Load Balancing in the L_p Norm

Baruch Awerbuch[*]
Dept. of Computer Science
Johns Hopkins University
Baltimore, MD 21218
baruch@cs.jhu.edu

Yossi Azar[†]
Dept. of Computer Science
Tel-Aviv University
Tel-Aviv 69978, Israel
azar@math.tau.ac.il

Edward F. Grove[‡]
Dept. of Computer Science
Duke University
Durham, NC 27708–0129
efg@cs.duke.edu

Ming-Yang Kao[§]
Dept. of Computer Science
Duke University
Durham, NC 27708–0129
kao@cs.duke.edu

P. Krishnan[¶]
Computing Systems Res. Lab.
AT&T Bell Laboratories
Holmdel, NJ 07733–3030
pk@research.att.com

Jeffrey Scott Vitter[‖]
Dept. of Computer Science
Duke University
Durham, NC 27708–0129
jsv@cs.duke.edu

Abstract

In the load balancing problem, there is a set of servers, and jobs arrive sequentially. Each job can be run on some subset of the servers, and must be assigned to one of them in an online fashion. Traditionally, the assignment of jobs to servers is measured by the L_∞ norm; in other words, an assignment of jobs to servers is quantified by the maximum load assigned to any server. In this measure the performance of the greedy load balancing algorithm may be a logarithmic factor higher than optimal [3]. In many applications, the L_∞ norm is not a suitable way to measure how well the jobs are balanced. If each job sees a delay that is proportional to the number of jobs on its server, then the average *delay among all jobs is proportional to the sum of the squares of the numbers of jobs assigned to the servers. Minimizing the average delay is equivalent to minimizing the Euclidean (or L_2) norm. For any fixed p, $1 \leq p < \infty$, we show that the greedy algorithm performs within a constant factor of optimal with respect to the L_p norm. The constant grows linearly with p but does not depend on the size of the problem, i.e., the number of servers and jobs.*

[*] Also affiliated with MIT Lab. for Computer Science. Supported by ARPA/Army contract DABT63-93-C-0038, NSF contract 9114440-CCR, and DARPA contract N00014-J-92-1799.

[†] Supported in part by Allon Fellowship and by the Israel Science Foundation.

[‡] Supported in part by ARO grant DAAH04–93–G–0076.

[§] Supported in part by NSF grant CCR–9101385.

[¶] Supported in part by an IBM Fellowship, by NSF grant CCR–9007851, by ARO grant DAAH04–93–G–0076, and by AFOSR grant F49620–94–1–0217. This work was done while the author was visiting Duke University from Brown University.

[‖] Supported in part by NSF grant CCR–9007851 and by ARO grant DAAH04–93–G–0076.

1 Introduction

In the *load balancing problem*, jobs arrive sequentially. There is a set of n servers. Each job has an associated subset of the servers, called its *permissible* servers, on which it may run. Each job has a load that varies according to the job and to the server to which the job is assigned. Each job is assigned to one of its permissible servers in an online manner. The goal is to assign the jobs so as to spread the load as evenly as possible among the servers. The greedy load balancing algorithm, which we call Greedy, assigns each job to the permissible server so as to minimize the L_p norm of the loads of the servers created by assigning this job to the servers in the current state. In the case where all jobs have an equal load on every permissible server, Greedy puts each job on the permissible server that has the fewest jobs currently assigned, breaking ties arbitrarily.

We use the standard definition of *competitiveness* to analyze online algorithms. An algorithm A is said to be C-competitive if for every sequence σ of incoming jobs,

$$Cost_A(\sigma) \leq C \cdot Cost_{\mathrm{OPT}}(\sigma),$$

where OPT is the optimal offline algorithm, and $Cost_X(\sigma)$ is the cost of running X on σ.

Online load balancing has been considered by many researchers [7, 8, 9, 11, 6, 10]. Azar et al. [3, 4, 5] studied the problem of load balancing, motivated by the cellular phone system. Customers arrive and wish to be connected to a server. The goal is to minimize the maximum number of customers assigned to any one server. That is, $Cost_X(\sigma)$ is the maximum number of customers assigned to any server by algorithm

X on input sequence σ. In [3], the authors assume that customers do not disconnect, and show that the greedy algorithm is optimal with a competitive ratio of $\Theta(\log n)$. For the general case that the load due to a job depends upon the server to which it is assigned a competitive ratio of $\Theta(\log n)$ is achieved by somewhat more complicated algorithm [1]. The work in [4] and [5] deals with the case in which customers are allowed to disconnect.

But is the maximum load the right cost to minimize? This measure focuses on the worst server, and ignores how well the remaining servers are balanced. Consider an algorithm χ that assigns x_i jobs to server i, for $1 \le i \le n$. Let X be a column vector with $X^{\mathrm{T}} = (x_1, \dots, x_n)$. (We use X^{T} to denote the transpose of vector or matrix X.) The L_p norm and L_∞ norm of X are

$$|X|_p = \left(\sum_{1 \le i \le n} |x_i|^p \right)^{1/p} \quad \text{and} \quad |X|_\infty = \max_{1 \le i \le n} \{|x_i|\}.$$

The L_2 norm is the Euclidean norm, which measures the length of the vector X in Euclidean space. Note also that $(|X|_2)^2 = X^{\mathrm{T}} X$.

If we assume that each job sees a delay in service that is proportional to the number of jobs that are assigned to its server, then by minimizing the sum of squares (equivalently, by minimizing the L_2 norm) we minimize the average delay of the jobs in the system. The difference between this approach and traditional load balancing is that we try to minimize the *average* delay rather than the *maximum* delay. Our main result is the following theorem:

Theorem 1.1 *The greedy load balancing algorithm* GREEDY *is $O(p)$-competitive in the L_p norm, and any deterministic algorithm must be $\Omega(p)$-competitive.*

Note that the competitive ratio does not depend on the number of servers or the numbers of jobs, i.e., it is fixed for fixed p.

Our techniques and results can be extended to the case where customers are allowed to disconnect. Here the goal is to minimize the L_p norm of the vector of size nT of the load of each server on each unit of time (where T is the duration of the whole process). Using the techniques in [5] we can get a constant ($O(p)$) competitive algorithm for the L_p norm assuming that the duration of each job is known once it appears in the system. If the duration is unknown until the job departs it appears that by using some of the techniques of [2] one could achieve similar results by allowing reroutings.

In Section 2, we examine the case in which all jobs have a load of 1 on every permissible server. We start by considering how optimal adversaries behave. We classify jobs according to how they are processed by GREEDY. Given any particular assignment of jobs to servers by GREEDY, we determine how the adversary minimizes its cost. We then bound the cost of GREEDY in terms of the number of jobs assigned to each server by an optimal adversary, and reduce the problem to bounding the norm of a certain matrix. We get a bound of $2p$ on the competitive ratio in the L_p norm. In Section 3 we show that this is optimal to within a constant factor.

We then generalize the problem in Section 4 to allow for jobs whose load depends upon the server to which they are assigned. We switch to a more algebraic style of analysis, and we get a ratio of $1 + \sqrt{2}$ for the L_2 norm, and $cp + O(\log p)$ for general L_p norms, where $c \approx 1.77$ is the solution to the equation $c \ln c = 1$.

2 All Jobs Have Equal Load

In this section we examine the case in which all jobs have the same load on every permissible server. We relate this problem to finding the norm of a certain matrix. This is a rather intuitive approach, and is particularly interesting because the matrix also shows up in [12]. In the Section 4, we will generalize the problem and get stronger results, but the proofs will be primarily algebraic in nature.

2.1 Partitioning the Jobs

We start by analyzing the structure of a request sequence that gives rise to a particular output of GREEDY. Consider the output of GREEDY when run on σ. Let the servers be $S = \{1, \dots, n\}$, where server i is assigned to at least as many jobs as server $i+1$. For some of these servers it may be that no job is assigned. For each server i, we build a tower of height

$$h_i = \#\text{ of jobs assigned to server } i.$$

We know $h_i \ge h_{i+1}$. Each unit of height in a tower corresponds to a job. The higher the unit, the later the corresponding job was assigned.

Example 2.1 Let c be the highest job in the first tower. Let us first assume that $h_1 > h_2 + 1$. Then c must have only server 1 in its set of permissible servers. If any other server were permissible, GREEDY would have assigned c to the other server. Thus the adversary must also have assigned c to server 1. If $h_1 = h_2 + 1$ and $h_2 > h_3$, then c can have servers 1 and 2 as permissible servers, but no others. $\square$

Remark 2.1 *Let a job c be at a height h in some tower. Then the permissible servers for c must be a subset of $\{i : h_i \geq h - 1\}$.*

An adversary can reorder the jobs in the input sequence so that they come in order of height. Then GREEDY assigns the jobs to the same servers as before. In this new ordering, the permissible servers for a job of height h may include all servers of height at least $h - 1$. Since the adversary chooses among permissible servers of minimum load, the adversary can still keep the assignment of GREEDY the same. Let $t = |\{j : h_j = h\}|$. The adversary can use at most t servers besides the servers $\{1, \ldots, t\}$ to serve these jobs at height h, since these t jobs can be assigned to at most t different servers. We can rearrange the numbering so that the other servers the adversary uses for the jobs at height h have the smallest numbers out of all servers of height $h - 1$. Let

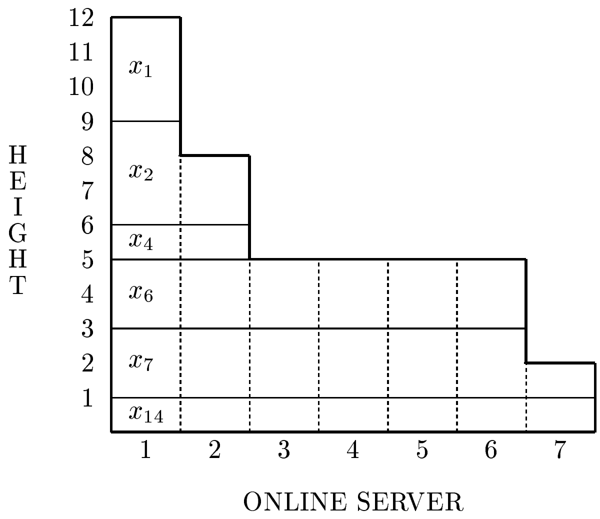
$$x_i = \#\text{ of jobs with permissible servers } \{1, \ldots, i\}.$$

Remark 2.2 *Let a job c be at a height h in some tower. Then we may assume without loss of generality that the permissible servers for c are exactly*

$$\{i : h_i \geq h - 1 \text{ and } i \leq 2 \cdot |\{j : h_j = h\}|\}.$$

Note that if $x_i > 0$ then $x_i \geq i/2$.

Example 2.2 Suppose that we run GREEDY with $n = 15$ servers, producing an assignment in which the servers receive jobs as follows: 0, 5, 0, 5, 0, 12, 0, 8, 0, 5, 0, 2, 5, 0, 0. We reorder the servers as in the preceding discussion, and we get $x_1 = 3$, $x_2 = 5$, $x_4 = 2$, $x_6 = 12$, $x_7 = 13$, and $x_{14} = 7$. Other values of x_i are 0. Note that the jobs counted by x_4 are not counted in x_6. The adversary cannot serve those jobs with servers 5 or 6 by the way we ordered the servers. Similarly, the adversary cannot assign the jobs counted by x_{14} to server 15.



$\square$

2.2 Adversary Assignment

We now consider how an adversary would minimize its cost subject to a particular set of values $\{x_i\}$. We give the adversary the additional power to assign jobs fractionally. For example, the adversary can assign a single job that has permissible servers $\{1, 2\}$ to be half on server 1 and half on server 2. Any bound on the competitive ratio using this stronger adversary applies to the original problem. Consider an assignment (possibly fractional) by the adversary of jobs to permissible servers that minimizes its cost. Let

$$a_i = \#\text{ of jobs assigned by the adversary to server } i.$$

Lemma 2.1 *For $1 \leq i \leq n - 1$, we have $a_i \geq a_{i+1}$.*

Proof: By Remark 2.2, any job assigned to server $i+1$ can also be assigned to server i. If a_i were smaller than a_{i+1}, the adversary could shift part of a job from server $i + 1$ to server i, reducing its cost. $\square$

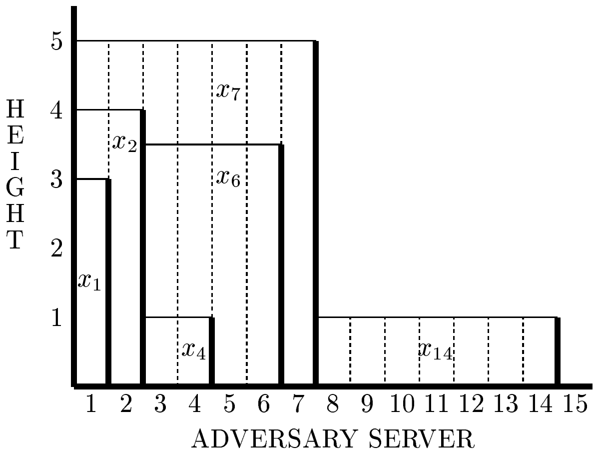
The minimum cost assignment is actually very intuitive in terms of a physical analogy: We build an infinitely high barrier at 0. For each j up to n, we build a barrier at j of height

$$w_j = \min_{1 \leq i \leq j} \left\{ \frac{x_i + x_{i+1} + \cdots + x_j}{j + 1 - i} \right\}.$$

These barriers break up the interval $[0, n]$ into n bins. Then, for j ranging from 1 to n, we pour x_j units of water into the jth bin. Water can pour over into lower-numbered bins when it overflows the barriers. If I is the value of i where w_j achieves its minimum value, the water from x_j flows to evenly top off bins I

through j. By assigning the x_j jobs according to where the water ends up, the adversary attains a minimum cost.

Example 2.3 Let us consider the same values of x_i as in Example 2.2. We show the barriers in the figure below with thick lines. The areas bounded by thick lines and thin lines show where the water ends up. The jobs counted by x_7 contribute 1 to servers 1 and 2, 1.5 to servers 3 through 6, and 5 to server 7.



□

Lemma 2.2 *The adversary's minimum cost assignment of jobs to servers is given by*

$$a_j = \max_{j\le k\le n} \min_{1\le i\le j} \left\{ \frac{x_i + x_{i+1} + \cdots + x_k}{k+1-i} \right\}.$$

Proof: We have $\sum a_j = \sum x_j$ by the water analogy. Let K be the values of k that maximize the above term. For the sake of contradiction, consider any j for which a_j is smaller. We then have

$$a_j < \min_{1\le i\le j} \left\{ \frac{x_i + x_{i+1} + \cdots + x_K}{K+1-i} \right\} \le \frac{x_j + \cdots + x_K}{K+1-j}.$$

Each server from j to K is assigned at most a_j jobs, so some part of some job from the $x_j + \cdots + x_K$ jobs must be assigned to a server s with $s < j$. But then we could move part of a job on server s to server j and reduce the cost, which contradicts the minimality of the adversary's assignment. □

2.3 The Greedy Assignment

Now we change our perspective. Given a set of values $\{a_j\}$ describing how the adversary assigned its servers, how badly can Greedy perform? Basically, for fixed $\{a_j\}$ the adversary's power is to set $\{x_j\}$ so as to maximize the cost of Greedy.

Lemma 2.3 Greedy *assigns at most* $2\sum_{k=j}^{n} x_k/k$ *jobs to server* j.

Proof: Consider the jobs counted by x_k. Let h be the height of a highest such job, and let $t = |\{j : h_j = h\}|$. These highest t jobs contribute 1 to the height of each server from 1 to t. The remaining $x_k - t$ jobs are evenly assigned by Greedy among the servers $\{1, \ldots, k\}$ and contribute an additional $(x_k - t)/k$ to the height of each server from 1 to k. For any $k \ge j$, the jobs counted by x_k contribute at most $2x_k/k$ to the height of server j, since if $x_k > 0$ then $t \ge k/2$ by Remark 2.2. □

In the rest of this section, rather than bounding the behavior of Greedy directly, we will bound the behavior described in Lemma 2.3. Let

$$r_j = 2\sum_{j\le k\le n} \frac{x_k}{k} \quad \text{and} \quad R^{\mathrm{T}} = (r_1, \ldots, r_n).$$

Remark 2.3 *It suffices to bound* $(\sum r_j^p)/(\sum a_j^p)$.

Let us allow the adversary to set $x_j = a_j$. (This might violate Remark 2.2, but that just means that we are giving the adversary the extra power to use a set of a_j values that are not consistent.) This setting is in fact the worst case. Note that $a_n \le x_n$, since the only jobs the adversary may assign to station n are those counted in x_n. In order to maximize $\sum r_i^2$, it is clearly best to minimize x_n. So we set $x_n = a_n$. Once we know $x_n = a_n$, it follows similarly that x_{n-1} should be set to a_{n-1} to maximize the sum, and inductively $x_i = a_i$ for all i.

Remark 2.4 *It suffices to bound* $(\sum r_j^p)/(\sum x_j^p)$.

We define an upper triangular matrix G so that $2GX = R$. Let $M = G^{\mathrm{T}}G$.

$$G(i,j) \quad = \quad \begin{cases} 1/j & \text{for } 1 \le i \le j \le n, \\ 0 & \text{for } 1 \le j < i \le n; \end{cases}$$

$$M(i,j) \quad = \quad \frac{1}{\max\{i,j\}}.$$

For example, when $n = 4$, we have

$$G = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 0 & 1/2 & 1/3 & 1/4 \\ 0 & 0 & 1/3 & 1/4 \\ 0 & 0 & 0 & 1/4 \end{pmatrix};$$

$$M = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/2 & 1/3 & 1/4 \\ 1/3 & 1/3 & 1/3 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}.$$

For a general L_p norm, we have an upper bound on the competitive ratio of GREEDY of $2\,||G||_p$, where $||G||_p = \sup\{|GX|_p/|X|_p : X > 0\}$. Note also that for any vector Y of positive reals, the adversary can force a ratio approaching $|GY|_p/|Y|_p$ by setting $x_i = \lfloor s\,y_i \rfloor$, and letting s approach ∞. In this case, GREEDY puts about $\sum_{k=j}^{n} s\,y_k/k$ jobs onto server j.

Remark 2.5 *The competitive ratio of* GREEDY *lies between* $||G||_p$ *and* $2\,||G||_p$.

Yao et al. [12] show independently that $||G||_p \le p$ (in their Corollary to Lemma 5.7). This proves that GREEDY is $2p$-competitive when all jobs have the same load on all permissible servers. We improve this bound on the competitive ratio to about $1.77p$ in Section 4.2.

Theorem 2.1 GREEDY*is* $\Theta(p)$*-competitive when all jobs have the same load on all permissible servers.*

2.4 Analysis of Euclidean Norm

Here is an alternate analysis of the performance of GREEDY in the L_2 norm. Since M is symmetric, we can express it as $Q^{\mathrm{T}}DQ$ where D is a diagonal matrix containing the eigenvalues of M, and Q is a matrix with $Q^{-1} = Q^{\mathrm{T}}$. Call λ the largest eigenvalue of M. Then it suffices to bound

$$
\begin{aligned}
\frac{(|R|_2)^2}{(|X|_2)^2} &= \frac{(|2GX|_2)^2}{(|X|_2)^2} \\
&= 4\frac{(GX)^{\mathrm{T}}(GX)}{X^{\mathrm{T}}X} \\
&= 4\frac{X^{\mathrm{T}}G^{\mathrm{T}}GX}{X^{\mathrm{T}}X} \\
&= 4\frac{X^{\mathrm{T}}Q^{\mathrm{T}}DQX}{X^{\mathrm{T}}Q^{\mathrm{T}}QX} \\
&= 4\frac{Y^{\mathrm{T}}DY}{Y^{\mathrm{T}}Y} \\
&\le 4\lambda,
\end{aligned}
$$

where $Y = QX$.

Remark 2.6 GREEDY *is* $2\sqrt{\lambda}$*-competitive in the Euclidean norm.*

In order to bound the competitive ratio of GREEDY, we wish to bound the eigenvalues of M. Consider any possible eigenvalue $\lambda \ge 4$. The determinant of $M - \lambda \cdot I$ should be 0. We will do elimination on $M - \lambda \cdot I$ to get a lower triangular matrix with negative values on the diagonal. That implies the determinant is non-zero, and contradicts the possibility that λ was an eigenvalue. This proves that the eigenvalues of M are at most 4.

We zero the columns (above the diagonal) from n down to 2. When we zero the ith column, we add some amount to every element in the initial $(i-1)\times(i-1)$ submatrix. Let t_i be the total added to each element of the $i \times i$ submatrix before we zero out the ith column ($t_n = 0$). Notice that the same amount is added to each element of the submatrix.

Each of the first $i-1$ entries in column or row i is currently equal to $t_i + 1/i$. The diagonal entry is $d_i = 1/i - \lambda + t_i$. We add $f_i = (-1/d_i)\cdot(t_i + 1/i)$ times the ith row to each row above it in the matrix. This clears the ith column above the diagonal. In so doing, we add $f_i \cdot (t_i + 1/i)$ to each element of the $(i-1)\times(i-1)$ initial submatrix. We get the recurrence

$$t_{i-1} = t_i - \frac{1}{d_i}\left(t_i + \frac{1}{i}\right)^2.$$

Lemma 2.4 *We have* $0 \le t_i < 1/i$ *and* $d_i < 0$, *for all* $i \ge 1$.

Proof: By induction, with the base case that $t_n = 0$. Recall we are assuming that $\lambda \ge 4$. We get

$$
\begin{aligned}
d_i &= \frac{1}{i} - \lambda + t_i < \frac{2}{i} - \lambda; \\
-\frac{1}{d_i} &\le \frac{1}{4 - \frac{2}{i}} < \frac{1}{4\left(1 - \frac{1}{i}\right)}; \\
t_{i-1} &< \frac{1}{i} + \frac{1}{4\left(1 - \frac{1}{i}\right)}\left(\frac{1}{i} + \frac{1}{i}\right)^2 \\
&= \frac{1}{i} + \frac{1}{i(i-1)} = \frac{1}{i-1}; \\
d_i &= \frac{1}{i} - \lambda + t_i < 1 - 4 + 1 < 0.
\end{aligned}
$$

We get $t_{i-1} \ge 0$ once we note $d_i < 0$. □

The matrix has non-zero determinant, and λ cannot be an eigenvalue. Thus the eigenvalues of M are at most 4, and GREEDY is 4-competitive in the L_2 norm.

3 Lower Bound

It is easy to get a lower bound of $\Omega(p)$ in the L_p norm against any deterministic algorithm for the case of all jobs having equal load.. Let the number of servers be $n = 2^k$, for some $k \ge 1$. The adversary proceeds in $k = \lg n$ phases. Initially, all 2^k servers are *active*. In each phase, the adversary matches the servers into pairs. For each pair of servers (a, b), the adversary presents one job whose permissible servers are a and b to the algorithm. The adversary assigns its job in opposition to the algorithm. If the algorithm assigns the job to a, then the adversary assigns it to b, and vice versa. The servers assigned jobs by the algorithm (half of the servers from the start of the phase) remain active for the next phase.

Given 2^k servers initially, the adversary places one job on each of $2^k - 1$ servers, and the algorithm will place a total of i jobs on each of 2^{k-1-i} servers, for $1 \leq i \leq k-1$, and k jobs on one server. The competitive ratio is

$$\left(\frac{k^p + \sum_{i=1}^{k-1} i^p 2^{k-1-i}}{2^k - 1} \right)^{1/p} \sim \frac{p \lg e}{e} = \Theta(p).$$

4 Generalized Load Balancing

We have been assuming that the load on any server is the number of jobs assigned to it. We now look at a generalized version of load balancing where each job has a *load vector* associated with it. When a job is assigned to a server, the *load* of the server increases by the amount specified by the corresponding coordinate of the load vector.

We will show that an obvious extension of GREEDY has a competitive ratio of $O(p)$ for any L_p norm, and in particular we get a ratio of $1+\sqrt{2}$ for the Euclidean (L_2) norm.

Each job j is represented by its "load vector" $\vec{r}(j) = (r_1(j), r_2(j), \ldots, r_n(j))$, where $r_i(j) \geq 0$. Let $\ell_i(j)$ denote the load on server i after we have already assigned jobs 1 through j. Assigning job j to server i increases the load on that server by $r_i(j)$, in other words:

$$\ell_k(j) = \begin{cases} \ell_k(j-1) + r_k(j) & \text{if } k = i; \\ \ell_k(j-1) & \text{otherwise.} \end{cases}$$

Let $Y(j) = (\ell_1(j), \ldots \ell_n(j))$ be the load vector of the server after we have already assigned jobs 1 through j. Consider a sequence of jobs defined by $\sigma = (\vec{r}(1), \vec{r}(2), \ldots, \vec{r}(t))$. Denote by $\ell_i^*(j)$ the load on server i achieved by the optimal algorithm $\mathcal{A}^*$ after assigning jobs 1 through j in σ and $Y^*(j)$ as the load vector of the servers. From now on we omit the parenthesis "(j)" for $j = t$; for example, ℓ_i^* denotes $\ell_i^*(t)$. We measure the performance of the online algorithm by the supremum over all possible sequences of $|Y|_p / |Y^*|_p$. We denote by $J(i)$ and $J^*(i)$ the set of jobs that were assigned by the online and optimal algorithms to server i, respectively.

How can we assign a job with a varying load in a greedy fashion? If we are trying to minimize the L_p norm, then we minimize the increase in the pth power of the load. When job j arrives we compute weights to the servers,

$$\text{Increase}(j) = (\ell_i(j-1) + r_i(j))^p - \ell_i^p(j-1)$$

and assign the job to a server with minimum increase.

4.1 The case $p = 2$

Theorem 4.1 GREEDY *is* $1+\sqrt{2}$ *competitive with respect to the* L_2 *norm.*

Proof: For a fixed j let i' be the server to which job j was assigned by the online algorithm. Similarly, let i^* be the server to which job j was assigned by the optimal algorithm. We have

$$\begin{aligned} & \ell_{i'}^2(j) - \ell_{i'}^2(j-1) \\ & \quad = (\ell_{i'}(j-1) + r_{i'}(j))^2 - \ell_{i'}^2(j-1) \\ & \quad \leq (\ell_{i^*}(j-1) + r_{i^*}(j))^2 - \ell_{i^*}^2(j-1) \\ & \quad = 2\ell_{i^*}(j-1) r_{i^*}(j) + r_{i^*}^2(j) \\ & \quad \leq 2\ell_{i^*} r_{i^*}(j) + r_{i^*}^2(j), \end{aligned}$$

where the first inequality follows from the definition of the algorithm. We sum all the above inequalities for all j and classifying them according to the server's indices, $J(i)$ and $J^*(i)$. This yields

$$\begin{aligned} & \sum_i \sum_{j \in J(i)} \ell_i^2(j) - \ell_i^2(j-1) \\ & \quad \leq \sum_i \sum_{j \in J^*(i)} (2\ell_i r_i(j) + r_i^2(j)). \end{aligned} \tag{1}$$

The sum on the left-hand side of (1) telescopes for each i. Also

$$\sum_{j \in J^*(i)} r_i(j) = \ell_i^*$$

and thus

$$\sum_{j \in J^*(i)} r_i^2(j) \leq \ell_i^{*2}.$$

Substituting these bounds into (1), we get

$$\begin{aligned} \sum_i \ell_i^2 & \leq \sum_i 2\ell_i \ell_i^* + \ell_i^{*2} \\ & \leq 2\sqrt{\sum_i \ell_i^2 \sum_i \ell_i^{*2}} + \sum_i \ell_i^{*2}. \end{aligned}$$

p	c	x	$1.77p$
2	1.47	4.52	3.54
3	1.67	6.64	5.31
4	1.76	8.61	7.08
5	1.80	10.51	8.85
10	1.84	19.72	17.70
50	1.80	91.13	88.50
100	1.78	179.70	177.00
500	1.77	885.96	885.00

Table 1: Bounds on the competitive ratio x implied by (6). The values of c and x were derived using a C program. The column for $1.77p$ is given for help in comparsion with the experimentally determined value for x.

The last inequality follows from the Cauchy-Schwartz inequality. Let us denote the ratio of the 2-norms by

$$x = \sqrt{\frac{\sum_i \ell_i^2}{\sum_i \ell_i^{*2}}}.$$

We can divide the above inequality by $\sum_i \ell_i^{*2}$ to get

$$x^2 \leq 2x + 1$$

and hence

$$x \leq 1 + \sqrt{2}.$$

□

4.2 The general case $p > 2$

Theorem 4.2 GREEDY *is $\Theta(p)$-competitive with respect to the L_p norm.*

As p grows, our bound on the competitive ratio is $cp + O(\log p)$, where $c \approx 1.77$ is the solution to the equation $c \ln c = 1$, as indicated in Table 1. It should be noted that the competitive ratio can never exceed that of the L^∞ norm, which is $\Theta(\log n)$.

Proof: As before, for a fixed j, let i' be the server to which job j was assigned by the online algorithms. Similarly, let i^* be the server to which job j was assigned by the optimal algorithm. We have

$$\begin{aligned} &\ell_{i'}^p(j) - \ell_{i'}^p(j-1) \\ &\quad = (\ell_{i'}(j-1) + r_{i'}(j))^p - \ell_{i'}^p(j-1) \\ &\quad \leq (\ell_{i^*}(j-1) + r_{i^*}(j))^p - \ell_{i^*}^p(j-1) \\ &\quad \leq (\ell_{i^*} + r_{i^*}(j))^p - \ell_{i^*}^p \\ &\quad \leq p(\ell_{i^*} + r_{i^*}(j))^{p-1} r_{i^*}(j). \end{aligned} \qquad (2)$$

The first inequality follows from the greedy nature of the algorithm. The second inequality follows the fact that $\ell_{i^*}(j-1) \leq \ell_{i^*}(t) = \ell_{i^*}$. The third inequality makes use of the derivative px^{p-1} of the function x^p at $x = \ell_{i^*} + r_{i^*}(j)$.

Lemma 4.1 *We can bound the term $(\ell_{i^*} + r_{i^*}(j))^{p-1}$ in (2) by*

$$c\ell_{i^*}^{p-1} + \left(r_{i^*}(j)\left(\frac{p-1}{\ln c} + 1\right)\right)^{p-1}, \qquad (3)$$

for any $c > 1$.

Proof: The second term of (3) clearly upper bounds $(\ell_{i^*} + r_{i^*}(j))^{p-1}$ when $\ell_{i^*} \leq r_{i^*}(j)(p-1)/\ln c$. We now show that the first term of (3) is an upper bound on $(\ell_{i^*} + r_{i^*}(j))^{p-1}$ for the other case, namely, when $\ell_{i^*} > r_{i^*}(j)(p-1)/\ln c$. In that case, we have

$$\begin{aligned} \frac{r_{i^*}(j)(p-1)}{\ell_{i^*}} &< \ln c; \\ \exp\left(\frac{r_{i^*}(j)(p-1)}{\ell_{i^*}}\right) &< c; \\ \left(1 + \frac{r_{i^*}(j)}{\ell_{i^*}}\right)^{p-1} &< c; \\ (\ell_{i^*} + r_{i^*}(j))^{p-1} &< c\ell_{i^*}^{p-1}. \end{aligned}$$

□

We substitute the upper bound (3) into (2) and get

$$\begin{aligned} &\ell_{i'}^p(j) - \ell_{i'}^p(j-1) \\ &\quad \leq p(\ell_{i^*} + r_{i^*}(j))^{p-1} r_{i^*}(j) \\ &\quad \leq p\left(c\ell_{i^*}^{p-1} + \left(r_{i^*}(j)\left(\frac{p-1}{\ln c} + 1\right)\right)^{p-1}\right) r_{i^*}(j) \\ &\quad \leq cp\ell_{i^*}^{p-1} r_{i^*}(j) + p\left(\frac{p-1}{\ln c} + 1\right)^{p-1} r_{i^*}^p(j). \end{aligned}$$

We sum the above inequality for all j and classify them according to the server's indices $J(i)$ and $J^*(i)$. This yields

$$\sum_i \sum_{j \in J(i)} \ell_i^p(j) - \ell_i^p(j-1) \leq$$
$$p\left(\frac{p-1}{\ln c} + 1\right)^{p-1} r_i^p(j) + \sum_i \sum_{j \in J^*(i)} cp\ell_i^{p-1} r_i(j). \qquad (4)$$

The left-hand side of (4) telescopes to give

$$\sum_i \ell_i^p.$$

On the right-hand size we have

$$\sum_{j\in J^*(i)} r_i(j) = \ell_i^*,$$

and thus for $p \geq 1$ we have

$$\sum_{j\in J^*(i)} r_i^p(j) \leq \ell_i^{*p}.$$

Substituting these bounds into (4), we get

$$\begin{aligned}\sum_i \ell_i^p &\leq cp\sum_i \ell_i^{p-1}\ell_i^* + p\left(\frac{p-1}{\ln c}+1\right)^{p-1}\sum_i \ell_i^{*p}\\ &\leq cp\left(\sum_i \ell_i^p\right)^{(p-1)/p}\left(\sum_i \ell_i^{*p}\right)^{1/p}\\ &\quad + p\left(\frac{p-1}{\ln c}+1\right)^{p-1}\sum_i \ell_i^{*p}, \qquad (5)\end{aligned}$$

where (5) follows from Holder's inequality:

$$\sum_i a_i^\alpha b_i^\beta \leq \left(\sum_i a_i\right)^\alpha \left(\sum_i b_i\right)^\beta,$$

for $\alpha+\beta = 1$. We use $a_i = \ell_i^p$, $b_i = \ell_i^{*p}$, $\alpha = (p-1)/p$, and $\beta = 1/p$.

Let us define x to be the competitive ratio of the greedy online algorithm, that is,

$$x = \frac{(\sum_i \ell_i^p)^{1/p}}{(\sum_i \ell_i^{*p})^{1/p}}.$$

Dividing (5) by $\sum_i \ell_i^{*p}$ and expressing the result in terms of x, we get

$$\begin{aligned}x^p &\leq cpx^{p-1} + p\left(\frac{p-1}{\ln c}+1\right)^{p-1};\\ x &\leq cp + p\left(\frac{\frac{p-1}{\ln c}+1}{x}\right)^{p-1}. \qquad (6)\end{aligned}$$

It is easy to see for large enough c that we get $x = \Theta(p)$. □

To get more detailed information on the best bound on x implied by (6), we consider the corresponding recurrence

$$x = cp + p\left(\frac{\frac{p-1}{\ln c}+1}{x}\right)^{p-1}, \qquad (7)$$

for all $c > 1$. Table 1 shows the bounds on the competitive ratio x implied by (6), where the choice of c is optimized for each p.

As p gets larger, the optimal value of c for use in (7) converges to the solution of the equation $c\ln c = 1$, which is $c \approx 1.77$.

Theorem 4.3 *The minimal solution x to (7) is $x = c^*p + \Theta(\log p)$, when $c = c^* \approx 1.77$ is the solution to the equation $c\ln c = 1$.*

Proof: It isn't hard to show from (7) that for $c = c^*$ we have $x(c^*) = cp + \Theta(\log p)$. To get a particular bound on the lower order terms we can bootstrap by combining the bound with (7) to get

$$\begin{aligned}x(c^*) = c^*p + p\left(\frac{\frac{p-1}{\ln c^*}+1}{x}\right)^{p-1} &= c^*p + \Theta(\log p);\\ \left(\frac{\frac{p-1}{\ln c^*}+1}{x}\right)^{p-1} &= \Theta\left(\frac{\log p}{p}\right);\\ \exp\left((p-1)\ln\left(\frac{\frac{p-1}{\ln c^*}+1}{x}\right)\right) &= \Theta\left(\frac{\log p}{p}\right).\end{aligned}$$

Taking logarithms and dividing by $p-1$, we get

$$\ln\left(\frac{\frac{p-1}{\ln c^*}+1}{x}\right) = -\frac{\ln p}{p} + \frac{\ln\ln p}{p} + O\left(\frac{1}{p}\right). \quad (8)$$

Let $x(c^*) = c^*p + g(p) = c^*p(1 + g(p)/c^*p)$. Using the fact that $c^*\ln c^* = 1$, we have

$$\ln\left(\frac{\frac{p-1}{\ln c^*}+1}{x}\right) = \ln\left(1 - \frac{g(p)}{c^*p} + O\left(\frac{1}{p}\right)\right).$$

By (8) and the fact that $\ln(1+y) = y + O(y^2)$, for small y, we have

$$\begin{aligned}\frac{g(p)}{c^*p} &= \frac{\ln p}{p} - \frac{\ln\ln p}{p} + O\left(\frac{1}{p}\right);\\ g(p) &= c^*\ln p - c^*\ln\ln p + O(1).\end{aligned}$$

Therefore, by the definition of $g(p)$,

$$x(c^*) = c^*p + c^*\ln p - c^*\ln\ln p + O(1).$$

The rest of the proof consists of showing in a similar manner that $x \leq c^*p + \ln p$ implies that $c = c^* + O((\log p)/p)$, which implies that $x = c^*p + \Theta(\log p)$. □

5 Conclusions

In this paper we have analyzed the performance of the greedy algorithm for the online load balancing problem, where instead of examining the maximum load on a server, we measure the L_p norm of the load assignment. For the case $p = 2$, the L_2 norm is related to the average delay seen by the jobs on the servers. We have shown that the greedy algorithm has a bounded competitive ratio with respect to the L_p norm, for any fixed $p \geq 1$. In particular, the greedy algorithm is $(1+\sqrt{2})$-competitive under the L_2 norm, and is $(cp + O(\log p))$-competitive with respect to the L_p norm, for $p \geq 1$, where $c \approx 1.77$ is the solution to the equation $c \ln c = 1$. These results are optimal to within a constant factor.

References

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

[2] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 321–327, January 1994.

[3] Y. Azar, J. Naor, and R. Rom, "The Competitiveness of On-Line Assignments," *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms* (January 1992).

[4] Y. Azar, A. Y. Broder, and A. R. Karlin, "On-line Load Balancing," *Proceedings of the 33rd Symposium on Foundations of Computer Science* (October 1992), 218–225.

[5] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. R. Pruhs, and O. Waarts, "Online Load Balancing of Temporary Tasks," *Proceedings of the 1993 Workshop on Algorithms and Data Structures* (August 1993).

[6] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 51-58, 1992.

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

[8] R.L. Graham, E.L. Lawler, J.K Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[9] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[10] S. Phillips and J. Westbrook. Online load balancing and network flow. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 402–411, 1993.

[11] D. Shmoys, J. Wein, and D.P. Williamson. Scheduling parallel machines on-line. In *Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science*, pages 131–140, 1991.

[12] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," in these *Proceedings of the 36th Symposium on Foundations of Computer Science* (October 1995).