Tight bounds for selfish and greedy load balancing[*]

Ioannis Caragiannis[†] Michele Flammini[‡] Christos Kaklamanis[§]
Panagiotis Kanellopoulos[¶] Luca Moscardelli[‖]

December 23, 2009

Abstract

We study the load balancing problem in the context of a set of clients each wishing to run a job on a server selected among a subset of permissible servers for the particular client. We consider two different scenarios. In *selfish load balancing*, each client is selfish in the sense that it chooses, among its permissible servers, to run its job on the server having the smallest latency given the assignments of the jobs of other clients to servers. In *online load balancing*, clients appear online and, when a client appears, it has to make an irrevocable decision and assign its job to one of its permissible servers. Here, we assume that the clients aim to optimize some global criterion but in an online fashion. A natural local optimization criterion that can be used by each client when making its decision is to assign its job to that server that gives the minimum increase of the global objective. This gives rise to *greedy* online solutions. The aim of this paper is to determine how much the quality of load balancing is affected by selfishness and greediness.

We characterize almost completely the impact of selfishness and greediness in load balancing by presenting new and improved, tight or almost tight bounds on the price of anarchy of selfish load balancing as well as on the competitiveness of the greedy algorithm for online load balancing when the objective is to minimize the total latency of all clients on servers with linear latency functions. In addition, we prove a tight upper bound on the price of stability of linear congestion games.

[*]A preliminary version of the results in this paper appeared in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06)*, LNCS 4168, Springer, pp. 184-195, 2006. This work was partially supported by the European Union under IST FET Integrated Project 015964 AEOLUS and COST Action 293 GRAAL, and by a "Caratheodory" basic research grant from the University of Patras.

[†]Research Academic Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece. E-mail: `caragian@ceid.upatras.gr`

[‡]Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, Coppito 67100, L'Aquila, Italy. E-mail: `flammini@di.univaq.it`

[§]Research Academic Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece. E-mail: `kakl@ceid.upatras.gr`

[¶]Research Academic Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece. E-mail: `kanellop@ceid.upatras.gr`

[‖]Department of Sciences, University of Chieti-Pescara, Viale Pindaro 42, 65127, Pescara, Italy. Email: `moscardelli@sci.unich.it`

1 Introduction

We study the load balancing problem in the context of a set of clients each wishing to run a job on a server selected among a subset of permissible servers for the particular client. We consider two different scenarios. In the first, called *selfish load balancing* (or *load balancing games*), each client is selfish in the sense that it chooses, among its permissible servers, to run its job on the server having the smallest latency given the assignments of the jobs of other clients to servers. In the second scenario, called *online load balancing*, clients appear online and, when a client appears, it has to make an irrevocable decision and assign its job to one of its permissible servers. Here, we assume that the clients are not selfish and aim to optimize some global objective but in an online fashion (i.e., without any knowledge of clients that may arrive in the future). A natural local optimization criterion that can be used by each client when making its decision is to assign its job to the server that gives the minimum increase of the global objective. This gives rise to greedy online solutions. The aim of this paper is to answer the question of how much the quality of load balancing is affected by selfishness and greediness.

Load balancing games. Load balancing games are special cases of the well-known *congestion games* introduced by Rosenthal [29] and studied in a sequence of papers [6, 10, 11, 14, 16, 18, 26, 30, 31]. In congestion games there is a set E of resources, each resource e having a non-negative and non-decreasing latency function f_e defined over non-negative numbers, and a set of n players. Each player i has a set of strategies $S_i \subseteq 2^E$ (each strategy of player i is a set of resources). An assignment $A = (A_1, ..., A_n)$ is a vector of strategies, one strategy for each player. The cost of a player for an assignment A is defined as $cost(i) = \sum_{e \in A_i} f_e(n_e(A))$, where $n_e(A)$ is the number of players using resource e in A, while the *social cost* of an assignment is the total cost of all players. An assignment is a *pure Nash equilibrium* if no player has any incentive to unilaterally deviate to another strategy, i.e., $cost_i(A) \leq cost_i(A_{-i}, s)$ for any player i and for any $s \in S_i$, where (A_{-i}, s) is the assignment produced if just player i deviates from A_i to s. This inequality is also known as the *Nash condition.* In *weighted congestion games*, each player has a weight w_i and the latency of a resource e depends on the total weight of the players that use e. For this case, a natural social cost function is the weighted sum of the costs of all players (or the weighted average of their costs). In *linear congestion games*, the latency function of resource e is of the form $f_e(x) = \alpha_e x + \beta_e$ with non-negative constants α_e and β_e. Load balancing games are congestion games where the strategies of players are singleton sets. In load balancing terminology, we use the terms server and client instead of the terms resource and player. The set of strategies of a client contains the servers that are permissible for the client. A load balancing game is called *symmetric* when all servers are permissible for any client.

We evaluate the quality of solutions of a load balancing game by comparing the social cost of Nash equilibria to the cost of the optimal assignment (i.e., the minimum cost). We use the notions of *price of anarchy* introduced in a seminal work of Koutsoupias and Papadimitriou [23] (see also [27]) and *price of stability* [3] (or *optimistic price of anarchy*) defined as follows. The price of anarchy/stability of a load balancing game is defined as the ratio of the maximum/minimum social cost over all Nash equilibria over the optimal cost. The price of anarchy/stability for a class of load balancing games is simply the highest price of anarchy/stability among all games belonging to that class.

The papers [15, 17, 18, 19, 22, 25] study various games which can be thought of as special cases of congestion games with respect to the complexity of computing equilibria of best/worst social cost and the price of anarchy when the social cost is defined as the maximum latency experienced by any player. The social cost of the total latency has been studied in [6, 10, 24, 33]. The authors in [24] study symmetric load balancing games with linear latency functions and show tight bounds on the price of anarchy of 4/3 for different servers and 9/8 for identical servers with weighted clients. The price of anarchy of symmetric load balancing games with polynomial or convex latency functions is studied in [20, 21]. In two papers, Awerbuch et al. [6] and Christodoulou and Koutsoupias [10] prove tight bounds on the price of anarchy of congestion games with linear latency functions. Among other results, they show that the price of anarchy of pure Nash equilibria is 5/2 while for mixed Nash equilibria or pure Nash equilibria of weighted clients it is $\frac{3+\sqrt{5}}{2} \approx 2.618$. Tight bounds on the price of anarchy of congestion games with polynomial latency functions are presented in [1]; these improve previous results in [6, 10].

Does the fact that load balancing games are significantly simpler than congestion games in general have any implications for their price of anarchy? We give a negative answer to this question for linear latency functions by showing that the 5/2 upper bound (as well as the $\frac{3+\sqrt{5}}{2}$ upper bound for weighted clients) is tight. This is interesting since the upper bounds for congestion games (as well as an earlier upper bound of 5/2 proved specifically for load balancing [33]) are obtained using only the *Nash inequality* (i.e., the inequality obtained by summing up the Nash condition inequalities over all players' strategies) and the definition of the social cost. So, it is somewhat surprising that load balancing games are as general as congestion games in terms of their price of anarchy and that the Nash inequality provides sufficient information to characterize their price of anarchy.

An important special case of load balancing is when servers have identical linear latency functions. Here, better upper bounds on the price of anarchy can be obtained. Note that this is not the case for congestion games since, as it was observed in [10], any congestion game can be transformed to a congestion game on identical resources (and, hence, the lower bounds of [6, 10] hold for congestion games with identical resources as well). Suri et al. [33] prove that the price of anarchy of selfish load balancing on identical servers is between 2.012067 and $1 + 2/\sqrt{3} \approx 2.1547$. Again, the upper bound is obtained by using the Nash inequality and the definition of the social cost. We improve this result by showing that the lower bound is essentially tight. Besides the Nash inequality, our proof also exploits structural properties of games with high price of anarchy. We argue that such games can be represented as a directed graph (called the *game graph*) and, then, structural properties of such a game follow as structural properties of this graph. Furthermore, for weighted clients and identical servers, we prove that the price of anarchy is at least 5/2.

Tight bounds on the price of stability of load balancing games have been proved in [3]. The price of stability of linear congestion games has been studied in [11] where it was shown that it lies between $1+1/\sqrt{3} \approx 1.577$ and 1.6. The technique used to obtain the upper bound is to consider pure Nash equilibria with potential not larger than the potential of the optimal assignment and bound their social cost in terms of the optimal cost using the Nash inequality. Using the same technique but with a more refined analysis, we show that the lower bound is tight.

Greedy load balancing. From the algorithmic point of view, load balancing has been studied extensively, including papers studying online versions of the problem (e.g., [2, 4, 5, 7, 9, 13, 28, 32, 33]). In online load balancing, clients appear in online fashion; when a client appears, it has to make an irrevocable decision and assign its job to a server. In our model, servers have linear latency functions and the objective is to minimize the total latency, i.e., the sum of the latencies experienced by all clients. Clients may also own jobs with non-negative weights; in this case, the objective is to minimize the weighted sum of the latencies experienced by all clients. A natural greedy algorithm proposed in [5] for this problem is to assign each client to the server that yields the minimum increase to the total latency (ties are broken arbitrarily). This results to *greedy assignments*. Given an instance of online load balancing, an assignment of clients to servers is called a greedy assignment if the assignment of a client to a server minimizes the increase in the cost of the instance revealed up to the time of its appearance. Following the standard performance measure in competitive analysis, we evaluate the performance of this algorithm in terms of its *competitiveness* (or *competitive ratio*). The competitiveness of the greedy algorithm on an instance is the maximum ratio of the cost of any greedy assignment over the optimal cost and its competitiveness on a class of load balancing instances is simply the maximum competitiveness over all instances in the particular class.

The performance of greedy load balancing with respect to the total latency has been studied in [5, 33]. Awerbuch et al. [5] consider a more general model where each client owns a job with a load vector denoting the impact of the job to each server (i.e., how much the assignment of the job to a server will increase its load) and the objective is to minimize the L_p norm of the load of the servers. In the context similar to the one studied in the current paper, their results imply a $3+2\sqrt{2} \approx 5.8284$ upper bound. This result applies also in the case of weighted clients where the objective is to minimize the weighted average latency. Suri et al. [33] consider the same model as ours and show upper bounds of 17/3 and $2+\sqrt{5} \approx 4.2361$ for different servers and identical servers, respectively. In a way similar to the study of the price of anarchy of congestion games, [33] develops a *greedy inequality* which is used to obtain the upper bounds on competitiveness. They also present a lower bound of 3.0833 for the competitiveness of greedy assignments in the case of identical servers. Christodoulou et al. [12] have analyzed a different than greedy online algorithm for load balancing and proved that it has competitiveness at most $2+\sqrt{5} \approx 4.2361$.

The main question left open by the work of [33] is whether the existence of different servers does hurt the competitiveness of greedy load balancing. We give a positive answer to this question as well. By a rather counterintuitive construction, we show that the 17/3 upper bound of [33] is tight. This is interesting since it indicates that the greedy inequality is powerful enough to characterize the competitiveness of greedy load balancing. We also consider the case of identical servers where we almost close the gap between the upper and lower bounds of [33] by showing that the competitiveness of greedy load balancing is between 4 and $\frac{2}{3}\sqrt{21}+1 \approx 4.05505$. In the proof of the upper bound, we use the greedy inequality but, more importantly, we also use arguments for the structure of greedy and optimal assignments of instances that yield a high competitiveness. In a similar way to the case of selfish load balancing, we argue that such instances can be represented as directed graphs (called *greedy graphs*) that enjoy particular structural properties. In the case of weighted clients, we present a tight lower bound of $3+2\sqrt{2}$ on identical servers matching the upper bound of [5]. The results presented in this paper are summarized in Table 1.

Problem	**Measure**	**Result**	**Comments**
unweighted congestion game	price of stability, upper bound	$1 + 1/\sqrt{3}$	Section 2, Theorem 3. Matches a lower bound from [11]
unweighted load balancing	price of anarchy, lower bound	2.5	Section 3.1, Theorem 4. Matches an upper bound from [6, 10] for unweighted congestion games
unweighted load balancing, identical servers	price of anarchy, upper bound	≈ 2.012	Section 3.3. Matches a lower bound from [33]
unweighted load balancing	competitiveness of greedy, lower bound	17/3	Section 4.1, Theorem 10. Matches an upper bound from [33]
unweighted load balancing, identical servers	competitiveness of greedy, upper bound	≈ 4.055	Section 4.2, Theorem 13. Improves an upper bound from [33]
unweighted load balancing, identical servers	competitiveness of greedy, lower bound	4	Section 4.2, Theorem 14. Improves a lower bound from [33]
weighted load balancing	price of anarchy, lower bound	$\frac{3+\sqrt{5}}{2}$	Section 5, Theorem 15. Matches an upper bound from [6] for congestion games
weighted load balancing, identical servers	price of anarchy, lower bound	2.5	Section 5, Theorem 16
weighted load balancing, identical servers	competitiveness of greedy, lower bound	$3 + 2\sqrt{2}$	Section 5, Theorem 17. Matches an upper bound from [5] for unrelated servers

Table 1: Summary of our results.

Roadmap. The rest of the paper is structured as follows. We present the bounds on the price of stability of linear congestion games in Section 2. The bounds on the price of anarchy of selfish load balancing are presented in Section 3, while the bounds on the competitiveness of greedy load balancing are presented in Section 4. In Section 5 we present extensions of the results to selfish and greedy load balancing when clients are weighted and conclude with open problems in Section 6.

2 The price of stability of linear congestion games

We present a tight upper bound on the price of stability of linear congestion games. Our proof uses the main idea in the proof of [11] and bounds the social cost of any Nash equilibrium having a potential smaller than the potential of the optimal assignment. In the proof we also make use of the Nash inequality which together with the inequality on the potentials yields the upper bound. However, the two inequalities may not be equally important in order to achieve the best possible bound and this is taken into account in our analysis. We now state the Nash inequality (see for example [10, 11, 33]) as applied to our setting. It follows by summing the Nash condition inequalities of all clients.

Lemma 1 (Nash inequality) *For any congestion game, where each resource e has latency function $f_e(x) = \alpha_e x + \beta_e$, with a pure Nash equilibrium and an optimal assignment of n_e and o_e players at each resource e, respectively, it holds that*

$$\sum_e \left(\alpha_e n_e^2 + \beta_e n_e\right) \le \sum_e o_e(\alpha_e n_e + \alpha_e + \beta_e).$$

We use Rosenthal's potential function [29]. We remind that, assuming a strategy profile A for a congestion game with linear latency function $f_j(x) = \alpha_j x + \beta_j$, we define the potential of the strategy to be $Pot(A) = \sum_{j=1}^{m} \sum_{i=1}^{n_j(A)} f_j(i)$, where m is the number of resources, and $n_j(A)$ denotes the number of clients using resource j in A. By its definition, the potential function has the property that for any two assignments differing only in the strategy of a single client, the difference of the potentials and the difference of the cost experienced by that client in the two assignments have the same sign. Furthermore, the potential function has local minima at pure Nash equilibria and, in order to establish an upper bound on the price of stability, it suffices to bound the social cost of pure Nash equilibria whose potential is less than or equal to the potential of the optimal assignment.

In our proof, we will need the following technical lemma.

Lemma 2 *For any non-negative integers x, y and $\gamma = 2\sqrt{3} - 3$, it holds that*

$$(1-\gamma)xy + y - \gamma x + \gamma y^2 \le \left(\frac{1-\gamma}{2}\right)^2 x^2 + (1+\gamma)y^2.$$

Proof: Define the function $g(x, y)$ as the subtraction of the left part from the right part in the above inequality. Substituting γ we have

$$\begin{aligned} g(x,y) &= \left(7-4\sqrt{3}\right)x^2 + y^2 - \left(4-2\sqrt{3}\right)xy - y + \left(2\sqrt{3}-3\right)x \\ &= \left(\left(2-\sqrt{3}\right)x - y + \frac{\sqrt{3}}{2}\right)^2 + \left(\sqrt{3}-1\right)y - \frac{3}{4}. \end{aligned}$$

In order to prove the lemma, it suffices to show that $g(x, y) \ge 0$ for any non-negative integer values of x and y. First, we observe that if $y \ge 2$ it is $\left(\sqrt{3}-1\right)y - \frac{3}{4} \ge 0$. Hence, $g(x, y) \ge 0$ for any integer $y \ge 2$. Also, $g(x, 0) = \left(\left(2-\sqrt{3}\right)x + \frac{\sqrt{3}}{2}\right)^2 - \frac{3}{4} \ge 0$ for any integer $x \ge 0$. For $y = 1$, by trivial calculations we obtain that the parabolic function $g(x, 1) = \left(\left(2-\sqrt{3}\right)x - 1 + \frac{\sqrt{3}}{2}\right)^2 + \sqrt{3} - \frac{7}{4}$ is equal to zero for $x = 0$ and $x = 1$. So, it is non-negative for any non-negative integer value of x. ■

We are now ready to prove the following result. A matching lower bound is presented in [11].

Theorem 3 *The price of stability of congestion games with linear latency functions is at most* $1 + 1/\sqrt{3}$.

Proof: Consider a linear congestion game, an optimal assignment and a pure Nash equilibrium of not larger potential. We will show that the social cost of this Nash equilibrium (and, as a consequence, the social cost of the best Nash equilibrium) is no more than $1 + \frac{1}{\sqrt{3}}$ times the cost of the optimal assignment.

Denote by n_j and o_j the number of clients using resource j in the Nash and optimal assignment, respectively. By the inequality of the potentials, we obtain that

$$\begin{aligned}
\sum_{j=1}^{m}\sum_{i=1}^{n_j} f_j(i) &\leq \sum_{j=1}^{m}\sum_{i=1}^{o_j} f_j(i) \quad \Rightarrow \\
\sum_{j=1}^{m}\left(\alpha_j\left(n_j+1\right)n_j + 2\beta_j n_j\right) &\leq \sum_{j=1}^{m}\left(\alpha_j\left(o_j+1\right)o_j + 2\beta_j o_j\right) \quad \Rightarrow \\
\sum_{j=1}^{m}\left(\alpha_j n_j^2 + \beta_j n_j\right) &\leq \sum_{j=1}^{m}\alpha_j\left(o_j^2 + o_j - n_j\right) + \sum_{j=1}^{m}\beta_j\left(2o_j - n_j\right) \qquad (1)
\end{aligned}$$

By the Nash inequality, we obtain that

$$\sum_{j=1}^{m}\left(\alpha_j n_j^2 + \beta_j n_j\right) \leq \sum_{j=1}^{m}\left(\alpha_j\left(n_j+1\right)o_j + \beta_j o_j\right) \qquad (2)$$

Let $\gamma = 2\sqrt{3} - 3$. By multiplying (1) by γ and (2) by $1-\gamma$ and adding them and using Lemma 2, we obtain that

$$\begin{aligned}
\sum_{j=1}^{m}\left(\alpha_j n_j^2 + \beta_j n_j\right) &\leq \sum_{j=1}^{m}\alpha_j\left(\left(1-\gamma\right)n_j o_j + o_j - \gamma n_j + \gamma o_j^2\right) \\
&\quad + \sum_{j=1}^{m}\beta_j\left(\left(1+\gamma\right)o_j - \gamma n_j\right) \\
&\leq \sum_{j=1}^{m}\alpha_j\left(\left(\frac{1-\gamma}{2}\right)^2 n_j^2 + \left(1+\gamma\right)o_j^2\right) \\
&\quad + \sum_{j=1}^{m}\beta_j\left(\left(\frac{1-\gamma}{2}\right)^2 n_j + \left(1+\gamma\right)o_j\right) \\
&= \left(\frac{1-\gamma}{2}\right)^2 \sum_{j=1}^{m}\left(\alpha_j n_j^2 + \beta_j n_j\right) + \left(1+\gamma\right)\sum_{j=1}^{m}\left(\alpha_j o_j^2 + \beta_j o_j\right).
\end{aligned}$$

Therefore, the price of stability is at most

$$\frac{\sum_{j=1}^{m}\left(\alpha_j n_j^2 + \beta_j n_j\right)}{\sum_{j=1}^{m}\left(\alpha_j o_j^2 + \beta_j o_j\right)} \leq \frac{1+\gamma}{1-\left(\frac{1-\gamma}{2}\right)^2} = 1 + \frac{1}{\sqrt{3}}.$$

■

3 Bounds on the price of anarchy

In this section, we present tight bounds on the price of anarchy. We first show that the known upper bound of Suri et al. [33] on the price of anarchy of load balancing games on different servers with linear latency functions is tight (Section 3.1). Then, we present better upper bounds in the case of identical servers starting from simple bounds that already improve the previous results from [33] (Section 3.2) and concluding with a computer-assisted proof (in Section 3.3) which essentially yields an upper bound matching the corresponding lower bound of [33].

For the study of the price of anarchy of non-symmetric load balancing games, we can consider games in which each client has at most two strategies. This is clearly sufficient when proving lower bounds. In order to prove upper bounds, we observe that for any game, there exists another game with at most two strategies per client which has the same price of anarchy. Given any load balancing game, let O and N be the optimal assignment and the Nash equilibrium that yields the worst social cost for this game, respectively. The game with the same clients and servers in which each client has its strategies in O and N as strategies also has the same optimal assignment and the same Nash equilibrium (and, consequently, the same price of anarchy). We represent such games as directed graphs (called *game graphs*) having a node for each server and a directed edge for each client; the direction of each edge is from the strategy of the client in the optimal assignment to the strategy of the client in the Nash equilibrium. A self-loop indicates that the client has the same strategy in the optimal assignment and the Nash equilibrium.

3.1 Servers with different latency functions

The next theorem states that the upper bound of 5/2 presented in [33] (and also implied by the results in [6, 10] for linear congestion games) is tight. This bound was known to be tight for linear congestion games in general but the constructions in the lower bounds in [6, 10] are not load balancing games.

Theorem 4 *For any $\epsilon > 0$, there is a load balancing game with linear latency functions whose price of anarchy is at least $5/2 - \epsilon$.*

Proof: We construct a game graph G consisting of a complete binary tree with $k+1$ levels and $2^{k+1}-1$ nodes with a line of $k+1$ edges and $k+1$ additional nodes hung at each leaf. So, graph G has $2k+2$ levels $0, ..., 2k+1$, with 2^i nodes at level i for $i = 0, ..., k$ and 2^k nodes at levels $k+1, ..., 2k+1$. The servers corresponding to nodes of level $i = 0, ..., k-1$ have latency functions $f_i(x) = (2/3)^i\, x$, the servers corresponding to nodes of level $i = k, ..., 2k$ have latency functions $f_i(x) = (2/3)^{k-1}\,(1/2)^{i-k}\, x$, and the servers corresponding to nodes of level $2k+1$ have latency functions $f_{2k+1}(x) = (2/3)^{k-1}\,(1/2)^k\, x$.

Consider the assignment where all clients select servers corresponding to the endpoint of their corresponding edge which is closer to the root of the game graph. This assignment is a Nash equilibrium, since servers corresponding to nodes of level $i = 0, ..., k-1$ have two clients and latency $2\,(2/3)^i$, servers corresponding to nodes of level $i = k, ..., 2k$ have one client and latency $(2/3)^{k-1}\,(1/2)^{i-k}$, and servers corresponding to nodes of level $2k+1$ have no client. Therefore, due to the definition of the latency functions, a client assigned to a server corresponding to a node of level $i = 0, ..., 2k$ would experience exactly the same latency if it changed its decision and chose the server corresponding to the node of level $i+1$.

The cost of this assignment is

$$\begin{aligned} cost &= \sum_{i=0}^{k-1} 4 \cdot 2^i \left(2/3\right)^i + \sum_{i=k}^{2k} 2^k \left(2/3\right)^{k-1} \left(1/2\right)^{i-k} \\ &= 15 \left(4/3\right)^k - \left(2/3\right)^{k-1} - 12. \end{aligned}$$

To compute an upper bound for the cost of the optimal assignment, it suffices to consider the assignment where all clients select the servers corresponding to nodes which are further from the root. We obtain that the cost *opt* of the optimal assignment is

$$\begin{aligned} opt &\leq \sum_{i=1}^{k-1} 2^i \left(2/3\right)^i + \sum_{i=k}^{2k} 2^k \left(2/3\right)^{k-1} \left(1/2\right)^{i-k} + 2^k \left(2/3\right)^{k-1} \left(1/2\right)^k \\ &= 6 \left(4/3\right)^k - 4. \end{aligned}$$

Hence, for any $\epsilon > 0$ and for sufficiently large k, the price of anarchy of the game is larger than $5/2 - \epsilon$. ■

3.2 Identical servers

In the case of identical servers with linear latency functions we can show a tight bound on the price of anarchy of approximately 2.012067; a matching lower bound has been presented in [33]. First, we present the main idea in our analysis to obtain a slightly weaker result which already improves the previously known upper bound of $1 + \frac{2}{\sqrt{3}} \approx 2.1547$ [33]. Then, we further improve our analysis.

We will upper-bound the ratio of the social cost of the worst Nash equilibrium to the optimal social cost of games with at most two strategies per client which satisfy a particular property. We say that server j is of type n_j/o_j meaning that it has n_j clients in the Nash equilibrium and o_j clients in the optimal assignment (equivalently, server j has in-degree n_j and out-degree o_j in the game graph). We first show that for any game we can construct another game that has at least the same price of anarchy and, furthermore, satisfies the following 2-*neighborhood property*: in the game graph, the incoming edge of any server of type 1/1 originates from a server of type 0/1. Then, the idea behind the proof is to account for the contribution of servers of type 1/1 and 0/1 in the social cost together.

In general, latency functions would be of the form $f(x) = \alpha x + \beta$ where $\alpha > 0$ and $\beta \geq 0$. Then, the price of anarchy is given by the ratio $\frac{\sum_j \left(\alpha n_j^2 + \beta n_j\right)}{\sum_j \left(\alpha o_j^2 + \beta o_j\right)}$ which is at most $\frac{\sum_j n_j^2}{\sum_j o_j^2}$. Hence, without loss of generality, we may assume that the latency function is of the form $f(x) = x$.

In the proof of our weakest bound (Theorem 6), we make use of the following technical lemma.

Lemma 5 *Let* $\psi = \frac{6+\sqrt{21}}{6}$, $\xi = \frac{7\sqrt{21}-12}{30}$ *and define the functions* $g(x, y) = xy + (1 + \xi) y - \xi x$ *and* $h(x, y) = \frac{1}{4\psi} x^2 + \psi y^2$. *For any non-negative integers* x, y *such that either* $x \neq 1$ *or* $y \neq 1$, *it holds that* $g(x, y) \leq h(x, y)$. *Furthermore,* $g(0, 1) + g(1, 1) = h(0, 1) + h(1, 1)$.

Proof: We start by noting that $g(0, 1) + g(1, 1) = \xi + 3 = \frac{78+7\sqrt{21}}{30}$ and $h(0, 1) + h(1, 1) = 2\psi + \frac{1}{4\psi} = \frac{78+7\sqrt{21}}{30}$. Define the function

$$f(x, y) = h(x, y) - g(x, y) = \left(\frac{1}{2\sqrt{\psi}} x - \sqrt{\psi} y + \xi \sqrt{\psi}\right)^2 + \left(2\xi\psi - \xi - 1\right) y - \xi^2 \psi.$$

In order to prove the lemma, it suffices to show that $f(x,y) \geq 0$ for any non-negative integer values of x and y when either $x \neq 1$ or $y \neq 1$. First, observe that if $y \geq 2$, then $y \geq \frac{\xi^2\psi}{2\xi\psi-\xi-1} = \frac{125\sqrt{21}}{336} - \frac{9}{16} \approx 1.14$, which implies that $(2\xi\psi - \xi - 1)\, y - \xi^2\psi \geq 0$. Hence, $f(x,y) \geq 0$ for any integer $y \geq 2$. Also, $f(x,0) = \left(\frac{1}{2\sqrt{\psi}}x + \xi\sqrt{\psi}\right)^2 - \xi^2\psi \geq 0$ for any integer $x \geq 0$. For $y = 1$, by straightforward calculations we obtain that the parabolic function $f(x,1) = \left(\frac{1}{2\sqrt{\psi}}x + (\xi - 1)\sqrt{\psi}\right)^2 + 2\psi\xi - \xi - \xi^2\psi - 1$ is positive for $x = 0$, negative for $x = 1$ and equal to zero for $x = 2$. So, it is non-negative for any non-negative integer value of x besides 1. ■

Theorem 6 *The price of anarchy of selfish load balancing on identical servers is at most* $\frac{2}{3}\sqrt{21} - 1 \approx 2.05505$.

Proof: Consider a load balancing game on servers with latency function $f(x) = x$ and clients having at most two strategies. Without loss of generality, we may assume that the game satisfies the 2-neighborhood property, i.e., the incoming edge of any server j of type $1/1$ originates from a server of type $0/1$ in the game graph. If this is not the case, we show how to construct another game with not smaller price of anarchy. If a client c had server j as its only strategy (this corresponds to a self-loop in the corresponding game graph), then we may construct a new game by excluding server j and client c from the original one; the new game has worse price of anarchy since both the social cost of the optimal assignment and the social cost of the Nash equilibrium are decreased by 1. So, let j' and j'' be the servers to which server j is connected corresponding to clients c_1 and c_2 selecting servers j' and j in the optimal assignment and servers j and j'' in the Nash assignment, respectively. Clearly, $n_{j''} \leq 2$, since, otherwise, client c_2 would have an incentive to use server j in the Nash equilibrium. Assume that server j' is of type $n_{j'}/o_{j'}$ for $n_{j'} > 0$ or $o_{j'} > 1$. If $n_{j'} > 0$, we can construct a new game by excluding server j and substituting clients c_1 and c_2 by a client selecting server j' in the optimal assignment and server j'' in the Nash assignment. The new game has worse price of anarchy, since both the cost of the optimal assignment and the cost of the Nash equilibrium are decreased by 1. If $o_{j'} > 1$, then we can add a new server j'_1 and change the strategy of client c_1 to $\{j'_1, j\}$. The new game has worse price of anarchy since the cost of the Nash equilibrium remains the same, while the cost of the optimal assignment decreases.

Now, denote by F the set of servers of type $1/1$ and by S the set of servers of type $0/1$ which are connected through an edge to a server in F in the game graph. Also, for each server j in F we denote by $S(j)$ the server of S from which the client destined for j originates. By the Nash inequality, we obtain that $\sum_j n_j^2 \leq \sum_j (o_j n_j + o_j)$ and, since $\sum_j n_j = \sum_j o_j$, we have that

$$
\begin{aligned}
\sum_j n_j^2 &\le \sum_j (n_j o_j + o_j) \\
&= \sum_j \left(n_j o_j + \frac{18+7\sqrt{21}}{30} o_j - \frac{7\sqrt{21}-12}{30} n_j \right) \\
&= \sum_{j \notin F \cup S} g(n_j, o_j) + \sum_{j \in F} \left(g(n_{S(j)}, o_{S(j)}) + g(n_j, o_j) \right) \\
&\le \sum_{j \notin F \cup S} h(n_j, o_j) + \sum_{j \in F} \left(h(n_{S(j)}, o_{S(j)}) + h(n_j, o_j) \right) \\
&= \frac{6-\sqrt{21}}{10} \sum_j n_j^2 + \frac{6+\sqrt{21}}{6} \sum_j o_j^2
\end{aligned}
$$

where the first equality follows since $\sum_j n_j = \sum_j o_j$, the second equality follows by the definition of function g, the second inequality follows by Lemma 5, and the last equality follows by the definition of function h. We obtain that the price of anarchy is

$$
\frac{\sum_j n_j^2}{\sum_j o_j^2} \le \frac{2}{3}\sqrt{21} - 1.
$$

■

3.3 Tightening the analysis

The main idea in order to improve the analysis in the proof of Theorem 6 is to strengthen the properties of the games that have to be considered and account for the contributions of servers of type 1/1 together with the servers in their neighborhood in the game graph (in a way that is well defined below). By extending the neighborhood that we consider, we obtain better and better upper bounds which converge to the lower bound of 2.012067 presented in [33]. We present the formal proof for an upper bound of 2.029656 and a series of better bounds obtained using more complicated computer-assisted proofs.

Consider the game graph of the game satisfying the 2-neighborhood property. We call a 4-*path* any directed path of at most four nodes in the game graph starting with a server of type 0/1 and having a server of type 1/1 as its second node. Let p be such a path starting with the server j_0 and containing server j_1 as its second node. Denote by j_2 the third server in the path p and assume that j_2 is of type n_{j_2}/o_{j_2}. The path may terminate at server j_2 if it has no outgoing edges in the game graph. Otherwise, path p has a fourth server j_3 of type n_{j_3}/o_{j_3}. An example of a 4-path is presented in Figure 1. We say that a game satisfies the 4-*neighborhood property* if in any 4-path, the third server j_2 has $n_{j_2} = 2$ and the fourth server j_3, if it exists, has $n_{j_3} = 3$ and neither of them has any self-loop.

We show that given any game that satisfies the 2-neighborhood property and has price of anarchy at least 5/3, there exists a game satisfying the 4-neighborhood property which has at least the same price of anarchy. Consider a 4-path consisting of servers j_0, j_1, j_2 and possibly j_3.

Since j_2 is connected to at least server j_1, it has $n_{j_2} > 0$. If $n_{j_2} \ge 3$, then client c_2 between j_1 and j_2 would have an incentive to use server j_1 in the Nash equilibrium. If $n_{j_2} = 1$, then we

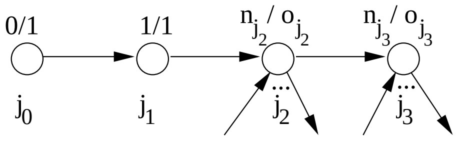


Figure 1: An example of a 4-path.

can replace the clients c_1 and c_2 by a new client selecting server j_0 in the optimal assignment and server j_2 in the Nash assignment and obtain a game with higher price of anarchy, since both the cost of the optimal assignment and of the Nash equilibrium are decreased by 1.

If server j_3 exists, then since j_3 is connected to at least server j_2, it has $n_{j_3} \geq 1$. If $n_{j_3} = 1$, then we can introduce a new server j^* and change the strategy set of client c_3 to $\{j^*, j_3\}$ to obtain another game in which server j_3 is connected to a server of type 0/1. The social cost of the Nash assignment is the same while the optimal social cost does not increase. If $n_{j_3} = 2$, we distinguish between two cases for o_{j_2}. If $o_{j_2} = 1$, then we may remove servers j_0, j_1, and j_2 and clients c_1, c_2 and c_3 and change the strategy of the client c_4 which is the second client connected to server j_2 in the Nash equilibrium so that it connects to j_3 instead of j_2. In this way, the social cost of the Nash assignment is decreased by 5 while the optimal social cost is decreased by 3; overall the price of anarchy increases since the original game had price of anarchy larger than 5/3. If $o_{j_2} > 1$, we remove the client c_3, change the strategy of client c_4 so that it connects to server j_3 in the Nash equilibrium and introduce two new servers j_0^* and j^* of types 0/1 and 1/1, respectively, and clients c_5 and c_6 connecting j_0^* to j^* and j^* to j_2, respectively. The social cost of the Nash assignment is increased by 1 while the optimal cost decreases by at least 1. Overall, the price of anarchy increases. Clearly, if $n_{j_3} \geq 4$, then client c_3 would have an incentive to use server j_2.

It remains to show that neither j_2 nor j_3 have self-loops. We will actually show that any game whose game graph has self-loops at nodes of in-degree 2 or 3 can be converted to a game without such self-loops with higher price of anarchy.

Lemma 7 *For any game, there exists a game having at least the same price of anarchy and whose game graph has no self-loops at nodes with in-degree* 2 *and* 3*.*

Proof: Starting from a game whose game graph has self-loops at some nodes of in-degree 2 (respectively, 3), we will construct another game whose game graph has no self-loops at nodes of in-degree 2 (resp., 3) and has higher price of anarchy.

Consider a game with game graph G that has t self-loops at nodes of in-degree 2 (resp., 3). Construct the graph G' by first putting twelve (resp., twenty) copies of G. Denote by L the set of self-loops at nodes of G with in-degree 2 (resp., 3).

For each self-loop in L we apply the following procedure in order to augment G'. Let v be the node of G with in-degree 2 (resp., 3) having the self-loop. Connect the outgoing edges of twelve (resp., twenty) constructions like the one in Figure 2a (resp., Figure 2c) to the twelve (resp., twenty) copies of node v, with one outgoing edge connected to each copy. Furthermore, connect the twelve (resp., twenty) copies of v to the input edges of the construction of Figure 2b (resp., Figure 2d) and remove the twelve (resp., twenty) copies of the self-loop from G'. An example is depicted in Figure 2e.

We first show that graph G' is a game graph, i.e., the assignment where each client selects the server to which the corresponding edge points to is a Nash equilibrium. Notice that each

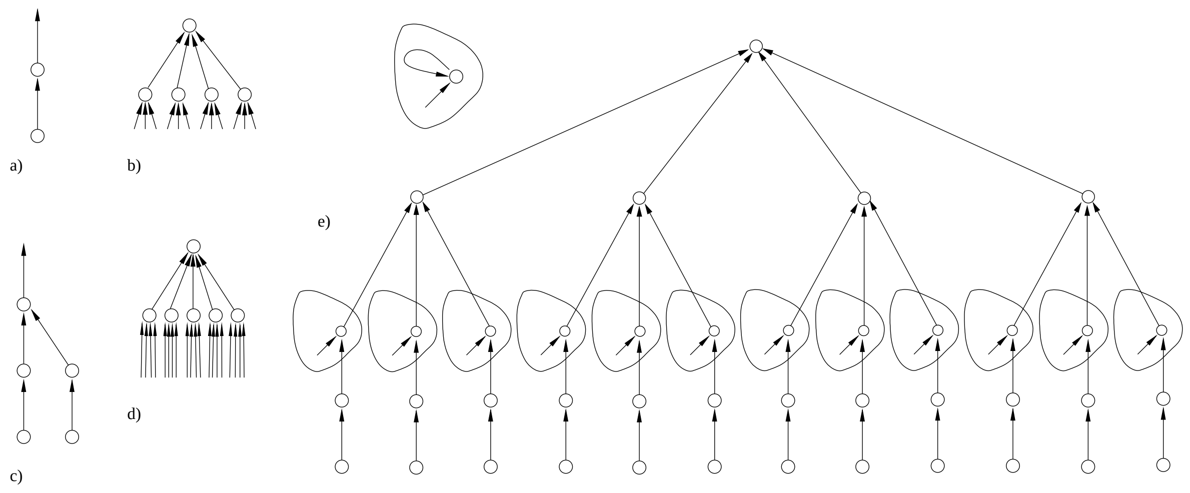


Figure 2: a-d) Constructions used in the proof of Lemma 7. e) Connecting the copies of a node with a self-loop and in-degree 2 in the original game graph with the constructions in a) and b).

copy of a node of G has in- and out-degree in G' equal to those of the corresponding node in G. Hence, a client corresponding to any edge of G' which is a copy of an edge in G has no incentive to deviate (since the client corresponding to the edge in G had no incentive to deviate either). Also, edges with at least one endpoint in the constructions of Figures 2a, 2b, 2c, and 2d point from a node of in-degree i to a node of in-degree $i+1$ (for $i = 0, 1, ..., 4$). Hence, the corresponding client would experience the same latency if it changed its strategy and, hence, no such client has an incentive to deviate either.

We will now show that the new game has higher price of anarchy than the original one. In order to show this, we will also use the fact that the price of anarchy of the original game is at most $\frac{2}{3}\sqrt{21} - 1$ which follows by Theorem 6.

Denote by *cost* the social cost of the Nash equilibrium of the original game (where each client selects the server to which the corresponding edge points in the game graph G) and by *opt* the cost of the assignment where each client selects the server from which the corresponding edge originates in the game graph G. For the case of nodes of in-degree 2, the cost of the Nash equilibrium for the new game is

$$12 \cdot \mathit{cost} + 12t \cdot 1^2 + 4t \cdot 3^2 + t \cdot 4^2 = 12 \left(\mathit{cost} + \frac{16t}{3} \right).$$

The first term comes from the contribution of the nodes in the twelve copies of G, the second term comes from the contribution of the nodes in the constructions of Figure 2a, while the last two terms come from the contribution of nodes in the construction of Figure 2b. Similarly, the assignment where all clients select the server at the origin of the corresponding edge has cost

$$12 \cdot \mathit{opt} + 24t \cdot 1^2 + 4t \cdot 1^2 = 12 \left(\mathit{opt} + \frac{7t}{3} \right).$$

The first term comes from the contribution of the nodes in the twelve copies of G, the second term comes from the contribution of nodes in the constructions of Figure 2a, while the last term comes from the contribution of nodes in the constructions of Figure 2b.

So, the price of anarchy of the new game is at least

$$\frac{cost + 16t/3}{opt + 7t/3} > \frac{cost}{opt},$$

since, by Theorem 6, it is $cost/opt \leq \frac{2}{3}\sqrt{21} - 1 < 16/7$.

Respectively, for the case of nodes with in-degree 3, the cost of the Nash equilibrium of the new game is

$$20 \cdot cost + 40t \cdot 1^2 + 20t \cdot 2^2 + 5t \cdot 4^2 + t \cdot 5^2 = 20\left(cost + \frac{45t}{4}\right)$$

while the assignment where all clients select the server at the origin of the corresponding edge has cost

$$20 \cdot opt + 100t \cdot 1^2 + 5t \cdot 1^2 = 20\left(opt + \frac{21t}{4}\right).$$

So, the price of anarchy of the new game is at least

$$\frac{cost + 45t/4}{opt + 21t/4} > \frac{cost}{opt},$$

since, by Theorem 6, it is $cost/opt \leq \frac{2}{3}\sqrt{21} - 1 < 45/21$. ■

Given a server j in a 4-path p of a game satisfying the 4-neighborhood property, we define $part(j,p)$ as follows. For servers j_0, j_1, j_2 and j_3 (if it exists), it is

$$\begin{aligned} part(j_0,p) = part(j_1,p) &= \begin{cases} 1, & \text{if } o_{j_2} = 0 \\ \frac{1}{o_{j_2}}, & \text{if } o_{j_2} > 0 \end{cases} \\ part(j_2,p) &= \begin{cases} \frac{1}{2}, & \text{if } o_{j_2} = 0 \\ \frac{1}{2o_{j_2}}, & \text{if } o_{j_2} > 0 \end{cases} \\ part(j_3,p) &= \frac{1}{6}. \end{aligned}$$

Denote by P_4 the set of all 4-paths in the game graph. The above definition satisfies that $\sum_{p \in P_4} part(j,p) \leq 1$, for each server j and, in particular, $\sum_{p \in P_4} part(j,p) = 1$, for each server of type 1/1. Intuitively, we amortize the contribution of a server to the social cost over the 4-paths containing that server, and $part(j,p)$ is the fraction of the contribution of j that we assign to path p.

In our proof of the stronger upper bound, we use the following two technical lemmas. Let $\psi = \frac{34+\sqrt{629}}{34}$ and $\xi = \frac{37\sqrt{629}-204}{1054}$. We define the functions $g(x,y) = xy + (1+\xi)y - \xi x$ and $h(x,y) = \frac{1}{4\psi}x^2 + \psi y^2$.

Lemma 8 *For any non-negative integers x, y such that either $x \neq 1$ or $y \neq 1$, it holds that $g(x,y) \leq h(x,y)$.*

Proof: Define the function

$$f(x,y) = h(x,y) - g(x,y) = \left(\frac{1}{2\sqrt{\psi}}x - \sqrt{\psi}y + \xi\sqrt{\psi}\right)^2 + (2\xi\psi - \xi - 1)\,y - \xi^2\psi.$$

In order to prove the lemma, it suffices to show that $f(x,y) \geq 0$ for any non-negative integer values of x and y when either $x \neq 1$ or $y \neq 1$. First, observe that if $y \geq 2$, then $y \geq \frac{\xi^2\psi}{2\xi\psi-\xi-1} \approx 1.17$, which implies that $(2\xi\psi - \xi - 1)\,y - \xi^2\psi \geq 0$. Hence, $f(x,y) \geq 0$ for any integer $y \geq 2$. Also, $f(x,0) = \left(\frac{1}{2\sqrt{\psi}}x + \xi\sqrt{\psi}\right)^2 - \xi^2\psi \geq 0$ for any integer $x \geq 0$. For $y = 1$, by straightforward calculations we obtain that the parabolic function $f(x,1) = \left(\frac{1}{2\sqrt{\psi}}x + (\xi-1)\sqrt{\psi}\right)^2 + 2\psi\xi - \xi - \xi^2\psi - 1$ is positive for $x = 0$, negative for $x = 1$ and equal to zero for $x = 2$. So, it is non-negative for any non-negative integer value of x besides 1. ∎

Lemma 9 *For any* 4*-path* p*, it holds that*

$$\sum_{j\in p} part(j,p) g(n_j,o_j) \leq \sum_{j\in p} part(j,p) h(n_j,o_j).$$

Proof: We consider a 4-path p with servers j_0, j_1, j_2 of type 0/1, 1/1, $2/o_{j_2}$ and a fourth server j_3 of type $3/o_{j_3}$ if $o_{j_2} > 0$. We distinguish between the cases $o_{j_2} = 0$ and $o_{j_2} > 0$.

In the first case, by simple calculations, we show that $g(0,1)+g(1,1)+\frac{1}{2}g(2,0) \leq h(0,1)+h(1,1)+\frac{1}{2}h(2,0)$.

In the second case, we have to show that $g(0,1) + g(1,1) + \frac{1}{2}g(2,o_{j_2}) + \frac{o_{j_2}}{6}g(3,o_{j_3}) \leq h(0,1) + h(1,1) + \frac{1}{2}h(2,o_{j_2}) + \frac{o_{j_2}}{6}h(3,o_{j_3})$. By straightforward calculations, we obtain that $g(2,o_{j_2}) - h(2,o_{j_2}) \leq g(2,1) - h(2,1) = 0$ for any $o_{j_2} \geq 1$, and that $g(3,o_{j_3}) - h(3,o_{j_3}) \leq g(3,1) - h(3,1)$ for any $o_{j_3} \geq 0$. So, the proof completes by showing that $g(0,1) + g(1,1) + \frac{1}{6}g(3,1) \leq h(0,1) + h(1,1) + \frac{1}{6}h(3,1)$. ∎

By using the fact that $\sum_j n_j = \sum_j o_j$, the definition of functions g and h, and Lemmas 8 and 9, we obtain that

$$\begin{aligned}
\sum_j n_j^2 &\leq \sum_j (n_j o_j + o_j) = \sum_j (n_j o_j + (1+\xi)o_j - \xi n_j) = \sum_j g(n_j,o_j) \\
&= \sum_{p\in P_4}\sum_{j\in p} g(n_j,o_j) part(j,p) + \sum_j \left(1 - \sum_{p\in P_4} part(j,p)\right) g(n_j,o_j) \\
&\leq \sum_{p\in P_4}\sum_{j\in p} h(n_j,o_j) part(j,p) + \sum_j \left(1 - \sum_{p\in P_4} part(j,p)\right) h(n_j,o_j) \\
&= \sum_j h(n_j,o_j) = \frac{1}{4\psi}\sum_j n_j^2 + \psi \sum_j o_j^2
\end{aligned}$$

which yields that the price of anarchy is

$$\frac{\sum_j n_j^2}{\sum_j o_j^2} \leq \frac{4\psi^2}{4\psi - 1} = \frac{323 - 6\sqrt{629}}{85} \approx 2.029656.$$

The analysis can be extended by considering games satisfying the *κ-neighborhood property* for $\kappa > 4$. We call a *κ-path* any directed path of at most κ nodes in the game graph starting with a server of type 0/1 and having a server of type 1/1 as its second node. A game satisfies the κ-neighborhood property if it satisfies the $(\kappa-1)$-neighborhood property and for any

κ-path in the game graph, the κ-th node, if it exists, has in-degree $\kappa - 1$ and no self-loops. Again, it can be shown that for any game there exists a game satisfying the κ-neighborhood property having at least the same price of anarchy. In order to upper-bound the price of anarchy, we define $part(j,p)$ which denotes how much server j participates in the κ-path p and the functions $g_\kappa(x,y) = xy + (1+\xi_\kappa)y - \xi_\kappa x$ and $h_\kappa(x,y) = \frac{1}{4\psi_\kappa}x^2 + \psi_\kappa y^2$. We seek for values of ψ_κ and ξ_κ so that the functions g_κ and h_κ satisfy lemmas similar to Lemmas 8 and 9 that minimize $\frac{4\psi_\kappa^2}{4\psi_\kappa - 1}$. Then, the analysis continues in the same way as in the proof above. We have implemented the proof in a C program for values $\kappa = 5, ..., 15$. The bounds obtained are depicted in Figure 3.

κ	Upper Bound	Lower Bound
4	2.029656065	1.8
5	2.019343848	1.9375
6	2.015325799	1.970588235
7	2.013332672	1.994252874
8	2.012388288	2.005703422
9	2.012186496	2.0100271
10	2.012110246	2.011232914
11	2.012080068	2.011769481
12	2.012071449	2.011970945
13	2.012068514	2.01202926
14	2.012067464	2.012053615
15	2.012067113	2.012062622

Figure 3: Upper bounds obtained for $\kappa = 4, ..., 15$. The third column has the lower bounds obtained by the constructions of [33] with $\kappa + 1$ levels.

For $\kappa = 4, ..., 15$, by appropriately defining $part(j,p)$, the κ-paths that make the inequality of Lemma 9 tight consist of the first κ servers with types in the following sequence 0/1, 1/1, 2/1, 3/1, 4/2, 5/2, 6/2, 7/2, 8/3, 9/3, 10/3, 11/4, 12/4, 13/4, 14/5. These are essentially the lower bound constructions of [33]. For $\kappa \geq 4$, such a construction with $\kappa + 1$ levels has the servers in the first κ levels to be of type in the above sequence while servers of the last level κ are of type $\kappa/0$.

4 Greedy load balancing

In this section we study greedy load balancing by focusing on servers with linear latency functions. Similarly to the case of selfish load balancing, in the study of the competitiveness of greedy load balancing we consider load balancing instances in which each client has at most two strategies. This is clearly sufficient when proving lower bounds. In order to prove upper bounds, we observe that for any instance, there exists another instance with at most two strategies per client for which greedy has the same competitiveness. Given any load balancing instance, let O and N be the optimal assignment and the greedy assignment of the highest cost for this instance, respectively. The instance with the same clients and servers in which each client has its strategies in O and N as strategies also has the same optimal assignment and the same greedy assignment (and, consequently the same competitiveness). We represent such instances as directed graphs (called *greedy graphs*) having a node for each server and a directed edge with timing information for each client; the direction of each edge is from the strategy of the client in the optimal assignment to the strategy of the client in the greedy assignment and the timing information denotes the time the client appears.

The cost of an assignment is again the total latency. When each server j has been assigned n_j clients and its latency function is $f_j(x) = \alpha_j x + \beta_j$, then the total cost of the greedy assignment equals $\sum_j \left(\alpha_j n_j^2 + \beta_j n_j\right)$. As discussed in [33], the greedy algorithm does not necessarily lead to equilibrium assignments. In the greedy algorithm, each client is essentially choosing the best possible server at the time it makes its decision. When all servers have the same latency function $f(x) = \alpha x + \beta$, each client c is simply choosing the server with the minimum number of clients.

4.1 Different servers

First, we show that the upper bound of [33] for different servers is tight.

Theorem 10 *For any $\epsilon > 0$, greedy load balancing has competitiveness at least $17/3 - \epsilon$.*

Proof: We first present an instance that yields a lower bound arbitrarily close to 5. We construct an instance $I_k(\alpha, t)$ represented by a greedy graph which is a complete binary tree with k levels $0, 1, ..., k-1$ and with its edges directed towards the root. Denote by S_i the set of servers at level i. The root R has latency function $f_R(x) = \alpha_R x = \alpha x$ and the latency functions $f_s(x) = \alpha_s x$ for the other nodes are defined as follows: For $i = 0, ..., k-2$, given a server s at level i, its left child $\ell(s)$ has $\alpha_{\ell(s)} = \alpha_s/5$ and the right child $r(s)$ has $\alpha_{r(s)} = 3\alpha_s/5$. Given a server s at level $k-2$, its left child $\ell(s)$ has $\alpha_{\ell(s)} = \alpha_s$ and its right child $r(s)$ has $\alpha_{r(s)} = 3\alpha_s$. These definitions yield $\sum_{s\in S_i} \alpha_s = \alpha(4/5)^i$ for $i = 0, ..., k-2$, and $\sum_{s\in S_{k-1}} \alpha_s = 4\alpha(4/5)^{k-2}$. Given any non-leaf server s, the client connecting s with its left child appears prior to the client connecting s with its right child and, if s is not the root, the client connecting s to its parent appears after the clients connecting s with its children. The client connecting the root with its right child has timing t and is the client that arrives last.

Consider the assignment where each client selects the server corresponding to the node closer to the root. We will show that this is a greedy assignment. Indeed, consider the two clients c_1 and c_2 connecting a server s to $\ell(s)$ and $r(s)$, respectively, and recall that c_1 appears before c_2 and, furthermore, the client connecting s to its parent (if $s \neq R$) appears after c_2. We first consider the case when $\ell(s)$ and $r(s)$ are leaves. When c_1 appears, both $\ell(s)$ and s have zero clients and the same latency functions, and, therefore, we can assign c_1 to s, since the increase of the cost is α_s for both choices. Moreover, when c_2 appears, we can also assign it to s since $r(s)$ has zero clients and latency function $f_{r(s)}(x) = 3\alpha_s x$ and s has one client and latency function $f_s(x) = \alpha_s x$. Thus, the increase in the cost is $3\alpha_s$ for both choices. Now, assume that $\ell(s)$ and $r(s)$ are not leaves. When c_1 appears, server $\ell(s)$ has already two clients and latency function $f_{\ell(s)}(x) = \alpha_s x/5$, while server s has zero clients and latency function $f_s(x) = \alpha_s x$. Therefore, we can assign c_1 to s, since the increase in the cost is α_s for both choices. Moreover, when c_2 appears, we can also assign it to s since $r(s)$ has two clients and latency function $f_{r(s)}(x) = 3\alpha_s x/5$, while server s has one client and latency function $f_s(x) = \alpha_s x$. Thus, the increase in the cost is $3\alpha_s$ for both choices.

In order to bound the optimal cost it suffices to consider the assignment of each client to the server which is closer to the leaves. Denote by $opt(I_k(\alpha,t))$ and $gr(I_k(\alpha,t))$ the optimal cost and the cost of the greedy assignment of an instance $I_k(\alpha,t)$, respectively. We have $opt(I_k(\alpha,t)) \leq \sum_{i=1}^{k-1} \sum_{s\in S_i} \alpha_s = 4\alpha$ and $gr(I_k(\alpha,t)) = 4\sum_{i=0}^{k-2} \sum_{s\in S_i} \alpha_s = 20\alpha\left(1 - (4/5)^{k-1}\right)$.

We will now use the instance described above to obtain the 17/3 lower bound. We construct a greedy graph consisting of a complete binary tree with k levels and with its edges directed towards the root. Denote by S'_i the set of servers of level i. The root R has latency function $f_R(x) = x$. The latency functions for the other servers are defined as follows. For $i = 0, ..., k-2$, given a server s at level i, its left child $\ell(s)$ has $\alpha_{\ell(s)} = 3\alpha_s/7$ and the right child $r(s)$ has $\alpha_{r(s)} = 5\alpha_s/7$. Given a server s at level $k-2$, its left child $\ell(s)$ has $\alpha_{\ell(s)} = 3\alpha_s/5$ and its right child $r(s)$ has $\alpha_{r(s)} = \alpha_s$. These definitions yield $\sum_{s\in S'_i} \alpha_s = (8/7)^i$ for $i = 0, ..., k-2$, and $\sum_{s\in S'_{k-1}} \alpha_s = \frac{8}{5}(8/7)^{k-2}$. Given any server s which is not a leaf, the client connecting s to its left child $\ell(s)$ has timing $t_{\ell(s)}$ and the client connecting s to its right child $r(s)$ has timing $t_{r(s)}$ with $t_{\ell(s)} < t_{r(s)}$. If s is not the root, the client connecting s to its parent appears after the clients connecting s to its children.

We augment the instance as follows: For each leaf s of the binary tree connected with its parent node through an edge of timing t, we include a copy $I_k^{s,1}$ of instance $I_k(\alpha_s/5, t-3)$ and a copy $I_k^{s,2}$ of instance $I_k(3\alpha_s/5, t-3)$ whose roots $R_{s,1}$ and $R_{s,2}$ are connected through edges of timing $t-2$ and $t-1$ with s. For each non-leaf node s, we include a copy I_k^s of instance $I_k(\alpha_s/5, t_{\ell(s)} - 2)$ whose root R_s is connected through an edge of timing $t_{\ell(s)} - 1$ with s, where $t_{\ell(s)}$ is the timing of the edge connecting s with its left child in the binary tree. The construction is depicted in Figure 4.

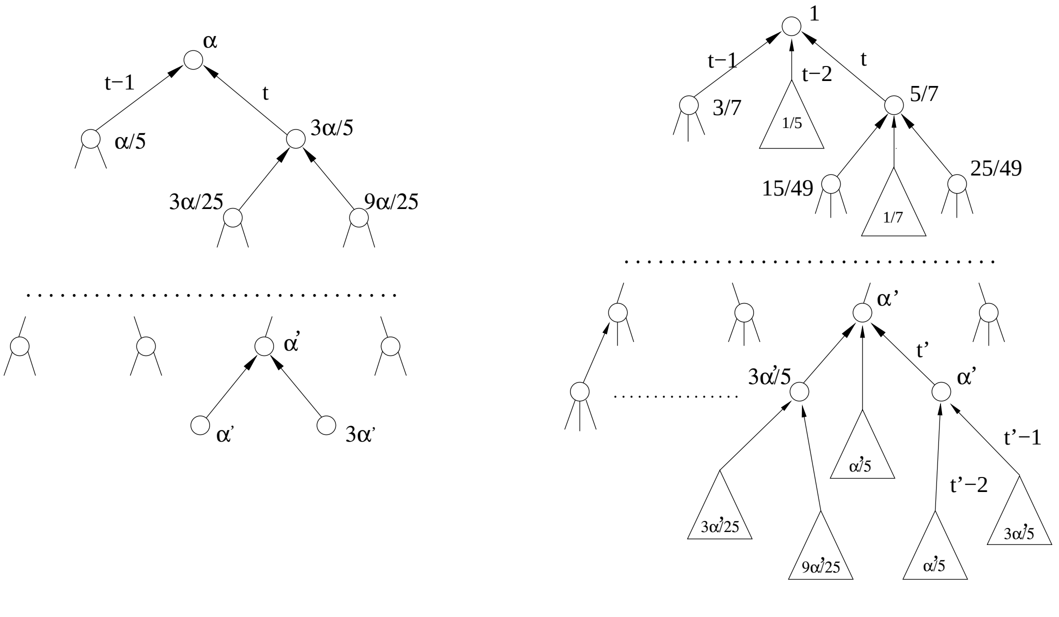


Figure 4: The construction in the proof of Theorem 10. The instance $I(\alpha, t)$ is depicted at the left and is used as a triangle in the construction at the right part.

Consider the assignment where each client selects the server corresponding to the node closer to the root. We will show that this is a greedy assignment by arguing about the choices of all different sets of clients. We begin by considering clients corresponding to edges inside the copies (of I_k^s when s in not a leaf and $I_k^{s,1}$ and $I_k^{s,2}$ when s is a leaf). Clearly, since those edges arrive prior to the edges connecting the copies to the binary tree and by the discussion above, the assignment of those edges to the nodes closer to the root is a valid outcome of the greedy algorithm. We now examine clients corresponding to edges outside the copies. Consider client c_1 corresponding to an edge $e = (R_{s,1}, s)$ connecting the root $R_{s,1}$ of copy

$I_k^{s,1}$ to a leaf s of the binary tree. $R_{s,1}$ has latency function $f_{R_{s,1}}(x) = \alpha_s x/5$ and already two clients, while s has latency function $f_s(x) = \alpha_s x$ and no clients. So, the increase in the total cost will be equal to α_s for both choices of client c_1. Similary, consider client c_2 corresponding to an edge $e = (R_{s,2}, s)$ connecting the root $R_{s,2}$ of copy $I_k^{s,2}$ to a leaf s of the binary tree. $R_{s,2}$ has latency function $f_{R_{s,2}}(x) = 3\alpha_s x/5$ and already two clients, while s has latency function $f_s(x) = \alpha_s x$ and one client. So, the increase in the total cost will be equal to $3\alpha_s$ for both choices of client c_2. Now, consider client c_3 corresponding to an edge $e = (R_s, s)$ connecting the root R_s of copy I_k^s to a non-leaf s of the binary tree. R_s has latency function $f_{R_s}(x) = \alpha_s x/5$ and already two clients, while s has latency function $f_s(x) = \alpha_s x$ and no clients. So, the increase in the total cost will be equal to α_s for both choices of client c_3. Until now, we have argued about clients connecting copies of instance $I_k(\alpha, t)$ to the binary tree. We proceed to handle the clients connecting nodes of the binary tree to their parents. Consider client c_4 corresponding to an edge $e = (\ell(s), s)$ connecting the left child $\ell(s)$ to its parent s, for the case where $\ell(s)$ is a leaf of the binary tree. $\ell(s)$ has latency function $f_{\ell(s)}(x) = 3\alpha_s x/5$ and already two clients, while s has latency function $f_s(x) = \alpha_s x$ and one client. So, the increase in the total cost will be equal to $3\alpha_s$ for both choices of client c_4. Similarly, consider client c_5 corresponding to an edge $e = (r(s), s)$ connecting the right child $r(s)$ to its parent s, for the case where $r(s)$ is a leaf of the binary tree. Both $r(s)$ and s have already two clients and the same latency functions. Therefore, the increase in the total cost will be equal to $5\alpha_s$ for both choices of client c_5. Moreover, consider client c_6 corresponding to an edge $e = (\ell(s), s)$ connecting the left child $\ell(s)$ to its parent s, for the case where $\ell(s)$ is not a leaf of the binary tree. $\ell(s)$ has latency function $f_{\ell(s)}(x) = 3\alpha_s x/7$ and already three clients, while s has latency function $f_s(x) = \alpha_s x$ and one client. So, the increase in the total cost will be equal to $3\alpha_s$ for both choices of client c_6. Finally, consider client c_7 corresponding to an edge $e = (r(s), s)$ connecting the right child $r(s)$ to its parent s, for the case where $r(s)$ is not a leaf of the binary tree. $r(s)$ has latency function $f_{r(s)}(x) = 5\alpha_s/7$ and already three clients, while s has latency function $f_s(x) = \alpha_s x$ and already two clients. Therefore, the increase in the total cost will be equal to $5\alpha_s$ for both choices of client c_7.

In order to bound the optimal cost it suffices to consider the assignment of each client to the server which is closer to the leaves. In this assignment, each non-root server is assigned one client. Therefore, the cost of the servers that form I_k^R is $1/5 + \mathit{opt}(I_k^R)$, the cost of each non-root node s is α_s, the cost of the servers that form I_k^s is $\alpha_s/5+\mathit{opt}(I_k^s)$, while the cost of the servers that form $I_k^{s,1}$ and $I_k^{s,2}$ is $\alpha_s/5 + \mathit{opt}(I_k^{s,1})$ and $3\alpha_s/5 + \mathit{opt}(I_k^{s,2})$, respectively. Denote by *opt* and *gr* the optimal cost and the cost of the greedy assignment of the construction. We have that

$$\begin{aligned}
\mathit{opt} &\leq \frac{1}{5} + \mathit{opt}(I_k^R) + \sum_{i=1}^{k-2}\sum_{s\in S'_i}\left(\frac{6\alpha_s}{5} + \mathit{opt}(I_k^s)\right) + \sum_{s\in S'_{k-1}}\left(\frac{9\alpha_s}{5} + \mathit{opt}(I_k^{s,1}) + \mathit{opt}(I_k^{s,2})\right)\\
&\leq \frac{1}{5} + \frac{4}{5} + \sum_{i=1}^{k-2}\sum_{s\in S'_i}\left(\frac{6\alpha_s}{5} + \frac{4\alpha_s}{5}\right) + \sum_{s\in S'_{k-1}}\left(\frac{9\alpha_s}{5} + \frac{4\alpha_s}{5} + \frac{12\alpha_s}{5}\right)\\
&= 1 + 2\sum_{i=1}^{k-2}\sum_{s\in S'_i}\alpha_s + 5\sum_{s\in S'_{k-1}}\alpha_s\\
&= 21(8/7)^{k-1} - 15,
\end{aligned}$$

and

$$\begin{aligned}
gr &= \sum_{i=0}^{k-2}\sum_{s\in S'_i}\left(9\alpha_s+gr(I_k^s)\right)+\sum_{s\in S'_{k-1}}\left(4\alpha_s+gr(I_k^{s,1})+gr(I_k^{s,2})\right)\\
&= \sum_{i=0}^{k-2}\sum_{s\in S'_i}\left(9\alpha_s+4\alpha_s\left(1-(4/5)^{k-1}\right)\right)+\sum_{s\in S'_{k-1}}\left(4\alpha_s+16\alpha_s\left(1-(4/5)^{k-1}\right)\right)\\
&= \left(13-4(4/5)^{k-1}\right)\sum_{i=0}^{k-2}\sum_{s\in S'_i}\alpha_s+\left(20-16(4/5)^{k-1}\right)\sum_{s\in S'_{k-1}}\alpha_s\\
&= 119(8/7)^{k-1}-91+28(4/5)^{k-1}-\frac{252}{5}(32/35)^{k-1}.
\end{aligned}$$

We conclude that for any $\epsilon>0$ and for sufficiently large k, the competitiveness of the greedy assignment is at least $17/3-\epsilon$. ■

4.2 Identical servers

We also study the case of identical servers with latency function $f(x)=x$. By reasoning about the structure of load balancing instances of particular properties, we prove an upper bound of $\frac{2}{3}\sqrt{21}+1\approx 4.05505$ (Theorem 13) on the competitiveness of the greedy algorithm, improving the previous bound of $2+\sqrt{5}\approx 4.2361$ from [33].

In our proof, we use the greedy inequality developed in [33] as well as a technical lemma (Lemma 12).

Lemma 11 (Greedy inequality, Suri et al. [33]) *For any load balancing instance on servers with latency functions $f_j(x)=\alpha_j x+\beta_j$, with a greedy and an optimal assignment of n_j and o_j clients at each server j, respectively, $\sum_j\left(\alpha_j n_j^2+\beta_j n_j\right)\le\sum_j o_j(2\alpha_j n_j+\alpha_j+\beta_j)$.*

Lemma 12 *Let $\psi=\frac{9+\sqrt{21}}{6}$ and $\xi=\frac{7\sqrt{21}-3}{30}$ and define the functions $g(x,y)=2xy+(1+\xi)y-\xi x$ and $h(x,y)=\frac{1}{\psi}x^2+\psi y^2$. For any non-negative integers x,y such that either $x\neq 1$ or $y\neq 1$, it holds that $g(x,y)\le h(x,y)$. Furthermore, $g(0,1)+g(1,1)=h(0,1)+h(1,1)$.*

Proof: We start by noting that $g(0,1)+g(1,1)=\xi+4=\frac{117+7\sqrt{21}}{30}$ and $h(0,1)+h(1,1)=2\psi+\frac{1}{\psi}=\frac{117+7\sqrt{21}}{30}$. Define the function

$$f(x,y)=h(x,y)-g(x,y)=\left(\frac{1}{\sqrt{\psi}}x-\sqrt{\psi}y+\frac{\xi\sqrt{\psi}}{2}\right)^2+(\xi\psi-\xi-1)\,y-\frac{\xi^2\psi}{4}.$$

In order to prove the lemma, it suffices to show that $f(x,y)\ge 0$ for any non-negative integer values of x and y when either $x\neq 1$ or $y\neq 1$. First, observe that if $y\ge 3$, then $y\ge\frac{\xi^2\psi}{4(\xi\psi-\xi-1)}=1+\frac{25\sqrt{21}}{84}\approx 2.36$, which implies that $(\xi\psi-\xi-1)\,y-\frac{\xi^2\psi}{4}\ge 0$. Hence, $f(x,y)\ge 0$ for any integer $y\ge 3$. Also, $f(x,0)=\left(\frac{1}{\sqrt{\psi}}x+\frac{\xi\sqrt{\psi}}{2}\right)^2-\frac{\xi^2\psi}{4}\ge 0$ for any integer $x\ge 0$. For $y=1$, by straightforward calculations we obtain that the parabolic function $f(x,1)=\left(\frac{1}{\sqrt{\psi}}x-\sqrt{\psi}+\frac{\xi\sqrt{\psi}}{2}\right)^2+\xi\psi-\xi-1-\frac{\xi^2\psi}{4}$ is positive for $x=0$, negative for $x=1$ and equal to zero for $x = 2$. So, it is non-negative for any non-negative integer value of x besides 1. For $y = 2$, by straightforward calculations we obtain that the parabolic function $f(x,2) = \left(\frac{1}{\sqrt{\psi}}x - 2\sqrt{\psi} + \frac{\xi\sqrt{\psi}}{2}\right)^2 + 2(\xi\psi - \xi - 1) - \frac{\xi^2\psi}{4}$ is equal to zero for $x = 3$ and positive for $x = 2$ and $x = 4$. So, it is non-negative for any non-negative integer value of x. ■

Theorem 13 *Greedy load balancing on identical servers has competitiveness at most* $\frac{2}{3}\sqrt{21} + 1 \approx 4.05505$.

Proof: Consider a load balancing instance on servers with latency function $f(x) = x$ and clients having at most two strategies. We will upper-bound the ratio of the cost of the greedy assignment to the optimal cost of instances with at most two strategies per client which satisfy a particular property. We say that server j is of type n_j/o_j meaning that it has n_j clients in the greedy assignment and o_j clients in the optimal assignment (equivalently, server j has in-degree n_j and out-degree o_j in the greedy graph). We first show that for any instance we can construct another instance that has at least the same competitiveness and, furthermore, satisfies the following 2-*neighborhood property*: the incoming edge of any server of type 1/1 originates from a server of type 0/1. Then, the idea behind the proof is to account for the contribution of servers of type 1/1 and 0/1 in the cost of the greedy assignment together.

Consider a server j of type 1/1. If a client c had server j as its only permissible server (this would correspond to a self-loop in the corresponding greedy graph), then we could construct a new instance by excluding server j and client c from the original one. In this way, we would obtain a new instance where both the optimal cost and the cost of the greedy assignment are decreased by 1 (and, hence, the competitiveness of the greedy assignment increases). So, let j' and j'' be the servers to which server j is connected in the greedy graph through edges corresponding to clients c_1 and c_2 that select servers j' and j in the optimal assignment and servers j and j'' in the greedy assignment, respectively. We can assume that for each server of type 1/1, its input client appears prior to its output client. Indeed, if c_2 appears prior to c_1, we can introduce a new server j^* and make client c_2 originate from j^* without changing its timing. In this way the server of type 1/1 is replaced by two servers of types 1/0 and 0/1, respectively, without changing the cost of the greedy and the optimal assignment.

Assume that server j' is of type $n_{j'}/o_{j'}$ with $n_{j'} > 0$ and, furthermore, that at least one of its input clients appears prior to c_1. Then, we could remove server j and replace clients c_1 and c_2 by a new client c' from server j' to server j'' having the same timing with c_2 to obtain another greedy graph in which both the optimal cost and the cost of the greedy assignment are decreased by 1. So, no input client of j' appears prior to c_1. Then, we can introduce a new server j^* and make client c_1 originate from j^* instead of j' (without changing its timing) to obtain a new greedy graph. If $o_{j'} > 1$ in the original instance, then the optimal cost of the new instance would decrease by at least 3 while the cost of the greedy assignment would remain the same. If $o_{j'} = 1$, we obtain a new instance in which the greedy assignment has the same competitiveness with the original instance and in which the input client of j originates from a server of type 0/1.

Now, denote by F the set of servers of type 1/1 and by S the set of servers of type 0/1 which are connected through an edge to a server in F in the greedy graph. Also, for each server j in F we denote by $S(j)$ the server of S from which the client destined for j originates.

By the greedy inequality, $\sum_j n_j^2 \le \sum_j o_j(2n_j+1)$, and since $\sum_j n_j = \sum_j o_j$, we have that

$$\begin{aligned}\sum_j n_j^2 &\le \sum_j (2n_jo_j+o_j) = \sum_j \left(2n_jo_j + \frac{27+7\sqrt{21}}{30}o_j - \frac{7\sqrt{21}-3}{30}n_j\right)\\ &= \sum_{j\notin S\cup F} g(n_j,o_j) + \sum_{j\in F}\left(g(n_{S(j)},o_{S(j)})+g(n_j,o_j)\right)\\ &\le \sum_{j\notin S\cup F} h(n_j,o_j) + \sum_{j\in F}\left(h(n_{S(j)},o_{S(j)})+h(n_j,o_j)\right)\\ &= \frac{9-\sqrt{21}}{10}\sum_j n_j^2 + \frac{9+\sqrt{21}}{6}\sum_j o_j^2\end{aligned}$$

where the first equality follows since $\sum_j n_j = \sum_j o_j$, the second equality follows by the definition of function g, the second inequality follows by Lemma 12, and the last equality follows by the definition of function h. Hence, we obtain that the competitiveness is

$$\frac{\sum_j n_j^2}{\sum_j o_j^2} \le \frac{2}{3}\sqrt{21}+1 \approx 4.05505.$$

■

We also present an almost matching lower bound.

Theorem 14 *For any $\epsilon > 0$, greedy load balancing on identical servers has competitiveness at least $4-\epsilon$.*

Proof: We assume that there are m servers $s_1, s_2, ..., s_m$, and k groups of clients $g_1, ..., g_k$, where group g_j has m/j^2 clients c_i^j, $1 \le i \le m/j^2$. We assume that m is such that all groups have integer size. Each client c_i^j has $s_1, s_2, ..., s_i$ as permissible servers. The clients appear in non-increasing order according to index i (ties are broken arbitrarily), i.e., c_m^1, c_{m-1}^1, ..., $c_{m/4+1}^1$, $c_{m/4}^2$, $c_{m/4}^1$, $c_{m/4-1}^2$, $c_{m/4-1}^1$, ..., $c_{m/9+1}^2$, $c_{m/9+1}^1$, $c_{m/9}^3$, $c_{m/9}^2$, $c_{m/9}^1$, ..., etc.

To upper bound the optimal cost *opt*, it suffices to consider the assignment where each client c_i^j chooses server s_i. We obtain that

$$\begin{aligned}\mathit{opt} &\le \sum_{i=1}^{k-1} i^2(|g_i|-|g_{i+1}|) + k^2|g_k| = m + m\sum_{i=1}^{k-1} i^2\left(\frac{1}{i^2}-\frac{1}{(i+1)^2}\right)\\ &= m\left(1+2\sum_{i=1}^{k-1}1/(i+1) - \sum_{i=1}^{k-1}1/(i+1)^2\right) \le m(2H_k+\zeta_1)\end{aligned}$$

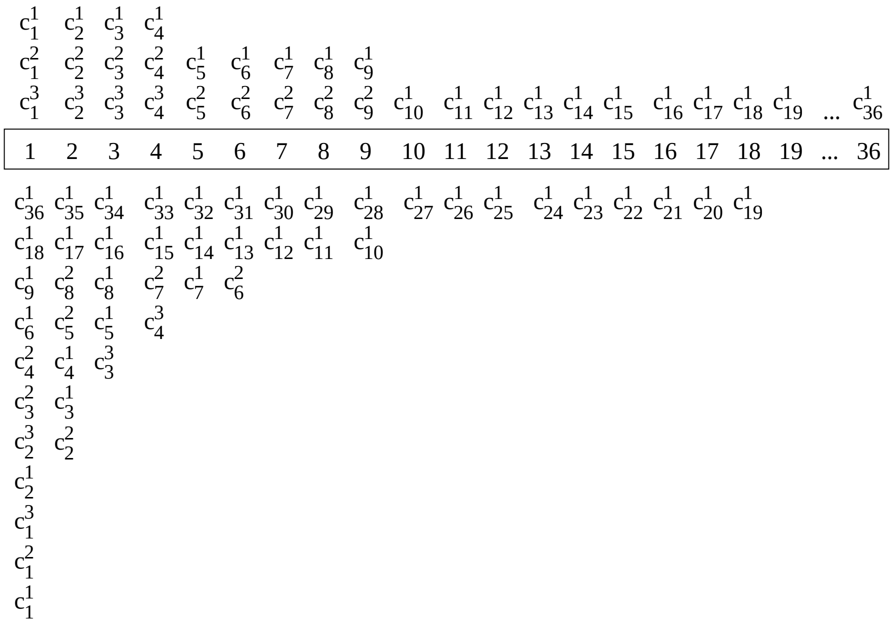
for some positive constant ζ_1, where H_k is the k-th Harmonic number.

A greedy assignment is obtained by making each client select the server with the smallest index among its permissible servers having the minimum number of clients. In the analysis we make use of sets of clients called *rows*. A client belongs to row row_i if, when it selects its server, it is the i-th client selecting that server. For example, clients $c_m^1, c_{m-1}^1, ...c_{m/2+1}^1$ select servers $s_1, ..., s_{m/2}$, respectively; each of them is the first client in its server, so they belong to row_1. Then, $c_{m/2}^1, ..., c_{m/4+1}^1$ select servers $s_1, ..., s_{m/4}$; they belong to row_2. We can verify that the set of servers selected by clients in row_{i+1} is subset of the set of servers selected by clients in row_i for $i = 1, ..., 2k-3$, that rows row_{2i-1} and row_{2i} contain clients of groups $g_1, ..., g_i$, and that $|row_{2i}| = \frac{m}{(i+1)^2}$ and $|row_{2i-1}| = \frac{m}{i(i+1)}$ for any $i = 1, ..., k-1$. So, for $i = 1, ..., 2k-3$, the number of servers receiving exactly i clients in the greedy assignment is $|row_i| - |row_{i+1}|$. We compute a lower bound on the cost gr of the greedy assignment by considering only the servers with at most $2k-4$ clients. We have that

$$\begin{aligned}
gr &\geq \sum_{i=1}^{k-2} \left((2i-1)^2(|row_{2i-1}| - |row_{2i}|) + (2i)^2(|row_{2i}| - |row_{2i+1}|)\right) \\
&= m\sum_{i=1}^{k-2} \left((2i-1)^2 \left(\frac{1}{i(i+1)} - \frac{1}{(i+1)^2}\right) + (2i)^2 \left(\frac{1}{(i+1)^2} - \frac{1}{(i+1)(i+2)}\right)\right) \\
&\geq m\sum_{i=1}^{k-2} \left(\frac{8}{i+1} - \frac{20}{(i+1)^2}\right) \geq m(8H_k - \zeta_2)
\end{aligned}$$

for some positive constant ζ_2. We conclude that for any $\epsilon > 0$ and sufficiently large k and m, the competitiveness of the greedy assignment is at least $4 - \epsilon$. ■

An example of the construction used in the proof of Theorem 14 is presented in Figure 5.



c^1_1 c^1_2 c^1_3 c^1_4
c^2_1 c^2_2 c^2_3 c^2_4 c^1_5 c^1_6 c^1_7 c^1_8 c^1_9
c^3_1 c^3_2 c^3_3 c^3_4 c^2_5 c^2_6 c^2_7 c^2_8 c^2_9 c^1_{10} c^1_{11} c^1_{12} c^1_{13} c^1_{14} c^1_{15} c^1_{16} c^1_{17} c^1_{18} c^1_{19} ... c^1_{36}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 ... 36
c^1_{36} c^1_{35} c^1_{34} c^1_{33} c^1_{32} c^1_{31} c^1_{30} c^1_{29} c^1_{28} c^1_{27} c^1_{26} c^1_{25} c^1_{24} c^1_{23} c^1_{22} c^1_{21} c^1_{20} c^1_{19}
c^1_{18} c^1_{17} c^1_{16} c^1_{15} c^1_{14} c^1_{13} c^1_{12} c^1_{11} c^1_{10}
c^1_9 c^2_8 c^1_8 c^2_7 c^1_7 c^2_6
c^1_6 c^2_5 c^1_5 c^3_4
c^2_4 c^1_4 c^3_3
c^2_3 c^1_3
c^3_2 c^2_2
c^1_2
c^3_1
c^2_1
c^1_1

Figure 5: An example with 36 servers and 3 groups of clients of size 36, 9 and 4 respectively. The clients appear in the following order: c^1_{36}, c^1_{35}, ..., c^1_{10}, c^2_9, c^1_9, c^2_8, c^1_8, c^2_7, c^1_7, c^2_6, c^1_6, c^2_5, c^1_5, c^3_4, c^2_4, c^1_4, c^3_3, c^2_3, c^1_3, c^3_2, c^2_2, c^1_2, c^3_1, c^2_1 and c^1_1. A valid assignment of cost 83 and the greedy assignment of cost 250 are presented at the top and bottom of the figure, respectively.

5 Weighted clients

In this section, we consider selfish and greedy load balancing with weighted clients and servers with linear latency functions. First, we show that the upper bound of $\frac{3+\sqrt{5}}{2} \approx 2.618$ on the price of anarchy of weighted congestion games [6, 10] is tight even for load balancing games.

Theorem 15 *For any $\epsilon > 0$, there exists a load balancing game with linear latency functions and with weighted clients whose price of anarchy is at least $\frac{3+\sqrt{5}}{2} - \epsilon$.*

Proof: Denote by $\phi = \frac{1+\sqrt{5}}{2}$ the golden ratio. We construct a game with $k+1$ servers so that server j has latency function $f_j(x) = \frac{1}{\phi^{2j}}x$ for $j = 0, ..., k-1$, and $f_k(x) = \frac{1}{\phi^{2(k-1)}}x$. For $j = 0, ..., k-1$ there is a client of weight ϕ^j having servers j and $j+1$ as its strategies.

Consider the assignment where client j selects server j ($j = 0, ..., k-1$). Since servers $k-1$ and k have the same latency functions, and server k is not used by any client, client $k-1$ has no incentive to deviate from server $k-1$ to server k. Now, consider client j with $0 \leq j \leq k-2$. By the definition of the latency functions $f_j(x)$ and the weight of client j, we have that client j experiences latency ϕ^{-j} at server j. If client j deviates to server $j+1$, then the load at server $j+1$ would be $\phi^{j+1} + \phi^j = \phi^{j+2}$ and the latency experienced by client j would be $f_{j+1}(\phi^{j+2}) = \phi^{-j}$ again. So, for $j = 0, \ldots, k-2$, client j has no incentive to deviate either. We conclude that the assignment is a Nash equilibrium. Clearly, its social cost is k.

In order to upper bound the optimal cost, it suffices to consider the assignment where client j selects server $j+1$ for $j = 0, ..., k-1$. Its cost is $\sum_{j=0}^{k-1} \phi^j f_{j+1}(\phi^j) = \sum_{j=0}^{k-2} \phi^j f_{j+1}(\phi^j) + \phi^{k-1} f_k(\phi^{k-1}) = (k-1)/\phi^2 + 1$. So, for any $\epsilon > 0$ and sufficiently large k, the price of anarchy is larger than $\phi^2 - \epsilon = \frac{3+\sqrt{5}}{2} - \epsilon$. ■

The proof of the above theorem makes use of different servers. In the case of identical servers, we have a slightly weaker lower bound.

Theorem 16 *For any $\epsilon > 0$, there exists a load balancing game with weighted clients and identical servers whose price of anarchy is at least $5/2 - \epsilon$.*

Proof: We construct a game graph G consisting of a complete ternary tree with $k+1$ levels with a binary tree of $k+1$ levels hung at each leaf so that the root of the binary tree coincides with the leaf of the ternary tree and an additional node hung at each leaf of the binary trees. So, graph G has $2k+2$ levels $0, ..., 2k+1$, with 3^i nodes at level i for $i = 0, ..., k$, $3^k 2^{i-k}$ nodes at levels $k+1, ..., 2k$ and 6^k nodes at level $2k+1$. The servers corresponding to the nodes have the same latency function $f(x) = x$ and the clients corresponding to edges connecting nodes of levels i and $i+1$ have weight $w_i = (2/3)^i$ for $i = 0, ..., k-1$, $w_i = (2/3)^{k-1}(1/2)^{i-k}$ for $i = k, ..., 2k-1$ and $w_{2k} = (1/3)^{k-1}$. The construction is depicted in Figure 6.

Consider the assignment where all clients select servers corresponding to the endpoint of their corresponding edge which is closer to the root of the game graph. We will show that this assignment is a Nash equilibrium. We have to consider five sets of clients corresponding to: edges connecting nodes of levels i and $i+1$ for $i = 0, \ldots, k-2$, edges connecting nodes of levels $k-1$ and k, edges connecting nodes of levels i and $i+1$ for $i = k, \ldots, 2k-2$, edges connecting nodes of levels $2k-1$ and $2k$ and edges connecting nodes of levels $2k$ and $2k+1$. Consider a client c_i corresponding to an edge connecting nodes v_i and v_{i+1} of levels i and $i+1$ with $0 \leq i \leq k-2$. The server corresponding to node v_i is used by three clients of weight

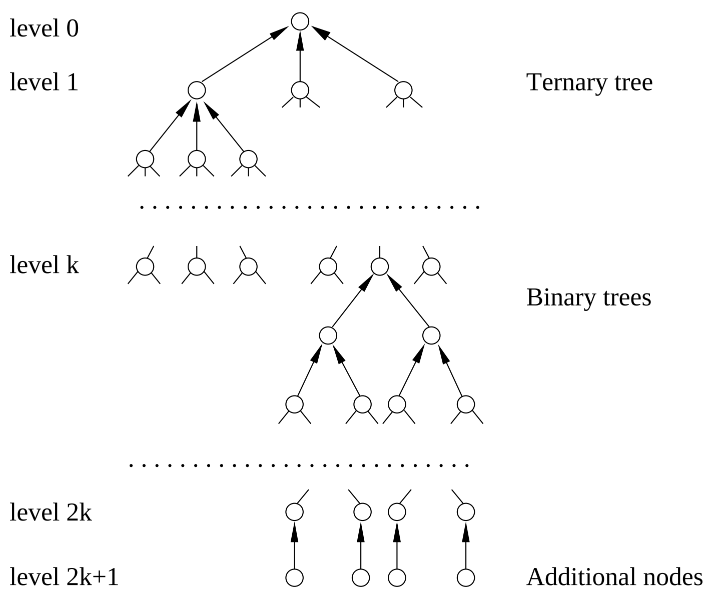


Figure 6: The construction in the proof of Theorem 16.

$(2/3)^i$ each. Hence, the latency experienced by client c_i is $3(2/3)^i$. The server corresponding to node v_{i+1} is used by three clients of weight $(2/3)^{i+1}$ each. If client c_i deviated to this server, its latency would be $(2/3)^i + 3(2/3)^{i+1} = 3(2/3)^i$ again. Hence, client c_i has no incentive to deviate. Now, consider a client c_{k-1} corresponding to an edge connecting nodes v_{k-1} and v_k of levels $k-1$ and k. The server corresponding to node v_{k-1} is used by three clients of weight $(2/3)^{k-1}$ each. Hence, the latency experienced by client c_{k-1} is $3(2/3)^{k-1}$. The server corresponding to node v_k is used by two clients of weight $(2/3)^{k-1}$ each. If client c_{k-1} deviated to this server, its latency would be $3(2/3)^{k-1}$ again. So, client c_{k-1} has no incentive to deviate. Furthermore, consider a client c_i corresponding to an edge connecting nodes v_i and v_{i+1} of levels i and $i+1$ with $k \leq i \leq 2k-2$. The server corresponding to node v_i is used by two clients of weight $(2/3)^{k-1}(1/2)^{i-k}$ each. Hence, the latency experienced by client c_i is $(2/3)^{k-1}(1/2)^{i-k-1}$. The server corresponding to node v_{i+1} is used by two clients of weight $(2/3)^{k-1}(1/2)^{i+1-k}$ each. If client c_i deviated to this server, its latency would also be $(2/3)^{k-1}(1/2)^{i-k} + (2/3)^{k-1}(1/2)^{i-k} = (2/3)^{k-1}(1/2)^{i-k-1}$. Again, client c_i has no incentive to deviate. We continue by considering a client c_{2k-1} corresponding to an edge connecting nodes v_{2k-1} and v_{2k} of levels $2k-1$ and $2k$. The server corresponding to node v_{2k-1} is used by two clients of weight $(1/3)^{k-1}$ each. Hence, the latency experienced by client c_{2k-1} is $2(1/3)^{k-1}$. The server corresponding to node v_{2k} is used by one client of weight $(1/3)^{k-1}$. If client c_{2k-1} deviated to this server, its latency would be $2(1/3)^{k-1}$ again. We conclude that client c_{2k-1} has no incentive to deviate. Finally, consider a client c_{2k} corresponding to an edge connecting nodes v_{2k} and v_{2k+1} of levels $2k$ and $2k+1$. The server corresponding to node v_{2k} is used only by c_{2k} and the latency is $(1/3)^{k-1}$, while the server corresponding to node v_{2k+1} is not used by any client. If client c_{2k} deviated to this server, its latency would also be $(1/3)^{k-1}$. Again, client c_{2k} has no incentive to deviate. We conclude that the assignment is a Nash equilibrium. Its cost is

$$\begin{aligned} cost &= \sum_{i=0}^{k-1} \left((3w_i)^2 3^i\right) + \sum_{i=k}^{2k-1} \left((2w_i)^2 3^k 2^{i-k}\right) + 6^k w_{2k}^2 \\ &= 9\sum_{i=0}^{k-1} (4/3)^i + 12\,(4/3)^{k-1} \sum_{i=k}^{2k-1} (1/2)^{i-k} + 6^k\,(1/9)^{k-1} \\ &= 45(4/3)^k - 9\,(2/3)^k - 27. \end{aligned}$$

To compute an upper bound for the cost of the optimal assignment, it suffices to consider the assignment where all clients select the servers corresponding to nodes which are further from the root. We obtain that the cost *opt* of the optimal assignment is

$$\begin{aligned} opt &\leq \sum_{i=0}^{k-1} \left(w_i^2 3^{i+1}\right) + \sum_{i=k}^{2k-1} \left(w_i^2 3^k 2^{i+1-k}\right) + 6^k w_{2k}^2 \\ &= 3\sum_{i=0}^{k-1} (4/3)^i + 6\,(4/3)^{k-1} \sum_{i=k}^{2k-1} (1/2)^{i-k} + 6^k\,(1/9)^{k-1} \\ &= 18\,(4/3)^k - 9. \end{aligned}$$

Hence, for any $\epsilon > 0$ and for sufficiently large k, the ratio of the social cost of the Nash equilibrium to the social cost of the optimal assignment is larger than $5/2 - \epsilon$. ■

For greedy load balancing, the next lower bound states that the upper bound of [5] for a more general version of the problem (namely, online scheduling on unrelated machines) is already tight for greedy load balancing of weighted clients on identical servers.

Theorem 17 *For any $\epsilon > 0$, there exists a load balancing instance with weighted clients and identical servers for which greedy has competitiveness at least $3 + 2\sqrt{2} - \epsilon$.*

Proof: We describe the recursive procedure *LB* which, on input a non-negative integer k, computes a greedy graph G_k as follows: If $k = 0$, G_k consists of two servers connected through a directed edge of weight 1 and timing information 1. Otherwise it executes *LB* on input $k-1$ twice to obtain two identical greedy graphs G_{k-1}^1 and G_{k-1}^2 and introduces a directed edge of weight $2^{k/2}$ and timing information $k+1$ connecting the server of maximum in-degree in G_{k-1}^1 to the server of maximum in-degree in G_{k-1}^2. Finally, the procedure *LB* outputs as G_k the union of G_{k-1}^1 and G_{k-1}^2 together with the new edge (see Figure 7).

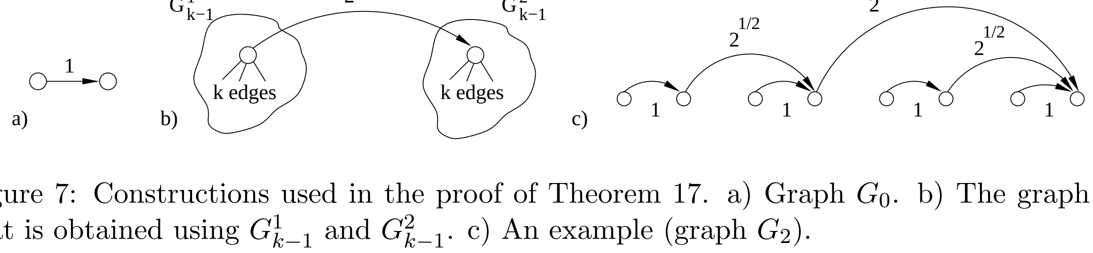


Figure 7: Constructions used in the proof of Theorem 17. a) Graph G_0. b) The graph G_k that is obtained using G_{k-1}^1 and G_{k-1}^2. c) An example (graph G_2).

G_k is a greedy graph, since at each time step the client that appears may choose between two servers having the same load and the edge corresponding to the client has a timing information that is greater than the timing information of all edges appearing in the two copies G_{k-1}^1 and G_{k-1}^2. In the greedy assignment, each client is assigned to the server corresponding to the node that is the head of the directed edge, while in order to bound the optimal cost it suffices to consider the assignment where each client selects the server corresponding to the tail of the directed edge.

Denote by $gr(i)$ the cost of greedy assignment and by $opt(i)$ the cost of the assignment that upperbounds the optimal assignment in G_i. Clearly, $opt(0) = gr(0) = 1$. We obtain the following two recursive relations:

$$\begin{aligned} opt\,(i) &= 2\,opt\,(i-1) + 2^i \\ gr\,(i) &= 2\,gr\,(i-1) + 2^i + 2^{1+i/2}\sum_{j=0}^{i-1} 2^{j/2}. \end{aligned}$$

The first term in both relations follows since graph G_i is obtained by two copies G_{i-1}^1 and G_{i-1}^2 of G_{i-1}. The second term of the first relation follows by observing that the servers having the maximum in-degree in G_{i-1}^1 and G_{i-1}^2 have out-degree 0. So, the total weight of the outgoing edges from the server of maximum in-degree in G_{i-1}^1 is increased from 0 to $2^{i/2}$. The last two terms in the second relation follow by observing that the maximum in-degree in the two copies of G_{i-1} is i and the edges incident to the two nodes of maximum in-degree have weights $1, \sqrt{2}, 2, ..., 2^{\frac{i-1}{2}}$. So, when we add the new edge to obtain G_i, the total weight of the incoming edges into the server of maximum in-degree in G_{i-1}^2 is increased from $\sum_{j=0}^{i-1} 2^{j/2}$ to $\sum_{j=0}^{i} 2^{j/2}$ and the last two terms in the second relation represent the increase in the (weighted) total latency.

Using these relations, we can inductively show that $opt(k) = 2^k\,(k+1)$ and $gr(k) = 2^k\left(\left(3+2\sqrt{2}\right)k - 5 - 4\sqrt{2}\right) + 2^{1+k/2}\left(3+2\sqrt{2}\right)$. Hence, for any $\epsilon > 0$ and for sufficiently large k, the ratio of the cost of the greedy assignment to the optimal cost of G_k is at least $3 + 2\sqrt{2} - \epsilon$. ■

6 Open problems

Although most of our results are tight, there are still several interesting questions about load balancing, especially when the latency functions are linear. An intriguing open problem is to compute tight bounds for the price of stability of weighted load balancing games. It is also interesting to close the gap between the lower bound of 5/2 on the price of anarchy for selfish load balancing games of weighted clients on identical servers and the upper bound of $\frac{3+\sqrt{5}}{2}$ which has been proved for congestion games [6]. We believe that our lower bound is tight. We have considered pure Nash equilibria of load balancing games. Some of our results hold or can be extended to hold for mixed and correlated equilibria [11] as well. There is also a small gap between 4 and 4.05505 for the competitiveness of greedy load balancing on identical servers. We believe that it can be further narrowed by extending our upper bound technique. Finally, we remark that by slightly modifying the arguments in the proofs of Theorems 14 and 17, it follows that the lower bounds hold for any deterministic online load balancing algorithm; hence, the greedy algorithm is (almost) optimal in these particular cases.

Investigating whether the use of randomization can lead to better competitiveness deserves further attention. Recent results in this direction are presented in [8].

References

[1] S. Aland, D. Dumrauf, M. Gairing, B. Monien, and F. Schoppmann. Exact price of anarchy for polynomial congestion games. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS '06)*, LNCS 3884, Springer, pp. 218-229, 2006.

[2] 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)*, pp. 493-500, 1997.

[3] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4): 1602-1623, 2008.

[4] A. Avidor, Y. Azar and J. Sgall. Ancient and new algorithms for load balancing in the L_p norm. *Algorithmica*, 29(3): 422-441, 2001.

[5] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the L_p norm. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, pp. 383-391, 1995.

[6] B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, pp. 57-66, 2005.

[7] Y. Azar and A. Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, pp. 331-337, 2005.

[8] I. Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*, pp. 972-981, 2008.

[9] 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.

[10] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, pp. 67-73, 2005.

[11] G. Christodoulou and E. Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA '05)*, LNCS 3669, Springer, pp. 59-70, 2005.

[12] G. Christodoulou, V. Mirrokni, and A. Sidiropoulos. Convergence and approximation in potential games. In *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS '06)*, pp. 349-260, 2006.

[13] R. A. Cody and E. G. Coffman. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *Journal of the ACM*, 23(1): 103-115, 1976.

[14] R. Cominetti, J. R. Correa, N. E. Stier Moses. Network games with atomic players. In *Proceedings of the 33rd International Colloquium on Automata, Languages, and Programming (ICALP '06)*, LNCS 4168, pp. 525-536, 2006.

[15] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1), 2007.

[16] A. Fabrikant, C. Papadimitriou and K. Talwar. On the complexity of pure equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC '04)*, pp. 604-612, 2004.

[17] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. *Theoretical Computer Science*, 410(36): 3305-3326, 2009.

[18] D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348(2-3): 226-239, 2005.

[19] 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)*, pp. 613-622, 2004.

[20] M. Gairing, T. Lücking, M. Mavronicolas and B. Monien. The price of anarchy for polynomial social cost. *Theoretical Computer Science*, 369(1-3): 116-135, 2006.

[21] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. Nash equilibria in discrete routing games with convex latency functions. *Journal of Computer and System Sciences*, 74(7): 1199-1225, 2008.

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

[23] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS '99)*, LNCS 1563, Springer, pp. 404-413, 1999.

[24] T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. A new model for selfish routing. *Theoretical Computer Science*, 406(3): 187-206, 2008.

[25] M. Mavronicolas and P. Spirakis. The price of selfish routing. *Algorithmica*, 48(1): 91-126, 2007.

[26] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14: 124-143, 1996.

[27] C. Papadimitriou. Algorithms, games and the internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC '01)*, pp. 749-753, 2001.

[28] S. Phillips and J. Westbrook. Online load balancing and network flow. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC '93)*, pp. 402-411, 1993.

[29] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2: 65-67, 1973.

[30] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2): 236-259, 2002.

[31] T. Roughgarden and E. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior*, 47(2): 389-403, 2004.

[32] D. Shmoys, J. Wein and D. Williamson. Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24(6): 1313-1331, 1995.

[33] S. Suri, C. Tóth and Y. Zhou. Selfish load balancing and atomic congestion games. *Algorithmica*, 47(1): 79-96, 2007.