Load balancing of temporary tasks in the ℓ_p norm

Yossi Azar[*] Amir Epstein[†] Leah Epstein[‡]

Abstract

We consider the on-line load balancing problem where there are m identical machines (servers). Jobs arrive at arbitrary times, where each job has a weight and a duration. A job has to be assigned upon its arrival to exactly one of the machines. The duration of each job becomes known only upon its termination (this is called temporary tasks of unknown durations). Once a job has been assigned to a machine it cannot be reassigned to another machine. The goal is to minimize the maximum over time of 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 at most 2.23 of the optimum. We show (an asymptotic) lower bound of 1.79 on the competitive ratio of the greedy algorithm. We also show a lower bound of 1.44 on the competitive ratio of any deterministic algorithm.

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 competitive ratio of greedy with respect to the ℓ_p norm. We show that the greedy algorithm performs within at most $2 - \Omega(1/p)$ of the optimum. We also show a lower bound of $2 - O(\ln p/p)$ on the competitive ratio of any on-line algorithm.

1 Introduction

We are given m parallel identical machines and a number of independent jobs (tasks) arriving at arbitrary times, where each job has a weight and a duration. A job should be assigned upon its arrival to exactly one of the machines based only on the previous jobs without any knowledge on the future jobs, thus increasing the **load** on this machine by its weight for the duration of the job. The duration of each job becomes known only upon its termination (this is called temporary tasks of unknown durations). The **load** of a machine is the sum of the weights of the jobs assigned to it. For any ℓ_p norm we define the **cost** of an assignment for an input sequence of jobs as the maximum over time of the ℓ_p norm of the load vector. Specifically, the ℓ_∞ norm is the makespan (or maximum load) and the ℓ_2 norm is the Euclidean norm, which is equivalent to the sum of the squares of the load vector. The goal of an assignment algorithm is to assign all the jobs so as to minimize the cost.

Consider for example the case where the weight of a job corresponds to its machine disk access frequency. Each job may see a delay that is proportional to the load on the machine it is assigned to. Then the *average* delay is proportional to the sum of squares of the machines loads (namely the ℓ_2 norm of the corresponding machines load vector) whereas the *maximum* delay is proportional to the maximum load.

[*]School of Computer Science, Tel-Aviv University. E-Mail: azar@math.tau.ac.il. Supported in part by the Israel Science Foundation.

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

[‡]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-Mail: lea@.idc.ac.il. Supported in part by the Israel Science Foundation.

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 first summarize **our results**.

- For the sum of the squares of loads, we show that the greedy algorithm is at most 2.2293-competitive for any number of machines. In fact, for $m = 2$ greedy algorithm is optimal and its competitive ratio is 1.309 and for $m = 3$ we can improve the upper bound to $\frac{19}{9} \approx 2.1111$.
- For the sum of the squares of loads, we show that there is no on-line algorithm that is 1.4472-competitive.
- For the sum of the squares of loads, we show a lower bound of $\frac{4}{3} - O(\frac{1}{m})$ on the competitive ratio of any on-line algorithm and we show a lower bound of $\frac{4}{3}$ for m divisible by 3.
- For the sum of the squares of loads, we show (an asymptotic) lower bound of 1.7906 on the competitive ratio of the greedy algorithm.
- For the general ℓ_p norm (for any $p > 1$), we show that the greedy algorithm is at most $2 - \Omega(1/p)$-competitive for any number of machines.
- For the general ℓ_p norm (for any $p > 1$), we show (an asymptotic) lower bound of $2 - O(\ln p/p)$ on the competitive ratio of any on-line algorithm.
- For the general ℓ_p norm (for any $p > 1$), we show that for m=2 the greedy algorithm is an optimal on-line algorithm.

Temporary tasks, ℓ_∞ norm: For the problem of on-line load balancing of temporary tasks the upper bound is $2 - \frac{1}{m}$. This upper bound was proved for permanent tasks (tasks that never depart) by Graham [12], nevertheless Graham's analysis of the upper bound holds also for temporary tasks. The results in [4] show that his algorithm is optimal by constructing a lower bound of $2 - \frac{1}{m}$ on the competitive ratio of any deterministic on-line algorithm.

Permanent tasks, ℓ_∞ norm: The permanent tasks model is the model where tasks only arrive (on-line), but never depart. This is the classic ancient problem of scheduling jobs on identical machines minimizing the makespan (or maximum load). Graham [12] showed that the greedy load balancing algorithm is $2 - \frac{1}{m}$-competitive in this case. The greedy algorithm is an optimal on-line algorithm only for $m \leq 3$ [9].

Bartal et al. [6] were the first to show an algorithm whose competitive ratio is strictly below $c < 2$ (for all m). More precisely, their algorithm achieves a competitive ratio of $2 - \frac{1}{70}$. Later, the algorithm was generalized by Karger, Phillips and Torng [16] to yield an upper bound of 1.945. Subsequently, Albers [1] designed 1.923-competitive algorithm. Fleischer and Wahl [10] improved this result to a ratio of 1.9201.

Bartal et al. [7] showed a lower bound of 1.8370 for the problem. This result was improved by albers [1] to a ratio of 1.852 and then by Gormley et al. [11] to a ratio of 1.853. The best lower bound currently known is due to Rudin [15], who showed a lower bound of 1.88.

Permanent tasks, ℓ_p norm: Chandra and Wong [8] were the first to consider the problem of minimizing the sum of squares of the machine loads. They showed 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 [17]. Chandra and Wong [8] 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 ∞. The problem of on-line load balancing in the general ℓ_p norm (for any $p > 1$)

for permanent tasks was considered in [3]. The results in [3] show that for the sum of the squares, the greedy algorithm performs within $\frac{4}{3}$ of the optimum, and no on-line algorithm achieves a better competitive ratio. For the sum of the squares [3] also provided on-line algorithm with competitive ratio $\frac{4}{3} - \delta$, for some fixed δ, for any sufficiently large number of machines. For the general ℓ_p norm the results show that the competitive ratio of greedy algorithm is $2 - \Theta((\ln p)/p)$.

Off-line results: Azar et al. [5] presented a polynomial time approximation scheme for the problem of off-line load balancing of temporary tasks (in the ℓ_∞ norm) in the case where the number of machines is fixed. For the case in which the number of machines is given as part of the input (i.e, not fixed) they showed that no polynomial algorithm can achieve a better approximation ratio than $\frac{3}{2}$ unless $P = NP$.

For the problem of off-line load balancing of permanent tasks (in the ℓ_∞ norm), there is a polynomial time approximation scheme for any fixed number of machines [13, 18] and also for arbitrary number of machines by Hochbaum and Shmoys [14].

Off-line scheduling and load balancing of permanent tasks 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. 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].

2 Definitions and preliminaries

In the **load balancing problem** we are given m identical machines (servers) and a finite sequence of events. We denote the input sequence by $\sigma = \sigma_1, \ldots, \sigma_r$. Each event σ_i is an arrival or departure of a job (task). We view σ as a sequence of times, the time σ_i is the moment after the $i^{\underline{th}}$ event happened. We denote the weight of a job j by w_j, its arrival time by a_j and its departure time (which is unknown until it departs) by d_j. An on-line algorithm has to assign a job upon its arrival without knowing the future jobs and the durations of jobs that have not departed yet. We compare the performance of on-line algorithms and an optimal off-line algorithm that knows the sequence of jobs and their durations in advance. Let $J_i = \{j \mid a_j \leq \sigma_i < d_j\}$ be the active jobs at time σ_i. A **schedule** S is an assignment which assigns each job j to a single machine k, $1 \leq k \leq m$. For every schedule S, the **load** of machine k at time σ_i, denoted $L^i_k(S)$, is the sum of weights of all jobs assigned to machine k in S, and active at this time. The **vector of loads at time** σ_i is $L^i(S) = (L^i_1(S), \ldots, L^i_m(S))$. Our cost measure is the ℓ_p norm. Hence the **cost** of a schedule S at time σ_i is defined as $\|L^i(S)\|_p = \left(\sum_{k=1}^m (L^i_k(S))^p\right)^{\frac{1}{p}}$. The **cost** of a schedule S is defined as $\|L(S)\|_p = \max_i \|L^i(S)\|_p$. We denote the load vector with the maximum cost, by $L(S) = (L_1(S), \ldots, L_m(S))$.

The **optimal cost**, denoted *OPT*(S), is the minimal cost over all possible schedules for the given sequence of S.

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 $C(S) = \|L(S)\|_p / OPT(S)$. Let A be an on-line assignment algorithm. The **competitive ratio of** A **for 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 **for 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 where we measure the sum of squares of loads, since then the cost is $(\|L(S)\|_2)^2$. Consequently, the competitive ratios for 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 $\mathit{OPT}(R) = \inf\{\mathit{OPT}(S) \mid R \text{ is a shape of } S\}$. As we shall see, the infimum can be replaced by a minimum. The **competitive ratio** of a shape R is $C(R) = \|L(R)\|_p/\mathit{OPT}(R)$.

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 partition 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.

The following lemma, which appears in [3] shows that this really is the optimal Schedule. The proof of the lemma appears in Appendix B.1.

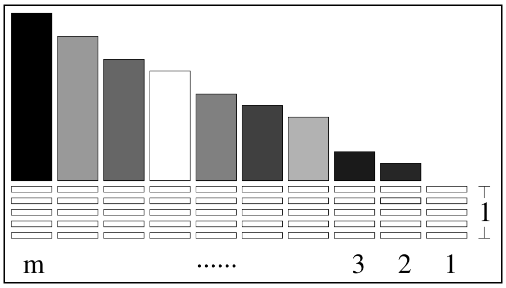

Figure 1: A shape R.

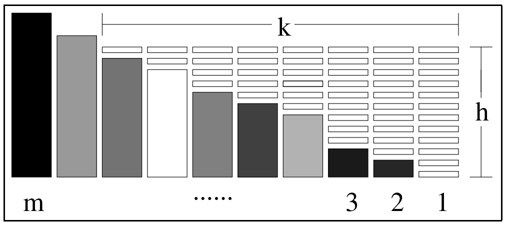

Figure 2: Optimal assignment of R

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

Now we extend the notion of shape and define continuous shapes, which are defined similarly to shapes. It is an extension, which treats the machines as points in the interval $[0, m]$. The load of machines is a function defined on that interval and is the sum of two functions. One function is the total load of very small jobs on each machine. The other function is the load of one big job on each machine. Formally a **continuous shape** is a pair $R = (a, u)$, where a and u are functions (not necessarily continuous), defined in the interval $[0, m]$. The **function of loads** of a shape is defined as $L(R) = a + u$. Where $a(t)$ represents the total load of very small jobs at point t in the interval $[0, m]$ and $u(t)$ represents the load of one big job at point t in the interval $[0, m]$. From the convexity of the function x^p it follows that the **optimal cost of a continuous shape** R is obtained by assigning some big jobs each on a separate machine and the remaining jobs are balanced evenly on the rest of the machines. Formally w.l.o.g let $u(t)$ be a non-decreasing function. There exists t' , s.t $u'(t) = u(t)$ and

$$a'(t) = \begin{cases} 0 & t \le t' \\ \frac{1}{m-t'}(\int_0^m a(t)dt + \int_{t'}^m u(t)dt) - u(t) & t' < t \end{cases}$$

and for t_1, t_2, s.t $t_1 \le t' < t_2$, it holds that $u'(t_1) \ge a'(t_2) + u'(t_2)$
and $R' = (a', u')$ gives the optimal load $L(R')$ for the shape R.

Transition from a shape to a continuous shape is defined as follows. Let $R = (a, u)$ be a shape, then its continuous shape $R' = (a', u')$ is

$$a'(t) = \{ \; a_i \qquad i - 1 < t \le i$$

$$u'(t) = \{ \; u_i \qquad i - 1 < t \le i$$

3 The greedy algorithm

In this section we analyze 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).

Note that a job departs from a machine it was assigned to.

To obtain a constant upper bound for the performance ratio of Greedy, we show that each schedule can be replaced by a very special continuous shape so that the competitive ratio does not decrease. Computing the competitive ratio is then shown to be equivalent to computing the maximum of a certain function with equality constraints over the reals. A shape $R = (h, x)$ is called **partial flat** if there exists an integer $1 \le k \le m$, and a real $c > 0$ such that the following conditions hold:

$$\begin{array}{lll} h_i = c & \text{for} & i = 1, \dots, k \\ x_i \ge 0 & \text{for} & i = 1, \dots, k \\ h_i = x_i = 0 & \text{for} & i = k+1, \dots, m. \end{array}$$

When $k = m$ the shape is called **flat**. A shape $R = (h, x)$ is called **separate** if there exists an integer $1 \le k \le m$ and a real $c > 0$, such that the following conditions hold:

$$\begin{array}{lll} h_i = 0 & \text{for} & i = 1, \dots, k \\ x_i \ge c & \text{for} & i = 1, \dots, k \\ h_i = c & \text{for} & i = k+1, \dots, m \\ x_i = 0 & \text{for} & i = k+1, \dots, m. \end{array}$$

Let S be a schedule obtained by Greedy and let $L(S)$, be the load when Greedy reaches the maximum cost. Let h be the load vector of all jobs except the last job assigned to each machine, we treat these jobs as very small jobs and call them sand jobs. Let x be the weight vector of the last job assigned to each machine. The shape $R = (h, x)$ is a shape of the schedule S, we call it the **Greedy** shape.

Lemma 3.1 *Let $R = (h, x)$ be the Greedy shape of a Greedy schedule S, normalized such that $(OPT(S))^p = m$. Then there exists a partial flat shape $R' = (h', x')$ such that $\|L(R)\|_p \leq \|L(R')\|_p$ and $OPT(R) \geq OPT(R')$ with the non-zero components of h' equal to 1, and the off-line shape of the shape R' is a separate shape.*

Proof: It is easy to see that $h_i \leq 1$ (otherwise when the last job was assigned to machine i before Greedy reached the maximum cost, $(OPT(S))^p > m$, which is a contradiction).

W.l.o.g we assume that the machines are ordered so that $h_i + x_i$ are non increasing. We perform the following transformation. Note that in each step of the transformation the cost of greedy can only increase and the cost of the off-line algorithm can only decrease. This is in particular due to the convexity of the function x^p. Figure 3 shows the obtained shape at the end of each transformation step.

1. In this step we move to a continuous shape $R = (h(t), x(t))$, where $h(t)$ and $x(t)$ are functions in the interval $[0, m]$. We treat each point t in that interval as a machine and the value $h(t)+x(t)$ as its load. Now we transform regular jobs (jobs or part of jobs) that are placed below height 1 to sand jobs. Next we push the sand jobs left such that the sand jobs height will be equal to 1 from point 0 to point V_0, where V_0 is the total volume of the sand jobs. Formally, let $R = (h(t), x(t))$ be the current shape and let t_0 be maximal, such that $h(t) + x(t) \geq 1$ then the new shape $R' = (h'(t), x'(t))$ is obtained as follows. Denote

$$V_0 = t_0 + \int_{t_0}^{m} (h(t) + x(t))dt$$

then

$$h'(t) = \begin{cases} 1 & t \leq V_0 \\ 0 & V_0 < t \end{cases}$$

$$x'(t) = \begin{cases} h(t) + x(t) - 1 & t \leq t_0 \\ 0 & t_0 < t \end{cases}$$

2. Jobs that in the off-line algorithm are scheduled on machines with sand jobs are pushed left to have the same height as the load of those machines in the off-line algorithm (which are balanced). Formally, let $R = (h(t), x(t))$ be the current shape then the new shape $R' = (h'(t), x'(t))$ is obtained as follows. Let t_1 be a minimal point such that machine t_1 has sand jobs in the off-line algorithm, let w be its total load in the off-line algorithm and let V_1 be the volume of the regular jobs on machines $[t_1, m]$. Denote

$$V_1 = \int_{t_1}^{m} x(t)dt$$

then

$$h'(t) = h(t)$$

$$x'(t) = \begin{cases} x(t) & t \leq t_1 \\ w & t_1 < t \leq t_1 + \frac{V_1}{w} \\ 0 & \text{otherwise} \end{cases}$$

3. Sand jobs on machines with no regular jobs are transformed continuously to regular jobs of height equal to w (as defined in the previous step). We put these jobs on the leftmost machines possible that have only sand jobs. We perform this process from right to left. This process continues until all machines in the greedy shape have sand jobs and a regular job. Formally, let $R = (h(t), x(t))$ be the current shape then the new shape $R' = (h'(t), x'(t))$ is obtained as follows. Let t_1 be maximal such that machine t_1 has sand jobs and a regular job. Let t_2 be maximal such that machine t_2 has jobs (any jobs). Let $s = \frac{t_1 \cdot w + t_2}{w+1}$, then

$$h'(t) = \begin{cases} 1 & t \le s \\ 0 & \text{otherwise} \end{cases}$$

$$x'(t) = \begin{cases} x(t) & t \le t_1 \\ w & t_1 < t \le s \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that in each of the transformation steps the cost of greedy can only increase and the cost of the off-line algorithm can only decrease due to the convexity of the function x^p. We denote the shape obtained by the transformation by $R' = (h', x')$. This shape has jobs on machines $[0, s]$ for some real number $0 < s < m$. Each machine $t \in [0, s]$ has sand jobs of height 1 and a regular job of height $x'(t) > 0$, other machines have no jobs assigned to them, hence this is a partial flat shape. The off-line shape has jobs of height $x'(t)$ on machines $t \in [0, s]$ and sand jobs of total volume s evenly assigned to machines $(s, m]$. In addition $\min_{t \in [0,s]} x'(t) \ge w = s/(m - s)$, hence the off-line shape is a separate shape. This completes the proof.

■

Lemma 3.2 *Let $R = (h, x)$ be a partial flat shape such that $h(t) = 1$ for $0 \le t \le s$. Assume that the off-line shape of R is a separate shape. Then there is a shape R'=(h,x'), such that for $0 \le t \le s$, $x'(t) = y$ for some value $y > 0$ and otherwise $x'(t) = 0$ and it holds that $\|L(R)\|_p \le \|L(R')\|_p$ and OPT(R) = OPT(R') .*

Proof: Let $\delta \cdot m = s$. Define

$$y = \left(\frac{1}{\delta \cdot m} \int_0^{\delta \cdot m} x^p(t)dt\right)^{\frac{1}{p}}.$$

Clearly *OPT*(R) = *OPT*(R'). Now

$$\begin{aligned} \|L(R)\|_p &= \left(\int_0^{\delta \cdot m} (1 + x(t))^p dt\right)^{\frac{1}{p}} \\ &\le \left(\int_0^{\delta \cdot m} 1 dt\right)^{\frac{1}{p}} + \left(\int_0^{\delta \cdot m} x^p(t)dt\right)^{\frac{1}{p}} \\ &= (\delta \cdot m)^{\frac{1}{p}} + (\delta \cdot m)^{\frac{1}{p}} \cdot y \\ &= (\delta \cdot m)^{\frac{1}{p}} \cdot (1 + y) \\ &= \|L(R')\|_p \end{aligned}$$

where the inequality follows from the triangle inequality for the ℓ_p norm. This completes the proof. ■

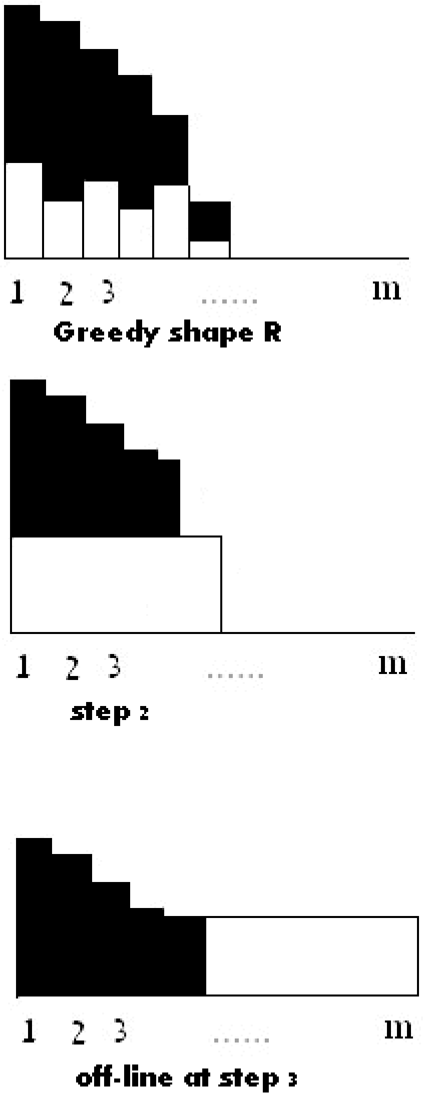

Figure 3: The transformation steps

Define the function $f(\delta, x)$ with the constraint $g(\delta, x)$.

$$f(\delta, x) = \delta \cdot (1+x)^p, \tag{1}$$

$$g(\delta, x) = \delta \cdot x^p + \frac{\delta^p}{(1-\delta)^{p-1}} = 1. \tag{2}$$

Theorem 3.3 *The competitive ratio of Greedy algorithm satisfies*

$$C_{Greedy} \leq (f_{max})^{\frac{1}{p}}$$

where f_{max} is the maximum of f with constraint (2), in the domain $0 \leq x, 0 \leq \delta \leq \frac{1}{2}$.

Proof: Let R_0 be the Greedy shape of a schedule S obtained by Greedy. For simplicity we transform to a new shape R_1 by normalizing all job weights such that

$$(\mathit{OPT}(R_1))^p = m. \tag{3}$$

If all the h components of R_1 are equal to zero then the Greedy schedule is optimal. Otherwise by applying Lemma 3.1 and Lemma 3.2 we obtain from R_1 a partial flat shape $R_2 = (h, x)$, in which all the non zero components of x are the same and its off-line shape is a separate shape such that $\|L(R_1)\|_p \leq \|L(R_2)\|_p$ and $\mathit{OPT}(R_1) \geq \mathit{OPT}(R_2)$. We have

$$\begin{aligned}
(\|L(R_2)\|_p)^p &= \delta \cdot m \cdot (1+x)^p, & (4)\\
(\mathit{OPT}(R_2))^p &= \int_0^{\delta \cdot m} x^p(t)dt + (1-\delta)m(\frac{\delta \cdot m}{(1-\delta)\cdot m})^p = \delta \cdot m \cdot x^p + \frac{\delta^p \cdot m}{(1-\delta)^{p-1}}, & (5)\\
x &\geq \frac{\delta \cdot m}{(1-\delta)\cdot m} = \frac{\delta}{1-\delta}. & (6)
\end{aligned}$$

The last inequality restricts the weight of a regular job to be greater then the total weight of sand jobs on machines with sand jobs in the off-line shape. For simplicity we divide equalities (4) and (5) by m, this does not change the ratio between the cost and the off-line cost of shape R_2. Which gives the following

$$\begin{aligned}
f(\delta, x) &= \delta \cdot (1+x)^p, & (7)\\
1 &\geq \delta \cdot x^p + \frac{\delta^p}{(1-\delta)^{p-1}}, & (8)\\
x &\geq \frac{\delta}{1-\delta}. & (9)
\end{aligned}$$

The left side of the equality is $f(\delta, x)$ by definition. The first inequality results from (5), since $(\mathit{OPT}(R_2))^p \leq (\mathit{OPT}(R_1))^p \leq m$ and the division by m.

Substituting (9) in (8) gives

$$1 \geq \frac{\delta^{p+1}}{(1-\delta)^p} + \frac{\delta^p}{(1-\delta)^{p-1}} = \frac{\delta^p}{(1-\delta)^p}$$

which yields $\delta \leq \frac{1}{2}$. We obtain the following relation for f

$$f(\delta, x) = \frac{\frac{1}{m}(\|L(R_2)\|_p)^p}{\frac{1}{m}\cdot m} \geq \frac{(\|L(R_1)\|_p)^p}{(\mathit{OPT}(R_1))^p} \geq \frac{(\|L(S)\|_p)^p}{(\mathit{OPT}(S))^p}$$

where the first inequality follows from (3) and the fact that $\|L(R_1)\|_p \leq \|L(R_2)\|_p$. Hence to bound the competitive ratio of Greedy we need to solve the following maximum problem in the domain $0 \leq \delta \leq \frac{1}{2}$ and $0 \leq x$. We need to find the maximum of $f(\delta, x)$ under the constraint (8). It is easy to see that the maximum of f is obtained when (8) is an equality. Hence we obtain the following maximum problem for f with constraint g (1), (2) in the domain $0 \leq x, 0 \leq \delta \leq \frac{1}{2}$. This completes the proof. ■

The following theorem results from Theorem 3.3. The proof appears in Appendix B.2.

Theorem 3.4 *For $p = 2$ the competitive ratio of Greedy algorithm is $C^2_{Greedy} \leq 2.2293$.*

Theorem A.2 in the Appendix gives a somewhat weaker result to theorem 3.4, using a different method for approximating the upper bound.

The following Theorem gives an upper bound of $\frac{19}{9} \approx 2.1111$ for the competitive ratio of Greedy for $p = 2$ and $m = 3$, which is an improvement of the constant upper bound. This result follows from Theorem A.1, which appears in the Appendix.

Theorem 3.5 *For $p = 2$ and $m = 3$ the competitive ratio of Greedy algorithm, $C^2_{Greedy,3} \le \frac{19}{9} \approx 2.1111$.*

Now we turn to the case of general p. The proof of the following theorems appear in Appendices B.3 and B.4.

Theorem 3.6 *For any $p > 1$ it holds $C_{Greedy} \le 2 - \Omega\left(\frac{1}{p}\right)$.*

Theorem 3.7 *For $m = 2$ the greedy algorithm is optimal and its competitive ratio is*

$$C_{Greedy,2} = \sup_{x \ge 0} \left(\frac{1 + (1+x)^p}{2^p + x^p} \right)^{\frac{1}{p}}, \tag{10}$$

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

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

From the above theorem, which claims that for $m = 2$ Greedy is optimal and its competitive ratio is the same as in the permanent tasks case, we obtain according to [3] that for the sum of the squares and for two machines Greedy is optimal and its competitive ratio is $(\sqrt{5} + 3)/4 \approx 1.309$.

4 Lower bounds

4.1 Lower bounds for sum of squares

In this section we give lower bounds for $p = 2$. We prove a lower bound for any algorithm (the proof is for $m = 3$). Then we prove a weaker lower bound for any $m \ge 3$. Finally we prove a lower bound for Greedy for a large number of machines ($m \to \infty$).

Theorem 4.1 *For any on-line assignment algorithm A, $C^2_A \ge C^2_{A,3} \ge 1.4472$.*

Proof: Consider the following sequence for three machines. First three unit jobs and one job of weight $x \ge 1$ arrive. Then two unit jobs depart. At last one job of weight 2 arrive. Consider the first three unit jobs. If algorithm A assigns two or more jobs to 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, algorithm A assigns one unit job to every machine (the off-line algorithm assigns two unit jobs to machine 1 and one unit job to machine 2). Now the next job of weight x arrives. Algorithm A assigns it to one of the machines say 1 (the off-line algorithm assigns it to machine 3). Then two unit jobs depart, which are the jobs on machines 2,3 (the jobs on machine 1 in the off-line algorithm). At last a job of weight 2 arrive. The best algorithm A can do is to assign it to one of the empty machines 2 or 3 (the off-line algorithm assigns it to machine 1). Its cost is at least $(1 + x)^2 + 2^2$, whereas the optimum cost is $2^2 + 1 + x^2$. The maximal ratio ≈ 1.4472 is achieved for $x = \sqrt{5}$. ■

Theorem 4.2 *For any number of machines $m \geq 3$ and any on-line assignment algorithm A, $C^2_{A,m} \geq 4/3 - O(\frac{1}{m})$. For m divisible by 3, $C^2_{A,m} \geq 4/3$.*

Proof: Let $m = 3k$. We consider the following sequence. First $4k$ unit jobs arrive. Then $2k$ unit jobs depart. Finally k jobs of weight 2 arrive. Consider the arrival of the first $4k$ unit jobs. Algorithm A assigns these jobs. W.l.o.g we assume that machines $1, \dots, m$ are sorted in nondecreasing order of load (the off-line algorithm assigns two jobs on each machine $1, \dots, k$ and one job on each machine $k+1, \dots, 3k$). Then $2k$ jobs depart. There exists a minimal $t \geq 2k$, such that machines $1, \dots, t$ are assigned at least $2k$ jobs. Then $2k$ jobs from machines $1, \dots, t$ depart as follows, all jobs from machines $1, \dots, t-1$ and some jobs from machine t (in the off-line algorithm the jobs on machines $1, \dots, k$ depart). At the end of this step machines $1, \dots, 2k$ are empty. Next k jobs of weight 2 arrive. The best algorithm A can do is to assign each job to an empty machine (In the off-line these jobs are assigned to machines $1, \dots, k$). Finally there are jobs of total weight $4k$ assigned to no more than $2k$ machines. Due to the convexity of the function x^p, the minimum cost is obtained when all machines have the same load, therefore its cost is at least $2k \cdot (2)^2$. The optimum cost is $k \cdot 2^2 + 2k \cdot 1^2$, which yields a ratio of $4/3$. For m not divisible by 3 a similar proof gives a ratio of $4/3 - O(\frac{1}{m})$.

■

Theorem 4.3 *For the greedy algorithm, $C^2_{Greedy} \geq 1.7906$.*

Proof: First we prove a weaker lower bound of 1.7281 for the greedy algorithm, by solving an ordinary differential equation analytically. Then by a similar proof we obtain a more complex ordinary differential equation, which has no simple solution. Hence we use a computer program to compute the competitive ratio in this case, which gives a lower bound of 1.7906.

We start with the first proof. We see the m machines as m points in the interval $(0, 1]$, machine i as the point $\frac{i}{m} \in (0, 1]$, and the load of the machines as a function $f(t)$, $f(t) = l_i$ for $\frac{(i-1)}{m} < t \leq \frac{i}{m}$, where l_i is the load of machine i. For each machine i the total load is the value of f in the interval $(\frac{i-1}{m}, \frac{i}{m}]$ and the total load of all machines is the total volume of f in the interval $(0, 1]$ multiplied by m. Let $f(k/m)$ be the load of machine k at the end of step k and let $F(k/m)$ be the volume of the jobs assigned to machines $k, \dots, m$ at the beginning of step k in the following process. For convenience we number the steps from m to 1 in decreasing order. In this process we keep the volume of jobs fixed and equal to 1 at the end of each step. We start with the arrival of infinitesimally small jobs of total volume 1, we call jobs of this type sand jobs. Both the off-line and greedy algorithms assign these jobs evenly on all the machines (total height 1 on each machine). At step k a job of height x ($x \geq 1$) arrives. Greedy assigns this job to the machine with minimum load w.l.o.g to machine k which is the one with the largest index among all machines with the minimum load $1, \dots, k$ (otherwise we swap indices) and the off-line algorithm performs the same assignment. Then the sand jobs on machines $1, \dots, k-1$ depart in greedy. In the off-line algorithm the departing jobs are composed of all the sand jobs of machine k and equal amounts of sand jobs from machines $1, \dots, k-1$ with the appropriate volume. Next sand jobs arrive with total volume $1 - F(k/m) - \frac{x}{m}$ (=1-total volume of machines $k, \dots, m$), thus keeping the total volume equal to 1. Greedy and the off-line algorithms assign the sand jobs to machines $1, \dots, k-1$ evenly, such that these machines will have the same load. At the end of step k

$$f(k/m) = \frac{1 - F(k/m)}{k/m} + x.$$

The computation of the lower bound for Greedy according to the above scenario, which is technical, appears in Appendix B.5. ■

4.2 Lower bound for general $p > 1$

In this section we construct a lower bound for general $p > 1$. The proof of the following theorem appears in Appendix B.6.

Theorem 4.4 *For any $p > 1$ and any on-line algorithm A it holds that $C_A \geq 2 - O\left(\frac{\ln p}{p}\right)$.*

Acknowledgment: We would like to thank Adi Avidor for letting us use Figures 1 and 2.

References

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29:459–473, 1999.

[2] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[3] A. Avidor, Y. Azar, and J. Sgall. Ancient and new algorithms for load balancing in the ℓ_p norm. *Algorithmica*, 29:422–441, 2001. Also in *Proc. 9th ACM-SIAM SODA*, 1998, pp. 426-435.

[4] Y. Azar and L. Epstein. On-line load balancing of temporary tasks on identical machines. In *5th Israeli Symp. on Theory of Computing and Systems*, pages 119–125, 1997.

[5] Y. Azar, O. Regev, J. Sgall, and G. Woeginger. Off-line temporary tasks assignment. *Theoretical Computer Science*, 287:419–428, 2002.

[6] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51(3):359–366, 1995.

[7] Y. Bartal, H. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[8] 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.

[9] U. Faigle, W. Kern, and G. Turan. On the performance of online algorithms for partition problems. *Acta Cybernetica*, 9:107–119, 1989.

[10] R. Fleischer and M. Wahl. Online scheduling revisited. *Journal of Scheduling*, 3(5):343–353, 2000.

[11] T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 564–565, 2000.

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

[13] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:416–429, 1969.

[14] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. Assoc. Comput. Mach.*, 34(1):144–162, January 1987.

[15] J.F. Rudin III. *Improved bounds for the on-line scheduling problem*. PhD thesis, The University of Texas at Dallas, May 2001.

[16] D. Karger, S. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996.

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

[18] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, 23:116–127, 1976.

Appendix

A The competitive ratio of Greedy

We show for $p = 2$ and $m = 3$ an upper bound of $\frac{19}{9} \approx 2.1111$ for the competitive ratio of Greedy, which is an improvement of the constant upper bound. Actually we give an upper bound in a more general way in the following theorem.

Theorem A.1 *For $p = 2$ the competitive ratio of Greedy algorithm is bounded by $C^2_{Greedy,m} \leq 3 - \frac{3}{m} + \frac{1}{m^2}$.*

Proof: We start with general p. Consider time T, when greedy reaches the maximum cost. Let t_i be the time just after the last job was assigned to machine i, such that $t_i \leq T$. Let $h_i(t_i)$ be the load of machine i, just before assigning that job and let $x_i(t_i)$ be the weight of that job. In general let $x_j(t_i)$ be the weight of the last job assigned to machine j at time t_i. Let $h_j(t_i)$ be the total load of jobs on machine j at time t_i except $x_j(t_i)$. Note that there exists a machine k such that $T = t_k$. Let Greedy, Opt(t_i) and Opt be the costs of Greedy, the optimal off-line algorithm at time t_i and the optimal off-line algorithm respectively. Denote

$$h(t_i) = \frac{\sum_{j=1}^{m}(h_j(t_i) + x_j(t_i))}{m}. \tag{11}$$

From the convexity of the function x^p

$$m \cdot h(t_i)^p \leq Opt(t_i)^p \leq Opt^p. \tag{12}$$

Substituting (11) in (12) yields

$$\sum_{j=1}^{m}(h_j(t_i) + x_j(t_i)) \leq m^{1-\frac{1}{p}} \cdot Opt. \tag{13}$$

For machine i we have (using volume considerations)

$$\begin{aligned} m \cdot h_i(T) + x_i(T) &\leq m \cdot h_i(t_i) + x_i(t_i) \\ &\leq \sum_{j=1}^{m}(h_j(t_i) + x_j(t_i)) \\ &\leq m^{1-\frac{1}{p}} \cdot Opt \end{aligned} \tag{14}$$

where the first inequality follows from the fact that no jobs arrive to machine i between times t_i and T (note that jobs can depart from machine i at that time interval). The second inequality follows from the definition of Greedy, $\forall j \neq i \; h_i(t_i) \leq h_j(t_i) + x_j(t_i)$. The last inequality is obtained by (13). The inequality (14) is equivalent to

$$h_i(T) \leq m^{-\frac{1}{p}} \cdot Opt - \frac{x_i(T)}{m}. \tag{15}$$

For convenience we denote $h_i(T)$ by h_i and $x_i(T)$ by x_i. Now we turn to the case **p=2**.
Substituting $p = 2$ in inequality (15) gives

$$h_i(T) \leq \frac{Opt}{\sqrt{m}} - \frac{x_i(T)}{m}. \tag{16}$$

It is obvious that

$$\sum_{i=1}^{m} x_i^2 \leq Opt^2 \tag{17}$$

as all jobs x_i are present at time T. In addition we use the following

$$\begin{aligned}
\sum_{i=1}^{m} h_i \cdot x_i &\leq \sum_{i=1}^{m} \left(\frac{Opt}{\sqrt{m}} - \frac{x_i}{m} \right) \cdot x_i & (18)\\
&= \frac{Opt}{\sqrt{m}} \cdot \sum_{i=1}^{m} x_i - \frac{1}{m} \cdot \sum_{i=1}^{m} x_i^2 & (19)\\
&\leq \frac{Opt}{\sqrt{m}} \cdot (\sqrt{m} \cdot Opt) - \frac{1}{m} \cdot \sum_{i=1}^{m} x_i^2 & (20)\\
&= Opt^2 - \frac{1}{m} \cdot \sum_{i=1}^{m} x_i^2 & (21)
\end{aligned}$$

where the first inequality follows from (15). The second inequality follows from (13) by assigning $p = 2$.

We bound the competitive ratio of greedy

$$\begin{aligned}
Greedy^2 &= \sum_{i=1}^{m} (h_i + x_i)^2 \\
&= \sum_{i=1}^{m} h_i \cdot (h_i + x_i) + \sum_{i=1}^{m} x_i \cdot (h_i + x_i) \\
&\leq \sum_{i=1}^{m} \left(\frac{Opt}{\sqrt{m}} - \frac{x_i}{m} \right) \cdot (h_i + x_i) + \sum_{i=1}^{m} x_i \cdot (h_i + x_i) \\
&= \frac{Opt}{\sqrt{m}} \cdot \sum_{i=1}^{m} (h_i + x_i) + (1 - \frac{1}{m}) \cdot \sum_{i=1}^{m} x_i \cdot (h_i + x_i) \\
&\leq (\frac{Opt}{\sqrt{m}}) \cdot (\sqrt{m} \cdot Opt) + (1 - \frac{1}{m}) \cdot \sum_{i=1}^{m} x_i \cdot (h_i + x_i) \\
&= Opt^2 + (1 - \frac{1}{m}) \cdot (\sum_{i=1}^{m} h_i \cdot x_i + \sum_{i=1}^{m} x_i^2) \\
&\leq Opt^2 + (1 - \frac{1}{m}) \cdot \left[Opt^2 + (1 - \frac{1}{m}) \cdot \sum_{i=1}^{m} x_i^2 \right] \\
&\leq Opt^2 + (1 - \frac{1}{m}) \cdot \left[Opt^2 + (1 - \frac{1}{m}) \cdot Opt^2 \right] \\
&= (3 - \frac{3}{m} + \frac{1}{m^2}) \cdot Opt^2
\end{aligned}$$

where the first inequality follows from (15). The second inequality follows from (13). The third inequality follows from (21). The last inequality follows from (17). This completes the proof. ■

The following theorem gives a somewhat weaker result to Theorem 3.4, using different method for approximation of the upper bound.

Theorem A.2 *For $p = 2$ the competitive ratio of Greedy algorithm is $C^2_{Greedy,m} \leq 2.2361$.*

Proof: Let S be a schedule obtained by Greedy and let $R_1 = (h, x)$ be the shape of the schedule S. For simplicity we normalize all job weights, such that

$$(OPT(R_1))^2 = m. \tag{22}$$

This does not change the competitive ratio of the schedule S. W.l.o.g we assume that in the off-line shape jobs $x_1, \ldots, x_m$ are assigned to machines $1, \ldots, m$ respectively (the machines are ordered ,s.t. x_i are non increasing) and sand jobs are assigned to machines $k+1, \ldots, m$, s.t. the total load of each machine $k+1, \ldots, m$ is the same.

It is easy to see that

$$h_i \leq 1 \tag{23}$$

(otherwise when the last job was assigned to machine i before Greedy reached the maximum cost, $(OPT(S))^2 > m$, which is a contradiction)

and

$$\forall i,\ k+1 \leq i \leq m,\ x_i \leq 1 \tag{24}$$

(otherwise $\forall i,\ 1 \leq i \leq m,\ x_i > 1$, therefore $(OPT(R_1))^2 > m$, which is a contradiction).

We bound the cost of shape R_1

$$\begin{aligned}
(\|L(R_1)\|_2)^2 &= \sum_{i=1}^{m} (h_i + x_i)^2 \\
&= \sum_{i=1}^{k} h_i \cdot (h_i + x_i) + \sum_{i=1}^{k} x_i \cdot (h_i + x_i) + \sum_{i=k+1}^{m} h_i \cdot (h_i + x_i) + \sum_{i=k+1}^{m} x_i \cdot (h_i + x_i) \\
&\leq \sum_{i=1}^{k} h_i \cdot (h_i + x_i) + \sum_{i=1}^{k} x_i \cdot (h_i + x_i) + 2 \cdot \sum_{i=k+1}^{m} (h_i + x_i) \\
&= \sum_{i=1}^{k} x_i^2 + 2 \cdot \sum_{i=1}^{k} h_i \cdot x_i + \sum_{i=1}^{k} h_i^2 + 2 \cdot \sum_{i=k+1}^{m} (h_i + x_i) \\
&\leq \sum_{i=1}^{k} x_i^2 + 2 \cdot \sum_{i=1}^{k} h_i \cdot x_i + \sum_{i=1}^{k} h_i + 2 \cdot \sum_{i=k+1}^{m} (h_i + x_i) \\
&\leq \sum_{i=1}^{k} x_i^2 + 2 \cdot \sqrt{\sum_{i=1}^{k} h_i} \cdot \sqrt{\sum_{i=1}^{k} x_i^2} + 2 \cdot (\sum_{i=k+1}^{m} h_i + \sum_{i=k+1}^{m} x_i) + \sum_{i=1}^{k} h_i
\end{aligned} \tag{25}$$

where the first inequality follows from (23) and (24). The second inequality follows from the fact that $h_i^2 \leq h_i$ (since $h_i \leq 1$). The last inequality follows from Cauchy Schwartz inequality.

From (22) we obtain the following equation for the off-line cost of shape R_1

$$\sum_{i=1}^{k} x_i^2 + (m-k)\left(\frac{(\sum_{i=1}^{m} h_i + \sum_{i=k+1}^{m} x_i)}{m-k}\right)^2 = (OPT(R_1))^2 = m$$

which is equivalent to

$$\sum_{i=1}^{k} x_i^2 + \frac{(\sum_{i=1}^{m} h_i + \sum_{i=k+1}^{m} x_i)^2}{m-k} = m. \tag{26}$$

Denote

$$\begin{aligned}
\alpha \cdot m &= \sum_{i=1}^{k} x_i^2, \\
\beta \cdot m &= \sum_{i=k+1}^{m} h_i + \sum_{i=k+1}^{m} x_i, \\
\gamma \cdot m &= \sum_{i=1}^{k} h_i, \\
\delta &= \frac{k}{m}
\end{aligned}$$

where

$$0 \leq \alpha, \beta, \gamma, \delta \leq 1. \tag{27}$$

Substituting the above new variables in (25) and (26) gives

$$(\|L(R_1)\|_2)^2 = \alpha \cdot m + 2 \cdot \sqrt{\alpha \cdot \gamma} \cdot m + 2\beta \cdot m + \gamma \cdot m, \tag{28}$$

$$\alpha \cdot m + \frac{(\beta+\gamma)^2 \cdot m}{1-\delta} = m. \tag{29}$$

Dividing the above equations by m gives

$$f(\alpha, \beta, \gamma) = \alpha + 2 \cdot \sqrt{\alpha \cdot \gamma} + 2\beta + \gamma, \tag{30}$$

$$\alpha + \frac{(\beta+\gamma)^2}{1-\delta} = 1, \tag{31}$$

where in the first equation we replace $\frac{1}{m} \cdot (\|L(R_1)\|_2)^2$, by $f(\alpha, \beta, \gamma)$.
We add the following constraint

$$\gamma \leq \delta. \tag{32}$$

This constraint results from the definitions of γ, δ and the fact that $h_i \leq 1$.

We obtain the following relation for f

$$f(\alpha,\beta,\gamma)=\frac{\frac{1}{m}(\|L(R_1)\|_2)^2}{\frac{1}{m}\cdot m}=\frac{(\|L(R_1)\|_2)^2}{(\mathit{OPT}(R_1))^2}\geq\frac{(\|L(S)\|_2)^2}{(\mathit{OPT}(S))^2}.$$

Hence to bound the competitive ratio of Greedy we can solve the following maximum problem for f in the domain (27). We have to find the maximum of $f(\alpha,\beta,\gamma)$ under the constraints (31), (32).

It is easy to see that the maximum of $f(\alpha,\beta,\gamma)$ under the constraints (31) and (32) is achieved when $\gamma=\delta$. Hence we have to find the maximum of $f(\alpha,\beta,\gamma)$ under the following constraint

$$\alpha+\frac{(\beta+\gamma)^2}{1-\gamma}=1. \tag{33}$$

We find the maximum of $f(\alpha,\beta,\gamma)$ under the constraint (33) using the Lagrange multipliers method.

By Theorem A.3 the maximum of f is achieved at $\alpha\approx 0.7236$, $\beta\approx 0.1708$, $\gamma\approx 0.2764$, and $C^2_{Greedy}\leq f(\alpha,\beta,\gamma)\approx 2.2361$. This completes the proof.

■

Theorem A.3 *Let*

$$f(\alpha,\beta,\gamma) = \alpha+2\cdot\sqrt{\alpha\cdot\gamma}+2\beta+\gamma, \tag{34}$$

$$g(\alpha,\beta,\gamma) = \alpha+\frac{(\beta+\gamma)^2}{1-\gamma}-1=0. \tag{35}$$

The maximum of f under the constraint g in the domain $0\leq\alpha,\beta,\gamma\leq 1$ is $f_{max}\approx 2.2361$.

Proof: Equation (35) is equivalent to

$$\beta+\gamma=\sqrt{(1-\alpha)(1-\gamma)}. \tag{36}$$

Substituting (36) in (34) gives

$$f(\alpha,\gamma)=\alpha+2\cdot\sqrt{\alpha\cdot\gamma}+2\sqrt{(1-\alpha)(1-\gamma)}-\gamma. \tag{37}$$

First we find the maximum of f on the boundary

1. If $\alpha=0$. Substituting α in (37) gives $f\leq 2$.
2. If $\alpha=1$. Substituting α in (35) gives $\beta=\gamma=0$. Substituting α,γ in (37) gives $f\leq 1$.
3. If $\beta=0$. Substituting β in (35) and finding the maximum of f under this constraint using Maple gives $f_{max}\approx f(\alpha=0.7913,\gamma=0.3642)\approx 2.2293$.
4. If $\beta=1$. Substituting β in (35) gives $\alpha=\gamma=0$. Substituting α,γ in (34) gives $f\leq 2$.
5. If $\gamma=0$. Substituting γ in (37) gives $f\leq 2$.
6. If $\gamma=1$. Substituting γ in (37) gives $f\leq 2$.

Now we find the local extremum points of f using the lagrange multipliers method.

Solving in Maple gives $f_{max}\approx f(\alpha=0.7236,\beta=0.1708,\gamma=0.2764)\approx 2.2361$. This completes the proof.

■

B Omitted proofs

B.1 Proof of Lemma 2.1

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 $\|(h, \ldots, h, u_{k+1}, \ldots, u_m)\|_p$.

B.2 Proof of Theorem 3.4

By Theorem 3.3 the competitive ratio of Greedy is obtained by solving the following maximum problem for f, which is obtained by substituting $p = 2$ in (1) and (2).

$$f(\delta, x) = \delta \cdot (1 + x)^2, \tag{38}$$

$$1 = \delta \cdot x^2 + \frac{\delta^2}{1-\delta}. \tag{39}$$

We solve this maximum problem. (39) gives

$$x = \sqrt{\frac{1 - \frac{\delta^2}{1-\delta}}{\delta}}. \tag{40}$$

Substituting equation (40) in equation (38) gives

$$f(\delta) = \delta \cdot \left(1 + \sqrt{\frac{1 - \frac{\delta^2}{1-\delta}}{\delta}}\right)^2. \tag{41}$$

From (41) we have to find the maximum of $f(\delta)$ in the interval $\delta \in [0, \frac{1}{2}]$.

Using Maple we can see that the maximum of f is achieved at $\delta \approx 0.3642, (x \approx 1.474)$, and $C^2_{Greedy} \leq f(\delta) \approx 2.2293$. This completes the proof.

B.3 Proof of Theorem 3.6

By Theorem 3.3 we have

$$1 = g(\delta, x) \geq \delta \cdot x^p.$$

Hence

$$\delta \leq \frac{1}{x^p}.$$

Substituting δ in (1) gives

$$f(\delta, x) \leq \frac{(1+x)^p}{x^p} = f_1(x). \tag{42}$$

Substituting $\delta \leq \frac{1}{2}$ in (1) gives

$$f(\delta, x) \leq \frac{1}{2}(1+x)^p = f_2(x).$$

Since $f_1(x)$ is a monotonically decreasing function of x and $f_2(x)$ is a monotonically increasing function of x we obtain

$$f(\delta, x) \leq f_{max} \leq f_2(x_0 = 2^{\frac{1}{p}}) = \frac{1}{2}\left(1 + 2^{\frac{1}{p}}\right)^p$$

where $x_0 = 2^{\frac{1}{p}}$ is a solution of the equation

$$f_1(x) = f_2(x).$$

Hence

$$C_{Greedy} \leq (f_{max})^{\frac{1}{p}} \leq \frac{1 + 2^{\frac{1}{p}}}{2^{\frac{1}{p}}} = 1 + \left(\frac{1}{2}\right)^{\frac{1}{p}} = 1 + e^{-\frac{1}{p}\ln 2} = 2 - \Omega\left(\frac{1}{p}\right).$$

This completes the proof.

B.4 Proof of Theorem 3.7

First we show that any schedule S obtained by Greedy has a flat shape R which is a shape of $L(S)$. Consider time T, when greedy reaches the maximum cost. Let $L = L(S)$ be the vector of loads of S at time T. W.l.o.g we assume that L_1 is the smallest component of L. We claim that the shape $R = (a, u)$ where $a = L_1$ and $u_i = L_i - a$, is a flat shape of $L(S)$. Clearly $(L(R) = L(S))$. Consider machine 2 with $u_2 > 0$. Let j be the last job assigned to the machine 2 until time T. At the time of its assignment, the load of machine 1 must have been at most a, as otherwise the greedy cost at that time would be greater then $\|L(S)\|_p$, which is a contradiction. Hence $w_j \geq L_2 - a = u_2$, and the shape $R = (a, u)$ is a flat shape.

In [3] it is shown that in the permanent tasks case for $m = 2$ the greedy algorithm is optimal and its competitive ratio is given in equation (10). The proof of the upper bound was based on the fact that each schedule S obtained by greedy has a flat shape, hence this upper bound is true also for the temporary tasks case. The proof of the lower bound gives a schedule, which is a legal schedule also for the temporary tasks case, hence this lower bound is true also for the temporary tasks case. This completes the proof.

B.5 Detailed proof of Theorem 4.3

When $m \to \infty$ $t = k/m$ is a continuous variable in the interval $[0, 1]$ and we get the following equation

$$f(t) = \frac{1 - F(t)}{t} + x. \tag{43}$$

We have $f(t) = -\frac{dF(t)}{dt}$ (since $F(t) = \displaystyle\int_t^1 f(u)du$) and we get

$$-\frac{dF(t)}{dt} = \frac{1 - F(t)}{t} + x.$$

Now we have the following first order differential equation

$$\begin{aligned} -t \cdot \frac{dF(t)}{dt} + F(t) - x \cdot t - 1 &= 0 \\ F(1) &= 0 \end{aligned}$$

It is easy to verify its solution

$$F(t) = -x \cdot t \cdot \ln(t) - t + 1. \tag{44}$$

Substituting equation (44) in equation (43) gives

$$f(t) = x \cdot \ln(t) + x + 1. \tag{45}$$

The above process continues until assigning the job with weight x to the machine represented by t_0, where $F(t_0) = 1$, i.e until the volume of all machines of greedy that were assigned a job of weight x approaches 1 (until there are no sand jobs that can depart from machines $0 \leq t < t_0$). From (44) we get

$$-x \cdot t_0 \cdot \ln(t_0) - t_0 + 1 = 1$$

which gives

$$t_0 = e^{-\frac{1}{x}}.$$

At the end of the above process each machine in the interval $[t_0, 1]$ has sand jobs and one big job of weight x. In the off-line algorithm each machine in the interval $[t_0, 1]$ has one job of weight x and the other machines have sand jobs assigned equally to them. The maximum cost of greedy and the off-line algorithms is obtained at the end of the above process due to the convexity of the function x^p. Let $Greedy(x)$ and $Opt(x)$, be the costs of greedy and the off-line algorithms as a function of x respectively.

$$\begin{aligned} Greedy^2(x) &= \int_{e^{-\frac{1}{x}}}^{1} f^2(t)dt \\ &= \int_{e^{-\frac{1}{x}})}^{1} (x \cdot \ln(t) + x + 1)^2 dt \\ &= \int_{e^{-\frac{1}{x}}}^{1} [x^2 \cdot \ln^2(t) + 2x \cdot (x+1) \cdot \ln(t) + (x+1)^2]dt \\ &= \left[x^2 \cdot (t \cdot \ln^2(t) - 2t \cdot \ln(t) + 2t) + 2x \cdot (x+1) \cdot (t \cdot (\ln(t) - t) + (x+1)^2 \cdot t\right]_{e^{-\frac{1}{x}}}^{1} \\ &= \left[t \cdot (x^2 \cdot \ln^2(t) + 2x \cdot \ln(t) + x^2 + 1)\right]_{e^{-\frac{1}{x}}}^{1} \\ &= x^2 \cdot (1 - e^{-\frac{1}{x}}) + 1, \end{aligned}$$

$$
\begin{aligned}
Opt^2(x) &= \left(1-e^{-\frac{1}{x}}\right)\cdot x^2 + e^{-\frac{1}{x}}\cdot\left[\frac{1-(1-e^{-\frac{1}{x}})\cdot x}{e^{-\frac{1}{x}}}\right]^2 \\
&= \frac{e^{-\frac{1}{x}}\cdot(1-e^{-\frac{1}{x}})\cdot x^2 + [1-(1-e^{-\frac{1}{x}})\cdot x]^2}{e^{-\frac{1}{x}}} \\
&= e^{\frac{1}{x}}\cdot\left[(x^2-2x)\cdot(1-e^{-\frac{1}{x}})+1\right].
\end{aligned}
$$

Let

$$C^2(x) = \frac{Greedy^2(x)}{Opt^2(x)}$$

and

$$C^2 = \max_{1\le x} C^2(x).$$

For $x \geq 1$ the maximum value of $C^2(x)$ is obtained approximately at $x \approx 1.2612$ and its value is $C^2 \approx 1.7281$. Hence $C^2_{Greedy} \geq C^2 \approx 1.7281$.

Now we give a similar proof to improve the lower bound. The process of events is similar with the difference that here we keep the cost of the off-line algorithm fixed and equal to 1 instead of keeping the volume fixed and equal to 1 at the end of each step. In this proof we use the same notations as in the first proof. Consider the off-line algorithm at the end of step t when assigning a job with weight x to machine t. According to the invariant constraint we have.

$$(1-t)\cdot x^2 + t\cdot h^2 = Opt^2(x) = 1.$$

Where h is the weight of the sand jobs on machines $[0,t]$ at the end of step t. We define h as a function of t. The above equation gives

$$h(t) = \sqrt{\frac{1-(1-t)\cdot x^2}{t}}.$$

At the end of step t

$$f(t) = \frac{(1-t)\cdot x + t\cdot h(t) - F(t)}{t} + x. \tag{46}$$

We have $f(t) = -\frac{dF(t)}{dt}$ and we get

$$-\frac{dF(t)}{dt} = \frac{(1-t)\cdot x + t\cdot h(t) - F(t)}{t} + x.$$

Now we have the following first order differential equation

$$
\begin{aligned}
-t\cdot\frac{dF(t)}{dt} + F(t) - t\cdot h(t) - x &= 0 \\
F(1) &= 0
\end{aligned}
$$

The solution to this equation is not simple and was calculated using a computer program, which gave the following result. For $x \geq 1$ the maximum value of $C^2(x)$ is obtained approximately at $x \approx 1.3888$ and its value is $C^2 \approx 1.7906$. Hence $C^2_{Greedy} \geq C^2 \approx 1.7906$, which completes the proof.

B.6 Proof of Theorem 4.4

Let $m \to \infty$. As in the proof of Theorem 4.3 we consider the machines as points in the interval $(0, 1]$, each machine is represented by a point $t \in (0, 1]$, and the load of the machines is represented as a function $f(t)$ in that interval. Let $0 < \alpha < 1$. We consider the following sequence. First infinitesimally small jobs of total volume 1 arrive. Next, jobs of total volume $(1-\alpha)$ depart. Finally unit jobs of total volume 1 arrive. Consider the arrival of the infinitesimally small jobs. Algorithm A assigns these jobs, w.l.o.g we assume that machines $(0, \ldots, 1]$ are in non increasing order of load (the off-line algorithm assigns these jobs evenly on all the machines). Let $s \leq \alpha$ be maximal such that machines $(s, \ldots, 1]$ are assigned jobs of total volume $(1-\alpha)$. Then jobs of total volume $(1-\alpha)$ depart from machines $(s, \ldots, 1]$ (in the off-line algorithm these jobs depart evenly from all the machines). We denote by $x \leq \alpha$ the fraction of machines with assigned jobs of total height greater then 1. Next the unit jobs of total volume 1 arrive. The best Greedy can do is to assign jobs of total volume $(1-\alpha)$ evenly to machines $(\alpha, \ldots, 1]$ and then to assign jobs of total volume α to the α least loaded machines, which are composed of machines $(x, \ldots, \alpha]$, each machine with jobs of total height less then 1 and w.l.o.g to machines $(\alpha, \alpha+x]$, each machine with jobs of total height 1 (the off-line algorithm assigns these jobs evenly to all the machines). Let A and Opt be the costs of algorithm A and the off-line algorithms respectively.

$$
\begin{aligned}
A^p &\geq x \cdot 2^p + \alpha \left[\frac{\alpha + (\alpha - x)}{\alpha}\right]^p \\
&= x \cdot 2^p + \alpha \left(2 - \frac{x}{\alpha}\right)^p
\end{aligned}
$$

where the first term from left represents the cost of machines $(\alpha, \ldots, \alpha + x]$ and the other term is a lower bound for the cost of machines $(0, \ldots, \alpha]$.

$$
Opt^p = (1+\alpha)^p.
$$

Hence

$$
\begin{aligned}
C_A^p &\geq \left(\frac{A}{Opt}\right)^p && (47)\\
&\geq \frac{x \cdot 2^p + \alpha \left(2 - \frac{x}{\alpha}\right)^p}{(1+\alpha)^p}. && (48)
\end{aligned}
$$

We choose $\alpha = \frac{1}{p}$. We consider two cases. In both cases we use the inequality $e^{-x} \geq 1 - x$. For $x \geq \frac{\alpha}{p} = \frac{1}{p^2}$ we obtain

$$
C_A^p \geq \frac{\alpha}{p} \cdot \frac{2^p}{(1+\alpha)^p} = \frac{1}{p^2} \cdot \frac{2^p}{(1+\frac{1}{p})^p} = 2^p \frac{e^{-2\ln p}}{(1+\frac{1}{p})^p}.
$$

Hence

$$
C_A \geq 2 \frac{e^{-2\frac{\ln p}{p}}}{1+\frac{1}{p}} \geq 2 \frac{1 - 2\frac{\ln p}{p}}{1 + \frac{1}{p}} = 2 - O\left(\frac{\ln p}{p}\right).
$$

For $x < \frac{\alpha}{p} = \frac{1}{p^2}$ we obtain

$$C_A^p \geq \alpha \cdot \frac{(2-\frac{1}{p})^p}{(1+\alpha)^p} = \frac{1}{p} \cdot \frac{(2-\frac{1}{p})^p}{(1+\frac{1}{p})^p} = e^{-\ln p} \cdot \frac{(2-\frac{1}{p})^p}{(1+\frac{1}{p})^p}.$$

Hence

$$C_A \geq e^{-\frac{\ln p}{p}} \cdot \frac{2-\frac{1}{p}}{1+\frac{1}{p}} \geq (1 - \frac{\ln p}{p}) \cdot \frac{2-\frac{1}{p}}{1+\frac{1}{p}} = 2 - O\left(\frac{\ln p}{p}\right).$$

This completes the proof.