Perfectly Balanced Allocation[⋆]

Artur Czumaj[1], Chris Riley[2], and Christian Scheideler[2]

[1] Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA,
czumaj@cis.njit.edu

[2] Department of Computer Science, Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218, USA,
{chrisr,scheideler}@cs.jhu.edu

Abstract. We investigate randomized processes underlying load balancing based on the multiple-choice paradigm: m balls have to be placed in n bins, and each ball can be placed into one out of 2 randomly selected bins. The aim is to distribute the balls as evenly as possible among the bins. Previously, it was known that a simple process that places the balls one by one in the least loaded bin can achieve a maximum load of $m/n + \Theta(\log\log n)$ with high probability. Furthermore, it was known that it is possible to achieve (with high probability) a maximum load of at most $\lceil m/n \rceil + 1$ using maximum flow computations.
In this paper, we extend these results in several aspects. First of all, we show that if $m \geq c\,n\,\log n$ for some sufficiently large c, then a *perfect* distribution of balls among the bins can be achieved (i.e., the maximum load is $\lceil m/n \rceil$) with high probability. The bound for m is essentially optimal, because it is known that if $m \leq c'\,n\,\log n$ for some sufficiently small constant c', the best possible maximum load that can be achieved is $\lceil m/n \rceil + 1$ with high probability. Next, we analyze a simple, randomized load balancing process based on a local search paradigm. Our first result here is that this process always converges to a best possible load distribution. Then, we study the convergence speed of the process. We show that if m is sufficiently large compared to n, then *no matter* with which ball distribution the system starts, if the imbalance is $\varDelta$, then the process needs only $\varDelta \cdot n^{O(1)}$ steps to reach a perfect distribution, with high probability. We also prove a similar result for $m \approx n$, and show that if $m = O(n \log n/\log\log n)$, then an optimal load distribution (which has the maximum load of $\lceil m/n \rceil + 1$) is reached by the random process after a polynomial number of steps, with high probability.

Keywords: load balancing, local search algorithms, stochastic processes.

1 Introduction

The study of balls-into-bins games or occupancy problems has a long history (see e.g. [1,2,3,4,5,8,10,11,12,18]). These problems have numerous applications, e.g., in graph theory, queueing theory, hashing, and randomized rounding. In general, the goal of a

[⋆] Research supported in part by NSF grant CCR-0105701.

balls-and-bins algorithm is to assign a set of independent objects (tasks, jobs, memory blocks) to a set of resources (servers, disks) so that the load is distributed among the bins as evenly as possible.

In the classical single-choice game, each ball is placed into a bin chosen *independently and uniformly at random* (*i.u.r.*). For the case of n bins and $m \geq n \log n$ balls it is well known that there exists a bin receiving $m/n + \Theta(\sqrt{m \log n/n})$ balls. This result holds not only in expectation but also with high probability. (We say that an event A occurs *with high probability* (*w.h.p.*) if $\mathbf{Pr}[A] \geq 1 - n^{-\alpha}$ for an arbitrarily chosen constant $\alpha \geq 1$.) On the other hand, it was shown by Azar *et al.* [1] and Berenbrink *et al.* [2] that if the balls are placed in a sequential (on-line) fashion and each ball is assigned to the currently least loaded of the two locations (ties broken arbitrarily), then the maximum load of any bin is $m/n+\Theta(\log\log n)$ with high probability. It can also be proven [1,2] that any protocol that assigns the balls to the bins in an on-line fashion (that is, the decision where the ball is placed is performed only on the base of the placement of the previously placed balls) cannot be stochastically better than the scheme above. In particular, this implies that in any on-line scheme, with high probability, there is a bin with load $m/n + \Theta(\log\log n)$.

On the other side, some authors have been studying *off-line assignments*. In off-line assignments, after first selecting the two locations for all the balls, one seeks an optimal placement of the balls assuming each ball can choose only among its two locations and the locations of all balls are known to the algorithm (off-line case). This problem arises naturally in numerous applications, for example, in hashing, scheduling, load balancing, and video on demand (see, e.g., [1,7,9,14,15,16]). (For example, Sanders *et al.* [16] discussed in depth applications to support fast parallel access to external memory systems with parallel disks and Karp [7] discussed applications in video on demand; Karp called our problem k-orientability.)

Let the *minmax load* be the minimum, over all possible placements of the balls into bins, of the maximum load in the system. Azar *et al.* [1] showed that for $n = \Theta(m)$, the minmax load is $\Theta(1)$, with high probability. Later, Frieze (personal communication in [1]) and, independently, Czumaj and Stemann [5], tightened this bound and, in particular, showed that for $n = m$, the minmax load is exactly 2, with high probability. Sanders *et al.* [16] extended the result from [1,5] to arbitrary m and proved the following result.

Theorem 1. [16] *The minmax load is at most* $\lceil m/n \rceil + 1$, *with high probability.* □

Notice that since the minmax load cannot be smaller than $\lceil m/n \rceil$, this bound is optimal up to an additive constant 1. Furthermore, it is easy to see that there exists a positive constant λ, such that if $m \leq \lambda n \ln n$, then the bound in Theorem 1 is tight[3]. Our first contribution is that this bound for m is asymptotically tight in the following sense: there is a constant c such that if $m \geq cn \ln n$, then a *perfect balance* is possible:

Theorem 2. *There exists a positive constant c such that for every $m \geq cn \ln n$, the minmax load is exactly $\lceil m/n \rceil$, with high probability.*

[3] Indeed, if we choose at random two locations for each of the $\lambda n \ln n$ balls, then there will be a bin that has not been chosen by any ball. Therefore, there is a bin whose load is 0 w.h.p. and hence it is impossible that all bins have identical load of m/n, w.h.p.

Stochastic load balancing. Next, we present a novel approach to off-line assignments and discuss a new stochastic process (algorithm) that achieves *optimal maximum* load. Sanders *et al.* [16] described a polynomial time algorithm that finds an optimal assignment of the balls into bins minimizing the maximum load (which in this optimal allocation is equal to the minmax load). Their algorithm uses maximum flow computations.

A drawback of the approach by Sanders *et al.* is that it requires global (centralized) knowledge about locations of all balls, which is far too space consuming if m is large. This makes also the algorithm difficult (if suitable at all) for implementations in distributed or decentralized systems (like, for example, systems of parallel disks as discussed in [9,16]). Therefore, as our second contribution, we present a simple, memoryless, *local search* algorithm that can balance the load of the bins in the system as much as this is possible. The idea behind our algorithm is to begin with an arbitrary assignment of the balls to the bins, and then to use a stochastic replacement process that gradually improves the balance of the bins' load.

Suppose that initially all the balls have chosen their locations in $\{1, \ldots, n\}$ and each ball is (arbitrarily) placed in one of its two locations. The *Self-Balancing Algorithm* repeats the following *Self-Balancing Step*:

Self-Balancing Step:

```
Pick independently and uniformly at random a pair of bins $(b_1, b_2)$.
If there is a ball placed in $b_1$ with alternative location in bin $b_2$, then
    Pick any ball $x$ that is placed in $b_1$ with alternative location in bin $b_2$;
    Place $x$ into the least loaded bin (among $b_1$ and $b_2$);
        If tie, that is, bin $b_1$ has (without $x$) the same load as bin $b_2$, then
            place $x$ into a randomly chosen of the two bins.
```

We prove two theorems about the Self-Balancing Algorithm (throughout our analysis, unless stated otherwise, terms *"with high probability"* are with respect to the random choices of the two locations of each ball, as well as the random choices of balls in the Self-Balancing Algorithm).

The first theorem shows that the Self-Balancing Algorithm will gradually converge to states in which the maximum load is best possible.

Theorem 3. *If the Self-Balancing Algorithm is run sufficiently long (i.e., the Self-Balancing Step is repeated sufficiently many times), then the maximum load of any bin in the system is equal to the minmax load with probability 1. (The probability 1 is with respect to the random choices of balls in the Self-Balancing Algorithm only.)*

In particular, if the Self-Balancing Algorithm is run sufficiently long then the maximum load of any bin in the system is smaller than or equal to $\lceil m/n \rceil + 1$ *with high probability. If, additionally,* $m > c\,n \ln n$ *for a sufficiently large constant c, then the maximum load is exactly* $\lceil m/n \rceil$ *with high probability.*

The Self-Balancing Algorithm is a simple example of a local search algorithm, similar to load balancing algorithms existing in the literature before, see, e.g., [6,13]. Theorem 3 shows the non-trivial property that no matter with which state (i.e., assignment of balls to bins) the Self-Balancing Algorithm starts, it will always converge to a state in which the maximum load is optimally small. Notice that in many local search approaches one frequently arrives at a "dead-lock" situation, in which the balancing may be far away from optimal and no re-balancing progress is possible (that is, a locally optimal solution is not in a global optimum). Theorem 3 shows that this is not the case for the Self-Balancing Algorithm. (Observe, however, that if we removed the randomized rule for tie breaking, then — as one can easily show — the algorithm would not necessarily converge to an optimal state.)

The next theorem considers the heavily loaded case and deals with the speed of the "convergence" of the Self-Balancing Algorithm to a state in which the maximum load is upper bounded by $\lceil m/n \rceil$. Let the *imbalance* of the system be its distance from a best possible distribution, or more precisely, $\sum_{i=1}^{n} \max\{0, \text{load of bin } i - \lceil m/n \rceil\}$.

Theorem 4. *If $m \gg n$, then after a polynomial number (with respect to n only) of Self-Balancing Steps the maximum load in the system is equal to $\lceil m/n \rceil$, with high probability. Furthermore, if the system imbalance is Δ, then the number of steps is $\Delta \cdot n^{O(1)}$, with high probability.*

Notice that if the balls are allocated to the bins in the on-line fashion using the least loaded bin approach, as in [1,2], the system imbalance is $\Delta = O(n \log \log n)$, with high probability [2]. Therefore, Theorem 4 implies the following corollary.

Corollary 1. *If $m \gg n$, then in time $O(m) + n^{O(1)}$ one can find a perfect load distribution with the maximum load of the system equal to $\lceil m/n \rceil$, with high probability.* □

As we argued before, one cannot extend the result from Theorem 4 to the case $m \approx n$, because then the minmax load is expected to be equal to $\lceil m/n \rceil + 1$ (instead of $\lceil m/n \rceil$). Our next theorem shows however that if m is close to n, then the Self-Balancing Algorithm still rapidly converges to the optimal distribution.

Theorem 5. *If $m = O(n \log n / \log \log n)$, then after a polynomial number (with respect to n) of Self-Balancing Steps the maximum load in the system is smaller than or equal to $\lceil m/n \rceil + 1$, with high probability.*

Notational conventions. To simplify the presentation of the paper, we will use a shorthand μ to denote m/n and $\widehat{\mu}$ to denote $\lceil m/n \rceil = \lceil \mu \rceil$. We shall identify the balls with the integers in $\{1, \dots, m\} = [m]$ and the bins with the integers in $\{1, \dots, n\} = [n]$. Let the *load* of a bin $b \in [n]$ be equal to the number of balls placed in b. Notice that the average load among all the bins is μ.

2 Perfect Balancing for $\Omega(n \log n)$ Balls

In this section we prove Theorem 2, that is, we show that if $m \geq cn \log n$ for certain suitable constant c, then the minmax load is $\lceil m/n \rceil = \widehat{\mu}$, with high probability. It is easy to see that it is sufficient to prove this bound in the case $\mu = \widehat{\mu}$, and therefore from now on we assume that μ is an integer.

Let $\mathfrak{B}$ denote the set of n bins in the system. Let us fix an allocation of m balls to n bins in $\mathfrak{B}$ such that each ball has two locations in $\mathfrak{B}$ (we allow a ball to have both locations in the same bin). For any $U \subseteq \mathfrak{B}$, let $\Psi[U]$ denote the number of balls having all locations in the bins in U. Then, one can show the following result (see [16,17]).

Lemma 1. *[17, Theorem 1] The minmax load is equal to* $\max_{U \subseteq \mathfrak{B},\, U \neq \emptyset} \left\lceil \frac{\Psi[U]}{|U|} \right\rceil$. □

Consider the stochastic process of assigning two locations of the m balls to the n bins in $\mathfrak{B}$ i.u.r. For any set $U \subseteq \mathfrak{B}$, let C_U be the random variable denoting the value of $\Psi[U]$. Furthermore, let $\mathcal{E}_U$ be the random indicator of the event that $C_U > \mu \cdot |U|$ and let $\mathcal{E} = \bigvee_{U \subseteq \mathfrak{B},\, U \neq \emptyset} \mathcal{E}_U$. Our goal is to show that

$$\mathbf{Pr}[\mathcal{E}] \;\le\; n^{-\gamma} \; . \tag{1}$$

for a constant γ depending on c.

Let $\mathfrak{B}_k = \{U \subseteq \mathfrak{B} : |U| = k\}$. Then, by the union bound, to prove (1) it is enough to prove the following bound for every[4] k, $1 \le k \le n-1$, and for every set $U \in \mathfrak{B}_k$:

$$\mathbf{Pr}[\mathcal{E}_U] \;\le\; \frac{1}{n^{\gamma+1} \cdot \binom{n}{k}} \; . \tag{2}$$

From now on, we concentrate on proving inequality (2). Let us observe that for any set $U \in \mathfrak{B}_k$, the value of C_U is a binomial random variable with the parameters m and $(k/n)^2$, which we denote by $\mathbb{B}(m,(k/n)^2)$. Therefore, $\mathbf{Pr}[\mathcal{E}_U] = \mathbf{Pr}[\mathbb{B}(m,(k/n)^2) > m \cdot k/n] \le \mathbf{Pr}[\mathbb{B}(m,(k/n)^2) \ge m \cdot k/n]$ and our goal now is to investigate bounds for $\mathbf{Pr}[\mathbb{B}(m,(k/n)^2) \ge m \cdot k/n]$.

We begin with three simple results about concentration of binomial random variables.

Lemma 2.

1. *For any* $t \ge 6\, m\, q^2$, $\mathbf{Pr}[\mathbb{B}(m,q^2) \ge t] \;\le\; 2^{-t}$.
2. *For any* $0 < q < 1$, $\mathbf{Pr}[\mathbb{B}(m,q^2) \ge q \cdot m] \;\le\; \exp(-2\, q^2\, (1-q)^2\, m)$.
3. *For any* $0 < q < 1$, *if* $\frac{1}{u-1} \le q$ *for certain* $u > 1$, *then* $\mathbf{Pr}[\mathbb{B}(m,q^2) \ge q \cdot m] \;\le\; (u/e)^{m\,(1-q)}$. □

Let $m \ge c\, n \ln n$ for a large constant c. Let $U \in \mathfrak{B}_k$ and $q = k/n$. Let us first consider the case $k/n = q \le 0.1$. Then, if we set $t = m\, k/n$, then we have $t \ge 6 \cdot \mathbf{E}[\mathbb{B}(m,q^2)]$, and hence by Lemma 2 (1) and by the inequality $\binom{n}{k} \le n^k$, we get (provided c is a large enough constant):

$$\mathbf{Pr}[\mathcal{E}_U] \le 2^{-t} = 2^{-m\,k/n} \le e^{-(\gamma+1)\ln n - k\,\ln n} = \frac{1}{n^{\gamma+1} \cdot n^k} \le \frac{1}{n^{\gamma+1} \cdot \binom{n}{k}} \; . \tag{3}$$

Next, we consider $0.1\, n \le k < 2/3\, n$. Then, by Lemma 2 (2) and by observing that $\binom{n}{k} \le 2^n$, we have (again, if we set $m = c\, n\, \ln n$ for a large enough constant c)

$$\mathbf{Pr}[\mathcal{E}_U] \le \exp(-2(\tfrac{k\,(n-k)}{n^2})^2 m) \le e^{-0.001 m} \le e^{-(\gamma+1)\ln n - k \ln n} \le \frac{1}{n^{\gamma+1} \cdot \binom{n}{k}} \; . \tag{4}$$

[4] We do not have to consider the case $U = \mathfrak{B}_n$ because in that case $\mathcal{E}_U$ trivially never holds.

The remaining case is when $k/n = q \geq 2/3$. Then, we can apply Lemma 2 (3) with $u = 2.5$ to obtain

$$\mathbf{Pr}[\mathcal{E}_U] \leq \mathbf{Pr}[\mathbb{B}(m, q^2) \geq qm] \leq (2.5/e)^{m/3} \leq e^{-(\gamma+1)\ln n - k \ln n} \leq \frac{1}{n^{\gamma+1} \cdot \binom{n}{k}} \; . \tag{5}$$

Therefore, from inequalities (3 – 5), we have that for every integer k, $1 \leq k \leq n-1$, and for every $U \in \mathfrak{B}_k$, we have $\mathbf{Pr}[\mathcal{E}_U] \leq \frac{1}{n^{\gamma+1} \cdot \binom{n}{k}}$. This implies that $\mathbf{Pr}[\mathcal{E}] \leq n^{-\gamma}$, which in turn yields Theorem 2. □

3 Convergence to Optimal Assignment

In this section we sketch the proof of Theorem 3. We begin with basic definitions and notation. A placement of the balls after performing t repetitions of the Self-Balancing Step, $t \geq 0$, is called the *tth assignment*, and is denoted by $\mathcal{A}_t$. To each assignment $\mathcal{A}_t$ we assign a *load vector*, which is vector $\mathbb{L}_t = \langle \mathbb{L}_t(1), \dots, \mathbb{L}_t(n) \rangle$ such that $\mathbb{L}_t(j)$ denotes the load of the jth fullest bin in $\mathcal{A}_t$. For any two load vectors $\mathbb{L} = \langle \mathbb{L}(1), \dots, \mathbb{L}(n) \rangle$ and $\mathbb{L}^* = \langle \mathbb{L}^*(1), \dots, \mathbb{L}^*(n) \rangle$, we say $\mathbb{L}$ *majorizes* $\mathbb{L}^*$, denoted by $\mathbb{L} \succeq \mathbb{L}^*$, if for every j, $1 \leq j \leq n$, we have $\sum_{r=1}^{j} \mathbb{L}(r) \geq \sum_{r=1}^{j} \mathbb{L}^*(r)$. Furthermore, we write $\mathbb{L} \succ \mathbb{L}^*$ if $\mathbb{L} \succeq \mathbb{L}^*$ and there is at least one j with $\sum_{r=1}^{j} \mathbb{L}(r) > \sum_{r=1}^{j} \mathbb{L}^*(r)$.

Our first lemma describes the way the load vector can change in the course of the algorithm. Informally, it says that after any repetition of Self-Balancing Step the load vector will never worsen.

Lemma 3. *For any $t \geq 0$, independently of the random choices performed by the Self-Balancing Algorithm, we always have $\mathbb{L}_t \succeq \mathbb{L}_{t+1}$.* □

Let us observe two important consequences of Lemma 3. Firstly, this lemma implies that the maximum load never increases. Secondly, Lemma 3 yields the following claim:

Lemma 4. *The number of changes in the load vector is upper bounded by $m \cdot n$.* □

Now, since we know the algorithm gradually converges to a more balanced distribution of the bins' loads, we formally describe the states to which it converges. We say, a system is *stable in step τ*, if independently of the random choices performed in the iterations $T > \tau$ of the Self-Balancing Algorithm we will have $\mathbb{L}_\tau = \mathbb{L}_T$ for every $T > \tau$. In order to characterize stable states formally, we define a *directed multigraph* representing the state of the system (see also, e.g., [5,16], for similar representations).

Definition 1. *A* directed multigraph $G = (V, E)$ representing the system *is a directed multigraph with the vertex set $V = \{1, \dots, n\}$ corresponding to the bins in the system and the edge multiset E (loops are allowed) corresponding to the assignment of the balls in the system. Each edge is associated with a ball, has as the endpoints the two locations of the associated ball, and it is directed from (outwards) the bin containing the associated ball.*

We denote by $G_t = (V, E_t)$ the directed multigraph representing $\mathcal{A}_t$. For any vertex v of G we denote by out-deg(v) the *out-degree* of v in G; if G is not clear from the context, then we also use the notation out-deg$_G(v)$. The *in-degree* is defined analogously. Notice that since the choices of the locations of each bin are performed at random, the *undirected* version of any G_t is a *random multigraph* with n vertices and m edges (where each endpoint of each edge is selected independently and uniformly at random).

The following lemma follows directly from Definition 1.

Lemma 5. *If $G_t = (V, E_t)$ is a directed multigraph representing $\mathcal{A}_t$, then for any j, $1 \le j \le n$, the out-degree of vertex j is equal to the load of bin j in $\mathcal{A}_t$.* □

Let $G_\tau = (V, E_\tau)$ be the directed multigraph representing $\mathcal{A}_\tau$. A directed path $(v_1, v_2, \ldots, v_\ell)$ in G_τ is called a *slope* if out-deg$(v_1) \ge$ out-deg$(v_\ell)+2$ and out-deg(v_i) $\ge$ out-deg(v_{i+1}) for every i, $1 \le i < \ell$. If $(v_1, v_2, \ldots, v_\ell)$ is a slope in G_τ, then we can *straighten* $(v_1, v_2, \ldots, v_\ell)$ by modifying the directions of the edges in G_τ (following the rules in the Self-Balancing Algorithm) so that the load vector will change (see also a scheme presented in Figure 1). Indeed, let us consider the case that $\ell \ge 3$ (the case $\ell = 2$ can be handled similarly), and assume (actually, without loss of generality) that out-deg(v_1) = out-deg$(v_2) + 1$, out-deg(v_j) = out-deg(v_{j+1}) for $2 \le j < \ell - 1$, and that out-deg$(v_{\ell-1})$ = out-deg$(v_\ell) + 1$. Then, we reverse directions of the edges (v_j, v_{j+1}) for all $1 \le j < \ell - 1$ (this can be easily done according to the rules in the Self-Balancing Algorithm). After applying these changes, the bin corresponding to the vertex v_1 decreased its load by 1, the bin corresponding to the vertex v_ℓ increased its load by 1, and the load of all other bins remains the same. This implies that the load vectors $\mathbb{L}$ of $\mathcal{A}_\tau$ and $\mathbb{L}'$ of the new system state fulfill $\mathbb{L} \succ \mathbb{L}'$.

The following key lemma provides a necessary and sufficient condition for a system to be stable at step t. (Notice that the *only if* part follows from our arguments above.)

Lemma 6. *A system is stable at step τ if and only if the directed multigraph $G_\tau = (V, E_\tau)$ representing $\mathcal{A}_\tau$ has no slope.* □

The next lemma describes a relationship between stable states and the maximum load in the system.

Lemma 7. *Consider a system of m balls and n bins with the minmax load κ. Then, if the system is stable in step τ then the maximum load of $\mathcal{A}_\tau$ is κ.*

Proof. The proof is by contradiction. Let us consider a system of m balls and n bins with the minmax load κ. Let us suppose the system is in a stable state $\mathcal{A}_\tau$ represented by the directed multigraph $G_\tau = (V, E_\tau)$, and, for the purposes of contradiction, let us assume that the maximum out-degree in G_τ is greater than κ.

Since $\mathcal{A}_\tau$ is a stable state, we know by Lemma 6 that G_τ has no slope. Let us pick any vertex $v \in V$ with out-deg$_{G_\tau}(v) > \kappa$. Let U be the set of all vertices in G_τ (not including v) that are reachable from v by a directed path in G_τ. Since G_τ has no slope, all vertices in U must have the out-degree at least out-deg$_{G_\tau}(v) - 1 \ge \kappa$. Therefore, if we define $U^* = U \cup \{v\}$, then there are at least $|U| \cdot \kappa + (\kappa + 1)$ balls having both locations in the bins corresponding to the vertices in U^*. This, however, by Lemma 1, means that minmax load is at least $\frac{1}{|U|+1} \cdot (|U| \cdot \kappa + (\kappa + 1)) > \kappa$, which is a contradiction to our initial assumption that the minmax load of the system is κ. □

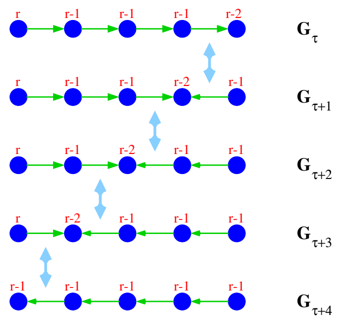

Fig. 1. Illustration describing the *straightening* procedure that changes the out-degrees of the vertices $v_1, v_2, \ldots, v_s$ (with $s = 5$) on the slope performed in the proof of Lemma 6. In this case, initially we have out-deg$(v_1) = r$, out-deg$(v_2) =$ out-deg$(v_3) =$ out-deg$(v_4) = r - 1$, and out-deg$(v_5) = r - 2$.

Now we are ready to complete the proof of Theorem 3. By Lemma 7, the system is not stable if and only if the directed multigraph $G_\tau = (V, E_\tau)$ representing $\mathcal{A}_\tau$ has a slope $(v_1, \ldots, v_\ell)$ for certain positive ℓ. Thus, if the system is not stable, then let us consider any shortest slope. Then, with a positive probability, in the next $\ell - 1$ iterations in the Self-Balancing Algorithm we will perform slope straightening of $(v_1, \ldots, v_\ell)$, which will decrease the load of v_1 by 1, increase the load of v_ℓ by 1, and leave the remaining loads the same. Hence, if $\mathcal{A}_t$ is not stable, then after sufficiently many iterations of Self-Balancing Step, with probability 1 the load vector will be modified. Since the load vector may change at most $n\,m$ times, if we combine the arguments above with Lemma 7, after sufficiently many iterations in Self-Balancing Step, with probability 1 the system will be in a stable state in which the maximum load equals the minmax load. □

4 Convergence to Optimal Assignment for $m \gg n$

In this section we briefly sketch the proof of Theorem 4, which estimates the convergence speed of the Self-Balancing Algorithm for $m \gg n$. First of all, let us recall that by Lemma 4, the load vector may change at most $n\,m$ times. Therefore, we only have to show that if the system is not stable, then after a polynomial number of steps of the Self-Balancing Algorithm the system will change its load vector with high probability. The following is the key theorem of our analysis (the proof is deferred to the full version of the paper).

Theorem 6. *Let $n^5 \log m = o(\mu)$. Let ξ be an arbitrary constant. Let b be a bin with any load greater than or equal to $\mu + \xi$. Then, with probability at least $1 - m^{-O(1)}$,*

- *either every bin has load greater than or equal to $\mu + \xi$,*
- *or the directed multigraph representing the current state of the system has a directed path of length at most 2 from the vertex corresponding to b to some other vertex u whose out-degree is strictly smaller than $\mu + \xi$.*

In view of this theorem, with high probability, as long as the maximum load in the system is strictly larger than $\widehat{\mu}$, the directed multigraph representing the state of the system has *always* a slope $(v_0, \dots, v_\tau)$ with $\tau \leq 2$, *no matter* how the directions of the edges are set. (Indeed, in that case there is a bin b with the load larger than $\widehat{\mu}$, and if we set $\xi = 0$, then it is impossible that every bin in the system has load greater than or equal to μ. Therefore, by Theorem 6, there must exist a directed path of length at most 2 from the vertex corresponding to b to some other vertex u, such that the out-degree of u is strictly smaller than μ. Therefore, either this path or its sub-path must be a slope.) Therefore, with probability at least $O(1/n^4)$, the Self-Balancing Algorithm will, in at most two steps, perform slope straightening of $(v_0, \dots, v_\tau)$ such that the out-degree of v_0 decreases from some ℓ to $\ell-1$ and no other vertex on the path increases its out-degree to more than $\ell - 1$. Therefore, the system will change its load vector with probability at least $O(1/n^4)$. Hence, with high probability the system will change its load after $O(n^4)$ Self-Balancing Steps, and thus, after $O(mn^5)$ steps the Self-Balancing Algorithm will reach a state in which the maximum load equals to the minmax load.

Actually, it is easy to see that our arguments above can be used to show that if the imbalance of the system is Δ (where $\Delta = \sum_{i=1}^{n} \max\{\mathbb{L}(i) - \widehat{\mu}, 0\}$), then the process needs only $\Delta \cdot n^{O(1)}$ steps to reach a perfect distribution, with high probability. This yields the proof in the heavily loaded case. □

5 Convergence to Optimal Assignment for $m = O(n)$

In this section we deal with the proof of Theorem 5 and consider the convergence speed of the Self-Balancing Algorithm in the lightly loaded case. We focus only on the case $m = O(n)$; we believe that this is the most challenging case and therefore we will elaborate on its proof. The analysis of the case $m = O(n \log n / \log\log n)$, $m = \omega(n)$, is deferred to the full version of the paper.

The main idea behind the proof is to use similar arguments as in the previous section, but this time we cannot assume that we have a slope of a constant length. The analysis requires the following three key properties. The first property, proven in [16], is that if the pairs of locations for all the balls are chosen i.u.r., then (with high probability, depending only on the random choices of the locations) in any state of the system, if there is a bin with load greater than $\widehat{\mu} + 1$ then there is a slope of length $O(\log n)$. The second property is that the sum of the degrees (in- and out-degrees) of all vertices on this slope path is at most $O(\log n)$. The third property is that the probability that a given slope path will be straightened is inversely proportional to the sum of the degrees of the vertices on this path. With these properties, we can show that the probability that in the next $O(mn \log n)$ Self-Balancing Steps a slope of length $O(\log n)$ is chosen and then straightened by the algorithm (without interfering with the other bins (vertices)) is at least $O(1/n^{O(1)})$. This implies that (with high probability) in the next $O(n^{O(1)})$ steps the Self-Balancing Algorithm will change the load vector. Therefore, (with high probability) after $n^{O(1)}$ steps the Self-Balancing Algorithm will reach a state, in which, by Theorem 1, the maximum load is at most $\widehat{\mu} + 1$, with high probability.

We describe now our analysis in more detail. We first develop some properties of the directed multigraphs discussed in Section 3. We begin with a lemma proven implicitly in [16, Lemma 14].

Lemma 8. **[16]** *Let $G_t = (V, E_t)$ be a directed multigraph representing certain $\mathcal{A}_t$. Let $m = O(n)$. Then, with high probability (depending only on the random locations of the balls), either $\mathcal{A}_t$ has the maximum load of at most $\widehat{\mu} + 1$ or G_t has a slope of length $O(\log n)$.* □

Our approach is to explore Lemma 8. First of all, from now on, we shall condition on the fact that there is an assignment of the balls among the bins with maximum load $\widehat{\mu} + 1$. (By Theorem 1, this fact holds with high probability.) Then, by Lemma 8, we know that the system is either in the state when the maximum load is $\widehat{\mu} + 1$, in which case we do not have to prove anything, or there is slope in G_t of length $O(\log n)$. We consider only the latter case.

We work in rounds, each round corresponding to $O(n^3 \log^2 n)$ repetitions of Self-Balancing Step. All rounds are independent. At the beginning of each round we take any slope π in G of length $O(\log n)$ that is promised by Lemma 8 (if no such a path exists, then we know that we are already in a state with maximum load smaller than or equal to $\widehat{\mu} + 1$). We prove in Lemma 10 that with probability greater than or equal to $\frac{1}{poly(n)}$ we will successfully straighten the slope in this round. From this and Theorem 3 it follows easily that after a polynomial number of rounds of the Self-Balancing Algorithm we reach a stable state having the maximum load at most $\widehat{\mu} + 1$, with high probability.

Now, our ultimate goal is to analyze the probability that a slope of length $O(\log n)$ will be straightened in $O(n^3 \log^2 n)$ iterations of the Self-Balancing Algorithm. We begin with an auxiliary lemma about random (undirected) multigraphs (the proof is deferred to the full version of the paper).

Lemma 9. *Let b and c be arbitrary positive constants. If G is a random* undirected *multigraph with n vertices and $m \le b\,n$ edges, then, with high probability G does not have any simple path of length less than or equal to $c\ \log n$ for which the sum of the degrees of the vertices on the path is greater than $d \cdot \log n$, where d is a constant.* □

Our next and key result shows that the probability that the Self-Balancing Algorithm will straighten a given slope path is inversely proportional to the sum of the degrees of the vertices on this path.

Lemma 10. *Let b and c be arbitrary positive constants. Let G be an arbitrary directed multigraph with n vertices and $m \le b\,n$ edges. Suppose there is a slope path $\pi = (v_1, \ldots, v_\ell)$ in G. Then, with probability greater than*

$$\left(1 - \frac{1}{n^{10}}\right) \cdot \frac{1}{n^2} \cdot \left(\prod_{i=2}^{\ell} \frac{1}{1 + \textit{out-deg}(v_1) + \textit{in-deg}(v_i)}\right) \quad ,$$

the load vector will change after less than or equal to $2\,\ell\,m\ \log n$ iterations.

Proof. We only sketch the proof and defer more details to the full version of the paper.

Consider any slope $\pi = (v_1, v_2, \ldots, v_\ell)$ of shortest length in the system. Recall that out-deg$(v_1) - 1 =$ out-deg$(v_2) =$ out-deg$(v_3) = \cdots =$ out-deg$(v_{\ell-1}) =$ out-deg$(v_\ell) +$ 1. If $\ell = 2$, then the probability that the load vector will change in the next step is at least as large as the probability that we will choose the edge (v_1, v_2), which is equal to

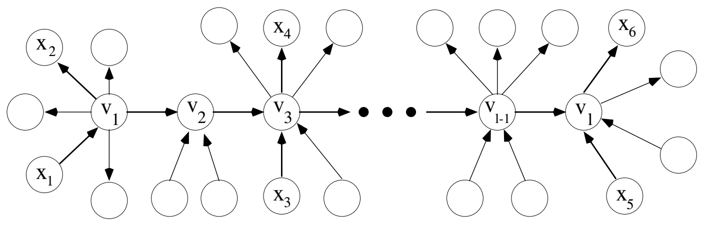

Fig. 2. A slope $\pi = (v_1, v_2, \ldots, v_\ell)$ with incident edges. We only include those edges (y, z) with out-deg(y) = out-deg$(z) + 1$

$1/n^2$. Hence, in this case the lemma easily follows. Therefore, from now on we shall assume that $\ell \geq 3$, i.e. there are no edges (y, z) in G with out-deg$(y) >$ out-deg$(z) + 1$.

We use the terminology from Figure 2. Initially, we have a slope $\pi = (v_1, \ldots, v_\ell)$ of length $\ell - 1$. In each iteration of the Self-Balancing Algorithm we will hit a certain edge chosen at random and in this way we may modify the graph and the load vector. We observe that if we hit an edge that does not belong to π nor is incident to π, then any eventual modification of that edge will not influence path π. Therefore, we only have to consider the following eight cases, when an edge of the following form is chosen: (i) (v_1, v_2), (ii) $(v_{\ell-1}, v_\ell)$, (iii) (x_1, v_1), (iv) (v_1, x_2), (v) (x_3, v_3), (vi) (v_3, x_4), (vii) (x_5, v_ℓ), and (viii) (v_ℓ, x_6). We say a *very good edge is hit* if we hit an edge from cases (iv) or (ix); a *good edge is hit* if we hit an edge from cases (i), (iii), (vi), or (vii); a *bad edge is hit* if we hit an edge from cases (v), or (viii). Very good edges create an edge (y, z) with out-deg$(y) >$ out-deg$(z) + 1$, good edges make the slope shorter, and bad edges make it longer.

Now, we consider a round lasting $2\,\ell\,n^2\ \log n$ iterations and observe only very good edge hits, good edge hits, and bad edge hits. A round is called *successful* if no bad edge is hit until we either have a very good edge hit and then straighten the obtained path or we modify the slope path (we straighten it) by only good edges. One can show that with probability greater than or equal to $1 - 1/n^{10}$ a round is either successful or we made a bad edge hit. Notice that there are at most out-deg(v_1) + in-deg(v_ℓ) bad edges at the beginning, and there is at least one good edge at any time. Certainly, under the assumption that either a bad edge or $(v_{\ell-1}, v_\ell)$ is picked, the probability that $(v_{\ell-1}, v_\ell)$ is picked is at least $1/(1 + \text{out-deg}(v_1) + \text{in-deg}(v_\ell))$. Once $(v_{\ell-1}, v_\ell)$ is picked, we concentrate on the edge $(v_{\ell-2}, v_{\ell-1})$, and so on. Using this approach, we get that the probability that a round is successful is lower bounded by $\frac{1}{n^2} \cdot \left(\prod_{i=2}^{\ell} \frac{1}{1+\text{out-deg}(v_1)+\text{in-deg}(v_i)}\right)$. This completes the proof. □

We can reduce our analysis to the case when for the slope $\pi = (v_1, \ldots, v_\ell)$ we have out-deg$(v_1) = \widehat{\mu} + 2$ and $\ell = O(\log n)$. Therefore, by Lemma 9 we know that $\sum_{i=1}^{\ell}$ in-deg$(v_i) = O(\log n)$, with high probability. Hence, by Lemma 10, the probability that in a round lasting $2nm \log n$ iterations we change the load vector is greater than or equal to $\frac{1}{poly(n)}$. Hence, after $poly(n)$ rounds (iterations) of the Self-Balancing Algorithm we shall modify the load vector with high probability. Now, since the load vector can be modified at most $m \cdot n$ times before we reach the stable state, the theorem follows. □

References

1. Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.
2. P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking. Balanced allocations: The heavily loaded case. *STOC*, pp. 745–754, 2000.
3. R. Cole, A. Frieze, B. M. Maggs, M. Mitzenmacher, A. W. Richa, R. K. Sitaraman, and E. Upfal. On balls and bins with deletions. *RANDOM*, pp. 145–158, 1998.
4. R. Cole, B. M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A. W. Richa, K. Schröder, R. K. Sitaraman, and B. Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. *STOC*, pp. 378–388, 1998.
5. A. Czumaj and V. Stemann. Randomized allocation processes. *Random Structures and Algorithms*, 18(4):297–331, 2001. A preliminary version appeared in *FOCS*, pp. 194–203, 1997.
6. B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM J. Comput.*, 29(1):29–64, September 1999.
7. R. M. Karp. Random graphs, random walks, differential equations and the probabilistic analysis of algorithms. *STACS*, pp. 1–2, 1998.
8. R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. *Algorithmica*, 16(4/5):517–542, 1996.
9. J. Korst. Random duplicated assignment: An alternative to striping in video servers. *ACM MULITIMEDIA*, pp. 219–226, 1997.
10. M. J. Luczak and E. Upfal. Reducing network congestion and blocking probability through balanced allocation. *FOCS*, pp. 587–595, 1999.
11. M. Mitzenmacher. Load balancing and density dependent jump Markov processes. *FOCS*, pp. 213–222, 1996.
12. M. Mitzenmacher, A. W. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. In *Handbook of Randomized Computing*, Rajasekaran et al., eds., Volume I, pp. 255-312, Kluwer Academic Press, 2001.
13. Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of Markov chains and the analysis of iterative load-balancing schemes. *FOCS*, pp. 694–703, 1998.
14. P. Sanders. Asynchronous scheduling of redundant disk arrays. *SPAA*, pp. 89–98, 2000.
15. P. Sanders. Reconciling simplicity and realism in parallel disk models. *SODA*, pp. 67–76, 2001.
16. P. Sanders, S. Egner, and J. Korst. Fast concurrent access to parallel disks. *Algorithmica*, 35(1):21–55, 2003. A preliminary version appeared in *SODA*, pp. 849–858, 2000.
17. L. A. M. Schoenmakers. A new algorithm for the recognition of series parallel graphs. Technical Report CS-R9504, CWI — Centrum voor Wiskunde en Informatica, January 1995.
18. B. Vöcking. How asymetry helps load balancing. *FOCS*, pp. 131–141, 1999.