Tight Bounds for Parallel Randomized Load Balancing

[Extended Abstract] *

Christoph Lenzen
Computer Engineering
and Networks Laboratory
ETH Zurich
Switzerland
lenzen@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering
and Networks Laboratory
ETH Zurich
Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

We explore the fundamental limits of distributed balls-into-bins algorithms, i.e., algorithms where balls act in parallel, as separate agents. This problem was introduced by Adler et al., who showed that *non-adaptive* and *symmetric* algorithms cannot reliably perform better than a maximum bin load of $\Theta(\log\log n/\log\log\log n)$ within the same number of rounds. We present an adaptive symmetric algorithm that achieves a bin load of two in $\log^* n + \mathcal{O}(1)$ communication rounds using $\mathcal{O}(n)$ messages in total. Moreover, larger bin loads can be traded in for smaller time complexities. We prove a matching lower bound of $(1-o(1))\log^* n$ on the time complexity of symmetric algorithms that guarantee small bin loads at an asymptotically optimal message complexity of $\mathcal{O}(n)$. The essential preconditions of the proof are (i) a limit of $\mathcal{O}(n)$ on the total number of messages sent by the algorithm and (ii) anonymity of bins, i.e., the port numberings of balls are not globally consistent. In order to show that our technique yields indeed tight bounds, we provide for each assumption an algorithm violating it, in turn achieving a constant maximum bin load in constant time.

As an application, we consider the following problem. Given a fully connected graph of n nodes, where each node needs to send and receive up to n messages, and in each round each node may send one message over each link, deliver all messages as quickly as possible to their destinations. We give a simple and robust algorithm of time complexity $\mathcal{O}(\log^* n)$ for this task and provide a generalization to the case where all nodes initially hold arbitrary sets of messages. Completing the picture, we give a less practical, but asymptotically optimal algorithm terminating within $\mathcal{O}(1)$ rounds. All these bounds hold with high probability.

*A full version of this paper is available as technical report [23].

STOC'11, June 6–8, 2011, San Jose, California, USA.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs among Complexity Measures

General Terms

Algorithms, Theory

Keywords

randomized allocation, load balancing, asymptotic bounds

1. INTRODUCTION

Some argue that in the future understanding parallelism and concurrency will be as important as understanding sequential algorithms and data structures. Indeed, clock speeds of microprocessors have flattened about 5-6 years ago. Ever since, efficiency gains must be achieved by parallelism, in particular using multi-core architectures and parallel clusters.

Unfortunately, parallelism often incurs a coordination overhead. To be truly scalable, also coordination must be parallel, i.e., one cannot process information sequentially, or collect the necessary coordination information at a single location. A striking and fundamental example of coordination is load balancing, which occurs on various levels: canonical examples are job assignment tasks such as sharing work load among multiple processors, servers, or storage locations, but the problem also plays a vital role in e.g. low-congestion circuit routing, channel bandwidth assignment, or hashing, cf. [33].

A common archetype of all these tasks is the well-known balls-into-bins problem: Given n balls and n bins, how can one place the balls into the bins quickly while keeping the maximum bin load small? As in other areas where centralized control must be avoided (sometimes because it is impossible), the key to success is *randomization.* Adler et al. [1] devised parallel randomized algorithms for the problem whose running times and maximum bin loads are essentially doubly-logarithmic. They provide a lower bound which is asymptotically matching the upper bound. However, their lower bound proof requires two critical restrictions: algorithms must (i) break ties symmetrically and (ii) be non-adaptive, i.e., each ball restricts itself to a fixed number of candidate bins before communication starts.

In this work, we present a simple *adaptive* algorithm achieving a maximum bin load of two within $\log^* n + \mathcal{O}(1)$ rounds of communication, with high probability. This is achieved with $\mathcal{O}(1)$ messages in expectation per ball and bin, and $\mathcal{O}(n)$ messages in total. We show that our method is *robust* to model variations. In particular, it seems that being adaptive helps solving some practical problems elegantly and efficiently; bluntly, if messages are lost, they will simply be retransmitted. Moreover, our algorithms can be generalized to the case where the number of balls differs from the number of bins.

Complementing this result, we prove that—given the constraints on bin load and communication complexity—the running time of our algorithm is $(1 + o(1))$-optimal for symmetric algorithms. Our bound necessitates a new proof technique; it is not a consequence of an impossibility to gather reliable information in time (e.g. due to asynchronicity, faults, or explicitly limited local views of the system), rather it emerges from bounding the total amount of communication. Thus, we demonstrate that breaking symmetry to a certain degree, i.e., reducing entropy far enough to guarantee small bin loads, comes at a cost exceeding the apparent minimum of $\Omega(n)$ total bits and $\Omega(1)$ rounds. In this light, a natural question to pose is how much initial entropy is required for the lower bound to hold. We show that the crux of the matter is that bins are initially anonymous, i.e., balls do not know globally unique addresses of the bins. For the problem where bins are consistently labeled $1, \ldots, n$, we give an algorithm running in constant time that sends $\mathcal{O}(n)$ messages, yet achieves a maximum bin load of three. Furthermore, if a small-factor overhead in terms of messages is tolerated, the same is also possible without a global address space. Therefore, our work provides a complete classification of the parallel complexity of the balls-into-bins problem.

Our improvements on parallel balls-into-bins are developed in the context of a parallel load balancing application involving an even larger amount of concurrency. We consider a system with n well-connected processors, i.e., each processor can communicate directly with every other processor.[1] However, there is a bandwidth limitation of one message per unit of time on each connection. Assume that each processor needs to send (and receive) up to n messages, to arbitrary destinations. In other words, there are up to n^2 messages that must be delivered, and there is a communication system with a capacity of n^2 messages per time unit. What looks trivial from an "information theoretic" point of view becomes complicated if message load is not well balanced, i.e., if only few processors hold all the n messages for a single recipient. If the processors knew of each others' intentions, they could coordinatedly send exactly one of these messages to each processor, which would subsequently relay it to the target node. However, this simple scheme is infeasible for reasonable message sizes: In order to collect the necessary information at a single node, it must receive up to n^2 numbers over its n communication links.

In an abstract sense, the task can be seen as consisting of n balls-into-bins problems which have to be solved *concurrently*. We show that this parallel load balancing problem can be solved in $\mathcal{O}(\log^* n)$ time, with high probability, by a generalization of our symmetric balls-into-bins algorithm. The resulting algorithm inherits the robustness of our balls-into-bins technique, for instance it can tolerate a constant fraction of failing edges. Analogously to the balls-into-bins setting, an optimal bound of $\mathcal{O}(1)$ on the time complexity can be attained, however, the respective algorithm is rather impractical and will be faster only for entirely unrealistic values of n. We believe that the parallel load balancing problem will be at the heart of future distributed systems and networks, with applications from scientific computing to overlay networks.

[1]This way, we can study the task of load balancing independently of routing issues.

2. RELATED WORK

Probably one of the earliest applications of randomized load balancing has been hashing. In this context, Gonnet [15] proved that when throwing n balls uniformly and independently at random (u.i.r.) into n bins, the fullest bin has load $(1 + o(1)) \log n / \log \log n$ in expectation. It is also common knowledge that the maximum bin load of this simple approach is $\Theta(\log n / \log \log n)$ with high probability (w.h.p.)[2] [10].

With the growing interest in parallel computing, since the beginning of the nineties the topic received increasingly more attention. Karp et al. [17] demonstrated for the first time that two random choices are superior to one. By combining two (possibly not fully independent) hashing functions, they simulated a parallel random access machine (PRAM) on a distributed memory machine (DMM) with a factor $\mathcal{O}(\log \log n \log^* n)$ overhead; in essence, their result was a solution to balls-into-bins with maximum bin load of $\mathcal{O}(\log \log n)$ w.h.p. Azar et al. [3] generalized their result by showing that if the balls choose sequentially from $d \geq 2$ u.i.r. bins greedily the currently least loaded one, the maximum load is $\log \log n / \log d + \mathcal{O}(1)$ w.h.p.[3] They prove that this bound is stochastically optimal in the sense that any other strategy to assign the balls majorizes[4] their approach. The expected number of bins each ball queries during the execution of the algorithm was later improved to $1 + \varepsilon$ (for any constant $\varepsilon > 0$) by Czumaj and Stemann [8]. This is achieved by placing each ball immediately if the load of an inspected bin is not too large, rather then always querying d bins.

So far the question remained open whether strong upper bounds can be achieved in a parallel setting. Adler et al. [1] answered this affirmatively by devising a parallel greedy algorithm obtaining a maximum load of $\mathcal{O}(d + \log \log n / \log d)$ within the same number of rounds w.h.p. Thus, choosing $d \in \Theta(\log \log n / \log \log \log n)$, the best possible maximum bin load of their algorithm is $\mathcal{O}(\log \log n / \log \log \log n)$. On the other hand, they prove that a certain subclass of algorithms cannot perform better with probability larger

[2]I.e., with probability at least $1 - 1/n^c$ for a freely choosable constant $c > 0$.

[3]There is no common agreement on the notion of w.h.p. Frequently it refers to probabilities of at least $1 - 1/n$ or $1 - o(1)$, as so in the work of Azar et al.; however, their proof also provides their result w.h.p. in the sense we use throughout this paper.

[4]Roughly speaking, this means that any other algorithm is as least as likely to produce bad load vectors as the greedy algorithm. An n-dimensional load vector is worse than another, if after reordering the components of both vectors descendingly, any partial sum of the first $i \in \{1, \ldots, n\}$ entries of the one vector is greater or equal to the corresponding partial sum of the other.

Table 1: Comparison of parallel algorithms for $m = n$ balls. Committing balls into bins counts as half a round with regard to time complexity.

algorithm	symmetric	adaptive	choices	rounds	maximum bin load	messages
naive [15]	yes	no	1	0.5	$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$	n
par. greedy [1]	yes	no	2	2.5	$\mathcal{O}\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$\mathcal{O}(n)$
par. greedy [1]	yes	no	$\Theta\left(\frac{\log\log n}{\log\log\log n}\right)$	$\Theta\left(\frac{\log\log n}{\log\log\log n}\right)$	$\mathcal{O}\left(\frac{\log\log n}{\log\log\log n}\right)$	$\mathcal{O}\left(\frac{n\log\log n}{\log\log\log n}\right)$
collision [38]	yes	no	2	$r + 0.5$	$\mathcal{O}\left(\left(\frac{\log n}{\log\log n}\right)^{1/r}\right)$	$\mathcal{O}(n)$
$\mathcal{A}_b^2$	yes	yes	$\mathcal{O}(1)$ (exp.)	$\log^* n + \mathcal{O}(1)$	2	$\mathcal{O}(n)$
$\mathcal{A}_b(r)$	yes	yes	$\mathcal{O}(1)$ (exp.)	$r + \mathcal{O}(1)$	$\frac{\log^{(r)} n}{\log^{(r+1)} n} + r + \mathcal{O}(1)$	$\mathcal{O}(n)$
$\mathcal{A}_c(l)$	yes	yes	$\mathcal{O}(l)$ (exp.)	$\log^* n - \log^* l + \mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(ln)$
$\mathcal{A}(\sqrt{\log n})$	no	yes	$\mathcal{O}(1)$ (exp.)	$\mathcal{O}(1)$	3	$\mathcal{O}(n)$

than $1 - 1/\operatorname{polylog} n$. The main characteristics of this subclass are that algorithms are *non-adaptive*, i.e., balls have to choose a fixed number of d candidate bins before communication starts, and *symmetric*, i.e., these bins are chosen u.i.r. Moreover, communication takes place only between balls and their candidate bins. In this setting, Adler et al. show also that for any constant values of d and the number of rounds r the maximum bin load is $\Omega((\log n/\log\log n)^{1/r})$ with constant probability. Recently, Even and Medina extended their bounds to a larger spectrum of algorithms by removing some artificial assumptions [12]. A matching algorithm was proposed by Stemann [38], which for $d = 2$ and $r \in \mathcal{O}(\log\log n)$ achieves a load of $\mathcal{O}((\log n/\log\log n)^{1/r})$ w.h.p.; for $r \in \Theta(\log\log n/\log\log\log n)$ this implies a constantly bounded bin load. Even and Medina also proposed a 2.5-round "adaptive" algorithm [11].[5] Their synchronous algorithm uses a constant number of choices and exhibits a maximum bin load of $\Theta(\sqrt{\log n/\log\log n})$ w.h.p., i.e., exactly the same characteristics as parallel greedy with 2.5 rounds and two choices. In comparison, within this number of rounds our technique is capable of achieving bin loads of $(1 + o(1))\log\log n/\log\log\log n$ w.h.p.[6] See Table 1 for a comparison of our results to parallel algorithms. Our adaptive algorithms outperform all previous solutions for the whole range of parameters.

Given the existing lower bounds, since then the only possibility for further improvement has been to search for non-adaptive or asymmetric algorithms. Vöcking [40] introduced the sequential "always-go-left" algorithm which employs asymmetric tie-breaking in order to improve the impact of the number of possible choices d from logarithmic to linear. Furthermore, he proved that dependency of random choices does not offer asymptotically better bounds. His upper bound holds also true if only two bins are chosen randomly, but for each choice $d/2$ consecutive bins are queried [18]. Table 2 summarizes sequential balls-into-bins algorithms. Note that not all parallel algorithms can also be run sequentially.[7] However, this is true for our protocols; our approach translates to a simple sequential algorithm competing in performance with the best known results [8, 40]. This algorithm could be interpreted as a greedy algorithm with $d = \infty$.

Most of the mentioned work considers also the general case of $m \neq n$. If $m > n$, this basically changes expected loads to m/n, whereas values considerably smaller than n (e.g. $n^{1-\varepsilon}$) admit constant maximum bin load. It is noteworthy that for $d \geq 2$ the imbalance between the most loaded bins and the average load is $\mathcal{O}(\log\log n/\log d)$ w.h.p. irrespective of m. Recently, Peres et al. [36] proved a similar result for the case where "$d = 1 + \beta$" bins are queried, i.e., balls choose with constant probability $\beta \in (0, 1)$ the least loaded of two bins, otherwise uniformly at random. In this setting, the imbalance becomes $\Theta((\log n)/\beta)$ w.h.p.

In addition, quite a few variations of the basic problem have been studied. Since resources often need to be assigned to dynamically arriving tasks, infinite processes have been considered (e.g. [3, 8, 29, 30, 31, 38, 40]). In [32] it is shown that, in the sequential setting, memorizing good choices from previous balls has similar impact as increasing the number of fresh random choices. Awerbuch et al. [2] studied arbitrary L_p norms instead of the maximum bin load (i.e., the L_∞ norm) as quality measure, showing that the greedy strategy is p-competitive to an offline algorithm. Several works addressed weighted balls (e.g. [6, 7, 20, 36, 39]) in order to model tasks of varying resource consumption. The case of heterogeneous bins was examined as well [41]. In recent years, balls-into-bins has also been considered from a game theoretic point of view [5, 19].

Results related to ours have been discovered before for hashing problems. A number of publications presents algorithms with running times of $\mathcal{O}(\log^* n)$ (or very close) in PRAM models [4, 14, 27, 28]. At the heart of these routines as well as our balls-into-bins solutions lies the idea to use an in each iteration exponentially growing share of the available resources to deal with the remaining keys or bins,

[5] If balls cannot be allocated, they get an additional random choice. However, one could also give all balls this additional choice and let some of them ignore it, i.e., this kind of adaptivity cannot circumvent the lower bound.

[6] This follows by setting $a := (1 + \varepsilon)\log\log n/\log\log\log n$ (for arbitrary small $\varepsilon > 0$) in the proof of Corollary 3.4; we get that merely $n/(\log n)^{1+\varepsilon}$ balls remain after one round, which then can be delivered in 1.5 more rounds w.h.p. using $\mathcal{O}(\log n)$ requests per ball.

[7] Stemann's collision protocol, for instance, requires bins to accept balls only if a certain number of pending requests is not exceeded. Thus the protocol cannot place balls until all random choices are communicated.

Table 2: Comparison of sequential algorithms for $m = n$ balls.

algorithm	symmetric	adaptive	choices	maximum bin load	bin queries
naive [15]	yes	no	1	$\mathcal{O}\big(\frac{\log n}{\log\log n}\big)$	n
greedy [3]	yes	no	$d \geq 2$	$\frac{\log\log n}{\log d} + \mathcal{O}(1)$	$\mathcal{O}(dn)$
always-go-left [40]	no	no	$d \geq 2$	$\mathcal{O}\big(\frac{\log\log n}{d}\big)$	$\mathcal{O}(dn)$
adpt. greedy [8]	yes	yes	$1 + o(1)$ (exp.); at most $d \geq 2$	$\mathcal{O}\big(\frac{\log\log n}{\log d}\big)$	$(1 + o(1))n$
$\mathcal{A}_{\text{seq}}$	yes	yes	$\mathcal{O}(1)$ (exp.)	2	$(2 + o(1))n$

respectively. Implicitely, this approach already occured in previous work by Raman [37]. For a more detailed review of these papers, we refer the interested reader to [16]. Despite differences in the models, our algorithms and proofs exhibit quite a few structural similarities to the ones applicable to hashing in PRAM models. From our point of view, there are two main differences distinguishing our upper bound results on symmetric algorithms. Firstly, the parallel balls-into-bins model permits to use the algorithmic idea in its most basic form. Hence, our presentation focuses on the properties decisive for the $\log^* n + \mathcal{O}(1)$ complexity bound of the basic symmetric algorithm. Secondly, our analysis shows that the core technique is highly robust and can therefore tolerate a large number of faults.

The lower bound by Adler et al. (and the generalization by Even and Medina) is stronger than our lower bound, but it applies to algorithms which are severely restricted in their abilities only. Essentially, these restrictions uncouple the algorithm's decisions from the communication pattern; in particular, communication is restricted to an initially fixed random graph, where each ball contributes d edges to u.i.r. bins. This prerequisite seems reasonable for systems where the initial communication overhead is large. In general, we find it difficult to motivate that a non-constant number of communication rounds is feasible, but an initially fixed set of bins may be contacted only. In contrast, our lower bound also holds for adaptive algorithms. In fact, it even holds for algorithms that allow for address forwarding, i.e., balls may contact any bin deterministically after obtaining its globally unique address.[8] In other words, it arises from the assumption that bins are (initially) anonymous (cf. Problems 4.1 and 4.2), which fits a wide range of real-world systems.

Like Linial in his seminal work on 3-coloring the ring [24], we attain a lower bound of $\Omega(\log^* n)$ on the time required to solve the task efficiently. This connection is more than superficial, as both bounds essentially arise from a symmetry breaking problem. However, Linial's argument uses a highly symmetric ring topology.[9] This is entirely different from our setting, where any two parties may potentially exchange information. Therefore, we cannot argue on the basis that nodes will learn about a specific subset of the global state contained within their local horizon only. Instead, the random decisions of a balls-into-bins algorithm define a graph describing the flow of information. This graph is not a simple random graph, as the information gained by this communication feeds back to its evolution over time, i.e., future communication may take the local topology of its current state into account.

A different lower bound technique is by Kuhn et al. [21], where a specific locally symmetric, but globally asymmetric graph is constructed to render a problem hard. Like in our work, [21] restricts its arguments to graphs which are locally trees. The structure of the graphs we consider imposes to examine subgraphs which are trees as well; subgraphs containing cycles occur too infrequently to constitute a lower bound. The bound of $\Omega(\log^* n)$ from [14], applicable to hashing in a certain model, which also argues about trees, has even more in common with our result. However, neither of these bounds needs to deal with the difficulty that the algorithm may influence the evolution of the communication graph in a complex manner. In [21], input and communication graph are identical and fixed; in [14], there is also no adaptive communication pattern, as essentially the algorithm may merely decide on how to further separate elements that share the same image under the hash functions applied to them so far.

Various other techniques for obtaining distributed lower bounds exist [13, 26], however, they are not related to our work. If graph-based, the arguments are often purely information theoretic, in the sense that some information must be exchanged over some bottleneck link or node in a carefully constructed network with diameter larger than two [25, 35]. In our setting, such information theoretic lower bounds will not work: Any two balls may exchange information along n edge-disjoint paths of length two, as the graph describing which edges could *potentially* be used to transmit a message is complete bipartite. In some sense, this is the main contribution of this paper: We show the existence of a coordination bottleneck in a system without a physical bottleneck.

The remainder of this extended abstract is organized as follows. We start out with the presentation of a simple symmetric balls-into-bins algorithm achieving a time complexity of $\log^* n + \mathcal{O}(1)$ at the same maximum bin load. From this basic technique a number of results are inferred. In particular, the applied proof method allows for solving the aforementioned load balancing problem (Problem 3.6) in $\mathcal{O}(\log^* n)$ rounds. In Section 4, after classifying some parallel balls-into-bins models, we sketch the proof of the lower bound demonstrating the $(1 + o(1))$-optimality of the given symmetric algorithm. Subsequently, in Section 5, we show that if any of the prerequisites of the lower bound is dropped, an according constant-time constant-load solution can be devised. As a consequence, also Problem 3.6 permits a constant-time randomized algorithm. Finally, Section 6 draws some conclusions. Due to lack of space, in the forthcoming we will omit all formal proofs in favor of an informal

[8] This address is initially known to the respective bin only, but it may be forwarded during the course of an algorithm.

[9] This general approach to argue about a simple topology has been popular when proving lower bounds [9, 22, 34].

presentation of the key ideas and concepts. The proofs are available in the accompanying technical report [23].

3. SYMMETRIC ALGORITHMS

In this section, we are going to present our basic symmetric balls-into-bins technique and related results. "Symmetric" here essentially means that initially all bins "look the same", i.e., there is no consistent labeling of the bins that is known to all balls.

3.1 Model

The system consists of n bins and n balls, and we assume it to be fault-free. We employ a synchronous message passing model, where one round consists of the following steps:

1. Balls perform (finite, but otherwise unrestricted) local computations and send messages to arbitrary bins.
2. Bins receive these messages, do local computations, and send messages to any balls they have been contacted by in this or earlier rounds.
3. Balls receive these messages and may commit to a bin (and terminate).[10]

Moreover, balls and bins each have access to an unlimited source of unbiased random bits, i.e., all algorithms are randomized. The considered task now can be stated concisely.

PROBLEM 3.1 (PARALLEL BALLS-INTO-BINS). *We want to place each ball into a bin. The goals are to minimize the total number of rounds until all balls are placed, the maximum number of balls placed into a bin, and the amount of involved communication.*

Essentially, our model is the one of Adler et al. [1], but without the assumption that balls need to choose a fixed set of communication partners in advance, i.e., we consider *adaptive* algorithms.

3.2 Basic Algorithm

Our first algorithm illustrates the fundamental difference introduced by adaptivity best. The strategy is very simple. First, we try to place the balls with caution, i.e., a small number of messages per ball. A large fraction of the balls will be successfully placed even if bins accept only a single ball. This holds w.h.p., i.e., essentially it is guaranteed. Thus, it is safe with regard to message complexity to increase the number of bins each remaining ball contacts in order to find a suitable bin. It turns out that this trivial observation already leads to an efficient algorithm.

Set $k(1) := 1$ and $i := 1$. Algorithm $\mathcal{A}_b$ executes the following loop until termination:

1. Balls contact $\lfloor k(i) \rfloor$ u.i.r. bins, requesting permission to be placed into them.
2. Each bin admits permission to one of the requesting balls (if it received any) and declines all other requests.
3. Any ball receiving at least one permission chooses an arbitrary of the respective bins to be placed into, informs it, and terminates.
4. Set $k(i+1) := \min\left\{k(i)e^{\lfloor k(i) \rfloor/5}, \sqrt{\log n}\right\}$ and $i := i+1$.

THEOREM 3.2. *$\mathcal{A}_b$ solves Problem 3.1, guaranteeing the following properties:*

- *It terminates after $\log^* n + \mathcal{O}(1)$ rounds w.h.p.*
- *Each bin in the end contains at most $\log^* n + \mathcal{O}(1)$ balls w.h.p.*
- *The total number of messages is in $\mathcal{O}(n)$ w.h.p.*
- *Balls send and receive $\mathcal{O}(1)$ messages in expectation and $\mathcal{O}(\sqrt{\log n})$ many w.h.p.*
- *Bins send and receive $\mathcal{O}(1)$ messages in expectation and $\mathcal{O}(\log n / \log\log n)$ many w.h.p.*

To see why the algorithm terminates this fast, assume for the moment that in each round, a constant fraction of all requests issued in Step 1 of the algorithm is chosen uniformly at random and accepted in Step 2. Thus, each request a single ball sends has an independent constant probability of being answered affirmatively, implying that the probability of a ball surviving round i of the algorithm is exponentially small in $k(i)$. As long as the number of balls still is fairly large, this implies that the number of surviving balls drops by a factor of $e^{-\Omega(k(i))}$ w.h.p. This explains the choice of $k(i+1)$ in the last step, as we can increase k rapidly without risking to generate too many messages in total. We can cap the growth of k at $\sqrt{\log n}$ because if merely $e^{-\Omega(\sqrt{\log n})}n$ balls need still to be placed, the success probability of a single request is $1 - e^{-\Omega(\sqrt{\log n})}$. Thus each residual ball is placed with probability $1 - e^{-\Omega(\log n)}$ in each subsequent round, implying termination within a constant number of rounds w.h.p.

The formal proof is mostly dealing with the technicality that the considered random events are not (entirely) independent. Nonetheless, it reveals a remarkable property. We show that a uniformly random constant fraction of all requests is accepted, and that this is the case even if we consider the bins that receive a single request only. This observation implies that Theorem 3.2 (and its corollaries) are highly resilient to model variations and faults. For instance, the asymptotic bounds hold if a constant fraction of all messages is lost[11] or if whenever a bin receives more than one request all respective messages are lost,[12] and all our algorithms work equally well in asynchronous systems.

3.3 Variations

Next, we discuss a number of variations of the basic algorithm. For instance, we can ensure a bin load of at most two without increasing the time complexity.

[10] Note that (for reasonable algorithms) this step does not interfere with the other two. Hence, the literature typically takes each execution of the first to steps as one round and accounts for this step as "half a round" when stating the time complexity of balls-into-bins algorithms; we adopted this convention in the related work section.

[11] As long as these faults are not correlated in a bad way, e.g. completely cutting off individual balls from communication.

[12] This could e.g. model interference in a system employing wireless communication.

COROLLARY 3.3. *We modify $\mathcal{A}_b$ into $\mathcal{A}_b^2$ by ruling that any bins having already accepted two balls refuse any further requests in Step 2, and in Step 4 we set*

$$k(i+1) := \min\left\{k(i)e^{\lfloor k(i)\rfloor/10}, \log n\right\}.$$

Then the statements of Theorem 3.2 remain true except that balls now send w.h.p. $\mathcal{O}(\log n)$ messages instead of $\mathcal{O}(\sqrt{\log n})$. In turn, the maximum bin load of the algorithm becomes two.

On the other hand, we can enforce a constant time complexity at the expense of an increase in maximum bin load.

COROLLARY 3.4. *For any $r \in \mathbb{N}$, $\mathcal{A}_b$ can be modified into an Algorithm $\mathcal{A}_b(r)$ that guarantees a maximum bin load of $\log^{(r)} n / \log^{(r+1)} n + r + \mathcal{O}(1)$ w.h.p. and terminates within $r + \mathcal{O}(1)$ rounds w.h.p. Its message complexity respects the same bounds as the one of $\mathcal{A}_b$.*

Algorithm $\mathcal{A}_b^2$ is related to a simple sequential greedy algorithm that queries for each ball sufficiently many bins to find one that has load less than two.

LEMMA 3.5. *An adaptive sequential balls-into-bins algorithm $\mathcal{A}_{seq}$ exists guaranteeing a maximum bin load of two, requiring at most $(2+o(1))n$ random choices and bin queries w.h.p.*

3.4 An Application: Information Distribution

Consider a fully connected system of n nodes. We want to solve the following problem.

PROBLEM 3.6 (INFORMATION DISTRIBUTION TASK). *Each node $v \in V$ is given a (finite) set of messages*

$$\mathcal{S}_v = \left\{m_v^i \mid i \in I_v\right\}$$

with destinations $d(m_v^i) \in V$, $i \in I_v$. Each such message explicitly contains $d(m_v^i)$, i.e., messages have size $\Omega(\log n)$. Moreover, messages can be distinguished (e.g., by also including the sender's identifier and the position in an internal ordering of the messages of that sender). The goal is to deliver all messages to their destinations, minimizing the total number of rounds. By

$$\mathcal{R}_v := \left\{m_w^i \in \bigcup_{w\in V} \mathcal{S}_w \,\middle|\, d(m_w^i) = v\right\}$$

we denote the set of messages a node $v \in V$ shall receive. We abbreviate $M_s := \max_{v\in V} |\mathcal{S}_v|$ and $M_r := \max_{v\in V} |\mathcal{R}_v|$, i.e., the maximum numbers of messages a single node needs to send or receive, respectively.

This problem can be solved efficiently by basically applying Algorithm $\mathcal{A}_b$ in parallel to each set of messages $\mathcal{R}_v$ with the goal to distribute these messages as evenly as possible. Subsequently they can be sent to their destinations quickly. For the purpose of this extended abstract, we state the following bound only and refer the interested reader to the technical report for more details.

THEOREM 3.7. *Problem 3.6 can be solved in*

$$\mathcal{O}\left(\frac{M_s + M_r}{n}\right)$$

rounds w.h.p.

4. LOWER BOUND

In this section, we will derive our lower bound on the parallel complexity of the balls-into-bins problem. After classifying different degrees of anonymity of bins, we proceed by sketching the proof technique, which we believe to be of independent interest.

4.1 Bin Anonymity

A natural restriction for algorithms solving Problem 3.1 is to assume that random choices cannot be biased, i.e., bins are completely anonymous. This is formalized by the following definition.

PROBLEM 4.1 (SYMMETRIC BALLS-INTO-BINS). *We call an instance of Problem 3.1* symmetric parallel balls-into-bins problem*, if balls and bins identify each other by u.i.r. port numberings. We call an algorithm solving this problem* symmetric.

Thus, whenever a ball executing a symmetric balls-into-bins algorithm contacts a new bin, it essentially draws uniformly at random. This is a formalization of the central aspect of the notion of symmetry used by Adler et al. [1]. Note that all algorithms from Section 3 are symmetric.

The main result states that the time complexity of symmetric algorithms cannot be improved by any constant factor.[13] What is more, our lower bound holds for a communication model that is even stronger.

PROBLEM 4.2 (ACQUAINTANCE BALLS-INTO-BINS). *We call an instance of Problem 3.1* acquaintance balls-into-bins problem*, if the following holds. Initially, bins are anonymous, i.e., balls identify bins by u.i.r. port numberings. However, once a ball contacts a bin, it learns its globally unique address, by which it can be contacted reliably. Thus, by means of forwarding addresses, balls can learn to contact specific bins directly. The addresses are abstract in the sense that they can be used for this purpose only.*[14] *We call an algorithm solving this problem* acquaintance algorithm.

For this class of algorithm the following lower bound can be shown.

THEOREM 4.3. *Any acquaintance algorithm sending $\mathcal{O}(n)$ messages in total and no more than λn messages per node w.h.p. (where $\lambda < 1$ is a constant) either incurs a maximum bin load of more than $L \in \mathbb{N}$ w.h.p. or runs for at least $(1 - o(1)) \log^* n - \log^* L$ rounds, irrespective of the size of messages. This holds also if bins may contact other bins.*

4.2 Proof Outline

We need to bound the amount of information balls can collect during the course of the algorithm. As balls may contact any bins they heard of, this is described by exponentially growing neighborhoods in the graph where edges are created whenever a ball picks a communication partner at random.

[13] Unless considerably more communication is used; this is discussed in Section 5 in more detail.

[14] This requirement is introduced to prohibit the use of these addresses for symmetry breaking, as is possible for asymmetric algorithms. One may think of the addresses e.g. as being random from a large universe, or the address space might be entirely unknown to the balls.

DEFINITION 4.4 (BALLS-INTO-BINS GRAPHS). *The (bipartite and simple)* balls-into-bins graph $G_{\mathcal{A}}(r)$ *associated with an execution of the acquaintance algorithm* $\mathcal{A}$ *running for* $r \in \mathbb{N}$ *rounds is constructed as follows. The node set* $V := V_\circ \cup V_\sqcup$ *consists of* $|V_\circ| = |V_\sqcup| = n$ *bins and balls. In each round* $i \in \{1, \dots, r\}$*, each ball* $b \in V_\circ$ *adds an edge connecting itself to bin* $v \in V_\sqcup$ *if* b *contacts* v *by a random choice in that round. By* $E_{\mathcal{A}}(i)$ *we denote the edges added in round* i *and* $G_{\mathcal{A}}(t) = (V, \cup_{i=1}^{t} E_{\mathcal{A}}(i))$ *is the graph containing all edges added until and including round* t*.*

The proof argues about certain symmetric subgraphs in which not all balls can decide on bins concurrently without incurring large bin loads. As can be seen by a quick calculation, any connected subgraph containing a cycle is unlikely to occur frequently. For an adaptive algorithm, it is possible that balls make a larger effort in terms of sent messages to break symmetry once they observe a "rare" neighborhood. Therefore, it is mandatory to reason about subgraphs which are trees.

We would like to argue that any algorithm suffers from generating a large number of trees of uniform ball and bin degrees. If we root such a tree at an arbitrary bin, balls cannot distinguish between their parents and children according to this orientation. Thus, they will decide on a bin that is closer to the root with probability inverse proportional to their degree. If bin degrees are by factor $f(n)$ larger than ball degrees, this will result in an expected bin load of the root of $f(n)$. However, this line of reasoning is too simple. As edges are added to G in different rounds, these edges can be distinguished by the balls. Moreover, even if several balls observe the same local topology in a given round, they may randomize the number of bins they contact during that round, destroying the uniformity of degrees. For these reasons, we (i) rely on a more complicated tree in which the degrees are a function of the round number and (ii) show that for every acquaintance algorithm a stronger algorithm exists that indeed generates many such trees w.h.p.

Due to space constraints and the fact that the definition of these derived algorithms is very technical, in this extended abstract we confine ourselves to the presentation of the considered trees. In these structures, all involved balls up to a certain distance from the root see exactly the same topology. This means that (i) in each round, all involved balls created exactly the same number of edges by contacting bins randomly, (ii) each bin has a degree that depends on the round when it was contacted first only, (iii) all edges of such bin are formed in exactly this round, and (iv) this scheme repeats itself up to a distance that is sufficiently large for the balls not to see any irregularities that might help in breaking symmetry. These properties are satisfied by the following recursively defined tree structure.

DEFINITION 4.5 (LAYERED $(\Delta^\sqcup, \Delta^\circ, D)$-TREES). *A* layered $(\Delta^\sqcup, \Delta^\circ, D)$-tree *of* $\ell \in \mathbb{N}_0$ levels *rooted at bin* R *is defined as follows, where* $\Delta^\sqcup = (\Delta_1^\sqcup, \dots, \Delta_\ell^\sqcup)$ *and* $\Delta^\circ = (\Delta_1^\circ, \dots, \Delta_\ell^\circ)$ *are the vectors of bins' and balls' degrees on different levels, respectively.*

If $\ell = 0$*, the "tree" is simply a single bin. If* $\ell > 0$*, the subgraph of* $G_{\mathcal{A}}(\ell)$ *induced by* $\mathcal{N}_R^{(2D)}$ *is a tree, where ball degrees are uniformly* $\sum_{i=1}^{\ell} \Delta_i^\circ$*. Except for leaves, a bin that is added to the structure in round* $i \in \{1, \dots, \ell\}$ *has degree* $\Delta_i^\sqcup$ *with all its edges in* $E_{\mathcal{A}}(i)$*. See Figure 1 for an illustration.*

Intuitively, layered trees are crafted to present symmetric neighborhoods to nodes which are not aware of leaves. Thus, if bins' degrees are large compared to balls' degrees, not all balls can decide simultaneously without risking to overload bins.

For the notion of layered trees to be of any use, one needs to show that these structures indeed occur for a non-trivial number of rounds when one of the aforementioned stronger algorithms is run. This is an intricate problem, as one can neither argue by properties of random graphs only, nor is it sufficient to rely on indistinguishability type arguments alone. Roughly, we reason along the following lines. Initially, balls have no information about the system. Thus, w.h.p. a large fraction of the balls will contact a constant number of bins. This will result in $\Omega(n)$ many layered trees of depth one. Since the topology of layered trees looks identical to all nodes sufficiently far from leaves, these nodes cannot (all) contact many bins with a large probability without violating the bounds on message complexity. Thus, we can control the number of new edges these balls create. If all such balls contact the same number of bins, this permits to show that it is likely that many layered trees of two levels will be part of the balls-into-bins graph after two rounds of communication. An acquaintance algorithm now could avoid this by choosing a different number of new contacts for the balls in such a structure. We address this by permitting each of these balls to contact as many bins as *all* balls in a tree of one level *together*. Obviously, this will make it only easier for an algorithm to assign the balls to bins, yet we can still show that many layered trees of two levels will occur. This is the key characteristic of the class of stronger algorithms we examine; the remaining properties are concerned with technicalities. We refer to the accompanying technical report for further details [23] and the full proof.

In summary, we flip back and forth between two arguments. On the one hand, in each round for a considerable fraction of the balls the system looks locally "critical" in the sense that symmetry remains unbroken and indistinguishability constraints the amount of communication that can be safely invested to break symmetry. On the other hand, if not too much communication is used, many symmetric structures will remain after communication took place. Naturally, this situation is quite unstable: If we have T disjoint layered trees in the graph in a given round, the balls in these trees may use up to n/T messages on average to break symmetry. Calculation shows that this implies a tower-like growth of the size of layered trees and a respective decrease in their number. In other words, symmetry cannot be overcome everywhere for $(1 - o(1)) \log^* n$ rounds.

So, why do layered trees indeed prevent balls from deciding on bins or enforce large bin loads? Essentially, this can already be seen by example of an "ordinary" tree. Since the tree is deeper than the local horizon of the balls, they cannot root the tree, i.e., decide consistently to be placed into bins further away from an imagined root. If in such a tree bins have large degrees in comparison to balls, this means that bins have an accordingly large expected load. By symmetry arguments it can be shown that this implies that it is indeed likely that a bin gets overloaded.

5. CONSTANT-TIME ALGORITHMS

Considering Theorem 4.3 and the results from Section 3, three questions come to mind.

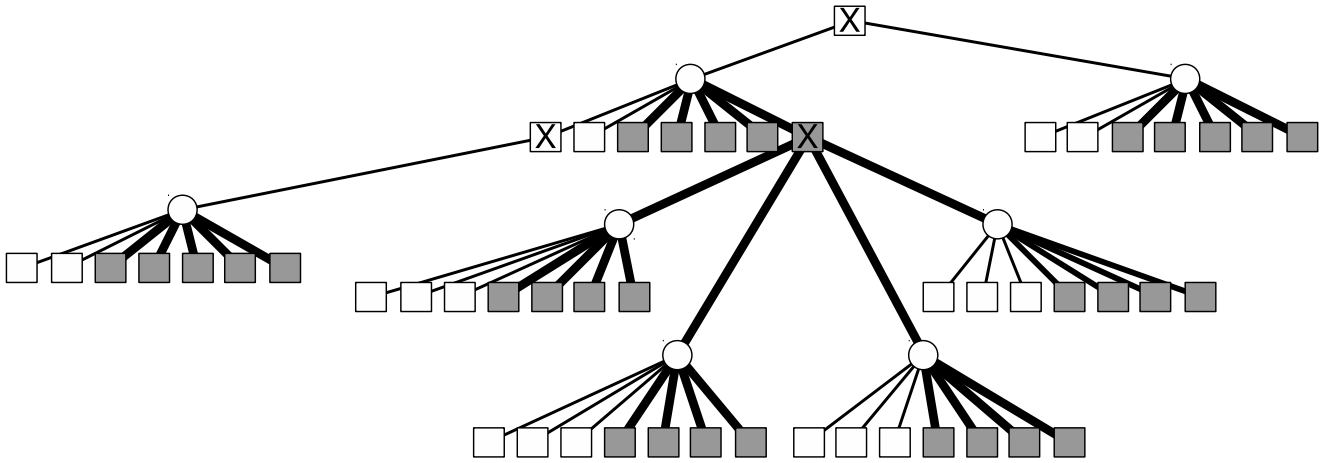

Figure 1: Part of a $((2,5),(3,5),D)$-tree rooted at the topmost bin. Bins are squares and balls are circles; neighborhoods of all balls and the bins marked by an "X" are depicted completely, the remainder of the tree is left out. Thin edges and white bins were added to the structure in the first round, thick edges and grey bins in the second. Up to distance $2D$ from the root, the pattern repeats itself, i.e., the $(2D-d)$-neighborhoods of all balls up to depth d appear identical.

- Does the lower bound still hold if random choices may be *asymmetric*, i.e., non-uniform choice distributions are possible?
- What happens if the bound of $\mathcal{O}(n)$ on the total number of messages is relaxed?
- Can a constant-time solution be devised if we stick to $\mathcal{O}(n)$ total messages but do not impose any upper bound on the number of messages for an individual node?

The third question is answered quickly by the following routine:

1. With probability, say, $1/\sqrt{n}$, a ball contacts $\sqrt{n}$ bins.
2. These balls perform a leader election on the resulting graph (using random identifiers).
3. The leader contacts all bins and coordinates a perfect distribution of the balls.

However, this algorithm introduces a central coordination instance. If this was a feasible solution, there would be no need for a parallel balls-into-bins algorithm in the first place.

Thus, the other two questions are more intriguing. To give an answer to the first one, we need to specify precisely what dropping the assumption of symmetry means.

Problem 5.1 (Asymmetric Balls-into-Bins). *An instance of Problem 3.1 is an* asymmetric parallel balls-into-bins problem*, if balls identify bins by globally unique addresses* $1,\dots,n$*. We call an algorithm solving this problem* asymmetric*.*

"Asymmetric" here means that biased random choices are permitted. This is impossible for symmetric or acquaintance algorithms, where the uniformly random port numberings even out any non-uniformity in the probability distribution of contacted port numbers.

In this extended abstract, we confine ourselves to presenting a simple algorithm demonstrating the basic idea of our solution. Given $l \in \mathcal{O}(\log n)$ that is a factor of n, $\mathcal{A}_1(l)$ is defined as follows.

1. Each ball contacts one bin chosen uniformly at random from the set $\{il \mid i \in \{1,\dots,n/l\}\}$.
2. Bin il, $i \in \{1,\dots,n/l\}$, assigns up to $3l$ balls to bins $il,\dots,(i+1)l-1$, such that each bin gets at most three balls.
3. The remaining balls (and the bins) proceed as if executing the symmetric Algorithm $\mathcal{A}_b^2$, however, with k initialized to $k(1) := 2^{\alpha l}$ for an appropriately chosen constant $\alpha > 0$.

Essentially, we create buckets of non-constant size l in order to ensure that the load of these buckets is slightly better balanced than it would be the case for individual bins. This enables the algorithm to place more than a constant fraction of the balls immediately.

However, this algorithm is somewhat unsatisfactory, since a subset of the bins has to deal with an expected communication load of $l + \mathcal{O}(1) \in \omega(1)$. In the accompanying technical report, we propose the more intricate Algorithm $\mathcal{A}(l)$ without this shortcoming.

Theorem 5.2. *Algorithm $\mathcal{A}(l)$ solves Problem 5.1 with a maximum bin load of three. It terminates after* $\log^* n - \log^* l + \mathcal{O}(1)$ *rounds w.h.p. Both balls and bins send and receive a constant number of messages in expectation. Balls send and receive at most $\mathcal{O}(\log n)$ messages w.h.p., bins $\mathcal{O}(\log n/\log\log n)$. The total number of messages is $\mathcal{O}(n)$ w.h.p.*

5.1 Symmetric Constant-Time Solution Using $\omega(n)$ Messages

A similar approach is feasible for symmetric algorithms if we permit $\omega(n)$ messages in total. Basically, Algorithm $\mathcal{A}(l)$ relies on asymmetry to achieve better coordination among the participants of the system. Instead, we may settle for better organizing a constant fraction of the bins; in turn, balls will need to send $\omega(1)$ messages to find such a bin with probability $1 - o(1)$.

COROLLARY 5.3. *For $l \in \mathcal{O}(\log n)$, an Algorithm $\mathcal{A}_c(l)$ exists that sends $\mathcal{O}(ln)$ messages w.h.p. and solves Problem 4.1 with a maximum bin load of $\mathcal{O}(1)$ within $\log^* n - \log^* l + \mathcal{O}(1)$ rounds w.h.p. Balls send and receive $\mathcal{O}(l)$ messages in expectation and $\mathcal{O}(\log n)$ messages w.h.p.*

6. CONCLUSIONS

We presented asymptotically optimal randomized load balancing algorithms for distributed systems. Our results demonstrate that adaptivity yields substantial improvements on previous parallel balls-into-bins algorithms. Given that in a totally anonymous setting it is possible to achieve a bin load of two within $\log^* n + \mathcal{O}(1)$ rounds, we hope that the proposed techniques may serve to improve future load balancing primitives for decentralized systems.

To show optimality of our approach, we provided a lower bound showing that for symmetric balls-into-bins algorithms that are asymptotically optimal with regard to communication complexity and maximum bin load, the proposed algorithm is $(1 + o(1))$-optimal with respect to time complexity. To this end, we devised a proof technique that we consider to be of interest in its own right. To the best of our knowledge, we demonstrated for the first time the existence of an overhead in either time or communication complexity in an essentially fully connected system that does not arise from asynchronicity, faults, or limited availability of randomization.

Acknowledgements

We would like to thank Thomas Locher and Reto Spöhel.

7. REFERENCES

[1] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel Randomized Load Balancing. In *Proc. 27th Symposium on Theory of Computing (STOC)*, pages 238–247, 1995.

[2] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load Balancing in the L_p Norm. In *Proc. 36th Symposium on Foundations of Computer Science (FOCS)*, pages 383–391, 1995.

[3] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced Allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.

[4] H. Bast and T. Hagerup. Fast and Reliable Parallel Hashing. In *Proc. 3rd Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 50–61, 1991.

[5] P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. W. Goldberg, Z. Hu, and R. Martin. Distributed Selfish Load Balancing. *SIAM Journal on Computing*, 37(4):1163–1181, 2007.

[6] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin. On Weighted Balls-into-Bins Games. *Theoretical Computer Science*, 409(3):511–520, 2008.

[7] P. Berenbrink, F. Meyer auf der Heide, and K. Schröder. Allocating Weighted Jobs in Parallel. In *Proc. 9th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 302–310, 1997.

[8] A. Czumaj and V. Stemann. Randomized Allocation Processes. *Random Structures and Algorithms*, 18(4):297–331, 2001.

[9] A. Czygrinow, M. Hańćkowiak, and W. Wawrzyniak. Fast Distributed Approximations in Planar Graphs. In *Proc. 22nd Symposium on Distributed Computing (DISC)*, pages 78–92, 2008.

[10] D. Dubhashi and D. Ranjan. Balls and Bins: A Study in Negative Dependence. *Random Structures and Algorithms*, 13:99–124, 1996.

[11] G. Even and M. Medina. Revisiting Randomized Parallel Load Balancing Algorithms. In *Proc. 16th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 209–221, 2009.

[12] G. Even and M. Medina. Parallel Randomized Load Balancing: A Lower Bound for a More General Model. In *Proc. 36th Conference on Theory and Practice of Computer Science (SOFSEM)*, pages 358–369, 2010.

[13] F. Fich and E. Ruppert. Hundreds of Impossibility Results for Distributed Computing. *Distributed Computing*, 16(2–3):121–163, 2003.

[14] J. Gil, F. M. auf der Heide, and A. Wigderson. The Tree Model for Hashing: Lower and Upper Bounds. *SIAM Journal on Computing*, 25(5):936–955, 1996.

[15] G. H. Gonnet. Expected Length of the Longest Probe Sequence in Hash Code Searching. *Journal of the ACM*, 28(2):289–304, 1981.

[16] T. Hagerup. The Log-Star Revolution. In *Proc. 9th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 259–278, 1992.

[17] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM Simulation on a Distributed Memory Machine. *Algorithmica*, 16:517–542, 1996.

[18] K. Kenthapadi and R. Panigrahy. Balanced Allocation on Graphs. In *Proc. 7th Symposium on Discrete Algorithms (SODA)*, pages 434–443, 2006.

[19] R. Kleinberg, G. Piliouras, and E. Tardos. Load Balancing without Regret in the Bulletin Board Model. In *Proc. 28th Symposium on Principles of Distributed Computing (PODC)*, pages 56–62, 2009.

[20] E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. Approximate Equilibria and Ball Fusion. *Theory of Computing Systems*, 36(6):683–693, 2003.

[21] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! *Computing Research Repository*, abs/1011.5470, 2010.

[22] C. Lenzen and R. Wattenhofer. Leveraging Linial's Locality Limit. In *Proc. 22nd Symposium on Distributed Computing (DISC)*, pages 394–407, 2008.

[23] C. Lenzen and R. Wattenhofer. Tight Bounds for Parallel Randomized Load Balancing. *Computing Research Repository*, abs/1102.5425, 2011.

[24] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[25] Z. Lotker, B. Patt-Shamir, and D. Peleg. Distributed MST for Constant Diameter Graphs. *Distributed Computing*, 18(6), 2006.

[26] N. Lynch. A Hundred Impossibility Proofs for Distributed Computing. In *Proc. 8th Symposium on Principles of distributed computing (PODC)*, pages 1–28, 1989.

[27] Y. Matias and U. Vishkin. Converting High Probability into Nearly-Constant Time—with Applications to Parallel Hashing. In *Proc. 23rd Symposium on Theory of Computing (STOC)*, pages 307–316, 1991.

[28] F. Meyer auf der Heide, C. Scheideler, and V. Stemann. Exploiting Storage Redundancy to Speed up Randomized Shared Memory Simulations. *Theoretical Computer Science*, 162(2):245–281, 1996.

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

[30] M. Mitzenmacher. How Useful is Old Information? Technical report, Systems Research Center, Digital Equipment Corporation, 1998.

[31] M. Mitzenmacher. On the Analysis of Randomized Load Balancing Schemes. In *Proc. 10th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 292–301, 1998.

[32] M. Mitzenmacher, B. Prabhakar, and D. Shah. Load Balancing with Memory. In *Proc. 43th Symposium on Foundations of Computer Science (FOCS)*, pages 799–808, 2002.

[33] M. Mitzenmacher, A. Richa, and R. Sitaraman. *Handbook of Randomized Computing*, volume 1, chapter The Power of Two Random Choices: A Survey of the Techniques and Results, pages 255–312. Kluwer Academic Publishers, Dordrecht, 2001.

[34] M. Naor. A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.

[35] D. Peleg and V. Rubinovich. A Near-Tight Lower Bound on the Time Complexity of Distributed MST Construction. In *Proc. 40th Symposium on Foundations of Computer Science (FOCS)*, pages 253–261, 1999.

[36] Y. Peres, K. Talwar, and U. Wieder. The $(1+\beta)$-Choice Process and Weighted Balls-into-Bins. In *Proc. 21th Symposium on Discrete Algorithms (SODA)*, pages 1613–1619, 2010.

[37] R. Raman. The Power of Collision: Randomized Parallel Algorithms for Chaining and Integer Sorting. In *Proc. 10th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 161–175, 1990.

[38] V. Stemann. Parallel Balanced Allocations. In *Proc. 8th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 261–269, 1996.

[39] K. Talwar and U. Wieder. Balanced Allocations: The Weighted Case. In *Proc. 39th Symposium on Theory of Computing (STOC)*, pages 256–265, 2007.

[40] B. Vöcking. How Asymmetry Helps Load Balancing. *Journal of the ACM*, 50(4):568–589, 2003.

[41] U. Wieder. Balanced Allocations with Heterogenous Bins. In *Proc. 19th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 188–193, 2007.