Selfish Load Balancing

Berthold Vöcking

Abstract

Suppose that a set of weighted tasks shall be assigned to a set of machines with possibly different speeds such that the load is distributed evenly among the machines. In computer science, this problem is traditionally treated as an optimization problem. One of the classical objectives is to minimize the *makespan*, i.e., the maximum load over all machines. Here we study a natural game theoretic variant of this problem: We assume that the tasks are managed by selfish agents, i.e., each task has an agent that aims at placing the task on the machine with smallest load. We study the Nash equilibria of this game and compare them with optimal solutions with respect to the makespan. The ratio between the worst-case makespan in a Nash equilibrium and the optimal makespan is called the *price of anarchy*. In this chapter, we study the price of anarchy for such load balancing games in four different variants, and we investigate the complexity of computing equilibria.

20.1 Introduction

The problem of load balancing is fundamental to networks and distributed systems. Whenever a set of tasks should be executed on a set of resources, one needs to balance the load among the resources in order to exploit the available resources efficiently. Often also fairness aspects have to be taken into account. Load balancing has been studied extensively and in many variants. One of the most fundamental load balancing problems is *makespan scheduling on uniformly related machines*. This problem is defined by *m machines* with speeds $s_1, \ldots, s_m$ and *n tasks* with weights $w_1, \ldots, w_n$. Let $[n] = \{1, \ldots, n\}$ denote the set of tasks and $[m] = \{1, \ldots, m\}$ the set of machines. One seeks for an assignment $A : [n] \to [m]$ of the tasks to the machines that is as balanced as possible. The *load* of machine $j \in [m]$ under assignment A is defined as

$$\ell_j = \sum_{\substack{i \in [n] \\ j = A(i)}} \frac{w_i}{s_j}.$$

The *makespan* is defined to be the maximum load over all machines. The objective is to minimize the makespan. If all machines have the same speed, then the problem is

known as *makespan scheduling on identical machines*, in which case we shall assume $s_1 = s_2 = \cdots = s_m = 1$.

In computer science, load balancing is traditionally viewed as an algorithmic problem. We design and analyze algorithms, either centralized or distributed, that compute the mapping A. Suppose, however, there is no global authority that can enforce an efficient mapping of the tasks to the machines. For example, in a typical Internet application, tasks might correspond to requests for downloading large files that users send to servers. To maximize the quality of service, each of the users aims at contacting a server with smallest load. This naturally leads to the following game theoretic setting in which we will be able to analyze what happens to the makespan if there is no global authority but selfish users aiming at maximizing their individual benefit decide about the assignment of tasks to machines.

This chapter differs from the other chapters in Part III of this book in two important aspects. At first, the considered objective function, the makespan, is nonutilitarian. At second, our analysis does not only consider pure but also mixed equilibria. By using the makespan as objective function, our analysis simultaneously captures the aspects of efficiency and fairness. By considering mixed equilibria, our analysis explicitly takes into account the effects of uncoordinated random behavior.

20.1.1 Load Balancing Games

We identify agents and tasks, i.e., task $i \in [n]$ is managed by agent i. Each agent can place its task on one of the machines. In other words, the set of *pure strategies* for an agent is $[m]$. A combination of pure strategies, one for each task, yields an assignment $A : [n] \to [m]$. We assume that the *cost* of agent i under the assignment A corresponds to the load on machine $A(i)$, i.e., its cost is $\ell_{A(i)}$. The *social cost* of an assignment is denoted cost(A) and is defined to be the makespan, i.e., $\text{cost}(A) = \max_{j \in [m]} \left(\ell_j \right)$.

Agents may use *mixed strategies*, i.e., probability distributions on the set of pure strategies. Let p_i^j denote the probability that agent i assigns its task to machine j, i.e., $p_i^j = \mathbb{P}[A(i) = j]$. A *strategy profile* $P = (p_i^j)_{i \in [n], j \in [m]}$ specifies the probabilities for all agents and all machines. Clearly, every strategy profile P induces a random mapping A. For $i \in [n]$, $j \in [m]$, let x_i^j be a random variable that takes the value 1 if $A(i) = j$ and 0, otherwise. The expected load of machine j under the strategy profile P is thus

$$\mathbb{E}[\ell_j] = \mathbb{E}\left[\sum_{i \in [n]} \frac{w_i \, x_i^j}{s_j} \right] = \sum_{i \in [n]} \frac{w_i \, \mathbb{E}[x_i^j]}{s_j} = \sum_{i \in [n]} \frac{w_i \, p_i^j}{s_j}.$$

The *social cost of a strategy profile* P is defined as the expected makespan, i.e.,

$$\text{cost}(P) = \mathbb{E}[\text{cost}(A)] = \mathbb{E}\left[\max_{j \in [m]} \left(\ell_j \right) \right].$$

We assume that every agent aims at minimizing its expected cost. From point of view of agent i, the expected cost on machine j, denoted by c_i^j, is $c_i^j = \mathbb{E}[\ell_j \mid A(i) = j]$.

For any profile P,

$$c_i^j = \frac{w_i + \sum_{k \neq i} w_k\, p_k^j}{s_j} \;=\; \mathbb{E}[\ell_j] + (1 - p_i^j) \cdot \frac{w_i}{s_j}. \tag{20.1}$$

In general, a strategy profile of a game is a *Nash equilibrium* if there is no incentive for any agent to unilaterally change its strategy. For the load balancing game, such a profile is characterized by the property that every agent assigns positive probabilities only to those machines that minimize its expected cost. This is formalized as follows.

Proposition 20.1 *A strategy profile P is a Nash equilibrium if and only if* $\forall i \in [n] : \forall j \in [m] : p_i^j > 0 \Rightarrow \forall k \in [m] : c_i^j \leq c_i^k.$

The existence of a Nash equilibrium in mixed strategies is guaranteed by the theorem of Nash, see Chapters 1 and 2. A strategy profile P is called *pure* if, for each agent, there exists only one machine with positive probability. A Nash equilibrium in pure strategies is called a *pure Nash equilibrium.* Applying the proposition above to pure profiles and the corresponding assignments yields the following characterization of a pure Nash equilibrium.

Proposition 20.2 *An assignment A is a pure Nash equilibrium if and only if* $\forall i \in [n] : \forall k \in [m] : c_i^{A(i)} \leq c_i^k.$

In words, an assignment is a pure Nash equilibrium if and only if no agent can improve its cost by unilaterally moving its task to another machine. A special property of load balancing games is that they always admit pure Nash equilibria.

Proposition 20.3 *Every instance of the load balancing game admits at least one pure Nash equilibrium.*

PROOF An assignment A induces a *sorted load vector* $(\lambda_1, \ldots, \lambda_m)$, where λ_j denotes the load on the machine that has the j-th highest load. If an assignment is not a Nash equilibrium, then there exists an agent i that can perform an *improvement step*, i.e., it can decrease its cost by moving its task to another machine. We show that the sorted load vector obtained after performing an improvement step is lexicographically smaller than the one preceding it. Hence, a pure Nash equilibrium is reached after a finite number of improvement steps.

Suppose, given any sorted load vector $(\lambda_1, \ldots, \lambda_m)$, agent i performs an improvement step and moves its task from machine j to machine k where the indices are with respect to the positions of the machines in the sorted load vector. Clearly, $k > j$. The improvement step decreases the load on machine j and it increases the load on machine k. However, the increased load on machine k is smaller than λ_j as, otherwise, agent i would not decrease its cost. Hence, the number of machines with load at least λ_j is decreasing. Furthermore, the loads on all other machines with load at least λ_j are left unchanged. Consequently, the improvement step yields a sorted load vector i.e. lexicographically smaller than $(\lambda_1, \ldots, \lambda_m)$. $\square$

Thus improvement steps naturally lead to a pure Nash equilibrium. This issue is also discussed for a broader class of games, so-called potential games, in Chapter 19. Let us remark that this convergence result implies that there exists even a pure Nash equilibrium that minimizes the makespan. Given any optimal assignment, such an equilibrium can be found by performing improvement steps until a Nash equilibrium is reached because improvement steps do not increase the makespan. Thus, for load balancing games with social cost equal to the makespan, it does not make much sense to study the ratio between the social cost in a best Nash equilibrium and the optimal social cost. This ratio is called the "price of stability." It is studied in Chapters 17–19 in the context of other games. In this chapter, we are mainly interested in the ratio between the social cost of the worst Nash equilibrium and the optimal social cost, the so-called the "price of anarchy."

20.1.2 Example of a Load Balancing Game

Suppose that there are two identical machines both of which have speed 1 and four tasks, two *small tasks* of weight 1 and two *large tasks* of weight 2. An optimal assignment would map a small and a large task to each of the machines so that the load on both machines is 3. This assignment is illustrated in Figure 20.1(a).

Now consider an assignment A that maps the two large tasks to the first machine and the two small tasks to the second machine as illustrated in Figure 20.1(b). This way, the first machine has a load of 4 and the second machine has a load of 2. Obviously, a small task cannot improve its cost by moving from the second to the first machine. A large task cannot improve its cost by moving from the first to the second machine either as its cost would remain 4 if it does. Thus assignment A constitutes a pure Nash equilibrium with $cost(A) = 4$. Observe that all assignments that yield a larger makespan than 4 cannot be a Nash equilibrium as, in this case, one of the machines has a load of at least 5 and the other has a load of at most 1 so that moving any task from the former to the latter would decrease the cost of this task. Thus, for this instance of the load balancing game, the social cost of the worst pure Nash equilibrium is 4.

Clearly, the worst mixed equilibrium cannot be better than the worst pure equilibrium as the set of mixed equilibria is a superset of the set of pure equilibria, but can it really be worse? Suppose that each task is assigned to each of the machines with probability
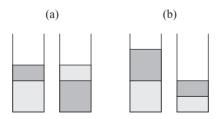

Figure 20.1. (a) Illustration of the optimal assignment of an instance of the load balancing game with two large tasks of size 2 and two small tasks of size 1 as described in the example given in Section 20.1.2. The social cost of this assignment is 3. (b) Illustration of a pure Nash equilibrium for the same instance. The social cost of this assignment is 4, which is the maximum among all pure Nash equilibria for this instance.

$\frac{1}{2}$. This corresponds to a strategy profile P with $p_i^j = \frac{1}{2}$ for $1 \leq i \leq 4, 1 \leq j \leq 2$. The expected load on machine j is thus

$$\mathbb{E}[\ell_j] = \sum_{1 \leq i \leq 4} w_i \, p_i^j = 2 \cdot 2 \cdot \frac{1}{2} + 2 \cdot 1 \cdot \frac{1}{2} = 3.$$

It is important to notice that the expected cost of a task on a machine is larger than the expected load of the machine, unless the task is assigned with probability 1 to this machine. For example, if we assume that task 1 is a large task then Equation 20.1 yields

$$c_1^1 = \mathbb{E}[\ell_1] + (1 - p_1^1) \, w_1 = 3 + \frac{1}{2} \cdot 2 = 4,$$

and, if task 3 is a small task, then

$$c_3^1 = \mathbb{E}[\ell_1] + (1 - p_3^1) \, w_3 = 3 + \frac{1}{2} \cdot 1 = 3.5.$$

For symmetry reasons, the expected cost of each task under the considered strategy profile P is the same on both machines so that P is a Nash equilibrium. The social cost of this Nash equilibrium, cost(P), is defined to be the expected makespan, $\mathbb{E}[\text{cost}(A)]$, of the random assignment A induced by P. The makespan, cost(A), is a random variable. This variable can possibly take one of the four values 3, 4, 5, or 6. There are $2^4 = 16$ different assignments of four tasks to two machines. The number of assignments that yield a makespan of 3 is 4, 4 is 6, 5 is 4, and 6 is 2. Consequently, the social cost of the mixed Nash equilibrium is

$$\text{cost}(P) = \mathbb{E}[\text{cost}(A)] = \frac{1}{16} \, (3 \cdot 4 + 4 \cdot 6 + 5 \cdot 4 + 6 \cdot 2) = 4.25.$$

Thus mixed equilibria can, in fact, be worse than the worst pure equilibrium.

20.1.3 Definition of the Price of Anarchy

Not surprisingly, the example above shows that uncoordinated, selfish behavior can lead to suboptimal assignments. We are interested in the ratio between the social cost (makespan) of a worst-case Nash equilibrium, i.e., the Nash equilibrium with highest social cost, and the social cost of an optimal assignment.

Definition 20.4 (Price of anarchy)   For $m \in \mathbb{N}$, let $\mathcal{G}(m)$ denote the set of all instances of load balancing games with m machines. For $G \in \mathcal{G}(m)$, let Nash(G) denote the set of all strategy profiles being a Nash equilibrium for G, and let opt(G) denote the minimum social cost over all assignments. Then the price of anarchy is defined by

$$\text{PoA}(m) = \max_{G \in \mathcal{G}(m)} \max_{P \in \text{Nash}(G)} \frac{\text{cost}(P)}{\text{opt}(G)}.$$

In the following, we study the price of anarchy in load balancing games in four different variants in which we distinguish, as a first criterion, between games with identical and uniformly related machines and, as a second criterion, between pure

Nash equilibria and mixed Nash equilibria. Technically, when considering the price of anarchy for load balancing games with identical machines then we restrict the set $\mathcal{G}(m)$ to instances in which the m machines have all the same speed. When considering the price of anarchy with respect to pure equilibria then the set Nash(G) refers only to pure Nash equilibria rather than mixed equilibria; i.e., we take the maximum only among pure equilibrium assignments rather than among possibly mixed equilibrium strategy profiles.

   The motivation behind studying the price of anarchy is to quantify the increase of the social cost due to selfish behavior. With this motivation in mind, does it make more sense to consider pure or mixed equilibria? If one wants to study a distributed system in which agents repeatedly perform improvement steps until they reach a Nash equilibrium, then pure equilibria are the right solution concept. However, there might be other means by which agents come to a Nash equilibrium. In particular, if one views load balancing games as one shot games, then mixed equilibria are a very reasonable solution concept. Moreover, upper bounds about the price of anarchy for mixed equilibria are more robust than upper bounds for pure equilibria as mixed equilibria are more general than pure ones. In this chapter, we consider both of these equilibrium concepts. Our analysis begins with the study of pure equilibria as they are usually easier to handle than mixed equilibria whose analysis requires a bit of probability theory.

20.2 Pure Equilibria for Identical Machines

Our analysis of equilibria in load balancing games begins with the most basic case, namely the study of pure equilibria on identical machines. Our first topic is the price of anarchy. As a second topic, we investigate how long it takes until a pure Nash equilibrium is reached when the agents repeatedly perform "best response" improvement steps.

20.2.1 The Price of Anarchy

In case of pure equilibria and identical machines, the analysis of the price of anarchy is quite similar to the well-known analysis of the greedy load balancing algorithm that assigns the tasks one after the other in arbitrary order giving each task to the least loaded machine. Graham (1966) has shown that the approximation factor of the greedy algorithm is $2 - \frac{1}{m}$. We show that the price of anarchy for pure equilibria is, in fact, slightly better than the approximation factor of the greedy algorithm.

   Theorem 20.5   *Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m identical machines. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then, it holds that*

$$cost(A) \leq \left(2 - \frac{2}{m+1}\right) \cdot opt(G).$$

PROOF Let j^* be a machine with the highest load under assignment A, and let i^* be a task of smallest weight assigned to this machine. Without loss of generality, there are at least two tasks assigned to machine j^* as, otherwise, $\text{cost}(A) = \text{opt}(G)$ so that the upper bound given in the theorem follows trivially. Thus $w_{i^*} \leq \frac{1}{2}\text{cost}(A)$.

Suppose there is a machine $j \in [n] \setminus \{j^*\}$ with load less than $\ell_{j^*} - w_{i^*}$. Then moving the task i^* from j^* to j would decrease the cost for this task. Hence, as A is a Nash equilibrium, it holds

$$\ell_j \geq \ell_{j^*} - w_{i^*} \geq \text{cost}(A) - \frac{1}{2}\text{cost}(A) = \frac{1}{2}\text{cost}(A).$$

Now observe that the cost of an optimal assignment cannot be smaller than the average load over all machines so that

$$\begin{aligned}
\text{opt}(G) &\geq \frac{\sum_{i \in [n]} w_i}{m} \\
&= \frac{\sum_{j \in [m]} \ell_j}{m} \\
&\geq \frac{\text{cost}(A) + \frac{1}{2}\text{cost}(A)(m-1)}{m} \\
&= \frac{(m+1)\text{cost}(A)}{2m}.
\end{aligned}$$

As a consequence,

$$\text{cost}(A) \leq \frac{2m}{m+1} \cdot \text{opt}(G) = \left(2 - \frac{2}{m+1}\right) \cdot \text{opt}(G).$$

$\square$

Observe that the example of a game instance with two identical machines given in Section 20.1.2 has a price of anarchy of $\frac{4}{3} = 2 - \frac{2}{m+1}$, for $m = 2$. Exercise 20.2 generalizes this example. It shows that, for every $m \in \mathbb{N}$, there exists an instance G of the load balancing game with m identical machines and $2m$ tasks that has a Nash equilibrium assignment $A : [n] \to [m]$ with

$$\text{cost}(A) = \left(2 - \frac{2}{m+1}\right) \cdot \text{opt}(G).$$

Thus the upper bound on the price of anarchy given in Theorem 20.5 is tight.

20.2.2 Convergence Time of Best Responses

Our analysis about the price of anarchy leaves open the question of how agents may find or compute a Nash equilibrium efficiently. In the existence proof for pure equilibria in Proposition 20.3, we have implicitly shown that every sequence of improvement steps by the agents leads to a Nash equilibrium. However, if players do not converge to an equilibrium in reasonable time, then it might also not matter if the finally reached equilibrium is good. This naturally leads to the question of how many improvement steps are needed to reach a Nash equilibrium. The following result shows that, in case

of identical machines, there is a short sequence of improvement steps that leads from any given initial assignment to a pure Nash equilibrium. An agent is said to be *satisfied* if it cannot reduce its cost by unilaterally moving its task to another machine. The *max-weight best response policy* activates the agents one after the other always activating an agent with maximum weight among the unsatisfied agents. An activated agent plays a *best response*; i.e., the agent moves its task to the machine with minimum load.

Theorem 20.6   *Let $A : [n] \to [m]$ denote any assignment of n tasks to m identical machines. Starting from A, the max-weight best response policy reaches a pure Nash equilibrium after each agent was activated at most once.*

PROOF   We claim, once an agent $i \in [n]$ was activated and played its best response, it never gets unsatisfied again. This claim immediately implies the theorem. Our analysis starts with two observations both of which holding only for identical machines. At first, we observe that an agent is satisfied if and only if its task is placed on a machine on which the load due to the other tasks is minimal. At second, we observe that a best response never decreases the minimum load among the machines. As a consequence, a satisfied agent can get unsatisfied only for one reason: the load on the machine holding its task increases because another agent moves its task to the same machine. Suppose that agent k is activated after agent i, and it moves its task to the machine holding task i. Let j^* denote the machine on which i is placed and to which k is moved. For $j \in [m]$, let ℓ_j denote the load on machine j at the time immediately after the best response of agent k. Since the assignment of k to j^* is a best response and as $w_k \leq w_i$ because of the max-weight policy, it follows

$$\ell_{j^*} \leq \ell_j + w_k \leq \ell_j + w_i,$$

for all $j \in [m]$. Hence, after the best response of k, agent i remains satisfied on machine j^* as it cannot reduce its cost by moving from j^* to any other machine. □

Let us remark that the order in which the agents are activated is crucial. For example, if one would always activate an agent of minimum weight among the unsatisfied agents, then there are instances of load balancing games on identical machines where one needs an exponential number of best response steps to reach a pure Nash equilibrium (Even-Dar et al., 2003).

20.3 Pure Equilibria for Uniformly Related Machines

We now switch from identical to uniformly related machines. First, we study the price of anarchy. Then we discuss the complexity of computing equilibria.

20.3.1 The Price of Anarchy

The analysis in Section 20.2.1 shows that, in case of identical machines, the makespan of a pure Nash equilibrium is less than twice the optimal makespan. In this section, we
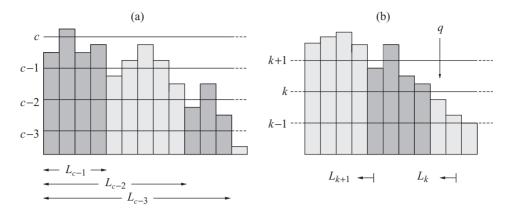
Figure 20.2. (a) Illustration of the definition of the lists $L_{c-1}, L_{c-2}, \ldots, L_0$ from the proof of Theorem 20.7. (b) Illustration of the lists L_k and L_{k+1} and the machine q used in the proof of Lemma 20.8.

show that the makespan of pure equilibria on uniformly related machines can deviate by more than a constant factor. The price of anarchy is bounded, however, by a slowly growing function in the number of machines. Our analysis begins with an upper bound on the price of anarchy followed by the presentation of a family of game instances that match this upper bound up to a small constant factor.

Theorem 20.7 *Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m machines of speed $s_1, \ldots, s_m$. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then, it holds that*

$$cost(A) = \mathcal{O}\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

PROOF Let $c = \lfloor cost(A)/opt(G) \rfloor$. We show $c \leq \Gamma^{-1}(m)$, where Γ^{-1} denotes the inverse of the *gamma function*, an extension of the factorial function with the property that $\Gamma(k) = (k-1)!$, for every positive integer k. This yields the theorem as

$$\Gamma^{-1}(m) = \Theta\left(\frac{\log m}{\log \log m}\right).$$

Without loss of generality, let us assume $s_1 \geq s_2 \geq \cdots \geq s_m$, and let $L = [1, 2, \ldots, m]$ denote the list of machines in nonincreasing order of speed. For $k \in \{0, \ldots, c-1\}$, let L_k denote the maximum length prefix of L such that the load of each server in L_k is at least $k \cdot opt(G)$. Figure 20.2(a) illustrates this definition. We will show the following recurrence on the lengths of these lists.

$$|L_k| \geq (k+1) \cdot |L_{k+1}| \qquad (0 \leq k \leq c-2)$$
$$|L_{c-1}| \geq 1$$

Solving the recurrence yields $|L_0| \geq (c - 1)! = \Gamma(c)$. Now observe that $L_0 = L$ and, hence, $|L_0| = m$. Consequently, $m \geq \Gamma(c)$ so that $c \leq \Gamma^{-1}(m)$, which proves the theorem.

It remains to prove the recurrence. We first prove $|L_{c-1}| \geq 1$. For the purpose of a contradiction, assume that the list L_{c-1} is empty. Then the load of machine 1 is less than $(c - 1) \cdot \text{opt}(G)$ in the equilibrium assignment A. Let i be a task placed on a machine j with load at least $c \cdot \text{opt}(G)$. Moving i to machine 1 reduces the cost of i to strictly less than

$$(c - 1) \cdot \text{opt}(G) + \frac{w_i}{s_1} \leq (c - 1) \cdot \text{opt}(G) + \text{opt}(G) \leq c \cdot \text{opt}(G),$$

where the inequality $\frac{w_i}{s_1} \leq \text{opt}(G)$ follows from the fact that s_1 is the speed of the fastest machine. Consequently, agent i is able to unilaterally decrease its cost by moving its task from machine j to machine 1, which contradicts the assumption that A is a Nash equilibrium. Thus, we have shown that $|L_{c-1}| \geq 1$.

Next, we show $|L_k| \geq (k + 1) \cdot |L_{k+1}|$, for $0 \leq k \leq c - 2$. Let A^* be an optimal assignment, i.e., an assignment whose makespan is equal to $\text{opt}(G)$. The following lemma relates the placement of tasks in the equilibrium assignment A to the placement of tasks in the optimal assignment A^*.

Lemma 20.8 *Suppose i is a task with $A(i) \in L_{k+1}$. Then $A^*(i) \in L_k$.*

PROOF If $L \setminus L_k = \emptyset$ then this claim follows trivially. Let q be the smallest index in $L \setminus L_k$, i.e., machine q is one of the machines with maximum speed among the machines $L \setminus L_k$. By the definition of the group L_k, the load of q is less than $k \cdot \text{opt}(G)$, i.e., $\ell_q < k \cdot \text{opt}(G)$. Figure 20.2(b) illustrates the situation.

By the definition of the groups, $A(i) \in L_{k+1}$ implies $\ell_{A(i)} \geq (k + 1) \cdot \text{opt}(G)$. For the purpose of a contraction, assume $w_i \leq s_q \cdot \text{opt}(G)$. Then moving task i to machine q would reduce the cost of i to

$$\ell_q + \frac{w_i}{s_q} < k \cdot \text{opt}(G) + \text{opt}(G) \leq \ell_{A(i)},$$

which contradicts the assumption that A is a Nash equilibrium. Hence, every task i with $A(i) \in L_{k+1}$ satisfies $w_i > s_q \cdot \text{opt}(G)$. Now, for the purpose of a contradiction, suppose $A^*(i) = j$ and $j \in L \setminus L_k$. Then the load on j under A^* would be at least

$$\frac{w_i}{s_j} > \frac{s_q \cdot \text{opt}(G)}{s_j} \geq \text{opt}(G)$$

because $s_j \leq s_q$. However, this contradicts that A^* is an optimal assignment. Consequently, $A^*(i) \in L_k$. $\square$

By the definition of L_{k+1}, the sum of the weights that A assigns to a machine $j \in L_{k+1}$ is at least $(k + 1) \cdot \text{opt}(G) \cdot s_j$. Hence, the total weight assigned to the machines in L_{k+1} is at least $\sum_{j \in L_{k+1}} (k + 1) \cdot \text{opt}(G) \cdot s_j$. By Lemma 20.8 an optimal assignment has to assign all this weight to the machines in L_k such that

the load on each of these machines is at most $opt(G)$. As a consequence,

$$\sum_{j \in L_{k+1}} (k+1) \cdot opt(G) \cdot s_j \leq \sum_{j \in L_k} opt(G) \cdot s_j.$$

Dividing by $opt(G)$ and subtracting $\sum_{j \in L_{k+1}} s_j$ from both sides yields

$$\sum_{j \in L_{k+1}} k \cdot s_j \leq \sum_{j \in L_k \setminus L_{k+1}} s_j.$$

Now let s^* denote the speed of the slowest machine in L_{k+1}, i.e., $s^* = s_{|L_{k+1}|}$. For all $j \in L_{k+1}$, $s_j \geq s^*$, and, for all $j \in L_k \setminus L_{k+1}$, $s_j \leq s^*$. Hence, we obtain

$$\sum_{j \in L_{k+1}} k \cdot s^* \leq \sum_{j \in L_k \setminus L_{k+1}} s^*,$$

which implies $|L_{k+1}| \cdot k \leq |L_k \setminus L_{k+1}| = |L_k| - |L_{k+1}|$. Thus, $|L_k| \geq (k+1) \cdot |L_{k+1}|$. This completes the proof of Theorem 20.7. $\square$

We now prove a lower bound showing that the upper bound on the price of anarchy given in Theorem 20.7 is essentially tight.

Theorem 20.9   *For every $m \in \mathbb{N}$, there exists an instance G of the load balancing game with m machines and $n \leq m$ tasks that has a Nash equilibrium assignment $A : [n] \rightarrow [m]$ with*

$$cost(A) = \Omega\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

PROOF   Recall the definition of the gamma function from the proof of Theorem 20.7. We describe a game instance G together with an equilibrium assignment A satisfying

$$cost(A) \geq \frac{1}{2} \cdot \left(\Gamma^{-1}(m) - 2 - o(1)\right) \cdot opt(G),$$

which yields the theorem.

Our construction uses $q + 1$ disjoint groups of machines denoted $G_0, \ldots, G_q$ with $q \approx \Gamma^{-1}(m)$. More, precisely, we set

$$q = \lfloor \Gamma^{-1}(m/3) - 1 \rfloor \geq \Gamma^{-1}(m) - 2 - o(1).$$

For $0 \leq k \leq q$, group G_k consists of $q!/k!$ machines of speed 2^k each of which is assigned k tasks of weight 2^k. Let us remark that $0! = 1$. The total number of machines in these groups is thus

$$\sum_{k=0}^{q} |G_k| = q! \sum_{k=0}^{q} \frac{1}{k!} \leq 3\,\Gamma(q+1) \leq m$$

because $\sum_{k=0}^{q} \frac{1}{k!} \leq 3$ and $3\Gamma(q+1) \leq m$, which follows directly from the definition of q. As m might be larger than the number of the machines in the groups, there might be some machines that do not belong to any of the groups. We assume

that these machines have the same parameters as the machines in group G_0; i.e., they have speed $2^0 = 1$ and A does not assign a tasks to them.

We need to show that the described assignment is a Nash equilibrium. An agent with a task on a machine from group G_k has cost k. It can neither reduce its cost by moving its task to a machine in group G_j with $j \geq k$ as these machines have at least a load of k, nor can it reduce its cost by moving its task to a machine in group G_j with $j < k$ as the load on such a machine, after the task moved to this machine, would be

$$j + \frac{2^k}{2^j} = j + 2^{k-j} \geq j + (k - j + 1) = k + 1$$

since $2^t \geq t + 1$, for every $t \geq 1$. Hence, none of the agents can unilaterally decrease its cost. In other words, A is a Nash equilibrium.

The social cost of the equilibrium assignment A is q. Next we show that $\mathrm{opt}(G) \leq 2$ so that the theorem follows. We construct an assignment with load at most 2 on every machine. For each $k \in \{1, \ldots, q\}$, the tasks mapped by A to the machines in group G_k are now assigned to the machines in group G_{k-1}. Observe that the total number of tasks that A maps to the machines in G_k is

$$k \cdot |G_k| = k \cdot \frac{q!}{k!} = \frac{q!}{(k-1)!} = |G_{k-1}|.$$

Hence, we can assign the tasks in such a way that each machine in group G_{k-1} receives exactly one of the tasks that A mapped to a machine in group G_k. This task has a weight of 2^k and the speed of the machine is 2^{k-1}. Hence, the load of each machine in this assignment is at most 2, which completes the proof.   □

20.3.2 Algorithms for Computing Pure Equilibria

The proof of Proposition 20.3 reveals that, starting from any initial assignment, a pure Nash equilibrium is reached after a finite number of improvement steps. Theorem 20.6 shows that there exists a sequence of improvement steps of length $O(n)$ in case of identical machines and this sequence can be computed efficiently. However, in the case of uniformly related machines, it is not known whether there always exists a short sequence of improvement steps and whether such a sequence can be efficiently computed like in the case of identical machines. However, the well-known *LPT (largest processing time)* scheduling algorithm allows us to efficiently compute a Nash equilibrium. This algorithm inserts the tasks in a nonincreasing order of weights, assigning each task to a machine that minimizes the cost of the task at its insertion time.

Theorem 20.10   *The LPT algorithm computes a pure Nash equilibrium for load balancing games on uniformly related machines.*

PROOF   Let the tasks be numbered from 1 to n in the order of their insertion. Let *time* $t \in \{0, \ldots, n\}$ denote the point of time after the first t tasks have been inserted. We show by an induction that the partial assignment $A : [t] \rightarrow [m]$ computed by LPT at time t is a Nash equilibrium. By our induction assumption

the tasks $1, \ldots, t-1$ are *satisfied* at time $t-1$; i.e., none of these tasks can improve its cost by a unilateral deviation. When task t is inserted, it might be mapped to a machine $j^* \in [m]$ that holds already some other tasks. We only have to show that these tasks do not get unsatisfied because of the increased load on j^* because of the assignment of task t. Let $i < t$ be one of the tasks mapped to machine j^*. For $j \in [m]$, let ℓ_j denote the load on machine j at time t. Since the assignment of task t to machine j^* minimizes the cost of agent t and as $w_t \leq w_i$,

$$\frac{\ell_{j^*}}{s_{j^*}} \leq \frac{\ell_j + w_t}{s_j} \leq \frac{\ell_j + w_i}{s_j},$$

for all $j \in [m]$. Hence, also at time t, agent i is satisfied on machine j^* as it cannot reduce its cost by moving from j^* to another machine.   □

The assignment computed by the LPT algorithm is not only a Nash equilibrium but it also approximates the optimal makespan within a ratio of at most $\frac{5}{3}$ for uniformly related machines and $\frac{4}{3} - \frac{1}{3m}$ for identical machines, see Friesen (1987) and Graham (1966), respectively. As makespan scheduling is NP-hard even on identical machines, one cannot hope for an efficient algorithm that computes an assignment with optimal makespan, unless $P \neq NP$. However, the polynomial time approximation scheme of Hochbaum and Shmoys (1988) computes an assignment of tasks to uniformly related machines minimizing the makespan within a ratio of $(1 + \epsilon)$, for any given $\epsilon > 0$. This assignment is not necessarily a Nash equilibrium. Feldmann et al. (2003a) present an efficient algorithm that transforms any given assignment into an equilibrium assignment without increasing the makespan. This approach is called *Nashification*. Combining the polynomial time approximation scheme with the Nashification approach yields a polynomial time algorithm that computes an equilibrium assignment for scheduling on uniformly related machines minimizing the makespan within a factor of $(1 + \epsilon)$, for any given $\epsilon > 0$.

20.4 Mixed Equilibria on Identical Machines

The example with two identical machines presented in Section 20.1.2 shows that the social cost can increase if players make use of randomization. Let us now study this effect systematically. We analyze by how much the price of anarchy is increased when the set of strategies is extended from pure to mixed strategies. First, we consider an extreme case of randomization in which every agent randomizes over all strategies.

20.4.1 Fully Mixed Equilibria

The *support* of an agent is the set of strategies to which the agent assigns positive probability. In a *fully mixed strategy profile* all pure strategies are in the support of every agent. There is exactly one fully mixed strategy profile for load balancing games on identical machines i.e. a Nash equilibrium. In this *fully mixed Nash equilibrium* every player assigns every task with probability $\frac{1}{m}$ to each of the machines, i.e., $P = (p_i^j)$ with $p_i^j = \frac{1}{m}$, for every $i \in [n]$ and $j \in [m]$. The fully mixed Nash equilibrium maximizes

the randomization and, hence, seems to be a good candidate to study the effects of randomization.

Our analysis begins with a particularly simple class of load balancing games: Suppose that we have not only identical machines but also identical tasks. That is, we assume that there are m machines of speed 1 and n tasks of weight 1. In the unique fully mixed Nash equilibrium for such a game, each task is assigned to each machine with probability $\frac{1}{m}$. This strategy profile is a Nash equilibrium as the expected cost c_i^j of any task i on any machine j is the same. In particular, Equation 20.1 yields

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - \frac{1}{m}\right) = 2 - \frac{1}{m}.$$

This setup corresponds to a well-studied balls-and-bins experiment from probability theory in which n balls are assigned independently, uniformly at random to m bins, which is also discussed in Chapter 17. How bad is such a fully mixed Nash equilibrium in comparison to an optimal assignment that distributes the tasks evenly among the machines? An optimal assignment minimizes the makespan, and the optimal makespan is obviously $\lceil \frac{n}{m} \rceil$. The expected makespan of the fully mixed strategy profile corresponds to the expected *maximum occupancy* of the corresponding balls-and-bins experiment, i.e., the expected number of balls in the fullest bin. The following proposition yields a simple formula for this quantity that is exact up to constant factors for any choice of m and n.

Proposition 20.11   *Suppose that $n \geq 1$ balls are placed independently, uniformly at random into $m \geq 1$ bins. Then the expected maximum occupancy is*

$$\Theta\left(\frac{\ln m}{\ln\left(1 + \frac{m}{n}\ln m\right)}\right).$$

Let us illustrate the formula for the expected maximum occupancy given in the proposition with a few examples. If $n \geq m \log m$, then the expected maximum occupancy is $\Theta(\frac{n}{m})$ as, in this case, $\ln\left(1 + \frac{m}{n}\ln m\right) = \Theta\left(\frac{m}{n}\ln m\right)$. If $n \leq m^{1-\epsilon}$, for any fixed $\epsilon > 0$, then the expected maximum occupancy is $\Theta(1)$. Observe, in both of these cases, the ratio between the expected makespan for the fully mixed equilibrium and the makespan of an optimal assignment is $O(1)$. It turns out that this ratio is maximized when setting $m = n$. In this case, the expected maximum occupancy is $\Theta(\log m / \log \log m)$ while the optimal makespan is 1. This yields the following result.

Theorem 20.12   *For every $m \in \mathbb{N}$, there exists an instance G of a load balancing game with m identical machines and $n = m$ tasks that has a Nash equilibrium strategy profile P with*

$$cost(P) = \Omega\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

As the fully mixed Nash equilibrium is the equilibrium that maximizes the randomization, one could guess that this is also the equilibrium that maximizes the ratio between the expected makespan and the optimal makespan for load balancing games. This guess is known as the so-called *fully mixed Nash equilibrium conjecture*. This conjecture is appealing as it would yield a simple characterization of the worst-case Nash equilibrium for load balancing games. Unfortunately, however, the conjecture is wrong. With the help of Proposition 20.11, we can easily construct a counterexample. Let $m = 2^{2k}$, for some $k \in \mathbb{N}$. This way, $\sqrt{m}$ as well as $\log m$ are integers. Now consider the following instance of the load balancing game on m identical machines. Suppose that there are $\sqrt{m}$ *large tasks* of weight 1, and $(m - \sqrt{m}) \cdot \log m$ *small tasks* of weight $\frac{1}{\log m}$. The balls-and-bins analysis above shows that the maximum number of large tasks that are assigned to the same machine by a fully mixed Nash equilibrium is $O(1)$, and the maximum number of small tasks assigned to the same machine is $O(\log m)$. Hence, the expected makespan of the fully mixed Nash equilibrium is $O(1)$. Now consider the following strategy profile: Assign the large tasks uniformly at random to the first $\sqrt{m}$ machines (called group A) and the small tasks uniformly at random to the other machines (called group B). This profile is a Nash equilibrium as Equation 20.1 yields that, for a large task, the expected cost on a machine of group A is less than the expected cost on a machine of group B and, for a small task, the expected cost on a machine of group B is less than the expected cost on a machine of group A. In this equilibrium, the expected maximum occupancy among the large tasks is $\Theta(\frac{\log m}{\log \log m})$, which shows that there is a mixed Nash equilibrium whose expected makespan is larger than the expected makespan of the fully mixed Nash equilibrium by a factor of $\Omega(\frac{\log m}{\log \log m})$.

20.4.2 Price of Anarchy

The fully mixed Nash equilibrium is not necessarily the worst-case Nash equilibrium for every instance of the load balancing game on identical machines. Nevertheless, the following analysis shows that the lower bound on the price of anarchy that we obtained from studying this kind of equilibria is tight.

Theorem 20.13   *Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m identical machines. Let $P = (p_i^j)_{i \in [n], j \in [m]}$ denote any Nash equilibrium strategy profile. Then, it holds that*

$$cost(P) = \mathcal{O}\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

PROOF   Without loss of generality, we assume that all machines have speed 1. Recall that $cost(P) = \mathbb{E}[\max_{j \in [m]}(\ell_j)]$, i.e., $cost(P)$ corresponds to the expected maximum load over all machines or, in other words, the expected makespan. Our analysis starts with proving an upper bound on the maximum expected load instead of the expected maximum load.

We claim that, for every $j \in [m]$, $\mathbb{E}[\ell_j] \leq (2 - \frac{2}{m+1}) opt(G)$. The proof for this claim follows the course of the analysis for the upper bound on the price of anarchy for pure equilibria. More specifically, the proof of Theorem 20.5 can be

adapted as follows to mixed equilibria: Instead of considering a smallest weight task i^* placed on a maximum load machine j^*, one defines i^* to be the smallest weight task with positive probability on a machine j^* maximizing the expected load. Also in all other occurrences one considers the expected load instead of the load.

We conclude that the maximum expected load is less than $2 \operatorname{opt}(G)$. Next we show that the expected maximum load deviates at most by a factor of $\mathcal{O}(\frac{\log m}{\log \log m})$ from the maximum expected load. We use a weighted Chernoff bound in order to show that it is unlikely that there is a machine that deviates by a large factor from its expectation.

Lemma 20.14 (weighted Chernoff bound)   *Let $X_1, \ldots, X_N$ be independent random variables with values in the interval $[0, z]$ for some $z > 0$, and let $X = \sum_{i=1}^{N} X_i$, then for any t it holds that $\mathbb{P}[\sum_{i=1}^{N} X_i \geq t] \leq (e \cdot \mathbb{E}[X] / t)^{t/z}$.*

A description how to derive this and other variants of the Chernoff bound can be found, e.g., in Mitzenmacher and Upfal (2005).

Fix $j \in [m]$. Let w denote the largest weight of any task. Applying the weighted Chernoff bound shows that, for every t,

$$\mathbb{P}[\ell_j \geq t] \leq \min \left\{ 1, \left(\frac{e \cdot \mathbb{E}[\ell_j]}{t} \right)^{t/w} \right\} \leq \left(\frac{2 e \operatorname{opt}(G)}{t} \right)^{t/\operatorname{opt}(G)}.$$

because $\mathbb{E}[\ell_j] \leq 2 \operatorname{opt}(G)$ and $w \leq \operatorname{opt}(G)$. Now let $\tau = 2 \operatorname{opt}(G) \frac{\ln m}{\ln \ln m}$. Then, for any $x \geq 0$,

$$\begin{aligned}
\mathbb{P}[\ell_j \geq \tau + x] &\leq \left(\frac{e \ln \ln m}{\ln m} \right)^{2 \ln m / \ln \ln m + x / \operatorname{opt}(G)} \\
&\leq \left(\frac{1}{\sqrt{\ln m}} \right)^{2 \ln m / \ln \ln m} \cdot e^{-x/\operatorname{opt}(G)} \\
&= m^{-1} \cdot e^{-x/\operatorname{opt}(G)},
\end{aligned}$$

where the second inequality holds asymptotically as, for sufficiently large m, $\frac{\ln m}{e \ln \ln m} \geq \sqrt{\log m}$ and $\frac{\ln m}{e \ln \ln m} \geq e$.

Now with the help of the tail bound we can upper-bound $\operatorname{cost}(P)$ as follows. For every nonnegative random variable X, $\mathbb{E}[X] = \int_0^\infty \mathbb{P}[X \geq t] dt$. Consequently,

$$\operatorname{cost}(P) = \mathbb{E}\left[\max_{j \in [m]} \ell_j \right] = \int_0^\infty \mathbb{P}\left[\max_{j \in [m]} \ell_j \geq t \right] dt.$$

Substituting t by $\tau + x$ and then applying the union bound yields

$$\operatorname{cost}(P) \leq \tau + \int_0^\infty \mathbb{P}\left[\max_{j \in [m]} \ell_j \geq \tau + x \right] dx \leq \tau + \int_0^\infty \sum_{j \in [m]} \mathbb{P}[\ell_j \geq \tau + x] dx.$$

Finally, we apply the tail bound derived above and obtain

$$\text{cost}(P) \leq \tau + \int_0^\infty e^{-x/\text{opt}(G)}dx = \tau + \text{opt}(G),$$

which yields the theorem as $\tau = 2\,\text{opt}(G)\,\frac{\ln m}{\ln\ln m}$. $\square$

20.5 Mixed Equilibria on Uniformly Related Machines

Finally, we come to the most general case, namely mixed equilibria on uniformly related machines. The following theorem shows that the price of anarchy for this case is only slightly larger than the one for mixed equilibria on identical machines or pure equilibria on uniformly related machines. The analysis combines the methods from both of these more restricted cases: First, we show that the maximum expected makespan is bounded by

$$\mathcal{O}\left(\frac{\log m}{\log\log m}\right) \cdot \text{opt}(G)$$

using the same kind of arguments as in the analysis of the price of anarchy for pure equilibria on uniformly related machines. Then, as in the case of mixed equilibria on identical machines, we use a Chernoff bound to show that the expected maximum load is not much larger than the maximum expected load. In fact, this last step loses only a factor of order $\log\log m / \log\log\log m$, which results in an upper bound on the price of anarchy of

$$\mathcal{O}\left(\frac{\log m}{\log\log\log m}\right).$$

After proving this upper bound, we present a corresponding lower bound by adding some randomization to the lower bound construction for pure equilibria on uniformly related machines, which increases also the lower bound by a factor of order $\log\log m / \log\log\log m$ and, hence, yields a tight result about the price of anarchy.

Theorem 20.15   *Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m machines of speed $s_1, \ldots, s_m$. Let P be any Nash equilibrium strategy profile. Then, it holds that*

$$cost(P) = \mathcal{O}\left(\frac{\log m}{\log\log\log m}\right) \cdot opt(G).$$

PROOF   As in the case of identical machines, our analysis starts with proving an upper bound on the maximum expected load instead of the expected maximum load. To simplify the notation, we assume $\text{opt}(G) = 1$, which can be achieved by scaling the weights appropriately. Let $c = \lfloor \max_{j \in [m]} \left(\mathbb{E}[\ell_j] \right) \rfloor$. We first prove an upper bound on c following the analysis for pure Nash equilibria in Theorem 20.7. Without loss of generality, assume $s_1 \geq s_2 \geq \cdots \geq s_m$. Let $L = [1, 2, \ldots, m]$ denote the list of machines in non increasing order of speed. For $k \in \{0, \ldots, c - 1\}$, let L_k denote the maximum length prefix of L such that the expected load

of each server in L_k is at least k. Analogously to the analysis in the proof of Theorem 20.7, one shows the recurrence $|L_k| \geq (k + 1) \cdot |L_{k+1}|$, for $0 \leq k \leq c - 2$, and $|L_{c-1}| \geq 1$. Solving the recurrence yields $|L_0| \geq (c - 1)! = \Gamma(c)$. Thus, $|L_0| = m$ implies $c \leq \Gamma^{-1}(m) = \Theta(\ln m / \ln \ln m)$. Now let

$$C = \max \left\{ c + 1, \frac{\ln m}{\ln \ln m} \right\} = \Theta \left(\frac{\ln m}{\ln \ln m} \right).$$

In the rest of the proof, we show that the expected makespan of the equilibrium assignment can exceed C at most by a factor of order $\ln \ln m / \ln \ln \ln m$ so that the expected makespan is $\mathcal{O}(\ln m / \ln \ln \ln m)$, which proves the theorem as we assume $\text{opt}(G) = 1$.

As the next step, we prove a tail bound on ℓ_j, for any fixed $j \in [m]$ and, afterward, we use this tail bound to derive an upper bound on the expected makespan. For a machine $j \in [m]$, let $T_j^{(1)}$ denote the set of tasks i with $p_i^j \geq \frac{1}{4}$ and $T_j^{(2)}$ the set of tasks i with $p_i^j \in (0, \frac{1}{4})$. Let $\ell_j^{(1)}$ and $\ell_j^{(2)}$ denote random variables that describe the load on link j only taking into account the tasks in $T_j^{(1)}$ and $T_j^{(2)}$, respectively. Observe that $\ell_j = \ell_j^{(1)} + \ell_j^{(2)}$. For the tasks in $T_j^{(1)}$, we immediately obtain

$$\ell_j^{(1)} \leq \sum_{i \in T_j^{(1)}} \frac{w_i}{s_j} \leq 4 \sum_{i \in T_j^{(1)}} \frac{w_i \, p_i^j}{s_j} = 4 \, \mathbb{E}[\ell_j^{(1)}] \leq 4C. \qquad (20.2)$$

To prove an upper bound on $\ell_j^{(2)}$, we use the weighted Chernoff bound from Lemma 20.14. This bound requires an upper bound on the maximum weight. As a first step to bound the weights, we prove a result about the relationship between the speeds of the machines in the different groups that are defined by the prefixes. For $0 \leq k \leq c - 2$, let $G_k = L_k \setminus L_{k+1}$, and let $G_{c-1} = L_{c-1}$. For $0 \leq k \leq c - 1$, let $s(k)$ denote the speed of the fastest machine in G_k. Clearly, $s(c - 1) \geq s(c - 2) \geq \cdots \geq s(1) \geq s(0)$. We claim that this sequence is, in fact, geometrically decreasing.

Lemma 20.16 *For $0 \leq k \leq c - 4, s(k + 2) \geq 2 s(k)$.*

PROOF To prove the claim, we first observe that there exists a task j^* with $w_{j^*} \leq s(k + 2)$ that has positive probability on a machine in L_{k+3}. This is because an optimal assignment strategy has to move some of the expected load from the machines in L_{k+3} to machines in $L \setminus L_{k+3}$ and it can only assign those tasks to machines in $L \setminus L_{k+3}$ whose weights are not larger than the maximum speed among this set of machines, which is $s(k + 2)$. Now suppose $s(k) > \frac{1}{2}s(k + 2)$. The expected load of the fastest machine in $G_k = L_k \setminus L_{k+1}$ is at most $k + 1$. Thus the expected cost of j^* on the fastest machine in G_k is at most

$$k + 1 + \frac{w_{j^*}}{s(k)} < k + 1 + \frac{2w_{j^*}}{s(k + 2)} \leq k + 3.$$

This contradicts that the expected cost of j^* in the considered Nash equilibrium is at least $k + 3$ as it has positive probability on a machine in L_{k+3}. Thus, Lemma 20.16 is shown. $\square$

Now we apply Lemma 20.16 to prove an upper bound on the weights of the tasks in the set $T_j^{(2)}$.

Lemma 20.17   *For every $j \in [m]$ and $i \in T_j^{(2)}$, $w_i \leq 12\, s_j$.*

PROOF   Let i be a task from $T_j^{(2)}$, i.e., $p_i^j \in (0, \frac{1}{4})$. Let $j \in G_k$, for $0 \leq k \leq c - 1$. The expected cost of i on j is

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - p_i^j\right) \frac{w_i}{s_j} \geq k + \frac{3w_i}{4s_j}.$$

Suppose that $k \geq c - 3$. In this case, $w_i > 12\, s_j$ implies $c_i^j > k + \frac{3}{4} \cdot 12 \geq c + 6$, which contradicts that, under the Nash equilibrium profile, the expected cost of any task on the fastest machine is at most $c + 1$. Hence, the lemma is shown for $k \geq c - 3$. Now suppose $k \leq c - 4$. Let q denote the fastest machine from G_{k+2}. Lemma 20.16 yields $s_q = s(k + 2) \geq 2s(k) \geq 2\, s_j$. Hence, the expected cost of i on q is

$$c_i^q = \mathbb{E}[\ell_q] + \left(1 - p_i^q\right) \frac{w_i}{s_q} \leq k + 3 + \frac{w_i}{2s_j}.$$

As $p_i^j > 0$, the Nash equilibrium condition yields $c_i^j \leq c_i^q$. Consequently,

$$k + \frac{3w_i}{4s_j} \leq k + 3 + \frac{w_i}{2s_j},$$

which implies $w_i \leq 12\, s_j$ and, hence, completes the proof of Lemma 20.17. $\square$

Let $z = \max_{i \in T_j^{(2)}}(w_i/s_j)$. Lemma 20.17 implies $z \leq 12$. Now applying the weighted Chernoff bound from Lemma 20.14 yields that, for every $\alpha > 0$,

$$\mathbb{P}[\ell_j^{(2)} \geq \alpha C] \leq \left(\frac{e \cdot \mathbb{E}[\ell_j^{(2)}]}{\alpha C}\right)^{\alpha C / z} \leq \left(\frac{e}{\alpha}\right)^{\alpha C / 12}$$

since $\mathbb{E}[\ell_j^{(2)}] \leq C$. We define $\tau = 24\, C \ln \ln m / \ln \ln \ln m$. As C is of order $\ln m / \ln \ln m$, it follows that τ is of order $\ln m / \ln \ln \ln m$. Let $x \geq 0$. We substitute $\tau + x$ for αC and obtain

$$\mathbb{P}[\ell_j^{(2)} \geq \tau + x] \leq \left(\frac{eC}{\tau + x}\right)^{(\tau + x)/12}$$

$$\leq \left(\frac{e \ln \ln \ln m}{24 \ln \ln m}\right)^{2C \ln \ln m / \ln \ln \ln m + x/12}.$$

Observe that $24 \ln \ln m / (e \ln \ln \ln m)$ is lower-bounded by $\sqrt{\ln \ln m}$ and also lower-bounded by e^2. Furthermore, $C \geq \ln m / \ln \ln m$. Applying these bounds yields

$$\mathbb{P}[\ell_j^{(2)} \geq \tau + x] \leq \left(\frac{1}{\sqrt{\ln \ln m}}\right)^{2 \ln m / \ln \ln \ln m} \cdot e^{-x/6} = m^{-1} \cdot e^{-x/6}.$$

As a consequence,

$$\mathbb{E}\left[\max_{j \in [m]} \ell_j^{(2)}\right] = \int_0^\infty \mathbb{P}\left[\max_{j \in [m]} \ell_j^{(2)} \geq t\right] dt$$

$$\leq \tau + \int_0^\infty \mathbb{P}\left[\max_{j \in [m]} \ell_j^{(2)} \geq \tau + x\right] dx$$

$$\leq \tau + \int_0^\infty \sum_{j \in [m]} \mathbb{P}[\ell_j^{(2)} \geq \tau + x] dx.$$

Now applying our tail bound yields

$$\mathbb{E}\left[\max_{j \in [m]} \ell_j^{(2)}\right] \leq \tau + \int_0^\infty e^{-x/6} dx = \tau + 6. \tag{20.3}$$

Finally, we combine Equations 20.2 and 20.3 and obtain

$$\text{cost}(P) = \mathbb{E}\left[\max_{j \in [m]} \ell_j\right] \leq 4C + \tau + 6 = \mathcal{O}\left(\frac{\log m}{\log \log \log m}\right),$$

which completes the proof of Theorem 20.15. $\square$

Next we show that the upper bound given in Theorem 20.15 is tight by showing that for every number of machines there exists a game instance that matches the upper bound up to a constant factor.

Theorem 20.18 *For every $m \in \mathbb{N}$, there exists an instance G of the load balancing game with m machines and $n \leq m$ tasks that has a Nash equilibrium strategy profile P with*

$$\text{cost}(P) = \Omega\left(\frac{\log m}{\log \log \log m}\right) \cdot opt(G).$$

PROOF The starting point for our construction is the game and the Nash assignment A from the proof of Theorem 20.9. We use mixed strategies in only one of the groups, namely in the group G_k with $k = \lceil q/2 \rceil$. Let M denote the number of machines in this group, i.e., $M = q!/k! \geq (q/2)^{\lfloor q/2 \rfloor}$. Observe that $\log M = \Theta(q \log q) = \Theta(\log m)$.

Let T denote the set of tasks mapped by A to one of the machines in G_k. The tasks in T have weight 2^k. Each of these tasks is now assigned uniformly at random to a machine group G_k, i.e., $p_i^j = \frac{1}{M}$, for each $j \in G_k$ and each $i \in T$. For all other tasks the strategy profile P corresponds without any change to the pure strategy profile of assignment A. Observe that the randomization increases the expected cost of the tasks. The expected cost of a task $i \in T$ on a machine

$j \in G_k$ is now

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - p_i^j\right) \frac{w_i}{s_j} = k + \left(1 - \frac{1}{M}\right) < k + 1.$$

In the proof of Theorem 20.9, we have shown that the cost of a task i of weight 2^k on a machine of group G_j with $j \neq k$ is at least $k + 1$. Thus, the strategy profile P is a Nash equilibrium.

It remains to compare the social cost of the equilibrium profile P with the optimal cost. The structure of the optimal assignment is not affected by the modifications. It has social cost $\text{opt}(G) = 2$. Now we give a lower bound for the social cost of P. This social cost is, obviously, bounded from below by the maximum number of tasks that are mapped to the same machine in the group G_k. Applying Proposition 20.11 with M bins and $N = kM$ balls shows that the expected makespan is

$$\Omega\left(\frac{\ln M}{\ln\left(1 + \frac{1}{k}\ln M\right)}\right) = \Omega\left(\frac{\log m}{\log\log\log m}\right),$$

where the last estimate holds as $k = \Theta\left(\log m / \log\log m\right)$ and $\log M = \Theta(\log m)$. This completes the proof of Theorem 20.18. $\quad\square$

20.6 Summary and Discussion

In this chapter, we studied the price of anarchy in load balancing games in four different variants. Table 20.1 summarizes the results about the price of anarchy that we have presented. In the case of pure equilibria on identical machines, the price of anarchy is bounded from above by a small constant term. In all other cases, the price of anarchy is bounded from above by a slowly growing, sublogarithmic function in the number of machines. One might interpret these results as a first game theoretic explanation why the resources in a large distributed system like the Internet that widely lacks global control are shared in a more or less efficient and fair way among different users with different interests, although the considered model is clearly oversimplifying in several aspects.

It is an interesting coincidence that both the price of anarchy for pure equilibria on uniformly related machines as well as the price of anarchy for mixed equilibria

Table 20.1. *The price of anarchy for pure and mixed equilibria in load balancing games on identical and uniformly related machines*

	Identical	Uniformly related
Pure	$2 - \frac{2}{m+1}$	$\Theta\left(\frac{\log m}{\log\log m}\right)$
Mixed	$\Theta\left(\frac{\log m}{\log\log m}\right)$	$\Theta\left(\frac{\log m}{\log\log\log m}\right)$

on identical machines are of order $\log m / \log \log m$. Although both of these models result in essentially the same price of anarchy, the reasons for the increase in the social cost are quite different: In the case of pure equilibria on uniformly related machines, equilibrium assignments correspond to local optima with respect to moves of single tasks. That is, tasks are placed in a suboptimal but nevertheless coordinated fashion. On the contrary, in case of mixed equilibria, the increase in cost is due to collisions between uncoordinated random decisions. If one combines these two effects, then one loses only another very small factor of order $\log \log m / \log \log \log m$, which results in a price of anarchy of order $\log m / \log \log \log m$ for mixed equilibria on uniformly related machines.

Obviously, the price of anarchy for load balancing games as we have defined them in the beginning of this chapter is well understood. As mentioned above, however, this model is very simplistic. To make these results more realistic, one needs to incorporate other aspects from practical application areas like, e.g., more realistic cost functions or other ways to define the social cost. We give pointers to studies of quite a few variants of load balancing games in the bibliographic notes. In Christodoulou et al. (2004), it is made an interesting attempt that adds an algorithmic or constructive element to the analysis of the price of anarchy. The idea behind so-called "coordination mechanisms" is not to study the price of anarchy for a fixed system, but to design the system in such a way that the increase in cost or the loss in performance due to selfish behavior is as small as possible. Similar aspects are also discussed in Chapter 17. We believe that this is a promising direction of research that might result in practical guidelines of how to build a distributed system that does not suffer from selfish behavior but might even exploit the selfishness of the agents.

Besides the price of anarchy, we have studied the question of how agents reach a Nash equilibrium. We have observed that any sequence of improvement steps reaches a pure Nash equilibrium after a finite number of steps. In case of identical machines the max-weight best-response policy reaches an equilibrium in only $O(n)$. In case of uniformly related machines, it is open whether there exists a short sequence of improvement steps that lead from any given assignment to a pure Nash equilibrium. We think that this question is of great importance as Nash equilibria are only of interest if agents can reach them quickly. It is not clear that the only reasonable approach for the agents to reach a Nash equilibrium in a distributed way is to use improvement steps. There might also be other, possibly more strategic or more coordinated behavioral rules that quickly converge to a Nash equilibrium or to an approximate Nash equilibrium. For example, Chapter 29 considers some approaches from evolutionary game theory in the context of routing in networks. It is an interesting research problem to design distributed protocols that ensure that agents reach a Nash equilibrium quickly. Pointers to first results toward this direction can be found in the bibliographic notes.

20.7 Bibliographic Notes

The concept of the price of anarchy was introduced by Koutsoupias and Papadimitriou (1999). In their seminal work, they study load balancing in form of a routing game

consisting of two nodes connected by parallel edges with possibly different speeds. Each agent has an amount of traffic that the agent seeks to map to one of the edges such that the load on this edge is as small as possible. In our notation, the parallel edges between the source and the sink correspond to the machines and the pieces of traffic of the agents correspond to the tasks. Let us remark that originally the ratio between the social cost in a worst-case Nash equilibrium and the optimal social cost was called *coordination ratio* but in this chapter we switched to the now commonly used term *price of anarchy*. The game theoretic model underlying the load balancing games is also known as *KP model*.

The results presented in Table 20.1 have been obtained in the following studies. The upper bound of $2 - \frac{2}{m+1}$ on the price of anarchy for pure equilibria in load balancing games with identical machines goes back to the scheduling literature (Finn and Horowitz, 1979), where the same ratio occurs in form of an approximation factor for a local search optimization heuristic. The lower bound on the price of anarchy for mixed equilibria on identical machines is presented in Koutsoupias and Papadimitriou (1999). The analysis for the corresponding upper bound is obtained in Czumaj and Vöcking (2002) and Koutsoupias et al. (2003). Let us remark that the analysis in Czumaj and Vöcking (2002) is tight up to a constant additive term. It shows that the price of anarchy for mixed equilibria in load balancing games on identical machines is $\Gamma^{-1}(m) \pm \Theta(1)$. The upper and lower bounds on the price of anarchy for pure and mixed equilibria in load balancing games with uniformly related machines are from Czumaj and Vöcking (2002) as well. This work also contains a tight characterization of the price of anarchy as a function of the ratio between the speeds of the fastest and the slowest machine.

The existence proof for pure equilibria presented in Section 20.1.1 can be found in Fotakis et al. (2002) and Even-Dar et al. (2003). The result from Section 20.3.2 that the LPT algorithm computes a pure Nash equilibrium is presented in Fotakis et al. (2002) together with several further results about the complexity of computing pure and mixed equilibria in load balancing games. The uniqueness of the fully mixed Nash equilibrium is shown in Mavronicolas and Spirakis (2001). Exercise 20.5 reworks the nice proof for this result. The counterexample to the *fully mixed Nash equilibrium conjecture* presented in Section 20.4.1 is from Fischer and Vöcking (2005). Finally, the results from Section 20.2.2 about the convergence of best response sequences are from Even-Dar et al. (2003).

Let us remark that this chapter does by far not give a complete overview of the rich literature about different variants of games for load balancing or routing on parallel links. We conclude this chapter with a few pointers to further literature. Load balancing games with more general cost functions are considered, e.g., in Caragiannis et al. (2006), Czumaj et al. (2002), Libman and Orda (1999, 2001). Other definitions of the social cost are considered, e.g., in Caragiannis et al. (2006), Gairing et al. (2004a, 2004b) and Suri et al. (2004). Another interesting variant of load balancing games assumes that agents come with subsets of the machines on which they have to place their tasks. The price of anarchy in such a restricted assignment model is investigated in Awerbuch et al. (2003), Gairing et al. (2006), and Suri et al. (2004). The price of anarchy with respect to equilibria that are robust against coalitions is studied in Andelman et al. (2007). An important aspect that we have only touched in this chapter is the complexity of

computing Nash equilibria for load balancing games. Further work dealing with the computation of Nash equilibria can be found, e.g., in Even-Dar et al. (2003), Feldmann et al. (2003a), Fotakis et al. (2002), Fischer and Vöcking (2005), and Gairing et al. (2004a). Recent work deals also with the convergence time of distributed load balancing processes in which agents make parallel attempts for improvement steps until they find a Nash equilibrium (Berenbrink et al., 2006; Even-Dar and Mansour, 2005). Another interesting topic is load balancing games with incomplete information that have been considered, e.g., in Beier et al. (2004) and Gairing et al. (2005). Finally, let us remark that the concept of coordination mechanisms has been suggested in Christodoulou et al. (2004) and some further results on this topic can be found in Immorlica et al. (2005).

Several other results for load balancing and routing on parallel links have been collected in the surveys (Czumaj, 2004; Feldmann et al., 2003b; Koutsoupias, 2003).

Bibliography

N. Andelman, M. Feldman, and Y. Mansour. Strong price of anarchy. In *Proc. 18th Annual ACM-SIAM Symp. on Discrete Algorithms*, 2007.

B. Awerbuch, Y. Azar, Y. Richter, and D. Tsur. Tradeoffs in worst-case equilibria. In *Proc. 1st International Workshop on Approximation and Online Algorithms (WAOA)*, pp. 41–52, 2003.

R. Beier, A. Czumaj, P. Krysta, and B.Vöcking. Computing equilibria for congestion games with (im)perfect information. In *Proc. 15th Annual ACM-SIAM Symp. Discrete Algorithms*, pp. 746–755, 2004.

P. Berenbrink, T. Friedetzky, L.A. Goldberg, P.W. Goldberg, Z. Hu, and R.A. Martin. Distributed selfish load balancing. In *Proc. 17th Annual ACM-SIAM Symp. Discrete Algorithms*, pp. 354–363, 2006.

I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. In *Proc. 33rd Intl. Colloq. on Automata, Languages, and Programming*, pp. 311–322, 2006.

G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination Mechanisms. In *Proc. 31st Intl. Colloq. on Automata, Languages and Programming*, pp. 345–357, 2004.

A. Czumaj. Selfish Routing on the Internet. Chapter 42 in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, edited by J. Leung, CRC Press, Boca Raton, FL, 2004.

A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proc. 34th Annual ACM Symp. Theory of Computing*, pp. 287–296, 2002.

A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proc. 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 413–420, 2002.

E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibria. In *Proc. 30th International Colloq. on Automata, Languages and Programming*, pp. 502–513, 2003.

E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *Proc. 16th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 772–781, 2005.

R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. 30th International Colloq. on Automata, Languages and Programming*, pp. 414–426, 2003a.

R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Selfish routing in non-cooperative networks: a survey. In *Proc. 28th International Symp. on Mathematical Foundations of Computer Science*, pp. 21–45, 2003b.

G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT,* 19(3):312–320, 1979.

S. Fischer and B. Vöcking. On the structure and complexity of worst-case equilibria. In *Proc. 1st Workshop on Internet and Network Economics*, pp. 151–160, 2005.

D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. 29th Intl. Colloquium on Automata, Languages and Programming (ICALP)*, pp. 123–134, 2002.

D.K. Friesen. Tighter bounds for LPT scheduling on uniform processors. *SIAM J. Computing*, 16(3):554–560, 1987.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. Computing Nash equilibria for scheduling on restricted parallel links. In *Proc. 36th Annual ACM Symp. on Theory of Computing*, pp. 613–622, 2004a.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. The price of anarchy for polynomial social cost. In *Proc. 29th Intl. Symp. on Mathematical Foundations of Computer Science*, pp. 574–585, 2004b.

M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. Nash equilibria in discrete routing games with convex latency functions. In *Proc. 31st Intl. Colloq. on Automata, Languages and Programming*, pp. 645–657, 2004c.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. The Price of Anarchy for Restricted Parallel Links. *Parallel Process. Lett.*, 16(1):117–132, 2006.

M. Gairing, B. Monien, and K. Tiemann. Selfish routing with incomplete information. In *Proc. 17th Annual ACM Symp. on Parallel Algorithms*, pp. 203–212, 2005.

R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45: 1563–1581, 1966.

R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17: 263–269, 1969.

D.S. Hochbaum and D.B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors. *SIAM J. Computing*, 17(3):539–551, 1988.

N. Immorlica, L. Li, V.S. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *Proc. 1st Workshop on Internet and Network Economics*, pp. 55–69, 2005.

E. Koutsoupias. Selfish task allocation. *Bulletin of the EATCS (81)*, pp. 79–88, 2003.

E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion. *Theory of Computing Systems*, 36(6):683–693, 2003.

E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. In *Proc. 16th Annual Symp. on Theoretical Aspects of Computer Science*, pp. 404–413, 1999.

L. Libman and A. Orda. The designer's perspective to atomic noncooperative networks. *IEEE/ACM Trans. Networking*, 7(6):875–884, 1999.

L. Libman and A. Orda. Atomic resource sharing in noncooperative networks. *Telecommun. Systems*, 17(4):385–409, 2001.

M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. 33rd ACM Symp. on Theory of Computing*, pp. 510–519, 2001.

M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

S. Suri, C. Toth, and Y. Zhou. Selfish load balancing and atomic congestion games. In *Proc. 16th Annual ACM Symp. on Parallel Algorithms and Architectures*, pp. 188–195, 2005.

─────────────────── **Exercises** ───────────────────

20.1   Let G be any instance of the load balancing game with three tasks that should be placed on two identical machines. Show that any pure Nash equilibrium for G is optimal, i.e., $\mathrm{cost}(A) = \mathrm{opt}(G)$ for any equilibrium assignment A.
Remark: Interestingly, the example presented in Section 20.1.2 that yields the worst-case price of anarchy for two identical machines uses only four tasks.

20.2   Show, for every $m \in \mathbb{N}$, there exists an instance G of the load balancing game with m identical machines and $2m$ tasks that has a Nash equilibrium assignment $A : [n] \to [m]$ with

$$\mathrm{cost}(A) = \left(2 - \frac{2}{m+1} \right) \cdot \mathrm{opt}(G).$$

Hint: Generalize the example with two machines given in Section 20.1.2.

20.3   Prove that the price of anarchy for pure equilibria on instances of the load balancing game with two tasks and two machines with possibly different speeds corresponds to the golden ratio $\phi = \frac{1}{2}(1 + \sqrt{5})$. That is, show that

a)   there is a game instance G admitting an equilibrium assignment A with $\mathrm{cost}(A) = \phi \cdot \mathrm{opt}(G)$.

b)   for every game instance G and every equilibrium assignment A for this instance, it holds $\mathrm{cost}(A) \le \phi \cdot \mathrm{opt}(G)$.

20.4   Consider an instance of the load balancing game with two tasks both of which have weight 1 and two machines, one of speed 1 and the other of speed $s > 0$.

(a)   Show that there does not exist a fully mixed Nash equilibrium if $s \le \frac{1}{2}$ or $s \ge 2$.

(b)   Show that there exists a unique fully mixed Nash equilibrium if $\frac{1}{2} < s < 2$. Describe the strategy profile of this equilibrium as a function of s.

20.5   Show that there exists at most one fully mixed Nash equilibrium for every instance of the load balancing game.
Hint: Describe the conditions on the probabilities p_i^j imposed by a fully mixed Nash equilibrium in form of a system of linear equations and show that this system has a unique solution. If all the values for the variables p_i^j in this solution are positive then the solution describes a fully mixed Nash equilibrium. Otherwise, there does not exist a fully mixed equilibrium.

20.6   Suppose that we are given an instance G of the load balancing game with m identical machines and n tasks whose weights are bounded from above by $\alpha \cdot \mathrm{opt}(G)$, for $0 < \alpha < 1$.

(a)   Show that $\mathrm{cost}(A) < (1 + \alpha) \cdot \mathrm{opt}(G)$, for every equilibrium assignment A.

(b)   Let $\alpha = \frac{1}{\log m}$. Show that $\mathrm{cost}(A) = \mathcal{O}(\mathrm{opt}(G))$, for every equilibrium strategy profile P.