begin{cases} s(u_i) & \text{if } i\in[|\mathcal{U}|],\\ \frac{S}{K} & \text{if } i\in\{n-1,n\}.\end{cases}$$

- There are $m=K+1$ identical links.

Clearly, this is a polynomial time transformation. We prove that this is a transformation from PARTITION to WORST PURE NE.

(1.) The instance of PARTITION is positive:
Consider a partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1,\ldots,\mathcal{U}_K$ such that $\sum_{u_i\in\mathcal{U}_j}s(u_i)=\frac{S}{K}$ for all $j\in[K]$. Use this partition to define a pure assignment $\mathbf{L}$ as follows:

- For each item $u_i\in\mathcal{U}_j$, $j\in[K]$, user i is assigned to link j.
- Users $n-1$ and n are assigned to link m.

We have

$$\delta_j(\mathbf{L}) \;=\; \sum_{u_i \in \mathcal{U}_j} s(u_i) \;=\; \frac{S}{K}$$

for all $j \in [K]$, and

$$\delta_m(\mathbf{L}) \;=\; w_{n-1} + w_n \;=\; \frac{2S}{K}\,.$$

Clearly, all users in $[|\mathcal{U}|]$ are satisfied. Since

$$\delta_m(\mathbf{L}) \;=\; \delta_j(\mathbf{L}) + w_{n-1} \;=\; \delta_j(\mathbf{L}) + w_n$$

for all $j \in [K]$, users $n-1$ and n are also satisfied. So, $\mathbf{L}$ is a pure Nash equilibrium. Moreover,

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}) \;=\; \frac{2S}{K}\,.$$

Now, consider any pure Nash equilibrium $\mathbf{L}'$ in which the users $i \in [|\mathcal{U}|]$ are not assigned to K links such that they cause load $\frac{S}{K}$ on each of these links. Assume, by way of contradiction, that users $n-1$ and n are assigned to the same link. Say that were link m. Clearly, there exists a link, say 1, with $\delta_1(\mathbf{L}') < \frac{S}{K}$. Hence,

$$\begin{aligned}\delta_m(\mathbf{L}') &\geq w_{n-1} + w_n \\ &= \frac{S}{K} + w_n \\ &> \delta_1(\mathbf{L}') + w_n\,,\end{aligned}$$

showing that users $n-1$ and n are unsatisfied, a contradiction to the fact that $\mathbf{L}'$ is a Nash equilibrium. It follows that users $n-1$ and n are assigned to different links in the Nash equilibrium $\mathbf{L}'$. Now, assume, by way of contradiction, that $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') > \frac{(m+1)\frac{S}{K}}{m} + \xi$. Clearly, there exists a links, say m, with

$$\delta_m(\mathbf{L}') \;=\; \mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \;>\; \frac{(m+1)\frac{S}{K}}{m} + \xi\,.$$

Since users $n-1$ and n are assigned to different links, this implies that a user $i \in [|\mathcal{U}|]$ is assigned to link m. Moreover, since the total load is $(m+1)\frac{S}{K}$, there exists a link, say 1, with

$$\delta_1(\mathbf{L}') \;<\; \frac{(m+1)\frac{S}{K}}{m}\,.$$

So,

$$\begin{aligned}\delta_m(\mathbf{L}') &> \frac{(m+1)\frac{S}{K}}{m} + \xi \\ &\overset{w_i \leq \xi}{\geq} \frac{(m+1)\frac{S}{K}}{m} + w_i \\ &> \delta_1(\mathbf{L}') + w_i\,,\end{aligned}$$

showing that user i is unsatisfied, again a contradiction to the fact that $\mathbf{L}'$ is a Nash equilibrium. We get

$$\mathsf{SC}_\infty(\mathbf{w},n,\mathbf{L}') \leq \frac{(m+1)\frac{S}{K}}{m}+\xi\,.$$

Hence,

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}')} &\geq \frac{2\cdot\frac{S}{K}}{\frac{(m+1)\frac{S}{K}}{m}+\xi}\\ &> \frac{2\cdot\frac{S}{K}}{\frac{S}{K}\cdot\frac{(m+1)}{m}+\frac{S}{K}\cdot\left(\frac{2}{2-\frac{2}{m+1}-\varepsilon}-\frac{m+1}{m}\right)}\\ &= 2-\frac{2}{m+1}-\varepsilon\,.\end{aligned}$$

Thus, no such Nash equilibrium $\mathbf{L}'$ approximates the worst pure Nash equilibrium within the claimed factor.

(2.) The instance of PARTITION is negative:
For any partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1,\ldots,\mathcal{U}_K$, there exists a $j\in[K]$ such that $\sum_{u_i\in\mathcal{U}_j}s(u_i)<\frac{S}{K}$. Thus, the users $i\in[|\mathcal{U}|]$ can not be assigned to K links such that they cause load $\frac{S}{K}$ on each of these links. As seen in the previous case, this implies that the social cost of any pure Nash equilibrium $\mathbf{L}$ is bounded by

$$\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) \leq \frac{(m+1)\frac{S}{K}}{m}+\xi\,.$$

Moreover,

$$\begin{aligned}\mathsf{OPT}_\infty(\mathbf{w},m) &\geq \frac{W}{m}\\ &= \frac{(m+1)\frac{S}{K}}{m}\,,\end{aligned}$$

and we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} &\leq \frac{\frac{(m+1)\frac{S}{K}}{m}+\xi}{\frac{(m+1)\frac{S}{K}}{m}}\\ &\leq \frac{\frac{(m+1)\frac{S}{K}}{m}+\frac{(m+1)\frac{S}{K}}{m}\left(1-\frac{2}{m+1}-\varepsilon\right)}{\frac{(m+1)\frac{S}{K}}{m}}\\ &= \frac{1+1-\frac{2}{m+1}-\varepsilon}{1}\\ &= 2-\frac{2}{m+1}-\varepsilon\,.\end{aligned}$$

Thus, all Nash equilibria approximate the worst pure Nash equilibrium within the claimed factor.

Therefore, if we had a polynomial-time $(2-\frac{2}{m+1}-\varepsilon)$-approximation algorithm for WORST PURE NE, we could use it to decide whether an instance of PARTITION is positive in the following way: We apply the approximation algorithm to the corresponding instance of WORST PURE NE and we answer *yes* if and only if it returns a solution with social cost $\frac{2S}{K}$. ■

4.4.2 Mixed Nash Equilibria

Fotakis *et al.* [50] showed that, in contrast to pure Nash equilibria, it is $\#\mathcal{P}$-complete to compute the social cost for a mixed Nash equilibrium (Theorem 4.30). However, there exists a fully polynomial randomized approximation scheme for NASH EQUILIBRIUM SOCIAL COST (Theorem 4.31). In contrast to these results, for a given instance and given indicator variables, a Nash equilibrium can be computed in polynomial time (Theorem 4.32).

For *two* identical links, the price of anarchy is $\frac{3}{2}$ (Theorem 4.33). For an arbitrary number of links, Czumaj and Vöcking [29] and Koutsoupias *et al.* [92] independently proved the upper bound $O(\frac{\log m}{\log\log m})$ (Theorem 4.34, page 112). This bound is asymptotically tight since throwing *m balls* into *m bins* results in an expected maximum number of balls in a bin of the same order of magnitude (Theorem 4.35, page 112).

Theorem 4.30 (Fotakis *et al.* [50]) *Consider the model of arbitrary users and identical links. Then,* NASH EQUILIBRIUM SOCIAL COST *is $\#\mathcal{P}$-complete.*

Theorem 4.31 (Fotakis *et al.* [50]) *Consider the model of arbitrary users and identical links. Then, there exists a fully polynomial randomized approximation scheme for* NASH EQUILIBRIUM SOCIAL COST.

Theorem 4.32 (Monien [111]) *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w},m)$ and indicator variables $I_{ij}\in\{0,1\}$ for all $i\in[n]$ and $j\in[m]$, there exists a Nash equilibrium $\mathbf{P}$ if and only if the system of inequalities*

$$\begin{aligned}
\delta_j(\mathbf{P})+w_i &\geq \lambda_i(\mathbf{P}) && \text{for all } i\in[n], j\in[m]\\
\delta_j(\mathbf{P}) &= \sum_{i\in[n]} I_{ij}(\delta_j(\mathbf{P})+w_i-\lambda_i(\mathbf{P})) && \text{for all } j\in[m]\\
w_i &= \sum_{j\in[m]} I_{ij}(\delta_j(\mathbf{P})+w_i-\lambda_i(\mathbf{P})) && \text{for all } i\in[n]
\end{aligned}$$

has a solution. For every solution $(\delta_1(\mathbf{P}),\ldots,\delta_m(\mathbf{P}),\lambda_1(\mathbf{P}),\ldots,\lambda_n(\mathbf{P}))$,

$$p_{ij} = \begin{cases} \dfrac{\delta_j(\mathbf{P})+w_i-\lambda_i(\mathbf{P})}{w_i} & \text{for all } i\in[n], j\in[m] \text{ with } I_{ij}=1,\\ 0 & \text{otherwise,}\end{cases}$$

defines a Nash equilibrium.

Theorem 4.33 (Koutsoupias and Papadimitriou [93]) *Consider the model of arbitrary users and two identical links. Then,*

$$\mathsf{PoA} = \frac{3}{2}.$$

Theorem 4.34 (Czumaj and Vöcking [29], Koutsoupias *et al.* [92]) *Consider the model of arbitrary users and identical links. Then,*

$$\mathsf{PoA} \leq \Gamma^{-1}(m) + \Theta(1) = O\left(\frac{\log m}{\log\log m}\right).$$

Theorem 4.35 (Gonnet [62], Koutsoupias and Papadimitriou [93]) *Consider the model of identical users and identical links. Then,*

$$\mathsf{PoA} = \Omega\left(\frac{\log m}{\log\log m}\right).$$

4.4.3 Fully Mixed Nash Equilibria

In the remainder of this section, we consider fully mixed Nash equilibria. Mavronicolas and Spirakis [107] showed that in the model of identical links there always exists a unique fully mixed Nash equilibrium (Theorem 4.36). Thus, we can compare it to a worst Nash equilibrium. The following results provide evidence for the FMNE Conjecture. In particular, the FMNE Conjecture holds for pure Nash equilibria (Theorem 4.37, page 113). Moreover, the conjecture is valid in case of two arbitrary users and identical links (Theorem 4.38, page 113), and in case of an even number of identical users and two identical links (Theorem 4.39, page 113). In contrast to the yet unproved claim of the FMNE Conjecture, each user indeed experiences the worst expected individual cost in the fully mixed Nash equilibrium (Proposition 4.40, page 113).

Theorem 4.36 (Mavronicolas and Spirakis [107]) *Consider the model of arbitrary users and identical links. Then, for any instance $(\mathbf{w}, m)$, there exists a unique fully mixed Nash equilibrium $\mathbf{F}$ with $f_{ij} = \frac{1}{m}$ for all users $i \in [n]$ and links $j \in [m]$.*

Example 4.7 (continued) *For the given instance, Theorem 4.36 implies that in the fully mixed Nash equilibrium $f_{ij} = \frac{1}{3}$ for all $i \in [7]$ and $j \in [3]$. Thus, every assignment is equiprobable (with probability $\frac{1}{3^7} = \frac{1}{2187}$), and the social cost of $\mathbf{F}$ reduces to*

$$\begin{aligned}\mathsf{SC}_\infty(\mathbf{w}, 3, \mathbf{F}) &= \frac{1}{2187} \sum_{\langle \ell_1, \ell_2, \ldots, \ell_n \rangle \in [3]^7} \left(\max_{j \in [3]} \frac{\sum_{k \in [n]: \ell_k = j} w_k}{c_j} \right) \\ &= \frac{1163}{81}.\end{aligned}$$

Recall that $\mathsf{OPT}_\infty(\mathbf{w}, 3) = 9$ for the given instance. Theorem 4.11 (page 98) implies that

$$\begin{aligned}\mathsf{SC}_\infty(\mathbf{w}, 3, \mathbf{L}) &\leq \left(\frac{4}{3} - \frac{1}{3m}\right) \cdot \mathsf{OPT}_\infty(\mathbf{w}, 3) \\ &= \frac{11}{9} \cdot 9 \\ &= 11 \\ &< \frac{1163}{81} \\ &= \mathsf{SC}_\infty(3, m, \mathbf{F})\end{aligned}$$

for the pure Nash equilibrium $\mathbf{L}$ computed by LPT.

Theorem 4.37 (Gairing *et al.* [59]) *Consider the model of arbitrary users and identical links, restricted to pure Nash equilibria. Then, the* FMNE *Conjecture is valid.*

Theorem 4.38 (Fotakis *et al.* [50]) *Consider the model of two arbitrary users and identical links. Then, the* FMNE *Conjecture is valid.*

Theorem 4.39 (Lücking *et al.* [103]) *Consider the model of an even number of identical users and two identical links. Then, the* FMNE *Conjecture is valid.*

Proposition 4.40 (Gairing *et al.* [59]) *Consider the model of arbitrary users and identical links. Then, for any instance* $(\mathbf{w}, m)$ *and associated Nash equilibrium* $\mathbf{P}$*, it is* $\lambda_i(\mathbf{P}) \leq \lambda_i(\mathbf{F})$.

4.5 Makespan Social Cost and Related Links

In this section, we are engaged in the KP-model with makespan social cost and related links. Often the term *uniform links* is used to refer to this model in literature. Subsection 4.5.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.5.2 hold for general (i.e. mixed) Nash equilibria. Subsection 4.5.3 concentrates on the fully mixed Nash equilibrium. In order to illustrate the results, we use the following instance.

Example 4.41 *Consider the following instance* $(\mathbf{w}, \mathbf{c})$*: We have* $n = 5$ *arbitrary users and* $m = 4$ *related links. There are one user with traffic* $w_1 = 4$*, two users with traffics* $w_2 = w_3 = 3$ *and two users with traffics* $w_4 = w_5 = 2$*. The capacities of the links are* $c_1 = 5$*,* $c_2 = 4$*,* $c_3 = 3$ *and* $c_4 = 2$.

4.5.1 Pure Nash Equilibria

4.5.1.1 Computation of Nash Equilibria

Identical Users. We first consider the model of identical users. For any instance $(n, \mathbf{c})$ and associated pure assignment $\mathbf{L}$, we can write $\delta_j(\mathbf{L}) = n_j(\mathbf{L})$ for all $j \in [m]$, where $n_j(\mathbf{L})$ is the number of users assigned to link j. This is possible since all users are identical, that is, $w_i = 1$ for all $i \in [n]$. In this model, all pure Nash equilibria have optimum social cost (Proposition 4.42). In a more general setting, Gairing *et al.* [58] gave an algorithm which computes a pure Nash equilibrium in $O(m \log n \log m)$ time (Theorem 4.43, page 114). Again, we can also use selfish steps to convert any given assignment into a pure Nash equilibrium. The length of such a sequence may be $\Omega(nm)$ (Theorem 4.44, page 114). However, since a pure assignment is a Nash equilibrium if and only if the users assigned to links with maximum latency are satisfied (Lemma 4.45, page 115), the length of a sequence of *greedy* selfish steps using the rule MAX LOAD MACHINE is at most n (Proposition 4.46, page 115).

Proposition 4.42 *Consider the model of identical users and related links. Then, for any instance* (n, m) *and associated pure Nash equilibrium* $\mathbf{L}$*, it is* $\mathsf{SC}_\infty(n, \mathbf{c}, \mathbf{L}) = \mathsf{OPT}_\infty(n, \mathbf{c})$.

Proof: Fix any instance $(n, \mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\infty(n, \mathbf{c}, \mathbf{L}) = \mathsf{OPT}_\infty(n, \mathbf{c})$. Such a Nash equilibrium exists by Theorem 4.6 (page 91). Assume, by way of

contradiction, that there exists a pure Nash equilibrium $\mathbf{L}'$ with $\mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}') > \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L})$. Then, there exist links $j_1, j_2 \in [m]$ with

$$\begin{aligned}\mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}') &= \frac{n_{j_2}(\mathbf{L}')}{c_{j_2}} \\ &> \frac{n_{j_1}(\mathbf{L})}{c_{j_1}} \\ &= \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L})\,.\end{aligned}$$

If $n_{j_2}(\mathbf{L}') \leq n_{j_2}(\mathbf{L})$, then,

$$\begin{aligned}\mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}') &= \frac{n_{j_2}(\mathbf{L}')}{c_{j_2}} \\ &\leq \frac{n_{j_2}(\mathbf{L})}{c_{j_2}} \\ &\leq \frac{n_{j_1}(\mathbf{L})}{c_{j_1}} \\ &= \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}) \\ &< \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}')\,,\end{aligned}$$

a contradiction. So, assume $n_{j_2}(\mathbf{L}') > n_{j_2}(\mathbf{L})$. Then, there exists a link $\ell \in [m]$ with $n_\ell(\mathbf{L}') < n_\ell(\mathbf{L})$, that is, $n_\ell(\mathbf{L}') \leq n_\ell(\mathbf{L}) - 1$. It follows that

$$\begin{aligned}\mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}') &= \frac{n_{j_2}(\mathbf{L}')}{c_{j_2}} \\ &> \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}) \\ &= \frac{n_{j_1}(\mathbf{L})}{c_{j_1}} \\ &\geq \frac{n_\ell(\mathbf{L})}{c_\ell} \\ &\geq \frac{n_\ell(\mathbf{L}')+1}{c_\ell}\,,\end{aligned}$$

implying that

$$\frac{n_{j_2}(\mathbf{L}')}{c_{j_2}} > \frac{n_\ell(\mathbf{L}')+1}{c_\ell}\,.$$

This shows that all users assigned to link j_2 in $\mathbf{L}'$ are unsatisfied, a contradiction to the definition of Nash equilibrium. ■

Theorem 4.43 (Gairing *et al.* [58]) *Consider the model of identical users and related links. Then, a pure Nash equilibrium can be computed in $O(m \log n \log m)$ time.*

Theorem 4.44 (Even-Dar *et al.* [42]) *Consider the model of identical users and related links. Then, there exists an instance $(n,\mathbf{c})$ and associated pure assignment for which the maximum length of a sequence of (not necessarily greedy) selfish steps is at least $\Omega(nm)$ before reaching a Nash equilibrium.*

Lemma 4.45 *Consider the model of identical users and related links. Then, a pure assignment* $\mathbf{L}$ *associated to an instance* $(n,\mathbf{c})$ *is a Nash equilibrium if and only if all users on links* $j \in [m]$ *with* $\frac{n_j(\mathbf{L})}{c_j} = \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L})$ *are satisfied.*

Proof: Fix any instance $(n,\mathbf{c})$ and associated pure assignment $\mathbf{L}$. If $\mathbf{L}$ is a Nash equilibrium, then, by definition of Nash equilibrium, all users are satisfied. So, assume, by way of contradiction, that all users on a link $j_1 \in [m]$ with $\frac{n_{j_1}(\mathbf{L})}{c_{j_1}} = \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L})$ are satisfied, and that there exists a user on a link $j_2 \in [m]$, $j_2 \neq j_1$, who is unsatisfied, that is,

$$\frac{n_{j_2}(\mathbf{L})}{c_{j_2}} > \frac{n_\ell(\mathbf{L})+1}{c_\ell}$$

for some $\ell \in [m]$. This implies

$$\frac{n_{j_1}(\mathbf{L})}{c_{j_1}} \geq \frac{n_{j_2}(\mathbf{L})}{c_{j_2}} > \frac{n_\ell(\mathbf{L})+1}{c_\ell},$$

showing that all users on link j_1 are unsatisfied, a contradiction to our assumption. Thus, $\mathbf{L}$ is a Nash equilibrium. ■

Proposition 4.46 *Consider the model of identical users and related links. Then, for any instance* $(n,\mathbf{c})$ *and associated pure assignment, the length of a sequence of greedy selfish steps using the rule* MAX LOAD MACHINE *is at most n before reaching a Nash equilibrium.*

Proof: Fix any instance $(n,\mathbf{c})$ and associated pure assignment $\mathbf{L}$. By Lemma 4.45, we have to apply the rule MAX LOAD MACHINE as long as users on links with maximum latency are unsatisfied. We proceed by showing that such a greedy selfish step of a user from a link $j_1 \in [m]$ with $\frac{n_{j_1}(\mathbf{L})}{c_{j_1}} = \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L})$ to a link $j_2 \in [m]$ makes no satisfied user unsatisfied.

Denote $\mathbf{L}'$ the pure assignment after this greedy selfish step. Assume, by way of contradiction, that a satisfied user $i \in [n]$ became unsatisfied. Clearly, all users on link j_2 stay satisfied, showing that $\ell_i \neq j_2$. Moreover, since only the load on link j_1 decreased, user i can only improve by moving to link j_1. We have

$$\begin{aligned}
\lambda_{i\ell_i}(\mathbf{L}') &> \lambda_{ij_1}(\mathbf{L}') \\
&= \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}) \\
&\geq \lambda_{i\ell_i}(\mathbf{L}) \\
&= \lambda_{i\ell_i}(\mathbf{L}'),
\end{aligned}$$

a contradiction. Thus, selfish steps make no satisfied user unsatisfied, proving that the number of selfish steps is bounded by n, as needed. ■

Arbitrary Users. We now examine arbitrary users. Fotakis *et al.* [50] showed that the LPT-algorithm (see Algorithm 3, page 93) can also be used to compute some pure Nash equilibrium in the model of related links. Friesen [54] proved that the computed assignment approximates an optimum assignment within factor $\frac{5}{3}$ (see Table 4.2, page 92). We use the instance in Example 4.41 (page 113) to illustrate the LPT-algorithm.

Theorem 4.47 (Fotakis *et al.* [50], Friesen [54]) *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w}, m)$, LPT *computes a pure Nash equilibrium* $\mathbf{L}$ *with*

$$\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L}) \;\leq\; \frac{5}{3}\cdot\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})\,,$$

using $O((n+m)\log m)$ *time.*

Example 4.41 (continued) *For the given instance,* LPT *returns a pure Nash equilibrium* $\mathbf{L} = \langle 1,2,3,4,1\rangle$ *with social cost* $\frac{6}{5}$ *whereas the optimum assignment* $\langle 2,3,1,1,4\rangle$ *has social cost* 1 *(see Figure 4.5). Thus,*

$$\frac{\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})} \;=\; \frac{6}{5}\,.$$

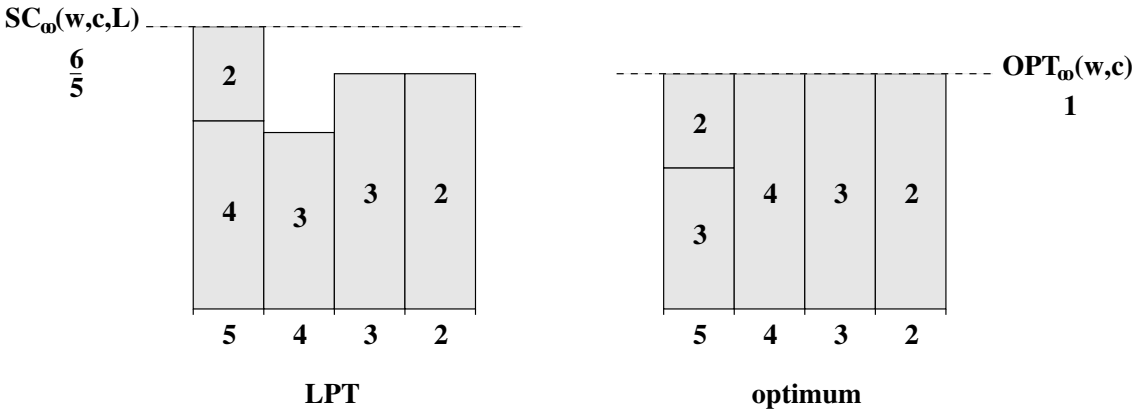


Figure 4.5: Pure Nash equilibrium $\mathbf{L} = \langle 1,2,3,4,1\rangle$ returned by LPT applied to the instance in Example 4.41 (page 113) (left hand side), and an optimum assignment $\langle 2,3,1,1,4\rangle$ of this instance (right hand side).

Although each sequence of selfish steps eventually ends in a pure Nash equilibrium, it is unknown whether there always exists such a sequence of length polynomial in the number of users and links, respectively. Fabrikant *et al.* [44] showed that the local computation of a pure Nash equilibrium (i.e. with help of selfish steps) in general networks is $\mathcal{PLS}$-complete (see [78] for a definition of the class $\mathcal{PLS}$). For the KP-model, however, the complexity is unknown.

Feldmann *et al.* [46] introduced another approach to nashify a pure assignment $\mathbf{L}$ associated to a given instance $(\mathbf{w},\mathbf{c})$. Their algorithm, called NASHIFY-RELATED and stated as Algorithm 5 (page 117), does not only use selfish steps but also **moves** in which the individual costs of users increase. A crucial observation for the proof of the correctness of NASHIFY-RELATED is stated in Lemma 4.48 (page 117) (which is a generalization of Lemma 4.15, page 101). It shows that a greedy selfish step of a user from its current link to a link with at least the same capacity can only make users with smaller traffic unsatisfied. NASHIFY-RELATED works in two phases:

(1.) In the first phase, NASHIFY-RELATED fills up links with small capacities with users with small traffic as close to $\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L})$ as possible (but without exceeding $\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L})$), and it collects all these users in set S.

(2.) In the second phase, NASHIFY-RELATED performs greedy selfish steps for unsatisfied users in S.

Algorithm 5 (NASHIFY-RELATED)

Input: an instance $(\mathbf{w},\mathbf{c})$ and associated assignment $\mathbf{L}$
Output: an assignment $\mathbf{L}'$

```
(1)   begin
      // phase 1
(2)   i ← n;
(3)   S ← {n};
(4)   while i ≥ 1 do
(5)       move user i to link with highest possible index without exceeding SC_∞(w,c,L);
(6)       if i was moved or i ∈ S or ℓ_i ≤ ℓ_{i+1} then
(7)           S ← S ∪ {i};
(8)           i ← i − 1;
(9)       else
(10)          move user i to link with smallest possible index without exceeding SC_∞(w,c,L);
(11)          if i was moved then
(12)              S ← S ∪ {i};
(13)              i ← n;
(14)          else break;
      // phase 2
(15)  while ∃i ∈ S do
(16)      make greedy selfish step for user i = min(S);
(17)      S ← S\{i};
(18)  return the resulting assignment L';
(19)  end
```

Note that, whenever user $i \in [n]$ is moved, its strategy ℓ_i has to be updated, which is not explicitly mentioned in the pseudo-code. Applying Lemma 4.49 (page 118) and implementing the algorithm in a proper way yields the correctness of NASHIFY-RELATED and running time $O(m^2 n)$ (Theorem 4.50, page 119).

Lemma 4.48 *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w},\mathbf{c})$ *and associated pure assignment, a greedy selfish step of an unsatisfied user* $i_1 \in [n]$ *with traffic* w_{i_1} *from a link* $j_1 \in [m]$ *to a link* $j_2 \in [m]$ *with* $c_{j_1} \leq c_{j_2}$ *makes no satisfied user* $i_2 \in [n]$ *with traffic* $w_{i_2} \geq w_{i_1}$ *unsatisfied.*

Proof: Fix any instance $(\mathbf{w},\mathbf{c})$ and associated pure assignment $\mathbf{L}$, and denote $\mathbf{L}'$ the assignment after the greedy selfish step of user i_1. Assume, by way of contradiction, that user i_2 becomes unsatisfied due to the greedy selfish step of user i_1. Since only the load on link j_1 and j_2 changed, we have to distinguish between two cases:

(1.) First, assume that $\ell_{i_2} \neq j_2$, and that user i_2 can improve by moving to link j_1. Since user i_1 improved by moving from link j_1 to link j_2, we know that

$$\frac{\delta_{j_1}(\mathbf{L})}{c_{j_1}} > \frac{\delta_{j_2}(\mathbf{L}) + w_{i_1}}{c_{j_2}}. \tag{4.19}$$

Therefore,

$$\begin{aligned}
\frac{\delta_{\ell_{i_2}}(\mathbf{L}')}{c_{\ell_{i_2}}} &> \frac{\delta_{j_1}(\mathbf{L}') + w_{i_2}}{c_{j_1}} \\
&= \frac{\delta_{j_1}(\mathbf{L}) - w_{i_1} + w_{i_2}}{c_{j_1}} \\
&= \frac{\delta_{j_1}(\mathbf{L})}{c_{j_1}} + \frac{w_{i_2} - w_{i_1}}{c_{j_1}} \\
&\overset{(4.19)}{>} \frac{\delta_{j_2}(\mathbf{L}) + w_{i_1}}{c_{j_2}} + \frac{w_{i_2} - w_{i_1}}{c_{j_1}} \\
&\overset{c_{j_1} \leq c_{j_2}}{\geq} \frac{\delta_{j_2}(\mathbf{L}) + w_{i_1}}{c_{j_2}} + \frac{w_{i_2} - w_{i_1}}{c_{j_2}} \\
&= \frac{\delta_{j_2}(\mathbf{L}) + w_{i_2}}{c_{j_2}}.
\end{aligned}$$

So, if user i_2 can improve by moving to link j_1 after the greedy selfish step of user i_1, then user i_2 could already improve by moving to link j_1 before the greedy selfish step of user i_1. This is a contradiction to the assumption that user i_2 was satisfied.

(2.) Now, consider the case $\ell_{i_2} = j_2$. For all $\ell \in [m] \setminus \{j_2\}$, we have

$$\begin{aligned}
\frac{\delta_{\ell}(\mathbf{L}') + w_{i_2}}{c_{\ell}} &\overset{w_{i_2} \geq w_{i_1}}{\geq} \frac{\delta_{\ell}(\mathbf{L}') + w_{i_1}}{c_{\ell}} \\
&\geq \frac{\delta_{j_2}(\mathbf{L}')}{c_{j_2}}.
\end{aligned}$$

Therefore, user i_2 is satisfied after the greedy selfish step of user i_1, a contradiction to the assumption that user i_2 becomes unsatisfied. ■

Lemma 4.49 (Feldmann *et al.* [46]) *Consider the model of arbitrary users and related links. Then, after phase 1 of* NASHIFY-RELATED*,*

(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.) if $i, i+1 \in S$, then $\ell_i \leq \ell_{i+1}$, and

(4.) every user $i \in S$ can only improve by moving to a link with smaller index.

Theorem 4.50 (Feldmann *et al.* [46]) *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w}, \mathbf{c})$ *and associated pure assignment* $\mathbf{L}$*,* NASHIFY-RELATED *computes a pure Nash equilibrium* $\mathbf{L}'$ *from* $\mathbf{L}$ *with* $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L}') \leq \mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L})$ *using at most* $(m+1)n$ *moves and* $O(m^2 n)$ *time.*

Example 4.41 (continued) *For the given instance and associated pure assignment* $\mathbf{L} = \langle 2, 1, 1, 1, 1 \rangle$*,* NASHIFY-RELATED *(in phase 1) first moves users* 4 *and* 5 *to link* 4 *and then users* 2 *and* 3 *to link* 3 *without exceeding* $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L})$*. Then, (in phase 2)* NASHIFY-RELATED *uses selfish steps to reach a Nash equilibrium (see Figure 4.6).*

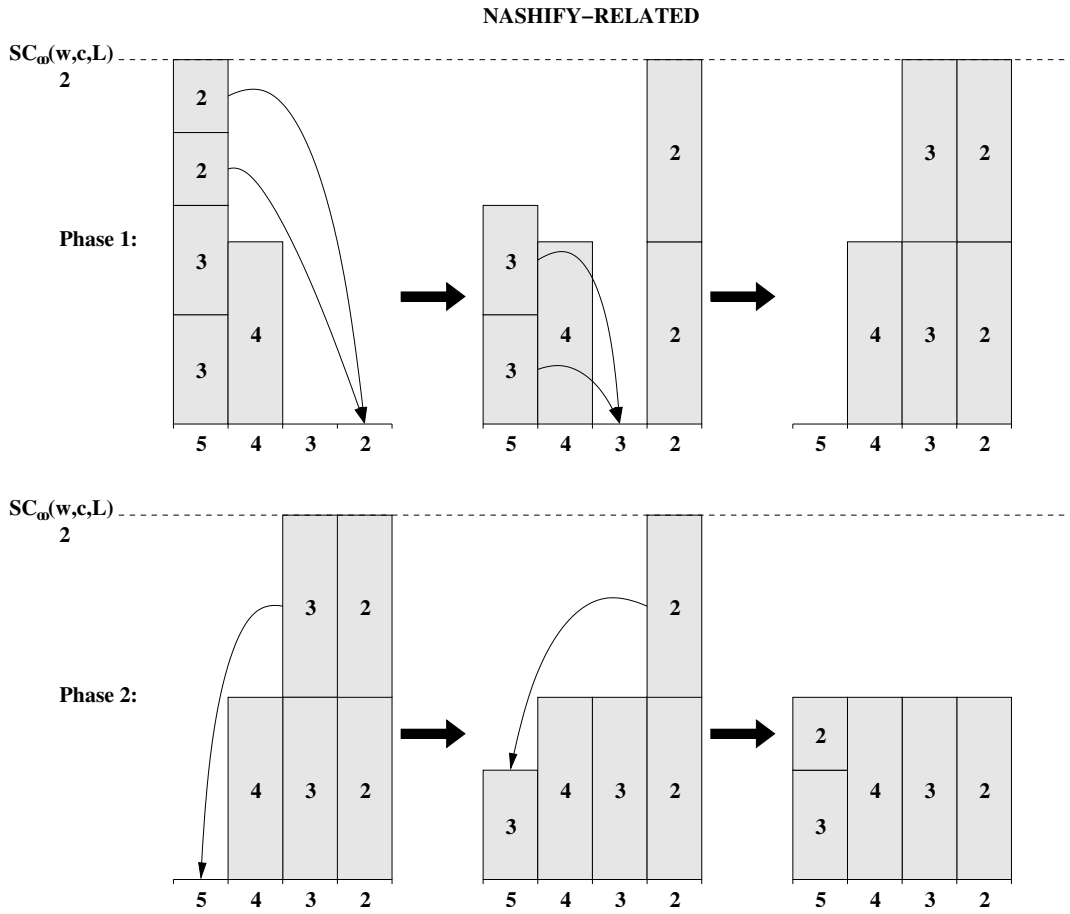


Figure 4.6: NASHIFY-RELATED, applied to the instance in Example 4.41 (page 113) and associated pure assignment $\mathbf{L} = \langle 2, 1, 1, 1, 1 \rangle$. In phase 1, each small arrow corresponds to a *move* (top) whereas in phase 2, each small arrow corresponds to a *greedy selfish step* (bottom).

4.5.1.2 Computation of Best Nash Equilibria

In contrast to the model of identical users where it is trivial to compute a best pure Nash equilibrium (see Proposition 4.42, page 113, and Theorem 4.43, page 114), Theorem 4.22 (page 104) implies that BEST PURE NE is $\mathcal{NP}$-complete for arbitrary users. Clearly, there

exists no pseudo-polynomial algorithm to solve BEST PURE NE since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense. For constant m, however, we can give such an algorithm (Theorem 4.51). Moreover, the algorithm NASHIFY-IDENTICAL enables us to use any approximation algorithm for scheduling *n jobs* on m related *machines* (see Table 4.2, page 92, for a list of such approximation algorithms) to get an approximation algorithm for BEST PURE NE. In particular, using the PTAS of Hochbaum and Shmoys [71], this approach yields a PTAS for BEST PURE NE (Theorem 4.52). We cannot expect to find an FPTAS since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense [50].

Theorem 4.51 *Consider the model of arbitrary users and related links. Then, there exists a pseudo-polynomial-time algorithm for* m-BEST PURE NE.

Proof: See Theorem 4.104 (page 161) for a generalization of this result. ■

Theorem 4.52 (Feldmann *et al.* [46]) *Consider the model of arbitrary users and related links. Then, there exists a* PTAS *for* BEST PURE NE.

4.5.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

We next turn our attention to worst pure Nash equilibria. For the model of identical users, such a Nash equilibrium can be computed in polynomial time (see Proposition 4.42, page 113, and Theorem 4.43, page 114). For arbitrary users, Theorem 4.26 (page 106) implies that WORST PURE NE is $\mathcal{NP}$-complete. Again, there exists no pseudo-polynomial algorithm to solve WORST PURE NE since WORST PURE NE is $\mathcal{NP}$-complete in the strong sense, but for constant m, we can give such an algorithm (Theorem 4.53).

For identical users, the price of anarchy is 1 (see Proposition 4.42, page 113). For arbitrary users, Czumaj and Vöcking [29] showed the asymptotically tight bound $\Gamma^{-1}(m)+1$ on the price of anarchy (Theorems 4.54 and Theorem 4.56, page 121). We will see later (Corollary 4.63, page 131) that the upper bound in Theorem 4.54 can be slightly improved to $\Gamma^{-1}(m)$. Clearly, Theorem 4.29 (page 108) implies that, for any ε with $0 < \varepsilon \leq 1 - \frac{2}{m+1}$, we can not hope to find a polynomial-time $(2 - \frac{2}{m+1} - \varepsilon)$-approximation algorithm for WORST PURE NE. Up to now, no other result is known.

Theorem 4.53 *Consider the model of arbitrary users and related links. Then, there exists a pseudo-polynomial-time algorithm for* m-WORST PURE NE.

Proof: See Theorem 4.105 (page 161) for a generalization of this result. ■

Theorem 4.54 (Czumaj and Vöcking [29]) *Consider the model of arbitrary users and related links, restricted to pure Nash equilibria. Then,*

$$\begin{aligned} \text{PoA} &\leq \min\left\{\Gamma^{-1}(m)+1\ ,\ 2\cdot\log\left(\frac{c_1}{c_m}\right)+O(1)\right\} \\ &= O\left(\min\left\{\frac{\log m}{\log\log m}\ ,\ \log\left(\frac{c_1}{c_m}\right)\right\}\right)\ . \end{aligned}$$

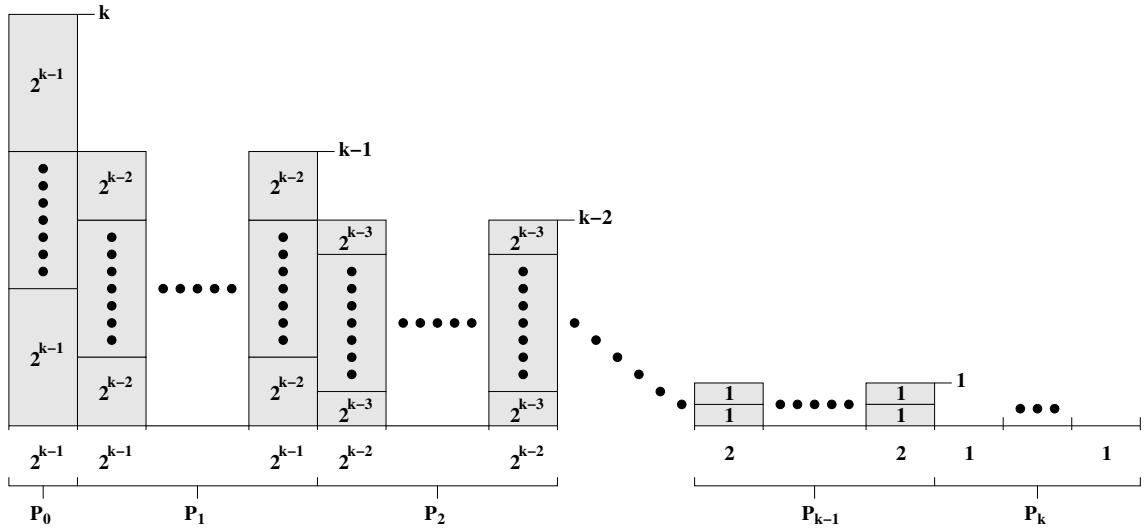


Figure 4.7: Instance and associated pure assignment in Example 4.55.

Example 4.55 *Fix any $k \in \mathbb{N}$, and consider the following instance $(\mathbf{w}, \mathbf{c})$ and associated pure assignment $\mathbf{L}$ (illustrated in Figure 4.7):*

- *We have k disjoint subsets $\mathcal{U}_1, \ldots \mathcal{U}_k$ with $|\mathcal{U}_1| = k$ users with traffics 2^{k-1} and*

$$|\mathcal{U}_i| \;=\; 2^{i-1} \cdot (k-1) \prod_{j \in [i-1]} (k-j)$$

  users with traffics 2^{k-i} for all $i \in [k] \setminus \{1\}$.

- *We have $(k+1)$ disjoint subsets $\mathcal{P}_0, \ldots, \mathcal{P}_k$ with $|\mathcal{P}_0| = 1$ links with capacity 2^{k-1}, $|\mathcal{P}_1| = |\mathcal{U}_1| - 1$ links with capacity 2^{k-1}, and $|\mathcal{P}_i| = |\mathcal{U}_i|$ links with capacity 2^{k-i} for all $i \in [k] \setminus \{1\}$.*

- *The assignment $\mathbf{L}$ is defined as follows: All users in $\mathcal{U}_1$ are assigned to the link in $\mathcal{P}_0$; on each link in $\mathcal{P}_i$, $i \in [k-1]$, there are $2(k-i)$ users from $\mathcal{U}_{i+1}$, respectively; the links from $\mathcal{P}_k$ remain empty.*

Theorem 4.56 *For every $k \in \mathbb{N}$, there exists an instance $(\mathbf{w}, \mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$ with*

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

Proof: Consider the instance $(\mathbf{w}, \mathbf{c})$ and associated pure assignment $\mathbf{L}$ given in Example 4.55. We first show in Claim 4.57 that $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = 1$, and that $\mathbf{L}$ is a pure Nash equilibrium with $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L}) = k$. We then show the lower bound on k.

Claim 4.57 *Consider the instance $(\mathbf{w}, \mathbf{c})$ and associated pure assignment $\mathbf{L}$ given in Example 4.55. Then,*

(1.) $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) = 1$, *and*

(2.) *the assignment* $\mathbf{L}$ *is a pure Nash equilibrium with social cost* $\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L}) = k$.

Proof:

(1.) The traffic of a user in $\mathcal{U}_i$ is equal to the capacity of a link in $\mathcal{P}_i$, and $|\mathcal{U}_i| = |\mathcal{P}_i|$ for all $i \in [k] \setminus \{1\}$. Moreover, the traffic of each user in $\mathcal{U}_1$ is equal to the capacity of a link in $\mathcal{P}_0 \cup \mathcal{P}_1$, and $|\mathcal{P}_0| + |\mathcal{P}_1| = |\mathcal{U}_1|$. So, each user can be assigned to a link with a capacity equal to its traffic such that all users are solo. Thus, $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) = 1$.

(2.) The latency of all links $j \in \mathcal{P}_i$, $i \in [k] \cup \{0\}$, is

$$\Lambda_j(\mathbf{L}) \;=\; \frac{\delta_j(\mathbf{L})}{c_j} \;=\; (k-i)\,.$$

Thus,

$$\delta_j(\mathbf{L}) \;=\; (k-i)\cdot c_j \;=\; \begin{cases} k\cdot 2^{k-1} & \text{if } j \in \mathcal{P}_0, \\ (k-i)\cdot 2^{k-i} & \text{if } j \in \mathcal{P}_i,\ i \in [k]. \end{cases} \tag{4.20}$$

Clearly, in $\mathbf{L}$ every user is assigned to exactly one link, and $\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L}) = k$. Thus, $\mathbf{L}$ is a valid assignment. We proceed to prove that $\mathbf{L}$ is a pure Nash equilibrium.

Assume, by way of contradiction, that $\mathbf{L}$ is not a pure Nash equilibrium. Then, there exists an unsatisfied user with traffic w on a link $j_1 \in \mathcal{P}_{i_1}$ who wants to move to a link $j_2 \in \mathcal{P}_{i_2}$, $i_2 > i_1$, that is, we have

$$\frac{\delta_{j_1}(\mathbf{L})}{c_{j_1}} > \frac{\delta_{j_2}(\mathbf{L}) + w}{c_{j_2}}\,. \tag{4.21}$$

We proceed by case analysis:
$\underline{i_1 = 0}$: In this case, $w = 2^{k-1}$, and we get

$$\begin{aligned} \frac{\delta_{j_1}(\mathbf{L})}{c_{j_1}} &\overset{(4.20)}{=} k \\ &\overset{(4.21)}{>} \frac{\delta_{j_2}(\mathbf{L}) + w}{c_{j_2}} \\ &\overset{(4.20)}{=} \frac{(k-i_2)\cdot c_{j_2} + 2^{k-1}}{c_{j_2}} \\ &= (k-i_2) + \frac{2^{k-1}}{2^{k-i_2}} \\ &= (k-i_2) + 2^{i_2-1} \\ &\overset{i_2 \geq 1}{\geq} (k-i_2) + i_2 \\ &= k\,, \end{aligned}$$

a contradiction.
$\underline{i_1 \geq 1}$: In this case, $w = 2^{k-(i_1+1)}$, and we get

$$\begin{aligned}
\frac{\delta_{j_1}(\mathbf{L})}{c_{j_1}} &\overset{(4.20),\ page\ 122}{=} (k-i_1) \\
&\overset{(4.21),\ page\ 122}{>} \frac{\delta_{j_2}(\mathbf{L})+w}{c_{j_2}} \\
&\overset{(4.20),\ page\ 122}{=} \frac{(k-i_2)\cdot c_{j_2}+2^{k-(i_1+1)}}{c_{j_2}} \\
&= (k-i_2)+\frac{2^{k-(i_1+1)}}{2^{k-i_2}} \\
&= (k-i_2)+2^{i_2-(i_1+1)} \\
&\overset{i_2-i_1\geq 1}{\geq} (k-i_2)+(i_2-i_1) \\
&= (k-i_1)\,,
\end{aligned}$$

a contradiction. Thus, the pure assignment $\mathbf{L}$ is a Nash equilibrium, as needed. ■

We proceed by proving the lower bound on k. We have

$$\begin{aligned}
m &= \sum_{0\leq i\leq k} |\mathcal{P}_i| \\
&= k+(k-1)\cdot \sum_{2\leq i\leq k} 2^{i-1}\cdot \prod_{j\in[i-1]} (k-j) \\
&= k+(k-1)\cdot 2^{k-1}\cdot (k-1)!\cdot \left(1+\sum_{2\leq i\leq k-1} \frac{1}{2^{k-i}}\cdot \frac{1}{(k-i)!}\right) \\
&\leq k+(k-1)\cdot 2^k\cdot (k-1)! \\
&\leq 2^k\cdot k! \\
&\leq \alpha\cdot k^k
\end{aligned}$$

for some constant $\alpha \in \mathbb{R}^+$. Let $r = \alpha\cdot k^k$. Since

$$\begin{aligned}
\log r &= k\cdot\log k+\log\alpha \\
&= k\cdot\log k\cdot(1+o(1))
\end{aligned}$$

and

$$\begin{aligned}
\log\log r &= \log k+\log\log k+o(1) \\
&= \log k\cdot(1+o(1))\,,
\end{aligned}$$

this implies

$$\begin{aligned}
\Gamma^{-1}(m) &\overset{(4.4),\ page\ 80}{\leq} \frac{\log r}{\log\log r}\cdot(1+o(1)) \\
&= k\cdot(1+o(1))\,,
\end{aligned}$$

as needed. ■

4.5.2 Mixed Nash Equilibria

We now state upper bounds on the price of anarchy and on the individual price of anarchy for mixed Nash equilibria on related links. The first upper bounds were proved by Koutsoupias and Papadimitriou [93]. For two links, they showed that the price of anarchy is $\frac{1+\sqrt{5}}{2}$, that is, the *golden ratio* (Theorem 4.58). It is interesting to note that this bound (with respect to *mixed* Nash equilibria) matches the upper bound for *jump optimal schedules* (with respect to *pure* jump optimal schedules) proved by Cho and Sahni [21] (see Table 4.3, page 93). For an arbitrary number of links, Czumaj and Vöcking [29] proved the asymptotically tight bound $\Theta(\frac{\log m}{\log\log\log m})$ by first bounding the maximum expected latency on the links, and then using this result to bound the expected maximum latency by applying a *Hoeffding inequality* (Theorem 4.59, page 125).

All these bounds depend either on the number of links or on the relation between the fastest and slowest link. We now introduce a new structural parameter ρ, defined as the ratio of the sum of link capacities of links to which the largest traffic can be assigned causing latency at most $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})$ and the sum of all link capacities. More formally, let

$$\mathcal{M}_1 \quad = \quad \{j \in [m] \mid w_1 \leq c_j \cdot \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})\} \; .$$

Then,

$$\rho \quad = \quad \frac{\sum_{j\in\mathcal{M}_1} c_j}{C} \; .$$

With help of ρ we are able to prove an upper bound $\Gamma^{-1}(\frac{1}{\rho})$ on the individual price of anarchy (Theorem 4.60, page 125). Clearly, $\frac{w_1}{c_1} \leq \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})$ and $C \leq m \cdot c_1$, implying that $\rho \geq \frac{1}{m}$. This allows us to bound the maximum expected latency by $\Gamma^{-1}(m) \cdot \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})$ improving on the best known upper bound $(\Gamma^{-1}(m)+1) \cdot \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})$ of Czumaj and Vöcking [29] (Corollary 4.62, page 131), and the individual price of anarchy by $\Gamma^{-1}(m)$ (Corollary 4.63, page 131). Moreover, since the individual price of anarchy and the price of anarchy coincide when restricting to pure Nash equilibria, the generalized bound directly leads to an improved bound $\Gamma^{-1}(\frac{1}{\rho})$ on the price of anarchy in this setting (Corollary 4.64, page 131). The generalized bound is tight up to an additive constant for *all m* (Proposition 4.65, page 131) whereas the upper bound $\Gamma^{-1}(m)$ is tight only for *large m* (see Theorem 4.56, page 121).

Feldmann *et al.* [46] showed that the individual price of anarchy is also bounded from above by $\frac{1+\sqrt{4m-3}}{2}$ (Theorem 4.66, page 132). For pure Nash equilibria, this matches the upper bound for *jump optimal schedules* of Cho and Sahni [21] (see Table 4.3, page 93). The bound is *not* asymptotically tight, but for small numbers of links ($m \leq 19$) better than the asymptotically tight bound $\Gamma^{-1}(m)$.

Theorem 4.58 (Koutsoupias and Papadimitriou [93]) *Consider the model of arbitrary users and two related links. Then,*

$$\mathsf{PoA} \quad = \quad \phi \, ,$$

where $\phi = \frac{1+\sqrt{5}}{2}$ *is the* **golden ratio**.

Theorem 4.59 (Czumaj and Vöcking [29]) *Consider the model of arbitrary users and related links. Then,*

$$\mathsf{PoA} \;=\; \Theta\left(\min\left\{\frac{\log m}{\log\log\log m}, \frac{\log m}{\log\left(\frac{\log m}{\log\left(\frac{c_1}{c_m}\right)}\right)}\right\}\right).$$

Theorem 4.60 *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w},\mathbf{c})$ *and associated Nash equilibrium* $\mathbf{P}$*, it is*

$$\frac{\mathsf{IC}_\infty(\mathbf{w},\mathbf{c},\mathbf{P})}{\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})} \;<\; \begin{cases} \frac{3}{2}+\sqrt{\frac{1}{\rho}-\frac{3}{4}} & \text{if } \frac{1}{3}\le\rho\le 1, \\ 2+\sqrt[3]{\frac{1}{\rho}-2} & \text{if } \frac{1}{37}\le\rho<\frac{1}{3}, \\ \Gamma^{-1}\left(\frac{1}{\rho}\right) & \text{if } \rho<\frac{1}{37}. \end{cases}$$

Proof: Without loss of generality, let $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})=1$ and $C=1$. For any integer $k\in\mathbb{N}$, consider an instance $(\mathbf{w},\mathbf{c})$ and associated mixed Nash equilibrium $\mathbf{P}$ with

$$k \;\le\; \mathsf{IC}_\infty(\mathbf{w},\mathbf{c},\mathbf{P}) \;<\; (k+1)\,.$$

In part (1.) and (2.), we give a lower bound on the total expected load that is necessary for such a mixed Nash equilibrium $\mathbf{P}$. In part (3.), we then use this lower bound to prove an upper bound on k. Denote $\tau_{ij}(\mathbf{P})$ the expected load on link j *excluding* the traffic of user i.

(1.) Let j_1 be the maximum index of a link in $\mathcal{M}_1$, that is, $\mathcal{M}_1=[j_1]$. Moreover, let $i_1\in[n]$ be a user and let $s_1\in\mathcal{M}_1$ be a link with $p_{i_1s_1}>0$ and

$$\begin{aligned}\lambda_{i_1s_1}(\mathbf{P}) &= \frac{\tau_{i_1s_1}(\mathbf{P})+w_{i_1}}{c_{s_1}} \\ &= \mathsf{IC}_\infty(\mathbf{w},m,\mathbf{P})\,.\end{aligned}$$

By the definition of Nash equilibrium, we have

$$\begin{aligned}k &\le \frac{\tau_{i_1s_1}(\mathbf{P})+w_{i_1}}{c_{s_1}} \\ &\le \frac{\tau_{i_1j}(\mathbf{P})+w_{i_1}}{c_j} \\ &\le \frac{\tau_{i_1j}(\mathbf{P})+w_1}{c_j} \qquad (4.22)\end{aligned}$$

for all $j\in\mathcal{M}_1$. Moreover, by definition of $\mathcal{M}_1$, we have

$$\begin{aligned}\frac{w_1}{c_j} &\le \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) \\ &= 1 \qquad (4.23)\end{aligned}$$

for all $j \in \mathcal{M}_1$. This implies that $w_1 \leq c_j$ for all $j \in \mathcal{M}_1$, and thus

$$\frac{\tau_{i_1 j}(\mathbf{P}) + c_j}{c_j} \overset{(4.23),\ page\ 125}{\geq} \frac{\tau_{i_1 j}(\mathbf{P}) + w_1}{c_j} \overset{(4.22),\ page\ 125}{\geq} k\,.$$

Therefore,

$$\begin{aligned} \delta_j(\mathbf{P}) &= \tau_{i_1 j}(\mathbf{P}) + p_{i_1 j} \cdot w_{i_1} \\ &\geq \tau_{i_1 j}(\mathbf{P}) \\ &\geq (k-1) \cdot c_j \end{aligned} \tag{4.24}$$

for all $j \in \mathcal{M}_1$. Let $C_1 = \sum_{j \in \mathcal{M}_1} c_j$. Summing up all expected loads $\delta_j(\mathbf{P})$ on links in $\mathcal{M}_1$, the total expected load $\Delta_1(\mathbf{P})$ of links in $\mathcal{M}_1$ is

$$\begin{aligned} \Delta_1(\mathbf{P}) &= \sum_{j \in \mathcal{M}_1} \delta_j(\mathbf{P}) \\ &\overset{(4.24)}{\geq} (k-1) \cdot C_1 \\ &\overset{definition\ of\ \rho}{=} \rho \cdot (k-1) \cdot C \\ &\overset{C=1}{=} \rho \cdot (k-1)\,. \end{aligned} \tag{4.25}$$

(2.) We show the following claim by induction on l:

Claim 4.61 *For all $l \in [k-1] \setminus \{1\}$, there exists a set $\mathcal{M}_l = [j_l] \setminus [j_{l-1}] \neq \emptyset$ such that*

(a.) the total capacity C_l of links in $\mathcal{M}_l$ is at least

$$C_l \geq \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j)\,,$$

(b.) for all j in $\mathcal{M}_l$, the expected load is bounded by

$$\delta_j(\mathbf{P}) \geq (k-l) \cdot c_j\,,$$

(c.) the total expected load $\Delta_l(\mathbf{P})$ on links in $\mathcal{M}_l$ is at least

$$\Delta_l(\mathbf{P}) \geq \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l} (k-j)\,,$$

(d.) and the difference between the total expected load on links in $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_l$ and the capacity of those links is bounded by

$$\sum_{j \in [l]} (\Delta_j(\mathbf{P}) - C_j) \geq \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l} (k-j)\,.$$

Proof: As our basis case, let $l = 2$. Let w_{i_2} be the smallest traffic of a user i_2 who chooses a link in $\mathcal{M}_1$ with positive probability, and let $s_2 \in \mathcal{M}_1$ be a link in $\mathcal{M}_1$ with $p_{i_2 s_2} > 0$. In an optimum assignment, at most load

$$\begin{aligned} C_1 &\overset{definition\ of\ \rho}{=} \rho \cdot C \\ &\overset{C=1}{=} \rho \end{aligned} \tag{4.26}$$

can be assigned to links in $\mathcal{M}_1$. Therefore, in an optimum assignment, the remaining expected load which is greater or equal to

$$\begin{aligned} \Delta_1(\mathbf{P}) - C_1 &\overset{(4.26)}{=} (\Delta_1(\mathbf{P}) - \rho) \\ &\overset{(4.25),\ page\ 126}{\geq} \rho \cdot (k-2) \end{aligned} \tag{4.27}$$

is assigned to links not in $\mathcal{M}_1$. This implies that there exists a set of links $\mathcal{M}_2 = [j_2] \setminus [j_1] \neq \emptyset$, j_2 minimum, with total capacity C_2 at least

$$\begin{aligned} C_2 &\geq \Delta_1(\mathbf{P}) - C_1 \\ &\overset{(4.27)}{\geq} \rho \cdot (k-2)\ , \end{aligned} \tag{4.28}$$

proving (a.). Moreover, by definition of Nash equilibrium, we have

$$\begin{aligned} (k-1) &\overset{(4.24),\ page\ 126}{\leq} \frac{\delta_{s_2}(\mathbf{P})}{c_{s_2}} \\ &\leq \frac{\tau_{i_2 s_2}(\mathbf{P}) + w_{i_2}}{c_{s_2}} \\ &\leq \frac{\tau_{i_2 j}(\mathbf{P}) + w_{i_2}}{c_j} \end{aligned} \tag{4.29}$$

for all $j \in \mathcal{M}_2$. Since all links in $\mathcal{M}_1$ have expected latency larger than $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = 1$, there exists a link $j \in [m] \setminus [j_2 - 1]$ to which a user with traffic at least w_{i_2} is assigned in an optimum assignment. We get

$$\begin{aligned} \frac{w_{i_2}}{c_j} &\leq \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \\ &= 1 \end{aligned} \tag{4.30}$$

for all $j \in \mathcal{M}_2$. This implies that $w_{i_2} \leq c_j$ for all $j \in \mathcal{M}_2$, and thus

$$\begin{aligned} \frac{\tau_{i_2 j}(\mathbf{P}) + c_j}{c_j} &\overset{(4.30)}{\geq} \frac{\tau_{i_2 j}(\mathbf{P}) + w_{i_2}}{c_j} \\ &\overset{(4.29)}{\geq} (k-1)\ . \end{aligned}$$

Therefore,

$$\begin{aligned} \delta_j(\mathbf{P}) &= \tau_{i_2 j}(\mathbf{P}) + p_{i_2 j} \cdot w_{i_2} \\ &\geq \tau_{i_2 j}(\mathbf{P}) \\ &\geq (k-2) \cdot c_j \end{aligned}$$

for all $j \in \mathcal{M}_2$, proving (b.). Summing up all expected loads $\delta_j(\mathbf{P})$ on links in $\mathcal{M}_2$, the total expected load $\Delta_2(\mathbf{P})$ of links in $\mathcal{M}_2$ is

$$\begin{aligned} \Delta_2(\mathbf{P}) &= \sum_{j \in \mathcal{M}_2} \delta_j(\mathbf{P}) \\ &\geq (k-2) \cdot C_2 \\ &\overset{(4.28),\ page\ 127}{\geq} \rho \cdot (k-2)^2 , \end{aligned} \tag{4.31}$$

proving (c.). In an optimum assignment, at most expected load $C_1 + C_2$ can be assigned to links in $\mathcal{M}_1 \cup \mathcal{M}_2$. So, the remaining expected load on links in $\mathcal{M}_1 \cup \mathcal{M}_2$ which has to be assigned to other links in an optimum assignment is at least

$$\begin{aligned} &\sum_{j \in [2]} (\Delta_j(\mathbf{P}) - C_j) \\ &= \Delta_1(\mathbf{P}) + \Delta_2(\mathbf{P}) - C_1 - C_2 \\ &\overset{(4.25),\ page\ 126,\ (4.31)}{\geq} (k-1) \cdot C_1 + (k-2) \cdot C_2 - C_1 - C_2 \\ &= (k-2) \cdot C_1 + (k-3) \cdot C_2 \\ &\overset{definition\ of\ \rho,\ (4.28),\ page\ 127}{\geq} \rho \cdot (k-2) + \rho \cdot (k-3) \cdot (k-2) \\ &= \rho \cdot (k-2)^2 , \end{aligned}$$

proving (d.), and thus the claim holds for the basis case.

For the induction step, let $l \geq 3$, and assume that Claim 4.61 (page 126) holds for $(l-1)$. Let w_{i_l} be the smallest traffic of a user i_l who assigns its traffic to a link in $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{l-1}$ with positive probability, and let $s_l \in \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{l-1}$ be a link with $p_{i_l s_l} > 0$. By induction hypothesis,

$$\sum_{j \in [l-1]} (\Delta_j(\mathbf{P}) - C_j) \quad \geq \quad \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j) .$$

This implies that there exists a set of links $\mathcal{M}_l = [j_l] \setminus [j_{l-1}] \neq \emptyset$, j_l minimum, with total capacity at least

$$\begin{aligned} C_l &\geq \sum_{j \in [l-1]} (\Delta_j(\mathbf{P}) - C_j) \\ &\geq \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j) , \end{aligned} \tag{4.32}$$

proving (a.). This holds since it is possible to assign all users to links with latency at most $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = 1$. Moreover, by definition of Nash equilibrium, we have

$$\begin{aligned} k - (l-1) &\leq \frac{\delta_{s_l}(\mathbf{P})}{c_{s_l}} \\ &\leq \frac{\tau_{i_l s_l}(\mathbf{P}) + w_{i_l}}{c_{s_l}} \\ &\leq \frac{\tau_{i_l j}(\mathbf{P}) + w_{i_l}}{c_j} \end{aligned} \tag{4.33}$$

for all $j \in \mathcal{M}_l$. Since all links in $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{l-1}$ have expected latency larger than $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) = 1$, there exists a link $j \in [m] \setminus [j_l - 1]$ to which a user with traffic at least w_{i_l} is assigned in an optimum assignment. We get

$$\begin{aligned} \frac{w_{i_l}}{c_j} &\leq \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) \\ &= 1 \end{aligned} \tag{4.34}$$

for all $j \in \mathcal{M}_l$. This implies that $w_{i_l} \leq c_j$ for all $j \in \mathcal{M}_l$, and thus

$$\frac{\tau_{i_l j}(\mathbf{P}) + c_j}{c_j} \overset{(4.34)}{\geq} \frac{\tau_{i_l j}(\mathbf{P}) + w_{i_l}}{c_j} \overset{(4.33),\ page\ 128}{\geq} k - (l-1)\,.$$

Therefore,

$$\begin{aligned} \delta_j(\mathbf{P}) &= \tau_{i_l j}(\mathbf{P}) + p_{i_l j} \cdot w_{i_l} \\ &\geq \tau_{i_l j}(\mathbf{P}) \\ &\geq (k-l) \cdot c_j \end{aligned}$$

for all $j \in \mathcal{M}_l$, proving (b.). Summing up all expected loads $\delta_j(\mathbf{P})$ on links in $\mathcal{M}_l$, the total expected load $\Delta_l(\mathbf{P})$ of links in $\mathcal{M}_l$ is

$$\begin{aligned} \Delta_l(\mathbf{P}) &= \sum_{j \in \mathcal{M}_l} \delta_j(\mathbf{P}) \\ &\geq (k-l) \cdot C_l \\ &\overset{(4.32),\ page\ 128}{\geq} \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l} (k-j)\,, \end{aligned} \tag{4.35}$$

proving (c.). In an optimum assignment, at most load $\sum_{j \in [l]} C_j$ can be assigned to links in $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_l$. So, the remaining expected load on links in $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_l$ which has to be assigned to other links in the optimum solution is at least

$$\begin{aligned} &\sum_{j \in [l]} (\Delta_j(\mathbf{P}) - C_j) \\ &= \sum_{j \in [l-1]} (\Delta_j(\mathbf{P}) - C_j) + \Delta_l(\mathbf{P}) - C_l \\ &\overset{(4.35)}{\geq} \sum_{j \in [l-1]} (\Delta_j(\mathbf{P}) - C_j) + (k-l) \cdot C_l - C_l \\ &\overset{Induction,\ (4.32),\ page\ 128}{\geq} \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j) \\ &\qquad + (k-l-1) \cdot \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j) \\ &= (k-l) \cdot \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l-1} (k-j) \\ &= \rho \cdot (k-2) \cdot \prod_{2 \leq j \leq l} (k-j)\,, \end{aligned}$$

proving (d.). This completes the proof of the inductive claim. ■

(3.) We proceed by showing an upper bound on k. Summing up over all $\Delta_l(\mathbf{P})$ we get the lower bound $\sum_{l\in[k-1]}\Delta_l(\mathbf{P}) < W$ on the total load W. The strict inequality follows from the fact that there exists at least one user with expected individual cost at least k. Using this lower bound, we now prove the upper bounds for the three cases of the claim by showing that a larger upper bound implies

$$\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) \;\geq\; \frac{W}{C} \;>\; 1 \;=\; \mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})\,,$$

a contradiction. Note that $\Gamma^{-1}(\frac{1}{\rho})$ is also an upper bound on the ratio for $\rho \geq \frac{1}{37}$. However, in the ranges $\frac{1}{3} \leq \rho \leq 1$ and $\frac{1}{37} \leq \rho < \frac{1}{3}$ the given upper bounds are better. Now, consider the three cases of the claim:

(a.) $\underline{\frac{1}{3} \leq \rho \leq 1}$: Assume $k = \frac{3}{2} + \sqrt{\frac{1}{\rho} - \frac{3}{4}}$. This implies $k \geq 2$ in the given range of ρ. Then,

$$\begin{aligned}
W &> \Delta_1(\mathbf{P}) + \Delta_2(\mathbf{P}) \\
&\geq \rho\cdot\left((k-1)+(k-2)^2\right) \\
&= \rho\cdot\left(k^2-3\cdot k+3\right) \\
&= \rho\cdot\left(\left(\frac{3}{2}+\sqrt{\frac{1}{\rho}-\frac{3}{4}}\right)^2 - 3\cdot\left(\frac{3}{2}+\sqrt{\frac{1}{\rho}-\frac{3}{4}}\right)+3\right) \\
&= \rho\cdot\left(\frac{9}{4}+3\cdot\sqrt{\frac{1}{\rho}-\frac{3}{4}}+\frac{1}{\rho}-\frac{3}{4}-\frac{9}{2}-3\cdot\sqrt{\frac{1}{\rho}-\frac{3}{4}}+3\right) \\
&= \rho\cdot\left(\frac{1}{\rho}\right) \\
&= 1 \\
&= C\,.
\end{aligned}$$

(b.) $\underline{\frac{1}{37} \leq \rho < \frac{1}{3}}$: Assume $k = 2 + \sqrt[3]{\frac{1}{\rho} - 2}$. This implies $k > 3$ in the given range of ρ. Then,

$$\begin{aligned}
W &> \Delta_1(\mathbf{P}) + \Delta_2(\mathbf{P}) + \Delta_3(\mathbf{P}) \\
&\geq \rho\cdot\left((k-1)+(k-2)^2+(k-2)^2(k-3)\right) \\
&= \rho\cdot\left(k-1+(k-2)^3\right) \\
&\overset{k>3}{>} \rho\cdot\left(2+(k-2)^3\right) \\
&= \rho\cdot\left(2+\frac{1}{\rho}-2\right) \\
&= 1 \\
&= C\,.
\end{aligned}$$

(c.) $\rho < \frac{1}{37}$: Assume $k = \Gamma^{-1}(\frac{1}{\rho})$. Using the facts that $\Gamma(x+1) = x \cdot \Gamma(x)$ for all real numbers $x > 0$ and $\Gamma(x) \leq x$ for all $1 \leq x \leq 3$, we get

$$\begin{aligned}
W &> \sum_{l \in [k-1]} \Delta_l(\mathbf{P}) \\
&> \Delta_{k-2}(\mathbf{P}) + \Delta_{k-1}(\mathbf{P}) \\
&\geq \rho \cdot (k-2) \cdot \left(\prod_{2 \leq j \leq k-2} (k-j) + \prod_{2 \leq j \leq k-1} (k-j) \right) \\
&> \rho \cdot (k-2) \cdot \left(\prod_{3 \leq j \leq k-1} (k-j) + \prod_{2 \leq j \leq k-1} (k-j) \right) \\
&\geq \rho \cdot (k-2) \cdot (\Gamma(k-2) + \Gamma(k-1)) \\
&= \rho \cdot (k-2) \cdot (\Gamma(k-2) + (k-2) \cdot \Gamma(k-2)) \\
&= \rho \cdot (k-2) \cdot ((k-1) \cdot \Gamma(k-2)) \\
&= \rho \cdot \Gamma(k) \\
&= \rho \cdot \Gamma\left(\Gamma^{-1}\left(\frac{1}{\rho} \right) \right) \\
&= 1 \\
&= C \, .
\end{aligned}$$

In each of the cases, we have $W > C$, and this lower bound on W also holds for any real number $x > k$. This is a contradiction to $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = 1$. ■

Corollary 4.62 *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w}, \mathbf{c})$ *and associated Nash equilibrium* $\mathbf{P}$*, it is*

$$\Lambda(\mathbf{P}) \leq \Gamma^{-1}(m) \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \, .$$

Corollary 4.63 *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w}, \mathbf{c})$ *and associated Nash equilibrium* $\mathbf{P}$*, it is*

$$\frac{\mathsf{IC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{P})}{\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})} \leq \Gamma^{-1}(m) \, .$$

Corollary 4.64 *Consider the model of arbitrary users and related links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \leq \Gamma^{-1}\left(\frac{1}{\rho} \right) \, .$$

Proposition 4.65 *Consider the model of arbitrary users and related links. Then, for every* $k \in \mathbb{N}$ *there exists an instance* $(\mathbf{w}, \mathbf{c})$ *and associated pure Nash equilibrium* $\mathbf{L}$ *with*

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

Proof: Consider the instance $(\mathbf{w},\mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$ from Example 4.55 (page 121). As seen in Claim 4.57 (page 121), it is $\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c}) = 1$, and $\mathbf{L}$ is a pure Nash equilibrium with $\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{L}) = k$. We now prove that $\Gamma^{-1}(\frac{1}{\rho}) - 3$ is a lower bound on k. By definition,

$$\rho \;=\; \frac{|\mathcal{P}_0 \cup \mathcal{P}_1| \cdot 2^{k-1}}{|\mathcal{P}_0| \cdot 2^{k-1} + \sum_{i\in[k]} |\mathcal{P}_i| \cdot 2^{k-i}} \,.$$

This implies

$$\begin{aligned}
\frac{1}{\rho} &= \frac{1}{k\cdot 2^{k-1}} \cdot \left(|\mathcal{P}_0| \cdot 2^{k-1} + \sum_{i\in[k]} |\mathcal{P}_i| \cdot 2^{k-i} \right) \\
&\overset{definition\ of\ \mathcal{P}_i}{=} \frac{1}{k\cdot 2^{k-1}} \cdot \left(k\cdot 2^{k-1} + \sum_{2\le i\le k} \left(2^{i-1}\cdot(k-1) \prod_{j\in[i-1]} (k-j) \right) \cdot 2^{k-i} \right) \\
&\le \frac{1}{k\cdot 2^{k-1}} \cdot \left(k\cdot 2^{k-1} + 2^{k-1}\cdot k \sum_{2\le i\le k} \prod_{j\in[i-1]} (k-j) \right) \\
&= 1 + \sum_{2\le i\le k} \prod_{j\in[i-1]} (k-j) \\
&= 1 + 2(k-1)! + \sum_{2\le i\le k-2} \prod_{j\in[i-1]} (k-j) \\
&\le 2(k-1)! + \sum_{2\le i\le k-2} \frac{(k-1)!}{2^{i-1}} \\
&\le 3(k-1)! \\
&\overset{k\ge 1}{\le} (k+2)! \\
&= \Gamma(k+3) \,.
\end{aligned}$$

This yields $k \ge \Gamma^{-1}(\frac{1}{\rho}) - 3$, as needed. ∎

Theorem 4.66 (Feldmann *et al.* [46]) *Consider the model of arbitrary users and related links. Then, for any instance $(\mathbf{w},\mathbf{c})$ and associated Nash equilibrium $\mathbf{P}$, it is*

$$\frac{\mathsf{IC}_\infty(\mathbf{w},\mathbf{c},\mathbf{P})}{\mathsf{OPT}_\infty(\mathbf{w},\mathbf{c})} \;\le\; \frac{1+\sqrt{4m-3}}{2} \,.$$

This bound is tight if and only if $m \le 5$. For pure Nash equilibria, the bound is tight if and only if $m \le 3$.

4.5.3 Fully Mixed Nash Equilibria

Mavronicolas and Spirakis [107] showed that, in contrast to the model of identical links, there does not necessarily exist a fully mixed Nash equilibrium. In fact, there exist instances without a fully mixed Nash equilibrium (Example 4.41, page 113). Furthermore, they proved that if a fully mixed Nash equilibrium exists, then it is unique and can be computed efficiently (Theorem 4.67, page 133). In case of its existence, we can compare the fully mixed

Nash equilibrium to a worst Nash equilibrium. The following results provide evidence for the FMNE Conjecture. In particular, the FMNE Conjecture holds for pure Nash equilibria (Theorem 4.68). Moreover, for the model of identical users, the conjecture holds up to a constant factor (Theorem 4.69) for generalized fully mixed Nash equilibria, which, in contrast to fully mixed Nash equilibria, always exist. For a thorough analysis of fully mixed Nash equilibria for the model of identical users and links with non-decreasing, non-constant latency functions, we refer to [58]. Finally, the conjecture is valid in case of two identical users (Theorem 4.70). In contrast to the yet unproved claim of the FMNE Conjecture, each user indeed experiences the worst expected individual cost in the fully mixed Nash equilibrium (Proposition 4.71).

Theorem 4.67 (Mavronicolas and Spirakis [107]) *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w},\mathbf{c})$*, there exists a fully mixed Nash equilibrium* $\mathbf{F}$ *if and only if*

$$\left(1-\frac{mc_j}{C}\right)\cdot\left(1-\frac{W}{(n-1)w_i}\right)+\frac{c_j}{C} \quad \in \quad (0,1)$$

for all $i\in[n]$ *and* $j\in[m]$*. If* $\mathbf{F}$ *exists, then* $\mathbf{F}$ *is unique and*

$$f_{ij} \quad = \quad \left(1-\frac{mc_j}{C}\right)\cdot\left(1-\frac{W}{(n-1)w_i}\right)+\frac{c_j}{C}$$

for all users $i\in[n]$ *and links* $j\in[m]$*.*

Example 4.41 (continued) *According to the formula in Theorem 4.67, the probability of user* 4 *on link* 4 *is*

$$f_{44} \quad = \quad \left(1-\frac{4\cdot 2}{14}\right)\left(1-\frac{14}{4\cdot 2}\right)+\frac{2}{14} \quad = \quad -\frac{5}{28} \quad \notin \quad (0,1)\,.$$

By Theorem 4.67, this implies that there exists no fully mixed Nash equilibrium.

Theorem 4.68 (Gairing *et al.* [59]) *Consider the model of arbitrary users and related links, restricted to pure Nash equilibria. Then, the* FMNE *Conjecture is valid.*

Theorem 4.69 (Fotakis *et al.* [50]) *Consider the model of identical users and related links. Then, for any instance* $(\mathbf{w},\mathbf{c})$*, associated Nash equilibrium* $\mathbf{P}$ *and generalized fully mixed Nash equilibrium* $\mathbf{F}$*, it is*

$$\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{P}) \quad \leq \quad 49.02\cdot\mathsf{SC}_\infty(\mathbf{w},\mathbf{c},\mathbf{F})\,.$$

Theorem 4.70 (Lücking *et al.* [103]) *Consider the model of two identical users and related links. Then, the* FMNE *Conjecture is valid.*

Proposition 4.71 (Gairing *et al.* [59]) *Consider the model of arbitrary users and related links. Then, for any instance* $(\mathbf{w},\mathbf{c})$ *and associated Nash equilibrium* $\mathbf{P}$*, it is* $\lambda_i(\mathbf{P})\leq\lambda_i(\mathbf{F})$*.*

4.6 Makespan Social Cost and Restricted Strategy Sets

We now consider the KP-model with makespan social cost and restricted strategy sets. Recall that every user $i \in [n]$ is only allowed to assign its traffic to links in its strategy set $R_i \subseteq [m]$. Clearly, this restriction can change the set of possible Nash equilibria. Subsection 4.6.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.6.2 hold for general (i.e. mixed) Nash equilibria.

4.6.1 Pure Nash Equilibria

4.6.1.1 Computation of Nash Equilibria

Identical Users. Since the model of identical users and identical links is a special case of the unrelated links model where the traffics are either 1 or ∞, we can compute a (best) pure Nash equilibrium by solving a bipartite cardinality matching problem (Theorem 4.72). For the model of identical users and related links, Even-Dar *et al.* [42] improved an upper bound of Milchtaich [109], showing that sequences of (not necessarily greedy) selfish steps can be used to compute a pure Nash equilibrium (Theorem 4.73).

Theorem 4.72 (Hopcroft and Karp [72], Lenstra *et al.* [97]) *Consider the model of identical users with restricted strategy sets and identical links. Then, for any instance (n,m), a best pure Nash equilibrium can be computed in $O(R\sqrt{n})$ time.*

Theorem 4.73 (Even-Dar *et al.* [42]) *Consider the model of identical users with restricted strategy sets and related links. Then, there exists an instance, an associated pure assignment and a rule such that the maximum length of a sequence of (not necessarily greedy) selfish steps is at most nm before reaching a Nash equilibrium using $O(nm\log m)$ time.*

Arbitrary Users. For the model of arbitrary users and related links, no polynomial-time algorithm to compute a pure Nash equilibrium is known. However, restricting to identical links, a nashification algorithm, in the sequel called NASHIFY-RESTRICTED, was given by Gairing *et al.* [56] by identifying some natural connections between the problem of computing a Nash equilibrium and *network flow* problems (see e.g. [1]). In the remainder of this subsection, we assume that all user traffics are positive integers. In order to present the algorithm, we first show how to represent a (partial) pure assignment via a residual network. We then introduce a blocking flow algorithm, called UNSPLITTABLE-BLOCKING-FLOW, and we show how it can be used by algorithm RECURSIVEUBF to alter a pure assignment such that all users with traffic w_1 are satisfied and the social cost does not increase. Finally, we show how NASHIFY-RESTRICTED uses RECURSIVEUBF to nashify any given pure assignment. We illustrate both the introduced definitions and algorithms with help of the following example.

Example 4.74 *Consider the following instance $(\mathbf{w},\mathbf{c})$: We have $n = 5$ users $i_1,\ldots,i_5$ and $m = 4$ identical links $j_1,\ldots,j_4$. We have $w_{i_1} = 6$ with $R_{i_1} = \{j_1,j_2\}$, $w_{i_2} = 5$ with $R_{i_2} = \{j_2,j_3\}$, $w_{i_3} = 4$ with $R_{i_3} = \{j_1,j_2\}$, $w_{i_4} = 2$ with $R_{i_4} = \{j_3,j_4\}$, and $w_{i_5} = 2$ with $R_{i_5} = \{j_3,j_4\}$. Moreover, we consider the pure assignment $\mathbf{L} = \langle j_1,j_2,j_1,j_3,j_3\rangle$ (illustrated in Figure 4.8).*

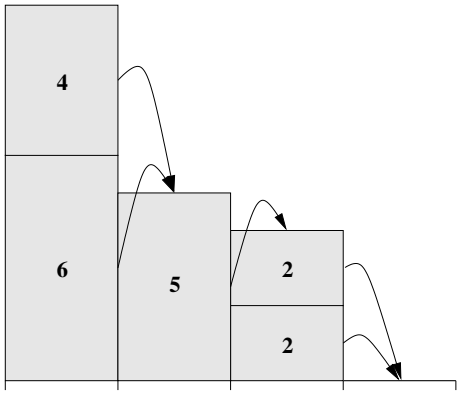


Figure 4.8: Assignment $\mathbf{L} = \langle j_1, j_2, j_1, j_3, j_3 \rangle$ of the instance given in Example 4.74 (page 134). Each small arrow points from an assigned user to the other link in its strategy set.

Residual Network Representation. In the following, we present a (partial) pure assignment with help of a residual network.

Definition 4.75 *Given a (partial) pure assignment $\mathbf{L} = \langle \ell_1, \ldots, \ell_n \rangle$, we define a directed bipartite graph $G_{\mathbf{L}} = (V, E_{\mathbf{L}})$, where $V = M \cup U$ such that each link is represented by a node in M and each user is represented by a node in U. Furthermore, $E_{\mathbf{L}} = E^1_{\mathbf{L}} \cup E^2_{\mathbf{L}}$ with*

$$\begin{aligned} E^1_{\mathbf{L}} &= \{(j,i) \mid j \in M, i \in U, j = \ell_i\} \text{ , and} \\ E^2_{\mathbf{L}} &= \{(i,j) \mid i \in U, j \in M, j \in R_i \setminus \{\ell_i\}\} \text{ .} \end{aligned}$$

Example 4.74 (continued) *The residual network $G_{\mathbf{L}}$ for the pure assignment $\mathbf{L} = \langle j_1, j_2, j_1, j_3, j_3 \rangle$ is illustrated in Figure 4.9.*

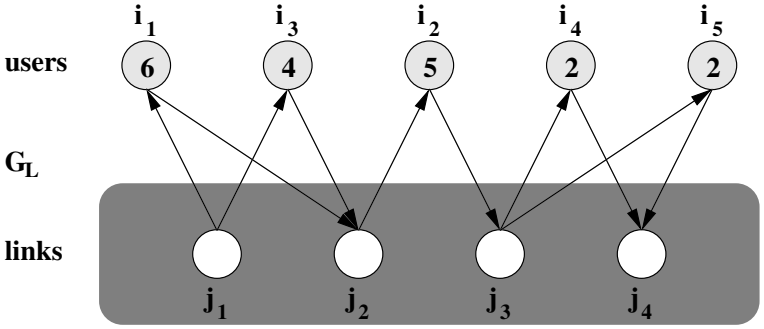


Figure 4.9: The residual network $G_{\mathbf{L}}$ for the pure assignment $\mathbf{L} = \langle j_1, j_2, j_1, j_3, j_3 \rangle$ given in Example 4.74 (page 134).

For a total pure assignment $\mathbf{L}$, we use the graph $G_{\mathbf{L}}$ from Definition 4.75 to define a graph $G_{\mathbf{L}}(w)$ where V stays the same, but from $E_{\mathbf{L}}$ we now only consider edges

$$E_{\mathbf{L}}(w) = E_{\mathbf{L}} \setminus \{(i,j) \mid i \in U, j \in M, w_i > w\} \text{ .}$$

This means that all users $i \in [n]$ with $w_i > w$ stay assigned to their link. We use $G_{\mathbf{L}}$ instead of $G_{\mathbf{L}}(w)$ if it is clear from the context which w is used.

UNSPLITTABLE-BLOCKING-FLOW. We now introduce an algorithm, called UNSPLITTABLE-BLOCKING-FLOW. Starting with any integer $w \in \mathbb{N}$ and any pure assignment $\mathbf{L}$, we use an integer a to control the approximation of an optimum assignment. The intention is to find an a which is a lower bound on $\mathsf{OPT}_\infty(\mathbf{w}, m)$, and then to compute a pure assignment $\mathbf{L}'$ with $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \leq a + w$. For any integer a, we partition the set of links M into three subsets:

$$\begin{aligned} M^- &= \{j \in M \mid \delta_j(\mathbf{L}) \leq a\} \\ M^0 &= \{j \in M \mid a+1 \leq \delta_j(\mathbf{L}) \leq a+w\} \\ M^+ &= \{j \in M \mid \delta_j(\mathbf{L}) \geq a+w+1\} \end{aligned}$$

In this setting, we do not have a dedicated source or sink. However, at each time nodes in M^+ and M^- can be interpreted as source and sink nodes, respectively. Note, that those sets change over time.

Example 4.74 (continued) *Let $w = 5$ and $a = 3$. Then, the partition of the links for the given pure assignment $\mathbf{L}$ looks as follows: $M^+ = \{j_1\}$, $M^0 = \{j_2, j_3\}$, and $M^- = \{j_4\}$.*

UNSPLITTABLE-BLOCKING-FLOW combines ideas from blocking flows with the idea of pushing users without splitting them. Roughly speaking, algorithm UNSPLITTABLE-BLOCKING-FLOW shifts users so that the latencies of links from M^- are never decreased, the latencies of links from M^+ are never increased, and links from M^0 remain in M^0. The algorithm is controlled by a **height function** $h : V \to \mathbb{N}_0$ with $h(j) = \mathrm{dist}_{G_\mathbf{L}(w)}(j, M^-)$ for all $j \in V$. We call an edge (u,v) **admissible** if $h(u) = h(v) + 1$. In an **admissible path**, all edges are admissible. For each node $u \in V$ with $0 < h(u) < \infty$, define $S(u)$ to be the **set of successors** of node u; this is the set of nodes to which u has an admissible edge, so that

$$S(u) = \{v \in V \mid (u,v) \in E_\mathbf{L} \text{ and } h(u) = h(v) + 1\}.$$

Note that $S(u)$ also defines the set of admissible edges leaving u. Let $s(u)$ be the first node in a list implementation of the set $S(u)$. We proceed to define:

Definition 4.76 *A link $j \in M$ with $0 < h(j) < \infty$ is called* **helpful** *if $\delta_j(\mathbf{L}) \geq a + 1 + w_{s(j)}$.*

Lemma 4.77 (Gairing *et al.* [56]) *Let v_0 be a helpful link of minimum height. Then, there exists a sequence $v_0, \ldots, v_r$, where $v_{2i} \in M$ for all $0 \leq i \leq r/2$ and $v_{2i+1} \in U$ for all $0 \leq i < r/2$ such that*

(1.) $(v_i, v_{i+1}) \in E_\mathbf{L}$ *and* $h(v_i) = h(v_{i+1}) + 1$,
(2.) $\delta_{v_0}(\mathbf{L}) \geq a + 1 + w_{s(v_0)}$,
(3.) $a + 1 \leq \delta_{v_{2i}}(\mathbf{L}) + w_{s(v_{2i-2})} - w_{s(v_{2i})} \leq a + w$ *for all* $0 < i < r/2$,
(4.) $\delta_{v_r}(\mathbf{L}) + w_{s(v_{r-2})} \leq a + w$.

We are now ready to present the algorithm UNSPLITTABLE-BLOCKING-FLOW, stated as Algorithm 6 (page 137). Initially, the height function h is computed as the distance in $G_\mathbf{L}(w)$ of each node to the set M^- of nodes. Then, the algorithm proceeds in phases. In each phase, first the minimum height $d = h(v)$ of a node $v \in M^+$ is computed. Inside each phase, we do not update the height function, but we successively choose a helpful link v of minimum

height and we push users along the helpful path induced by v and adjust the pure assignment accordingly. In order to update $G_{\mathbf{L}}(w)$, we have to change the direction of two arcs for each user push. The phase ends when there exists no further admissible path from a node $v \in M^+$ with $h(v) = d$ to some node in M^-. Before the new phase starts, we recompute h and we check whether we need to start a new phase or not. UNSPLITTABLE-BLOCKING-FLOW stops when either $M^- = \emptyset$ or for all $v \in M^+$ we have $h(v) = \infty$.

Algorithm 6 (UNSPLITTABLE-BLOCKING-FLOW)

Input: a pure assignment $\mathbf{L}$ and positive integers a, w
Output: a pure assignment $\mathbf{L}'$

(1) **begin**
(2) compute h;
(3) $\mathbf{L}' \leftarrow \mathbf{L}$;
(4) **while** $M^- \neq \emptyset$ **and** there exists a $v \in M^+$ with $h(v) < \infty$ **do**
(5) $d \leftarrow \min_{v \in M^+}(h(v))$;
(6) **while** there exists an admissible path from $v \in M^+, h(v) = d$, to M^- **do**
(7) choose helpful link v of minimum height;
(8) push users along helpful path defined by v;
(9) update $\mathbf{L}', G_{\mathbf{L}'}(w)$;
(10) recompute h;
(11) **return** $\mathbf{L}'$;
(12) **end**

Gairing *et al.* [56] showed that UNSPLITTABLE-BLOCKING-FLOW decreases the maximum load and increases the minimum load on the links, respectively (Lemma 4.78). Moreover, they showed properties of the resulting pure assignment $\mathbf{L}'$ (Lemma 4.79), and that UNSPLITTABLE-BLOCKING-FLOW can be implemented to run in $O(mR)$ time (Theorem 4.80, page 138).

Lemma 4.78 (Gairing *et al.* [56]) *For the pure assignment $\mathbf{L}'$ computed by* UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}, a, w)$*, we have*

$$\begin{aligned} \max_{j \in [m]} \delta_j(\mathbf{L}') &\leq \max_{j \in [m]} \delta_j(\mathbf{L}) \,, \quad \textit{and} \\ \min_{j \in [m]} \delta_j(\mathbf{L}') &\geq \min_{j \in [m]} \delta_j(\mathbf{L}) \,. \end{aligned}$$

Lemma 4.79 (Gairing *et al.* [56]) *For the pure assignment $\mathbf{L}'$ computed by* UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}, a, w)$*, one of the following conditions holds:*

(1.) $M^-(\mathbf{L}') = \emptyset$.
(2.) $M^+(\mathbf{L}') = \emptyset$.
(3.) There exists some set of links $B \subset [m]$ such that

(a.) $\delta_j(\mathbf{L}') \geq a + 1$ *for all* $j \in B$*, and*
(b.) $\delta_j(\mathbf{L}') \leq a + w$ *for all* $j \in [m] \setminus B$*, and*

(c.) $\ell_i \in B$ *implies* $R_i \subseteq B$ *for all* $i \in [n]$ *with* $w_i \leq w$.

Theorem 4.80 (Gairing *et al.* [56]) UNSPLITTABLE-BLOCKING-FLOW *can be implemented to run in* $O(mR)$ *time.*

Example 4.74 (continued) *On the left hand side of Figure 4.10, the height function in* $G_{\mathbf{L}}(w)$ *for* $w = 5$ *and* $a = 3$ *is illustrated with help of a layered network. Only link* j_1 *is helpful, and there exist two admissible paths:* $j_1, i_3, j_2, i_2, j_3, i_4, j_4$ *and* $j_1, i_3, j_2, i_2, j_3, i_5, j_4$. *On the right hand side, the result of pushing users along the first admissible path is shown. Here, link* j_3 *becomes helpful.*

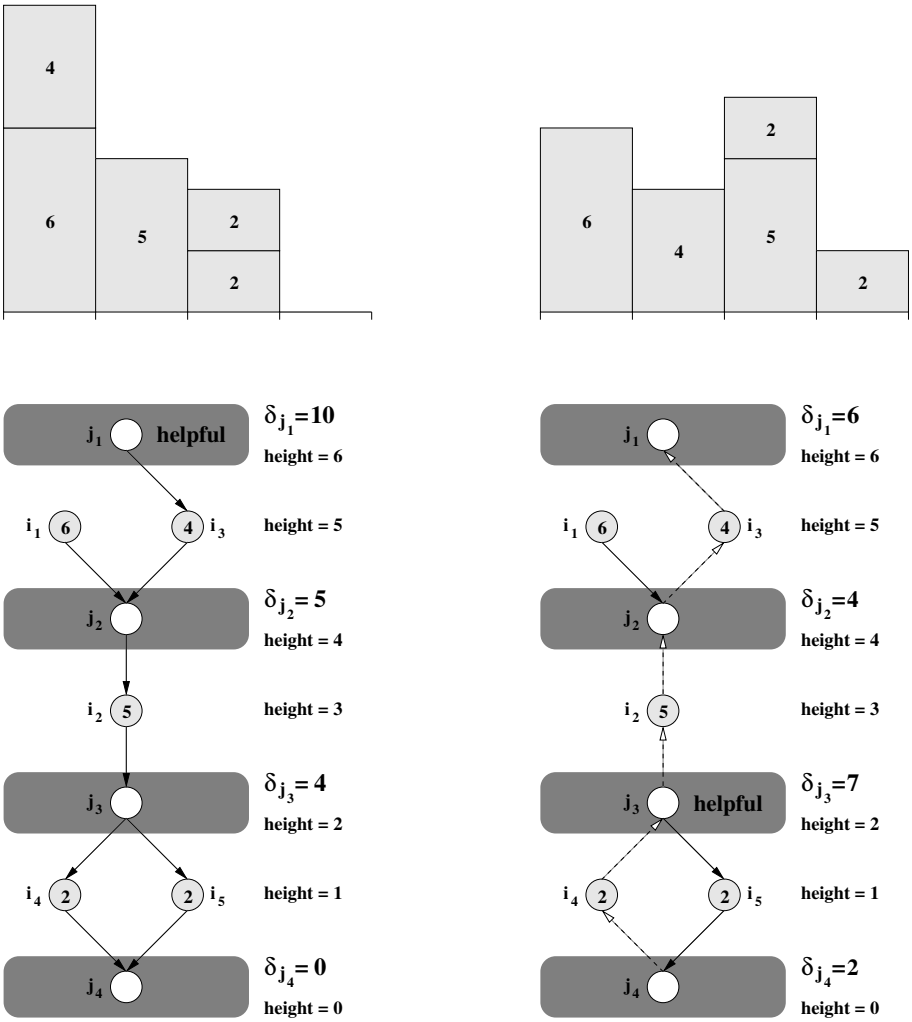


Figure 4.10: Height function in $G_{\mathbf{L}}(w)$ for $w = 5$ and $a = 3$ of Example 4.74 (page 134) illustrated with help of a layered network before (left hand side) and after pushing users along the admissible path $j_1, i_3, j_2, i_2, j_3, i_4, j_4$ (right hand side).

RECURSIVEUBF. We next show how UNSPLITTABLE-BLOCKING-FLOW is used by algorithm RECURSIVEUBF$(B, \mathbf{L}(B), [l,u], w)$, stated as Algorithm 7 (page 139).

If $l \leq \delta_j(\mathbf{L}(B)) \leq u + w$ for all links $j \in B$ prior to a call to RECURSIVEUBF$(B,[l,u],w)$, then it computes a pure assignment where no user with traffic at least w that is assigned to some link in B can improve by moving to some other link in B. By a series of calls to UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}'(B),a,w)$ we compute a pure assignment where M^- and M^+ are either both empty or both non-empty. Parameter a is chosen by binary search $a \in [l,u]$, $a \in \mathbb{N}$, as follows: If UNSPLITTABLE-BLOCKING-FLOW returns a pure assignment with $M^- = \emptyset$ and $M^+ \neq \emptyset$, then we increase a. On the other hand, if UNSPLITTABLE-BLOCKING-FLOW returns a pure assignment with $M^- \neq \emptyset$ and $M^+ = \emptyset$, then we decrease a.

Algorithm 7 (RECURSIVEUBF)

Input: a set of links B, a pure assignment $\mathbf{L}(B)$, an interval $[l,u]$ and a traffic size w
Output: a pure assignment $\mathbf{L}'(B)$

(1) **begin**
(2) $a \leftarrow \lceil (l+u)/2 \rceil$;
(3) **if** $a = u$ **then**
(4) **return** $\mathbf{L}(B)$;
(5) $\mathbf{L}(B) \leftarrow$ UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}(B),a,w)$;
(6) **if** $M^-(\mathbf{L}(B)) = \emptyset$ **and** $M^+(\mathbf{L}(B)) \neq \emptyset$ **then**
(7) $\mathbf{L}'(B) \leftarrow$ RECURSIVEUBF$(B,[a,u],w)$;
(8) **else if** $M^-(\mathbf{L}(B)) \neq \emptyset$ **and** $M^+(\mathbf{L}(B)) = \emptyset$ **then**
(9) $\mathbf{L}'(B) \leftarrow$ RECURSIVEUBF$(B,[l,a],w)$;
(10) **else if** $M^-(\mathbf{L}(B)) \neq \emptyset$ **and** $M^+(\mathbf{L}(B)) \neq \emptyset$ **then**
(11) split B (according to Lemma 4.79 (3.), page 137) into sets B' and $\overline{B'}$;
(12) $\mathbf{L}'(B') \leftarrow$ RECURSIVEUBF$(B',[a,u],w)$;
(13) $\mathbf{L}'(\overline{B'}) \leftarrow$ RECURSIVEUBF$(\overline{B'},[l,a],w)$;
(14) $\mathbf{L}'(B) \leftarrow \mathbf{L}'(B') \cup \mathbf{L}'(\overline{B'})$;
(15) **return** $\mathbf{L}'(B)$;
(16) **end**

If after the binary search, $M^- = \emptyset$ and $M^+ = \emptyset$, then we have computed a pure assignment where all users with traffic at least w are satisfied. If neither $M^- = \emptyset$ nor $M^+ = \emptyset$ it follows that condition (3.) from Lemma 4.79 (page 137) holds. Define B' as the set of links still reachable from M^+ and let $\overline{B'}$ be the complement of B' in B. In this case, we split our instance into two parts. One part with all links in B' and all users that are currently assigned to a link in B', the other part holds the complement. Whenever B is split into B' and $\overline{B'}$, condition (3.) from Lemma 4.79 (page 137) implies that no user v with $w_v \leq w$, assigned to a link in B', has a link from $\overline{B'}$ in its strategy set.

We recursively proceed with the binary search on a in both parts of the instance. For the part that corresponds to B', we increase a, while in the other part we decrease a. The recursive splitting of B defines a partition of the links into sets $B_1,\ldots,B_p$. At the end, all parts $B_1,\ldots,B_p$ are put together to form $\mathbf{L}'(B)$.

For each $B_k, k \in [p]$, define a lower bound $\mathsf{Low}(B_k)$ on the load of all links from B_k as the last value for a after the binary search on a in B_k. This implies:

Lemma 4.81 (Gairing *et al.* [56]) *If $l \leq \delta_j(\mathbf{L}(B)) \leq u + w$ for all $j \in B$ prior to a call to* RECURSIVEUBF*$(B,\mathbf{L}(B),[l,u],w)$, then* RECURSIVEUBF*$(B,\mathbf{L}(B),[l,u],w)$ returns a pure*

assignment $\mathbf{L}'(B)$ of users in B, a partition of B into p sets $B_1,\ldots,B_p$ for some p, and (implicitly) numbers $\mathsf{Low}(B_k)$ for $k\in[p]$ such that

(1.) $u \geq \mathsf{Low}(B_1) > \ldots > \mathsf{Low}(B_p) \geq l$ *for all* $k\in[p]$,

(2.) $\mathsf{Low}(B_k) \leq \delta_j(\mathbf{L}'(B)) \leq \mathsf{Low}(B_k)+w$ *for all* $j\in B_k$ *and for all* $k\in[p]$,

(3.) no user v with $w_v \leq w$ that is assigned to a link in B_k has a link from B_ℓ in its strategy set R_v if $\ell > k$.

By (3.) all users with traffic w are satisfied in the pure assignment computed by RECURSIVEUBF. In order to keep these users satisfied, we have to ensure that in further computations the lower bounds only increase and the upper bounds only decrease. We denote the upper bound by $\mathsf{Up}(B_k)$ for all links from B_k, and in coincidence with (2.) we set $\mathsf{Up}(B_k) = \mathsf{Low}(B_k)+w$.

NASHIFY-RESTRICTED. We are now ready to present the algorithm NASHIFY-RESTRICTED. Let $\widetilde{w}_1 > \ldots > \widetilde{w}_r$ be all different user traffics from $w_1,\ldots,w_n$. The idea is to compute a sequence of pure assignments $\mathbf{L}_0,\ldots,\mathbf{L}_r$ such that $\mathbf{L}_0=\mathbf{L}$, and such that for all pure assignments $\mathbf{L}_i$ with $i\in[r]$, all users j with $w_j \geq \widetilde{w}_i$ are satisfied. We call the computation of $\mathbf{L}_i$ from $\mathbf{L}_{i-1}$ **stage** i. The aim in stage i is to compute a pure assignment $\mathbf{L}_i$ from $\mathbf{L}_{i-1}$ such that in $\mathbf{L}_i$ all users u with $w_u \geq \widetilde{w}_i$ are satisfied.

Algorithm 8 (NASHIFY-RESTRICTED)
Input: a pure assignment $\mathbf{L}_0$
Output: a pure assignment $\mathbf{L}_r$

```
(1)  begin
     // stage 1
(2)  L_1 ← RecursiveUBF([m], L_0, [0, max_j δ_j(L_0)], w̃_1);
     // stage 2,...,r
(3)  for i ← 2 to r do
(4)      while there are sets of active links do
(5)          execute Sweep over the active links;
         // L_i is the current pure assignment
(6)  return L_r;
(7)  end
```

Algorithm 8 shows the high-level structure of NASHIFY-RESTRICTED. We first use the procedure RECURSIVEUBF to compute a pure assignment $\mathbf{L}_1$ where all users with traffic $\widetilde{w}_1$ are satisfied. Afterwards, we iteratively satisfy users with traffic $\widetilde{w}_2,\ldots,\widetilde{w}_r$ making sure that users with larger traffic remain satisfied. We do this by executing SWEEP over the sets of active links. In the following, we define what we mean by sets of active links, and we describe how a SWEEP over these sets of active links is executed.

Lemma 4.81 (page 139) implies that after stage 1, all users with traffic $\widetilde{w}_1$ are satisfied. Furthermore, the links are partitioned into p_1 sets $B_1,\ldots,B_{p_1}$ with $\mathsf{Up}(B_k) = \mathsf{Low}(B_k)+\widetilde{w}_1$

for all $k \in [p_1]$, and no user v with $w_v \leq \widetilde{w}_1$, that is assigned to a link from B_k can be assigned to a link from B_ℓ if $k < \ell$.

We now describe stage $i > 1$. The lower bound on the load of a link only increases and the upper bound only decreases. This implies that fixed users remain satisfied. At the beginning of stage i, we have a pure assignment $\mathbf{L}_{i-1}$, where the links are partitioned into p_{i-1} sets $B_1, \ldots, B_{p_{i-1}}$ with $\mathsf{Up}(B_k) = \mathsf{Low}(B_k) + \widetilde{w}_{i-1}$ for all $k \in [p_{i-1}]$, and no user v that is assigned to a link from B_k can be assigned to a link from B_ℓ if $k < \ell$.

During each stage i, we always maintain a pure assignment $\mathbf{L}'_i$ where the links are partitioned into q sets $C_1, \ldots, C_q$ for some q. They are ordered such that $\mathsf{Up}(C_k) > \mathsf{Up}(C_{k+1})$ and $\mathsf{Low}(C_k) \geq \mathsf{Low}(C_{k+1})$ for all $k \in [q-1]$.

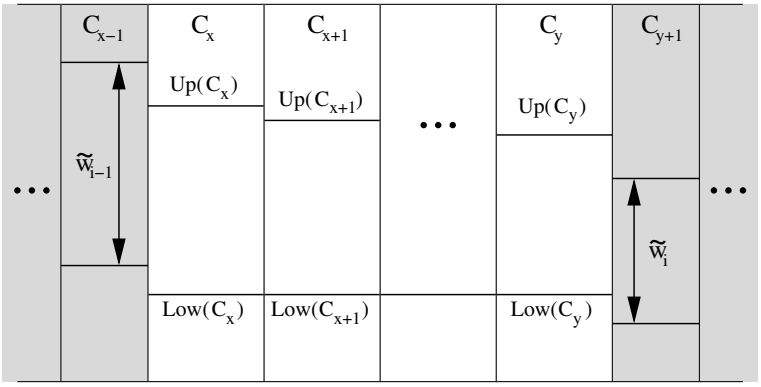


Figure 4.11: Sets of active links in stage i at the beginning of a sweep

At the beginning of a SWEEP, we have three classes of sets (see Figure 4.11):

- Some sets of links C_k, $k \in [x-1]$, have not been considered yet and fulfill $\mathsf{Up}(C_k) - \mathsf{Low}(C_k) = \widetilde{w}_{i-1}$.
- Moreover, some sets of links C_k, $y < k \leq q$, have been **done in stage** i already and fulfill $\mathsf{Up}(C_k) - \mathsf{Low}(C_k) = \widetilde{w}_i$.
- Finally, we have sets $C_x, \ldots, C_y$ of **active** links with $\widetilde{w}_i < \mathsf{Up}(C_k) - \mathsf{Low}(C_k) \leq \widetilde{w}_{i-1}$ and $\mathsf{Low}(C_k) = \mathsf{Low}(C_y)$ for all $x \leq k \leq y$.

Initially, $C_j = B_j$ for all $j \in [p_{i-1}]$, the links from $C_{p_{i-1}}$ are active, and the remaining links have not been considered. During a SWEEP, the number of partitions q may change. We will see in Lemma 4.84 (page 144) that at the beginning of each SWEEP, the **sweep property** introduced below holds.

Definition 4.82 (Sweep Property during stage i)

(1.) There is a partition of the links into q sets $C_1, \ldots, C_q$ for some q with $\mathsf{Low}(C_1) \geq \ldots \geq \mathsf{Low}(C_q)$ and $\mathsf{Up}(C_1) > \ldots > \mathsf{Up}(C_q)$.

(2.) If link $j \in C_k$, then $\mathsf{Low}(C_k) \leq \delta_j(\mathbf{L}'_i) \leq \mathsf{Up}(C_k)$.

(3.) No user v with $w_v \leq \widetilde{w}_i$ that is assigned to a link in C_k has a link from C_ℓ in its strategy set R_v if $\ell > k$.

(4.) There exist integers x,y with $1 \leq x \leq y \leq q$ and

(a.) $\mathsf{Up}(C_k) - \mathsf{Low}(C_k) = \widetilde{w}_{i-1}$ for $k \in [x-1]$,

(b.) $\mathsf{Up}(C_k) - \mathsf{Low}(C_k) = \widetilde{w}_i$ for $y < k \leq q$, and

(c.) $\widetilde{w}_i < \mathsf{Up}(C_k) - \mathsf{Low}(C_k) \leq \widetilde{w}_{i-1}$ and $\mathsf{Low}(C_k) = \mathsf{Low}(C_y)$ for all $x \leq k \leq y$.

We now use the definition of Sweep Property to define active links and links which are done in stage i more formally.

Definition 4.83 *Let x,y be as in Definition 4.82 (page 141). Then, a link $j \in C_k$, $x \leq k \leq y$, is called* **active***, and a link $j \in C_k$, $y < k \leq q$, is called* **done in stage** *i.*

SWEEP is stated as Algorithm 9 (page 143) and works on active links as follows: At the beginning of SWEEP, the sweep property holds. The aim of SWEEP is to process links in C_y such that they do not have to be considered again in this stage, or to make all links in C_{x-1} active by increasing the lower bound of all active links to $\mathsf{Low}(C_{x-1})$. In order to preserve the structure of our pure assignment, we choose $a = \min\{\mathsf{Up}(C_y) - \widetilde{w}_i, \mathsf{Low}(C_{x-1})\}$. We insert all sets into a list $\mathcal{L}$ such that $\mathcal{L} = [C_x, \ldots, C_y]$. Then, as long as there are at least two sets in $\mathcal{L}$, we do the following: We extract the first element, say D_1, of $\mathcal{L}$ and apply UNSPLITTABLE-BLOCKING-FLOW to the sub-instance defined by the set D_1. UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}(D_1), a, \widetilde{w}_i)$ returns a pure assignment $\mathbf{L}'$ where one of the following conditions hold:

(1.) $M^+(\mathbf{L}') = \emptyset$: In this case, all links in D_1 have load at most $a + \widetilde{w}_i$, and Lemma 4.78 (page 137) implies that this property is preserved. Let D_2 be the next element in $\mathcal{L}$. Before the call, $\mathsf{Up}(D_1) > \mathsf{Up}(D_2) > a + \widetilde{w}_i$ was true. After the call, the loads of all links in D_1 are bounded from above by $a + \widetilde{w}_i$. So, by setting $\mathsf{Up}(D_1) \leftarrow \mathsf{Up}(D_2)$, we get a new upper bound on the loads of the links in D_1, and we fulfill the requirement that upper bounds can be only decreased. D_1 and D_2 are merged, and the union of both sets is inserted into $\mathcal{L}$. This way, the number of sets in the list is decreased by 1.

(2.) $M^-(\mathbf{L}') = \emptyset$ and $M^+(\mathbf{L}') \neq \emptyset$: In this case, all links in D_1 have load at least a, and Lemma 4.78 (page 137) implies that this property is preserved. Thus, we are allowed to set $\mathsf{Low}(C_j) \leftarrow a$. We are done with D_1 during this execution of SWEEP.

(3.) $M^-(\mathbf{L}') \neq \emptyset$ and $M^+(\mathbf{L}') \neq \emptyset$: In this case, we split D_1 according to condition (3.) from Lemma 4.79 (page 137) into sets D_1' and $\overline{D_1'}$. Condition (3c.) implies that no user that is assigned to a link in D_1' can be assigned to a link in $\overline{D_1'}$. Since the load on each link in D_1' is at least a, we can set $\mathsf{Low}(D_1') \leftarrow a$. The load of each link in $\overline{D_1'}$ is at most $a + \widetilde{w}_i$. Thus, since the upper bound of the next element, say D_2, in $\mathcal{L}$ is $\mathsf{Up}(D_2) > a + \widetilde{w}_i$, we again can extract D_2 from $\mathcal{L}$, set $\mathsf{Up}(\overline{D_1'}) \leftarrow \mathsf{Up}(D_2)$, merge $\overline{D_1'}$ and D_2, and insert it in $\mathcal{L}$. We are done with D_1' during this execution of SWEEP.

Algorithm 9 (SWEEP)

Input: a list $\mathcal{L} = [C_x, \ldots, C_y]$ of the sets of active links

(1) **begin**
(2) $a \leftarrow \min\{\mathsf{Up}(C_y) - \widetilde{w}_i, \mathsf{Low}(C_{x-1})\}$;
(3) **while** $|\mathcal{L}| \geq 2$ **do**
(4) $D_1 \leftarrow \mathsf{ExtractFirst}(\mathcal{L})$;
(5) $\mathbf{L}' \leftarrow$ UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}(D_1), a, \widetilde{w}_i)$;
(6) **if** $M^+(\mathbf{L}') = \emptyset$ **then**
(7) $D_2 \leftarrow \mathsf{ExtractFirst}(\mathcal{L})$;
(8) $\mathsf{Up}(D_1) \leftarrow \mathsf{Up}(D_2)$;
(9) $D_1 \leftarrow D_1 \cup D_2$;
(10) $\mathsf{Insert}(D_1, \mathcal{L})$;
(11) **else if** $M^-(\mathbf{L}') = \emptyset$ **and** $M^+(\mathbf{L}') \neq \emptyset$ **then**
(12) $\mathsf{Low}(D_1) \leftarrow a$ and output: "links in D_1 are *done in this sweep*";
(13) **else if** $M^-(\mathbf{L}') \neq \emptyset$ **and** $M^+(\mathbf{L}') \neq \emptyset$ **then**
(14) split D_1 (according to Lemma 4.79 (3.), page 137) into sets D_1' and $\overline{D_1'}$;
(15) $\mathsf{Low}(D_1') \leftarrow a$ and output: "links in D_1' are *done in this sweep*";
(16) $D_2 \leftarrow \mathsf{ExtractFirst}(\mathcal{L})$;
(17) $\mathsf{Up}(\overline{D_1'}) \leftarrow \mathsf{Up}(D_2)$;
(18) $D_1 \leftarrow \overline{D_1'} \cup D_2$;
(19) $\mathsf{Insert}(D_1, \mathcal{L})$;
// Different handling of last set
(20) $D_1 \leftarrow \mathsf{ExtractFirst}(\mathcal{L})$;
(21) **if** $a = \mathsf{Up}(D_1) - \widetilde{w}_i$ **then**
(22) RECURSIVEUBF$(D_1, \mathbf{L}(D_1)[\mathsf{Low}(D_1), a], \widetilde{w}_i)$ and output: "links in D_1 are *done in stage i*";
(23) **else**
(24) $\mathbf{L}' \leftarrow$ UNSPLITTABLE-BLOCKING-FLOW$(\mathbf{L}(D_1), a, \widetilde{w}_i)$;
(25) **if** $M^-(\mathbf{L}') = \emptyset$ **then**
(26) $\mathsf{Low}(D_1) \leftarrow a$ and output: "links in D_1 are *done in this sweep*";
(27) **else if** $M^-(\mathbf{L}') \neq \emptyset$ **and** $M^+(\mathbf{L}') = \emptyset$ **then**
(28) $\mathsf{Up}(D_1) \leftarrow a + \widetilde{w}_i$;
(29) RECURSIVEUBF$(D_1, \mathbf{L}'(D_1), [\mathsf{Low}(D_1), a], \widetilde{w}_i)$ and output: "links in D_1 are *done in stage i*";
(30) **else if** $M^-(\mathbf{L}') \neq \emptyset$ **and** $M^+(\mathbf{L}') \neq \emptyset$ **then**
(31) split D_1 (according to Lemma 4.79 (3.), page 137) into sets D_1' and $\overline{D_1'}$;
(32) $\mathsf{Low}(D_1') \leftarrow a$ and output: "links in D_1' are *done in this sweep*";
(33) $\mathsf{Up}(\overline{D_1'}) \leftarrow a + \widetilde{w}_i$;
(34) RECURSIVEUBF$(\overline{D_1'}, \mathbf{L}'(\overline{D_1'})[\mathsf{Low}(\overline{D_1'}), a], \widetilde{w}_i)$ and output: "links in $\overline{D_1'}$ are *done in stage i*";
(35) **end**

So, in each case, the number of sets in list $\mathcal{L}$ is decreased by 1. Now, we consider the case that there is only one set, say D_1, in $\mathcal{L}$. This case has to be handled differently.

If $a = \mathsf{Up}(D_1) - \widetilde{w}_i$, then we simply apply RECURSIVEUBF to the sub-instance defined by D_1 in the interval $[\mathsf{Low}(D_1), a]$ with traffic size $\widetilde{w}_i$. Otherwise, we apply UNSPLITTABLE-BLOCKING-FLOW to the sub-instance defined by the set D_1. UNSPLITTABLE-BLOCKING-FLOW($\mathbf{L}(D_1), a, \widetilde{w}_i$) returns a pure assignment $\mathbf{L}'$ where one of the following conditions holds.

(1.) $M^-(\mathbf{L}') = \emptyset$: Here, we set $\mathsf{Low}(D_1) \leftarrow a$.

(2.) $M^-(\mathbf{L}') \neq \emptyset$ and $M^+(\mathbf{L}') = \emptyset$: In this case, we set $\mathsf{Up}(D_1) \leftarrow a + \widetilde{w}_i$ and apply RECURSIVEUBF to the sub-instance defined by D_1 in the interval $[\mathsf{Low}(D_1), a]$ with traffic size $\widetilde{w}_i$.

(3.) $M^-(\mathbf{L}') \neq \emptyset$ and $M^+(\mathbf{L}') \neq \emptyset$: Here, we split D_1 according to condition (3.) from Lemma 4.79 (page 137) into sets D'_1 and $\overline{D'_1}$. For D'_1, we set $\mathsf{Low}(D'_1) \leftarrow a$, and for $\overline{D'_1}$ we set $\mathsf{Up}(\overline{D'_1}) \leftarrow a + \widetilde{w}_i$ and we apply RECURSIVEUBF to the sub-instance defined by $\overline{D'_1}$ in the interval $[\mathsf{Low}(\overline{D'_1}), a]$ with traffic size $\widetilde{w}_i$.

After each sweep, by renumbering the partitions, we get a new pure assignment that again has the same structure as in Definition 4.82 (page 141). This completes the description of SWEEP. Gairing *et al.* [56] proved:

Lemma 4.84 (Gairing *et al.* [56]) *The sweep property holds at the beginning of each execution of* SWEEP. *Moreover, in each execution, either a non-empty set of links is added to the set of active links, or some non-empty set of links is stage-finalized.*

Lemma 4.85 (Gairing *et al.* [56]) *After stage i, every user v with traffic $w_v \geq \widetilde{w}_i$ is satisfied.*

Theorem 4.86 (Gairing *et al.* [56]) *Consider the model of arbitrary users with restricted strategy sets and identical links. Then, for any instance $(\mathbf{w}, m)$ and associated pure assignment $\mathbf{L}$,* NASHIFY-RESTRICTED($\mathbf{L}$) *computes a pure Nash equilibrium $\mathbf{L}'$ from $\mathbf{L}$ with $\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \leq \mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})$ using $O(rmR(\log W + m^2))$ time, where r is the number of distinct traffic sizes.*

4.6.1.2 Computation of Best Nash Equilibria

If both users and links are identical, then a best pure Nash equilibrium can be computed in polynomial time (see Theorem 4.72, page 134). For arbitrary users and identical links, Theorem 4.22 (page 104) implies that BEST PURE NE is $\mathcal{NP}$-complete. Since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense [50], there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m even for related links (Theorem 4.87). Moreover, in case of identical links, NASHIFY-RESTRICTED enables us to approximate an optimum Nash equilibrium within factor $2 - \frac{1}{w_1}$ (Theorem 4.88, page 145). Lenstra *et al.* [97] showed that MULTIPROCESSOR SCHEDULING is not approximable within a factor $\frac{3}{2} - \varepsilon$ for any ε with $0 < \varepsilon \leq \frac{1}{2}$ unless $\mathcal{P} = \mathcal{NP}$. The same result can be proved for BEST PURE NE in the model of arbitrary users with restricted strategy sets and identical links by adapting the proof of Lenstra *et al.* [97] to this setting (Theorem 4.89, page 145).

Theorem 4.87 *Consider the model of arbitrary users with restricted strategy sets and related links. Then, there exists a pseudo-polynomial-time algorithm for m-*BEST PURE NE.

Proof: See Theorem 4.104 (page 161) for a generalization of this result. ■

Theorem 4.88 (Gairing *et al.* [56]) *Consider the model of arbitrary users with restricted strategy sets and identical links. Then, for any instance* $(\mathbf{w},m)$ *a pure Nash equilibrium* $\mathbf{L}$ *with* $\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) \leq (2-\frac{1}{w_1}) \cdot \mathsf{OPT}_\infty(\mathbf{w},m)$ *can be computed in polynomial time.*

Theorem 4.89 *Consider the model of arbitrary users with restricted strategy sets and identical links. If, for any* ε *with* $0 < \varepsilon \leq \frac{1}{2}$, BEST PURE NE *is* $(\frac{3}{2}-\varepsilon)$*-approximable, then* $\mathcal{P} = \mathcal{NP}$.

Proof: We prove this result by reduction from 3-DIMENSIONAL MATCHING, that is, we employ a polynomial time transformation from 3-DIMENSIONAL MATCHING to BEST PURE NE. For any ε with $0 < \varepsilon \leq \frac{1}{6}$, we construct an instance of BEST PURE NE such that if we had a polynomial-time $(\frac{3}{2}-\varepsilon)$-approximation algorithm for BEST PURE NE, then we could decide whether an instance of 3-DIMENSIONAL MATCHING is positive in polynomial time. From this construction the theorem then follows.

Consider an arbitrary instance of 3-DIMENSIONAL MATCHING. We call the triples that contain x_i **triples of type** i. Let τ_i be the number of triples of type i. From this instance we construct an instance for the stated problem as follows:

- There are $m = |\mathcal{T}|$ links. Each link $j \in [m]$ corresponds to a triple in $t_j \in T$.
- There are $n = m + q$ users.
  - There are $2q$ **element users** with traffic 1 that correspond to the $2q$ elements of $Y \cup Z$ in the natural way. On link $j \in [m]$, corresponding to the triple $t_j = (x_r, y_s, z_t) \in \mathcal{T}$, the users corresponding to y_s and z_t can be processed.
  - There are $\tau_i - 1$ **dummy users** of type i with traffic 2 for all $i \in [q]$ (if $m < q$, then we construct some trivial *no* instance of BEST PURE NE). Note that the total number of dummy jobs is $m - q$.

Clearly, this is polynomial time transformation. We prove that this is a transformation from 3-DIMENSIONAL MATCHING to BEST PURE NE.

(1.) The instance of 3-DIMENSIONAL MATCHING is positive:
Consider a matching, that is, a subset $\mathcal{T}' \subseteq \mathcal{T}$ with $|\mathcal{T}'| = q$ such that no two elements in $\mathcal{T}'$ agree in any coordinate. Say that $\mathcal{T}' = \{t_1, \ldots, t_q\}$. Use this matching to define a pure assignment $\mathbf{L}$ as follows:

- For the triple $t_j = (x_r, y_s, z_t) \in \mathcal{T}'$, $j \in [q]$, the element users corresponding to y_s and z_t are assigned to link j.
- Each of the dummy users is solo on one of the links in $[m] \setminus [q]$.

Clearly, $\delta_j(\mathbf{L}) = 2$ for all $j \in [m]$. So, $\mathbf{L}$ is a Nash equilibrium. Moreover,

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}) \;=\; 2\,.$$

Now, consider any pure Nash equilibrium $\mathbf{L}'$ in which the element users and dummy users are not assigned to the links $[m]$ according to a matching. Thus, the $2q$ element users are assigned to more than q links. This implies that there exists at least one link $j \in [m]$ with $\delta_j(\mathbf{L}') \geq 3$. Hence,

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}') \;\geq\; 3\,,$$

and we get

$$\begin{aligned} \frac{\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}')}{\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})} &\geq \frac{3}{2} \\ &\overset{\varepsilon>0}{>} \frac{3}{2} - \varepsilon\,. \end{aligned}$$

Thus, no such Nash equilibrium $\mathbf{L}'$ approximates the best pure Nash equilibrium within the claimed factor.

(2.) The instance of 3-DIMENSIONAL MATCHING is negative:
There exists no matching, that is, a subset $\mathcal{T}' \subseteq \mathcal{T}$ such that $|\mathcal{T}'| = q$ and no elements in $\mathcal{T}'$ agree in any coordinate. As seen in the previous case, the element users and the dummy users can not be assigned to the links $[m]$ such that they cause load 2 on all of these links, showing that $\mathsf{OPT}_\infty(\mathbf{w}, m) \geq 3$.

Consider an arbitrary pure Nash equilibrium $\mathbf{L}$. Assume, by way of contradiction, that there exists a link $j \in [m]$ with $\delta_j(\mathbf{L}) \geq 4$. Clearly, at least one dummy user i is assigned to link j. Since the total load of all users is $2m$, there exists a link $\ell \in [m]$ with $\delta_\ell(\mathbf{L}) < 2$. We get

$$\begin{aligned} \delta_j(\mathbf{L}) &\geq 4 \\ &= 2 + w_i \\ &> \delta_\ell(\mathbf{L}) + w_i\,. \end{aligned}$$

This shows that user i is unsatisfied, contradicting the fact that $\mathbf{L}$ is a Nash equilibrium. Thus, $\delta_j(\mathbf{L}) \leq 3$ for all $j \in [m]$, and we get

$$\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L}) \;\leq\; 3\,.$$

Hence,

$$\begin{aligned} \frac{\mathsf{SC}_\infty(\mathbf{w}, m, \mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w}, m)} &\leq 1 \\ &\overset{\varepsilon\leq\frac{1}{2}}{\leq} \frac{3}{2} - \varepsilon\,. \end{aligned}$$

Thus, all Nash equilibria approximate the best pure Nash equilibrium within the claimed factor.

Therefore, if we had a polynomial-time $(\frac{3}{2} - \varepsilon)$-approximation algorithm for BEST PURE NE, we could use it to decide whether an instance of 3-DIMENSIONAL MATCHING is positive in the following way: We apply the approximation algorithm to the corresponding instance of BEST PURE NE and we answer *yes* if and only if it returns a solution with makespan 2. ■

4.6.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

Identical Links. For arbitrary users, Theorem 4.26 (page 106) implies that WORST PURE NE is $\mathcal{NP}$-complete. Since WORST PURE NE is $\mathcal{NP}$-complete in the strong sense [50], there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.90).

In contrast to the case of unrestricted strategy sets, the price of anarchy for pure Nash equilibria on identical links is not bounded from above by a constant. In particular, using similar techniques as in [29], we can prove an upper bound of $\Gamma^{-1}(m)$ for arbitrary users (Theorem 4.92, page 148). This upper bound is already tight up to an additive constant for identical users (Theorem 4.91). Awerbuch *et al.* [6] independently proved a more general bound (Theorem 4.94, page 152). Clearly, Theorem 4.29 (page 108) implies that, for any ε with $0 < \varepsilon \leq 1 - \frac{2}{m+1}$, we can not hope to find a polynomial-time $(2 - \frac{2}{m+1} - \varepsilon)$-approximation algorithm for WORST PURE NE. Up to now, no other result is known.

Theorem 4.90 *Consider the model of arbitrary users with restricted strategy sets and identical links. Then, there exists a pseudo-polynomial-time algorithm for m-*WORST PURE NE.

Proof: See Theorem 4.105 (page 161) for a generalization of this result. ■

Theorem 4.91 *Consider the model of identical users with restricted strategy sets and identical links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \;>\; \Gamma^{-1}(m) - 2 \;=\; \Omega\left(\frac{\log m}{\log\log m}\right).$$

Proof: Fix any $k \in \mathbb{N}$, and consider the following instance (n, m) and associated pure assignment $\mathbf{L}$:

- We have $k+1$ disjoint subsets $\mathcal{M}_0, \ldots, \mathcal{M}_k$ of links with $|\mathcal{M}_0| = 1$ and

  $$|\mathcal{M}_i| \;=\; (k-1) \prod_{j \in [i-1]} (k-j)$$

  for all $i \in [k] \setminus \{1\}$.

- We have k disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{k-1}$ of users. $\mathcal{U}_0$ contains k users with strategy set $\mathcal{M}_0 \cup \mathcal{M}_1$, and $\mathcal{U}_i$ contains $(k-i) \cdot |\mathcal{M}_i|$ users with strategy set $\mathcal{M}_i \cup \mathcal{M}_{i+1}$ for all $i \in [k-1]$.

- The pure assignment $\mathbf{L}$ is defined as follows: To link $\mathcal{M}_0$ we assign all k users in $\mathcal{U}_0$; to each link in $\mathcal{M}_i$ we assign $(k-i)$ users in $\mathcal{U}_i$ for all $i \in [k-1] \setminus \{1\}$; the links in $\mathcal{M}_k$ remain empty.

Clearly, all links in $\mathcal{M}_i$, $i \in [k] \cup \{0\}$, have latency $(k-i)$, and all users assigned to links in $\mathcal{M}_i$ are only allowed to choose links in $\mathcal{M}_i \cup \mathcal{M}_{i+1}$. Since the latencies on links in $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ differ by 1, all users are satisfied, showing that $\mathbf{L}$ is a Nash equilibrium. The social cost of $\mathbf{L}$

is $\mathsf{SC}_\infty(n,m,\mathbf{L}) = k$. Note that $k = |\mathcal{U}_0| = |\mathcal{M}_0| + |\mathcal{M}_1|$ users are assigned to the link in $\mathcal{M}_0$, and that for each $i \in [k-1]$, exactly

$$\begin{aligned}(k-i)\cdot|\mathcal{M}_i| &= (k-1)\prod_{j\in[i]}(k-j)\\ &= |\mathcal{M}_{i+1}|\end{aligned}$$

users are assigned to links in $\mathcal{M}_i$. Thus, we can assign each user in $\mathcal{U}_0$ to a link in $\mathcal{M}_0 \cup \mathcal{M}_1$, and each user in $\mathcal{U}_i$, $i \in [k-1]$, to a link in $\mathcal{M}_{i+1}$ such that all users are solo. So, $\mathsf{OPT}_\infty(n,m) = 1$, and we get

$$\frac{\mathsf{SC}_\infty(n,m,\mathbf{L})}{\mathsf{OPT}_\infty(n,m)} = k\,.$$

We proceed by proving the lower bound on k. Since $|\mathcal{M}_0| \le |\mathcal{M}_1| \le \ldots \le |\mathcal{M}_k|$, it follows that

$$\begin{aligned}m &\le (k+1)\cdot|\mathcal{M}_k|\\ &= (k+1)\cdot(k-1)\cdot(k-1)!\\ &< (k+1)!\\ &= \Gamma(k+2)\,.\end{aligned}$$

Thus, $k > \Gamma^{-1}(m) - 2$, and we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} &> \Gamma^{-1}(m) - 2\\ &\overset{(4.4),\ page\ 80}{=} \Omega\left(\frac{\log m}{\log\log m}\right),\end{aligned}$$

as needed. ■

Theorem 4.92 *Consider the model of arbitrary users with restricted strategy sets and identical links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \le \Gamma^{-1}(m) = O\left(\frac{\log m}{\log\log m}\right).$$

Proof: For any integer $k \in \mathbb{N}$, consider an instance $(\mathbf{w},m)$ and associated pure Nash equilibrium $\mathbf{L}$ with

$$k\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \le \mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) < (k+1)\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\,.$$

We give a lower bound on the number of links that are necessary for such a pure Nash equilibrium $\mathbf{L}$. We then use this lower bound to prove an upper bound on $(k+1)$. Denote $\mathcal{M}_0$ the set of links with latency at least $k\cdot\mathsf{OPT}_\infty(\mathbf{w},m)$, and let

$$\Delta_0(\mathbf{L}) = \sum_{j\in\mathcal{M}_0}\delta_j(\mathbf{L})\,.$$

Thus,

$$\begin{aligned}\Delta_0(\mathbf{L}) &= \sum_{j\in\mathcal{M}_0}\delta_j(\mathbf{L}) \\ &\geq k\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\cdot|\mathcal{M}_0|\,. \end{aligned} \tag{4.36}$$

We show the following claim by induction on l:

Claim 4.93 *For all $l\in[k-1]$, there exists a set of links $\mathcal{M}_l$, $\mathcal{M}_l\cap(\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1})=\emptyset$, such that*

(1.) the cardinality of $\mathcal{M}_l$ is at least

$$|\mathcal{M}_l| \geq (k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\,,$$

(2.) for all $j\in\mathcal{M}_l$, the load on link j is bounded by

$$\delta_j(\mathbf{L}) \geq (k-l)\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\,,$$

(3.) the total load $\Delta_l(\mathbf{L})$ on links in $\mathcal{M}_l$ is at least

$$\Delta_l(\mathbf{L}) \geq (k-1)\prod_{j\in[l]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\,,$$

(4.) and there exist users with total load at least

$$\sum_{0\leq i\leq l}(\Delta_i(\mathbf{L})-|\mathcal{M}_i|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)) \geq (k-1)\prod_{j\in[l]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\,,$$

assigned to links in $\mathcal{M}_0\cup\ldots\cup\mathcal{M}_l$, and containing links in their strategy sets not in $\mathcal{M}_0\cup\cdots\cup\mathcal{M}_l$.

Proof: As our basis case, let $l=1$. Since $\delta_j(\mathbf{L})\geq k\cdot\mathsf{OPT}_\infty(\mathbf{w},m)$ for all $j\in\mathcal{M}_0$, there exist users with total load at least $(k-1)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)$ assigned to links in $\mathcal{M}_0$ containing links not in $\mathcal{M}_0$ in their strategy sets. This holds since it is possible to assign all users to links such that the latency on each link is at most $\mathsf{OPT}_\infty(\mathbf{w},m)$. Denote $\mathcal{M}_1$, $\mathcal{M}_1\cap\mathcal{M}_0=\emptyset$, the set of these links. It follows that

$$|\mathcal{M}_1| \geq (k-1)\cdot|\mathcal{M}_0|\,, \tag{4.37}$$

proving (1.). Since each user causes latency at most $\mathsf{OPT}_\infty(\mathbf{w},m)$ on each link, the definition of Nash equilibrium implies

$$k\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \leq \delta_j(\mathbf{L})+\mathsf{OPT}_\infty(\mathbf{w},m)$$

for all $j\in\mathcal{M}_1$. Thus, for all links $j\in\mathcal{M}_1$, the load on link j is bounded by

$$\begin{aligned}\delta_j(\mathbf{L}) &\geq k\cdot\mathsf{OPT}_\infty(\mathbf{w},m)-\mathsf{OPT}_\infty(\mathbf{w},m) \\ &= (k-1)\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\,, \end{aligned} \tag{4.38}$$

proving (2.), and therefore

$$\begin{aligned}\Delta_1(\mathbf{L}) &= \sum_{j\in\mathcal{M}_1}\delta_j(\mathbf{L})\\ &\overset{(4.38),\ page\ 149}{\geq} (k-1)\cdot|\mathcal{M}_1|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \qquad (4.39)\\ &\overset{(4.37),\ page\ 149}{\geq} (k-1)^2\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ ,\end{aligned}$$

proving (3.). Moreover,

$$\begin{aligned}&\sum_{0\leq i\leq 1}(\Delta_i(\mathbf{L})-|\mathcal{M}_i|\cdot\mathsf{OPT}_\infty(\mathbf{w},m))\\ &= \Delta_0(\mathbf{L})-|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)+\Delta_1(\mathbf{L})-|\mathcal{M}_1|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\\ &\overset{(4.36),\ page\ 149,\ (4.39)}{\geq} k\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)-|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\\ &\quad+(k-1)\cdot|\mathcal{M}_1|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)-|\mathcal{M}_1|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\\ &= (k-1)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)+(k-2)\cdot|\mathcal{M}_1|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\\ &\overset{(4.37),\ page\ 149}{\geq} (k-1)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)+(k-1)(k-2)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\\ &= (k-1)^2\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ ,\end{aligned}$$

proving (4.), and thus the claim holds for the basis case.
For the induction step, let $l\geq 2$, and assume that Claim 4.93 (page 149) holds for $(l-1)$. By induction hypothesis,

$$\sum_{0\leq i\leq l-1}(\Delta_i(\mathbf{L})-|\mathcal{M}_i|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)) \;\geq\; (k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ .$$

Thus, there exist users with total load at least

$$(k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)$$

assigned to links in $\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1}$ containing links not in $\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1}$ in their strategy set. This holds since it is possible to assign all users to links such that the latency on each link is at most $\mathsf{OPT}_\infty(\mathbf{w},m)$. Denote $\mathcal{M}_l$, $\mathcal{M}_l\cap(\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1})=\emptyset$, the set of these links. It follows that

$$|\mathcal{M}_l| \;\geq\; (k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\ , \qquad (4.40)$$

proving (1.). Since each user causes latency at most $\mathsf{OPT}_\infty(\mathbf{w},m)$ on each link, the definition of Nash equilibrium implies

$$(k-(l-1))\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \;\leq\; \delta_j(\mathbf{L})+\mathsf{OPT}_\infty(\mathbf{w},m)$$

for all $j\in\mathcal{M}_l$. Thus, for all links $j\in\mathcal{M}_l$ the load on link j is bounded by

$$\delta_j(\mathbf{L}) \;\geq\; (k-l)\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ , \qquad (4.41)$$

proving (2.), and therefore

$$\begin{aligned}
\Delta_l(\mathbf{L}) &= \sum_{j\in\mathcal{M}_l}\delta_j(\mathbf{L}) \\
&\overset{(4.41),\ page\ 150}{\geq} (k-l)\cdot|\mathcal{M}_l|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \qquad (4.42)\\
&\overset{(4.40),\ page\ 150}{\geq} (k-1)\prod_{j\in[l]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ ,
\end{aligned}$$

proving (3.). Moreover,

$$\begin{aligned}
&\sum_{0\leq i\leq l}(\Delta_i(\mathbf{L})-|\mathcal{M}_i|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)) \\
&= \Delta_l(\mathbf{L})-|\mathcal{M}_l|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)+\sum_{0\leq i\leq l-1}(\Delta_i(\mathbf{L})-|\mathcal{M}_i|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)) \\
&\overset{Induction}{\geq} \Delta_l(\mathbf{L})-|\mathcal{M}_l|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)+(k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \\
&\overset{(4.42)}{\geq} (k-l)\cdot|\mathcal{M}_l|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)-|\mathcal{M}_l|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \\
&\quad +(k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \\
&\overset{(4.40),\ page\ 150}{\geq} (k-l-1)(k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \\
&\quad +(k-1)\prod_{j\in[l-1]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m) \\
&= (k-1)\prod_{j\in[l]}(k-j)\cdot|\mathcal{M}_0|\cdot\mathsf{OPT}_\infty(\mathbf{w},m)\ ,
\end{aligned}$$

proving (4.). This completes the proof of the inductive claim. ■

We proceed by showing the upper bound on $(k+1)$. Since $m\geq|\mathcal{M}_{k-1}|+|\mathcal{M}_{k-2}|$ for all $k\geq 2$, we get

$$\begin{aligned}
m &\geq |\mathcal{M}_{k-1}|+|\mathcal{M}_{k-2}| \\
&\overset{Claim\ 4.93(1.)\ (page\ 149)}{\geq} (k-1)\prod_{j\in[k-2]}(k-j)\cdot|\mathcal{M}_0|+(k-1)\prod_{j\in[k-3]}(k-j)\cdot|\mathcal{M}_0| \\
&= |\mathcal{M}_0|\cdot(k-1)\left(\prod_{j\in[k-2]}(k-j)+\prod_{j\in[k-3]}(k-j)\right) \\
&\geq |\mathcal{M}_0|\cdot(k-1)\left((k-1)!+(k-2)!\right) \\
&= |\mathcal{M}_0|\cdot k! \\
&\overset{|\mathcal{M}_0|=1}{=} \Gamma(k+1)\ .
\end{aligned}$$

Thus,

$$\begin{aligned}
k+1 &\leq \Gamma^{-1}(m) \\
&\overset{(4.4),\ page\ 80}{=} O\left(\frac{\log m}{\log\log m}\right)\ ,
\end{aligned}$$

as needed. ■

Theorem 4.94 (Awerbuch *et al.* [6]) *Consider the model of arbitrary users with restricted strategy sets and identical links, restricted to pure Nash equilibria. Let* $r = \frac{\mathsf{OPT}_\infty(\mathbf{w},m)}{w_1}$. *Then,*

$$\mathsf{PoA} \quad = \quad \Theta\left(\frac{\log m}{r \cdot \log(1+\frac{\log m}{r})}\right) .$$

Related Links. For arbitrary users, Theorem 4.26 (page 106) implies that WORST PURE NE is $\mathcal{NP}$-complete. Since WORST PURE NE is $\mathcal{NP}$-complete in the strong sense [50], there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.95).

For identical users, we can prove the upper bound $\Gamma^{-1}(n)+1$ on the price of anarchy, using similar techniques as in [29] (Theorem 4.96). This upper bound is tight up to an additive constant if $n = m$ (see Theorem 4.91, page 147). For arbitrary users, the price of lies in between $m-1$ and m (Theorem 4.98, page 155). We can not hope to approximate a worst Nash equilibrium within $m-2-\varepsilon$ for any ε with $0 < \varepsilon \leq m-3$ unless $\mathcal{P} = \mathcal{NP}$ (Theorem 4.99, page 157).

Theorem 4.95 *Consider the model of arbitrary users with restricted strategy sets and related links. Then, there exists a pseudo-polynomial-time algorithm for* m-WORST PURE NE.

Proof: See Theorem 4.105 (page 161) for a generalization of this result. ■

Theorem 4.96 *Consider the model of identical users with restricted strategy sets and related links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \quad \leq \quad \Gamma^{-1}(n)+1 \quad = \quad O\left(\frac{\log n}{\log\log n}\right) .$$

Proof: Since we consider identical users, the latency of a link $j \in [m]$ is the number of users assigned to j, divided by c_j. For any integer $k \in \mathbb{N}$, consider an instance $(n,\mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$ with

$$k \cdot \mathsf{OPT}_\infty(n,\mathbf{c}) \quad \leq \quad \mathsf{SC}_\infty(n,\mathbf{c},\mathbf{L}) \quad < \quad (k+1) \cdot \mathsf{OPT}_\infty(n,\mathbf{c}) .$$

We give a lower bound on the number of users that are necessary for such a pure Nash equilibrium $\mathbf{L}$. We then use this lower bound to prove an upper bound on $(k+1)$.

Assume, without loss of generality, that $c_j \geq \frac{1}{\mathsf{OPT}_\infty(n,\mathbf{c})}$ for all links $j \in [m]$ except the links with maximum latency. We can make this restriction, since in an optimum assignment no user is assigned to a link $j \in [m]$ with $c_j < \frac{1}{\mathsf{OPT}_\infty(n,\mathbf{c})}$. Moreover, if we delete such a link j and all users assigned to it from $\mathbf{L}$, then the resulting pure assignment is still a pure Nash equilibrium with the same social cost. Denote $\mathcal{M}_0$ the set of links with latency at least $k \cdot \mathsf{OPT}_\infty(n,\mathbf{c})$, let $U_0(\mathbf{L})$ be the number of users assigned to a link in $\mathcal{M}_0$ and let $C_0 = \sum_{j \in \mathcal{M}_0} c_j$. Then,

$$U_0(\mathbf{L}) \quad \geq \quad k \cdot \mathsf{OPT}_\infty(n,\mathbf{c}) \cdot C_0 . \tag{4.43}$$

We show the following claim by induction on l:

Claim 4.97 *For all $l \in [k-1]$, there exists a set of links $\mathcal{M}_l$, $\mathcal{M}_l \cap (\mathcal{M}_0 \cup \ldots \cup \mathcal{M}_{l-1}) = \emptyset$ such that*

(1.) the total capacity C_l of links in $\mathcal{M}_l$ is at least

$$C_l \quad \geq \quad (k-1) \prod_{j \in [l-1]} (k-j) \cdot C_0 \; ,$$

(2.) for all $j \in \mathcal{M}_l$, the latency of link j is bounded by

$$\Lambda_j(\mathbf{L}) \quad \geq \quad (k-l) \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \; ,$$

(3.) the number of users $U_l(\mathbf{L})$ on links in $\mathcal{M}_l$ is at least

$$U_l(\mathbf{L}) \quad \geq \quad (k-1) \prod_{j \in [l]} (k-j) \cdot C_0 \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \; ,$$

(4.) and there exist at least

$$\sum_{0 \leq i \leq l} (U_i(\mathbf{L}) - C_i \cdot \mathsf{OPT}_\infty(n, \mathbf{c})) \quad \geq \quad (k-1) \prod_{j \in [l]} (k-j) \cdot C_0 \cdot \mathsf{OPT}_\infty(n, \mathbf{c})$$

users, assigned to links in $\mathcal{M}_0 \cup \ldots \cup \mathcal{M}_l$, containing links in their strategy sets not in $\mathcal{M}_0 \cup \cdots \cup \mathcal{M}_l$.

Proof: As our basis case, let $l = 1$. Since $\Lambda_j(\mathbf{L}) \geq k \cdot \mathsf{OPT}_\infty(n, \mathbf{c})$ for all $j \in \mathcal{M}_0$, there exist at least $(k-1) \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \cdot C_0$ users assigned to links in $\mathcal{M}_0$ containing links not in $\mathcal{M}_0$ in their strategy sets. This holds since it is possible to assign all users to links such that the latency on each link is at most $\mathsf{OPT}_\infty(n, \mathbf{c})$. Denote $\mathcal{M}_1$, $\mathcal{M}_1 \cap \mathcal{M}_0 = \emptyset$, the set of these links. It follows that

$$\begin{aligned} C_1 \quad &= \quad \sum_{j \in \mathcal{M}_1} c_j \\ &\geq \quad (k-1) \cdot C_0 \; , \end{aligned} \tag{4.44}$$

proving (1.). Since $c_j \geq \frac{1}{\mathsf{OPT}_\infty(n, \mathbf{c})}$ for all $j \in \mathcal{M}_1$, the definition of Nash equilibrium implies that

$$\begin{aligned} k \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \quad &\leq \quad \Lambda_j(\mathbf{L}) + \frac{1}{c_j} \\ &\leq \quad \Lambda_j(\mathbf{L}) + \mathsf{OPT}_\infty(n, \mathbf{c}) \end{aligned}$$

for all $j \in \mathcal{M}_1$. Thus, for all links $j \in \mathcal{M}_1$ the latency on link j is bounded from below by

$$\Lambda_j(\mathbf{L}) \quad \geq \quad (k-1) \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \; , \tag{4.45}$$

proving (2.), and therefore

$$\begin{aligned} U_1(\mathbf{L}) \quad &\geq \quad \sum_{j \in \mathcal{M}_1} \Lambda_j(\mathbf{L}) \cdot C_1 \\ &\overset{(4.45)}{\geq} \quad (k-1) \cdot C_1 \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \\ &\overset{(4.44)}{\geq} \quad (k-1)^2 \cdot C_0 \cdot \mathsf{OPT}_\infty(n, \mathbf{c}) \; , \end{aligned} \tag{4.46}$$

proving (3.). Moreover,

$$\begin{aligned}
&\sum_{0\le i\le 1}(U_i(\mathbf{L})-C_i\cdot\mathsf{OPT}_\infty(n,\mathbf{c}))\\
&\quad= \quad U_0(\mathbf{L})-C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+U_1(\mathbf{L})-C_1\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad\overset{(4.43),\ page\ 152,\ (4.46),\ page\ 153}{\ge} \quad k\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})-C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\qquad\qquad +(k-1)\cdot C_1\cdot\mathsf{OPT}_\infty(n,\mathbf{c})-C_1\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad= \quad (k-1)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+(k-2)\cdot C_1\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad\overset{(4.44),\ page\ 153}{\ge} \quad (k-1)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+(k-1)(k-2)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad= \quad (k-1)^2\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\ ,
\end{aligned}$$

proving (4.), and thus the claim holds for the basis case.
For the induction step, let $l\ge 2$, and assume that Claim 4.97 (page 153) holds for $(l-1)$. By induction hypothesis,

$$\sum_{0\le i\le l-1}(U_i(\mathbf{L})-C_i\cdot\mathsf{OPT}_\infty(n,\mathbf{c})) \ \ge\ (k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\ .$$

Thus, there exist at least

$$(k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})$$

users assigned to links in $\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1}$ containing links not in $\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1}$ in their strategy set. This holds since it is possible to assign all users to links such that the latency on each link is at most $\mathsf{OPT}_\infty(n,\mathbf{c})$. Denote $\mathcal{M}_l$, $\mathcal{M}_l\cap(\mathcal{M}_0\cup\ldots\cup\mathcal{M}_{l-1})=\emptyset$, the set of these links. It follows that

$$\begin{aligned}
C_l \ &= \ \sum_{j\in\mathcal{M}_l}c_j\\
&\ge \ (k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\ , \qquad (4.47)
\end{aligned}$$

proving (1.). Since $c_j\ge\frac{1}{\mathsf{OPT}_\infty(n,\mathbf{c})}$ for all $j\in\mathcal{M}_l$, the definition of Nash equilibrium implies

$$\begin{aligned}
(k-l+1)\cdot\mathsf{OPT}_\infty(n,\mathbf{c}) \ &\le \ \Lambda_j(\mathbf{L})+\frac{1}{c_j}\\
&\le \ \Lambda_j(\mathbf{L})+\mathsf{OPT}_\infty(n,\mathbf{c})
\end{aligned}$$

for all $j\in\mathcal{M}_l$. Thus, for all links $j\in\mathcal{M}_l$ the latency on link j is bounded by

$$\Lambda_j(\mathbf{L}) \ \ge\ (k-l)\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\ , \qquad (4.48)$$

proving (2.), and therefore

$$\begin{aligned}
U_l(\mathbf{L}) \quad &\ge \quad \sum_{j\in\mathcal{M}_l}\Lambda_j(\mathbf{L})\cdot C_l \qquad (4.49)\\
&\overset{(4.48)}{\ge} \quad (k-l)\cdot C_l\cdot\mathsf{OPT}_\infty(n,\mathbf{c}) \qquad (4.50)\\
&\overset{(4.47)}{\ge} \quad (k-1)\prod_{j\in[l]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\ ,
\end{aligned}$$

proving (3.). Moreover,

$$\begin{aligned}
&\sum_{0\le i\le l}\left(U_i(\mathbf{L})-C_i\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\right)\\
&\quad= \quad U_l(\mathbf{L})-C_l\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+\sum_{0\le i\le l-1}\left(U_i(\mathbf{L})-C_i\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\right)\\
&\quad\overset{\textit{Induction}}{\ge} \quad U_l(\mathbf{L})-C_l\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+(k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad\overset{(4.50),\ \textit{page}\ 154}{\ge} \quad (k-l-1)\cdot C_l\cdot\mathsf{OPT}_\infty(n,\mathbf{c})+(k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad\overset{(4.47),\ \textit{page}\ 154}{\ge} \quad (k-l-1)(k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\qquad\qquad +(k-1)\prod_{j\in[l-1]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\quad= \quad (k-1)\prod_{j\in[l]}(k-j)\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\ ,
\end{aligned}$$

proving (4.). This completes the proof of the inductive claim. ■

We proceed by proving an upper bound on $(k+1)$. Since $\mathsf{SC}_\infty(n,\mathbf{c})<(k+1)\cdot\mathsf{OPT}_\infty(n,\mathbf{c})$, we have $\Lambda_j(\mathbf{L})<(k+1)\cdot\mathsf{OPT}_\infty(n,\mathbf{c})$ for all $j\in\mathcal{M}_0$. Since at least one user is assigned to each link in $\mathcal{M}_0$, this implies $c_j>\frac{1}{(k+1)\mathsf{OPT}_\infty(n,\mathbf{c})}$ for all $j\in\mathcal{M}_0$. Thus,

$$C_0 \quad > \quad \frac{1}{(k+1)\cdot\mathsf{OPT}_\infty(n,\mathbf{c})}\ . \tag{4.51}$$

Since $n\ge U_{k-1}(\mathbf{L})+U_{k-2}(\mathbf{L})$ for $k\ge 3$, we get

$$\begin{aligned}
n \quad &\ge \quad U_{k-1}(\mathbf{L})+U_{k-2}(\mathbf{L})\\
&\overset{\textit{Claim}\ 4.97(3.)\ (\textit{page}\ 153)}{\ge} \quad 2\cdot(k-1)\cdot(k-1)!\cdot C_0\cdot\mathsf{OPT}_\infty(n,\mathbf{c})\\
&\overset{(4.51)}{\ge} \quad (k-1)!\\
&= \quad \Gamma(k)\ .
\end{aligned}$$

Thus,

$$\begin{aligned}
k+1 \quad &\le \quad \Gamma^{-1}(n)+1\\
&\overset{(4.4),\ \underline{\textit{page}}\ 80}{=} \quad O\left(\frac{\log n}{\log\log n}\right)\ ,
\end{aligned}$$

as needed. ■

Theorem 4.98 *Consider the model of arbitrary users with restricted strategy sets and related links, restricted to pure Nash equilibria. Then,*

$$m-1 \quad\le\quad \mathsf{PoA} \quad<\quad m\ .$$

Proof: For any integer $k \in \mathbb{N}$, consider an instance $(\mathbf{w}, \mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$ with

$$k \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \;\leq\; \mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L}) \;<\; (k+1) \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \,.$$

Upper bound: We show by induction on $i \in [k]$ that there exist links $l_1, \ldots, l_i$ with latencies

$$\Lambda_{l_i}(\mathbf{L}) \;\geq\; (k-i+1) \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \,.$$

As our basis case, let $i = 1$. Since $\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L}) \geq k \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$, there exists a link $l_1 \in [m]$ with latency $\Lambda_{l_1}(\mathbf{L}) \geq k \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$, proving the basis case. For the induction step, let $i \geq 2$, and assume that the claim holds for $(i-1)$. By induction hypothesis, there exist $i-1$ links $l_1, \ldots, l_{i-1}$ with

$$\begin{aligned} \Lambda_{l_j}(\mathbf{L}) &\geq (k-j+1) \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \\ &> \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) \end{aligned}$$

for all $j \in [i-1]$. Thus, there exists a user on $l_1 \cup \ldots \cup l_{i-1}$ that is assigned to some other link l_i in an optimum assignment, and the latency on l_i is at least

$$\Lambda_{l_i}(\mathbf{L}) \;\geq\; (k-i+1) \cdot \mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$$

by definition of Nash equilibrium. This completes the proof of the inductive claim. Since there exists at least one additional link with latency smaller than $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})$, we have $m \geq k+1$, proving the upper bound.
Lower bound: Consider an instance $(\mathbf{w}, \mathbf{c})$ with capacities

$$c_j \;=\; \frac{(m-1)!}{(j-1)!}$$

for all $j \in [m]$, and $n = m-1$ users with traffic $w_i = c_i$ and strategy set $R_i = \{i, i+1\}$ for all $i \in [m-1]$. The assignment $\mathbf{L}$ is defined as follows: We assign user i to link $i+1$ for all $i \in [n]$. Note that each user $i \in [n] \setminus \{1\}$ experiences latency $\frac{w_i}{c_{i+1}} = i$ on its link $i+1$, and that moving to the other link i in its strategy set would lead to latency

$$\begin{aligned} \frac{w_{i-1} + w_i}{c_i} &= \frac{1}{c_i} \left(\frac{(m-1)!}{(i-2)!} + \frac{(m-1)!}{(i-1)!} \right) \\ &= \frac{1}{c_i} \left(i \cdot \frac{(m-1)!}{(i-1)!} \right) \\ &= i \,. \end{aligned}$$

Furthermore, since $c_1 = c_2$, user 1 experiences the same latency on link 1 and 2 and has no incentive to move from link 2 to link 1. This implies that $\mathbf{L}$ is a Nash equilibrium. In an optimum assignment, each user $i \in [n]$ chooses link i as its strategy, yielding $\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c}) = 1$. This implies

$$\frac{\mathsf{SC}_\infty(\mathbf{w}, \mathbf{c}, \mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w}, \mathbf{c})} \;=\; m-1 \,,$$

which completes the proof of the lower bound. ■

Theorem 4.99 *Consider the model of arbitrary users with restricted strategy sets and related links. If, for any* ε *with* $0 < \varepsilon \leq m-3$, WORST PURE NE *is* $(m-2-\varepsilon)$*-approximable, then* $\mathcal{P} = \mathcal{NP}$.

Proof: We prove this result by reduction from 2-PARTITION, that is, we employ a polynomial time transformation from 2-PARTITION to WORST PURE NE. For any ε with $0 < \varepsilon \leq m-3$, we construct an instance of WORST PURE NE such that if we had a polynomial-time $(m-2-\varepsilon)$-approximation algorithm for WORST PURE NE, then we could decide whether an instance of 2-PARTITION is positive in polynomial time. From this construction the theorem then follows.

Consider an arbitrary instance of 2-PARTITION, and let $S = \sum_{u_i \in \mathcal{U}} s(u_i)$. From this instance we construct an instance for the stated problem as follows:

- There are m links with capacity

$$c_j = \begin{cases} ((m-2)!+1) \cdot \frac{S}{2} & \text{if } j \in [2], \\ \frac{(m-2)!}{(j-2)!} \cdot \frac{S}{2} & \text{if } j \in [m] \setminus [2]. \end{cases}$$

- There are $n = |\mathcal{U}| + (m-2)$ users with traffic

$$w_i = \begin{cases} s(u_i) & \text{if } i \in [|\mathcal{U}|], \\ c_2 - \frac{S}{2} & \text{if } i = |\mathcal{U}|+1, \\ c_{i-|\mathcal{U}|+1} & \text{if } i \in [n] \setminus [|\mathcal{U}|+1], \end{cases}$$

  and strategy set

$$R_i = \begin{cases} \{1,2\} & \text{if } i \in [|\mathcal{U}|], \\ \{1,2,3\} & \text{if } i = |\mathcal{U}|+1, \\ \{i-|\mathcal{U}|+1, i-|\mathcal{U}|+2\} & \text{if } i \in [n] \setminus [|\mathcal{U}|+1]. \end{cases}$$

Clearly, this is a polynomial time transformation. We prove that this is a transformation from 2-PARTITION to WORST PURE NE.

(1.) The instance of 2-PARTITION is positive:
Consider a partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1, \mathcal{U}_2$ with $\sum_{u_i \in \mathcal{U}_1} s(u_i) = \sum_{u_i \in \mathcal{U}_2} s(u_i)$. Use this partition to define a pure assignment $\mathbf{L}$ (illustrated in Figure 4.12) as follows:

- For each item $u_i \in \mathcal{U}_1$, user i is assigned to link 1. For each item $u_i \in \mathcal{U}_2$, user i is assigned to link 2.
- Each user $i \in [n] \setminus [|\mathcal{U}|]$, is assigned to link $i - |\mathcal{U}| + 2$.

Clearly, $\Lambda_1(\mathbf{L}) = \Lambda_2(\mathbf{L}) = \frac{1}{(m-2)!+1}$ and $\Lambda_j(\mathbf{L}) = j-2$ for all $j \in [m] \setminus [2]$. So, all users in $[|\mathcal{U}|]$ are satisfied. Moreover, user $|\mathcal{U}|+1$ experiences latency 1 on its link 3, and moving to link 1 or 2 would also lead to latency 1. Furthermore, every user $i \in [n] \setminus [|\mathcal{U}|+1]$ experiences latency

$$\begin{aligned} \frac{w_i}{c_{i-|\mathcal{U}|+2}} &= \frac{c_{i-|\mathcal{U}|+1}}{c_{i-|\mathcal{U}|+2}} \\ &= i - |\mathcal{U}| + 2 \end{aligned}$$

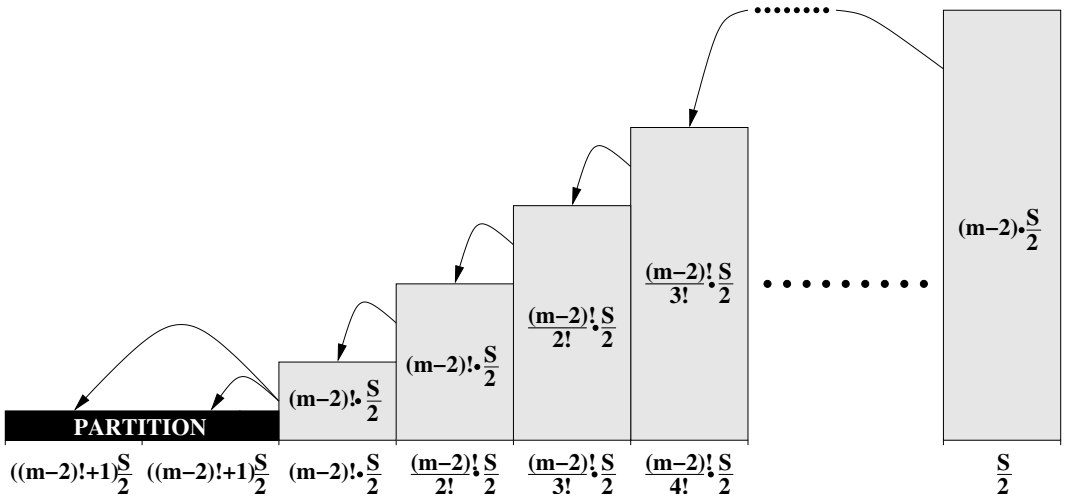


Figure 4.12: Assignment of the instance of WORST PURE NE constructed from a positive instance of PARTITION in the proof of Theorem 4.99 (page 157). Each small arrow points from an assigned user to another link in its strategy set.

on its link $i-|\mathcal{U}|+2$, whereas moving to the other link $i-|\mathcal{U}|+1$ in its strategy set would lead to latency

$$\begin{aligned}\frac{w_{i-1}+w_i}{c_{i-|\mathcal{U}|+1}} &= \frac{c_{i-|\mathcal{U}|}+c_{i-|\mathcal{U}|+1}}{c_{i-|\mathcal{U}|+1}} \\ &= 1+\frac{c_{i-|\mathcal{U}|}}{c_{i-|\mathcal{U}|+1}} \\ &= i-|\mathcal{U}|+2\,.\end{aligned}$$

This shows that the users in $[n]\setminus[|\mathcal{U}|]$ are also satisfied. So, $\mathbf{L}$ is a Nash equilibrium. Moreover,

$$\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) = m-2\,.$$

Now, consider any pure Nash equilibrium $\mathbf{L}'$ in which the users $i\in[|\mathcal{U}|]$ are not assigned to link 1 and 2 such that they cause load $\frac{S}{2}$ on both links. Assume, by way of contradiction, that user $|\mathcal{U}|+1$ is assigned to link 3. Clearly, one of the links 1 and 2, say 1, has load less than $\frac{S}{2}$. We get

$$\begin{aligned}\lambda_{|\mathcal{U}|+1}(\mathbf{L}') &= \frac{\delta_3(\mathbf{L}')}{c_3} \\ &\geq 1 \\ &= \frac{((m-2)!+1)\cdot\frac{S}{2}}{((m-2)!+1)\cdot\frac{S}{2}} \\ &= \frac{\frac{S}{2}+w_{|\mathcal{U}|+1}}{c_1} \\ &> \frac{\delta_1(\mathbf{L}')+w_{|\mathcal{U}|+1}}{c_1}\,,\end{aligned}$$

showing that user $|\mathcal{U}|+1$ is unsatisfied, a contradiction to the fact that $\mathbf{L}'$ is a Nash equilibrium. Thus, user $|\mathcal{U}|+1$ is assigned to link 1 or link 2. By definition of Nash equilibrium, this implies that all users $i \in [n] \setminus [|\mathcal{U}|+1]$ are assigned to link $i-|\mathcal{U}|+1$. Hence,

$$\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}') \;=\; 1\,,$$

and we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}')} &= m-2\\ &\overset{\varepsilon>0}{>} m-2-\varepsilon\,.\end{aligned}$$

Thus, no such Nash equilibrium $\mathbf{L}'$ approximates the worst pure Nash equilibrium within the claimed factor.

(2.) The instance of 2-PARTITION is negative:
For any partition of $\mathcal{U}$ into disjoint subsets $\mathcal{U}_1, \mathcal{U}_2$, we have $\sum_{u_i\in\mathcal{U}_1} s(u_i) \neq \sum_{u_i\in\mathcal{U}_2} s(u_i)$. This implies that either $\sum_{u_i\in\mathcal{U}_1} s(u_i) < \frac{S}{2}$ or $\sum_{u_i\in\mathcal{U}_2} s(u_i) < \frac{S}{2}$. Thus, the users $i \in [|\mathcal{U}|]$, can not be assigned to links 1 and 2 such that they cause load $\frac{S}{2}$ on each of these links. As seen in the previous case, this implies that the social cost of any pure Nash equilibrium $\mathbf{L}$ is $\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) = 1$. Clearly, $\mathsf{OPT}_\infty(\mathbf{w},m,\mathbf{L}) \geq 1$, and we get

$$\begin{aligned}\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} &\leq 1\\ &\overset{\varepsilon\leq m-3}{\leq} m-2-\varepsilon\,.\end{aligned}$$

This shows that all Nash equilibria approximate the worst pure Nash equilibrium within the claimed factor.

Therefore, if we had a polynomial-time $(m-2-\varepsilon)$-approximation algorithm for WORST PURE NE, we could use it to decide whether an instance of 2-PARTITION is positive in the following way: We apply the approximation algorithm to the corresponding instance of WORST PURE NE and we answer *yes* if and only if it returns a solution with social cost $m-2$. ■

4.6.2 Mixed Nash Equilibria

Awerbuch *et al.* [6] gave the only result on mixed Nash equilibria for restricted strategy sets. Using similar techniques as Czumaj and Vöcking [29], they showed an asymptotically tight bound on the price of anarchy.

Theorem 4.100 (Awerbuch *et al.* [6]) *Consider the model of arbitrary users with restricted strategy sets and identical links. Let* $r = \frac{\mathsf{OPT}_\infty(\mathbf{w},m)}{w_1}$. *Then,*

$$\mathsf{PoA} \;=\; \Theta\left(\frac{\log m}{r\cdot\log\log(1+\frac{\log m}{r})}\right)\,.$$

4.7 Makespan Social Cost and Unrelated Links

In this section, we consider the KP-model with makespan social cost for the most general case of unrelated links. In the following, let

$$W = \sum_{i\in[n]} \max_{j\in[m]} \left\{ w_{ij} \mid w_{ij} < \infty \right\} .$$

Subsection 4.7.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.7.2 hold for general (i.e. mixed) Nash equilibria. Subsection 4.7.3 concentrates on the fully mixed Nash equilibrium. In order to illustrate the results, we use the following instance.

Example 4.101 *Consider the following instance* $(\mathbf{w}, 2)$*: We have* $n = 3$ *users and* $m = 2$ *links. It is* $w_{11} = w_{21} = 10$, $w_{12} = w_{22} = 1$, $w_{31} = 1$, *and* $w_{32} = 19$.

4.7.1 Pure Nash Equilibria

4.7.1.1 Computation of Nash Equilibria

Up to now only little attention has been payed to Nash equilibria in the model of unrelated links. The problem of computing pure Nash equilibria for unrelated links appears to be intractable in the current state-of-the-art. Of course, a pure Nash equilibrium can be computed by performing sequences of (not necessarily greedy) selfish steps. However, up to now it is unknown whether, starting with any pure assignment, there always exists a sequence of length polynomial in the number of users and links ending in a pure Nash equilibrium. A trivial upper bound on the number of selfish steps before reaching a pure Nash equilibrium is m^n. Other bounds for special instances were given by Even-Dar *et al.* [42].

Theorem 4.102 (Even-Dar *et al.* [42]) *Consider the model of unrelated links. Then, for any instance* $(\mathbf{w}, m)$ *with* $w_{ij} \in \mathbb{N}$ *for all* $i \in [n]$ *and* $j \in [m]$ *and associated pure assignment, the length of a sequence of (not necessarily greedy) selfish steps is at most* $\frac{4^W}{2}$ *before reaching a Nash equilibrium*

Theorem 4.103 (Even-Dar *et al.* [42]) *Consider the model of unrelated links. Then, for any instance* $(\mathbf{w}, m)$ *with* $w_{ij} \in \mathbb{N}$ *for all* $i \in [n]$ *and* $j \in [m]$ *and associated pure assignment, the length of a sequence of (not necessarily greedy) selfish steps using the rule* MAX LOAD MACHINE *is at most*

$$\frac{4mW + m^{\frac{W}{m} + \max_{i\in[n], j\in[m]} w_{ij}}}{2}$$

before reaching a Nash equilibrium.

4.7.1.2 Computation of Best Nash Equilibria

Again, Theorem 4.22 (page 104) implies that BEST PURE NE is $\mathcal{NP}$-complete. Clearly, there exists no pseudo-polynomial algorithm to solve BEST PURE NE since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense [50]. However, for constant m we can give such an algorithm (Theorem 4.104, page 161). In general, we can not hope to approximate a best pure Nash equilibrium within a factor lower than $\frac{3}{2}$ by Theorem 4.89 (page 145).

Theorem 4.104 *Consider the model of unrelated links. Then, there exists a pseudo-polynomial-time algorithm for m-*BEST PURE NE.

Proof: Assume, without loss of generality, that $w_{ij} \in \mathbb{N} \cup \{\infty\}$. We start with a state set S_0 in which all links are empty. After inserting the first l users, the state set S_l consists of all $(m \times m)$-tuples

$$([\delta_j, \widetilde{w}_1, \ldots, \widetilde{w}_{j-1}, \widetilde{w}_{j+1}, \ldots, \widetilde{w}_m])_{j \in [m]} ,$$

where $\widetilde{w}_k$, $k \in [m] \setminus \{j\}$, is the smallest traffic of a user on link $j \in [m]$ if he moves to link k. We need at most $m|S_l|$ steps to create S_{l+1} from S_l, and

$$\begin{aligned} |S_l| &\leq W^m \left(\left(\max_{i \in [n], j \in [m]} w_{ij} \right)^{m-1} \right)^m \\ &< W^m \left(\max_{i \in [n], j \in [m]} w_{ij} \right)^{m^2} . \end{aligned}$$

Thus, the total number of steps is bounded by

$$nmW^m \left(\max_{i \in [n], j \in [m]} w_{ij} \right)^{m^2} .$$

Moreover, we at most have to store the assignments of S_l and S_{l+1}, respectively, needing at most space

$$2W^m \left(\max_{i \in [n], j \in [m]} w_{ij} \right)^{m^2} .$$

Checking all assignments in S_n to be a Nash equilibrium and finding the pure Nash equilibrium with minimum social cost takes at most time

$$m^2 W^m \left(\max_{i \in [n], j \in [m]} w_{ij} \right)^{m^2} .$$

This completes the proof of the claim. ■

4.7.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

Again, Theorem 4.26 (page 106) implies that WORST PURE NE is $\mathcal{NP}$-complete. The fact that WORST PURE NE is $\mathcal{NP}$-complete in the strong sense [50] implies that there exists no pseudo-polynomial algorithm to solve it. For constant m, however, we can give such an algorithm (Theorem 4.105).

Awerbuch *et al.* [6] showed an asymptotically tight bound on the price of anarchy (Theorem 4.106, page 162). A *tight* bound for the case that $w_{ij} < \infty$ for all $i \in [n]$ and $j \in [m]$ is given in Theorem 4.107 (page 162).

Theorem 4.105 *Consider the model of unrelated links. Then, there exists a pseudo-polynomial-time algorithm for m-*WORST PURE NE.

Proof: Clearly, the proof of Theorem 4.104 (page 161) can be adapted to m-WORST PURE NE. ■

Theorem 4.106 (Awerbuch *et al.* [6]) *Consider the model of unrelated links, restricted to pure Nash equilibria. Let* $s = \max_{i\in[n],j_1,j_2\in[m]} \left\{ \frac{w_{ij_1}}{w_{ij_2}} \;\middle|\; \frac{w_{ij_1}}{w_{ij_2}} < \infty \right\}$. *Then,*

$$\mathsf{PoA} \;=\; \Theta\left(s + \frac{\log m}{\log\left(1+\frac{\log m}{s}\right)} \right) .$$

Theorem 4.107 *Consider the model of unrelated links, restricted to pure Nash equilibria. Then, for any instance* $(\mathbf{w},m)$ *with* $w_{ij} < \infty$ *for all* $i \in [n]$ *and* $j \in [m]$*, it is*

$$\mathsf{PoA} \;=\; \frac{\max_{i\in[n],j\in[m]} w_{ij}}{\min_{i\in[n],j\in[m]} w_{ij}} .$$

Proof:
Upper bound: Fix any instance $(\mathbf{w},m)$. Without loss of generality, assume $\min_{i\in[n],j\in[m]} w_{ij} = 1$. In the following, let

$$\tau \;=\; \frac{\max_{i\in[n],j\in[m]} w_{ij}}{\min_{i\in[n],j\in[m]} w_{ij}} \;=\; \max_{i\in[n],j\in[m]} w_{ij} .$$

Assume, by way of contradiction, that there exists a pure Nash equilibrium $\mathbf{L}$ with

$$\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} \;>\; \tau ,$$

and assume, without loss of generality, that $\Lambda_1(\mathbf{L}) = \mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})$. Then, there exists an integer $k \in \mathbb{N}$ with

$$k\cdot\tau \;<\; \frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} \;\leq\; (k+1)\cdot\tau .$$

Clearly, every user causes at most load τ on each link $j \in [m]$. Thus, by definition of Nash equilibrium, we get

$$\Lambda_j(\mathbf{L}) \;>\; (k-1)\cdot\tau$$

for all $j \in [m]\setminus\{1\}$. This implies that at least $(k+1)$ users are assigned to link 1 and at least k users are assigned to each link $j \in [m]\setminus\{1\}$. Thus, the total number of users is $n \geq mk+1$, proving that $\mathsf{OPT}_\infty(\mathbf{w},m) \geq (k+1)$ and therefore

$$\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} \;\leq\; \frac{(k+1)\cdot\tau}{(k+1)} \;=\; \tau ,$$

a contradiction.
Lower bound: Consider the following instance: There are $n = 2$ users and $m = 2$ links. It is $w_{11} = w_{22} = 1$ and $w_{12} = w_{21} = k$. On the one hand, $\langle 1,2\rangle$ is an optimum assignment with

social cost $\mathsf{OPT}_\infty(\mathbf{w},m)=1$. On the other hand, the pure assignment $\mathbf{L}=\langle 2,1\rangle$ is a pure Nash equilibrium with social cost $\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})=k$. We get

$$\frac{\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},m)} \;=\; k \;=\; \frac{\max_{i\in[n],j\in[m]} w_{ij}}{\min_{i\in[n],j\in[m]} w_{ij}}\,,$$

as needed. ■

Example 4.101 (continued) *For the given instance, the pure assignment* $\mathbf{L}=\langle 1,1,2\rangle$ *is clearly a pure Nash equilibrium with social cost* $\mathsf{SC}_\infty(\mathbf{w},2,\mathbf{L})=20$ *whereas the optimum assignment* $\langle 2,2,1\rangle$ *has social cost* $\mathsf{OPT}_\infty(\mathbf{w},2)=2$ *(see Figure 4.13). Thus,*

$$\frac{\mathsf{SC}_\infty(\mathbf{w},2,\mathbf{L})}{\mathsf{OPT}_\infty(\mathbf{w},2)} \;=\; 10 \;<\; 19 \;=\; \frac{\max_{i\in[3],j\in[2]} w_{ij}}{\min_{i\in[3],j\in[2]} w_{ij}}\,.$$

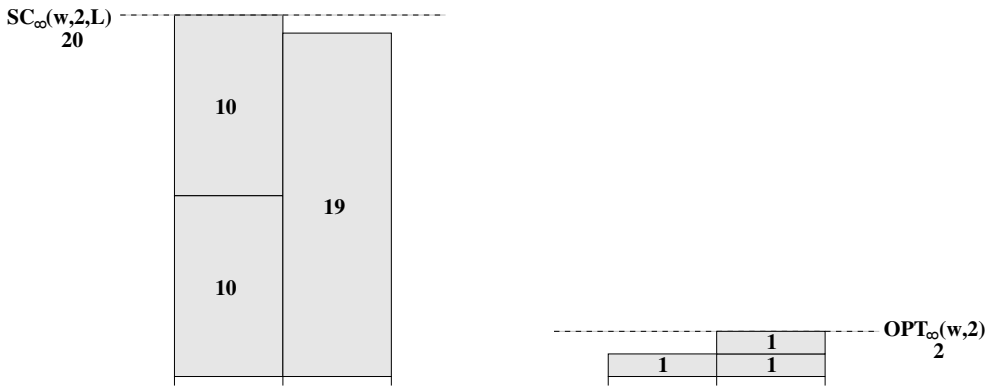


Figure 4.13: Pure Nash equilibrium $\mathbf{L}=\langle 1,1,2\rangle$ (left hand side) and optimum assignment $\langle 2,2,1\rangle$ (right hand side) for the instance in Example 4.101 (page 160).

4.7.2 Mixed Nash Equilibria

The only result on mixed Nash equilibria was proved by Awerbuch *et al.* [6]. They showed the following asymptotically tight bound on the price of anarchy.

Theorem 4.108 (Awerbuch *et al.* [6]) *Consider the model of unrelated links. Moreover, let* $s=\max_{i\in[n],j_1,j_2\in[m]}\left\{\frac{w_{ij_1}}{w_{ij_2}} \;\middle|\; \frac{w_{ij_1}}{w_{ij_2}}<\infty\right\}$. *Then,*

$$\mathsf{PoA} \;=\; \Theta\left(s+\frac{s\cdot\log m}{\log(1+s\cdot\log(\frac{\log m}{s}))}\right)\,.$$

4.7.3 Fully Mixed Nash Equilibria

We now consider fully mixed Nash equilibria. For any instance $(\mathbf{w}, m)$ with $n \leq m$, the minimum individual cost of any user $i \in [n]$ in a pure Nash equilibrium $\mathbf{L}$ is smaller than the minimum expected individual cost of user i in the fully mixed Nash equilibrium $\mathbf{F}$ (Proposition 4.109). Clearly, the social cost of any pure Nash equilibrium $\mathbf{L}$ is equal to the maximum of the expected latencies, while the social cost of a fully mixed Nash equilibrium $\mathbf{F}$ is at least the expected individual cost of any user. Hence, Proposition 4.109 implies that for $n \leq m$ the social cost of every pure Nash equilibrium $\mathbf{L}$ is at most the social cost of the fully mixed Nash equilibrium $\mathbf{F}$ (Theorem 4.110). Moreover, the FMNE Conjecture holds for $n = m = 2$ (Theorem 4.111). However, for the case $n = 3$ and $m = 2$ there exist instances (Example 4.101, page 160) for which the FMNE Conjecture does not hold (Theorem 4.112, page 165).

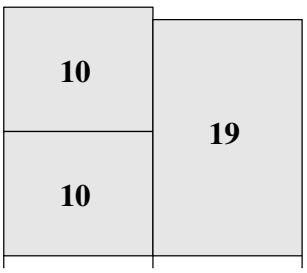
Proposition 4.109 (Lücking *et al.* [103]) *Consider the model of unrelated links. Then, for any instance $(\mathbf{w}, m)$ with $n \leq m$ and associated pure Nash equilibrium $\mathbf{L}$ for which a fully mixed Nash equilibrium $\mathbf{F}$ exists, it is $\lambda_i(\mathbf{L}) < \lambda_i(\mathbf{F})$ for all $i \in [n]$.*

Theorem 4.110 (Lücking *et al.* [103]) *Consider the model of unrelated links, restricted to pure Nash equilibria. If $n \leq m$, then the* FMNE *Conjecture is valid.*

Theorem 4.111 (Lücking *et al.* [103]) *Consider the model of unrelated links. If $n = m = 2$, then the* FMNE *Conjecture is valid.*

Example 4.101 (continued) *For the given instance, first consider the pure assignment $\mathbf{L} = \langle 1, 1, 2 \rangle$. Since*

$$
\begin{aligned}
\lambda_{11}(\mathbf{L}) &= w_{11} + w_{21} = 20\,, \\
\lambda_{12}(\mathbf{L}) &= w_{12} + w_{32} = 20\,, \\
\lambda_{21}(\mathbf{L}) &= w_{21} + w_{11} = 20\,, \\
\lambda_{22}(\mathbf{L}) &= w_{22} + w_{32} = 20\,, \\
\lambda_{32}(\mathbf{L}) &= w_{32} = 19\,, \\
\lambda_{31}(\mathbf{L}) &= w_{31} + w_{11} + w_{21} = 21\,,
\end{aligned}
$$



we have $\lambda_{11}(\mathbf{L}) \leq \lambda_{12}(\mathbf{L})$, $\lambda_{21}(\mathbf{L}) \leq \lambda_{22}(\mathbf{L})$, and $\lambda_{32}(\mathbf{L}) < \lambda_{31}(\mathbf{L})$. Thus, $\mathbf{L}$ is a pure Nash equilibrium, and its social cost is

$$\mathsf{SC}_\infty(\mathbf{w}, 2, \mathbf{L}) = \max\{\Lambda_1(\mathbf{L}), \Lambda_2(\mathbf{L})\} = \max\{20, 19\} = 20\,.$$

Now, consider the fully mixed assignment $\mathbf{F}$ *with* $f_{11} = f_{21} = \frac{10}{11}$, $f_{12} = f_{22} = \frac{1}{11}$, $f_{31} = \frac{1}{20}$, *and* $f_{32} = \frac{19}{20}$. *Since*

$$\begin{aligned}
\lambda_{11}(\mathbf{F}) &= w_{11} + f_{21}w_{21} + f_{31}w_{31} &&= \frac{4211}{220}\,,\\
\lambda_{12}(\mathbf{F}) &= w_{12} + f_{22}w_{22} + f_{32}w_{32} &&= \frac{4211}{220}\,,\\
\lambda_{21}(\mathbf{F}) &= w_{21} + f_{11}w_{11} + f_{31}w_{31} &&= \frac{4211}{220}\,,\\
\lambda_{22}(\mathbf{F}) &= w_{22} + f_{12}w_{12} + f_{32}w_{32} &&= \frac{4211}{220}\,,\\
\lambda_{31}(\mathbf{F}) &= w_{31} + f_{11}w_{11} + f_{21}w_{21} &&= \frac{211}{11}\,,\\
\lambda_{31}(\mathbf{F}) &= w_{32} + f_{12}w_{12} + f_{22}w_{22} &&= \frac{211}{11}\,,
\end{aligned}$$

it follows that $\mathbf{F}$ *is a fully mixed Nash equilibrium. Its social cost is*

$$\begin{aligned}
\mathsf{SC}_\infty(\mathbf{w},2,\mathbf{F}) &= f_{11}f_{21}f_{31}(w_{11}+w_{21}+w_{31}) + f_{11}f_{21}f_{32}\max\{w_{11}+w_{21},w_{32}\}\\
&\quad + f_{11}f_{22}f_{31}\max\{w_{11}+w_{31},w_{22}\} + f_{11}f_{22}f_{32}\max\{w_{11},w_{22}+w_{32}\}\\
&\quad + f_{12}f_{21}f_{31}\max\{w_{12},w_{21}+w_{31}\} + f_{12}f_{21}f_{32}\max\{w_{12}+w_{32},w_{21}\}\\
&\quad + f_{12}f_{22}f_{31}\max\{w_{12}+w_{22},w_{31}\} + f_{12}f_{22}f_{32}\max\{w_{12}+w_{22},w_{32}\}\\
&= \frac{10}{11}\cdot\frac{10}{11}\cdot\frac{1}{20}\cdot 21 + \frac{10}{11}\cdot\frac{10}{11}\cdot\frac{19}{20}\cdot\max\{20,19\}\\
&\quad + \frac{10}{11}\cdot\frac{1}{11}\cdot\frac{1}{20}\cdot\max\{11,1\} + \frac{10}{11}\cdot\frac{1}{11}\cdot\frac{19}{20}\cdot\max\{10,20\}\\
&\quad + \frac{1}{11}\cdot\frac{10}{11}\cdot\frac{1}{20}\cdot\max\{11,1\} + \frac{1}{11}\cdot\frac{10}{11}\cdot\frac{19}{20}\cdot\max\{1,20\}\\
&\quad + \frac{1}{11}\cdot\frac{1}{11}\cdot\frac{1}{20}\cdot\max\{2,1\} + \frac{1}{11}\cdot\frac{1}{11}\cdot\frac{19}{20}\cdot 21\\
&= \frac{48283}{2420} \quad < \quad 20\,.
\end{aligned}$$

Thus, $\mathsf{SC}_\infty(\mathbf{w},2,\mathbf{L}) > \mathsf{SC}_\infty(\mathbf{w},2,\mathbf{F})$, *showing that the* FMNE *Conjecture is not valid.*

Theorem 4.112 (Lücking *et al.* [103]) *Consider the model of unrelated links. If* $n = 3$ *and* $m = 2$, *then the* FMNE *Conjecture is not valid.*

4.8 Polynomial Social Cost and Identical Links

In this section, we consider the KP-model with polynomial social cost and identical links. Clearly, for any instance $(\mathbf{w},m)$ and associated assignment $\mathbf{P}$, the expected latency on a link is equal to its expected load, that is, $\Lambda_j(\mathbf{P}) = \delta_j(\mathbf{P})$ for all $j \in [m]$. Subsection 4.8.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.8.2 hold for fully mixed Nash equilibria. In order to illustrate the results, we use the following instance.

Example 4.113 *Consider the following instance* $(\mathbf{w},3)$*: We have* $n=7$ *arbitrary users,* $m=3$ *identical links and the polynomial cost function* $\pi^2(x)=x^2$*. There are two users with traffics* $w_1=w_2=5$*, two users with traffics* $w_3=w_4=4$*, and three users with traffics* $w_5=w_6=w_7=3$*.*

4.8.1 Pure Nash Equilibria

4.8.1.1 Computation of Nash Equilibria

Clearly, the usage of polynomial social cost as the cost measure of social welfare alters neither the definition of individual cost nor the set of Nash equilibria. This implies that all results on the convergence of sequences of selfish steps in Section 4.4 also hold in this model. Moreover, we can use selfish steps to nashify any given pure assignment since selfish steps do not increase social cost (Proposition 4.114). If the users are identical, then Proposition 4.8 (page 96) implies that all pure Nash equilibria have optimum polynomial social cost, and such a pure Nash equilibrium can be computed in $O(n)$ time (Proposition 4.115, page 167). For arbitrary users, we can still use the LPT-algorithm (see Algorithm 3, page 93) to compute a pure Nash equilibrium. Chandra and Wong [19] proved an upper bound on the performance quality of LPT with respect to the d-norm (Theorem 4.116, page 167). This bound reduces to $\frac{25}{24}$ if $d=2$ (for this special case, Leung and Wei [98] gave tighter bounds).

Proposition 4.114 *Consider the model of arbitrary users and identical links. Then, for any instance* $(\mathbf{w},m)$ *and associated pure assignment* $\mathbf{L}$*, a selfish step yields a pure assignment* $\mathbf{L}'$ *with* $\mathsf{SC}_{\pi^d(x)}(\mathbf{w},m,\mathbf{L}') \leq \mathsf{SC}_{\pi^d(x)}(\mathbf{w},m,\mathbf{L})$*.*

Proof: Fix any instance $(\mathbf{w},m)$ and associated pure assignment $\mathbf{L}$, and let $i \in [n]$ be an unsatisfied user who can improve by moving from link $j_1 \in [m]$ to link $j_2 \in [m]$, yielding a pure assignment $\mathbf{L}'$. Clearly,

$$\delta_{j_1}(\mathbf{L}) \;>\; \delta_{j_2}(\mathbf{L}) + w_i \;, \tag{4.52}$$

and $\delta_\ell(\mathbf{L}') = \delta_\ell(\mathbf{L})$ for all $\ell \in [m] \setminus \{j_1, j_2\}$. Since, by definition of polynomial social cost, all coefficients are non-negative, Equation (4.13) (page 88) implies that it suffices to show

$$\delta_{j_1}(\mathbf{L}')^t + \delta_{j_2}(\mathbf{L}')^t \;\leq\; \delta_{j_1}(\mathbf{L})^t + \delta_{j_2}(\mathbf{L})^t$$

for all $t \in [d]$. Fix any such $t \in [d]$. If $t=1$, then equality follows. So assume $t \geq 2$, and consider the function

$$\begin{aligned} f(w_i) &= \delta_{j_1}(\mathbf{L}')^t + \delta_{j_2}(\mathbf{L}')^t \\ &= (\delta_{j_1}(\mathbf{L}) - w_i)^t + (\delta_{j_2}(\mathbf{L}) + w_i)^t \;, \\ f'(w_i) &= t(\delta_{j_2}(\mathbf{L}) + w_i)^{t-1} - t(\delta_{j_1}(\mathbf{L}) - w_i)^{t-1} \;. \end{aligned}$$

We proceed by case analysis:

- $\underline{\delta_{j_1}(\mathbf{L}) - w_i \geq \delta_{j_2}(\mathbf{L}) + w_i}$: In this case, $f'(w_i) \leq 0$. Thus, $f(w_i)$ is monotonic decreasing in w_i, showing that

$$f(w_i) \overset{w_i>0}{\leq} f(0) \;=\; \delta_{j_1}(\mathbf{L})^t + \delta_{j_2}(\mathbf{L})^t \;.$$

- $\delta_{j_1}(\mathbf{L}) - w_i < \delta_{j_2}(\mathbf{L}) + w_i$: In this case, $f'(w_i) > 0$. Thus, $f(w_i)$ is strictly increasing in w_i. Thus, we get

$$f(w_i) \overset{(4.52),\ page\ 166}{<} f(\delta_{j_1}(\mathbf{L}) - \delta_{j_2}(\mathbf{L})) \quad = \quad \delta_{j_1}(\mathbf{L})^t + \delta_{j_2}(\mathbf{L})^t \ .$$

Thus, $\delta_{j_1}(\mathbf{L}')^t + \delta_{j_2}(\mathbf{L}')^t \leq \delta_{j_1}(\mathbf{L})^t + \delta_{j_2}(\mathbf{L})^t$ in both cases, as needed. ■

Proposition 4.115 *Consider the model of identical users and identical links. Then, for any instance (n,m) and associated pure Nash equilibrium $\mathbf{L}$, it is $\mathsf{SC}_{\pi^d(x)}(n,m,\mathbf{L}) = \mathsf{OPT}_{\pi^d(x)}(n,m)$, and such a pure Nash equilibrium $\mathbf{L}$ can be computed in $O(n)$ time.*

Theorem 4.116 (Chandra and Wong [19], Fotakis *et al.* [50]) *Consider the model of arbitrary users, identical links and polynomial cost function $\pi^d(x) = x^d$. Then, for any instance $(\mathbf{w},m)$, LPT computes a pure Nash equilibrium $\mathbf{L}$ with*

$$\frac{\mathsf{SC}_{x^d}(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_{x^d}(\mathbf{w},m)} \quad \leq \quad \left(\frac{3^d - 2^d}{d}\right)^d \left(\frac{d-1}{2\cdot 3^d - 3\cdot 2^d}\right)^{d-1} ,$$

using $O((n+m)\log m)$ time.

Example 4.113 (continued) *For the given instance,* LPT *returns a pure Nash equilibrium $\mathbf{L} = \langle 1,2,3,3,1,2,1\rangle$ with social cost*

$$\mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{L}) \quad = \quad 11^2 + 8^2 + 8^2 \quad = \quad 249$$

whereas the optimum assignment $\langle 1,2,1,2,3,3,3\rangle$ has social cost

$$\mathsf{OPT}_{x^2}(\mathbf{w},3) \quad = \quad 9^2 + 9^2 + 9^2 \quad = \quad 243$$

(see Figure 4.14). Thus,

$$\frac{\mathsf{SC}_{\infty}(\mathbf{w},3,\mathbf{L})}{\mathsf{OPT}_{\infty}(\mathbf{w},3)} \quad = \quad \frac{249}{243} .$$

Note that for this instance, the bound in Theorem 4.116 reduces to $\frac{25}{24}$.

4.8.1.2 Computation of Best Nash Equilibria

Clearly, it is easy to compute a best pure Nash equilibrium in case of identical users (see Proposition 4.115). For arbitrary users, the fact that selfish steps do not increase social cost (see Proposition 4.114, page 166) enables us to prove that for any instance $(\mathbf{w},m)$ there always exists a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_{\pi^d(x)}(\mathbf{w},m,\mathbf{L}) = \mathsf{OPT}_{\pi^d(x)}(\mathbf{w},m)$ (Proposition 4.117, page 168). In the same way as in Theorem 4.22 (page 104), we can show by reduction from BIN PACKING that BEST PURE NE is $\mathcal{NP}$-complete even for $m = 2$ links (Theorem 4.118, page 168). This shows that BEST PURE NE is $\mathcal{NP}$-complete in the strong sense, proving that there exists no pseudo-polynomial algorithm to solve BEST PURE NE. However, we can give such an algorithm for constant m (Theorem 4.119, page 168). Moreover, if $\pi^d(x) = x^d$, then we can use any approximation algorithm for scheduling *jobs* on identical *machines* with respect to the d-norm [3, 7], and then nashify via rule MAX WEIGHT JOB, using $O(nm)$ time (see Proposition 4.46, page 115), in order to get an approximation algorithm of the same quality for BEST PURE NE. In particular, the PTAS of Alon *et al.* [3] yields a PTAS for BEST PURE NE (Theorem 4.120, page 168).

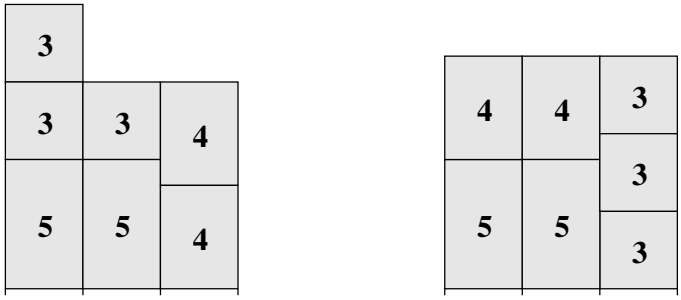


Figure 4.14: Pure Nash equilibrium $\mathbf{L} = \langle 1,2,3,3,1,2,1 \rangle$ returned by LPT applied to the instance in Example 4.113 (page 166) (left hand side), and an optimum assignment $\langle 1,2,1,2,3,3,3 \rangle$ of this instance (right hand side).

Proposition 4.117 *Consider the model of arbitrary users and identical links. Then, for any instance* $(\mathbf{w},m)$*, there exists a pure Nash equilibrium* $\mathbf{L}$ *with* $\mathsf{SC}_{\pi^d(x)}(\mathbf{w},m,\mathbf{L}) = \mathsf{OPT}_{\pi^d(x)}(\mathbf{w},m)$.

Proof: Fix any instance $(\mathbf{w},m)$. By Proposition 4.114 (page 166), selfish steps do not increase social cost. Thus, starting with an optimum assignment, every sequence of (not necessarily greedy) selfish steps ends in a pure Nash equilibrium $\mathbf{L}$ with $\mathsf{SC}_\infty(\mathbf{w},m,\mathbf{L}) \leq \mathsf{OPT}_\infty(\mathbf{w},m)$, as needed. ■

Theorem 4.118 *Consider the model of arbitrary users and identical links. Then,* BEST PURE NE *is* $\mathcal{NP}$*-complete even for* $m = 2$.

Theorem 4.119 *Consider the model of arbitrary users and identical links. Then, there exists a pseudo-polynomial-time algorithm for* m-BEST PURE NE.

Proof: The proof of Theorem 4.104 (page 161) can easily be adapted to this setting. ■

Theorem 4.120 *Consider the model of arbitrary users, identical links and polynomial cost function* $\pi^d(x) = x^d$*. Then, there exists a* PTAS *for* BEST PURE NE.

4.8.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

Clearly, it is also easy to compute a worst pure Nash equilibrium in case of identical users (see Proposition 4.115, page 167). In the same way as in Theorem 4.26 (page 106), we can show by reduction from BIN PACKING that WORST PURE NE is $\mathcal{NP}$-complete even for $m = 3$ links (Theorem 4.121, page 169). Clearly, this also shows that WORST PURE NE is $\mathcal{NP}$-complete in the strong sense. Thus, there exists no pseudo-polynomial algorithm to solve WORST PURE NE. For constant m, however, we can give such an algorithm (Theorem 4.122, page 169).

For identical users, the price of anarchy is 1 (see Proposition 4.115, page 167). We now shed light on the price of anarchy for the case of arbitrary users. For a given instance $(\mathbf{w},m)$, call a user $i \in [n]$ **bursty** if $w_i > \frac{W}{m}$. Intuitively, the traffic of a bursty user exceeds the fair share

of traffic for a link. Say that an instance $(\mathbf{w}, m)$ is **bursty** if some user $i \in [n]$ is bursty; else, $(\mathbf{w}, m)$ is **non-bursty**. We first prove that a bursty user is solo in both optimum assignments and pure Nash equilibria (Lemma 4.123). Then, roughly speaking, we prove that link loads are balanced in a non-bursty Nash equilibrium (Lemma 4.124, page 170). After proving a third technical but simple result (Proposition 4.125, page 171), we proceed by using these three results to prove that the price of anarchy for pure Nash equilibria is $\frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}} \left(\frac{d-1}{d}\right)^d$ if the polynomial cost function is $\pi^d(x) = x^d$ (Theorem 4.126, page 171). This generalizes a result in [102] where this bound is proved for the case $d = 2$ (Corollary 4.127, page 176).

Theorem 4.121 *Consider the model arbitrary users and identical links. Then,* WORST PURE NE *is $\mathcal{NP}$-complete even for $m = 3$.*

Theorem 4.122 *Consider the model of arbitrary users and identical links. Then, there exists a pseudo-polynomial-time algorithm for m-*WORST PURE NE.

Proof: The proof of Theorem 4.105 (page 161) can easily be adapted to this setting. ■

Lemma 4.123 *Consider the model of arbitrary users and identical links. Then, a bursty user is solo in both optimum assignments and pure Nash equilibria.*

Proof: Since, by definition of polynomial social cost, all coefficients are non-negative, Equation (4.13) (page 88) implies that it suffices to show the claim for all polynomial cost functions $\pi^t(x) = x^t$ with $t \in [d]$. So, let $t \in [d]$ be arbitrary, and fix an instance $(\mathbf{w}, m)$ with bursty user $i_1 \in [n]$.

(1.) Consider first an optimum assignment $\mathbf{Q} = \langle q_1, \ldots, q_n \rangle$. Note that

$$\delta_{q_{i_1}}(\mathbf{Q}) \;\geq\; w_{i_1} \;>\; \frac{W}{m}\,.$$

Since $\sum_{j \in [m]} \delta_j(\mathbf{Q}) = W$, there is some other link $j \in [m]$ with $j \neq q_{i_1}$ such that

$$\delta_j(\mathbf{Q}) \;<\; \frac{W}{m}\,.$$

Assume, by way of contradiction, that some user $i_2 \neq i_1$ is assigned to link q_{i_1}. Modify $\mathbf{Q}$ to obtain $\mathbf{Q}'$ by switching user i_2 to link j. Then,

$$\begin{aligned}
&\mathsf{SC}_{x^t}(\mathbf{w}, m, \mathbf{Q}') - \mathsf{SC}_{x^t}(\mathbf{w}, m, \mathbf{Q}) \\
&\quad= \quad \left(\delta_{q_{i_1}}(\mathbf{Q}')\right)^t + \left(\delta_j(\mathbf{Q}')\right)^t - \left(\delta_{q_{i_1}}(\mathbf{Q})\right)^t - \left(\delta_j(\mathbf{Q})\right)^t \\
&\quad\leq \quad w_{i_1}^t + \left(\delta_j(\mathbf{Q}) + w_{i_2}\right)^t - (w_{i_1} + w_{i_2})^t - \left(\delta_j(\mathbf{Q})\right)^t \\
&\quad\overset{\delta_j(\mathbf{Q}) < w_{i_1}}{<} \quad w_{i_1}^t + (w_{i_1} + w_{i_2})^t - (w_{i_1} + w_{i_2})^t - w_{i_1}^t \\
&\quad= \quad 0\,.
\end{aligned}$$

Since $\mathbf{Q}$ is optimum, $\mathsf{SC}_{x^t}(\mathbf{w}, m, \mathbf{Q}') \geq \mathsf{SC}_{x^t}(\mathbf{w}, m, \mathbf{Q})$, a contradiction.

(2.) Consider now any arbitrary pure Nash equilibrium $\mathbf{L} = \langle \ell_1, \ldots, \ell_n \rangle$. Note that

$$\delta_{\ell_{i_1}}(\mathbf{L}) \;\geq\; w_{i_1} \;>\; \frac{W}{m}\,.$$

Since $\sum_{j\in[m]} \delta_j(\mathbf{L}) = W$, there is some other link $j \in [m]$ with $j \neq \ell_{i_1}$ such that

$$\delta_j(\mathbf{L}) \;<\; \frac{W}{m}\,.$$

Assume, by way of contradiction, that some user $i_2 \neq i_1$ is assigned to link ℓ_{i_1}. Then,

$$\lambda_{i_2}(\mathbf{L}) \;\geq\; w_{i_1} + w_{i_2} \;>\; \frac{W}{m} + w_{i_2}\,.$$

However, if user i_2 switches to link j, its individual cost becomes

$$\delta_j(\mathbf{L}) + w_{i_2} \;<\; \frac{W}{m} + w_{i_2}\,.$$

Since $\mathbf{L}$ is a Nash equilibrium,

$$\delta_j(\mathbf{L}) + w_{i_2} \;\geq\; \lambda_{i_2}(\mathbf{L}) \;>\; \frac{W}{m} + w_{i_2}\,,$$

a contradiction. ■

Lemma 4.124 *Consider the model of arbitrary users and identical links. Then, for any non-bursty instance* $(\mathbf{w}, m)$ *and associated pure Nash equilibrium* $\mathbf{L}$*, it is*

$$\delta_j(\mathbf{L}) \;\leq\; 2 \min_{\ell\in[m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\}$$

for each link $j \in [m]$.

Proof: Fix any non-bursty instance $(\mathbf{w}, m)$ and associated pure Nash equilibrium $\mathbf{L}$. Assume, by way of contradiction, that there is some link $j \in [m]$ such that

$$\delta_j(\mathbf{L}) \;>\; 2 \min_{\ell\in[m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\}$$

Take link j so that it maximizes $\delta_\ell(\mathbf{L})$ over all links $\ell \in [m]$. Clearly, $\delta_j(\mathbf{L}) \geq \frac{W}{m}$. Moreover, if $\delta_j(\mathbf{L}) = \frac{W}{m}$, then $\delta_\ell(\mathbf{L}) = \frac{W}{m}$ for all links $\ell \in [m]$ and the claim follows. Hence, assume that $\delta_j(\mathbf{L}) > \frac{W}{m}$. We proceed by case analysis.

(1.) Assume first that there is a solo user $i \in [n]$ on link j, so that $\delta_j(\mathbf{L}) = w_i$. Since link j maximizes latency, $w_i \geq \delta_\ell(\mathbf{L})$ for all links $\ell \in [m]$. Moreover, our assumption implies that

$$w_i \;>\; \min_{\ell\in[m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\}\,.$$

It follows that

$$m \cdot w_i \;>\; \sum_{\ell\in[m]} \delta_\ell(\mathbf{L}) \;=\; W\,,$$

or $w_i > \frac{W}{m}$. Since $(\mathbf{w}, m)$ is a non-bursty instance, $w_i \leq \frac{W}{m}$, a contradiction.

(2.) Assume now that at least two users are assigned to link j. Consider the smallest traffic w_i of some user $i \in [n]$ among all users assigned to link j. Then, clearly, $w_i \leq \frac{\delta_j(\mathbf{L})}{2}$. Hence, by assumption,

$$\delta_j(\mathbf{L}) - w_i \;\geq\; \frac{\delta_j(\mathbf{L})}{2} \;>\; \min_{\ell \in [m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\} \; .$$

So,

$$\min_{\ell \in [m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\} + w_i \;<\; \delta_j(\mathbf{L}) \; .$$

Since $\mathbf{L}$ is a Nash equilibrium,

$$\min_{\ell \in [m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\} + w_i \;\geq\; \delta_j(\mathbf{L}) \; ,$$

a contradiction.

Since we obtained a contradiction in all possible cases, the proof is now complete. ■

Proposition 4.125 *Let $x, y_1, y_2 \in \mathbb{R}$ with $0 < x \leq y_1 \leq y_2$, and let $d \geq 2$. Then,*

$$(y_1 - x)^d + (y_2 + x)^d \;>\; y_1^d + y_2^d \; .$$

Proof: Let

$$\begin{aligned} f(x) &= (y_1 - x)^d + (y_2 + x)^d \; , \\ f'(x) &= -d(y_1 - x)^{d-1} + d(y_2 + x)^{d-1} \; . \end{aligned}$$

Since $y_2 + x > y_1 - x$, it follows that $f'(x) > 0$ for all x within the given range. Thus, $f(x)$ is strictly increasing in x, and we get

$$f(x) \;>\; f(0) \;=\; y_1^d + y_2^d \; ,$$

as needed. ■

Theorem 4.126 *Consider the model of arbitrary users, identical links and polynomial cost function $\pi^d(x) = x^d$ with $d \geq 2$, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \;=\; \frac{(2^d - 1)^d}{(d-1)(2^d - 2)^{d-1}} \left(\frac{d-1}{d} \right)^d \; .$$

Proof: In the proof, we make use of the following notation. Consider an instance $(\mathbf{w}, m)$ and associated pure assignment $\mathbf{L}$. Fix a set of links $\mathcal{M}$, inducing a set of users $\mathcal{U}$ that are assigned by the assignment $\mathbf{L}$ to links in $\mathcal{M}$. Then, $\mathbf{w} \setminus \mathcal{U}$ and $m \setminus \mathcal{M}$ denote the vector and the number of links resulting from $\mathbf{w}$ and m, respectively, by eliminating the entries corresponding to users in $\mathcal{U}$ and links in $\mathcal{M}$, respectively. $\mathbf{L} \setminus (\mathcal{U}, \mathcal{M})$ denotes the assignment induced by these eliminations. Let

$$\Delta(\mathbf{L}) \;=\; \min_{\ell \in [m]} \{\delta_\ell(\mathbf{L}) \mid \delta_\ell(\mathbf{L}) > 0\} \; .$$

We first prove the upper bound. Consider any arbitrary instance $(\mathbf{w}, m)$ with associated pure Nash equilibrium $\mathbf{L} = \langle \ell_1, \ldots, \ell_n \rangle$ and optimum assignment $\mathbf{Q} = \langle q_1, \ldots, q_n \rangle$. If $n \leq m$, then every user is solo in $\mathbf{L}$, showing that $\mathbf{L}$ is optimum. So, assume $n > m$. Clearly, $\delta_\ell(\mathbf{L}) > 0$ for all $\ell \in [m]$. We distinguish between two cases:

(1.) The instance $(\mathbf{w}, m)$ is non-bursty:
Recall that in this case, by Lemma 4.124 (page 170), for each link $j \in [m]$,

$$\delta_j(\mathbf{L}) \;\leq\; 2\Delta(\mathbf{L})$$

So, transform the set of loads $\{\delta_\ell(\mathbf{L}) \mid \ell \in [m]\}$ into a new set of loads $\{\widehat{\delta}_\ell \mid \ell \in [m]\}$ as the output of the following repetitive procedure:

---

(1) **for** each link $\ell \in [m]$ **do**;
(2) $\quad\widehat{\delta}_\ell \leftarrow \delta_\ell(\mathbf{L})$;
(3) **while** there are distinct $j_1, j_2 \in [m]$ with $\Delta(\mathbf{L}) < \widehat{\delta}_{j_1} \leq \widehat{\delta}_{j_2} < 2\Delta(\mathbf{L})$ **do**
(4) $\quad\widehat{\delta}_{j_1} \leftarrow \widehat{\delta}_{j_1} - \min\{\widehat{\delta}_{j_1} - \Delta(\mathbf{L}), 2\Delta(\mathbf{L}) - \widehat{\delta}_{j_2}\}$;
(5) $\quad\widehat{\delta}_{j_2} \leftarrow \widehat{\delta}_{j_2} + \min\{\widehat{\delta}_{j_1} - \Delta(\mathbf{L}), 2\Delta(\mathbf{L}) - \widehat{\delta}_{j_2}\}$;

Intuitively, our transformation procedure chooses at each step two intermediate loads $\widehat{\delta}_{j_1}$ and $\widehat{\delta}_{j_2}$ (that is, two loads that are not yet pushed either to the upper or to the lower end of the interval of link loads). It transfers the (strictly) positive quantity $\min\left\{\widehat{\delta}_{j_1} - \Delta(\mathbf{L}), 2\Delta(\mathbf{L}) - \widehat{\delta}_{j_2}\right\}$ from the small load $\widehat{\delta}_{j_1}$ to the large load $\widehat{\delta}_{j_2}$. Clearly, each step of the procedure either pushes the small load $\widehat{\delta}_{j_1}$ to the lower end $\Delta(\mathbf{L})$ of the interval of link loads, or pushes the large load $\widehat{\delta}_{j_2}$ to the upper end $2\Delta(\mathbf{L})$ of the interval of link loads (or both). So, clearly, when the procedure terminates, there is at most one intermediate load. Hence, by reordering links, we obtain that for some integer $k \in [m-1] \cup \{0\}$ for each link $j \in [m]$,

$$\widehat{\delta}_j \;=\; \begin{cases} 2\Delta(\mathbf{L}) & \text{if } j \in [k], \\ (1+x)\Delta(\mathbf{L}) & \text{if } j = k+1, \\ \Delta(\mathbf{L}) & \text{if } j \in [m] \setminus [k+1], \end{cases}$$

where $0 \leq x \leq 1$.

Consider any individual step of our repetitive procedure. Then, clearly,

$$\left(\widehat{\delta}_{j_1} - \min\left\{\widehat{\delta}_{j_1} - \Delta(\mathbf{L}), 2\Delta(\mathbf{L}) - \widehat{\delta}_{j_2}\right\}\right)^d + \left(\widehat{\delta}_{j_2} + \min\left\{\widehat{\delta}_{j_1} - \Delta(\mathbf{L}), 2\Delta(\mathbf{L}) - \widehat{\delta}_{j_2}\right\}\right)^d$$
$$\overset{\text{Proposition 4.125 (page 171)}}{>} \left(\widehat{\delta}_{j_1}\right)^d - \left(\widehat{\delta}_{j_2}\right)^d.$$

Note that this transformation procedure maps a set of loads to a new set of loads, without explicitly mapping an instance to a new instance. However, for the sake of our analysis, we will also consider that the transformation procedure maps an instance $(\mathbf{w}, m)$ and a Nash equilibrium $\mathbf{L}$ to a new instance $(\widehat{\mathbf{w}}, m)$ and a new Nash equilibrium $\widehat{\mathbf{L}}$. Note also that this transformation preserves (at each of its steps) the sum of loads. Hence, it also

preserves the total load so that $W = \widehat{W}$. Hence,

$$\begin{aligned}
\mathsf{SC}_{x^d}(\mathbf{w}, m, \mathbf{L}) &\leq \mathsf{SC}_{x^d}(\widehat{\mathbf{w}}, m, \widehat{\mathbf{L}}) \\
&= \sum_{j\in[m]} \left(\delta_j(\widehat{\mathbf{L}})\right)^d \\
&= k(2\Delta(\mathbf{L}))^d + ((1+x)\Delta(\mathbf{L}))^d + (m-k-1)\Delta(\mathbf{L})^d \\
&= \left(m + (2^d-1)k - 1 + (1+x)^d\right)\Delta(\mathbf{L})^d \,.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\mathsf{OPT}_{x^d}(\mathbf{w}, m) &\geq \frac{\left(\frac{W}{m}\right)^d}{m} \\
&= \frac{\widehat{W}^d}{m^{d-1}} \\
&= \frac{\left(\sum_{j\in[m]} \delta_j(\widehat{\mathbf{L}})\right)^d}{m^{d-1}} \\
&= \frac{(m+k+x)^d \Delta(\mathbf{L})^d}{m^{d-1}} \,.
\end{aligned}$$

It follows that

$$\mathsf{PoA} \leq \frac{(m + (2^d-1)k - 1 + (1+x)^d)m^{d-1}}{(m+k+x)^d} \,.$$

Define the real function

$$f(k) = \frac{(m + (2^d-1)k - 1 + (1+x)^d)m^{d-1}}{(m+k+x)^d}$$

of a real variable k (the quantity x is taken as a parameter, while m is a fixed constant). To maximize the function $f(k)$, observe that the first and second derivatives of $f(k)$ are

$$f'(k) = \frac{(2^d-1)m^{d-1}}{(m+k+x)^d} - \frac{(m + (2^d-1)k - 1 + (1+x)^d)m^{d-1}d}{(m+k+x)^{d+1}}$$

and

$$\begin{aligned}
f''(k) &= -\frac{2(2^d-1)m^{d-1}d}{(m+k+x)^{d+1}} + \frac{(m + (2^d-1)k - 1 + (1+x)^d)m^{d-1}(d^2+d)}{(m+k+x)^{d+2}} \\
&= \frac{m^{d-1}d[-2(2^d-1)(m+k+x) + (m + (2^d-1)k - 1 + (1+x)^d)(d+1)]}{(m+x+1)^{d+2}} \\
&= \frac{m^{d-1}d[(2^d-1)(d-1)k - 2(2^d-1)(m+x) + (m - 1 + (1+x)^d)(d+1)]}{(m+x+1)^{d+2}},
\end{aligned}$$

respectively. The only root of $f'(k)$ is

$$r = \frac{(2^d-1)(m+x) + d(-m+1-(1+x)^d)}{(2^d-1)(d-1)} \,.$$

For $k = r$, the second derivative evaluates to

$$\begin{aligned} f''(r) &= \frac{m^{d-1}d\left[(2^d-1)(d-1)r - 2(2^d-1)(m+x) + (m-1+(1+x)^d)(d+1)\right]}{(m+r+x)^{d+2}} \\ &= \frac{m^{d-1}d\left[-(2^d-1)(m+x) + m - 1 + (1+x)^d\right]}{(m+r+x)^{d+2}} \\ &= \frac{m^{d-1}d\left[-m(2^d-2) - (2^d-1)x + (1+x)^d - 1\right]}{(m+r+x)^{d+2}}. \end{aligned}$$

Since $-(2^d-1)x + (1+x)^d \le 2^d$ holds for all $0 \le x \le 1$, and since $m \ge 2$, it follows that $f''(r) < 0$. Thus, r is a local maximum of the function $f(k)$. Since $f(k)$ is a continuous function with a single extreme point that is a local maximum, it follows that

$$\begin{aligned} f(k) &\le f(r) \\ &= \frac{\left[m + \frac{2^d-1}{d-1}(m+x) + \frac{d}{d-1}(-m+1-(1+x)^d) - 1 + (1+x)^d\right] m^{d-1}}{\left[m + \frac{1}{d-1}(m+x) + \frac{d}{(d-1)(2^d-1)}(-m+1-(1+x)^d) + x\right]^d} \\ &= \frac{\left[m\frac{2^d-2}{d-1} + x\frac{2^d-1}{d-1} - (1+x)^d\frac{1}{d-1} + \frac{1}{d-1}\right] m^{d-1}}{\left[m(\frac{d}{d-1} - \frac{d}{(d-1)(2^d-1)}) + x\frac{d}{d-1} - (1+x)^d\frac{d}{(d-1)(2^d-1)} + \frac{d}{(d-1)(2^d-1)}\right]^d} \\ &= \frac{(2^d-1)^d}{(d-1)}\left(\frac{d-1}{d}\right)^d \cdot \frac{\left[m(2^d-2) + x(2^d-1) - (1+x)^d + 1\right] m^{d-1}}{\left[m(2^d-2) + x(2^d-1) - (1+x)^d + 1\right]^d} \\ &= \frac{(2^d-1)^d}{(d-1)}\left(\frac{d-1}{d}\right)^d \cdot \frac{m^{d-1}}{\left[m(2^d-2) + x(2^d-1) - (1+x)^d + 1\right]^{d-1}}. \end{aligned}$$

Note that the minimum value of the function $h(x) = x(2^d-1) - (1+x)^d$ for $x \in [0,1]$ is $h(0) = h(1) = -1$. Thus,

$$\begin{aligned} f(k) &\le \frac{(2^d-1)^d}{(d-1)}\left(\frac{d-1}{d}\right)^d \cdot \frac{m^{d-1}}{\left[m(2^d-2)\right]^{d-1}} \\ &= \frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}}\left(\frac{d-1}{d}\right)^d, \end{aligned}$$

as needed.

(2.) The instance $(\mathbf{w}, m)$ is bursty:
Denote $\mathcal{U}$ the (non-empty) set of bursty users. Recall that, by Lemma 4.123 (page 169), $\mathcal{U}$ induces sets of solo links $\mathcal{M}_{\mathbf{L}}$ and $\mathcal{M}_{\mathbf{Q}}$ for the Nash equilibrium $\mathbf{L}$ and the optimum assignment $\mathbf{Q}$, respectively, so that $|\mathcal{M}_{\mathbf{L}}| = |\mathcal{U}|$ and $|\mathcal{M}_{\mathbf{Q}}| = |\mathcal{U}|$. Since links are identical, we assume that $\mathcal{M}_{\mathbf{L}} = \mathcal{M}_{\mathbf{Q}} = \mathcal{M}$, with $|\mathcal{M}| \ge 1$. So,

$$\begin{aligned} \mathsf{SC}_{x^d}(\mathbf{w}, m, \mathbf{L}) &= \sum_{j \in \mathcal{M}} \left(\delta_j(\mathbf{L})\right)^d + \mathsf{SC}_{x^d}(\mathbf{w} \setminus \mathcal{U}, m \setminus \mathcal{M}, \mathbf{L} \setminus (\mathcal{U}, \mathcal{M})) \\ &= \sum_{i \in \mathcal{U}} w_i^d + \mathsf{SC}_{x^d}(\mathbf{w} \setminus \mathcal{U}, m \setminus \mathcal{M}, \mathbf{L} \setminus (\mathcal{U}, \mathcal{M})) \end{aligned}$$

and

$$\begin{aligned}\mathsf{OPT}_{x^d}(\mathbf{w},m) &= \mathsf{SC}_{x^d}(\mathbf{w},m,\mathbf{Q}) \\ &= \sum_{j\in\mathcal{M}} \big(\delta_j(\mathbf{L})\big)^d + \mathsf{SC}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M}, \mathbf{Q}\setminus(\mathcal{U},\mathcal{M})) \\ &= \sum_{i\in\mathcal{U}} w_i^d + \mathsf{SC}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M}, \mathbf{Q}\setminus(\mathcal{U},\mathcal{M}))\,.\end{aligned}$$

Note first that the assignment $\mathbf{L}\setminus(\mathcal{U},\mathcal{M})$ is a Nash equilibrium for the instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$. Moreover, since $\mathbf{Q}$ is an optimum assignment for the instance $(\mathbf{w},m)$, it follows that $\mathbf{Q}\setminus(\mathcal{U},\mathcal{M})$ is an optimum assignment for the instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$, so that

$$\mathsf{SC}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M}, \mathbf{Q}\setminus(\mathcal{U},\mathcal{M})) = \mathsf{OPT}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})\,.$$

Thus,

$$\mathsf{OPT}_{x^d}(\mathbf{w},m) = \sum_{i\in\mathcal{U}} w_i^d + \mathsf{OPT}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})\,.$$

It follows that

$$\begin{aligned}\frac{\mathsf{SC}_{x^d}(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_{x^d}(\mathbf{w},m)} &= \frac{\sum_{i\in\mathcal{U}} w_i^d + \mathsf{SC}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M}, \mathbf{L}\setminus(\mathcal{U},\mathcal{M}))}{\sum_{i\in\mathcal{U}} w_i^d + \mathsf{OPT}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})} \\ &\le \frac{\mathsf{SC}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M}, \mathbf{L}\setminus(\mathcal{U},\mathcal{M}))}{\mathsf{OPT}_{x^d}(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})}\,.\end{aligned}$$

So, consider the instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$ and the associated pure Nash equilibrium $\mathbf{L}\setminus(\mathcal{U},\mathcal{M})$. There are two possibilities depending on whether the instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$ is bursty or not.

- Assume first that the smaller instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$ is non-bursty. Then, we are reduced to the previous case of non-bursty instances, and the upper bound follows inductively.
- Assume now that the smaller instance $(\mathbf{w}\setminus\mathcal{U}, m\setminus\mathcal{M})$ is bursty. We repeatedly identify the set of bursty users for the smaller instance, and we reduce this smaller instance to an even smaller instance that may be bursty or non-bursty. This procedure eventually yields a non-bursty instance (even the trivial one with one user), and the claim for the original bursty instance follows inductively.

The proof of the upper bound is now complete. We continue to prove the lower bound. Let $r = (2^d - d - 1)$, and consider $m = (2^d-1)(d-1)$ links, $2r$ identical users with traffic 1 and $m(m-r)$ users with traffic $\frac{1}{m}$. Clearly, the users with traffic $\frac{1}{m}$ can be evenly distributed to $m-r$ links. Thus, the pure assignment $\mathbf{L}$ in which users with traffic 1 are evenly distributed to r links, and the remaining users are evenly distributed to the remaining $m-r$ links, is a Nash equilibrium with social cost

$$\begin{aligned}\mathsf{SC}_{x^d}(\mathbf{w},m,\mathbf{L}) &= 2^d\cdot r + 1^d\cdot(m-r) \\ &= 2^d(2^d-d-1) + (2^d-1)(d-1) - (2^d-d-1) \\ &= (2^d-1)(2^d-2)\,.\end{aligned}$$

Moreover, we can evenly distribute the total traffic $m+r$ to the links in the following way: assign the $2r$ users with traffic 1 to $2r$ distinct links (it is easy to see that $2r < m$ for all $d \geq 2$), evenly distribute $m(m-2r)$ users with traffic $\frac{1}{m}$ to the remaining $m-2r$ links, and evenly distribute the remaining mr users with traffic $\frac{1}{m}$ to all m links. We get

$$\begin{aligned}
\mathsf{OPT}_{x^d}(\mathbf{w},m) &= \left(\frac{m+r}{m}\right)^d m \\
&= \frac{\left[(2^d-1)(d-1)+(2^d-d-1)\right]^d}{\left[(2^d-1)(d-1)\right]^{d-1}} \\
&= \frac{\left[(2^d-2)d\right]^d}{\left[(2^d-1)(d-1)\right]^{d-1}} \\
&= \frac{(d-1)(2^d-2)^d}{(2^d-1)^{d-1}} \cdot \left(\frac{d}{d-1}\right)^d .
\end{aligned}$$

Thus,

$$\mathsf{PoA} \geq \frac{\mathsf{SC}_{x^d}(\mathbf{w},m,\mathbf{L})}{\mathsf{OPT}_{x^d}(\mathbf{w},m)} = \frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}} \left(\frac{d-1}{d}\right)^d ,$$

as needed. ■

Corollary 4.127 *Consider the model of arbitrary users and identical links, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} = \frac{9}{8} .$$

Example 4.128 *Consider the following instance $(\mathbf{w},3)$: We have $n=8$ arbitrary users, $m=3$ identical links and polynomial cost function $\pi^2(x)=x^2$. There are two users with traffics $w_1=w_2=3$, and six users with traffics $w_i=1$ for all $i \in [8] \setminus [2]$. The pure Nash equilibrium $\mathbf{L}=\langle 1,1,2,2,2,3,3,3\rangle$ has social cost*

$$\mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{L}) = 6^2+3^2+3^2 = 54$$

whereas the optimum assignment $\langle 1,2,1,2,3,3,3,3\rangle$ has social cost

$$\mathsf{OPT}_{x^2}(\mathbf{w},3) = 4^2+4^2+4^2 = 48 .$$

(see Figure 4.15). Thus,

$$\frac{\mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{L})}{\mathsf{OPT}_{x^2}(\mathbf{w},3)} = \frac{9}{8} .$$

4.8.2 Fully Mixed Nash Equilibria

We now turn our attention to fully mixed Nash equilibria. We first investigate the polynomial cost function $\pi^2(x)=x^2$ in Subsection 4.8.2.1. In Subsection 4.8.2.2, we then consider arbitrary polynomial cost functions.

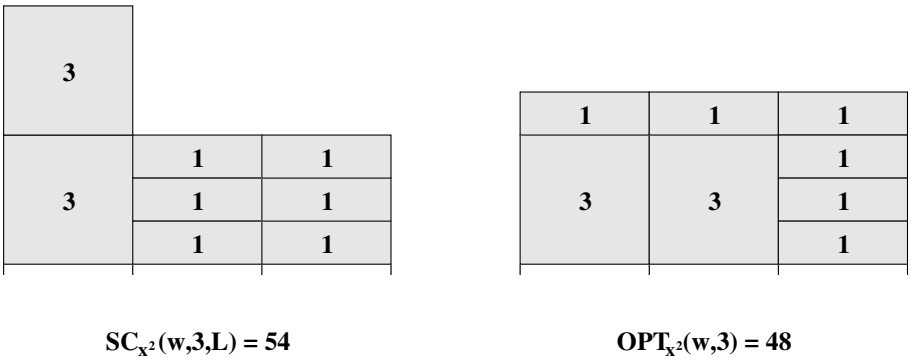


Figure 4.15: Pure Nash equilibrium $\mathbf{L} = \langle 1,1,2,2,2,3,3,3\rangle$ (left hand side), and optimum assignment $\langle 1,2,1,2,3,3,3,3\rangle$ of the instance in Example 4.128 (page 176) (right hand side).

4.8.2.1 Polynomial Cost Function $\pi^2(x) = x^2$

For the polynomial cost function $\pi^2(x) = x^2$, Lücking *et al.* [102] gave a closed formula for the social cost of the (unique) fully mixed Nash equilibrium (Theorem 4.129). If the users are also identical, then this formula simplifies significantly (Corollary 4.130). Moreover, Lücking *et al.* [102] proved that for identical users the FMNE Conjecture is valid (Theorem 4.131). Combining these results yields a tight bound on the price of anarchy (Theorem 4.132).

Theorem 4.129 (Lücking *et al.* [102]) *Consider the model of arbitrary users, identical links and polynomial cost function $\pi^2(x) = x^2$. Then,*

$$\mathsf{SC}_{x^2}(\mathbf{w},m,\mathbf{F}) \quad = \quad W_1 + \frac{2}{m}\cdot W_2 \ ,$$

where $W_1 = \sum_{i\in[n]} w_i^2$ and $W_2 = \sum_{i,k\in[n]:i<k} w_i w_k$.

Corollary 4.130 (Lücking *et al.* [102]) *Consider the model of identical users, identical links and polynomial cost function $\pi^2(x) = x^2$. Then,*

$$\mathsf{SC}_{x^2}(\mathbf{w},m,\mathbf{F}) \quad = \quad \frac{n(n+m-1)}{m} \ .$$

Theorem 4.131 (Lücking *et al.* [102]) *Consider the model of identical users, identical links and polynomial cost function $\pi^2(x) = x^2$. Then, the* FMNE *Conjecture is valid.*

Theorem 4.132 (Lücking *et al.* [102]) *Consider the model of identical users, identical links and polynomial cost function $\pi^2(x) = x^2$. Then,*

$$\mathsf{PoA} \quad = \quad 2 - \max\left\{\frac{1}{n},\frac{1}{m}\right\} \ .$$

Example 4.113 (continued) *For the given instance, the social cost of the fully mixed Nash equilibrium* **F** *by Theorem 4.129 (page 177) is*

$$\mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{F}) \quad = \quad 109 + \frac{2}{3}\cdot 310 \quad = \quad \frac{947}{3}$$

whereas the social cost of the worst pure Nash equilibrium $\mathbf{L} = \langle 1,2,3,2,1,2,1 \rangle$ *is*

$$\mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{L}) \;=\; 249 \;<\; \mathsf{SC}_{x^2}(\mathbf{w},3,\mathbf{F})\,.$$

4.8.2.2 Arbitrary Polynomial Cost Functions

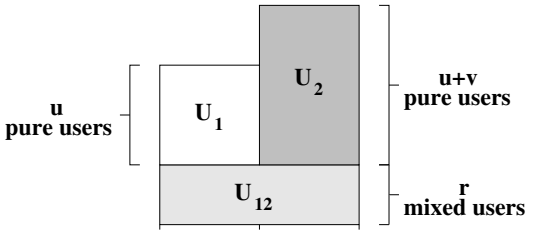
Identical Users and Two Identical Links. We next turn our attention to arbitrary polynomial cost functions, identical users and two identical links. We first use Proposition 4.3 (page 81) and Lemma 4.133 to prove the FMNE Conjecture (Theorem 4.134). Making use of this result, we then prove the tight bound $\frac{1}{2}\left(2^{d-1}+1\right)$ on the price of anarchy for the special case where the polynomial cost function is $\pi^d(x) = x^d$ (Theorem 4.136, page 182). We close by showing the upper bound $\frac{1}{2}\left(2^d + d - 1\right)$ on the price of anarchy for general polynomial cost functions (Corollary 4.137, page 183).

Lemma 4.133 (Lücking *et al.* [102]) *Consider the model of arbitrary users and identical links, and fix any instance* $(\mathbf{w},m)$ *and associated Nash equilibrium* $\mathbf{P}$. *If* $\mathsf{view}_{\mathbf{P}}(j_1) \subsetneq \mathsf{view}_{\mathbf{P}}(j_2)$ *for* $j_1, j_2 \in [m]$, *then* $\Lambda_{j_1}(\mathbf{P}) > \Lambda_{j_2}(\mathbf{P})$.

Theorem 4.134 *Consider the model of identical users and two identical links. Then, the* FMNE *Conjecture is valid.*

Proof: Fix any instance $(n,2)$, associated Nash equilibrium $\mathbf{P}$ and fully mixed Nash equilibrium $\mathbf{F}$. We can identify three sets of users in $\mathbf{P}$:

$$\begin{aligned}
\mathcal{U}_1 &= \{i \in [n] \mid \mathsf{support}_{\mathbf{P}}(i) = \{1\}\} \\
\mathcal{U}_2 &= \{i \in [n] \mid \mathsf{support}_{\mathbf{P}}(i) = \{2\}\} \\
\mathcal{U}_{12} &= \{i \in [n] \mid \mathsf{support}_{\mathbf{P}}(i) = \{1,2\}\}
\end{aligned}$$



Without loss of generality, let $|\mathcal{U}_1| \leq |\mathcal{U}_2|$.
Let $u = |\mathcal{U}_1|$, $v = |\mathcal{U}_2| - u$ and $r = |\mathcal{U}_{12}|$.

The proof of the theorem is structured as follows: We first prove that the claim holds if $\mathbf{P}$ is a pure Nash equilibrium. Then, we consider the case where $u = 0$. By Equation (4.13) (page 88), it suffices to show that the claim holds for all polynomial cost functions $\pi^t(x) = x^t$ with $t \in [d]$. For the case where $u > 0$, we introduce a Nash equilibrium $\mathbf{Q}$ as a perturbation of $\mathbf{F}$ with a special structure that is similar to the structure of $\mathbf{P}$ such that $\mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{Q}) \leq \mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{F})$. Due to this special structure, $\mathbf{Q}$ can be compared with $\mathbf{P}$ more easily. To compare the two, we use the fact that the claim holds for $u = 0$. We now continue with the details of the formal proof.

(1.) $\mathbf{P}$ is a pure Nash equilibrium: In this case, $r = 0$. Clearly, by definition of Nash equilibrium, $|\mathcal{U}_1|$ and $|\mathcal{U}_2|$ differ by 1 if n is odd, and $|\mathcal{U}_1| = |\mathcal{U}_2|$ otherwise. Thus, each pure Nash equilibrium $\mathbf{P}$ has unique social cost $\mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{P}) = \mathsf{OPT}_{\pi^d(x)}(n,2)$, proving the claim for pure Nash equilibria. So, assume $\mathbf{P}$ to be a (non-pure) Nash equilibrium.

(2.) Case $u = 0$: Clearly, $\mathsf{view}_{\mathbf{P}}(1) \subsetneq \mathsf{view}_{\mathbf{P}}(2)$. By Lemma 4.133 (page 178), this implies $\Lambda_1(\mathbf{P}) > \Lambda_2(\mathbf{P})$. If $r = 1$, then, by definition of Nash equilibrium, $|\mathcal{U}_2| = |\mathcal{U}_1| = u = 0$. Thus, the total number of users is $n = 1$, a contradiction to the assumption that $n \geq 2$. So, assume that $r \in [n-1] \setminus \{1\}$ mixed users are assigned to both links, and that $n - r$ pure users are assigned to link 2.

Consider any arbitrary mixed user $i \in \mathcal{U}_{12}$. The definition of Nash equilibrium implies

$$\begin{aligned} \Lambda_1(\mathbf{P}) - p_{i1} + 1 &= \Lambda_2(\mathbf{P}) - p_{i2} + 1 \\ &\overset{p_{i_2}=1-p_{i_1}}{=} \Lambda_2(\mathbf{P}) - (1 - p_{i1}) + 1 \\ &= \Lambda_2(\mathbf{P}) + p_{i1} \, . \end{aligned}$$

Thus,

$$p_{i1} = \frac{\Lambda_1(\mathbf{P}) - \Lambda_2(\mathbf{P}) + 1}{2} \, .$$

Since this holds for every mixed user, we write p_1 and p_2 for the probabilities of every mixed user on link 1 and 2, respectively. The definition of Nash equilibrium implies

$$\begin{aligned} (r-1) \cdot p_1 + 1 &= (r-1) \cdot p_2 + n - r + 1 \\ &\overset{p_{i_2}=1-p_{i_1}}{=} (r-1) \cdot (1 - p_1) + n - r + 1 \, . \end{aligned}$$

Thus,

$$\begin{aligned} p_1 &= \frac{n-1}{2(r-1)} &&= \frac{1}{2} + \frac{n-r}{2(r-1)} \, , \\ p_2 &= (1 - p_1) &&= \frac{1}{2} - \frac{n-r}{2(r-1)} \, , \end{aligned}$$

and we can write

$$\begin{aligned} \Lambda_1(\mathbf{P}) &= r \cdot p_1 \\ &= r \cdot \left(\frac{1}{2} + \frac{n-r}{2(r-1)} \right) \\ &= \frac{n}{2} - \frac{n-r}{2} + \frac{nr - r^2}{2(r-1)} \\ &= \frac{n}{2} + \frac{n-r}{2(r-1)} \\ &= \alpha + \beta \, , \end{aligned}$$

where

$$\begin{aligned} \alpha &= \frac{n}{2} \, , \\ \beta &= \frac{n-r}{2(r-1)} \end{aligned}$$

with $0 < \beta < \frac{1}{2}$. Moreover,

$$\Lambda_2(\mathbf{P}) \;=\; n - \Lambda_1(\mathbf{P}) \;=\; \alpha - \beta\,.$$

Since, by definition of polynomial social cost, all coefficients are non-negative, Equation (4.13) (page 88) implies that it suffices to show the claim for all polynomial cost functions $\pi^t(x) = x^t$ with $t \in [d]$. Fix any such $t \in [d]$. Denote

$$\begin{aligned} \widetilde{p}_1 &= \frac{\Lambda_1(\mathbf{P})}{r} = \frac{\alpha+\beta}{r}\,, \\ \widetilde{p}_2 &= \frac{\Lambda_2(\mathbf{P})}{n} = \frac{\alpha-\beta}{n} \end{aligned}$$

the average probability on link 1 and 2, respectively. Since x^t is convex, we get

$$\begin{aligned} \mathsf{SC}_{x^t}(n,2,\mathbf{P}) &\overset{(4.14),\ page\ 88}{=} H((\underbrace{p_1,\ldots,p_1}_{r}),x^t) + H((\underbrace{p_2,\ldots,p_2}_{r},\underbrace{1,\ldots,1}_{n-r}),x^t) \\ &\overset{Lemma\ 4.2\ (page\ 81)}{\leq} H(\widetilde{p}_1,r,x^t) + H(\widetilde{p}_2,n,x^t) \\ &\overset{Proposition\ 4.3\ (page\ 81)}{=} \sum_{i\in[t]} \widetilde{p}_1^{\,i}\cdot S(t,i)\cdot r^{\underline{i}} + \sum_{i\in[t]} \widetilde{p}_2^{\,i}\cdot S(t,i)\cdot n^{\underline{i}} \\ &\overset{definition\ of\ \widetilde{p}_1,\widetilde{p}_2}{=} \sum_{i\in[t]} \left(\frac{\alpha+\beta}{r}\right)^i \cdot S(t,i)\cdot r^{\underline{i}} + \sum_{i\in[t]} \left(\frac{\alpha-\beta}{n}\right)^i \cdot S(t,i)\cdot n^{\underline{i}} \\ &= \sum_{i\in[t]} S(t,i)\cdot\left[(\alpha+\beta)^i\cdot\frac{r^{\underline{i}}}{r^i} + (\alpha-\beta)^i\cdot\frac{n^{\underline{i}}}{n^i}\right]\,. \end{aligned} \tag{4.53}$$

Moreover, we have

$$\begin{aligned} \mathsf{SC}_{x^t}(n,2,\mathbf{F}) &\overset{(4.14),\ page\ 88}{=} H((f_{11},\ldots,f_{n1}),x^t) + H((f_{12},\ldots,f_{n2}),x^t) \\ &\overset{Theorem\ 4.36\ (page\ 112)}{=} 2\cdot H(\frac{1}{2},n,x^t) \\ &= 2\cdot H(\frac{\alpha}{n},n,x^t) \\ &\overset{Proposition\ 4.3\ (page\ 81)}{=} \sum_{i\in[t]} S(t,i)\left[2\alpha^i\cdot\frac{n^{\underline{i}}}{n^i}\right]\,. \end{aligned} \tag{4.54}$$

By Equations (4.53) and (4.54), it suffices to show the following claim:

Claim 4.135 *For all $i \geq 1$,*

$$\Delta \;=\; 2\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} - \left[(\alpha+\beta)^i\cdot\frac{r^{\underline{i}}}{r^i} + (\alpha-\beta)^i\cdot\frac{n^{\underline{i}}}{n^i}\right] \;\geq\; 0\,.$$

Proof: We prove the claim by induction on i, $i \geq 1$. For the basis case, let $i = 1$. We have

$$\Delta \;=\; 2\alpha - [\alpha+\beta+\alpha-\beta] \;=\; 0\,,$$

proving that the claim holds for the basis case. For the induction step, let $i \geq 2$, and assume that the claim holds for $(i-1)$. We have

$$\begin{aligned}\Delta &= 2\alpha^i \cdot \frac{\underline{n^i}}{n^i} - \Bigg[(\alpha+\beta)\cdot\frac{r-(i-1)}{r}\cdot\overbrace{(\alpha+\beta)^{i-1}\cdot\frac{\underline{r^{i-1}}}{r^{i-1}}}^{\geq 0} \\ &\qquad + (\alpha-\beta)\cdot\frac{n-(i-1)}{n}\cdot(\alpha-\beta)^{i-1}\cdot\frac{\underline{n^{i-1}}}{n^{i-1}}\Bigg] .\end{aligned}$$

Clearly,

$$\begin{aligned}(\alpha+\beta)\cdot\frac{r-(i-1)}{r} &= \left(\frac{(n-1)r}{2(r-1)}\right)\cdot\frac{r-(i-1)}{r} \\ &= \frac{n-1}{2}\cdot\frac{r-(i-1)}{r-1}\end{aligned}$$

is monotonic increasing in r for all $i \geq 2$. Thus,

$$\begin{aligned}\Delta &\overset{r<n}{\geq} 2\alpha^i \cdot \frac{\underline{n^i}}{n^i} - \Bigg[\alpha\cdot\frac{n-(i-1)}{n}\cdot(\alpha+\beta)^{i-1}\cdot\frac{\underline{r^{i-1}}}{r^{i-1}} \\ &\qquad + (\alpha-\beta)\cdot\underbrace{\frac{n-(i-1)}{n}\cdot(\alpha-\beta)^{i-1}\cdot\frac{\underline{n^{i-1}}}{n^{i-1}}}_{\geq 0}\Bigg] \\ &\overset{\beta>0}{\geq} 2\alpha^i\cdot\frac{\underline{n^i}}{n^i} \\ &\qquad - \left[\alpha\cdot\frac{n-(i-1)}{n}\cdot(\alpha+\beta)^{i-1}\cdot\frac{\underline{r^{i-1}}}{r^{i-1}} + \alpha\cdot\frac{n-(i-1)}{n}\cdot(\alpha-\beta)^{i-1}\cdot\frac{\underline{n^{i-1}}}{n^{i-1}}\right] \\ &= 2\alpha^i\cdot\frac{\underline{n^i}}{n^i} - \alpha\cdot\frac{n-(i-1)}{n}\cdot\left[(\alpha+\beta)^{i-1}\cdot\frac{\underline{r^{i-1}}}{r^{i-1}} + (\alpha-\beta)^{i-1}\cdot\frac{\underline{n^{i-1}}}{n^{i-1}}\right] \\ &\overset{Induction}{\geq} 2\alpha^i\cdot\frac{\underline{n^i}}{n^i} - \alpha\cdot\frac{n-(i-1)}{n}\cdot 2\alpha^{i-1}\cdot\frac{\underline{n^{i-1}}}{n^{i-1}} \\ &= 0 ,\end{aligned}$$

proving the inductive claim. ■

(3.) Case $u > 0$: Set $\widetilde{n} = r + v = n - 2u$. Moreover, denote $\widetilde{p}_{11}, \ldots, \widetilde{p}_{r1}$ the probabilities of the r mixed users on link 1, and denote $\widetilde{p}_{12}, \ldots, \widetilde{p}_{\widetilde{n}2}$ the probabilities of the r mixed and v pure users on link 2. Consider the following mixed Nash equilibrium $\mathbf{Q}$: On both links, there are u pure users, and the remaining $\widetilde{n} = n - 2u$ users are assigned to link 1 and 2 with probability $\frac{1}{2}$. Clearly, since $\pi^d(x)$ is convex,

$$\mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{Q}) \overset{\text{Lemma 4.2 (page 81)}}{\leq} \mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{F}) .$$

Again, it suffices to show that $\mathsf{SC}_{x^t}(n,2,\mathbf{P}) \leq \mathsf{SC}_{x^t}(n,2,\mathbf{Q})$ for all $t \in [d]$. Fix any such $t \in [d]$. If $\widetilde{n} = 1$, then

$$\begin{aligned}\mathsf{SC}_{x^t}(n,2,\mathbf{P}) &= \mathsf{SC}_{x^t}(n,2,\mathbf{Q}) \\ &= (u+1)^t + u^t ,\end{aligned}$$

proving the claim. So, assume $\widetilde{n} \geq 2$. We have

$$\begin{aligned} \mathsf{SC}_{x^t}(n,2,\mathbf{Q}) &\overset{(4.14),\ page\ 88}{=} H((q_{11},\ldots,q_{n1}),x^t)+H((q_{12},\ldots,q_{n2}),x^t) \\ &= 2\cdot H(\frac{1}{2},\widetilde{n},(x+u)^t) \\ &\overset{(4.14),\ page\ 88}{=} \mathsf{SC}_{(x+u)^t}(\widetilde{n},2,\widetilde{\mathbf{Q}}) \ , \end{aligned}$$

and

$$\begin{aligned} \mathsf{SC}_{x^t}(n,2,\mathbf{P}) &\overset{(4.14),\ page\ 88}{=} H((p_{11},\ldots,p_{n1}),x^t)+H((p_{12},\ldots,p_{n2}),x^t) \\ &= H((\widetilde{p}_{11},\ldots,\widetilde{p}_{r1}),(x+u)^t)+H((\widetilde{p}_{12},\ldots,\widetilde{p}_{\widetilde{n}2}),(x+u)^t) \\ &\overset{(4.14),\ page\ 88}{=} \mathsf{SC}_{(x+u)^t}(\widetilde{n},2,\widetilde{\mathbf{P}}) \ , \end{aligned}$$

where $\widetilde{\mathbf{Q}}$ is the fully mixed Nash equilibrium associated to the instance $(\widetilde{n},2)$, and $\widetilde{\mathbf{P}}$ is a mixed Nash equilibrium associated to the instance $(\widetilde{n},2)$ with the same structure as in case 1, that is, there are no pure users on link 1, v pure users on link 2, and r mixed users on both links. Since $(x+u)^t$ is a polynomial of degree $t \in [d]$ with non-negative coefficients, we can apply the first case, proving the claim. ■

Theorem 4.136 *Consider the model of identical users, two identical links and polynomial cost function $\pi^d(x) = x^d$. Then,*

$$\mathsf{PoA} \ = \ \frac{1}{2}\left(2^{d-1}+1\right) \ .$$

Proof: Fix any instance $(n,2)$. In Theorem 4.134 (page 178), we showed that the social cost of any Nash equilibrium is bounded from above by the social cost of the fully mixed Nash equilibrium. Thus, we only have to prove the claim for the (unique) fully mixed Nash equilibrium $\mathbf{F}$. By Theorem 4.36 (page 112), we have $f_{i1} = f_{i2} = \frac{1}{2}$ for all $i \in [n]$.

Upper bound: On the one hand,

$$\begin{aligned} \mathsf{SC}_{x^d}(n,2,\mathbf{F}) &\overset{(4.14),\ page\ 88}{=} 2\cdot H(\frac{1}{2},n,x^d) \\ &\overset{Proposition\ 4.3\ (page\ 81)}{=} 2\cdot\sum_{i\in[d]}\left(\frac{1}{2}\right)^i\cdot S(d,i)\cdot n^{\underline{i}} \ . \end{aligned}$$

On the other hand, since social cost is minimum if the users are evenly distributed to the two links, a trivial lower bound on the optimum is

$$\mathsf{OPT}_{x^d}(n,2) \ \geq \ 2\cdot\left(\frac{n}{2}\right)^d \ .$$

Thus, we get

$$\begin{aligned}
\frac{\mathsf{SC}_{x^d}(n,2,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,2)} &\leq \left(\frac{2}{n}\right)^d \cdot \sum_{i\in[d]} \left(\frac{1}{2}\right)^i \cdot S(d,i)\cdot n^{\underline{i}} \\
&\leq \left(\frac{2}{n}\right)^d \left(\frac{1}{2}\cdot S(d,1)\cdot n + \frac{1}{4}\sum_{2\leq i\leq d} S(d,i)\cdot n^{\underline{i}}\right) \\
&\overset{(4.8),\ page\ 80}{=} \left(\frac{2}{n}\right)^d \left(\frac{1}{2}\cdot S(d,1)\cdot n + \frac{1}{4}(n^d - S(d,1)\cdot n)\right) \\
&\overset{(4.5),\ page\ 80}{=} \left(\frac{2}{n}\right)^d \left(\frac{n}{2} + \frac{1}{4}n^d - \frac{n}{4}\right) \\
&= \frac{1}{2}\cdot 2^{d-1} + \frac{1}{2}\cdot\left(\frac{2}{n}\right)^{d-1} \\
&\overset{n\geq 2}{\leq} \frac{1}{2}\left(2^{d-1}+1\right) .
\end{aligned}$$

This completes the proof of the upper bound.
Lower bound: Let $n = 2$. Then, the social cost of the fully mixed Nash equilibrium is

$$\begin{aligned}
\mathsf{SC}_{x^d}(n,2,\mathbf{F}) &= 2\cdot\sum_{i\in[d]}\left(\frac{1}{2}\right)^i \cdot S(d,i)\cdot n^{\underline{i}} \\
&= 2\cdot\left(S(d,1) + \frac{1}{2}\cdot S(d,2)\right) \\
&\overset{(4.5),\ page\ 80,\ (4.6),\ page\ 80}{=} 2\cdot\left(1 + \frac{1}{2}\cdot(2^{d-1}-1)\right) \\
&= 2\cdot\left(\frac{1}{2}\cdot(2^{d-1}+1)\right) ,
\end{aligned}$$

whereas the social cost of an optimum assignment is 2, proving the lower bound. ■

Corollary 4.137 *Consider the model of identical users and two identical links. Then,*

$$\mathsf{PoA} \leq \frac{1}{2}\left(2^d + d - 1\right) .$$

Proof: Fix any instance $(n,2)$. By Theorem 4.134 (page 178), we only have to prove the claim for the (unique) fully mixed Nash equilibrium $\mathbf{F}$. Using the fact that $a_t \geq 0$ for all

$t \in [d] \cup \{0\}$, we get

$$\begin{aligned}
\frac{\mathsf{SC}_{\pi^d(x)}(n,2,\mathbf{F})}{\mathsf{OPT}_{\pi^d(x)}(n,2)} &\overset{\text{(4.13), page 88}}{=} \frac{a_0 + \sum_{t\in[d]} a_t \cdot \mathsf{SC}_{x^t}(n,2,\mathbf{F})}{a_0 + \sum_{t\in[d]} a_t \cdot \mathsf{OPT}_{x^t}(n,2)} \\
&\leq \frac{\sum_{t\in[d]} a_t \cdot \mathsf{SC}_{x^t}(n,2,\mathbf{F})}{\sum_{t\in[d]} a_t \cdot \mathsf{OPT}_{x^t}(n,2)} \\
&\leq \sum_{t\in[d]} \frac{a_t \cdot \mathsf{SC}_{x^t}(n,2,\mathbf{F})}{a_t \cdot \mathsf{OPT}_{x^t}(n,2)} \\
&= \sum_{t\in[d]} \frac{\mathsf{SC}_{x^t}(n,2,\mathbf{F})}{\mathsf{OPT}_{x^t}(n,2)} \\
&\overset{\text{Theorem 4.136 (page 182)}}{\leq} \sum_{t\in[d]} \frac{1}{2}(2^{t-1}+1) \\
&= \frac{1}{2}(2^d + d - 1)\,,
\end{aligned}$$

proving the claim. ■

Identical Users and an Arbitrary Number of Identical Links. We proceed to prove the validity of an approximate version of the FMNE Conjecture for the model of identical users and an *arbitrary number* of identical links (Theorem 4.138). Equipped with this result, we then prove the upper bound $\left(1+\frac{1}{n-1}\right)^d \cdot B_d$ on the price of anarchy for the special case where the polynomial cost function is $\pi^d(x) = x^d$ (Theorem 4.140, page 189), using similar techniques as in [58]. We close by showing the upper bound $\sum_{t\in[d]} \left(1+\frac{1}{n-1}\right)^t \cdot B_t$ on the price of anarchy for general polynomial cost functions (Corollary 4.141, page 191).

Theorem 4.138 *Consider the model of identical users and identical links. Then, for any instance* (n,m) *and associated Nash equilibrium* $\mathbf{P}$*, it is*

$$\mathsf{SC}_{\pi^d(x)}(n,m,\mathbf{P}) \;\leq\; \left(1+\frac{1}{n-1}\right)^d \cdot \mathsf{SC}_{\pi^d(x)}(n,m,\mathbf{F})\,.$$

Proof: Fix any instance (n,m), associated Nash equilibrium $\mathbf{P}$ and fully mixed Nash equilibrium $\mathbf{F}$. Since, by definition of polynomial social cost, all coefficients are non-negative, Equation (4.13) (page 88) implies that it suffices to show the claim for all polynomial cost functions $\pi^t(x) = x^t$ with $t \in [d]$. Fix any such $t \in [d]$. Let $\alpha = \frac{n}{m}$. Moreover, let

$$\begin{aligned}
\beta_j &= |\Lambda_j(\mathbf{P}) - \alpha|\,, \\
r_j &= |\mathsf{view}_{\mathbf{P}}(j)|\,, \\
q_j &= \min_{i\in[n]} \{p_{ij} \mid p_{ij} > 0\}
\end{aligned}$$

for all $j \in [m]$, and

$$\begin{aligned}
\mathcal{M}_1 &= \{j \in [m] \mid 0 < \Lambda_j(\mathbf{P}) \leq \alpha\}\,, \\
\mathcal{M}_2 &= \{j \in [m] \mid \Lambda_j(\mathbf{P}) > \alpha\}\,.
\end{aligned}$$

Clearly, for all $j \in \mathcal{M}_1 \cup \mathcal{M}_2$, we have $r_j \geq 1$ and

$$q_j \leq \frac{\Lambda_j(\mathbf{P})}{r_j} . \tag{4.55}$$

On the one hand,

$$\begin{aligned}
&\mathsf{SC}_{x^t}(n,m,\mathbf{P}) \\
&\overset{(4.14),\ page\ 88}{=} \sum_{j\in[m]} H((p_{1j},\ldots,p_{nj}),x^t) \\
&\overset{Lemma\ 4.2\ (page\ 81)}{\leq} \sum_{j\in\mathcal{M}_1} H(\frac{\alpha-\beta_j}{r_j},r_j,x^t) + \sum_{j\in\mathcal{M}_2} H(\frac{\alpha+\beta_j}{r_j},r_j,x^t) \\
&\overset{Proposition\ 4.3\ (page\ 81)}{=} \sum_{j\in\mathcal{M}_1}\sum_{i\in[t]} \left(\frac{\alpha-\beta_j}{r_j}\right)^i \cdot S(t,i)\cdot (r_j)^{\underline{i}} + \sum_{j\in\mathcal{M}_2}\sum_{i\in[t]} \left(\frac{\alpha+\beta_j}{r_j}\right)^i \cdot S(t,i)\cdot (r_j)^{\underline{i}} \\
&= \sum_{i\in[t]} S(t,i)\cdot\left[\sum_{j\in\mathcal{M}_1}\left(\frac{\alpha-\beta_j}{r_j}\right)^i \cdot (r_j)^{\underline{i}} + \sum_{j\in\mathcal{M}_2}\left(\frac{\alpha+\beta_j}{r_j}\right)^i \cdot (r_j)^{\underline{i}}\right] .
\end{aligned} \tag{4.56}$$

On the other hand,

$$\begin{aligned}
\mathsf{SC}_{x^t}(n,m,\mathbf{F}) &\overset{(4.14),\ page\ 88}{=} \sum_{j\in[m]} H((f_{1j},\ldots,f_{nj}),x^t) \\
&\overset{Theorem\ 4.36\ (page\ 112)}{=} m\cdot H(\frac{1}{m},n,x^t) \\
&\overset{Proposition\ 4.3\ (page\ 81)}{=} \sum_{i\in[t]} S(t,i)\cdot\left[m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i}\right] .
\end{aligned} \tag{4.57}$$

By Equations (4.56) and (4.57), it suffices to show:

Claim 4.139 *For all $i \geq 1$,*

$$\begin{aligned}
\Delta &= \left(\frac{n}{n-1}\right)^i \cdot\left[m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i}\right] - \left[\sum_{j\in\mathcal{M}_1}\left(\frac{\alpha-\beta_j}{r_j}\right)^i\cdot(r_j)^{\underline{i}} + \sum_{j\in\mathcal{M}_2}\left(\frac{\alpha+\beta_j}{r_j}\right)^i\cdot(r_j)^{\underline{i}}\right] \\
&\geq 0 .
\end{aligned}$$

Proof: We prove the claim by induction on i, $i \geq 1$. For the basis case, let $i = 1$. We have

$$\begin{aligned}
\Delta &= \left(\frac{n}{n-1}\right)\cdot m\cdot\alpha - \left[\sum_{j\in\mathcal{M}_1}(\alpha-\beta_j) + \sum_{j\in\mathcal{M}_2}(\alpha+\beta_j)\right] \\
&= \left(\frac{n}{n-1}\right)\cdot m\cdot\alpha - m\cdot\alpha \\
&> 0 ,
\end{aligned}$$

proving that the claim holds for the basis case.

For the induction step, let $i \geq 2$, and assume that Claim 4.139 (page 185) holds for $(i-1)$. Since $(r_j)^{\underline{i}} = 0$ for all $j \in \mathcal{M}_1 \cup \mathcal{M}_2$ with $r_j = 1$, we may assume that $r_j \geq 2$ for all $j \in \mathcal{M}_1 \cup \mathcal{M}_2$. Let $\beta = \max_{j \in \mathcal{M}_1} \beta_j$. By definition of Nash equilibrium,

$$\Lambda_j(\mathbf{P}) - q_j + 1 \quad \leq \quad \alpha - \beta + 1$$

for all $j \in \mathcal{M}_2$, which implies

$$\Lambda_j(\mathbf{P}) - \alpha \quad = \quad \beta_j + \beta \quad \leq \quad q_j \, .$$

Thus,

$$\beta_j + \beta \quad \leq \quad q_j \quad \overset{(4.55),\ page\ 185}{\leq} \quad \frac{\Lambda_j(\mathbf{P})}{r_j} \quad = \quad \frac{\alpha + \beta_j}{r_j} \, ,$$

showing that

$$\beta_j \quad \leq \quad \frac{\alpha - r_j \beta}{r_j - 1} \, . \tag{4.58}$$

We proceed by case analysis.

(1.) $\underline{\beta_j \leq (n - r_j)/(m(r_j - 1)) \text{ for all } j \in \mathcal{M}_2}$: In this case, we have

$$\begin{aligned} \alpha + \beta_j \quad &\leq \quad \frac{n}{m} + \frac{n - r_j}{m(r_j - 1)} \\ &= \quad \frac{n(r_j - 1) + n - r_j}{m(r_j - 1)} \\ &= \quad \frac{(n-1) r_j}{m(r_j - 1)} \end{aligned} \tag{4.59}$$

for all $j \in \mathcal{M}_2$. We get

$$\begin{aligned} \Delta \quad &\overset{\frac{n}{n-1} > 1}{>} \quad m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \left[\sum_{j \in \mathcal{M}_1} (\alpha - \beta_j) \cdot \frac{r_j - (i-1)}{r_j} \cdot \overbrace{\left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1}}^{\geq 0} \cdot (r_j)^{\underline{i-1}} \right. \\ &\qquad \left. + \sum_{j \in \mathcal{M}_2} (\alpha + \beta_j) \cdot \frac{r_j - (i-1)}{r_j} \cdot \left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right] \\ &\overset{\beta_j \geq 0}{\geq} \quad m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \left[\sum_{j \in \mathcal{M}_1} \alpha \cdot \frac{r_j - (i-1)}{r_j} \cdot \overbrace{\left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1}}^{\geq 0} \cdot (r_j)^{\underline{i-1}} \right. \\ &\qquad \left. + \sum_{j \in \mathcal{M}_2} (\alpha + \beta_j) \cdot \frac{r_j - (i-1)}{r_j} \cdot \left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right] . \end{aligned}$$

Clearly, the function $\frac{r_j-(i-1)}{r_j}$ is monotonic increasing in r_j for all $i \geq 2$. Thus,

$$\begin{aligned}
\Delta \overset{r_j \leq n}{>} \quad & m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \left[\sum_{j \in \mathcal{M}_1} \alpha \cdot \frac{n-(i-1)}{n} \cdot \left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right. \\
& \left. + \sum_{j \in \mathcal{M}_2} (\alpha + \beta_j) \cdot \frac{r_j-(i-1)}{r_j} \cdot \left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right] \\
\overset{(4.59),\ page\ 186}{\geq} \quad & m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \left[\sum_{j \in \mathcal{M}_1} \alpha \cdot \frac{n-(i-1)}{n} \cdot \left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right. \\
& \left. + \sum_{j \in \mathcal{M}_2} \left(\frac{(n-1)r_j}{m(r_j-1)} \right) \cdot \frac{r_j-(i-1)}{r_j} \cdot \underbrace{\left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}}}_{\geq 0} \right] .
\end{aligned}$$

Clearly,

$$\left(\frac{(n-1)r_j}{m(r_j-1)} \right) \cdot \frac{r_j-(i-1)}{r_j} \quad = \quad \frac{n-1}{m} \cdot \frac{r_j-(i-1)}{r_j-1}$$

is monotonic increasing in r_j for all $i \geq 2$. Thus,

$$\begin{aligned}
\Delta \overset{r_j \leq n}{>} \quad & m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \left[\sum_{j \in \mathcal{M}_1} \alpha \cdot \frac{n-(i-1)}{n} \cdot \left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right. \\
& \left. + \sum_{j \in \mathcal{M}_2} \left(\frac{n}{m} \right) \cdot \frac{n-(i-1)}{n} \cdot \left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right] \\
= \quad & m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \alpha \cdot \frac{n-(i-1)}{n} \cdot \left[\sum_{j \in \mathcal{M}_1} \left(\frac{\alpha - \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right. \\
& \left. + \sum_{j \in \mathcal{M}_2} \left(\frac{\alpha + \beta_j}{r_j} \right)^{i-1} \cdot (r_j)^{\underline{i-1}} \right] \\
\overset{Induction}{\geq} \quad & m \cdot \alpha^i \cdot \frac{n^{\underline{i}}}{n^i} - \alpha \cdot \frac{n-(i-1)}{n} \cdot m \cdot \alpha^{i-1} \cdot \frac{n^{\underline{i-1}}}{n^{i-1}} \\
= \quad & 0 ,
\end{aligned}$$

proving the inductive claim for the case where $\beta_j \leq (n-r_j)/(m(r_j-1))$ for all $j \in \mathcal{M}_2$.

(2.) $\exists j \in \mathcal{M}_2 : \beta_j > (n-r_j)/(m(r_j-1))$: We first show that $\beta \leq \frac{1}{m}$. Assume, by way of

contradiction, that $\beta > \frac{1}{m}$. Then, for any $j \in \mathcal{M}_2$ with $\beta_j > (n-r_j)/(m(r_j-1))$,

$$\begin{aligned}\beta_j &\overset{(4.58),\ page\ 186}{\leq} \frac{\alpha - r_j\cdot\beta}{r_j-1}\\ &\overset{\beta>\frac{1}{m}}{<} \frac{\frac{n}{m}-\frac{r_j}{m}}{r_j-1}\\ &= \frac{n-r_j}{m(r_j-1)}\\ &< \beta_j\ ,\end{aligned}$$

a contradiction. So, assume $\beta \leq \frac{1}{m}$. We get

$$\begin{aligned}\Delta &= \left(\frac{n}{n-1}\right)^i \cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} - \left[\sum_{j\in\mathcal{M}_1}(\alpha-\beta_j)\cdot\frac{r_j-(i-1)}{r_j}\cdot\overbrace{\left(\frac{\alpha-\beta_j}{r_j}\right)^{i-1}}^{\geq 0}\cdot(r_j)^{\underline{i-1}}\right.\\ &\qquad\left. + \sum_{j\in\mathcal{M}_2}(\alpha+\beta_j)\cdot\frac{r_j-(i-1)}{r_j}\cdot\left(\frac{\alpha+\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right]\\ &\overset{\beta_j\geq 0}{\geq} \left(\frac{n}{n-1}\right)^i \cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} - \left[\sum_{j\in\mathcal{M}_1}\alpha\cdot\frac{r_j-(i-1)}{r_j}\cdot\overbrace{\left(\frac{\alpha-\beta_j}{r_j}\right)^{i-1}}^{\geq 0}\cdot(r_j)^{\underline{i-1}}\right.\\ &\qquad\left. + \sum_{j\in\mathcal{M}_2}(\alpha+\beta_j)\cdot\frac{r_j-(i-1)}{r_j}\cdot\left(\frac{\alpha+\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right].\end{aligned}$$

Clearly, $\frac{r_j-(i-1)}{r_j}$ is monotonic increasing in r_j for all $i \geq 2$. Thus,

$$\begin{aligned}\Delta &\overset{r_j\leq n}{\geq} \left(\frac{n}{n-1}\right)^i \cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} - \left[\sum_{j\in\mathcal{M}_1}\alpha\cdot\frac{n-(i-1)}{n}\cdot\left(\frac{\alpha-\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right.\\ &\qquad\left. + \sum_{j\in\mathcal{M}_2}(\alpha+\beta_j)\cdot\frac{r_j-(i-1)}{r_j}\cdot\left(\frac{\alpha+\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right].\end{aligned}$$

We have

$$(\alpha+\beta_j)\cdot\frac{r_j-(i-1)}{r_j} \overset{(4.58),\ page\ 186}{\leq} \left(\alpha+\frac{\alpha-r_j\cdot\beta}{r_j-1}\right)\cdot\frac{r_j-(i-1)}{r_j} = f(r_j)\ ,$$

and the function $f(r_j)$ is monotonic increasing in r_j since $\alpha = \frac{n}{m} > \frac{1}{m} \geq \beta$ and therefore

$$f'(r_j) = \frac{(\alpha-\beta)(i-2)}{(r_j-1)^2} \geq 0$$

for all $i \geq 2$. Thus,

$$\begin{aligned} f(r_j) &\leq f(n) \\ &= \left(\alpha + \frac{\alpha - n\cdot\beta}{n-1}\right)\cdot\frac{n-(i-1)}{n} \\ &= \frac{n(\alpha-\beta)}{n-1}\cdot\frac{n-(i-1)}{n} \\ &\overset{\beta>0}{<} \frac{n}{n-1}\cdot\alpha\cdot\frac{n-(i-1)}{n}\,. \end{aligned} \tag{4.60}$$

We get

$$\begin{aligned} \Delta &\overset{(4.60)}{>} \left(\frac{n}{n-1}\right)^i\cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} \\ &\quad - \left[\sum_{j\in\mathcal{M}_1}\alpha\cdot\frac{n-(i-1)}{n}\cdot\left(\frac{\alpha-\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right. \\ &\qquad \left. + \sum_{j\in\mathcal{M}_2}\left(\frac{n}{n-1}\right)\cdot\alpha\cdot\frac{n-(i-1)}{n}\cdot\left(\frac{\alpha+\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right] \\ &\overset{\frac{n}{n-1}>1}{>} \left(\frac{n}{n-1}\right)^i\cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} \\ &\quad - \left(\frac{n}{n-1}\right)\cdot\alpha\cdot\frac{n-(i-1)}{n}\cdot\left[\sum_{j\in\mathcal{M}_1}\left(\frac{\alpha-\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right. \\ &\qquad \left. + \sum_{j\in\mathcal{M}_2}\left(\frac{\alpha+\beta_j}{r_j}\right)^{i-1}\cdot(r_j)^{\underline{i-1}}\right] \\ &\overset{Induction}{\geq} \left(\frac{n}{n-1}\right)^i\cdot m\cdot\alpha^i\cdot\frac{n^{\underline{i}}}{n^i} \\ &\quad - \left(\frac{n}{n-1}\right)\cdot\alpha\cdot\frac{n-(i-1)}{n}\cdot\left(\frac{n}{n-1}\right)^{i-1}\cdot\left[m\cdot\alpha^{i-1}\cdot\frac{n^{\underline{i-1}}}{n^{i-1}}\right] \\ &= 0\,, \end{aligned}$$

proving the inductive claim for this case and completing the proof of Claim 4.139 (page 185). ■

Combining Claim 4.139 (page 185) with Equation (4.56) (page 185) and Equation (4.57) (page 185) shows that

$$\mathsf{SC}_{x^t}(n,m,\mathbf{P}) \leq \left(1+\frac{1}{n-1}\right)^t\cdot\mathsf{SC}_{x^t}(n,m,\mathbf{F})$$

for all $t \in [d]$, as needed. ■

Theorem 4.140 *Consider the model of identical users, identical links and polynomial cost function $\pi^d(x) = x^d$. Then,*

$$\mathsf{PoA} \;\leq\; \left(1+\frac{1}{n-1}\right)^d \cdot B_d \;.$$

Proof: Fix any instance (n,m), associated Nash equilibrium $\mathbf{P}$ and fully mixed Nash equilibrium $\mathbf{F}$. Clearly,

$$\frac{\mathsf{SC}_{x^d}(n,m,\mathbf{P})}{\mathsf{OPT}_{x^d}(n,m)} \overset{\textit{Theorem } 4.138 \ (\textit{page } 184)}{\leq} \left(1+\frac{1}{n-1}\right)^d \cdot \frac{\mathsf{SC}_{x^d}(n,m,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,m)} \;.$$

Thus, it suffices to prove

$$\frac{\mathsf{SC}_{x^d}(n,m,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,m)} \;\leq\; B_d \;.$$

We proceed by case analysis:
$\underline{n \geq m}$: Since social cost is minimum if the users are evenly distributed to the links, a trivial lower bound on the optimum is

$$\mathsf{OPT}_{x^d}(n,m) \;\geq\; m \cdot \left(\frac{n}{m}\right)^d \;.$$

Thus,

$$\begin{aligned}
\frac{\mathsf{SC}_{x^d}(n,m,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,m)} &\leq \frac{1}{m}\cdot\left(\frac{m}{n}\right)^d \cdot \mathsf{SC}_{x^d}(n,m,\mathbf{F}) \\
&\overset{(4.14),\ \textit{page } 88}{=} \frac{1}{m}\cdot\left(\frac{m}{n}\right)^d \cdot \sum_{j\in[m]} H((f_{1j},\ldots,f_{nj}),x^d) \\
&\overset{\textit{Theorem } 4.36 \ (\textit{page } 112)}{=} \frac{1}{m}\cdot\left(\frac{m}{n}\right)^d \cdot m \cdot H(\frac{1}{m},n,x^d) \\
&\overset{\textit{Proposition } 4.3 \ (\textit{page } 81)}{=} \left(\frac{m}{n}\right)^d \cdot \sum_{i\in[d]} \left(\frac{1}{m}\right)^i \cdot S(d,i)\cdot n^{\underline{i}} \\
&= \sum_{i\in[d]} m^{d-i}\cdot S(d,i)\cdot \frac{n^{\underline{i}}}{n^d} \\
&\overset{n^{\underline{i}}\leq n^i}{\leq} \sum_{i\in[d]} \left(\frac{m}{n}\right)^{d-i}\cdot S(d,i) \\
&\overset{n\geq m}{\leq} \sum_{i\in[d]} S(d,i) \\
&\overset{(4.9),\ \textit{page } 81}{=} B_d \;.
\end{aligned}$$

$\underline{n < m}$: In this case, we have $\mathsf{OPT}_{x^d}(n,m) = n$. Thus

$$\begin{aligned}
\frac{\mathsf{SC}_{x^d}(n,m,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,m)} &= \frac{1}{n}\cdot \mathsf{SC}_{x^d}(n,m,\mathbf{F}) \\
&\overset{(4.14),\ page\ 88}{=} \frac{1}{n}\cdot \sum_{j\in[m]} H((f_{1j},\ldots,f_{nj}),x^d) \\
&\overset{Theorem\ 4.36\ (page\ 112)}{=} \frac{m}{n}\cdot H(\frac{1}{m},n,x^d) \\
&\overset{Proposition\ 4.3\ (page\ 81)}{=} \frac{m}{n}\cdot \sum_{i\in[d]} \left(\frac{1}{m}\right)^i \cdot S(d,i)\cdot n^{\underline{i}} \\
&= \sum_{i\in[d]} \left(\frac{1}{m}\right)^{i-1} \cdot S(d,i)\cdot \frac{n^{\underline{i}}}{n} \\
&\overset{n^{\underline{i}}\le n^i}{\le} \sum_{i\in[d]} \left(\frac{n}{m}\right)^{i-1} \cdot S(d,i) \\
&\overset{m>n}{<} \sum_{i\in[d]} S(d,i) \\
&\overset{(4.9),\ page\ 81}{=} B_d\ .
\end{aligned}$$

Note that

$$\frac{\mathsf{SC}_{x^d}(n,m,\mathbf{F})}{\mathsf{OPT}_{x^d}(n,m)} \ \to\ B_d$$

for $n = m \to \infty$. ■

Corollary 4.141 *Consider the model of identical users and identical links. Then,*

$$\mathsf{PoA} \ \le\ \sum_{t\in[d]} \left(1+\frac{1}{n-1}\right)^t \cdot B_t\ .$$

Proof: Fix any instance (n,m) and associated Nash equilibrium $\mathbf{P}$. Using the fact that $a_t \ge 0$ for all $t \in [d]\cup\{0\}$, we have

$$\begin{aligned}
\frac{\mathsf{SC}_{\pi^d(x)}(n,m,\mathbf{P})}{\mathsf{OPT}_{\pi^d(x)}(n,m)} &\overset{(4.13),\ page\ 88}{=} \frac{a_0+\sum_{t\in[d]} a_t\cdot \mathsf{SC}_{x^t}(n,m,\mathbf{P})}{a_0+\sum_{t\in[d]} a_t\cdot \mathsf{OPT}_{x^t}(n,m)} \\
&\le \frac{\sum_{t\in[d]} a_t\cdot \mathsf{SC}_{x^t}(n,m,\mathbf{P})}{\sum_{t\in[d]} a_t\cdot \mathsf{OPT}_{x^t}(n,m)} \\
&\le \sum_{t\in[d]} \frac{a_t\cdot \mathsf{SC}_{x^t}(n,m,\mathbf{P})}{a_t\cdot \mathsf{OPT}_{x^t}(n,m)} \\
&= \sum_{t\in[d]} \frac{\mathsf{SC}_{x^t}(n,m,\mathbf{P})}{\mathsf{OPT}_{x^t}(n,m)} \\
&\overset{Theorem\ 4.140\ (page\ 189)}{\le} \sum_{t\in[d]} \left(1+\frac{1}{n-1}\right)^t \cdot B_t\ ,
\end{aligned}$$

proving the claim. ■

4.9 Polynomial Social Cost and Related Links

In this section, we consider the KP-model with polynomial social cost and related links. Subsection 4.9.1 deals with pure Nash equilibria only, whereas the results quoted in Subsection 4.9.2 hold for general (i.e. mixed) Nash equilibria. Subsection 4.9.3 concentrates on the fully mixed Nash equilibrium. In order to illustrate the results, we use the following instance.

Example 4.142 *Consider the following instance $(n,\mathbf{c})$: We have $n=4$ identical users, $m=4$ related links and the polynomial cost function $\pi^2(x)=x^2$. There are one link with capacity $c_1=4$ and three links with capacities $c_2=c_3=c_4=1$.*

4.9.1 Pure Nash Equilibria

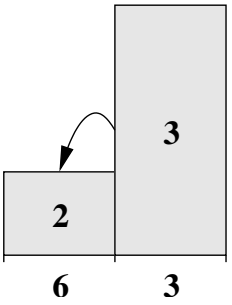
4.9.1.1 Computation of Nash Equilibria

As already mentioned, the usage of polynomial social cost as the cost measure of social welfare alters neither the definition of individual cost nor the set of Nash equilibria. Clearly, this implies that all results on the convergence of sequences of selfish steps in Section 4.5 also hold in this model, but selfish steps might increase polynomial social cost (Proposition 4.143). The LPT-algorithm (see Algorithm 3, page 93) still yields a pure Nash equilibrium for the case of related links (see Theorem 4.47, page 116). However, up to now there exists no performance analysis of this algorithm with respect to polynomial cost functions or even to the d-norm.

Proposition 4.143 *Consider the model of arbitrary users and related links. Then, a selfish step of a user from a link $j_1 \in [m]$ to a link $j_2 \in [m]$ with $c_{j_2} > c_{j_1}$ can increase polynomial social cost.*

Proof: Consider the following instance $(\mathbf{w},\mathbf{c})$:

There are 2 users with traffics $w_1=3$ and $w_2=2$, and two links with capacities $c_1=6$ and $c_2=3$. Let $\mathbf{L}=\langle 2,1\rangle$ be the pure assignment in which user 1 is assigned to link 2 and user 2 is assigned to link 1. Clearly, user 2 can improve by moving to link 1, yielding the pure assignment $\mathbf{L}'=\langle 1,1\rangle$. We have



$$\mathsf{SC}_{x^2}(\mathbf{w},\mathbf{c},\mathbf{L}) \;=\; \frac{2^2}{6}+\frac{3^2}{3} \;=\; \frac{11}{3}$$

whereas

$$\mathsf{SC}_{x^2}(\mathbf{w},\mathbf{c},\mathbf{L}') \;=\; \frac{5^2}{6} \;=\; \frac{25}{6} \;>\; \frac{11}{3}\,.$$

This proves the claim. ■

4.9.1.2 Computation of Best Nash Equilibria

Identical Users. In contrast to the case of identical links, there does not always exist a pure Nash equilibrium with optimum social cost even in case of identical users (Proposition 4.144, page 193). Moreover, it is not clear how selfish steps can be used to approximate a best pure Nash equilibrium since selfish steps can increase polynomial social cost (see

Proposition 4.143, page 192). However, for the case of identical users and polynomial cost function $\pi^d(x) = x^d$, $d \geq 2$, Lemma 4.145 enables us to use algorithm BESTPURENASHEQUILIBRIUM, stated as Algorithm 10 (page 195), to compute a best pure Nash equilibrium in $O(m \log n \log m)$ time (Theorem 4.146, page 194). Note that this result can not be generalized to arbitrary polynomial cost functions since Lemma 4.145 does not hold for $d = 1$.

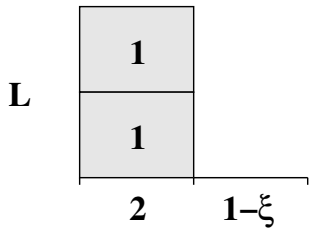
Proposition 4.144 *Consider the model of identical users, related links and polynomial cost function $\pi^d(x) = x^d$. Then, for any ε with $0 < \varepsilon \leq \frac{2^d}{3} - 1$, there exists an instance $(\mathbf{w}, \mathbf{c})$ with*

$$\frac{\mathsf{BC}_{x^d}(\mathbf{w},\mathbf{c})}{\mathsf{OPT}_{x^d}(\mathbf{w},\mathbf{c})} \geq \left(\frac{2^d}{3} - \varepsilon\right).$$

Proof: Consider the following instance $(2, \mathbf{c})$:

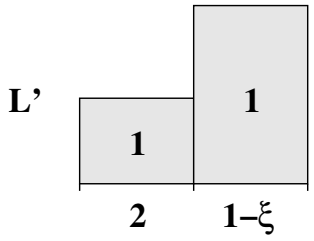
There are 2 identical users with traffics $w_1 = w_2 = 1$, and two links with capacities $c_1 = 2$ and $c_2 = 1 - \xi$, where

$$\xi = \frac{3\varepsilon}{\frac{2^{d+1}}{3} + \varepsilon}.$$



Let $\mathbf{L} = \langle 1, 1 \rangle$ be the pure assignment in which both users are assigned to link 1, and let $\mathbf{L}' = \langle 1, 2 \rangle$ be the pure assignment in which user 1 is assigned to link 1 and user 2 is assigned to link 2. Note that $\mathbf{L}$ is the only pure Nash equilibrium for the given instance. We have

$$\mathsf{SC}_{x^d}(2,\mathbf{c},\mathbf{L}) = \frac{2^d}{2}$$



whereas

$$\mathsf{SC}_{x^d}(2,\mathbf{c},\mathbf{L}') = \frac{1^d}{2} + \frac{1^d}{1-\xi} = \frac{3 \cdot 2^d}{2^{d+1} - 6\varepsilon}.$$

Thus,

$$\begin{aligned} \frac{\mathsf{BC}_{x^d}(\mathbf{w},\mathbf{c})}{\mathsf{OPT}_{x^d}(\mathbf{w},\mathbf{c})} &\geq \frac{\mathsf{SC}_{x^d}(2,\mathbf{c},\mathbf{L})}{\mathsf{SC}_{x^d}(2,\mathbf{c},\mathbf{L}')} \\ &= \frac{\frac{2^d}{2}}{\frac{3 \cdot 2^d}{2^{d+1} - 6\varepsilon}} \\ &= \frac{2^d}{3} - \varepsilon, \end{aligned}$$

as needed. ■

Lemma 4.145 *Let $c_{j_1}, c_{j_2} \in \mathbb{N}$ with $c_{j_1} > c_{j_2}$. Then, for all integers $d \geq 2$ and $k \in \mathbb{N}$,*

$$\frac{(k \cdot c_{j_1} - 1)^d}{c_{j_1}} + \frac{(k \cdot c_{j_2})^d}{c_{j_2}} < \frac{(k \cdot c_{j_1})^d}{c_{j_1}} + \frac{(k \cdot c_{j_2} - 1)^d}{c_{j_2}}.$$

Proof: We show by induction on d, $d \geq 2$, that

$$\Delta = \frac{(k \cdot c_{j_1} - 1)^d}{c_{j_1}} + \frac{(k \cdot c_{j_2})^d}{c_{j_2}} - \frac{(k \cdot c_{j_1})^d}{c_{j_1}} - \frac{(k \cdot c_{j_2} - 1)^d}{c_{j_2}} < 0 .$$

For the basis case, let $d = 2$. Then,

$$\begin{aligned}
\Delta &= \frac{(k \cdot c_{j_1} - 1)^2}{c_{j_1}} + \frac{(k \cdot c_{j_2})^2}{c_{j_2}} - \frac{(k \cdot c_{j_1})^2}{c_{j_1}} - \frac{(k \cdot c_{j_2} - 1)^2}{c_{j_2}} \\
&= \frac{1}{c_{j_1}} \left((k \cdot c_{j_1} - 1)^2 - (k \cdot c_{j_1})^2 \right) + \frac{1}{c_{j_2}} \left((k \cdot c_{j_2})^2 - (k \cdot c_{j_2} - 1)^2 \right) \\
&= \frac{1}{c_{j_1}} \left(-2k \cdot c_{j_1} + 1 \right) + \frac{1}{c_{j_2}} \left(2k \cdot c_{j_2} - 1 \right) \\
&= \frac{1}{c_{j_1}} - \frac{1}{c_{j_2}} \\
&\overset{c_{j_1} > c_{j_2}}{<} 0 ,
\end{aligned}$$

proving that the claim holds for the basis case. For the induction step, let $d \geq 3$, and assume that the claim holds for $(d-1)$. Then,

$$\begin{aligned}
\Delta &= \frac{(k \cdot c_{j_1} - 1)^d}{c_{j_1}} + \frac{(k \cdot c_{j_2})^d}{c_{j_2}} - \frac{(k \cdot c_{j_1})^d}{c_{j_1}} - \frac{(k \cdot c_{j_2} - 1)^d}{c_{j_2}} \\
&= (k \cdot c_{j_1} - 1) \cdot \frac{(k \cdot c_{j_1} - 1)^{d-1}}{c_{j_1}} + k \cdot c_{j_2} \cdot \frac{(k \cdot c_{j_2})^{d-1}}{c_{j_2}} \\
&\quad - k \cdot c_{j_1} \cdot \frac{(k \cdot c_{j_1})^{d-1}}{c_{j_1}} - (k \cdot c_{j_2} - 1) \cdot \frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}} \\
&= k \cdot c_{j_1} \cdot \left[\frac{(k \cdot c_{j_1} - 1)^{d-1}}{c_{j_1}} - \frac{(k \cdot c_{j_1})^{d-1}}{c_{j_1}} \right] + k \cdot c_{j_2} \cdot \left[\frac{(k \cdot c_{j_2})^{d-1}}{c_{j_2}} - \frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}} \right] \\
&\quad - \frac{(k \cdot c_{j_1} - 1)^{d-1}}{c_{j_1}} + \frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}} .
\end{aligned}$$

Clearly, $\frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}}$ is strictly increasing in c_{j_2}. Thus,

$$\begin{aligned}
\Delta &\overset{c_{j_1} > c_{j_2}}{<} k \cdot c_{j_1} \cdot \left[\frac{(k \cdot c_{j_1} - 1)^{d-1}}{c_{j_1}} - \frac{(k \cdot c_{j_1})^{d-1}}{c_{j_1}} \right] + k \cdot c_{j_2} \cdot \left[\frac{(k \cdot c_{j_2})^{d-1}}{c_{j_2}} - \frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}} \right] \\
&\overset{c_{j_1} > c_{j_2}}{<} k \cdot c_{j_2} \cdot \left[\frac{(k \cdot c_{j_1} - 1)^{d-1}}{c_{j_1}} - \frac{(k \cdot c_{j_1})^{d-1}}{c_{j_1}} + \frac{(k \cdot c_{j_2})^{d-1}}{c_{j_2}} - \frac{(k \cdot c_{j_2} - 1)^{d-1}}{c_{j_2}} \right] \\
&\overset{Induction}{<} 0 .
\end{aligned}$$

This completes the prove of the inductive claim. ■

Theorem 4.146 *Consider the model of identical users, related links and polynomial cost function $\pi^d(x) = x^d$ with $d \geq 2$. Then,* BESTPURENASHEQUILIBRIUM *computes a pure Nash equilibrium with minimum polynomial social cost using $O(m \log n \log m)$ time.*

Algorithm 10 (BESTPURENASHEQUILIBRIUM)
Input: an instance $(n, \mathbf{c})$
Output: a pure assignment $\mathbf{L}'$

(1) **begin**
(2) compute a pure Nash equilibrium $\mathbf{L}$;
(3) $S_1 \leftarrow \{j \in [m] \mid \Lambda_j(\mathbf{L}) = \mathsf{OPT}_\infty(n, \mathbf{c})\}$;
(4) $S_2 \leftarrow \{j \in [m] \mid \Lambda_j(\mathbf{L}) = \frac{\mathsf{OPT}_\infty(n,\mathbf{c}) \cdot c_j - 1}{c_j}\}$;
(5) **while** there exist links $j_1 \in S_1$ and $j_2 \in S_2$ with $c_{j_1} > c_{j_2}$ **do**
(6) move one user from link j_1 to link j_2;
(7) $S_1 \leftarrow S_1 \setminus \{j_1\}$;
(8) $S_2 \leftarrow S_1 \setminus \{j_2\}$;
(9) **return** the resulting pure assignment $\mathbf{L}'$;
(10) **end**

Proof: Fix any instance $(n, \mathbf{c})$. Assume, without loss of generality, that all capacities are integers. Let $\mathbf{L}$ be the pure Nash equilibrium computed in line (2) of the algorithm. Proposition 4.42 (page 113) and the definition of Nash equilibrium imply that

$$\Lambda_j(\mathbf{L}) \quad \in \quad \left[\frac{\mathsf{OPT}_\infty(n, \mathbf{c}) \cdot c_j - 1}{c_j}, \frac{\mathsf{OPT}_\infty(n, \mathbf{c}) \cdot c_j}{c_j}\right]$$

for all $j \in [m]$. Clearly, by moving a user from a link $j_1 \in S_1$ to a link $j_2 \in S_2$, this property is preserved. Thus, all users in the resulting pure assignment $\mathbf{L}'$ are satisfied, showing that $\mathbf{L}'$ is a Nash equilibrium. We proceed by showing that $\mathsf{SC}_{x^d}(n, \mathbf{c}, \mathbf{L}') = \mathsf{OPT}_{x^d}(n, \mathbf{c})$. Let

$$\begin{aligned} S_1' &= \{j \in [m] \mid \Lambda_j(\mathbf{L}') = \mathsf{OPT}_\infty(n, \mathbf{c})\}\,, \\ S_2' &= \left\{ j \in [m] \;\middle|\; \Lambda_j(\mathbf{L}') = \frac{\mathsf{OPT}_\infty(n, \mathbf{c}) \cdot c_j - 1}{c_j} \right\}. \end{aligned}$$

Assume, by way of contradiction, that $\mathsf{SC}_{x^d}(n, \mathbf{c}, \mathbf{L}') > \mathsf{OPT}_{x^d}(n, \mathbf{c})$, and let $\mathbf{Q}$ be an optimum assignment, that is, $\mathsf{SC}_{x^d}(n, \mathbf{c}, \mathbf{Q}) = \mathsf{OPT}_{x^d}(n, \mathbf{c})$. Then, there exist links $j_1 \in S_1'$ and $j_2 \in S_2'$, $c_{j_1} > c_{j_2}$, with

$$\begin{aligned} \Lambda_{j_1}(\mathbf{Q}) &= \mathsf{OPT}_\infty(n, \mathbf{c})\,, \\ \Lambda_{j_2}(\mathbf{Q}) &= \frac{\mathsf{OPT}_\infty(n, \mathbf{c}) \cdot c_j - 1}{c_j}\,, \end{aligned}$$

Lemma 4.145 (page 193) shows that by moving a user from link j_1 to link j_2 we get a pure assignment $\mathbf{Q}'$ with

$$\mathsf{SC}_{x^d}(n, \mathbf{c}, \mathbf{Q}') \quad < \quad \mathsf{SC}_{x^d}(n, \mathbf{c}, \mathbf{Q}) \quad = \quad \mathsf{OPT}_{x^d}(n, \mathbf{c})\,,$$

a contradiction.

By Theorem 4.43 (page 114), we can compute a pure Nash equilibrium in $O(m \log n \log m)$ time. The computation of S_1 and S_2 takes $O(m)$ time. Moving the users to the right-most links

also needs $O(m)$ time. This proves the running time. ■

Example 4.142 (continued) *For the given instance, each pure Nash equilibrium is of one of the following types: Either all users are assigned to link* 1*, or three users are assigned to link* 1 *and one user is assigned to one of the remaining links (note that all pure Nash equilibria of the second type have the same social cost). Consider* $\mathbf{L} = \langle 1,1,1,1\rangle$ *and* $\mathbf{L}' = \langle 1,1,1,2\rangle$ *as representatives of these two types (see Figure 4.16). We have*

$$\mathsf{SC}_{x^2}(4,\mathbf{c},\mathbf{L}) \;=\; \frac{4^2}{4} \;=\; 4$$

whereas

$$\mathsf{SC}_{x^2}(4,\mathbf{c},\mathbf{L}') \;=\; \frac{3^2}{4}+\frac{1^2}{1} \;=\; \frac{13}{4} \;<\; \mathsf{SC}_{x^2}(4,\mathbf{c},\mathbf{L})\,.$$

BESTPURENASHEQUILIBRIUM *returns the pure Nash equilibrium* $\mathbf{L}'$.

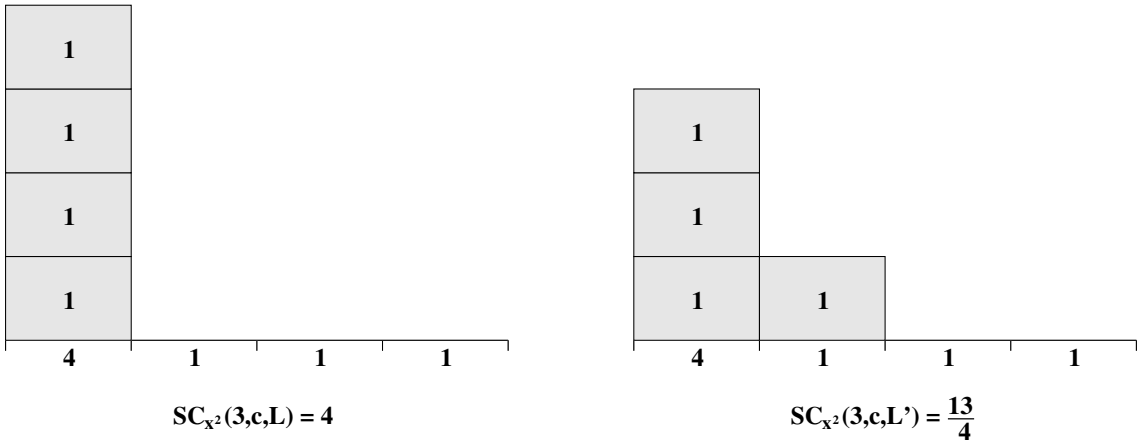


Figure 4.16: Non-optimum pure Nash equilibrium $\mathbf{L} = \langle 1,1,1,1\rangle$ of the instance in Example 4.142 (page 192) (left hand side), and optimum pure Nash equilibrium $\mathbf{L}' = \langle 1,1,1,2\rangle$ returned by BESTPURENASHEQUILIBRIUM (right hand side).

Arbitrary Users. Since, for the case of arbitrary users, BEST PURE NE is $\mathcal{NP}$-complete even for the model of identical links (see Theorem 4.118, page 168), this of course also holds for the more general model of related links. Up to now, no approximation algorithm is known. Since BEST PURE NE is $\mathcal{NP}$-complete in the strong sense, there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.147). Moreover, we can give a lower bound on the social cost of an optimum pure assignment in case that $\pi^2(x) = x^2$ (Proposition 4.148, page 197).

Theorem 4.147 *Consider the model of arbitrary users and related links. Then, there exists a pseudo-polynomial-time algorithm for* m-BEST PURE NE.

Proof: The proof of Theorem 4.104 (page 161) can easily be adapted to this setting. ■

Proposition 4.148 *Consider the model of arbitrary users, related links and polynomial cost function $\pi^2(x) = x^2$. Then,*

$$\mathsf{OPT}_{x^2}(\mathbf{w},\mathbf{c}) \;\geq\; \frac{W^2}{C}\,.$$

Proof: Fix any instance $(\mathbf{w},\mathbf{c})$ and associated optimum assignment $\mathbf{Q}$. Clearly, $\mathsf{SC}_{x^2}(\mathbf{w},\mathbf{c},\mathbf{Q})$ is symmetric in all loads $\delta_j(\mathbf{Q})$, and

$$\frac{W^2}{C} \;=\; \sum_{j\in[m]} \frac{\left(\frac{W\cdot c_j}{C}\right)^2}{c_j}\,.$$

Thus, it suffices to show that $\mathsf{SC}_{x^2}(\mathbf{w},\mathbf{c},\mathbf{Q})$ does not increase by replacing two loads, say $\delta_{j_1}(\mathbf{Q})$ and $\delta_{j_2}(\mathbf{Q})$, $j_1, j_2 \in [m]$, $j_1 \neq j_2$, by

$$\begin{aligned}\widehat{\delta}_{j_1} &= \frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))\cdot c_{j_1}}{c_{j_1}+c_{j_2}}\,,\\ \widehat{\delta}_{j_2} &= \frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))\cdot c_{j_2}}{c_{j_1}+c_{j_2}}\,.\end{aligned}$$

On the one hand,

$$\frac{\delta_{j_1}(\mathbf{Q})^2}{c_{j_1}}+\frac{\delta_{j_2}(\mathbf{Q})^2}{c_{j_2}} = \frac{1}{c_{j_1}c_{j_2}(c_{j_1}+c_{j_2})}\left[c_{j_2}(c_{j_1}+c_{j_2})\delta_{j_1}(\mathbf{Q})^2+c_{j_1}(c_{j_1}+c_{j_2})\delta_{j_2}(\mathbf{Q})^2\right]\,.$$

On the other hand,

$$\begin{aligned}\frac{\widehat{\delta}_{j_1}^2}{c_{j_1}}+\frac{\widehat{\delta}_{j_2}^2}{c_{j_2}} &= \frac{\left(\frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))\cdot c_{j_1}}{c_{j_1}+c_{j_2}}\right)^2}{c_{j_1}}+\frac{\left(\frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))\cdot c_{j_2}}{c_{j_1}+c_{j_2}}\right)^2}{c_{j_2}}\\ &= \frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))^2\cdot c_{j_1}}{(c_{j_1}+c_{j_2})^2}+\frac{(\delta_{j_1}(\mathbf{Q})+\delta_{j_2}(\mathbf{Q}))^2\cdot c_{j_2}}{(c_{j_1}+c_{j_2})^2}\\ &= \frac{(\delta_{j_1}(\mathbf{Q})+\delta j_2\mathbf{Q})^2}{(c_{j_1}+c_{j_2})}\\ &= \frac{1}{c_{j_1}c_{j_2}(c_{j_1}+c_{j_2})}\left[c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})^2+2c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})\delta j_2\mathbf{Q}+c_{j_1}c_{j_2}\delta_{j_2}(\mathbf{Q})^2\right]\,.\end{aligned}$$

Since

$$\begin{aligned}&\left[c_{j_2}(c_{j_1}+c_{j_2})\delta_{j_1}(\mathbf{Q})^2+c_{j_1}(c_{j_1}+c_{j_2})\delta_{j_2}(\mathbf{Q})^2\right]\\ &-\left[c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})^2+2c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})\delta_{j_2}(\mathbf{Q})+c_{j_1}c_{j_2}\delta_{j_2}(\mathbf{Q})^2\right]\\ &\quad= c_{j_2}(c_{j_1}+c_{j_2})\delta_{j_1}(\mathbf{Q})^2+c_{j_1}(c_{j_1}+c_{j_2})\delta_{j_2}(\mathbf{Q})^2\\ &\qquad -c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})^2-2c_{j_1}c_{j_2}\delta_{j_1}(\mathbf{Q})\delta_{j_2}(\mathbf{Q})-c_{j_1}c_{j_2}\delta_{j_2}(\mathbf{Q})^2\\ &\quad= (c_{j_2}\delta_{j_1}(\mathbf{Q}))^2-2(c_{j_2}\delta_{j_1}(\mathbf{Q}))(c_{j_1}\delta_{j_2}(\mathbf{Q}))+(c_{j_1}\delta_{j_2}(\mathbf{Q}))^2\\ &\quad= (c_{j_2}\delta_{j_1}(\mathbf{Q})-c_{j_1}\delta_{j_2}(\mathbf{Q}))^2\\ &\quad\geq 0\,,\end{aligned}$$

this proves the claim. ■

4.9.1.3 Price of Anarchy and Computation of Worst Nash Equilibria

Identical Users. For identical users and polynomial cost function $\pi^d(x) = x^d$ with $d \geq 2$, Lemma 4.145 (page 193) also enables us to compute a worst-case pure Nash equilibrium. The algorithm WORSTPURENASHEQUILIBRIUM works in the same way as BESTPURENASHEQUILIBRIUM, but moves users from links $j_1 \in S_1$ to links $j_2 \in S_2$ with $c_{j_1} < c_{j_2}$. This shows that a worst-case pure Nash equilibrium can be computed in $O(m \log n \log m)$ time (Theorem 4.149).

Bounds on the price of anarchy are only known when restricting to polynomial cost functions $\pi^d(x) = x^d$. In particular, if $d = 2$, then the price of anarchy is $\frac{4}{3}$ (Theorem 4.150). Moreover, for arbitrary $d \geq 2$, we can give the lower bound $\Omega(m^{d-2})$ on the price of anarchy which, in contrast to the constant bound for the case $d = 2$, is polynomial in m (Proposition 4.151).

Theorem 4.149 *Consider the model of identical users, related links and polynomial cost function $\pi^d(x) = x^d$ with $d \geq 2$. Then,* WORSTPURENASHEQUILIBRIUM *computes a pure Nash equilibrium with minimum polynomial social cost using $O(m \log n \log m)$ time.*

Theorem 4.150 (Lücking *et al.* [102]) *Consider the model of identical users, related links and polynomial cost function $\pi^2(x) = x^2$, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \;=\; \frac{4}{3}.$$

Example 4.142 (continued) *For the given instance, the worst-case pure Nash equilibrium $\mathbf{L} = \langle 1,1,1,1 \rangle$ has social cost 4 whereas the optimum assignment $\langle 1,1,2,3 \rangle$ has social cost 3 (see Figure 4.17). Thus,*

$$\frac{\mathsf{SC}_{x^2}(n,\mathbf{c},\mathbf{L})}{\mathsf{OPT}_{x^2}(n,\mathbf{c})} \;=\; \frac{4}{3}.$$

Proposition 4.151 *Consider the model of identical users, related links and polynomial cost function $\pi^d(x) = x^d$, restricted to pure Nash equilibria. Then,*

$$\mathsf{PoA} \;=\; \Omega\left(m^{d-2}\right).$$

Proof: Consider the following instance $(n,\mathbf{c})$: There are m links with capacities $c_1 = m-1$ and $c_j = 1$ for all $j \in [m] \setminus \{1\}$, and $n = m$ identical users. On the one hand, assigning all users to link 1 yields a pure Nash equilibrium $\mathbf{L}$ with social cost

$$\mathsf{SC}_{x^d}(n,\mathbf{c},\mathbf{L}) \;=\; \frac{m^d}{m} \;=\; m^{d-1}.$$

On the other hand, evenly distributing the users to the links such that all users are solo yields a pure assignment $\mathbf{L}'$ with social cost

$$\mathsf{SC}_{x^d}(n,\mathbf{c},\mathbf{L}') \;=\; (m-1)\cdot 1^d + \frac{1^d}{m-1}.$$

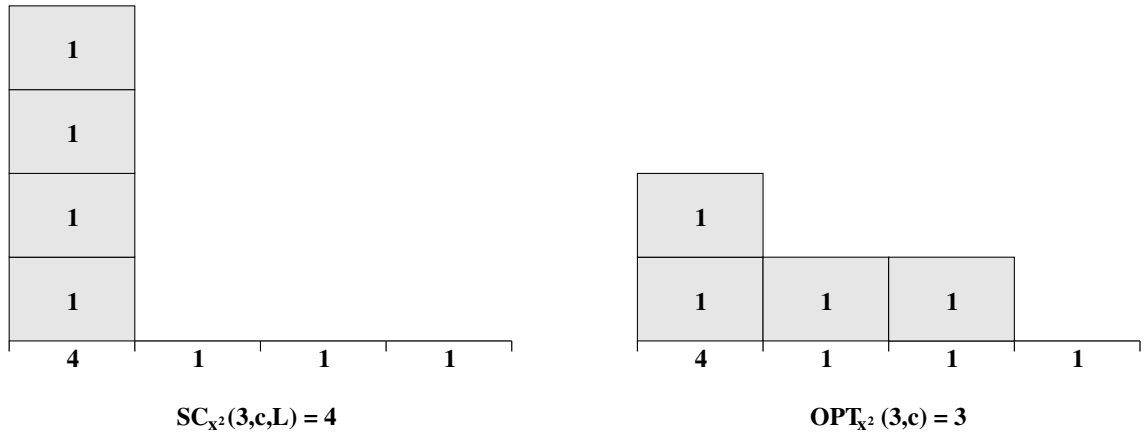


Figure 4.17: Worst-case pure Nash equilibrium $\mathbf{L} = \langle 1,1,1,1 \rangle$ (left hand side), and optimum assignment $\langle 1,1,2,3 \rangle$ of the instance in Example 4.142 (page 192) (right hand side).

Thus,

$$\begin{aligned} \mathsf{PoA} &\geq \frac{\mathsf{SC}_{x^d}(n,\mathbf{c},\mathbf{L})}{\mathsf{SC}_{x^d}(n,\mathbf{c},\mathbf{L}')} \\ &= \frac{m^{d-1}}{m-1+\frac{1}{m-1}} \\ &= \Omega\left(m^{d-2}\right), \end{aligned}$$

as needed. ■

Arbitrary Users. Clearly, WORST PURE NE is $\mathcal{NP}$-complete (see Theorem 4.121, page 169). Since WORST PURE NE is $\mathcal{NP}$-complete in the strong sense, there also exists no pseudo-polynomial algorithm to solve it. However, we can give such an algorithm for constant m (Theorem 4.152). Up to now, no upper bound on the price of anarchy is known, but it is at least $\Omega\left(m^{d-2}\right)$ (see Proposition 4.151, page 198).

Theorem 4.152 *Consider the model of arbitrary users and related links. Then, there exists a pseudo-polynomial-time algorithm for* m-WORST PURE NE.

Proof: The proof of Theorem 4.105 (page 161) can easily be adapted to this setting. ■

4.9.2 Mixed Nash Equilibria

As seen in Theorem 4.30 (page 111), the computation of makespan social cost for any given mixed assignment is $\#\mathcal{P}$-complete. For the polynomial social cost function $\pi^2(x) = x^2$, Lücking *et al.* [102] showed that its computation is possible in pseudo-polynomial time, that is, in $O(nmW)$ time. Recently, Rode [126] showed that for this special polynomial cost function, polynomial social cost can be expressed as a weighted sum of the expected individual costs

of the users, where the weights are the user traffics (see Equation (4.15), page 88). This observation allows to prove the following result:

Theorem 4.153 (Rode [126]) *Consider the model of arbitrary users, related links and polynomial cost function $\pi^2(x) = x^2$. Then, for any instance and associated assignment, polynomial social cost can be computed in $O(nm)$ time.*

4.9.3 Fully Mixed Nash Equilibria

We now concentrate on instances for which the fully mixed Nash equilibrium exists. We first consider the case of identical users and polynomial cost function $\pi^2(x) = x^2$. We start by giving a closed formula for the social cost of the (unique) fully mixed Nash equilibrium (Theorem 4.154). We proceed by showing that the social cost of any pure Nash equilibrium is bounded from above by the social cost of the fully mixed Nash equilibrium, proving the FMNE Conjecture for pure Nash equilibria (Theorem 4.155). Recently, Gairing *et al.* [58] proved the FMNE Conjecture for the model of identical users, links with non-decreasing convex latency functions and social cost defined as the sum of minimum expected individual cost of the users. Since this model and our model coincide for the polynomial cost function $\pi^2(x) = x^2$, related links and Nash equilibria (see Equation (4.15), page 88), this also proves the conjecture in our model (Theorem 4.156, page 201). We conclude by showing that for the polynomial cost function $\pi^3(x) = x^3$, the FMNE Conjecture does not hold even for $n = 3$ identical users (Theorem 4.157, page 201).

Theorem 4.154 (Lücking *et al.* [102]) *Consider the model of identical users, related links and polynomial cost function $\pi^2(x) = x^2$. Then,*

$$\mathsf{SC}_{x^2}(n, \mathbf{c}, \mathbf{F}) \quad = \quad \frac{n(n+m-1)}{C} \; .$$

Theorem 4.155 *Consider the model of identical users, related links and polynomial cost function $\pi^2(x) = x^2$. Then, for any pure Nash equilibrium $\mathbf{L}$, it is $\mathsf{SC}_{x^2}(n, \mathbf{c}, \mathbf{L}) \leq \mathsf{SC}_{x^2}(n, \mathbf{c}, \mathbf{F})$.*

Proof: Fix any instance $(n, \mathbf{c})$ and associated pure Nash equilibrium $\mathbf{L}$. We can write $\delta_j(\mathbf{L}) = n_j(\mathbf{L})$ for all $j \in [m]$, where $n_j(\mathbf{L})$ is the number of users assigned to link j. This is possible since all users are identical, that is, $w_i = 1$ for all $i \in [n]$. By definition of Nash equilibrium,

$$\frac{n_{j_2}(\mathbf{L})}{c_{j_2}} \quad \leq \quad \frac{n_{j_1}(\mathbf{L}) + 1}{c_{j_1}}$$

for all $j_1, j_2 \in [m]$. Thus,

$$n_{j_1}(\mathbf{L}) \quad \geq \quad \frac{c_{j_1}}{c_{j_2}} \cdot n_{j_2}(\mathbf{L}) - 1$$

for all $j_1, j_2 \in [m]$. Hence, for any fixed link $j_2 \in [m]$, we have

$$\begin{aligned} n &= \sum_{j_1 \in [m]} n_{j_1}(\mathbf{L}) \\ &\geq n_{j_2}(\mathbf{L}) + \sum_{j_1 \in [m] \setminus \{j_2\}} \left(\frac{c_{j_1}}{c_{j_2}} \cdot n_{j_2}(\mathbf{L}) - 1 \right) \\ &= \frac{C}{c_{j_2}} \cdot n_{j_2}(\mathbf{L}) - (m-1) \; , \end{aligned}$$

and we can write

$$n_{j_2}(\mathbf{L}) \;\leq\; \frac{c_{j_2}}{C}\cdot(n+(m-1))\,.$$

It follows that

$$\begin{aligned}
\mathsf{SC}_{x^2}(n,\mathbf{c},\mathbf{L}) &= \sum_{j\in[m]}\frac{n_j(\mathbf{L})^2}{c_j}\\
&\leq \left(\sum_{j\in[m]} n_j(\mathbf{L})\right)\cdot\frac{n+(m-1)}{C}\\
&= \frac{n(n+m-1)}{C}\\
&\overset{\textit{Theorem 4.154 (page 200)}}{=} \mathsf{SC}_{x^2}(n,\mathbf{c},\mathbf{F})\,,
\end{aligned}$$

as needed. ■

Theorem 4.156 (Gairing *et al.* [58]) *Consider the model of identical users, related links and polynomial cost function $\pi^2(x)=x^2$. Then, the* FMNE *Conjecture is valid.*

Theorem 4.157 *Consider the model of identical users, related links and polynomial cost function $\pi^d(x)=x^d$ with $d\geq 3$. Then, the* FMNE *Conjecture is not valid.*

Proof: Consider the following instance. There are $n=3$ identical users, $m=50$ related links with $c_1=\frac{49}{25}$ and $c_j=1$ for all $j\in[50]\setminus\{1\}$, and the polynomial cost function $\pi^3(x)=x^3$. By Theorem 4.67 (page 133), there exists a (unique) fully mixed Nash equilibrium $\mathbf{F}$ with probabilities

$$\begin{aligned}
f_{i1} &= \left(1-\frac{50\cdot\frac{49}{25}}{49+\frac{49}{25}}\right)\left(1-\frac{3}{2}\right)+\frac{\frac{49}{25}}{49+\frac{49}{25}} \;=\; \frac{1}{2}\,,\\
f_{ij} &= \left(1-\frac{50}{49+\frac{49}{25}}\right)\left(1-\frac{3}{2}\right)+\frac{1}{49+\frac{49}{25}} \;=\; \frac{1}{98}\,,
\end{aligned}$$

for all $i\in[3]$ and $j\in[50]\setminus\{1\}$. On the one hand, the social cost of $\mathbf{F}$ is

$$\begin{aligned}
\mathsf{SC}_{x^3}(3,m,\mathbf{F}) &= \left(\frac{1}{2}\right)^3\cdot\frac{3^3}{\frac{49}{25}}\\
&\quad+\left(\frac{1}{2}\right)^2\cdot\left(\frac{1}{98}\right)\left[\left(\frac{2^3}{\frac{49}{25}}+1\right)\cdot 3\cdot 49\right]\\
&\quad+\left(\frac{1}{2}\right)\left(\frac{1}{98}\right)^2\left[\left(\frac{1^3}{\frac{49}{25}}+2\cdot 1\right)\cdot 3\cdot 49\cdot 48+\left(\frac{1^3}{\frac{49}{25}}+2^3\right)3\cdot 49\right]\\
&\quad+\left(\frac{1}{98}\right)^3\left[(3\cdot 1)\cdot 49\cdot 48\cdot 47+\left(2^3+1\right)\cdot 3\cdot 49\cdot 48+\left(3^3\right)\cdot 49\right]\\
&= \frac{24183}{4802}\,.
\end{aligned}$$

On the other hand, consider the mixed assignment $\mathbf{P}$ where users 1 and 2 are assigned to link 1, and user 3 is assigned to all links $j \in [50] \setminus \{1\}$ with probability $p_{3j} = \frac{1}{49}$. Clearly, $\mathbf{P}$ is a Nash equilibrium with social cost

$$\mathsf{SC}_{x^3}(3,m,\mathbf{P}) \;=\; \frac{2^3}{\frac{49}{25}}+1 \;=\; \frac{249}{49} \;>\; \frac{24183}{4802} \;=\; \mathsf{SC}_{x^3}(3,m,\mathbf{F})\,.$$

This proves the claim. ■

4.10 Conclusion and Directions for Further Research

We gave a thorough analysis of a routing game introduced by Koutsoupias and Papadimitriou [93], widely known as the *KP-model*. Though it has received a lot of flourishing interest and attention resulting in many interesting results, some fundamental problems still remain tantalizingly open. We state only the most important of them:

- Though the FMNE Conjecture has been validated for numerous special cases, the FMNE Conjecture in general remains open.
- Up to now, no polynomial-time algorithm to compute a pure Nash equilibrium in the model of unrelated links is known. This problem appears to be intractable in the current state-of-the-art. However, finding such an algorithm for the special case of arbitrary users with restricted strategy sets and related links might be possible.
- Though the structure of the network in the KP-model is rather simple, it turned out to be adequate to investigate the influence of selfish behavior to the global performance of the system. A next step is to try to apply the gained results to prove results in more general settings, that is, general networks or/and general latency functions.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] M. Aigner. *Combinatorial Theory*. Springer-Verlag, 1979.

[3] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation Schemes for Scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 493–500, 1997.

[4] E. Anshelevich, A. Dasgupta, J. Kleinberg, T. Roughgarden, É. Tardos, and T. Wexler. The Price of Stability for Network Design with Fair Cost Allocation. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science (FOCS'04)*, pages 295–304, 2004.

[5] E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler. Near-Optimal Network Design with Selfish Agents. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 511–520, 2003.

[6] B. Awerbuch, Y. Azar, Y. Richter, and D. Tsur. Tradeoffs in Worst-Case Equilibria. In K. Jansen and R. Solis-Oba, editors, *Proceedings of the 1st International Workshop on Approximation and Online Algorithms (WAOA'03)*, Lecture Notes in Computer Science, Vol. 2909, Springer-Verlag, pages 41–52, 2003.

[7] Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. All-Norm Approximation Algorithms. *Journal of Algorithms*, 52(2):120–133, 2004.

[8] Y. Azar and O. Regev. Strongly Polynomial Algorithms for the Unsplittable Flow Problem. In K. Aardal and B. Gerards, editors, *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization (IPCO'01)*, Lecture Notes in Computer Science, Vol. 2081, Springer-Verlag, pages 15–29, 2001.

[9] B. S. Baker, D. J. Brown, and H. P. Katseff. A 5/4 Algorithm for Two-Dimensional Packing. *Journal of Algorithms*, 2(4):348–368, 1981.

[10] B. S. Baker, E. G. Coffman, and R. L. Rivest. Orthogonal Packings in Two Dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.

[11] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[12] M. J. Beckmann. On the Theory of Traffic Flow in Networks. *Traffic Quarterly*, 21:109–116, 1967.

[13] K. P. Belkhale and P. Banerjee. An Approximate Algorithm for the Partitionable Independent Task Scheduling Problem. In B.W. Wah, editor, *Proceedings of the 1990 International Conference on Parallel Processing (ICPP'90)*, Pennsylvania State University Press, pages 72–75, 1990.

[14] J. Blazewicz, M. Machowiak, G. Mounié, and D. Trystram. Approximation Algorithms for Scheduling Independent Malleable Tasks. In R. Sakellariou, J. Keane, J. R. Gurd, and L. Freeman, editors, *Proceedings of the 7th European Conference on Parallel Computing (Euro-Par'01)*, Lecture Notes in Computer Science, Vol. 2150, Springer-Verlag, pages 191–197, 2001.

[15] D. Braess. Über ein Paradoxon der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[16] P. Brucker. *Scheduling Algorithms*. Springer-Verlag, 2001.

[17] P. Brucker, J. Hurink, and F. Werner. Improving Local Search Heuristics for Some Scheduling Problems. Part II. *Discrete Applied Mathematics*, 72(1–2):47–69, 1997.

[18] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation Algorithms for the Unsplittable Flow Problem. In K. Jansen, S. Leonardi, and V. V. Vazirani, editors, *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'02)*, Lecture Notes in Computer Science, Vol. 2462, Springer-Verlag, pages 51–66, 2002.

[19] A. K. Chandra and C. K. Wong. Worst-Case Analysis of a Placement Algorithm Related to Storage Allocation. *SIAM Journal on Computing*, 4(3):249–263, 1975.

[20] C. K. Chau and K. M. Sim. The Price of Anarchy for Nom-Atomic Congestion Games with Symmetric Cost Maps and Elastic Demands. *Operations Research Letters*, 31(5):327–334, 2003.

[21] Y. Cho and S. Sahni. Bounds for List Schedules on Uniform Processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.

[22] R. Cole, Y. Dodis, and T. Roughgarden. How Much Can Taxes Help Selfish Routing? In *Proceedings of the 4th ACM Conference on Electronic Commerce (EC'03)*, pages 98–107, 2003.

[23] R. Cole, Y. Dodis, and T. Roughgarden. Pricing Network Edges for Heterogeneous Selfish Users. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 521–530, 2003.

[24] J. R. Correa, A. S. Schulz, and N. E. Stier Moses. Selfish Routing in Capacitated Networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.

[25] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. E. Santos, R. Subramonian, and T. von Eicken. LogP: A Practical Model for Parallel Computation. *Communications of the ACM*, 39(11):78–85, 1996.

[26] G. Cybenko. Dynamic Load Balancing for Distributed Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279–301, 1989.

[27] A. Czumaj. Selfish Routing on the Internet. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, 2004.

[28] A. Czumaj, P. Krysta, and B. Vöcking. Selfish Traffic Allocation for Server Farms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 287–296, 2002.

[29] A. Czumaj and B. Vöcking. Tight Bounds for Worst-Case Equilibria. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 413–420, 2002. Also accepted to *Journal of Algorithms* as Special Issue of SODA'02.

[30] S. C. Dafermos and F. T. Sparrow. The Traffic Assignment Problem for a General Network. *Journal of Research of the National Bureau of Standards, Series B*, 73(2):91–118, 1969.

[31] W. Fernandez de la Vega and V. Zissimopoulos. An Approximation Scheme for Strip Packing of Rectangles with Bounded Dimensions. *Discrete Applied Mathematics*, 82(1–3):93–101, 1998.

[32] T. Decker. Ein universelles Lastverteilungssystem und seine Anwendung bei der Isolierung reeller Nullstellen, PhD-thesis, University of Paderborn, 2000.

[33] T. Decker and W. Krandick. Parallel Real Root Isolation Using the Descartes Method. In *Proceedings of the 6th International Conference on High Performance Computing (HiPC'99)*, Lecture Notes in Computer Science, Vol. 1745, Springer-Verlag, pages 261–268, 1999.

[34] T. Decker, T. Lücking, and B. Monien. A 5/4-Approximation Algorithm for Scheduling Identical Malleable Tasks. In K. Jansen and R. Solis-Oba, editors, *Proceedings of the 1st International Workshop on Approximation and Online Algorithms (WAOA'03)*, Lecture Notes in Computer Science, Vol. 2909, Springer-Verlag, pages 95–108, 2003.

[35] R. Diekmann, A. Frommer, and B. Monien. Efficient Schemes for Nearest Neighbor Load Balancing. *Parallel Computing*, 25(7):789–812, 1999.

[36] R. Diekmann, F. Schlimbach, and C. Walshaw. Quality Balancing for Parallel Adaptive FEM. In A. Ferreira, J. D. P. Rolim, H. D. Schmidt, and S.-H. Teng, editors, *Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel (IRREGULAR'98)*, Lecture Notes in Computer Science, Vol. 1457, Springer-Verlag, pages 170–181, 1998.

[37] Y. Dinitz, N. Garg, and M. X. Goemans. On the Single-Source Unsplittable Flow Problem. *Combinatorica*, 19(1):17–41, 1999.

[38] G. Dobson. Scheduling Independent Tasks on Uniform Processors. *SIAM Journal on Computing*, 13(4):705–716, 1984.

[39] J. Du and Y.-T. Leung. Complexity of Scheduling Parallel Task Systems. *SIAM Journal on Discrete Mathematics*, 2(4):473–487, 1989.

[40] H. Dyckhoff. A Typology of Cutting and Packing Problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[41] R. Elsässer, B. Monien, and R. Preis. Diffusive Schemes for Load Balancing on Heterogeneous Networks. *Theory of Computing Systems*, 35(3):305–320, 2002.

[42] E. Even-Dar, A. Kesselmann, and Y. Mansour. Convergence Time to Nash Equilibria. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP'03)*, Lecture Notes in Computer Science, Vol. 2719, Springer-Verlag, pages 502–513, 2003.

[43] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC'03)*, pages 347–351, 2003.

[44] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The Complexity of Pure Nash Equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 604–612, 2004.

[45] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

[46] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the Coordination Ratio for a Selfish Routing Game. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP'03)*, Lecture Notes in Computer Science, Vol. 2719, Springer-Verlag, pages 514–526, 2003.

[47] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Selfish Routing in Non-Cooperative Networks: A Survey. In B. Rovan and P. Vojtás, editors, *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, Lecture Notes in Computer Science, Vol. 2747, Springer-Verlag, pages 21–45, 2003.

[48] G. Finn and E. Horowitz. A Linear Time Approximation Algorithm for Multiprocessor Scheduling. *BIT*, 19(3):312–320, 1979.

[49] L. Fleischer. Linear Tolls Suffice: New Bounds and Algorithms for Tolls in Single Source Networks. In J. Diaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP'04)*, Lecture Notes in Computer Science, Vol. 3142, Springer-Verlag, pages 544–554, 2004.

[50] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP'02)*, Lecture Notes in Computer Science, Vol. 2380, Springer-Verlag, pages 123–134, 2002.

[51] D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish Unsplittable Flows. In J. Diaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP'04)*, Lecture Notes in Computer Science, Vol. 3142, Springer-Verlag, pages 593–605, 2004.

[52] G. C. Fox, R. D. Williams, and P. C. Messina. *Parallel Computing Works!* Morgan Kaufmann Publishers, 1994.

[53] D. K. Friesen. Tighter Bounds for the Multifit Processor Scheduling Algorithm. *SIAM Journal on Computing*, 13(1):170–181, 1984.

[54] D. K. Friesen. Tighter Bounds for LPT Scheduling on Uniform Processors. *SIAM Journal on Computing*, 16(3):554–560, 1987.

[55] D. K. Friesen and M. A. Langston. Bounds for Multifit Scheduling on Uniform Processors. *SIAM Journal on Computing*, 12(1):60–70, 1983.

[56] M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. Computing Nash Equilibria for Scheduling on Restricted Parallel Links. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 613–622, 2004.

[57] M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. The Price of Anarchy for Polynomial Social Cost. In J. Fiala, V. Koubek, and J. Kratochvíl, editors, *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS'04)*, Lecture Notes in Computer Science, Vol. 3153, Springer-Verlag, pages 574–585, 2004.

[58] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. Nash Equilibria in Discrete Routing Games with Convex Latency Functions. In J. Diaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP'04)*, Lecture Notes in Computer Science, Vol. 3142, Springer-Verlag, pages 645–657, 2004.

[59] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and P. Spirakis. Extreme Nash Equilibria. In C. Blundo and C. Laneve, editors, *Proceedings of the 8th Italian Conference on Theoretical Computer Science (ICTCS'03)*, Lecture Notes in Computer Science, Vol. 2841, Springer-Verlag, pages 1–20, 2003. Also accepted to *Theoretical Computer Science*, Special Issue on *Game Theory Meets Theoretical Computer Science*.

[60] M. R. Garey and R. L. Graham. Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM Journal on Computing*, 4(2):187–200, 1975.

[61] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[62] G. H. Gonnet. Expected Length of the Longest Probe Sequence in Hash Code Searching. *Journal of the ACM*, 28(2):289–304, 1981.

[63] T. Gonzalez, O. H. Ibarra, and S. Sahni. Bounds for LPT Schedules on Uniform Processors. *SIAM Journal on Computing*, 6(1):155–166, 1977.

[64] R. L. Graham. Bounds for Certain Multiprocessing Anomalies. *The Bell System Technical Journal*, 45(1):1563–1581, 1966.

[65] R. L. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[66] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1994.

[67] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and Better Approximation Algorithms for Network Design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 365–372, 2003.

[68] J. Havil. *Gamma: Exploring Euler's Constant*. Princeton University Press, 2003.

[69] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[70] D. S. Hochbaum and D. B Shmoys. Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results. *Journal of the ACM*, 34(1):144–162, 1987.

[71] D. S. Hochbaum and D. B. Shmoys. A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[72] J. E. Hopcroft and R. M. Karp. An $n^{\frac{5}{2}}$ Algorithm for Maximum Matching in Bipartite Graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[73] E. Horowitz and S. Sahni. Exact and Approximate Algorithms for Scheduling Nonidentical Processors. *Journal of the ACM*, 23(2):317–327, 1976.

[74] O. Jahn, R. H. Möhring, A. S. Schulz, and N. E. Stier Moses. System-Optimal Routing of Traffic Flows with User Constraints in Networks with Congestion. Technical report, MIT Sloan School of Management Working Paper No. 4394-02, 2002.

[75] K. Jain and V. Vazirani. Applications of Approximation Algorithms to Cooperative Games. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 364–372, 2001.

[76] K. Jansen. Scheduling Malleable Parallel Tasks: An Asymptotic Fully Polynomial-Time Approximation Scheme. *Algorithmica*, 39(1):59–81, 2004.

[77] K. Jansen and L. Porkolab. Linear-Time Approximation Schemes for Scheduling Malleable Parallel Tasks. *Algorithmica*, 32(3):507–520, 2002.

[78] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How Easy is Local Search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[79] C. Kenyon and E. Rémila. Approximate Strip Packing. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS'96)*, pages 31–36, 1996.

[80] S. Khanna, S. Muthukrishnan, and M. Paterson. On Approximating Rectangular Tiling and Packing. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 384–393, 1998.

[81] J. Kleinberg. Single-Source Unsplittable Flow. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS'96)*, pages 68–77, 1996.

[82] J. Kleinberg, Y. Rabani, and É. Tardos. Fairness in Routing and Load Balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001.

[83] S. G. Kolliopoulos. Exact and Approximation Algorithms for Network Flow and Disjoint-Path Problems, PhD-thesis, Dartmouth College, 1998.

[84] S. G. Kolliopoulos and C. Stein. Improved Approximation Algorithms for Unsplittable Flow Problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 426–435, 1997.

[85] S. G. Kolliopoulos and C. Stein. Experimental Evaluation of Approximation Algorithms for Single-Source Unsplittable Flow. In G. Cornuéjols, R. E. Burkhard, and G. J. Woeginger, editors, *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization (IPCO'99)*, Lecture Notes in Computer Science, Vol. 1610, Springer-Verlag, pages 328–344, 1999.

[86] S. G. Kolliopoulos and C. Stein. Approximation Algorithms for Single-Source Unsplittable Flow. *SIAM Journal on Computing*, 31(3):919–946, 2002.

[87] P. Kolman and C. Scheideler. Improved Bounds for the Unsplittable Flow Problem. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 184–193, 2002.

[88] Y. A. Korilis, A. A. Lazar, and A. Orda. Architecting Noncooperative Networks. *IEEE Journal on Selected Areas in Communications*, 13(7):1241–1251, 1995.

[89] Y. A. Korilis, A. A. Lazar, and A. Orda. The Role of the Manager in a Noncooperative Network. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'96)*, pages 1285–1293, 1996.

[90] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving Network Optima Using Stackelberg Routing Strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.

[91] E. Koutsoupias. Selfish Task Allocation. *Bulletin of the EATCS*, 81:79–88, 2003.

[92] E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate Equilibria and Ball Fusion. *Theory of Computing Systems*, 36(6):683–693, 2003.

[93] E. Koutsoupias and C. Papadimitriou. Worst-Case Equilibria. In C. Meinel and S. Tison, editors, *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS'99)*, Lecture Notes in Computer Science, Vol. 1563, Springer-Verlag, pages 404–413, 1999.

[94] R. Krishnamurti and E. Ma. The Processor Partitioning Problem in Special-Purpose Partitionable Systems. In *Proceedings of the 1988 International Conference on Parallel Processing (ICPP'88), Vol. 1*, pages 434–443, 1988.

[95] H. W. Kuhn and S. Nasar eds. *The Essential John Nash*. Princeton University Press, 2002.

[96] V. S. A. Kumar and M. V. Marathe. Improved Results for Stackelberg Scheduling Strategies. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP'02)*, Lecture Notes in Computer Science, Vol. 2380, Springer-Verlag, pages 776–787, 2002.

[97] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. *Mathematical Programming*, 46:259–271, 1990.

[98] J. Y.-T. Leung and W.-D. Wei. Tighter Bounds on a Heuristic for a Partition Problem. *Information Processing Letters*, 56(1):51–57, 1995.

[99] L. Libman and A. Orda. The Designer's Perspective to Atomic Noncooperative Networks. *IEEE/ACM Transactions on Networking*, 7(6):875–884, 1999.

[100] L. Libman and A. Orda. Atomic Resource Sharing in Noncooperative Networks. *Telecommunication Systems*, 17(4):385–409, 2001.

[101] H. Lin, T. Roughgarden, and É. Tardos. A Stronger Bound on Braess's Paradox. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 340–341, 2004.

[102] T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. A New Model for Selfish Routing. In V. Diekert and M. Habib, editors, *Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS'04)*, Lecture Notes in Computer Science, Vol. 2996, Springer-Verlag, pages 547–558, 2004.

[103] T. Lücking, M. Mavronicolas, B. Monien, M. Rode, P. Spirakis, and I. Vrto. Which is the Worst-Case Nash Equilibrium? In B. Rovan and P. Vojtás, editors, *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, Lecture Notes in Computer Science, Vol. 2747, Springer-Verlag, pages 551–561, 2003.

[104] T. Lücking, B. Monien, and M. Rode. On the Problem of Scheduling Flows on Distributed Networks. In K. Diks and W. Rytter, editors, *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS'02)*, Lecture Notes in Computer Science, Vol. 2420, Springer-Verlag, pages 495–505, 2002.

[105] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[106] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[107] M. Mavronicolas and P. Spirakis. The Price of Selfish Routing. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 510–519, 2001.

[108] 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.

[109] I. Milchtaich. Congestion Games with Player-Specific Payoff Functions. *Games and Economic Behavior*, 13(1):111–124, 1996.

[110] D. Monderer and L. S. Shapley. Potential Games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[111] B. Monien. Algorithmische Spieltheorie, Vorlesungsskript, Universität Paderborn, 2004.

[112] N. E. Stier Moses. Selfish Versus Coordinated Routing in Network Games, PhD-thesis, Slaon School of Management, MIT, 2004.

[113] G. Mounié, C. Rapine, and D. Trystram. Efficient Approximation Algorithms for Scheduling Malleable Tasks. In *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'99)*, pages 23–32, 1999.

[114] G. Mounié, C. Rapine, and D. Trystram. A $\frac{3}{2}$-Approximation Algorithm for Independent Scheduling Malleable Tasks. *Submitted for publication*, 2001.

[115] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[116] E. Naroska and U. Schwiegelshohn. On an On-Line Scheduling Problem for Parallel Jobs. *Information Processing Letters*, 81(6):297–304, 2002.

[117] J. F. Nash. Equilibrium Points in *n*-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.

[118] J. F. Nash. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

[119] N. Nisan. Algorithms for Selfish Agents. In C. Meinel and S. Tison, editors, *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS'99)*, Lecture Notes in Computer Science, Vol. 1563, Springer-Verlag, pages 1–15, 1999.

[120] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 129–140, 1999.

[121] A. Orda, R. Rom, and N. Shimkin. Competitive Routing in Multiuser Communication Networks. *IEEE/ACM Transactions on Networking*, 1(5):510–521, 1993.

[122] M. J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2004.

[123] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[124] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 749–753, 2001.

[125] A. C. Pigou. *The Economics of Welfare*. Macmillan and Company, 1920.

[126] M. Rode. Nash Equilibria in Discrete Routing Games, PhD-thesis, University of Paderborn, 2004.

[127] R. W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[128] R. W. Rosenthal. The Network Equilibrium Problem in Integers. *Networks*, 3:53–59, 1973.

[129] T. Roughgarden. Designing Networks for Selfish Users is Hard. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 472–481, 2001.

[130] T. Roughgarden. How Unfair is Optimal Routing. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 203–204, 2002.

[131] T. Roughgarden. Selfish Routing, PhD-thesis, Cornell University, 2002.

[132] T. Roughgarden. The Price of Anarchy is Independent of the Network Topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003.

[133] T. Roughgarden. Stackelberg Scheduling Strategies. *SIAM Journal on Computing*, 33(2):332–350, 2004.

[134] T. Roughgarden. The Maximum Latency of Selfish Routing. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 980–981, 2004.

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

[136] T. Roughgarden and É. Tardos. Bounding the Inefficiency of Equilibria in Nonatomic Congestion Games. *Games and Economic Behaviour*, 47(2):389–403, 2004.

[137] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[138] P. Schuurman and T. Vredeveld. Performance Guarantees of Load Search for Multiprocessor Scheduling. In K. Aardal and B. Gerards, editors, *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization (IPCO'01)*, Lecture Notes in Computer Science, Vol. 2081, Springer-Verlag, pages 370–382, 2001.

[139] M. Skutella. Approximating the Single Source Unsplittable Min-Cost Flow Problem. *Mathematical Programming, Series B*, 91(3):493–514, 2002.

[140] A. Steinberg. A Strip-Packing Algorithm with Absolute Performance Bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.

[141] D. Trystram. Personal Communication, 2002.

[142] J. Turek, J. L. Wolf, and P. S. Yu. Approximate Algorithms for Scheduling Parallelizable Tasks. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'92)*, pages 323–332, 1992.

[143] B. Veltman, B. J. Lageweg, and J. K. Lenstra. Multiprocessor Scheduling with Communication Delays. *Parallel Computing*, 16(2–3):173–182, 1990.

[144] C. Walshaw, M. Cross, and M. Everett. Dynamic Load-Balancing for Parallel Adaptive Unstructured Meshes. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing (PPSC'97)*, 1997.

[145] J. G. Wardrop. Some Theoretical Aspects of Road Traffic Research. In *Proceedings of the Institute of Civil Engineers, Pt. II, Vol. 1*, pages 325–378, 1956.

Index