Ancient and new algorithms for load balancing in the ℓ_p norm *

Adi Avidor[†] Yossi Azar[‡] Jiří Sgall[§]

April 14, 1999

Abstract

We consider the on-line load balancing problem where there are m identical machines (servers) and a sequence of jobs. The jobs arrive one by one and should be assigned to one of the machines in an online fashion. The goal is to minimize the sum (over all machines) of the squares of the loads, instead of the traditional maximum load.

We show that for the sum of the squares the greedy algorithm performs within 4/3 of the optimum, and no on-line algorithm achieves a better competitive ratio. Interestingly, we show that the performance of greedy is not monotone in the number of machines. More specifically, the competitive ratio is 4/3 for any number of machines divisible by 3 but strictly less than 4/3 in all the other cases (although it approaches 4/3 for large number of machines). To prove that greedy is optimal, we show a lower bound of 4/3 for any algorithm for 3 machines. Surprisingly, we provide a new on-line algorithm that performs within $4/3 - \delta$ of the optimum, for some fixed $\delta > 0$, for any sufficiently large number of machines. This implies that the asymptotic competitive ratio of our new algorithm is strictly better than the competitive ratio of any possible on-line algorithm. Such phenomena is not known to occur for the classic maximum load problem.

Minimizing the sum of the squares is equivalent to minimizing the load vector with respect to the ℓ_2 norm. We extend our techniques and analyze the exact competitive ratio of greedy with respect to the ℓ_p norm. This ratio turns out to be $2-\Theta((\ln p)/p)$. We show that greedy is optimal for two machines but design an algorithm whose asymptotic competitive ratio is better than the ratio of greedy.

1 Introduction

Consider a set of jobs that are created in an on-line fashion and should be assigned to disks. Each job has a weight which is the frequency access to the disk. Define the load on a disk to be the sum of the weights of jobs assigned to it. If we would like to minimize the

*A Preliminary version of this paper appears in the proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1998, pp. 426-435.

[†]Department of Computer Science, Tel Aviv University.

[‡]Department of Computer Science, Tel-Aviv University. E-Mail: azar@math.tau.ac.il. Supported in part by the Israel Science Foundation and by the US-Israel Binational Science Foundation (BSF).

[§]E-mail sgall@math.cas.cz. Mathematical Institute, AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic; partially supported by grants A1019602 and A1019901 of GA AV ČR and postdoctoral grant 201/97/P038 of GA ČR.

maximum delay of an access request of a job to the disk then it is equivalent to minimizing the maximum load on a disk. However, in many cases the more appropriate goal would be to minimize the average delay of all access requests [6]. That is equivalent to minimizing the sum of the squares of the loads, since the delay of an access request is proportional to the load on the disk. That motivates us to define the following machine load balancing problem.

We are given m parallel identical machines and a number of independent jobs (tasks) arriving one by one at arbitrary times. Each job has an associated weight and should be assigned immediately to exactly one of the machines based only on the previous jobs without any knowledge on the future jobs. The **load** of a machine is the sum of the weights of the jobs assigned to it. The **cost** of an assignment for an input sequence of jobs is the sum of the squares of the loads of the machines (after all jobs are assigned) or equivalently the square of the Euclidean norm of the load vector. The goal of an assignment algorithm is to assign all the jobs while minimizing the cost.

We measure the performance of an on-line algorithm by its **competitive ratio**. An on-line algorithm is c-competitive if for each input the cost of the assignment produced by the algorithm is at most c time larger than the cost of the optimal assignment.

We show that for minimizing the sum of the squares of loads the greedy algorithm is $\frac{4}{3}$-competitive over any number of machines. We also show that no on-line algorithm can achieve a better ratio than $\frac{4}{3}$ for all m which implies that greedy may be viewed as an optimal algorithm. Interestingly, the competitive ratio of the greedy algorithm is not monotone in the number of machines. It is exactly $\frac{4}{3}$ for $m \equiv 0 \pmod 3$ but strictly smaller than $\frac{4}{3}$ for $m \not\equiv 0 \pmod 3$. However, the **asymptotic competitive ratio**, which is the limit of the competitive ratio for $m \to \infty$, is $\frac{4}{3}$. More precisely, for any $\delta > 0$ there is some M such that for all $m \geq M$ the competitive ratio of greedy is at least $\frac{4}{3} - \delta$. The reason that the greedy algorithm is optimal if we allow any number of machines is that we prove a lower bound of $\frac{4}{3}$ on any on-line algorithm for 3 machines. The greedy algorithm is also optimal for 2 and 4 machines, in both cases the competitive ratio is $(3+\sqrt{5})/4 \approx 1.3090$.

Surprisingly, we provide a new on-line algorithm with competitive ratio $\frac{4}{3} - \delta$, for some fixed $\delta > 0$, for any sufficiently large number of machines. This implies that the asymptotic competitive ratio of our new algorithm is strictly better than the competitive ratio of any possible on-line algorithm and strictly better than the asymptotic competitive ratio of greedy. Such phenomena is not known to occur for the classic makespan problem. In fact it is conjectured in [7] that the competitive ratio for the makespan is monotone increasing with the number of the machines. We are not aware of another natural on-line problem whose competitive ratio is different than the asymptotic competitive ratio (in particular it would have to be non-monotone in m).

We also consider the general ℓ_p norm (for any $p > 1$). Note that minimizing the sum of the squares is equivalent to minimizing the ℓ_2 norm of the load vector. We can also define the cost to be the ℓ_p norm of the machines' load vector for any $p > 1$. In particular, minimizing the ℓ_∞ norm precisely means minimizing the maximum load. By using the triangle inequality, it is not hard to show that for any ℓ_p norm the worst case performance of the greedy algorithm is at most 2. We determine the exact worst case performance of the greedy algorithm over an arbitrary number of machines. We show that this performance is $2 - \Theta(\frac{\ln p}{p})$ (for $p \to \infty$). Then we present a lower bound of $\frac{3}{2} - \Theta(\frac{1}{p})$ for any on-line assignment algorithm for any fixed number of machines. We show that for any $p > 1$ the greedy algorithm is optimal for 2 machines. In contrast, we design an algorithm whose asymptotic competitive ratio is strictly better than the asymptotic competitive ratio of greedy.

The case $p = \infty$ (i.e., ℓ_∞) is the classic ancient problem of scheduling jobs on identical machines minimizing the makespan (or maximum load). Graham [11] showed that the greedy load balancing algorithm is $2 - \frac{1}{m}$ competitive in this case. The greedy algorithm is optimal only for $m \leq 3$, for any $m > 3$ better algorithms exist [9, 7]. Bartal et al. [5] were the first to show an algorithm whose competitive ratio is below $2 - \delta$ for some constant $\delta > 0$ and arbitrary m. Very recently, Albers [1] designed a 1.923 competitive algorithm and showed a lower bound of 1.852.

Chandra and Wong [6] were the first to consider the problem of minimizing the sum of the squares of the loads of the machines. Cody and Coffman [8], in their study of placing a set of records on a sectored drum to minimize the average latency, considered essentially the same minimization problem. In [6] it is shown that if the jobs arrive in non-increasing weights order then the greedy algorithm yields a schedule whose cost is within $\frac{25}{24}$ of the optimal cost. This result was slightly improved by Leung and Wei [12]. Chandra and Wong [6] also considered the general ℓ_p norm (for any $p > 1$) and showed that the greedy algorithm on the sorted items achieves a constant performance bound. The constant depends on p and grows to $\frac{3}{2}$ when p grows to ∞.

Off-line scheduling and load balancing with respect to the ℓ_p norm has been considered in [2]. The off-line minimization problem is known to be NP-hard in the strong sense [10]. Alon et al. [2] provided a polynomial approximation scheme for scheduling jobs with respect to the ℓ_p norm for any $p > 1$. An example in which the optimal assignment for the sum of the squares is different than the optimal assignment in the ℓ_∞ norm is also given in [2].

Results on load balancing for unrelated machines with respect to the ℓ_∞ norm and ℓ_2 (and ℓ_p) norm appear in [3, 4]. The competitive ratio for unrelated machines is much worse ($\Theta(p)$) than the trivial bounds for identical machines and the techniques used are totally different.

2 Definitions and preliminaries

In the **load balancing problem** we are given m identical machines (servers) and a finite sequence of jobs (tasks). Each job j has a weight $w_j \geq 0$. A **schedule** S is an assignment which assigns each job j to a single machine i, $1 \leq i \leq m$. For every schedule S, the **load** of machine i, denoted $L_i(S)$, is the sum of weights of all jobs assigned to i in S. The **vector of loads** is $L(S) = (L_1(S), \ldots, L_m(S))$. An **assignment algorithm** is an algorithm which for an input sequence σ produces a schedule which assigns every job in σ to one of the m machines. An **on-line assignment algorithm** must assign a job j at its arrival to a machine based only on the previous jobs and their assignments; the decision is made without any knowledge about future job arrivals. Our measure of cost is the ℓ_p norm. Hence the **cost** of a schedule S is defined as

$$\|L(S)\|_p = \left(\sum_{i=1}^{m} (L_i(S))^p \right)^{\frac{1}{p}}.$$

Note that we can reorder the machines with no change in the cost. The **optimal cost**, denoted $OPT(S)$, is the minimal cost over all schedules S' which assigns the same jobs as S; this schedule can be computed by an off-line algorithm which knows all the jobs in advance.

We measure the performance of our algorithms by the **competitive ratio**. For a fixed $p > 1$, the **competitive ratio of a schedule** S is defined as $C(S) = \|L(S)\|_p / OPT(S)$ (putting $0/0 = 1$ to handle the empty schedule). Let A be an on-line assignment algorithm. The **competitive ratio of A over a fixed number $m \geq 1$ of machines** is defined as

$$C_{A,m} = \sup\{C(S) \mid S \text{ is a schedule produced by } A \text{ on } m \text{ machines}\}.$$

The **competitive ratio of A over an arbitrary number of machines** is defined as

$$C_A = \sup\{C_{A,m} \mid m \geq 1\}.$$

The previous definitions cover also the case when we measure the sum of the squares of loads, since then the cost is $(\|L(S)\|_2)^2$. Consequently, the competitive ratios w.r.t. the sum of the squares of loads are equal to $C^2(S)$, $C^2_{A,m}$ and C^2_A w.r.t. the ℓ_2 norm.

Now we define the notion of a shape of a schedule, which is an abstraction of a schedule where for every machine, all jobs assigned to it except for one are replaced by very small jobs with the same total load. In general it may be impossible to produce such a schedule by the same algorithm as the original one. Nevertheless, the concept of a shape is very useful for proving upper bounds on the competitive ratio, since the optimal assignment may improve (by partitioning the jobs) while the cost of the assignment does not change. Hence a shape is a pessimistic estimate of a schedule. A shape characterizes each machine by two numbers, a_i is the total load of the small jobs, and u_i is (a lower bound on) the weight of one large job.

Formally a **shape** is a pair $R = (a, u)$, where a and u are vectors of m nonnegative reals. The **vector of loads** of a shape is defined as $L(R) = a + u$. The shape $R = (a, u)$ is a **shape of a schedule** S if $L(R) = L(S)$ and for every $i \leq m$ with $u_i > 0$ there exists a job with weight $w_j \geq u_i$ assigned to the machine i in S. The **optimal cost of a shape** R is the infimum of the optimal costs of all schedules S with the shape R, formally $OPT(R) = \inf\{OPT(S) \mid R \text{ is a shape of } S\}$. As we shall see, the infimum can be replaced by a minimum. The **competitive ratio** of a shape R is $C(R) = \|L(R)\|_p / OPT(R)$.

It is possible to compute the optimal cost of the shape $R = (a, u)$ explicitly. It is the cost of a schedule in which some big jobs are scheduled each on a separate machine and the rest of the jobs are balanced evenly on the rest of the machines. Let the machines be ordered so that u_i are nondecreasing. For $1 \leq l \leq m$ let $h_l = (\sum_{i=1}^m a_i + \sum_{i=1}^l u_i)/l$. Let k be the largest l such that $h_l \geq u_l$ (k is always defined, since $h_1 \geq u_1$). We define the **height** of the shape to be $h(R) = h_k$.

It is easy to see that a good candidate for an optimal schedule for the shape R is to put on each machine one job of size exactly u_i and divide a_i into a few jobs so that they can be balanced exactly on the k machines; then the load vector is $(h_k, \ldots, h_k, u_{k+1}, \ldots, u_m)$. See the Figures 1 and 2 for examples where $a_i = 1$ for all i. Next, we show that this really is the optimal schedule.

Lemma 2.1 *Let $h = h(R)$ and let k be such that $h = h_k$ in the previous definition. Then $OPT(R) = \|(h, \ldots, h, u_{k+1}, \ldots, u_m)\|_p$.*

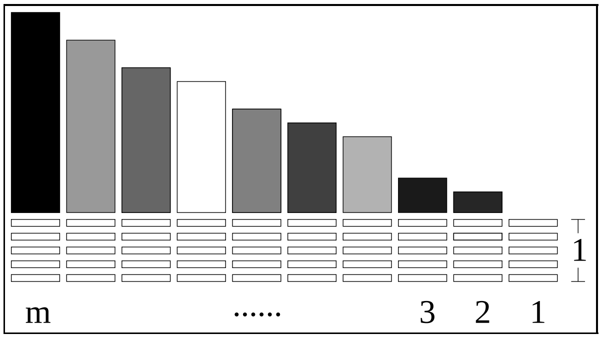

Figure 1: A shape R.

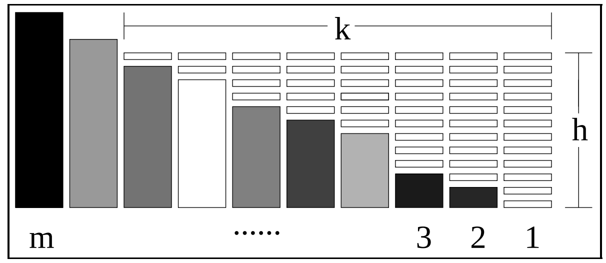

Figure 2: Optimal assignment of R

Proof: We have seen above that there exists a schedule which achieves this value. It remains to prove that for any schedule S with shape R, the cost $OPT(S)$ is at least the bound in the statement of the lemma.

From the definition of $h(R)$ it follows that $h < u_i$ for every $i > k$ (otherwise we would have chosen larger k). For $i > k$, let j_i be a job assigned to machine i in S with weight at least u_i (it exists, since $u_i > 0$).

Let S' be the optimal schedule for the jobs in S. First, S' has a machine with load at most h: there are at least k machines on which no job j_i, $i > k$, is scheduled, and their total load is at most kh. Second, if S' is optimal, then for any $i > k$, no other job is assigned to the same machine as the job j_i: Assume that the job j_i is scheduled on a machine with the load $b > 0$ of other jobs. We know that there is a machine with load $c \leq h < u_i \leq w_{j_i}$. However, if we replace the two machines with loads c and $b + w_{j_i}$ by two machines with load $b + c$ and w_{j_i}, the total cost decreases due to the convexity of the function x^p. Consequently, after a renumbering of the machines, the vector of loads $L = L(S')$ satisfies $L_i \geq u_i$ for each $i > k$ and $\sum_{i=1}^{m} L_i = hk + \sum_{i=k+1}^{m} u_i$. Using convexity again, the cost of any such schedule is at least $\|(h, \ldots, h, u_{k+1}, \ldots, u_m)\|_p$. ■

3 The greedy algorithm

In this section we determine the competitive ratio of the greedy algorithm defined below.

> **Algorithm Greedy**: Upon arrival of a job j assign it to the machine with the current minimum load (ties are broken arbitrarily).

To bound the performance of Greedy, we show that each schedule can be replaced by a very special shape so that the competitive ratio does not decrease. Computing the competitive ratio is then shown to be equivalent to computing a maximum of a certain function over the reals. A shape $R = (a, u)$ is called **flat** if all the components of a are the same. Abusing notation, we then denote by a both the value of the component and the vector of m such components. (For an example of a flat shape, see Figure 1.)

Lemma 3.1 *Let S be a schedule obtained by Greedy. Then there exists a flat shape R which is a shape of S.*

Proof: Let $L = L(S)$ be the vector of loads of S; w.l.o.g. assume that L_1 is the smallest component of L. We claim that the flat shape (a, u), where $a = L_1$ and $u_i = L_i - a$, is a shape of S. Clearly the loads are the same. Consider a machine with $u_i > 0$. Let j be the last job assigned to the machine i. At the time of its assignment, the load of the machine i must have been at most a, as otherwise Greedy would have scheduled j on the machine 1. Hence $w_j \geq L_i - a = u_i$. ■

Lemma 3.2 *Let $R = (a, u)$ be a flat shape. Then there exists a flat shape $R' = (a', u')$ such that $C(R) \leq C(R')$ and for any i, $u'_i = 0$ or $u'_i \geq h(R')$.*

Proof: Assume that $0 < u_1 < h = h(R)$. We claim that there exists a flat shape R' such that $\|L(R')\|_p \geq \|L(R)\|_p$, $OPT(R') = OPT(R)$, $h(R') = h$, u'_1 is either 0 or h, and the other components of the vector u' remain unchanged. Applying this claim inductively finishes the proof of the lemma.

To prove the claim, we define two candidate shapes and prove that one of them satisfies all the conditions. Let $R' = (a', u')$ and $R'' = (a'', u'')$, where $u' = (0, u_2, \ldots, u_m)$, $a' = a + u_1/m$, $u'' = (h, u_2, \ldots, u_m)$, and $a'' = a - (h - u_1)/m$. It is easy to verify that $a'' \geq 0$, $OPT(R') = OPT(R'') = OPT(R)$, and $h(R') = h(R'') = h$. Moreover,

$$L(R) = \frac{h - u_1}{h} L(R') + \frac{u_1}{h} L(R''),$$

i.e., the old vector of loads is a weighted average of the two new vectors of the loads. From the convexity of the function x^p it follows that

$$(\|L(R)\|_p)^p \leq \frac{h - u_1}{h} (\|L(R')\|_p)^p + \frac{u_1}{h} (\|L(R'')\|_p)^p.$$

Hence either $\|L(R)\|_p \leq \|L(R')\|_p$ or $\|L(R)\|_p \leq \|L(R'')\|_p$, and the claim is proved. ■

Lemma 3.3 *Let $k \leq m$ and let $R = (a, u)$ be a flat shape such that $u_i \geq h(R)$ for $i = 1, \ldots, k$. Then there exists x such that for $R' = (a, (x, \ldots, x, u_{k+1}, \ldots, u_m))$, it holds that $C(R) \leq C(R')$, $OPT(R') = OPT(R)$, $h(R') = h(R)$, and $x \geq h(R')$.*

Proof: Let x be the pth power mean of u_i, $i \leq k$, i.e., $x = (\sum_{i=1}^{k} u_i^p / k)^{1/p}$. Clearly x is at least the minimum of $u_1, \ldots, u_k$, hence the optimal schedule changes only in the load of the machines with these single jobs and $x \geq h(R) = h(R')$. The contribution of the k big jobs to $(OPT(R))^p$ is $\sum_{i=1}^{k} u_i^p$, and their contribution to $(OPT(R'))^p$ is kx^p. These contributions are the same and hence the optimal cost does not change. It remains to prove that $\|L(R')\|_p \geq \|L(R)\|_p$. Comparing the contribution of the first k machines, we need to prove that

$$\sum_{i=1}^{k} (a + u_i)^p \leq k(a + x)^p.$$

We apply Minkowski inequality (the triangle inequality for the ℓ_p norm) to the vectors $(a, \ldots, a)$ and $(u_1, \ldots, u_k)$ and obtain

$$\left(\sum_{i=1}^{k} (a + u_i)^p\right)^{\frac{1}{p}} \leq k^{\frac{1}{p}} a + \left(\sum_{i=1}^{k} u_i^p\right)^{\frac{1}{p}} = k^{\frac{1}{p}} (a + x).$$

Raising to pth power finishes the proof. ■

The above lemmas imply that in order to find the competitive ratio of the greedy algorithm we need to solve a restricted optimization problem. Define the function

$$f_p(x, \beta) \quad = \quad \frac{(1-\beta)(1+x)^p + \beta}{(1-\beta)x^p + \beta^{1-p}}$$

Theorem 3.4 *The competitive ratios of the greedy algorithm are*

$$\begin{aligned} C_{Greedy,m} &= \sup\{f_p(x, M/m)^{1/p} \mid x > 0,\ M = 1, \ldots, m\}, \\ C_{Greedy} &= \sup\{f_p(x, \beta)^{1/p} \mid x > 0,\ 0 < \beta \le 1\}. \end{aligned}$$

Furthermore, in both cases the supremum remains the same if it is restricted to $x > 1$ instead of $x > 0$.

Proof: Let $u(x, M, m)$ be a vector with $m - M$ components with value x and M zeros, and let $R = (1, u(x, M, m))$ be the corresponding flat shape. The function f_p is defined so that $f_p(x, M/m)^{1/p} = C(R)$ if $x \ge h(R)$.

By the above lemmas for every schedule S generated by Greedy there exists x and M such that the flat shape $R = (1, u(x, M, m))$ satisfies $C(S) \le C(R)$: First apply Lemma 3.1 to obtain a flat shape of S; by Lemma 2.1 the competitive ratio can only increase. Next apply Lemma 3.2 and Lemma 3.3 to obtain a flat shape (a, u) with u of the desired form. Last, normalize the shape so that $a = 1$, i.e., put $R = (1, u/a)$ (if $a = 0$, the greedy schedule is optimal). From the construction it follows that $M > 0$ and $x > h = m/M > 1$. Hence $C(R) = f_p(x, M/m)^{1/p}$ and this is at most the supremum of $f_p(x, M/m)^{1/p}$ over $x > 1$.

The supremum over $x > 0$ is at least the supremum over $x > 1$. Thus, for a fixed m, it is now sufficient, given $x > 0$ and $M > 0$, to find a sequence of jobs σ with the cost ratio at least $f_p(x, M/m)^{1/p}$. Consider the sequence of mM jobs with weight $1/M$ followed by $m - M$ jobs of weight x. Greedy produces a schedule with $m - M$ machines with load $1 + x$ and M machines with load 1, while it is possible to schedule $m - M$ machines with load x and M machines with load m/M (this is a valid schedule but it is not necessarily optimal if $x < h$); this leads to the desired ratio. For unrestricted m, given $x > 0$ and $\beta > 0$, fix a sequence of rational numbers $0 < M_i/m_i \le 1$ converging to β. The result now follows, since f_p is continuous. ■

In Theorem A.1 in the appendix we prove that the supremum of $f_p(x, \beta)$, $x > 0$, $0 < \beta \le 1$, is achieved at a unique point, moreover we give a method to compute the supremum and hence the competitive ratio. Here we state the results for the most interesting case of sum of squares, i.e., evaluating C^2_{Greedy} for $p = 2$.

Theorem 3.5 *For $p = 2$ the performance of Greedy is:*

$$\begin{aligned} C^2_{Greedy} &= \frac{4}{3}, \\ C^2_{Greedy,m} &= \frac{4}{3} \qquad \textit{for } m \equiv 0 \pmod 3, \\ C^2_{Greedy,m} &< \frac{4}{3} \qquad \textit{for } m \not\equiv 0 \pmod 3, \\ C^2_{Greedy,m} &= \frac{3+\sqrt{5}}{4} \approx 1.3090 \qquad \textit{for } m = 2, 4. \end{aligned}$$

Proof: According to Theorem A.1, the supremum of f_2 is achieved at $x = 3$, $\beta = 2/3$, and $C^2_{Greedy} = f_2(x, \beta) = 4/3$. The claim for m divisible by 3 is true, since the supremum of f_2 is achieved for $M = 2m/3$. The claim for m not divisible by 3 is true, since the supremum of f_p is unique, and $M/m \neq 2/3$ if m is not divisible by 3.

The last claim is obtained by maximizing the function $f_2(x, 1/2)$ for $m = 2$ and additionally $f_2(x, 1/4)$ and $f_2(x, 3/4)$ for $m = 4$. The calculus leads to simple quadratic equations, which give the optimal solution $x = 1 + \sqrt{5}$ and $M = m/2$. Details are omitted. ∎

For general $p > 1$ we state the following theorem, which is proved in the appendix B.

Theorem 3.6

$$C_{Greedy} = 2 - \Theta\left(\frac{\ln p}{p}\right).$$

We also give numerical calculations of C_{Greedy} for various p's in appendix C. The experimental results show that C_{Greedy} is monotone in p, in fact, that can be easily (but technically) proved for $p \geq p_0$.

4 Lower Bounds

In this section we prove that for $p = 2$ Greedy is optimal for fixed $m = 2, 3, 4$, and hence also for arbitrary m (since the worst performance of Greedy is achieved for $m = 3$). In Theorem 4.2 we give weaker lower bounds for $m > 4$, and then some bounds for general p.

Theorem 4.1 *For any on-line assignment algorithm A, $C^2_A \geq C^2_{A,3} \geq 4/3$ and $C^2_{A,2}, C^2_{A,4} \geq (3 + \sqrt{5})/4 \approx 1.3090$.*

Proof: Consider the sequence $(1, 1, 1, 3, 3, 3, 12)$ for 3 machines. First the 3 jobs of weight 1 arrive. If the algorithm A assigns two or more jobs with weight 1 on the same machine, it does not get any other job. Its cost is at least 5, the optimal cost is 3, and we are done. Otherwise, A assigns on every machine one job with weight 1. Now the next 3 jobs of weight 3 arrive. If A assigns two or more of these jobs on the same machine, is does not get any other job. Its cost is at least $(1 + 3 + 3)^2 + (1 + 3)^2 + 1^2$, whereas the optimum cost is $3 \cdot 4^2$ which again yields a ratio greater than $4/3$. Otherwise all machines have load 4 before the last job of weight 12 arrives. The cost of A on the full sequence is $(12 + 4)^2 + 2 \cdot 4^2$, whereas the optimum cost is $12^2 + (3 + 3)^2 + (3 + 1 + 1 + 1)^2$, which yields a ratio of $4/3$.

Consider the sequences $(1, 1, 1 + \sqrt{5})$ for $m = 2$ and $(1, 1, 1, 1, 1 + \sqrt{5}, 1 + \sqrt{5})$ for $m = 4$. If the algorithm assigns two jobs of weight 1 to the same machine, the ratio is at least $3/2 > (3 + \sqrt{5})/4$. Otherwise together with the large job(s) we get a ratio of $(3 + \sqrt{5})/4$. ∎

Theorem 4.2 *For any number of machines $m \geq 2$ and any on-line assignment algorithm A, $C^2_{A,m} \geq \sqrt{5} - 1 \approx 1.2361$.*

Proof: Denote $\alpha = \sqrt{5}/2 - 1 \approx 0.118$. We assume that m is even. The proof for m which is odd is similar and omitted. Consider the sequence of m jobs of weight 1 followed by $m/2$ jobs of weight $1+\sqrt{5}$. At the beginning m jobs of size 1 arrive. Let l be the number of empty machines after the on-line algorithm assigns the m jobs of size 1. An example of the best such assignment is shown in Figure 3.

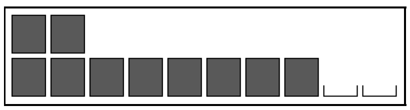

Figure 3: Best on-line assignment of the first m jobs with l empty machines

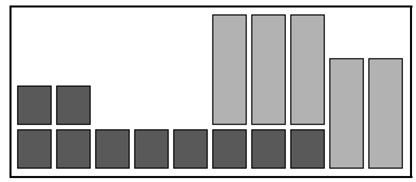

Figure 4: Best on-line assignment of all the jobs with l empty machines

First we assume $\alpha m \le l \le \frac{1}{2}m$. In this case the cost of the on-line algorithm is at least $l \cdot 2^2 + (m-2l)\cdot 1^2$ whereas the cost of optimum is $m \cdot 1^2$. This yields a ratio of at least $1+2l/m \ge 1+2\alpha = \sqrt{5}-1$ which completes the proof of this case. Since any assignment for $l \ge m/2$ results in a cost greater than the assignment for $l = m/2$ we may assume $l < \alpha m$. Now the next $m/2$ jobs of weight $\sqrt{5}+1$ arrive. An example of the best on-line assignment for these jobs is shown in Figure 4. The competitive ratio of the algorithm is at least

$$\frac{4l + (\frac{m}{2}-l) + (\frac{m}{2}-l)(2+\sqrt{5})^2 + l(\sqrt{5}+1)^2}{\frac{m}{2}\cdot(\sqrt{5}+1)^2 + \frac{m}{2}\cdot(1+1)^2} .$$

Since $l < \alpha m$, the competitive ratio is at least

$$((1-\alpha)\cdot 2\sqrt{5}+5)/(5+\sqrt{5}) = \sqrt{5}-1 .$$

■

Now we turn to the case of general p, starting with two machines.

Theorem 4.3 *The competitive ratio of Greedy for two machines is*

$$C_{Greedy,2} = \sup_{x \ge 0} \left(\frac{1+(1+x)^p}{2^p + x^p} \right)^{\frac{1}{p}},$$

and the supremum is achieved as a maximum at the unique solution $x \in (0,\infty)$ of the equation

$$x^{p-1}(1+(1+x)^{1-p}) = 2^p.$$

Proof: By Theorem 3.4, the competitive ratio is $\sup_{x\ge 0} f_p(x,1/2)^{1/p}$, since $f_p(x,1) = 1$. This yields the value of the competitive ratio given in the theorem. Since $f_p(0,1/2) < 1$ and $\lim_{x\to\infty} f_p(x,1/2) = 1$, the supremum must be achieved as the maximum of $f_p(x,1/2)$. Therefore the corresponding value of x satisfies $(f_p(x,1/2))' = 0$, which is equivalent to the equation in the theorem. The solution is unique since the left-hand side of the equation is increasing while the right-hand side is a constant. ■

Theorem 4.4 *For any $p > 1$ the competitive ratio of any on-line algorithm over two machines is at least $C_{Greedy,2}$.*

Proof: Consider the sequence $(1, 1, x)$. If the on-line algorithm assigns the first two jobs on different machines, the last job is given to the on-line algorithm. The resulted assignment of the sequence is the same as the worst case example of Greedy, and we are done. If the on-line algorithm assigns the first two jobs on the same machine, it does not get another job and its competitive ratio is $2^{(p-1)/p}$. It is easy to show by convexity arguments of the function y^p that for all $x \geq 0$

$$\frac{1+(1+x)^p}{2^p+x^p} \leq \frac{(2+x)^p}{2^p+x^p} = \left(\frac{2+x}{2}\right)^p \cdot \frac{2^p}{2^p+x^p} \leq \frac{2^p+x^p}{2} \cdot \frac{2^p}{2^p+x^p} = 2^{p-1}$$

and therefore

$$\left(\frac{1+(1+x)^p}{2^p+x^p}\right)^{\frac{1}{p}} \leq 2^{\frac{p-1}{p}}.$$

Hence, by Theorem 4.3 we are done. ∎

Similarly to Theorem 4.2 a lower bound for general $p > 1$ and arbitrary number $m \geq 2$ of machines can be proved. The lower bound depends on p and grows to $\frac{3}{2}$ as p grows to ∞.

Theorem 4.5 *For any $p > 1$ the competitive ratio $C_{A,m}$ of any on-line algorithm A for any $m \geq 2$ machines is at least c where*

$$c = \frac{3}{2}\left(\frac{1-\frac{1}{3^{p-1}}}{2\cdot(1+(\frac{3}{4})^p+\frac{1}{4^p}-\frac{1}{2^{p-2}})}\right)^{\frac{1}{p}} = \frac{3}{2} - \Theta\left(\frac{1}{p}\right).$$

Proof: Consider the sequence of m jobs of weight 1 and $m/2$ jobs of weight 2. The details are similar to the proof of Theorem 4.2 and are omitted. ∎

5 A better algorithm

Our algorithm uses the same intuition as the algorithms better than greedy for the classical makespan scheduling problem [5, 1]. Namely, instead of trying to balance all the machines, we maintain a constant fraction of the machines that are reserved for large jobs. In our case we schedule the small jobs on the remaining machines greedily. Note that this does not work for makespan scheduling: if many small jobs come, we schedule them evenly on a constant fraction of machines, and then a sequence of jobs with weight equal to the current maximum load forces the makespan arbitrarily close to twice the optimum.

We define an algorithm $A(t, \alpha)$ for any $t > 0$ and $0 < \alpha < 1$; we put $k = \lfloor \alpha m \rfloor$. The choice of t and α clearly influences the performance of the algorithm. However, since the exact analysis is beyond our current techniques, we are not trying to optimize them.

> **Algorithm** $A(t, \alpha)$: Before an arrival of a job, reorder the machines so that the loads are non-decreasing; let L_{k+1} be the $(k + 1)$th smallest load. Upon arrival of a job j with weight w, if $w \geq tL_{k+1}$, schedule the job on machine 1, otherwise schedule it on machine $k + 1$.

Theorem 5.1 *For every p, there exist m_0, t, α, and $\delta > 0$ such that for any $m > m_0$, $C_{A(t,\alpha),m} \leq C_{Greedy} - \delta$.*

The proof of Theorem 5.1 occupies the rest of this section and has the following structure. First, in Lemma 5.2 we prove that if the competitive ratio of a flat shape is very close to C_{Greedy}, then the shape must look very similar to the unique worst case example for Greedy found in Theorem 3.4. This is intuitively clear, but the precise formulation and proof are a bit technical; in particular we have to reexamine the transformations from Lemma 3.2 and Lemma 3.3. Using Lemma 5.2 we prove that for sufficiently small α, in any schedule S produced by the algorithm such that $C(S)$ is close to C_{Greedy}, there must be many machines with load at least $1+t$. Lemma 5.3 shows that in such a case there are k jobs with large load that are scheduled on machines with the load of remaining jobs upper bounded by some $y < 1$; here we use the crucial properties of the algorithm. Last, we show that the existence of such k jobs, together with previously established conditions, implies that the competitive ratio is bounded away from C_{Greedy}.

Let (x_p, β_p) be the unique maximum of the function $f_p(x, \beta)$ from the analysis of the greedy algorithm, Theorem A.1. Fix constants t and $\bar{h}$ such that $x_p > t > \bar{h} > 1/\beta_p$; such constants exist, since we have proved that $x_p\beta_p > 1$. Fix a constant $y < 1$, such that $y(1+t) - 1 > \bar{h} > 1/\beta_p$. Such y clearly exists. Note that it follows that $yt > 1$.

Lemma 5.2 *There exist constants $\gamma, B, \delta_1 > 0$ and m_1 such that for every $m > m_1$, if $Q = (1, v)$ is a flat shape satisfying $C(Q) \geq C_{Greedy} - \delta_1$, then $h(Q) \leq \bar{h}$, $(OPT(Q))^p \leq Bm$, and there are at least γm machines with $L_i(Q) \geq 1 + t$.*

Proof: Fix $\Delta_x > 0$ such that $t < x_p - \Delta_x$ and Δ_β such that $1/\bar{h} < \beta_p - \Delta_\beta$. Now choose δ_1 such that $(f_p(x, \beta))^{1/p} \geq C_{Greedy} - \delta_1$ implies $|x - x_p| \leq \Delta_x$ and $|\beta - \beta_p| \leq \Delta_\beta$. Such a δ_1 exists since f_p is continuous and has a unique maximum (and $f_p \to 1$ for $x \to \infty$).

Let Q be a shape from the assumption of the lemma, let $h = h(Q)$. Now we transform Q as follows. First, if there are machines i, i' with $0 < v_i \leq v_{i'} < h$, we decrease v_i and increase $v_{i'}$ by the same amount $\max\{v_i, h - v_{i'}\}$. This does not change the load vector of the optimal schedule, and the value of the modified shape can only increase due to convexity of x^p. We continue this process until there is at most one machine with $0 < v_i < h$. Next we apply Lemma 3.2 and then Lemma 3.3. As a result we obtain a flat shape $Q' = (a, v')$ such that all nonzero components of v' are the same, $h(Q') = h$, $OPT(Q') = OPT(Q)$, $C(Q') \geq C(Q)$, and $1 - h/m \leq a \leq 1 + h/m$. The bound on a follows since a can change only in Lemma 3.2, and this is applied only to one machine with $0 < v_i < h$. (In fact, this is the reason why we need to apply a different transformation if there are two or more such machines.) The competitive ratio of Q' is bounded by

$$C_{Greedy} - \delta_1 \leq C(Q) \leq C(Q') \leq f_p(x/a, a/h)$$

for x equal to the non-zero components of v'. By the choice of δ_1, $|x/a - x_p| \leq \Delta_x$ and $|a/h - \beta_p| \leq \Delta_\beta$.

Choose $m_0 \geq 2$ such that $1/\bar{h} \leq \beta - \Delta_\beta - 1/m_0$. Now it follows that

$$\frac{1}{\bar{h}} \leq \beta - \Delta_\beta - 1/m \leq \frac{a}{h} - \frac{1}{m} \leq \frac{1}{h},$$

hence $h \leq \bar{h}$.

Put $B = 2^p(1 + x_p + \Delta_x)^p$. By the choice of m_0 it also follows that $a \leq 2$. Hence

$$(OPT(Q))^p = (OPT(Q'))^p \leq m(a+x)^p \leq m(a + a(x_p + \Delta_x))^p \leq mB.$$

Let l be the number of machines with $L_i(Q) \geq 1 + t$. It remains to prove that $l \geq \gamma m$ for a suitable constant γ. First we prove that we can assume that all the loads are bounded by some constant. Given a number $s > \bar{h}$, let $Q'' = (1, v'')$ be defined by $v_i'' = \min\{s, v_i\}$. Note that $v_i'' \geq t$ for exactly l machines. Let $\Delta(a,b) = ((a+b)^p - a^p)^{1/p}$. Let $v_i = s + \sigma$. If we change only this v_i, $(\|L(Q)\|_p)^p$ decreases by $\Delta(s+1,\sigma)$ and $(OPT(Q))^p$ decreases by $\Delta(s,\sigma)$. A standard calculation shows that

$$\lim_{s\to\infty} \sup_{\sigma>0} \frac{\Delta(s+1,\sigma)}{\Delta(s,\sigma)} = 1.$$

Hence for a sufficiently large s and arbitrary $\sigma > 0$

$$\frac{\Delta(s+1,\sigma)}{\Delta(s,\sigma)} \leq C_{Greedy} - \delta_1.$$

Fix such s. Now it follows that $C(Q'') \geq C_{Greedy} - \delta_1$, since $C(Q) \geq C_{Greedy} - \delta_1$, and the ratio of the differences in each step is at most $C_{Greedy} - \delta_1$.

Now construct $Q''' = (a''', v''')$ from Q'' by the same transformations as Q' from Q. Let x be the resulting common value of all non-zero components of v'''. Since $C(Q''') \geq C_{Greedy} - \delta_1$, it must be the case that $x/a''' \geq x_p - \Delta_x$. Since $a''' \geq 1 - \bar{h}/m$, we have $x \geq \bar{t}$ for some constant $\bar{t} > t$ and sufficiently large m; fix $m_1 \geq m_0$ appropriately. Examining the transformations, x is the pth power mean of l numbers upper-bounded by s and $m - l$ numbers upper-bounded by t. Since $x \geq \bar{t} > t$, $l \geq \gamma m$ for some constant γ. ∎

Fix the constants γ, B, δ_1 and m_1 so that the previous lemma holds. Now consider a schedule S generated by the algorithm $A(t,\alpha)$. W.l.o.g. we assume that the $(k+1)$st smallest load is 1. (If it is 0, then S is an optimal schedule. Otherwise multiply all the weights by an appropriate constant; this does not change the behavior of the algorithm).

Define a shape $\bar{Q} = (b, v)$ by $b_i = \min\{1, L_i(S)\}$ and $v_i = L_i(S) - b_i$. For each i, when the last job was scheduled on the machine i, the load was at most 1, therefore $\bar{Q}$ is a shape of S and $C(S) \leq C(\bar{Q})$. Now let $Q = (1, v)$ be a flat shape corresponding to $\bar{Q}$. We want to show that $C(Q)$ is close to $C(\bar{Q})$ for a sufficiently small α. The load vector of Q is the same as $L(\bar{Q})$ except that the load of k machines increased to 1. Clearly $\|L(Q)\|_p \geq \|L(\bar{Q})\|_p$. Adding the extra load of at most k increases in the optimal schedule load of some machines to $h(Q)$. This contributes to $(OPT(Q))^p$ at most $kp(h(Q))^{p-1}$, since the derivative of x^p is increasing. Using also the facts that $m(h(Q))^p \leq (OPT(Q))^p$ and $h(Q) \geq 1$, we obtain

$$(OPT(Q))^p - (OPT(\bar{Q}))^p \leq kp(h(Q))^{p-1} \leq \frac{kp(OPT(Q))^p}{mh(Q)} \leq \alpha p(OPT(Q))^p$$

or

$$OPT(Q) \leq OPT(\bar{Q})/(1-\alpha p)^{1/p} \, .$$

Fix some $\delta_2 < \delta_1$ and put $\alpha_1 = \frac{1}{p}\left(1 - \left(\frac{C_{Greedy}-\delta_1}{C_{Greedy}-\delta_2}\right)^p\right)$. Assume for the rest of this section that $C(S) \geq C_{Greedy} - \delta_2$ and $\alpha < \alpha_1$. It follows that

$$\begin{aligned} C(Q) &\geq C(\bar{Q})(1-\alpha_1 p)^{1/p} \geq C(S)(1-\alpha_1 p)^{1/p} \\ &\geq (C_{Greedy} - \delta_2)(1-\alpha_1 p)^{1/p} = C_{Greedy} - \delta_1. \end{aligned}$$

Hence Q satisfies the assumptions of Lemma 5.2, and, in particular, there must be γm machines with load $L_i(S) = L_i(Q) \geq 1 + t$.

Lemma 5.3 *Let $l \geq 0$ be arbitrary, let S be any schedule generated by the algorithm $A(t, \alpha)$, normalized so that the $(k+1)$st smallest load is 1, and let z be the smallest load in S. If there are $l + k$ machines with $L_i(S) \geq 1 + t$, then there exists a shape $R = (a, u)$ of S such that the machines are partitioned into*

- *k "small" machines with $z \leq a_i \leq 1$ and $u_i = 0$,*
- *l "bad" machines with $a_i = z$ and $u_i \geq t + 1 - z$,*
- *k "good" machines with $a_i = y$ and $u_i \geq yt$, and*
- *$m - l - 2k$ "ordinary" machines with $a_i = 1$ and u_i arbitrary.*

Proof: Let c_i be the load of the machine i before the last job scheduled on it. We will construct R so that always $a_i \geq c_i$ and $u_i = L_i - a_i$. This implies that R is a shape of S.

For the k machines with smallest load we put $a_i = L_i$. These are the small machines. By the properties of the algorithm, no job was so far assigned to a machine with load more than 1, hence every remaining machine can be assigned $a_i = 1$ and be an ordinary one. If $L_i \geq 1 + t$, then the last job assigned to i has weight at least t, therefore according to the algorithm it was assigned to a machine with the smallest load at some previous time. Hence $c_i \leq z$ and we may assign $a_i = z$ making this machine bad. It remains to find the k good machines.

If $c_i \leq y$ for all machines with $L_i \geq 1 + t$, we choose k of them and make them good by putting $a_i = y$. Otherwise at some time τ a big job was assigned on a machine with load at least y. Hence at time τ the load of all machines was at least y. Let τ' be the first time when the $(k+1)$st smallest load was at least y. At that time all the k smallest loads were strictly less than y. Hence between time τ' and τ, the algorithm scheduled k jobs on the k small machines. These jobs have size at least ty, by the condition in the algorithm. Furthermore, no other jobs were scheduled on these machines later, as $ty > 1$. Hence these k machines have $c_i \leq y$ and can be made good. Some of them may be the machines with $L_i \geq 1 + t$, however, we still have at least l machines that can be made bad. ■

Let the shape R and the value z be as in Lemma 5.3 and let the shape Q be the same as before Lemma 5.3. Let $l = \gamma m - k$; it follows that $l \geq \gamma m - \alpha m \geq 0$ if $\alpha \leq \gamma$. To finish the proof of Theorem 5.1, we show that for suitable parameters $OPT(R)$ is significantly larger than $OPT(Q)$. Define $H = y(1+t) - 1$. Recall that by Lemma 5.2 and the choice of y, $h = h(Q) \leq \bar{h} < H \leq t$. The optimal schedule for R is obtained from the optimal schedule for Q so that we (i) move weight $l(1-z)$ from the machines with load $h = h(Q)$ to the big jobs from the bad machines, which have weight at least $t \geq H$, (ii) move weight $k(1+y)$ from the machines with load h to the big jobs from the good machines, which have weight at least H, and (iii) remove a weight of at most $k(1-z)$ (corresponding to the small machines) from machines with load h. We are always removing the weight from machines with load at most h and putting the weight to machines with load at least H. Using the fact that the derivative of x^p is nondecreasing, we bound the resulting difference as follows:

$$\begin{aligned}
&(OPT(R))^p - (OPT(Q))^p \\
&\quad \geq \ (l(1-z) + k(1-y)) \cdot (pH^{p-1} - ph^{p-1}) - k(1-z) \cdot ph^{p-1} \\
&\quad \geq \ (1-z)pm((\gamma - \alpha)(H^{p-1} - h^{p-1}) - \alpha h^{p-1}) + \lfloor \alpha m \rfloor p(1-y)(H^{p-1} - h^{p-1}).
\end{aligned}$$

Since $\gamma, p, H, \bar{h}$ are constants and h satisfies $h \leq \bar{h} < H$, we can choose α sufficiently small so that the first term is always positive. Then, for sufficiently large m, the second term is at least εm for some $\varepsilon > 0$. Hence, using $(OPT(Q))^p \leq Bm$, we obtain

$$(OPT(R))^p \geq (OPT(Q))^p + \varepsilon m \geq (OPT(Q))^p(1 + \varepsilon/B)$$

and

$$C(S) \ \leq \ C(R) = \frac{\|L(R)\|_p}{OPT(R)} \leq \frac{\|L(Q)\|_p}{OPT(Q)(1+\varepsilon/B)^{1/p}} \leq \frac{C_{Greedy}}{(1+\varepsilon/B)^{1/p}} \leq C_{Greedy} - \delta$$

for a sufficiently small $\delta > 0$. Choosing δ such that simultaneously $\delta < \delta_2$ finishes the proof of Theorem 5.1.

Appendix

A Competitive Ratio of Greedy

Define

$$\begin{aligned}
G &= \{(x, \beta) \in \mathcal{R}^2 \mid 0 < x, \ 0 < \beta < 1\} \\
\delta_p &= (p-1)p^{-\frac{p}{p-1}}
\end{aligned}$$

and the system of equations S, consisting of (1) and (2):

$$1 - \delta_p x = (1+x)^{1-p} \tag{1}$$

$$x\beta = p^{\frac{1}{p-1}}. \tag{2}$$

Theorem A.1 *The supremum of $f_p(x, \beta)$ over $x > 0$, $0 < \beta < 1$ is achieved as a unique maximum in G. The maximum is achieved at the unique point (x, β) which is the solution to the system S in the region G. Furthermore, the competitive ratio is*

$$C_{Greedy} = \left(1 + \frac{1}{x}\right)^{\frac{p-1}{p}}. \tag{3}$$

Proof: It is easy to verify that

$$\begin{aligned}
f_p(0,\beta) &= \beta^{p-1} \leq 1 \quad (p>1, 0 \leq \beta \leq 1) \\
\lim_{x\to\infty} \sup_{0<\beta<1} f_p(x,\beta) &= 1 \qquad (p>1) \\
\lim_{\beta\to 0^+} \sup_{x>0} f_p(x,\beta) &= 0 \\
f_p(x,1) &= 1
\end{aligned}$$

and therefore, the supremum of the function $f_p(x,\beta)$ is achieved as a local maximum in G.

Since the function $f_p(x,\beta)$ has derivatives in G, the point that achieves the maximum (which must exists) satisfies $\frac{\partial f_p}{\partial x} = \frac{\partial f_p}{\partial \beta} = 0$. In lemma A.2, we prove this system of equations implies the system S, i.e., (1) and (2).

Next we prove that the solution to the system S in G is unique. Denote:

$$\begin{aligned}
LHS(x) &= 1 - \delta_p x \\
RHS(x) &= (1+x)^{1-p}
\end{aligned}$$

Clearly, $LHS(0) = RHS(0)$, $0 > LHS'(0) > RHS'(0)$, the function $LHS(x)$ is linear, and $RHS(x)$ is convex and positive in $(-1,\infty)$. Therefore equation (1) has a unique solution for x in $(0,\infty)$, which implies the system S has a unique solution in G. Thus, $f_p(x,\beta)$ has a unique maximum in G. By lemma A.2 its value is $\left(1+\frac{1}{x}\right)^{p-1}$, which completes the proof. ■

Lemma A.2 *For $p>1$ and $(x,\beta)\in G$ the equations*

$$\frac{\partial f_p}{\partial x} = \frac{\partial f_p}{\partial \beta} = 0$$

imply the system S, i.e., (1) and (2), and the equation

$$f_p(x,\beta) = \left(1+\frac{1}{x}\right)^{p-1}. \tag{4}$$

Proof: Denote:

$$M = (1-\beta)(1+x)^p + \beta \tag{5}$$

$$N = (1-\beta)x^p + \beta^{1-p}. \tag{6}$$

Clearly $f_p = \frac{M}{N}$, and therefore $\frac{\partial f_p}{\partial x} = \frac{\partial f_p}{\partial \beta} = 0$ implies

$$f_p = \frac{M}{N} = \frac{\frac{\partial M}{\partial x}}{\frac{\partial N}{\partial x}} = \frac{\frac{\partial M}{\partial \beta}}{\frac{\partial N}{\partial \beta}}. \tag{7}$$

Calculating the derivatives we obtain

$$\frac{\frac{\partial M}{\partial x}}{\frac{\partial N}{\partial x}} = \frac{(1+x)^{p-1}}{x^{p-1}} \tag{8}$$

$$\frac{\frac{\partial M}{\partial \beta}}{\frac{\partial N}{\partial \beta}} = \frac{(1+x)^p - 1}{x^p + (p-1)\beta^{-p}} \tag{9}$$

Equating M/N and the right-hand side of (8) gives (4) immediately. Equating the right-hand sides of (8) and (9) and solving for β we get

$$\beta^{-p} = \frac{x^{p-1}}{p-1}(1 - (1+x)^{1-p}). \tag{10}$$

Equating M/N and (8) gives

$$\frac{(1+x)^p + \frac{\beta}{1-\beta}}{x^p + \frac{\beta}{1-\beta} \cdot \beta^{-p}} = \frac{(1+x)^{p-1}}{x^{p-1}}$$

Solving for $(1-\beta)/\beta$ yields

$$\frac{1-\beta}{\beta} = -(1+x)^{1-p} + \frac{\beta^{-p}}{x^{p-1}}.$$

Substituting equation (10) gives

$$\frac{1}{\beta} = \frac{p}{p-1}(1 - (1+x)^{1-p}). \tag{11}$$

Substituting the last equation back in (10) gives

$$\left(\frac{p}{p-1}\right)^p \cdot (1 - (1+x)^{1-p})^p = \frac{x^{p-1}}{p-1}(1 - (1+x)^{1-p}).$$

By cancellation and raising to the power of $\frac{1}{p-1}$ we prove (1):

$$1 - (1+x)^{1-p} = x \cdot (p-1)p^{-\frac{p}{p-1}} = x \cdot \delta_p.$$

Substituting the last equation back in (11) proves (2):

$$\frac{1}{\beta} = \frac{p}{p-1} \cdot (p-1)p^{-\frac{p}{p-1}} \cdot x = xp^{-\frac{1}{p-1}}.$$

■

B Approximating the Competitive Ratio of Greedy

In this section we prove Theorem 3.6, which claims that $C_{Greedy} = 2 - \Theta((\ln p)/p)$. Denote

$$g_p(x) = {\delta_p}^{-1}(1 - (1+x)^{1-p})$$

and define x_p to be the unique solution in $(0, \infty)$ of the equation $x = g_p(x)$. This equation is equivalent to (1), hence there exists such a unique solution by Theorem A.1 and it satisfies $x_p > 1$ by Theorem 3.4. By the monotonicity and continuity of g_p we get

$$\frac{1 - 2^{1-p}}{\delta_p} = g_p(1) < g_p(x_p) < \lim_{x \to \infty} g_p(x) = \frac{1}{\delta_p}.$$

Since $(1-2^{1-p})^{-1} = 1 + O(2^{-p})$, we get

$$\frac{1}{x_p} = \delta_p + O\left(\frac{1}{2^p}\right).$$

Since $e^z = 1 + z + O(z^2)$, we obtain

$$\begin{aligned}
\delta_p &= (p-1)p^{-\frac{p}{p-1}} = \left(1-\frac{1}{p}\right) e^{-\frac{\ln p}{p-1}} \\
&= \left(1-\frac{1}{p}\right)\left(1-\frac{\ln p}{p-1}+O\left(\frac{\ln^2 p}{p^2}\right)\right) = 1 - \frac{\ln ep}{p} + O\left(\frac{\ln^2 p}{p^2}\right).
\end{aligned}$$

Using Theorem A.1 we get

$$C_{Greedy} = \left(1+\delta_p+O\left(\frac{1}{2^p}\right)\right)^{1-\frac{1}{p}} = \left(2-\frac{\ln ep}{p}+O\left(\frac{\ln^2 p}{p^2}\right)\right)^{1-\frac{1}{p}} = 2 - \Theta\left(\frac{\ln p}{p}\right).$$

C Experimental Results

In this section we give numerical approximations for C_{Greedy}. First we prove the following lemma.

Lemma C.1 $\frac{1}{2^{p-1}} + \delta_p < 1$ *for* $p > 1$.

Proof: Denote:

$$\begin{aligned}
s(p) &= \tfrac{1}{2^{p-1}} + \delta_p \qquad &&\text{for} \quad p > 1 \\
t(p) &= \tfrac{\ln p}{p-1} &&\text{for} \quad p > 1.
\end{aligned}$$

If $p \geq 2$, then $p \leq 2^{p-1}$ and therefore

$$s(p) = \frac{1}{2^{p-1}} + (p-1)p^{-\frac{p}{p-1}} < \frac{1}{2^{p-1}} + \frac{p-1}{p} = \frac{1}{2^{p-1}} - \frac{1}{p} + 1 \leq 1.$$

Otherwise, $1 < p < 2$. For all $z \in \mathcal{R}$, $\frac{z}{e^z} \leq \frac{1}{e}$ and therefore

$$\begin{aligned}
\frac{ds}{dp} &= -\ln 2 \cdot \frac{1}{2^{p-1}} + \frac{\ln p}{p-1} \cdot \frac{1}{p^{\frac{p}{p-1}}} \\
&= -\ln 2 \cdot \frac{1}{2^{p-1}} + \frac{t(p)}{e^{t(p)}} \cdot \frac{1}{p} \leq -\ln 2 \cdot \frac{1}{2^{p-1}} + \frac{1}{e} \cdot \frac{1}{p} < 0.
\end{aligned}$$

The last inequality holds since $2^{p-1} < p \cdot e \ln 2$, for $1 < p < 2$. Therefore $s(p)$ in monotone decreasing in $(1, 2)$. Since, $\lim_{p \to 1^+} s(p) = 1$ the proof of the lemma is completed. ■

For a fixed $p > 1$, define $(a_n)_{n=0}^{\infty}$, $(b_n)_{n=0}^{\infty}$ by $a_0 = 1$, $a_{n+1} = g_p(a_n)$ and $b_0 = \infty$, $b_{n+1} = g_p(b_n)$, where the function g_p is defined in the previous section.

Lemma C.2

$$a_0 < a_1 < \ldots\ldots < a_n < x_p < b_n < \ldots\ldots < b_1 < b_0$$

and

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = x_p.$$

Proof: By lemma C.1 $a_0 < g_p(a_0) = a_1$. Obviously $b_1 < b_0$, and therefore the lemma may be easily derived by using the continuity and monotony of g_p. ■

It can be easily shown that when p decreases to 1, x_p increases to the unique solution in $(0, \infty)$ to the equation $\ln(1 + x) = \frac{x}{e}$ - x_{1+}, and the competitive ratio decreases to 1. In addition, when p increases to ∞, x_p decreases to 1 and the competitive ratio increases to 2. Figure 5 shows the competitive ratio, x_p, β_p, $x_p\beta_p$ for various p up to four decimal digits precision. Calculations are based on the sequences $(a_n)_{n=0}^{\infty}$, $(b_n)_{n=0}^{\infty}$ and the previous lemma.

p	C_{Greedy}	x_p	β_p	$x_p \cdot \beta_p$
$\searrow 1$	$\searrow 1$	$\nearrow x_{1+}$	$\searrow \beta_{1+} = \frac{e}{x_{1+}}$	$\nearrow e$
1.001	1.0002	4.7557	0.5713	2.7169
1.01	1.0019	4.7256	0.5724	2.7048
1.1	1.0186	4.4498	0.5829	2.5937
1.25	1.0449	4.0734	0.5994	2.4414
1.5	1.0850	3.6042	0.6243	2.2500
1.75	1.1214	3.2617	0.6465	2.1089
2	1.1547	3.0000	0.6667	2.0000
2.5	1.2137	2.6253	0.7017	1.8420
3	1.2646	2.3692	0.7311	1.7321
4	1.3485	2.0413	0.7776	1.5874
5	1.4150	1.8405	0.8125	1.4953
7	1.5135	1.6085	0.8599	1.3831
10	1.6096	1.4346	0.9003	1.2915
15	1.7032	1.3001	0.9333	1.2134
25	1.7952	1.1912	0.9600	1.1435
50	1.8804	1.1052	0.9800	1.0831
100	1.9321	1.0582	0.9900	1.0476
1000	1.9907	1.0079	0.9990	1.0069
10000	1.9988	1.0010	0.9999	1.0009
$\nearrow \infty$	$\nearrow 2$	$\searrow 1$	$\nearrow 1$	$\searrow 1$

Figure 5: Approximation of the competitive ratio for various p

References

[1] S. Albers. Better bounds for on-line scheduling. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 130–139, 1997.

[2] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms*, pages 493–500, 1997.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symp. on the Theory of Computing*, pages 623–631, 1993.

[4] B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan, and J. Vitter. Load balancing in the l_p norm. In *Proc. 36th IEEE Symp. on Found. of Comp. Science*, pages 383–391, 1995.

[5] Y. Bartal, A. Fiat, H. Karloff, and R. Vorha. New algorithms for an ancient scheduling problem. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 51–58, 1992.

[6] 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.

[7] B. Chen, A. van Vliet, and G. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16:221–230, 1994.

[8] R.A. Cody and E.G. Coffman, Jr. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *J. Assoc. Comput. Mach.*, 23(1):103–115, January 1976.

[9] G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *Siam Journal on Computing*, 22(2):349–355, 1993.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, San Francisco, 1979.

[11] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[12] J.Y.T. Leung and W.D. Wei. Tighter bounds on a heuristic for a partition problem. *Information Processing Letters*, 56:51–57, 1995.