Balanced Allocations

Yossi Azar* Andrei Z. Broder† Anna R. Karlin‡ Eli Upfal§

Abstract

Suppose that we sequentially place n balls into n boxes by putting each ball into a randomly chosen box. It is well known that when we are done, the fullest box has with high probability $(1+o(1))\ln n/\ln\ln n$ balls in it. Suppose instead, that for each ball we choose two boxes at random and place the ball into the one which is less full at the time of placement. We show that with high probability, the fullest box contains only $\ln\ln n/\ln 2+O(1)$ balls – exponentially less than before. Furthermore, we show that a similar gap exists in the infinite process, where at each step one ball, chosen uniformly at random, is deleted, and one ball is added in the manner above. We discuss consequences of this and related theorems for dynamic resource allocation, hashing, and on-line load balancing.

1 Introduction

Suppose that we sequentially place n balls into n boxes by putting each ball into a randomly chosen box. Properties of this random allocation process have been extensively studied in the probability and statistics literature.

*Department of Computer Science, Tel Aviv University, Israel. This research was supported in part by the Alon Fellowship and the Israel Science Foundation administered by the Israel Academy of Sciences. E-mail: azar@math.tau.ac.il.

†Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA. E-mail: broder@src.dec.com.

‡Department of Computer Science, University of Washington, Seattle, WA 98195, USA. E-mail: karlin@cs.washington.edu

§IBM Almaden Research Center, San Jose, CA 95120, USA, and Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel. Work at the Weizmann Institute supported in part by the Norman D. Cohen Professorial Chair of Computer Science. E-mail: eli@wisdom.weizmann.ac.il.

(See e.g. [20, 17].) One of the classical results in this area is that when the process has terminated, the fullest box has, with high probability[1], $(1 + o(1)) \ln n / \ln \ln n$ balls in it[2].

Consider a variant of the process above whereby each ball comes with d possible destinations, chosen independently and uniformly at random. (Hence the d destinations are not necessarily distinct.) The ball is placed in the least full box among the d possible locations. Surprisingly, even for $d = 2$, when the process terminates the fullest box has only $\ln \ln n / \ln 2 + O(1)$ balls in it. Thus, this apparently minor change in the random allocation process results in an exponential decrease in the maximum occupancy per location. The analysis of this process is summarized as follows

Theorem 1 *Suppose that m balls are sequentially placed into n boxes. Each ball is placed in the least full box, at the time of the placement, among d boxes, $d \geq 2$, chosen independently and uniformly at random. Then after all the balls are placed*

- *With high probability, as $n \to \infty$ and $m \geq n$, the number of balls in the fullest box is $(1 + o(1)) \ln \ln n / \ln d + \Theta(m/n)$.*
- *In particular, with high probability, as $n \to \infty$ and $m = n$, the number of balls in the fullest box is $\ln \ln n / \ln d + \Theta(1)$.*
- *Any other on-line strategy that places each ball into one of d randomly chosen boxes, results in stochastically more balls[3] in the fullest box.*

It is also interesting to study the infinite version of the random allocation process. There, at each step a ball is chosen uniformly at random and removed from the system, and a new ball appears. The new ball comes with d possible destinations, chosen independently at random, and it is placed into the least full box among these d possible destinations.

The analysis of the case $d = 1$ in this infinite stochastic process is simple since the location of any ball does not depend on the locations of other balls in the system. Thus, for $d = 1$, in the stationary distribution, with high probability the fullest box has $\Theta(\log n / \log \log n)$ balls. The analysis of the case $d \geq 2$ is significantly harder, since the locations of the current n balls might depend on the locations of balls that are no longer in the system. We prove that when $d \geq 2$, in the stationary distribution, the fullest box has

[1] We say that an event $\mathcal{E}$ occurs with high probability if $\mathbf{Pr}(\mathcal{E}) = 1 - o(1)$.

[2] G. Gonnet [16] has proven a more accurate result, $\Gamma^{-1}(n) - 3/2 + o(1)$.

[3] By this we mean that for any other strategy and any k the probability that the number of balls in the fullest box is greater than k is at least the probability that the number of balls in the fullest box is greater than k under the greedy strategy. See Corollary 8.

$\ln\ln n/\ln d + O(1)$ balls, with high probability. Thus, the same exponential gap holds in the infinite process. Theorem 2 is proven in section 4.

Theorem 2 *Consider the infinite process with $d \geq 2$, starting at time 0 in an arbitrary state. There is a constant c such that for any fixed $T > cn^2 \log\log n$, the fullest box at time T contains, with high probability, less than $\ln\ln n/\ln d + O(1)$ balls. Thus, in the stationary distribution, with high probability, no box contains more than $\ln\ln n/\ln d + O(1)$ balls.*

Karp, Luby, and Meyer auf der Heide [18] were the first to notice a dramatic improvement when switching from one hash function to two in the context of PRAM simulations. In fact it is possible to use a result from [18] to derive a weaker form of our static upper bound. (For details see [7].)

A preliminary version of this paper has appeared in [7]. Subsequently, Adler, Chakrabarti, Mitzenmacher, and Rasmussen [1] analyzed parallel implementation of the balanced allocation mechanism and obtained interesting communication vs. load tradeoffs.

A related question was considered by Broder, Frieze, Lund, Phillips, and Reingold [10]. In their model the set of choices is such that there is a placement that results in maximum load equal to one. The question they analyse is what is the expected maximum load under a random order of insertion under the greedy strategy.

More recent results, based on the balanced allocation paradigm have appeared in [23, 24, 25, 12].

1.1 Applications

Our results have a number of interesting applications to computing problems. We elaborate here on three of them:

Dynamic Resource Allocation. Consider a scenario in which a user or a process has to choose on-line between a number of identical resources (choosing a server to use among the servers in a network; choosing a disk to store a directory; etc.). To find the least loaded resource, users may check the load on all resources before placing their requests. This process is expensive, since it requires sending an interrupt to each of the resources. A second approach is to send the task to a random resource. This approach has minimum overhead, but if all users follow it, the difference in load between different servers will vary by up to a logarithmic factor. Our analysis suggests a more efficient solution. If each user samples the load of two resources and sends his request to the least loaded, the total overhead is small, and the load on the n resources varies by only a $O(\log\log n)$ factor.

Hashing. The efficiency of a hashing technique is measured by two parameters: the expected and the maximum access time. Our approach suggests a simple hashing technique, similar to hashing with chaining. We call it *2-way chaining*. It has $O(1)$ expected, and $O(\log \log n)$ maximum access time. We use two random hash functions. The two hash functions define two possible entries in the table for each key. The key is inserted to the least full location, at the time of the insertion. Keys in each entry of the table are stored in a linked list. Assume that n keys are sequentially inserted by this process into a table of size n. As shown in Section 5, the expected insertion and look-up time is $O(1)$, and our analysis summarized above, immediately implies that with high probability the maximum access time is $\ln \ln n / \ln 2 + O(1)$, versus the $\Theta(\log n / \log \log n)$ time when only one random hash function is used.

An advantage of our scheme over some other known techniques for reducing worst case behavior of hashing (e.g. [14, 13, 11]) is that it uses only two hash functions, it is easy to parallelize, and does not involve re-hashing of data. Other commonly used schemes partition the available memory into multiple tables, and use a different hash function in each table. For example, the Fredman, Komlos, Szemeredi scheme for perfect hashing [14], uses up to n different hash functions to get $O(1)$ worst case access time (not on-line however), and the algorithm of Broder and Karlin [11] uses $O(\log \log n)$ hash functions to achieve $O(\log \log n)$ maximum access time, on-line, but using re-hashings.

Karp, Luby, and Meyer auf der Heide [18] studied the use of two hash functions in the context of PRAM simulations. Other PRAM simulations using multiple hash functions were developed and analyzed in [21].

Competitive On-line Load Balancing. Consider the following on-line load balancing problem: We are given a set of n servers and a sequence of arrivals and departures of tasks. Each task comes with a list of servers on which it can be executed. The load balancing algorithm has to assign each task to a server on-line, with no information on future arrivals and departures of tasks. The goal of the algorithm is to minimize the maximum load on any server. The quality of an on-line algorithm is measured by the *competitive ratio*: the ratio between the maximum load it achieves and the maximum load achieved by the optimal off-line algorithm that knows the whole sequence in advance. This load balancing problem models for example, communication in heterogeneous networks containing workstations, I/O devices, etc. Servers correspond to communication channels and tasks to requests for communication links between devices. A network controller must coordinate the channels so that no channel is too heavily loaded.

On-line load balancing has been studied extensively against worst-case adversaries [9, 6, 5, 3, 8, 4]. For permanent tasks (tasks that arrive but never depart), Azar, Naor and Rom [9] showed that the competitive ratio of the greedy algorithm is $\log n$ and that no algorithm can do better. For temporary tasks (tasks that depart at unpredictable times), the works of Azar, Broder and Karlin [6] and Azar, Kalyanasundaram, Plotkin, Pruhs and Waarts [8] show that there is an algorithm with competitive ratio $\Theta(\sqrt{n})$ and that no algorithm can do better.

It is interesting to compare these high competitive ratios, obtained from inputs generated by an adversary, to the competitive ratio against randomly generated inputs. Our results show that under reasonable probabilistic assumptions the competitive ratios for both permanent and temporary tasks are significantly better. In the case of permanent tasks, if the set of servers on which a task can be executed is a small set (that is, constant size ≥ 2) chosen at random, the competitive ratio decreases from $\Theta(\log n)$ to $\Theta(\log \log n)$. In the case of temporary tasks, if we further assume that at each time step a randomly chosen existent task is replaced by a new task, then at any fixed time the ratio between the maximum online load and the maximum offline load is $\Theta(\log \log n)$ with high probability. Further details are presented in Section 6.

2 Definitions and Notation

We consider two stochastic processes: the finite process and the infinite process.

The Finite Process. There are n boxes, initially empty, and m balls. Each ball is allowed to go into $d \geq 1$ boxes chosen independently and uniformly at random. The balls arrive one by one, and a *placement algorithm* must decide on-line (that is, without knowing what choices are available to future balls) in which box to put each ball as it comes. Decisions are irrevocable. We will subsequently refer to this setup as a (m, n, d)-problem.

The Infinite Process. There are n boxes, initially containing n balls in an arbitrary state. (For example, all the balls could be in one box.) At each step, one random ball is removed, and one new ball is added; the new ball is allowed to go into $d \geq 1$ boxes chosen independently and uniformly at random. Once again, a placement algorithm must decide on-line (that is, without knowing what choices are available to future balls and without knowing which ball will be removed at any future time) in which box to put each arriving ball. Decisions are irrevocable.

We use the following notations for the random variables associated with a placement algorithm A. Note that the state at time t refers to the state immediately after the placement of the t'th ball.

$\lambda_j^A(t)$ called the *load* of box j, is the number of balls in box j at time t, resulting from algorithm A.

$\nu_k^A(t)$ is the number of boxes that have load k at time t.

$\nu_{\geq k}^A(t)$ is the number of boxes that have load $\geq k$ at time t, that is $\nu_{\geq k}^A(t) = \sum_{i \geq k} \nu_i^A(t)$.

h_t^A called the *height* of ball t (= the ball that arrives at time t), is the number of balls at time t in the box where ball t is placed. In other words, the first ball to be placed in a particular box has height 1, the second ball has height 2, etc.

$\mu_k^A(t)$ is the number of balls that have height k at time t.

$\mu_{\geq k}^A(t)$ is the number of balls that have height $\geq k$ at time t, i.e. $\mu_{\geq k}^A(t) = \sum_{i \geq k} \mu_i^A(t)$.

We omit the superscript A when it is clear which algorithm we are considering. Constants were chosen for convenience, and we made no attempts to optimize them.

Algorithm GREEDY assigns ball j to the box that has the lowest load among the d random choices that j has. We use the superscript G for GREEDY.

The basic intuition behind the proofs that follow is simple: Let $p_i = \mu_{\geq i}/n$. Since the available choices for each ball are independent, and $\nu_{\geq i} \leq \mu_{\geq i}$, we roughly have ("on average" and disregarding conditioning) $p_{i+1} \leq p_i^d$, which implies a doubly exponential decrease in p_i, once $\mu_{\geq i} < n/2$. Of course the truth is that $\mu_{\geq i+1}$ is strongly dependent on $\mu_{\geq i}$ and a rather complex machinery is required to construct a correct proof.

3 The Finite Process

We use the notation $B(n,p)$ to denote a binomially distributed random variable with parameters n and p, and start with the following standard lemma, whose proof is omitted.

Lemma 3 *Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables with values in an arbitrary domain, and let $Y_1, Y_2, \ldots, Y_n$ be a sequence of binary random variables, with the property that $Y_i = Y_i(X_1, \ldots, X_i)$. If*

$$\mathbf{Pr}(Y_i = 1 \mid X_1, \ldots, X_{i-1}) \leq p,$$

then

$$\mathbf{Pr}(\sum Y_i \geq k) \leq \mathbf{Pr}(B(n,p) \geq k)$$

and similarly if

$$\mathbf{Pr}(Y_i = 1 \mid X_1, \ldots, X_{i-1}) \geq p,$$

then

$$\mathbf{Pr}(\sum Y_i \leq k) \leq \mathbf{Pr}(B(n,p) \leq k)$$

□

We now turn to the analysis of the finite process. In what follows, we omit the argument t, when $t = m$, that is, when the process terminates. In the interest of a clearer exposition, we start with the case $m = n$, although the general case (Theorem 9) subsumes it.

Theorem 4 *The maximum load achieved by the* GREEDY *algorithm on a random (n, n, d)-problem is less than $\ln \ln n / \ln d + O(1)$ with high probability.*

Proof: Since the d choices for a ball are independent, we have

$$\mathbf{Pr}(h_t \geq i+1 \mid \nu_{\geq i}(t-1)) = \frac{(\nu_{\geq i}(t-1))^d}{n^d}.$$

Let $\mathcal{E}_i$ be the event that $\nu_{\geq i}(n) \leq \beta_i$ where β_i will be exposed later. (Clearly, $\mathcal{E}_i$ implies that $\nu_{\geq i}(t) \leq \beta_i$ for $t = 1, \ldots, n$.) Now fix $i \geq 1$ and consider a series of binary random variables Y_t for $t = 2, \ldots, n$, where

$$Y_t = 1 \textbf{ iff } h_t \geq i+1 \textbf{ and } \nu_{\geq i}(t-1) \leq \beta_i.$$

(Y_t is 1 if the height of the ball t is $\geq i+1$ despite the fact that the number of boxes that have load $\geq i$ is less than β_i.)

Let ω_j represent the choices available to the j'th ball. Clearly

$$\mathbf{Pr}(Y_t = 1 \mid \omega_1, \ldots, \omega_{t-1}) \leq \frac{\beta_i^d}{n^d} \stackrel{def}{=} p_i.$$

Thus we can apply Lemma 3 to conclude that

$$\mathbf{Pr}(\sum Y_t \geq k) \leq \mathbf{Pr}(B(n,p_i) \geq k). \tag{1}$$

Observe that conditioned on $\mathcal{E}_i$, we have $\mu_{\geq i+1} = \sum Y_t$. Therefore

$$\mathbf{Pr}(\mu_{\geq i+1} \geq k \mid \mathcal{E}_i) = \mathbf{Pr}(\sum Y_t \geq k \mid \mathcal{E}_i) \leq \frac{\mathbf{Pr}(\sum Y_t \geq k)}{\mathbf{Pr}(\mathcal{E}_i)}. \tag{2}$$

Combining equations (1) and (2) we obtain that

$$\mathbf{Pr}(\nu_{\geq i+1} \geq k \mid \mathcal{E}_i) \leq \mathbf{Pr}(\mu_{\geq i+1} \geq k \mid \mathcal{E}_i) \leq \frac{\mathbf{Pr}(B(n,p_i) \geq k)}{\mathbf{Pr}(\mathcal{E}_i)}. \tag{3}$$

We can bound large deviations in the binomial distribution with the formula (see for instance [2], Appendix A.)

$$\mathbf{Pr}(B(n,p_i) \geq ep_in) \leq e^{-p_in}, \tag{4}$$

which inspires us to set

$$\beta_i = \begin{cases} n, & i = 1,2,\ldots,5; \\ \dfrac{n}{2e}, & i = 6; \\ \dfrac{e\beta_{i-1}^d}{n^{d-1}}, & i > 6. \end{cases}$$

With these choices $\mathcal{E}_{\geq 6} = \{\nu_6 \leq n/(2e)\}$ holds with certainty, and from (3) and (4), for $i \geq 6$

$$\mathbf{Pr}(\neg\mathcal{E}_{i+1} \mid \mathcal{E}_i) \leq \frac{1}{n^2\mathbf{Pr}(\mathcal{E}_i)},$$

provided that $p_in \geq 2\ln n$. Since

$$\mathbf{Pr}(\neg\mathcal{E}_{i+1}) \leq \mathbf{Pr}(\neg\mathcal{E}_{i+1} \mid \mathcal{E}_i)\mathbf{Pr}(\mathcal{E}_i) + \mathbf{Pr}(\neg\mathcal{E}_i),$$

it follows that for $p_in \geq 2\ln n$

$$\mathbf{Pr}(\neg\mathcal{E}_{i+1}) \leq \frac{1}{n^2} + \mathbf{Pr}(\neg\mathcal{E}_i). \tag{5}$$

To finish the proof let i^* be the smallest i such that $\beta_{i^*}^d/n^d \leq 2\ln n/n$. Notice that $i^* \leq \ln\ln n/\ln d + O(1)$ since

$$\beta_{i+6} = \frac{ne^{(d^i-1)/(d-1)}}{(2e)^{d^i}} \leq \frac{n}{2^{d^i}}.$$

As before

$$\mathbf{Pr}(\nu_{\geq i^*+1} \geq 6 \ln n \mid \mathcal{E}_{i^*}) \leq \frac{\mathbf{Pr}(B(n, 2 \ln n/n) \geq 6 \ln n)}{\mathbf{Pr}(\mathcal{E}_{i^*})} \leq \frac{1}{n^2 \mathbf{Pr}(\mathcal{E}_{i^*})},$$

and thus

$$\mathbf{Pr}(\nu_{\geq i^*+1} \geq 6 \ln n) \leq \frac{1}{n^2} + \mathbf{Pr}(\neg \mathcal{E}_{i^*}). \tag{6}$$

Finally

$$\begin{aligned} \mathbf{Pr}(\mu_{\geq i^*+2} \geq 1 \mid \ \nu_{\geq i^*+1} \leq 6 \ln n) &\leq \frac{\mathbf{Pr}(B(n, (6 \ln n/n)^d) \geq 1)}{\mathbf{Pr}(\nu_{\geq i^*+1} \leq 6 \ln n)} \\ &\leq \frac{n(6 \ln n/n)^d}{\mathbf{Pr}(\nu_{\geq i^*+1} \leq 6 \ln n)} \end{aligned}$$

by the Markov inequality, and thus

$$\mathbf{Pr}(\mu_{\geq i^*+2} \geq 1) \leq \frac{(6 \ln n)^d}{n^{d-1}} + \mathbf{Pr}(\nu_{\geq i^*+1} \geq 6 \ln n) \tag{7}$$

Combining equations (7), (6), and (5), we obtain that

$$\mathbf{Pr}(\nu_{\geq i^*+2} \geq 1) \leq \frac{(6 \ln n)^d}{n^{d-1}} + \frac{i^*+1}{n^2} = o(1),$$

which implies that with high probability the maximum load achieved by GREEDY is less than $i^* + 2 = \ln \ln n / \ln d + O(1)$. $\square$

We now prove a matching lower bound.

Theorem 5 *The maximum load achieved by the* GREEDY *algorithm on a random (n, n, d)-problem is at least $\ln \ln n / \ln d - O(1)$ with high probability.*

Proof: Let $\mathcal{F}_i$ be the event that $\nu_{\geq i}(n(1 - 1/2^i)) \geq \gamma_i$ where γ_i will be exposed later. For the time being, suffices to say that $\gamma_{i+1} < \gamma_i/2$. We want to compute $\mathbf{Pr}(\neg \mathcal{F}_{i+1} \mid \mathcal{F}_i)$. To this aim, for t in the range $R = \{n(1 - 1/2^i) + 1, \ldots, n(1 - 1/2^{i+1})\}$, let Z_t be defined by

$$Z_t = 1 \textbf{ iff } h_t = i + 1 \textbf{ or } \nu_{\geq i+1}(t-1) \geq \gamma_{i+1},$$

and observe that while $\nu_{\geq i+1}(t-1) < \gamma_{i+1}$, if $Z_t = 1$ then the box where the t'th ball is placed had load exactly i at time $t - 1$. This means that all the d choices that ball t had, pointed to boxes with load $\geq i$ and at least one choice pointed to a box with load exactly i.

Now let ω_j represent the choices available to the j'th ball. Using

$$\begin{aligned}\mathbf{Pr}(A \vee B \mid C) &= \mathbf{Pr}(A \wedge \bar{B} \mid C) + \mathbf{Pr}(B \mid C) \\ &= \mathbf{Pr}(A \mid \bar{B} \wedge C)\mathbf{Pr}(\bar{B} \mid C) + \mathbf{Pr}(B \mid C) \geq \mathbf{Pr}(A \mid \bar{B} \wedge C),\end{aligned}$$

and in view of the observation above, we derive that

$$\mathbf{Pr}(Z_t = 1 \mid \omega_1, \ldots, \omega_{t-1}, \mathcal{F}_i) \geq \left(\frac{\gamma_i}{n}\right)^d - \left(\frac{\gamma_{i+1}}{n}\right)^d \geq \frac{1}{2}\left(\frac{\gamma_i}{n}\right)^d \stackrel{def}{=} p_i. \quad (8)$$

Applying Lemma 3 we get

$$\mathbf{Pr}\Big(\sum_{t \in R} Z_t \leq k \;\Big|\; \mathcal{F}_i\Big) \leq \mathbf{Pr}(B(n/2^{i+1}, p_i) \leq k)$$

We now choose

$$\begin{aligned}\gamma_0 &= n; \\ \gamma_{i+1} &= \frac{\gamma_i^d}{2^{i+3} n^{d-1}} = \frac{n}{2^{i+3}}\left(\frac{\gamma_i}{n}\right)^d = \frac{1}{2}\frac{n}{2^{i+1}} p_i.\end{aligned}$$

Since $\mathbf{Pr}(B(N,p) < Np/2) < e^{-Np/8}$, (see for instance [2], Appendix A), it follows that

$$\mathbf{Pr}(B(n/2^{i+1}, p_i) \leq \gamma_{i+1}) = o(1/n^2) \quad (9)$$

provided that $p_i n/2^{i+1} \geq 17 \ln n$. Let i^* be the largest integer for which this holds. Clearly $i^* = \ln\ln n / \ln d - O(1)$.

Now observe that by the definition of $\mathcal{F}$ and Z_t, the event $\{\sum_{t \in R} Z_t \geq \gamma_{i+1}\}$ implies $\mathcal{F}_{i+1}$. Thus in view of (8) and (9)

$$\mathbf{Pr}(\neg\mathcal{F}_{i+1} \mid \mathcal{F}_i) \leq \mathbf{Pr}\Big(\sum_{t \in R} Z_t \leq \gamma_{i+1} \;\Big|\; \mathcal{F}_i\Big) = o(1/n^2),$$

and therefore

$$\begin{aligned}\mathbf{Pr}(\mathcal{F}_{i^*}) &\geq \mathbf{Pr}(\mathcal{F}_{i^*} \mid \mathcal{F}_{i^*-1}) \times \mathbf{Pr}(\mathcal{F}_{i^*-1} \mid \mathcal{F}_{i^*-2}) \\ &\quad \times \cdots \times \mathbf{Pr}(\mathcal{F}_1 \mid \mathcal{F}_0) \times \mathbf{Pr}(\mathcal{F}_0) \\ &\geq (1 - 1/n^2)^{i^*} = 1 - o(1/n)\end{aligned}$$

which completes the proof. □

We now turn to showing that the greedy algorithm is stochastically optimal under our model, that is, we assume that each ball has d destinations chosen uniformly at random, and all balls have equal weight. (The optimality is not preserved if either condition is violated.) It suffices to consider only deterministic algorithms since randomized algorithms can be considered as a distribution over deterministic algorithms.

We say that a vector $\bar{v} = (v_1, v_2, \ldots, v_n)$ *majorizes* a vector $\bar{u}$, written $\bar{v} \succeq \bar{u}$ if for $1 \leq i \leq n$, we have $\sum_{1 \leq j \leq i} v_{\pi(j)} \geq \sum_{1 \leq j \leq i} u_{\sigma(j)}$, where π and σ are permutations of $1, \ldots, n$ such that $v_{\pi(1)} \geq v_{\pi(2)} \geq \cdots \geq v_{\pi(n)}$ and $u_{\sigma(1)} \geq u_{\sigma(2)} \geq \cdots \geq u_{\sigma(n)}$.

Lemma 6 *Let $\bar{v}$ and $\bar{u}$ be two positive integer vectors such that $v_1 \geq v_2 \geq \cdots \geq v_n$ and $u_1 \geq u_2 \geq \cdots \geq u_n$. If $\bar{v} \succeq \bar{u}$ then also $\bar{v} + \bar{e}_i \succeq \bar{u} + \bar{e}_i$, where $\bar{e}_i$ is the i'th unit vector, that is $\bar{e}_{i,j} = \delta_{i,j}$.*

Proof: Let $S_j(\bar{x})$ be the sum of the j largest components of the vector $\bar{x}$. Notice first that for all j

$$S_j(\bar{x}) \leq S_j(\bar{x} + \bar{e}_i) \leq S_j(\bar{x}) + 1. \tag{10}$$

By hypothesis, for all j, we have $S_j(\bar{v}) \geq S_j(\bar{u})$. To prove the lemma we show that for all j, we also have $S_j(\bar{v} + \bar{e}_i) \geq S_j(\bar{u} + \bar{e}_i)$. Fix j. By equation (10) if $S_j(\bar{v}) > S_j(\bar{u})$ then $S_j(\bar{v} + \bar{e}_i) \geq S_j(\bar{u} + \bar{e}_i)$. Now assume $S_j(\bar{v}) = S_j(\bar{u})$. There are three cases to consider:

Case 1: $i \leq j$. Then

$$S_j(\bar{v} + \bar{e}_i) = S_j(\bar{v}) + 1 = S_j(\bar{u}) + 1 = S_j(\bar{u} + \bar{e}_i).$$

Case 2: $i > j$ and $u_j > u_i$. Since $u_j \geq u_i + 1$, it follows that $S_j(\bar{u}) = S_j(\bar{u} + \bar{e}_i)$ and therefore

$$S_j(\bar{v} + \bar{e}_i) \geq S_j(\bar{v}) = S_j(\bar{u}) = S_j(\bar{u} + \bar{e}_i).$$

Case 3: $i > j$ and $u_j = u_{j+1} = \cdots = u_i$. Observe first that since $S_{j-1}(\bar{v}) \geq S_{j-1}(\bar{u})$, $S_j(\bar{v}) = S_j(\bar{u})$, and $S_{j+1}(\bar{v}) \geq S_{j+1}(\bar{u})$, we have

$$v_j \leq u_j \qquad \text{and} \qquad v_{j+1} \geq u_{j+1}.$$

Hence

$$v_j \geq v_{j+1} \geq u_{j+1} = u_j \geq v_j.$$

We conclude that $v_j = u_j = v_{j+1} = u_{j+1}$, and thus $S_{j+1}(\bar{v}) = S_{j+1}(\bar{u})$. Repeating the argument, we obtain that

$$v_j = u_j = v_{j+1} = u_{j+1} = \cdots = v_i = u_i,$$

and therefore

$$S_j(\bar{v} + \bar{e}_i) = S_j(\bar{v}) + 1 = S_j(\bar{u}) + 1 = S_j(\bar{u} + \bar{e}_i).$$

□

Let Ω be the set of all possible n^d choices for each ball and Ω^t be the set of sequences of choices for the first t balls.

Theorem 7 *For any online deterministic algorithm A, and $t \geq 0$, there is 1-1 correspondence $f : \Omega^t \to \Omega^t$ such that for any $\omega_t \in \Omega^t$ the vector of box loads associated to* GREEDY *acting on ω_t, written*

$$\bar{\lambda}^G(\omega_t) = (\lambda_1^G(\omega_t), \lambda_2^G(\omega_t), \ldots, \lambda_n^G(\omega_t))$$

is majorized by the vector of box loads associated to A acting on $f(\omega_t)$, that is

$$\bar{\lambda}^G(\omega_t) \preceq \bar{\lambda}^A(f(\omega_t)).$$

Proof: To simplify notation we assume $d = 2$. The proof for larger d is analogous. The proof proceeds by induction on t, the length of the sequence. The base case ($t = 0$) is obvious. Assume the theorem valid for t and let f_t be the mapping on Ω^t. Fix a sequence $\omega_t \in \Omega^t$. It suffices to show that we can refine f_t to obtain a 1-1 correspondence for all possible 1-step extensions of ω_t. Without loss of generality, renumber the boxes such that

$$\lambda_1^G(\omega_t) \geq \lambda_2^G(\omega_t) \geq \cdots \geq \lambda_n^G(\omega_t),$$

and let π be a permutation of $1, \ldots, n$ such that

$$\lambda_{\pi(1)}^A(f_t(\omega_t)) \geq \lambda_{\pi(2)}^A(f_t(\omega_t)) \geq \cdots \geq \lambda_{\pi(n)}^A(f_t(\omega_t)).$$

Let (i, j) be two choices for the $t+1$ ball. For every i, j we define

$$f_{t+1}(\omega_t \diamond (i, j)) = f_t(\omega_t) \diamond (\pi(i), \pi(j))$$

where "$\diamond$" represents extension of sequences.

Clearly f_{t+1} is 1-1. We need to show that

$$\bar{\lambda}^G(\omega_t \diamond (i, j)) \preceq \bar{\lambda}^A(f_t(\omega_t) \diamond (\pi(i), \pi(j))).$$

Notice that when the sequence ω_t is extended by the step (i,j), for any algorithm, exactly one component of the vector $\bar{\lambda}(\omega_t)$ changes, namely either $\lambda_i(\omega_t)$ or $\lambda_j(\omega_t)$ increases by one. Assume that $i \geq j$; then

$$\begin{aligned}\bar{\lambda}^G(\omega_t \diamond (i,j)) &= \bar{\lambda}^G(\omega_t) + \bar{e}_i \\ &\preceq \bar{\lambda}^A(f_t(\omega_t)) + \bar{e}_{\pi(i)} \\ &\preceq \bar{\lambda}^A\Big(f_t(\omega_t) \diamond (\pi(i), \pi(j))\Big),\end{aligned}$$

where the first inequality follows from the Lemma 6 and the second is due to the fact that

$$\bar{\lambda}^A(f_t(\omega_t)) + \bar{e}_{\pi(i)} \preceq \bar{\lambda}^A(f_t(\omega_t)) + \bar{e}_{\pi(j)}.$$

□

Corollary 8 *For any fixed k and any t*

$$\mathbf{Pr}(\max_i \lambda_i^A(t) > k) \geq \mathbf{Pr}(\max_i \lambda_i^G(t) > k).$$

We are now ready to discuss the general case of the finite process.

Theorem 9 *The maximum load achieved by the* GREEDY *algorithm on a random (m,n,d)-problem, with $d \geq 2$ and $m \geq n$, is, with high probability, less than $(1+o(1))\ln\ln n/\ln d + O(m/n)$.*

Proof: We start by replaying the proof of Theorem 4, taking into account the fact that there are now m balls. So let $\mathcal{E}_i$ be the event that $\nu_{\geq i}(m) \leq \beta_i$, and define $p_i = \beta_i^d/n^d$. Following the proof of Theorem 4 we derive that

$$\mathbf{Pr}(\nu_{\geq i+1} \geq k \mid \mathcal{E}_i) \leq \frac{\mathbf{Pr}(B(m,p_i) \geq k)}{\mathbf{Pr}(\mathcal{E}_i)}.$$

Suppose that for some value x we set $\beta_x = n^2/(2em)$ and show that $\mathcal{E}_x$ holds with high probability, that is

$$\mathbf{Pr}\left(\nu_x \geq \frac{n^2}{2em}\right) = o(1). \tag{11}$$

Then

$$\beta_{i+x} = \frac{n}{2^{d^i}}\left(\frac{me}{n}\right)^{(d^i-1)/(d-1)-d^i} \leq \frac{n}{2^{d^i}},$$

and continuing as before, we obtain that

$$\mathbf{Pr}(\mu \geq x + \ln\ln n/\ln d + 2) = o(1).$$

It remains to show that x can be taken to be $O(m/n) + o(\ln\ln n/\ln d)$. First assume that $m/n \geq w(n)$ where $w(n)$ is an increasing function of n, but $w(n) = o(\ln\ln n/\ln d)$. Then we claim that we can take $x = \lceil em/n \rceil$.

Consider a placement algorithm, denoted R, that always puts a ball in the box corresponding to the first choice offered. This is entirely equivalent with the case $d = 1$, the classical occupancy problem. The load within a box under this process is a binomial random variable $B(m, 1/n)$, and therefore (via (4)), the probability that the load within a box exceeds em/n is bounded by $e^{-m/n}$. Now consider the height of the t'th ball, denoted h_t^R. The probability that the box into which the t'th ball is placed has load greater than em/n is less than $e^{-m/n}$ and therefore the expected number of balls of height $\geq em/n$ satisfies

$$\mathbf{E}(\mu^R_{\geq em/n}) \leq me^{-m/n}.$$

Hence by Markov's inequality

$$\mathbf{Pr}\Big(\mu^R_{\geq em/n} \geq \frac{n^2}{2em}\Big) \leq \frac{2em^2}{n^2}e^{-m/n} = o(1),$$

since $m/n \to \infty$.

We claim that Theorem 7 implies

$$\mathbf{Pr}\Big(\mu^G_{\geq k} \geq r\Big) \leq \mathbf{Pr}\Big(\mu^R_{\geq k} \geq r\Big). \tag{12}$$

Indeed, suppose that there is an outcome ω_t for which GREEDY has exactly i boxes with load greater than or equal to k. As in the proof of Theorem 7, renumber the boxes such that

$$\lambda_1^G(\omega_t) \geq \lambda_2^G(\omega_t) \geq \cdots \geq \lambda_n^G(\omega_t).$$

Let $f_t(\omega_t)$ be the corresponding outcome for algorithm R and let π be a permutation of $1, \ldots, n$ such that

$$\lambda^R_{\pi(1)}(f_t(\omega_t)) \geq \lambda^R_{\pi(2)}(f_t(\omega_t)) \geq \cdots \geq \lambda^R_{\pi(n)}(f_t(\omega_t)).$$

Then

$$\mu^G_{\geq k}(\omega_t) = \sum_{1 \leq j \leq i} (\lambda_j^G(\omega_t) - (k-1))$$

and

$$\mu^R_{\geq k}(f_t(\omega_t)) \geq \sum_{1 \leq j \leq i} (\lambda^R_{\pi(j)}(f_t(\omega_t)) - (k-1)).$$

But Theorem 7 implies that

$$\sum_{1 \leq j \leq i} \lambda^R_j(f(\omega_t)) \geq \sum_{1 \leq j \leq i} \lambda^G_j(\omega_t),$$

and by considering all outcomes we obtain equation (12).

Therefore,

$$\mathbf{Pr}\left(\mu^G_{\geq em/n} \geq \frac{n^2}{2em}\right) \leq \mathbf{Pr}\left(\mu^R_{\geq em/n} \geq \frac{n^2}{2em}\right),$$

and since

$$\mathbf{Pr}\left(\nu^G_{\geq em/n} \geq \frac{n^2}{2em}\right) \leq \mathbf{Pr}\left(\mu^G_{\geq em/n} \geq \frac{n^2}{2em}\right),$$

we have that for $x = \lceil em/n \rceil$ equation (11) is satisfied.

To remove the assumption $m/n \geq w(n)$, we can simply imagine that the number of balls is increased to $\max(m, nw(n))$. Then the corresponding value of x becomes $O(\max(m, nw(n))/n) = O(m/n) + o(\ln \ln n / \ln d)$. □

4 The Infinite Process

In this section we consider the infinite process. Analogously to Theorem 7 it is possible to show that the greedy algorithm minimizes the expected maximum load on any box. We analyze its performance below. The main theorem of this section is

Theorem 10 *Assume that the infinite process starts in an arbitrary state. Under* GREEDY, *with* $d \geq 2$, *there is a constant* c *such that for any fixed* $T \geq cn^2 \log \log n$,

$$\mathbf{Pr}(\exists j, \lambda_j(T) \geq \ln \ln n / \ln d + O(1)) = o(1).$$

Thus in the stationary distribution the maximum load is $\ln \ln n / \ln d + O(1)$ *with high probability.*

Proof: For simplicity of presentation we state and prove the results only for $d = 2$. The proof assumes that at time $T - cn^2 \log \log n$ the process is in an arbitrary state and therefore we can let $T = cn^2 \log \log n$ with no loss of generality.

By the definition of the process, the number of balls of height at least i cannot change by more than 1 in a time step, that is $|\mu_{\geq i}(t+1) - \mu_{\geq i}(t)| \leq 1$. The random variable $\mu_{\geq i}(t)$ can be viewed as a random walk on the integers l, $0 \leq l \leq n$. The proof is based on bounding the maximum values taken by the variables $\mu_{\geq i}(t)$ by studying the underlying process.

We define an integer i^* and a decreasing sequence α_i, for $200 \leq i \leq i^*+1$ as follows:

$$\begin{aligned}
\alpha_{200} &= \frac{n}{200}, \\
\alpha_i &= \frac{100\alpha_{i-1}^2}{n}, \qquad \text{for } i > 200 \text{ and } \alpha_{i-1} \geq \sqrt{n \log_2 n}, \\
\alpha_{i^*} &= 100 \log_2 n, \qquad i^* = \text{the smallest } i \text{ for which} \\
&\qquad\qquad\qquad\quad \alpha_{i-1} < \sqrt{n \log_2 n}, \\
\alpha_{i^*+1} &= 100.
\end{aligned}$$

Clearly $i^* \leq \ln\ln n / \ln 2 + O(1)$. For future reference, observe also that for $200 < i \leq i^* + 1$

$$\alpha_i \geq \frac{100\alpha_{i-1}^2}{n} \tag{13}$$

We also define an increasing sequence of times: $t_{200} = 0$ and $t_i = t_{i-1} + n^2$ for $i > 200$. Thus $t_{i^*+1} = O(n^2 \log\log n) = O(T)$.

Let $\{\mu_{\geq i}[t^-, t^+] \leq \alpha\}$ denote the event that $\mu_{\geq i}(t) \leq \alpha$ for all t, such that $t^- < t \leq t^+$, and similarly, let $\{\mu_{\geq i}[t^-, t^+] > \alpha\}$ denote the event that $\mu_{\geq i}(t) > \alpha$ for all t, such that $t^- < t \leq t^+$. We define the events C_i as follows:

$$\begin{aligned}
C_{200} &= \{\nu_{\geq 200}[t_{200}, T] \leq 2\alpha_{200}\} \equiv \{\nu_{\geq 200}[0, T] \leq n/100\}; \\
C_i &= \{\mu_{\geq i}[t_i, T] \leq 2\alpha_i\}, \qquad \text{for } i > 200.
\end{aligned}$$

Note that C_{200} always holds, and for $i > 200$, the event $C - i$ implies that $\nu_{\geq i}[t_i, T] \leq 2\alpha_i$.

We shall prove inductively that for all $i = 200, \ldots, i^* + 1$

$$\mathbf{Pr}(\neg C_i) \leq \frac{2i}{n^2}. \tag{14}$$

This implies that the event $\{\mu_{\geq i^*+1}[t_{i^*+1}, T] \leq 200\}$ occurs with probability $1 - o(1)$, and therefore with high probability, for every j, $\lambda_j(T) \leq i^* + 201 = \ln\ln n / \ln 2 + O(1)$ which completes the proof of the main part of the Theorem.

Finally, we show that in the stationary distribution

$$\mathbf{Pr}(\forall j, \lambda_j \leq \log\log n + O(1)) = 1 - o(1).$$

Indeed, let S be the set of states such that $\forall j, \lambda_j(t) \leq \log\log n + O(1)$. Let $s(t)$ be the state of the chain at time t. Then the previous observation implies that

$$\mathbf{Pr}(s(t+T) \notin S \mid s(t)) = o(1).$$

Let π be the stationary distribution; then

$$\sum_{i \notin S} \pi_i = \sum_j \mathbf{Pr}(s(t+T) \notin S \mid s(t) = j) \cdot \pi_j = \sum_j \pi_j o(1) = o(1),$$

which completes the proof of the Theorem assuming equation (14). To prove it, we show that conditioned on C_{i-1}:

a. With high probability $\mu_{\geq i}(t)$ becomes less than α_i before time t_i. (This is shown in Lemma 11.)

b. If $\mu_{\geq i}(t)$ becomes less than α_i at any time before T, then from then on until T, with high probability, it does not become larger than $2\alpha_i$. (This is shown in Lemma 12.)

The two facts above imply that if C_{i-1} holds, then with high probability $\mu_{\geq i}[t_i, T] \leq 2\alpha_i$, that is C_i holds as well.

Base Case: The base case is trivial since $\mathbf{Pr}(\neg C_{200}) = \mathbf{Pr}(\neg\{\nu_{\geq 200}[0,T] \leq n/100\}) = 0$.

Induction: Suppose that

$$\mathbf{Pr}(\neg C_{i-1}) \leq \frac{2(i-1)}{n^2}, \tag{15}$$

where $200 < i \leq i^* + 1$.

Let $s(t)$ be the state at time t. It is easy to verify the following bounds on the underlying transition probabilities. For any t,

$$\mathbf{Pr}(\mu_{\geq i}(t+1) > \mu_{\geq i}(t) \mid s(t)) \leq \left(\frac{\nu_{\geq (i-1)}(t)}{n}\right)^2 \leq \left(\frac{\mu_{\geq (i-1)}(t)}{n}\right)^2 \tag{16}$$

and

$$\mathbf{Pr}(\mu_{\geq i}(t+1) < \mu_{\geq i}(t) \mid s(t)) \geq \frac{\mu_{\geq i}(t)}{n}\left(1 - \left(\frac{\nu_{\geq (i-1)}(t)}{n}\right)^2\right) \geq \frac{\mu_{\geq i}(t)}{2n} \tag{17}$$

¿From equations (16) and (17) we obtain that the transition probabilities satisfy

$$\mathbf{Pr}(\mu_{\geq i}(t+1) > \mu_{\geq i}(t) \mid \mu_{\geq i-1}(t) \leq 2\alpha_{i-1}) \leq \left(\frac{2\alpha_{i-1}}{n}\right)^2 \stackrel{def}{=} q_i^+,$$

and

$$\mathbf{Pr}(\mu_{\geq i}(t+1) < \mu_{\geq i}(t) \mid \mu_{\geq i}(t) \geq \alpha_i) \geq \frac{\alpha_i}{2n} \stackrel{def}{=} q_i^-.$$

Thus in view of (13)

$$q_i^+ \leq \frac{\alpha_i}{25n}.$$

We define two new binary random variables for $0 < t \leq T$ as follows:

$$X_t = 1 \textbf{ iff } \mu_{\geq i}(t) > \mu_{\geq i}(t-1) \textbf{ and } \mu_{\geq i-1}(t-1) \leq 2\alpha_{i-1},$$

and

$$Y_t = 1 \textbf{ iff } \mu_{\geq i}(t) < \mu_{\geq i}(t-1) \textbf{ or } \mu_{\geq i}(t-1) \leq \alpha_i.$$

Clearly

$$\mathbf{Pr}(X_t = 1) \leq q_i^+ \qquad \text{and} \qquad \mathbf{Pr}(Y_t = 1) \geq q_i^- \tag{18}$$

We also define F_i to be the event

$$F_i \stackrel{def}{=} \{\exists t^* \in [t_{i-1}, t_i] \text{ s.t. } \mu_{\geq i}(t^*) \leq \alpha_i\},$$

thus $\neg F_i$ is the event

$$\neg F_i = \{\mu_{\geq i}[t_{i-1}, t_i] > \alpha_i\}.$$

Two lemmas are necessary in order to conclude that $\mathbf{Pr}(\neg C_i) \leq 2i/n^2$.

Lemma 11 *Under the inductive hypothesis*

$$\mathbf{Pr}(\neg F_i \mid C_{i-1}) \leq \frac{1}{n^2}.$$

Proof: Notice that conditioned on C_{i-1}, the sum $\sum_{t\in[t_{i-1},t_i]} X_t$ is the number of times $\mu_{\geq i}(t)$ increased in the interval $[t_{i-1}, t_i]$; similarly, if within this interval $\mu_{\geq i}$ did not become less than α_i, then $\sum_{t\in[t_{i-1},t_i]} Y_t$ equals the number of times $\mu_{\geq i}(t)$ decreased in this interval. We conclude that

$$\begin{aligned}\mathbf{Pr}(\neg F_i \mid C_{i-1}) &\leq \mathbf{Pr}\Big(\sum_{t\in[t_{i-1},t_i]} Y_t - \sum_{t\in[t_{i-1},t_i]} X_t \leq n \;\Big|\; C_{i-1}\Big) \\ &\leq \frac{1}{\mathbf{Pr}(C_{i-1})}\mathbf{Pr}\Big(\sum_{t\in[t_{i-1},t_i]} Y_t - \sum_{t\in[t_{i-1},t_i]} X_t \leq n\Big)\end{aligned}$$

In view of equation (18) and Lemma 3, Chernoff type bounds imply that for every $i \leq i^* + 1$

$$\begin{aligned}\mathbf{Pr}\Big(\sum_{t\in[t_{i-1},t_i]} X_t > 2n^2 q_i^+\Big) &\leq \mathbf{Pr}\Big(B(n^2, q_i^+) \geq 2n^2 q_i^+\Big) \\ &\leq e^{-\Omega(n^2 q_i^+)} = o(1/n^c),\end{aligned}$$

and

$$\begin{aligned}\mathbf{Pr}\Big(\sum_{t\in[t_{i-1},t_i]} Y_t < \tfrac{1}{2}n^2 q_i^-\Big) &\leq \mathbf{Pr}(B(n^2, q_i^-) \leq \tfrac{1}{2}n^2 q_i^-) \\ &\leq e^{-\Omega(n^2 q_i^-)} = o(1/n^c),\end{aligned}$$

for any constant c. On the other hand, in view of (13),

$$\frac{1}{2}n^2 q_i^- - 2n^2 q_i^+ \geq \frac{1}{4}n\alpha_i - \frac{2}{25}n\alpha_i \geq \frac{n\alpha_i}{10} \geq n,$$

and therefore we conclude that

$$\mathbf{Pr}(\neg F_i \mid C_{i-1}) \leq \frac{1}{n^c \mathbf{Pr}(C_{i-1})},$$

for any constant c. Taking $c = 3$ and using the inductive hypothesis on C_{i-1} (equation (15)) completes the proof. □

Lemma 12 *Under the inductive hypothesis*

$$\mathbf{Pr}(\neg C_i \mid C_{i-1}, F_i) \leq \frac{1}{n^2}.$$

Proof: Since $\mathbf{Pr}(A \mid B \wedge C) \leq \mathbf{Pr}(A \wedge B|C)$ we get that

$$\begin{aligned}&\mathbf{Pr}(\neg C_i \mid C_{i-1}, F_i) \\ &\quad\leq \mathbf{Pr}(\neg C_i \wedge F_i \mid C_{i-1}) \\ &\quad\leq \mathbf{Pr}(\exists t_1, t_2 \in [t_{i-1}, T] \\ &\qquad\qquad \text{s.t. } \mu_{\geq i}(t_1) = \alpha_i, \mu_{\geq i}(t_2) = 2\alpha_i, \mu_{\geq i}[t_1, t_2] \geq \alpha_i \mid C_{i-1}) \\ &\quad\leq \sum_{t_{i-1}\leq t_1 < t_2 \leq T} \mathbf{Pr}(\mu_{\geq i}(t_1) = \alpha_i, \mu_{\geq i}(t_2) = 2\alpha_i, \mu_{\geq i}[t_1, t_2] \geq \alpha_i \mid C_{i-1}) \\ &\quad\leq \sum_{t_{i-1}\leq t_1 < t_2 \leq T} \mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t - \sum_{t\in[t_1,t_2]} Y_t \geq \alpha_i \mid C_{i-1}\Big) \\ &\quad\leq \frac{1}{\mathbf{Pr}(C_{i-1})} \sum_{t_{i-1}\leq t_1 < t_2 \leq T} \mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t - \sum_{t\in[t_1,t_2]} Y_t \geq \alpha_i\Big).\end{aligned}$$

Fix t_1 and t_2 and let $\Delta = t_2 - t_1$. We now consider 4 cases.

A. $\Delta \le n$ and $i \le i^*$.

$$\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t \ge \alpha_i\Big) \le \binom{\Delta}{\alpha_i}(q^+)^{\alpha_i} \le \left(\frac{e\Delta}{\alpha_i}\cdot\frac{\alpha_i}{25n}\right)^{\alpha_i} \le n^{-100}.$$

B. $\Delta \le n\log n$ and $i = i^*+1$.

$$\begin{aligned}\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t \ge \alpha_{i^*+1}\Big) &\le \binom{\Delta}{\alpha_{i^*+1}}(q^+)^{\alpha_{i^*+1}} \le \left(\frac{e\Delta}{\alpha_{i^*+1}}\cdot\frac{4\alpha_{i^*}^2}{n^2}\right)^{\alpha_{i^*+1}} \\ &\le \left(\frac{en\log n}{100}\cdot\frac{4\cdot 100^2\log^2 n}{n^2}\right)^{100} \le n^{-100}.\end{aligned}$$

C. $\Delta \ge n$ and $i \le i^*$.
Using again large deviation bounds and the fact that $\alpha_i\Delta \ge 100\log n$ we obtain that

$$\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} Y_t \le \frac{1}{2}q^-\Delta\Big) \le e^{-q^-\Delta/8} = e^{-\alpha_i\Delta/(16n)} \le n^{-6.1}$$

and that

$$\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t \ge \frac{1}{2}q^-\Delta\Big) \le \left(\frac{2eq^+\Delta}{q^-\Delta}\right)^{q^-\Delta/2} \le \left(\frac{4e}{25}\right)^{\alpha_i\Delta/(4n)} \le n^{-25}.$$

D. $\Delta \ge n\log n$ and $i = i^*+1$.
The same proof as case **C** using the fact that $\alpha_{i^*+1}\Delta \ge 100\log n$.

Thus in all four cases,

$$\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t - \sum_{t\in[t_1,t_2]} Y_t \ge \alpha_i\Big) \le \frac{1}{n^{6.1}},$$

therefore, under the induction hypothesis,

$$\frac{1}{\mathbf{Pr}(C_{i-1})}\sum_{t_{i-1}\le t_1<t_2\le T}\mathbf{Pr}\Big(\sum_{t\in[t_1,t_2]} X_t - \sum_{t\in[t_1,t_2]} Y_t \ge \alpha_i\Big) \le \frac{2T^2}{n^{6.1}} \le \frac{1}{n^2}.$$

□

Returning to the proof of equation (14), by using the induction hypothesis, Lemmas 11 and 12, and the law of total probability, we can complete the induction as follows:

$$\begin{aligned}
\mathbf{Pr}(\neg C_i) &= \mathbf{Pr}(\neg C_i \mid C_{i-1}) \cdot \mathbf{Pr}(C_{i-1}) \\
&\quad + \mathbf{Pr}(\neg C_i \mid \neg C_{i-1}) \cdot \mathbf{Pr}(\neg C_{i-1}) && \text{Now apply IH} \\
&\le \mathbf{Pr}(\neg C_i \mid C_{i-1}) + 2(i-1)/n^2 \\
&= \mathbf{Pr}(\neg C_i \mid C_{i-1}, F_i) \cdot \mathbf{Pr}(F_i \mid C_{i-1}) && \text{Now apply Lemma 12} \\
&\quad + \mathbf{Pr}(\neg C_i \mid C_{i-1}, \neg F_i) \cdot \mathbf{Pr}(\neg F_i \mid C_{i-1}) && \text{Now apply Lemma 11} \\
&\quad + 2(i-1)/n^2 \\
&\le 1/n^2 + 1/n^2 + 2(i-1)/n^2 = 2i/n^2.
\end{aligned}$$

□

5 Hashing

We define a simple hashing algorithm, called *2-way chaining*, by analogy with the popular direct chaining method. We use two random hash functions. For each key, the two hash functions define two indices in a table. Each table location contains a pointer to a linked list. When a new key arrives, we compare the current length of the two lists associated to the key, and key is inserted at the end of the shortest list. (The direct chaining method corresponds to having only one associated random index.)

For searching, the two hash values are computed, and two linked lists are searched in alternate order. (That is, after checking the i'th element of the first list, we check the i'th element of the second list, then element $i+1$ of the first list, and so on.) When the shorter list is exhausted, we continue searching the longer list until it is exhausted as well. (In fact, if no deletions are allowed we can stop after checking only one more element in the longer list. For the analysis below, this is immaterial.)

Assume that n keys are sequentially inserted by this process to a table of size n. Theorem 1 analysis implies that with high probability the maximum access time, which is bounded by twice the length of the longest list, is $2 \ln n \ln n / \ln 2 + O(1)$, versus the $\Theta(\log n / \log \log n)$ time when one random hash function is used. More generally, if m keys are stored in the table with d hash functions, then the maximum access time under this scheme is $2(1 + o(1)) \ln \ln n / \ln d + \Theta(m/n)$.

Next we show that the average access time of 2-way chaining is no more than twice the average access time of the standard direct chaining method.

As customary, we discuss the average access time separately for successful searches and unsuccessful searches. The latter, denoted C'_G, is bounded by twice the expected cost of checking a list chosen uniformly at random. Therefore

$$C'_G(m,n) \leq 2 + \frac{2m}{n}.$$

For successful searches, the cost C_G, is given by

$$C_G(m,n) \leq \frac{2}{m} \sum_{1 \leq i \leq m} h_i = \frac{2}{m} \sum_{1 \leq j \leq n} \binom{\lambda_j + 1}{2},$$

where all the notations are as in Section 2. Since we know that ν_k eventually decreases doubly exponentially, we can bound C_G via the inequality

$$C_G(m,n) \leq \frac{2}{m} \sum_{k>0} k\nu_{\geq k}.$$

However we can achieve better bounds, using the majorization Theorem 7. We start from the following:

Lemma 13 *Let $\bar{v} = (v_1, v_2, \ldots, v_n)$ and $\bar{u} = (u_1, v_2, \ldots, u_n)$ be two positive integer vectors. If $\bar{v} \succeq \bar{u}$ then*

$$\sum_{1 \leq i \leq n} v_i^2 \geq \sum_{1 \leq i \leq n} u_i^2.$$

This lemma is a special case of a well-known theorem from majorization (see e.g. [22]), but for completeness we present a proof.
Proof: Let $\bar{x}$ be a n-vector and let $(\bar{x}, \bar{u})$ denote the inner product of $\bar{x}$ and $\bar{u}$. Consider the linear program

$$\text{Maximize } (\bar{x}, \bar{u}) \text{ subject to } \bar{x} \preceq \bar{v} \text{ and } \bar{x} \geq 0.$$

It is easy to check that $\bar{x} = \bar{u}$ is a feasible point and that the optimal solution is $\bar{x} = \bar{v}$. Hence $(\bar{u}, \bar{u}) \leq (\bar{v}, \bar{u})$. Now consider the same program with the objective function $(\bar{x}, \bar{v})$. Then again $\bar{x} = \bar{u}$ is a feasible point and the optimal solution is $\bar{x} = \bar{v}$. Hence $(\bar{u}, \bar{u}) \leq (\bar{u}, \bar{v}) \leq (\bar{v}, \bar{v})$. $\square$

Consider now the standard direct chaining method. In our terminology it corresponds to the random placement algorithm R and it therefore majorizes G. It is well known that the cost for successful search for direct chaining is [19, ex. 6.4.34]

$$C_R(m,n) = \frac{1}{m} \sum_{1 \leq j \leq n} \binom{\lambda_j^R + 1}{2} = 1 + \frac{m-1}{2n}.$$

Applying the Lemma above we obtain that the cost of successful search in 2-way chaining satisfies

$$C_G(m,n) \leq 2 + \frac{m-1}{n}.$$

6 Competitive Online Load Balancing

6.1 Preliminaries

The online load balancing problem is defined as follows. Let M be a set of servers (or machines) that is supposed to run a set of tasks that arrive and depart in time. Each task j has associated to it a weight, or load, $w(j) \geq 0$, an arrival time $\tau(j)$, and a set $M(j) \subset M$ of servers capable of running it. We distinguish among two variants of this problem: the case of *permanent tasks*, tasks that arrive but never depart, and the case of *temporary tasks*, that depart the system at a time unknown in advance.

As soon as it arrives, each task must be assigned to exactly one of the servers capable of running it, and once assigned, it can not be transferred to a different server. The assigned server starts to run the task immediately, and continues to run it until the task departs.

When task j arrives, an *assignment algorithm* must select a server $i \in M(j)$, and assign task j to it.

The *load* on server i at time t, denoted $L_i^A(t)$, is the sum of the weights of all the tasks running on server i at time t under assignment algorithm A.

Let σ be a sequence of task arrivals and departures, and let $|\sigma|$ be the time of the last arrival. Then the cost, $C_A(\sigma)$, of an assignment algorithm A on the sequence σ, is defined as

$$C_A(\sigma) = \max_{0 \leq t \leq |\sigma|; i \in M} L_i^A(t).$$

An *on-line assignment algorithm* must assign an arriving task j at time $\tau(j)$ to a server in $M(j)$ knowing only $w(j)$, $M(j)$, the current state of the servers, and the past – the decision is made without any knowledge about future arrivals or departures. The *optimal off-line assignment algorithm*, denoted OPT, assigns arriving tasks knowing the entire sequence of task arrivals and departures and does so in a way that minimizes its cost.

The worst case behavior of an on-line algorithm A is characterized by the *competitive ratio* defined as the supremum over all sequences σ of the ratio $C_A(\sigma)/C_{\mathrm{OPT}}(\sigma)$.

To characterize the average behavior of A, let $C_A(\mathcal{P})$ (resp. $C_{\mathrm{OPT}}(\mathcal{P})$) be the expected cost of algorithm A (resp. OPT) on sequences σ generated by the distribution $\mathcal{P}$. The competitive ratio of an on-line algorithm A *against distribution* $\mathcal{P}$ is defined as the ratio $C_A(\mathcal{P})/C_{\mathrm{OPT}}(\mathcal{P})$.

Finally, the greedy algorithm is formally defined as follows:

Algorithm GREEDY**:** Upon arrival of a task j assign it to the server in $M(j)$ with the current minimum load (ties are broken arbitrarily).

6.1.1 Permanent Tasks

For permanent tasks, Azar, Naor and Rom [9] showed that the competitive ratio of the greedy algorithm is $\Theta(\log n)$ and that no algorithm can do better.

To bring this problem in our framework, we present our results for the case where for each task j the set of servers that can run it, $M(j)$, consists of $d \geq 2$ servers chosen uniformly at random (with replacement), the number of requests $|\sigma|$ equals n, and all weights are equal. Let $\mathcal{P}_d$ be the associated probability distribution on request sequences.

Lemma 14 *With probability* $1 - O(1/n)$, $C_{\mathrm{OPT}}(\mathcal{P}_d) = O(1)$.

Proof: We show that with high probability there is an assignment with cost 10 for the case $d = 2$. A fortiori the result is true for $d > 2$.

The problem can be reduced to showing that in a random n by n bipartite graph (U, V, E) where each node in U, has two random edges to V, there is an assignment of value 10. Arbitrarily break U into 10 pieces of size $n/10$ each. We show that each of these pieces contains a perfect matching. By Hall's theorem, the probability that there is no such assignment is bounded by the probability that there is a set of size k in one of the pieces of U whose neighborhood has size less than k. Ipso facto, this probability is at most

$$10 \sum_{k \leq n/10} \binom{n}{k-1} \binom{n/10}{k} \left(\left(\frac{k-1}{n} \right)^2 \right)^k .$$

Using standard approximations to the binomial coefficients, this is at most

$$10 \sum_{k \leq n/10} \left(\frac{en}{k-1} \right)^{k-1} \left(\frac{en}{10k} \right)^k \left(\left(\frac{k-1}{n} \right)^2 \right)^k .$$

Finally, rewriting and simplifying yields

$$\frac{10}{n}\sum_{k\le n/10}\frac{k-1}{e}\left(\frac{e^2n^2(k-1)^2}{10k(k-1)n^2}\right)^k=\frac{10}{n}\sum_{k\le n/10}\frac{k-1}{e}\left(\frac{e^2}{10}\right)^k=O\left(\frac{1}{n}\right).$$

□

(A more delicate analysis [15] shows that the maximum load achieved by the off-line case is 2 with high probability, for $d \geq 2$ and $m \leq 1.6n$.)

Lemma 15 *With high probability,* $C_{\text{GREEDY}}(\mathcal{P}_d) = O(\log\log n/\log d)$

Proof: Follows immediately from Theorem 4. □

Thus, we obtain the following theorem.

Theorem 16 *The competitive ratio of the* GREEDY *algorithm against the distribution* $\mathcal{P}_d$ *is* $O(\log\log n/\log d)$ *and no algorithm can do better.*

Proof: Follows from Lemma 14, Lemma 15, and Corollary 8. □

6.1.2 Temporary Tasks

For temporary tasks, the results of Azar, Broder and Karlin [6], and Azar, Kalyanasundaram, Plotkin, Pruhs and Waarts [8], showed that there is an algorithm with competitive ratio $\Theta(\sqrt{n})$ and that no algorithm can do better.

It is difficult to construct a natural distribution of task arrivals and departures. As an approximation, we consider the following stochastic process $\mathcal{S}$: First, n tasks arrive; for each task, the set of servers that can run it consists of $d \geq 2$ servers chosen uniformly at random (with replacement). Then the following repeats forever: a random task among those present departs, and a random task arrives, which again may be served by any one of d random servers. Clearly, in such an infinite sequence, eventually there will be n tasks which can only be served by one server, and so for trivial reasons the competitive ratio for long enough sequences is 1. However we can state a competitiveness result in the following way:

Theorem 17 *Let* $L_A[t]$ *be the maximum load on any server at time* t*, for tasks arriving according to the stochastic process* $\mathcal{S}$*, and assigned using algorithm* A*, that is,* $L_A[t] = \max_{i\in M} L_i^A(t)$*. Then for any fixed* $t > 0$*, with high probability,*

$$\frac{L_{\text{GREEDY}}[t]}{L_{\text{OPT}}[t]} = O(\log\log n).$$

Proof: Follows from Lemma 14, and Theorem 10. □

7 Experimental results

The bound proven in Theorem 4 for the $O(1)$ term in the formula for the upper bound on the maximum load is rather weak (≈ 8), so it might be the case that for practical values of n, the constant term dominates the $\ln\ln n/\ln d$ term. However experiments seem to indicate that this is not the case, and in fact even for small values of n the maximum load achieved with $d = 2$ is substantially smaller than the maximum load achieved with $d = 1$. For the values we considered, $256 \leq n \leq 16777216$, increasing d beyond 2 has only limited further effect. For each value of n and d we run 100 experiments. The results are summarized in Figure 1.

Acknowledgment. We wish to thank Martin Dyer, Alan Frieze, and Greg Nelson for several very useful discussions.

n	$d = 1$	$d = 2$	$d = 3$	$d = 4$
2^8	**3**..... 1% **4**.....40% **5**.....41% **6**.....15% **7**..... 3%	**2**.....10% **3**.....90%	**2**.....84% **3**.....16%	**2**.....99% **3**..... 1%
2^{12}	**5**.....12% **6**.....66% **7**.....17% **8**..... 4% **9**..... 1%	**3**.....99% **4**..... 1%	**2**.....12% **3**.....88%	**2**.....91% **3**..... 9%
2^{16}	**7**.....48% **8**.....43% **9**..... 9%	**3**.....64% **4**.....36%	**3**.....100%	**2**.....23% **3**.....77%
2^{20}	**8**.....28% **9**.....61% **10**.....10% **13**..... 1%	**4**.....100%	**3**.....100%	**3**.....100%
2^{24}	**9**.....12% **10**.....73% **11**.....13% **12**..... 2%	**4**.....100%	**3**.....100%	**3**.....100%

Figure 1: Experimental maximum load ($m = n$).

References

[1] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 238–247, 1995.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley and Sons, 1992.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *34th Annual Symposium on Foundations of Computer Science*, pages 32–40, Palo Alto, California, 3–5 Nov. 1993. IEEE.

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

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

[6] Y. Azar, A. Z. Broder, and A. R. Karlin. On-line load balancing. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 218–225, 1992.

[7] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 593–602, 1994.

[8] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. Online load balancing of temporary tasks. In *Workshop on Algorithms and Data Structures (WADS)*, pages 119–130, Montreal, Canada, 1993.

[9] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.

[10] A. Z. Broder, A. M. Frieze, C. Lund, S. Phillips, and N. Reingold. Balanced allocations for tree-like inputs. *IPL*, 55(6):329–332, 1995.

[11] A. Z. Broder and A. R. Karlin. Multilevel adaptive hashing. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 43–53, 1990.

[12] A. Czumaj and V. Stemann. Randomized allocation processes. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, 1997.

[13] M. Dietzfelbinger, A. R. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal on Computing*, 23(4):738–761, 1994.

[14] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with O(1) worst case access time. *Journal of the ACM*, 31:538–544, 1984.

[15] A. M. Frieze. Personal communication. 1994.

[16] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28(2):289–304, April 1981.

[17] N. L. Johnson and S. Kotz. *Urn Models and Their Application.* John Wiley & Sons, 1977.

[18] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 318–326, 1992.

[19] D. E. Knuth. *The Art of Computer Programming, Vol. III: Sorting and Searching.* Addison-Wesley, 1973.

[20] V. F. Kolchin, B. A. Sevastyanov, and V. P. Chistyakov. *Random Allocations.* John Wiley & Sons, 1978.

[21] P. D. Mackenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 153–162, 1994.

[22] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications.* Academic Press, 1979.

[23] M. Mitzenmacher. Load balancing and density dependent jump Markov processes (extended abstract). In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 213–222, 1996.

[24] M. Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing.* PhD thesis, University of California, Berkeley, 1996.

[25] V. Stemann. Parallel balanced allocations. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1996.