Distrib. Comput. (2016) 29:127–142
DOI 10.1007/s00446-014-0225-4

Tight bounds for parallel randomized load balancing

Christoph Lenzen · Roger Wattenhofer

Received: 6 June 2013 / Accepted: 27 June 2014 / Published online: 11 July 2014

Abstract Given a distributed system of n balls and n bins, how evenly can we distribute the balls to the bins, minimizing communication? The fastest *non-adaptive* and *symmetric* algorithm achieving a constant maximum bin load requires $\Theta(\log \log n)$ rounds, and any such algorithm running for $r \in \mathcal{O}(1)$ rounds incurs a bin load of $\Omega((\log n / \log \log n)^{1/r})$. In this work, we explore the fundamental limits of the general problem. We present a simple adaptive symmetric algorithm that achieves a bin load of 2 in $\log^* n + \mathcal{O}(1)$ communication rounds using $\mathcal{O}(n)$ messages in total. Our main result, however, is a matching lower bound of $(1 - o(1)) \log^* n$ on the time complexity of symmetric algorithms that guarantee small bin loads. 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 need not be 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.

An extended abstract of preliminary work appeared at STOC 2011 [25] and the corresponding article has been published on arxiv [24].

C. Lenzen (✉)
Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology,
Cambridge, MA, USA
e-mail: clenzen@cs.huji.ac.il; clenzen@csail.mit.edu

R. Wattenhofer
Computer Engineering and Networks Laboratory,
ETH Zurich, Zurich, Switzerland
e-mail: wattenhofer@tik.ee.ethz.ch

1 Introduction

Consider a synchronous distributed system of n identical balls and n bins. In each round of computation,

(i) each ball may send messages to $\mathcal{O}(\log n)$ bins,
(ii) bins may respond to the balls that contacted them, and
(iii) each ball may commit to a bin, notify it, and terminate.

The goal is that each ball commits to a bin, minimizing the maximal number of balls in a bin, the number of rounds, and the number of messages. There are no restrictions on local computations or the amount of randomness that is employed, so the main obstacle is the lack of global coordination: fast algorithms run in $\mathcal{O}(\log \log n)$ rounds, implying that balls and bins can exchange information with only a small part of the system, even if multi-hop communication is used.

Clearly, this is a very generic load balancing task, with many applications: canonical examples are job assignment tasks such as sharing work load among multiple processors, servers, or storage locations, but balls-into-bins games also play a vital role in e.g. low-congestion circuit routing, channel bandwidth assignment, or hashing, cf. [34].

Adler et al. [1] devised algorithms for the above problem whose running times and maximum bin loads are essentially doubly-logarithmic. They also provide a lower bound essentially showing that this running time is necessary for small bin loads.[1] 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. We believe that there are many systems that do not impose

[1] Strictly speaking, the bound is shown for constant round numbers only, but a generalization seems feasible.

these restrictions, especially (ii), motivating us to study more general classes of algorithms. In this work, we provide a complete characterization of the time/load-tradeoffs that can be achieved by *adaptive* algorithms.

1.1 Detailed contributions

Our main result, and the technically most challenging one, is a lower bound of $(1 - o(1)) \log^* n$ on the running time of *symmetric* algorithms that achieve small (near-constant) bin loads with $\mathcal{O}(n)$ messages (Sect. 4); symmetric algorithms are characterized by balls choosing the bins to contact uniformly. 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.

By providing the following complementing adaptive algorithms, we show that this bound is essentially tight in each assumption.[2]

- A simple symmetric algorithm achieving a maximum bin load of 2 within $\log^* n + \mathcal{O}(1)$ rounds of communication using $\mathcal{O}(n)$ messages. As a plus, the algorithm also works if communication is asynchronous and is highly resilient to faults. (Sect. 3)
- An asymmetric algorithm achieving load 3 in $\mathcal{O}(1)$ rounds, also using $\mathcal{O}(n)$ messages. (Sect. 5.1)
- A symmetric algorithm achieving load $\mathcal{O}(1)$ in $\mathcal{O}(\log^* n - \log^* l)$ rounds, using $\mathcal{O}(nl)$ messages. (Sect. 5.2)

All our results hold with high probability (w.h.p.), that is, with probability at least $1 - 1/n^c$ for an arbitrarily selected constant $c > 0$.

2 Related work

2.1 Upper bounds

Probably one of the earliest applications of randomized load balancing has been hashing. In this context, Gonnet [16] 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 $\Omega(\log n / \log \log n)$ w.h.p. [11].

With growing interest in parallel computing, since the beginning of the nineties the topic received increasingly more attention. Karp et al. [18] 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. [4] generalized their result by showing that if the balls choose sequentially from $d \geq 2$ u.i.r. bins greedily, i.e., the currently least loaded one, the maximum load is $\log \log n / \log d + \mathcal{O}(1)$ w.h.p.[3] Given that contacted bins are chosen u.i.r., 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 [9]. This is achieved by placing each ball immediately if the load of an inspected bin is not too large, rather than 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 much better with probability larger 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 these bounds to a larger spectrum of algorithms by removing some artificial assumptions [13]. A matching algorithm was proposed by Stemann [40], 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 $r \in \Theta(\log \log n)$ this implies a constantly bounded bin load. Even and Medina also

[2] We refer to [24] for a number of additional upper bounds; for the sake of a streamlined presentation, we focus on the main techniques and results here.

[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 balls-into-bins algorithms

Algorithm	Sym.	Adt.	Choices	Rounds	Maximum bin load	Messages
Naive [16]	Yes	No	1	1 (only commit)	$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$	n
Greedy [1]	Yes	No	2	2	$\mathcal{O}\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$\mathcal{O}(n)$
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 [40]	Yes	No	2	r	$\mathcal{O}\left(\left(\frac{\log n}{\log\log n}\right)^{1/r}\right)$	$\mathcal{O}(n)$
Theorem 3.2	Yes	Yes	$\mathcal{O}(1)$ exp.	$\log^* n + \mathcal{O}(1)$	2	$\mathcal{O}(n)$
Theorem 5.2	No	Yes	$\mathcal{O}(1)$ exp.	$\mathcal{O}(1)$	3	$\mathcal{O}(n)$
Theorem 5.4	Yes	Yes	$\mathcal{O}(l)$ exp.	$\log^* n - \log^* l + \mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(ln)$
[24]	Yes	No	d	$r + \frac{\log\log n}{\log d} + \mathcal{O}(1)$	$\frac{\log^{(r)} n}{\log^{(r+1)} n} + r + \frac{\log\log n}{\log d} + \mathcal{O}(1)$	$\mathcal{O}(n)$
[24]	Yes	Yes	$\mathcal{O}(1)$ exp., d	$r + \frac{\log\log n}{\log d} + \mathcal{O}(1)$	$\frac{\log^{(r)} n}{\log^{(r+1)} n} + r + \frac{\log\log n}{\log d} + \mathcal{O}(1)$	$\mathcal{O}(n)$

Varying the parameters d, l, or r yields different trade-offs

proposed a 2-round "adaptive" algorithm [12].[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 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. (see [24]). Table 1 shows 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, the only possibility for further improvement has been to search for non-adaptive or asymmetric algorithms. Vöcking [42] 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 [19].

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 in a constant number of rounds. 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. [37] 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.

[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.

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. [4,9,30–32,40,42]). In [33] 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. [3] 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. [7,8,21,37,41]) in order to model tasks of varying resource consumption. The case of heterogeneous bins was examined as well [43]. In recent years, balls-into-bins has also been considered from a game theoretic point of view [6,20].

Many algorithms for hashing problems bear similarity to our symmetric algorithm with running time $\log^* n + \mathcal{O}(1)$. In particular, a number of publications present algorithms with running times of $\mathcal{O}(\log^* n)$ (or very close) in PRAM models [2,5,15,29]. Furthermore, recent results on distributed coloring [39] permit to derive a symmetric balls-into-bins algorithm running in $\mathcal{O}(\log^* n)$ time, however, using significantly larger messages. While at first glance these routines operate in models that considerably differ from ours, at their heart lies the same idea we employ in the balls-into-bins setting: In each iteration, an exponentially growing share of the available resources is dedicated to dealing with the remaining keys, bins, or nodes, respectively. Implicitly, this approach already occurred in previous work by Raman [38]. For a more detailed review of results on hashing we refer the interested reader to [17].

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 constant-time constant-load asymmetric algorithm presented in Sect. 5.1 gives rise to a load-balancing primitive (see [25], Section 3.4), which has been applied to sort n^2 keys in $\mathcal{O}(1)$ rounds in fully connected networks [35]. It was recently shown how to achieve this result deterministically using an unrelated approach [23], hence we refrain from presenting this primitive in this article.

2.2 Lower bounds

The lower bound by Adler et al. [1] (and the generalization by Even and Medina [13]) on the round complexity of symmetric algorithms is larger than ours, 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 is useful in various settings, for instance if the initial communication overhead is large, or for hashing strategies where the bins are locations that need to be re-accessed later on. However, it seems natural to assume that, by default, the option to use a non-constant number of communication rounds goes hand in hand with the ability of balls to contact different bins in different rounds. Our lower bound also applies in this setting, i.e., for adaptive algorithms. It arises from the assumption that bins are anonymous, which fits a wide range of systems.

From a technical perspective, at first glance our lower bound appears similar to that from [1]: both bounds argue about tree structures in the graph on the balls and bins whose edges indicate a communication relation. This connection is however superficial. In the adaptive case, this graph is not a simple random graph, as the information nodes gain from earlier communication feeds back to its evolution over time, i.e., the communication in round r may depend on the local topology in round $r-1$. This property, i.e., that the communication graph evolves during the course of the algorithm, also distinguishes our result from other distributed lower bounds, cf. [14,22,26,28], in particular an $\Omega(\log^* n)$ lower bound on hashing (in a certain model) by Gil et al. [15]. Finally, we note that the information theoretic approach underlying the bounds in, e.g., [10,27,36] cannot be applied to our setting, since the graph describing which edges could *potentially* be used to transmit a message is complete bipartite. Thus, even with restricted message size, there is no strong bound on the amount of information that can be exchanged between nodes that holds a priori, rendering the technique ineffective. Hence, our lower bound is the first to prove the existence of a coordination bottleneck in a system without a physical bottleneck.

3 A simple symmetric algorithm

In this section, we present the symmetric algorithm $\mathcal{A}_{\text{sym}}$ achieving a bin load of 2 in $\log^* n + \mathcal{O}(1)$ rounds.

Algorithm $\mathcal{A}_{\text{sym}}$**:** Symmetric Algorithm sending $\mathcal{O}(n)$ messages w.h.p.

```
1 Set $k(1) := 1$
2 for $i = 1, \ldots$ until all balls have terminated do
3     Each non-terminated ball requests from $k(i)$ u.i.r. bins
      permission to be placed into them.
4     Each bin responds by admitting permission to (up to) 2
      requesting balls, minus the number of balls that already
      committed to it. These choices are arbitrary.
5     Any ball receiving at least one permission chooses an
      arbitrary of the respective bins to commit to, informs it, and
      terminates.
6     If $i = 1$, set $k(2) := 4$. Otherwise, set
      $k(i+1) := \min\{k(i)2^{k(i)/4}, \lceil \log n \rceil\}$.
7 end
```

The intuition behind this approach is that each message sent has, independently of others, a constant probability to be sent to a bin that is willing to accept the sending ball: there must always be at least $n/2$ bins with current load smaller than 2. Thus, the number of non-terminated balls decrases by a factor that is exponentially small in $k(i)$. In turn, it is safe to increase $k(i)$ exponentially for the next round without causing too many messages to be sent (if $\omega(n)$ messages are sent, the probability that the receiving bin accepts becomes $o(1)$). Once $k(i)$ becomes $\lceil \log n \rceil$, each remaining ball terminates within $\mathcal{O}(1)$ rounds w.h.p.

Lemma 3.1 *The following invariants hold w.h.p. in rounds $i \in \{2, \ldots, \log^* n + \mathcal{O}(1)\}$ of the above algorithm (if n is sufficiently large).*

- *The number of non-terminated balls n_i at the beginning of the round is bounded by $n/(5 \cdot 2^{i-3} k(i))$.*
- *The total number of messages sent in the round is at most $n/(5 \cdot 2^{i-3})$.*
- *Pick any message sent by a ball in round i. After fixing all random decisions up to and including this round except the destination of this message, it has a probability of at least 3/10 to receive a response.*
- $n_{i+1} \in \min\{2^{-k(i)/2} n_i, \mathcal{O}(\log n)\}$.

Proof For any constant $l \in \mathbb{N}_0$, the expected number of bins receiving exactly l messages in the first round is

$$\binom{n}{l}\left(\frac{1}{n}\right)^l \left(1-\frac{1}{n}\right)^{n-l} \in (1 \pm o(1))\frac{n}{l!e},$$

where $0! = 1$. It is known that these bounds also hold w.h.p. [11] (see [24] for more details). Summing up a constant number of terms, we can conclude that for sufficiently large n, it holds that $n_2 \leq n/10$ w.h.p. This shows the first statement for round 2.

Now consider round $i > 1$ and assume that the first statement is satisfied in round i. The second statement follows immediately. Also, the first statement implies that when fixing all but one message sent in this round, there must be at least

$$\frac{2n - (n - n_i) - n_i k(i)}{2} > \frac{n - 2n/5}{2} = \frac{3n}{10}$$

bins that will still accept a message. This shows the third statement for round i. The fourth statement for round i follows from the third by observing that each node sends $k(i)$ messages, thus the probability that none of them is sent to an accepting bin is bounded by $(7/10)^{k(i)} = (49/100)^{k(i)/2}$, and applying Chernoff's bound to see that $n_{i+1} \in \min\{2^{-k(i)/2} n_i, \mathcal{O}(\log n)\}$ w.h.p. Finally, the first statement for round $i+1$ follows (i) because $2^{-k(i)/2} n_i \leq 2^{-k(i)/4-1} n_i$ as $k(i)$ is an integer multiple of 4 for all $i \geq 2$ and (ii) clearly $n/(5 \cdot 2^{i-3} k(i)) \geq n/(5 \cdot 2^{i-3} \lceil \log n \rceil) \in \omega(\log n)$ for all $i \in \log^* n + \mathcal{O}(1)$.

Thus, we can use induction over i to prove all claims; since each individual statement follows w.h.p., applying the union bound over all $\mathcal{O}(\log^* n)$ statements shows that they hold concurrently w.h.p. □

From these invariants it is straightforward to derive the following theorem.

Theorem 3.2 *Algorithm $\mathcal{A}_{\text{sym}}$ satisfies the following:*

- *it terminates in $\log^* n + \mathcal{O}(1)$ rounds w.h.p.*
- *the maximum bin load is 2*
- *the total number of messages is $\mathcal{O}(n)$ w.h.p.*
- *each ball and bin sends and receives $\mathcal{O}(1)$ messages in expectation and $\mathcal{O}(\log n)$ messages w.h.p.*

Proof Basic calculations show that given the growth of $k(i)$, there is a round $i_0 \in \log^* n + \mathcal{O}(1)$ such that $k(i) = \lceil \log n \rceil$ for all $i \geq i_0$ (cf. Lemma 4.14). By the third statement of Lemma 3.1, we conclude that in each such round, each remaining ball terminates with probability at least $1 - n^{-\Omega(1)}$. By independence of the random choices in each round and the union bound, thus all balls terminate within $\log^* n + \mathcal{O}(1)$ rounds w.h.p. The fact that the maximum bin load is 2 is a direct consequence of bins accepting at most 2 balls.

The bound on the total number of messages immediately follows from the second invariant shown in Lemma 3.1. By symmetry, balls (bins) send and receive the same expected number of messages, showing that the expected values are all constant. As $\sum_{i=1}^{\log^* n + \mathcal{O}(1)} k(i) \in \mathcal{O}(\log n)$, each ball sends $\mathcal{O}(\log n)$ messages w.h.p. Trivially, no ball can receive more than $\lceil \log n \rceil$ messages, as it terminates in the round in which it receives its first message. Since all messages have u.i.r. destinations, the bound on the total number of messages and Chernoff's bound show that, w.h.p., bins receive (and thus also send) at most $\mathcal{O}(\log n)$ messages. □

We remark that it is simple to adapt algorithm $\mathcal{A}_{\text{sym}}$ to asynchrony. If bins also send messages if they refuse a ball, balls can make sure not to send too many messages by waiting for all $k(i)$ responses of "round i" (according to their local view) before sending the next batch of $k(i+1)$ messages; because the total number of messages remains linear w.h.p. provided that each message has a constant probability of being "successful", and each message has a constant probability of success as long as not more than $\mathcal{O}(n)$ messages are sent, the approach works out similar to the synchronous case.

Another interesting point is that $\mathcal{A}_{\text{sym}}$ is extremely simple and robust: each message merely needs to convey the information that it has been sent (in a given phase of a given round), local computations are trivial, and losing a constant (but roughly uniformly chosen) fraction of all messages will not break the algorithm (if $k(i)$ is increased more conservatively). For further details and variants of the algorithm, we refer to [24].

Finally, one can compare the results of the technique to those of non-adaptive algorithms when fixing a budget of d u.i.r. bins a ball may contact throughout the course of the algorithm. Table 1 lists the respective bounds from [24]. Essentially, one cuts off the growth of $k(i)$ at d, after which $\log \log n / \log d$ additional rounds suffice to place the remaining balls. For instance, a time complexity of $\mathcal{O}(\log^* n)$ can be achieved for $d \in \log^{1/\mathcal{O}(\log^* n)} n$. The advantage over a non-adaptive approach here lies in the fact that a total message complexity of $\mathcal{O}(n)$ is maintained despite $d \in \omega(1)$. Note, however, that for small values of d Stemann's collision algorithm (which uses $d = 2$) achieves a better trade-off between loads and running time using $\mathcal{O}(n)$ messages. It is an open question whether adaptiveness allows for a faster variant of the protocol.

4 Lower bound

In this section, we show that $\mathcal{A}_{\text{sym}}$ is near-optimal, in a very strict sense. After presenting some initial definitions and outlining the approach, we proceed to proving the following result.

Theorem 4.1 *For each $L \in \mathbb{N}$, there exists $t \in (1 - o(1)) \log^* n - \log^* L$ with the following property. If any symmetric Algorithm $\mathcal{A}$ that sends in total $\mathcal{O}(n)$ messages in* *expectation terminates within t rounds, then w.h.p. $n^{1-o(1)}$ bins have load larger than L.*

We will briefly present some generalizations of this result in Sect. 4.3.

4.1 Preliminaries and outline

The requirement of "symmetry", i.e., of unbiased random choices, so far has been formulated as property of the algorithm. However, it can also be interpreted as a property of the system. Instead of the bins having unique identifiers, balls identify them by so-called *port numberings*.

Definition 4.2 (Port numberings) A *port numbering* is a permutation of $\{1, \dots, n\}$. If ball b addresses bins by its port numbering p_b, this means that it sends each message to some port $i \in \{1, \dots, n\}$, which then is received by bin $p_b(i)$. In turn, b receives any message bin $p_b(i)$ sends at port i.

In other words, port numberings enable balls to distinguish bins (which clearly is necessary to e.g. re-contact a bin and commit to it), but do not guarantee that there is any correlation between the addresses different balls use to refer to a given bin.

Problem 4.3 (Symmetric balls-into-bins) An instance of the balls-into-bins problem is *symmetric*, if balls address bins by u.i.r. port numberings (i.e., each p_b is drawn u.i.r. from the symmetric group of permutations of $\{1, \dots, n\}$). We call an algorithm that can be implemented under this assumption *symmetric*.

This is a property that may naturally arise from the system, as opposed to the notion of symmetry introduced in [1], which imposes a restriction on algorithms' behavior.

Observation 4.4 *If at any point of the execution of a symmetric algorithm a ball contacts a bin it has not contacted yet, the contacted bin is drawn uniformly at random from all bins it has not contacted yet. This also holds when conditioning on arbitrary other events, provided that these do not constrain the port numbers of bins the ball has not contacted so far.*

In other words, u.i.r. port numberings "mask" any asymmetry the algorithm may seek to introduce based on the information the respective ball has gathered so far.

In order to show Theorem 4.1, we need to bound the fraction of the global state balls can access during the course of an algorithm running for t rounds. As a ball may systematically contact each bin it contacted before, this information is a subset of the information available in its $(2t)$-neighborhood in the graph where an edge between a ball and a bin is added in round i if the ball contacts the bin in round i for the first time.

Definition 4.5 (Balls-into-bins graph) The (bipartite and simple) *balls-into-bins graph* $G_{\mathcal{A}}(t)$ associated with an execution of the symmetric algorithm $\mathcal{A}$ that has run for $t \in \mathbb{N}_0$ rounds is constructed as follows. The node set $V := B \dot\cup U$ consists of $|B| = |U| = n$ balls and bins. In each round $i \in \{1, \dots, t\}$, each ball $b \in B$ adds an edge connecting itself to bin $u \in U$ if b contacts u for the first time 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. Note that $G_{\mathcal{A}}(0) = (V, \emptyset)$.

In the remainder of the section, we will consider such graphs only.

We will not examine $G_{\mathcal{A}}(t)$ for an arbitrary symmetric algorithm, since it does not sufficiently expose the symmetry between different executions of $\mathcal{A}$ (which are functions of the random inputs and port numberings). Instead, we will modify the communication pattern of $\mathcal{A}$, without affecting its output distribution.

Definition 4.6 (Simulation) Algorithm $\mathcal{A}'$ *simulates* Algorithm $\mathcal{A}$, if the distributions of bin loads and balls' termination times are identical for $\mathcal{A}'$ and $\mathcal{A}$.

At the heart of the proof of Theorem 4.1 lies an induction executed in Lemma 4.15. The claim of the induction is that for each $i \in \{1, \dots, t\}$, there is an algorithm $\mathcal{A}_i$ simulating $\mathcal{A}$ for which $G_{\mathcal{A}_i}(i)$ contains many disjoint copies of a certain highly symmetric subgraph; the induction is anchored by setting $\mathcal{A}_0 := \mathcal{A}$. For the induction step,

1. we argue that, w.h.p., for a constant fraction of these copies, the constituent balls send few messages in round i (Lemma 4.10),
2. there is an algorithm $\mathcal{A}_{i+1}$ that simulates $\mathcal{A}$ and in which the aforementioned balls all send the same number of messages (Lemma 4.11), and
3. this entails that many disjoint copies of a symmetric subgraph of similar structure will be present in $G_{\mathcal{A}_{i+1}}(i+1)$ w.h.p. (Lemma 4.13).

The induction will halt after $t \in (1 - o(1)) \log^* n - \log^* L$ steps (Lemma 4.14). The theorem then follows because (i) if all balls in a copy of the subgraph for round t commit to a bin, this incurs a large expected bin load (Lemma 4.16) and (ii) the induction proved that there are many such copies.

The critical subgraphs have a recursive structure: In each step of the induction, we piece together copies of the new subgraphs out of the copies of the preceding one and bins that have not been contacted by any balls yet.

Definition 4.7 ((Δ^U, Δ^B, D)-Trees) Given $\Delta^U = (\Delta_1^U, \dots, \Delta_i^U)$, $\Delta^B = (\Delta_1^B, \dots, \Delta_i^B)$, and an Algorithm $\mathcal{A}$, the subgraph of $G_{\mathcal{A}}(i)$ induced by the set of nodes in distance

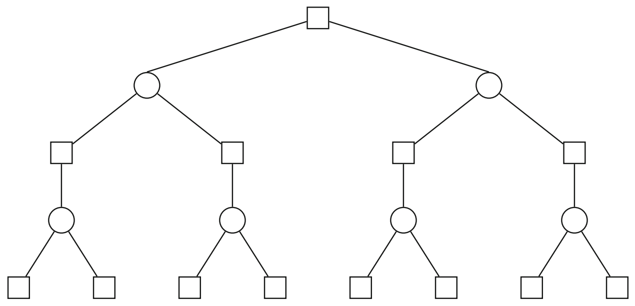

Fig. 1 A ((2), (3), 2)-tree rooted at the bin on *top*. Note that the 2-hop neighborhoods of the balls adjacent to the root are isomorphic. The leaf bins may have edges to balls outside the tree, which are not depicted

at most $2D$ from bin R is a (Δ^U, Δ^B, D)-*tree rooted at bin* R, if the following conditions are met.

– It is a tree.
– Each ball has degree $\sum_{j=1}^{i} \Delta_j^B$.
– R has degree Δ_i^U, with all incident edges in $E_{\mathcal{A}}(i)$.
– For each inner non-root bin, there is some round $j \in \{1, \dots, i\}$ such that its degree is Δ_j^U and all its incident edges are from $E_{\mathcal{A}}(j)$.

To simplify the notation, we may simply refer to i-trees without specifying Δ^U, Δ^B, or D explicitly when these parameters are clear from the context.

To get an idea of the structure and how it arises, observe that a $((2,3),(3,4),D)$-tree (cf. Fig. 2) rooted at R is constructed if the following conditions are met.

1. Each ball in distance $d \leq 2D$ from R in $G_{\mathcal{A}}(2)$ is in distance at most d from the root of a $((2),(3),D)$-tree (cf. Fig. 1) in $G_{\mathcal{A}}(1)$.
2. Each such ball contacts exactly 4 random bins in round 2 of $\mathcal{A}$.
3. All corresponding messages are received by previously isolated bins.
4. Each such isolated bin is contacted by exactly 3 such balls (1 for leaves), without creating any cycles.
5. No inner bin in the tree is contacted in round 2 by a random choice of a ball outside the tree.

Note that we examine the $(2D)$-neighborhood of R here, since D communication rounds in the balls-into-bins graph enable to relay information over at most $2D$ hops. We remark that we can choose the parameters Δ^U and D fairly freely, whereas Δ^B is under the control of the algorithm: Δ_i^B is the number of new bins contacted by a ball in an $(i-1)$-tree in round i; we introduce the simulating Algorithm $\mathcal{A}_i$ to ensure uniformity of this choice.

Azuma's inequality. The main body of the proof will be concerned with establishing that many t-trees will be constructed. Our inductive approach rests on the hypothesis that, in each step, many suitable "building blocks" are available with a large probability. To show that this is the case, we will leverage the following standard tail bound.

Theorem 4.8 (Azuma's inequality) *Let X be a random variable that is a function of independent random variables $X_1, \dots, X_N$. Assume that changing the value of a single X_i for some $i \in \{1, \dots, N\}$ changes the outcome of X by at most $\delta_i \in \mathbb{R}^+$. Then, for any $\tau \in \mathbb{R}_0^+$, we have*

$$P\big[|X - E[X]| > \tau\big] \leq 2e^{-\tau^2/\left(2\sum_{i=1}^{N}\delta_i^2\right)}.$$

In all applications of the theorem, we will have that $E[X] \in n^{1-o(1)}$ and choose $\tau = E[X]/2$. Picking the underlying random variables $X_1, \dots, X_N$ as the $\Theta(n)$ random bit strings and port numberings of the balls and bins, all we need is that changing random bits or port numbering of a single node will affect X by at most $n^{o(1)}$. To ensure this, we assume w.l.o.g. that bins respond to at most $\mathcal{O}(t \log n)$ balls in each round.[6] Since balls contact at most $\mathcal{O}(\log n)$ bins in each round, degrees in $G_{\mathcal{A}}(i)$ are thus uniformly bounded by $\Delta \in \mathcal{O}(t \log n)$. As we are interested in $t \in \mathcal{O}(\log^* n)$ rounds only, this yields that changing the behavior of a single node influences at most $(2\Delta)^{2t} \in n^{o(1)}$ nodes throughout the course of the algorithm.

Observation 4.9 *If we change the random bits and port numbering of a single node, this affects at most $n^{o(1)}$ nodes throughout the execution of any algorithm.*

Hence, Theorem 4.8 guarantees that $|X - E[X]| \leq E[X]/2$ w.h.p. whenever we can prove that (i) $E[X] \in n^{1-o(1)}$ and (ii) changing the behavior of $n^{o(1)}$ nodes affects X by $n^{o(1)}$.

4.2 Proof

Throughout this subsection, fix a symmetric algorithm $\mathcal{A}$ satisfying that all balls terminate at the latest in round t and the algorithm sends in total $\mathcal{O}(n)$ messages in expectation. To anchor the induction mentioned earlier, observe that trivially every bin is a 0-tree and set $\mathcal{A}_0 := \mathcal{A}$. For brevity, we fix some threshold $T_{\min} \in n/\log^{o(1/\log^* n)} n$ in the following; we will halt the induction at the maximal index $t < \log^* n$ for which the lower bound T_t on the number of disjoint i-trees in $G_{\mathcal{A}_t}(t)$ (that holds w.h.p.) is still larger than $T_{\min}$.

We now perform the first part of the induction step, showing that, for many trees present in round i, the constituent balls send few messages.

[6] Since each ball sends $\mathcal{O}(t \log n)$ messages and the algorithm sends $\mathcal{O}(n)$ messages in expectation, induction shows that w.h.p. no bin is contacted by more than $\mathcal{O}(t \log n)$ balls.

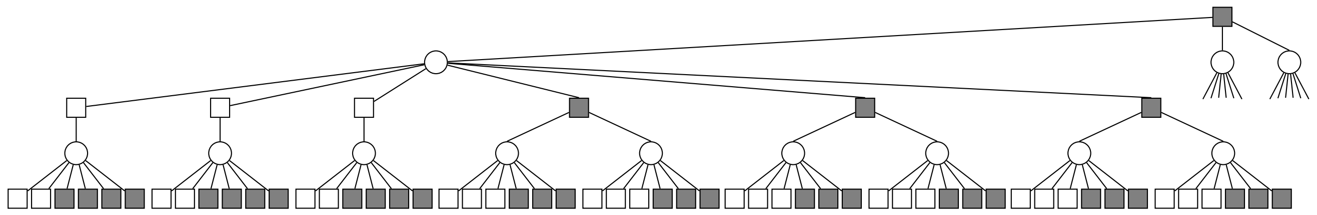

Fig. 2 A subtree of a ((2, 3), (3, 4), 2)-tree rooted at the bin on *top*. *White bins* have been contacted in the first round only, *grey bins* in the second. Note that (i) the 2-hop neighborhoods of the balls adjacent to the root are isomorphic, (ii) the neighborhoods of *grey non-leaf bins* are isomorphic, and (iii) the neighborhoods of *white non-leaf bins* are isomorphic. The further a node from the closest leaf, the larger the hop-distance for which the respective neighborhoods look identical. Again, leaf nodes may have been contacted by balls outside the tree, which is not depicted

Lemma 4.10 *Assume that $\mathcal{G}_{\mathcal{A}_i}(i)$ contains at least $T_i \geq T_{\min}$ disjoint $((\Delta_1^U, \dots, \Delta_i^U), (\Delta_1^B, \dots, \Delta_i^B), t+2)$-trees w.h.p., for some $0 \leq i < t$. Then, w.h.p., for each of $\Omega(T_i)$ such trees it holds that its constituent balls contact in total $\mathcal{O}(n/T_i)$ bins in round $i+1$ of $\mathcal{A}$.*

Proof We condition on the event $\mathcal{E}$ that indeed T_i disjoint trees are present in $\mathcal{G}_{\mathcal{A}_i}(i)$. Denote by M_j, $j \in \{1, \dots, T_i\}$ the random variable counting the number of messages sent in round $i+1$ by balls in the jth (disjoint) i-tree. Since $\mathcal{A}$ (and thus also $\mathcal{A}_i$ in round $i+1$) sends in total at most $\mathcal{O}(n)$ messages in expectation and $\mathcal{E}$ occurs w.h.p., the same holds true when conditioning on $\mathcal{E}$. Thus, we have that $\sum_{j=1}^{T_i} E[M_j] \in \mathcal{O}(n)$. Hence, at least half of the M_j satisfy that $E[M_j] \in \mathcal{O}(n/T_i)$.

Denote by $J \subseteq \{1, \dots, T_i\}$ the set of indices so that $E[M_j] \in \mathcal{O}(n/T_i)$ and by X_j, $j \in J$, the indicator variable being 1 if $M_j \leq 2E[M_j]$. By Markov's inequality, $P[X_j = 0] = P[M_j > 2E[M_j]] \leq 1/2$. Therefore, $X := \sum_{j \in J} X_j$ satisfies that $E[X] \geq |J|/2 \geq T_i/4$. By Observation 4.9 and Theorem 4.8 (where the underlying random variables are the random bit strings and port numberings of the nodes), it follows that

$$P\left[X \leq \frac{T_i}{8}\right] \leq P\left[|X - E[X]| > \frac{E[X]}{2}\right]$$
$$\in e^{-\Omega(T_i^2/(n \cdot n^{o(1)}))}$$
$$\subset n^{-\omega(1)}.$$

In other words, w.h.p. it holds that for $\Omega(T_i)$ trees we have that $M_j \in \mathcal{O}(n/T_i)$, as claimed. □

Next, in order to assemble $(i+1)$-trees, we need that the balls in the i-trees we use as building blocks will each contact *exactly* Δ_{i+1}^B new bins in rounds $i+1$. To this end, we "grant" the algorithm additional bins it may contact. The modified algorithm simulates $\mathcal{A}$, i.e., produces an identical load distribution (see Definition 4.6).

Lemma 4.11 *Assume that, for some $0 \leq i < t$, the following conditions are met.*

- *$\mathcal{A}_i$ simulates $\mathcal{A}$.*
- *$\mathcal{A}_i$ sends the same messages as $\mathcal{A}$ in rounds $i+1, \dots, t$.*
- *If $i > 0$, $\mathcal{A}_i$ sends in total $\mathcal{O}(in^2/T_{i-1})$ messages in expectation in rounds $1, \dots, i$, where $n \geq T_{i-1} \geq T_i$.*
- *W.h.p., $\mathcal{G}_{\mathcal{A}_i}(i)$ contains at least $T_i \geq T_{\min}$ disjoint $((\Delta_1^U, \dots, \Delta_i^U), (\Delta_1^B, \dots, \Delta_i^B), t+2)$-trees.*

Then an Algorithm $\mathcal{A}_{i+1}$ with the following properties exists.

(i) $\mathcal{A}_{i+1}$ simulates $\mathcal{A}$.
(ii) $\mathcal{A}_{i+1}$ sends the same messages as $\mathcal{A}$ in rounds $i+2, \dots, t$.
(iii) $\mathcal{A}_{i+1}$ sends in total $\mathcal{O}((i+1)n^2/T_i)$ messages in expectation in rounds $1, \dots, i+1$, and
(iv) W.h.p., for each of $\Omega(T_i)$ disjoint i-trees in $\mathcal{G}_{\mathcal{A}_i}(i) = \mathcal{G}_{\mathcal{A}_{i+1}}(i)$ it holds that each of its balls contacts exactly $\Delta_{i+1}^B \in \mathcal{O}(n/T_i)$ new bins in round $i+1$.

Proof Lemma 4.10 proves a statement very similar to (iv), except that the balls of an i-tree *in sum* send $\Delta_{i+1}^B \in \mathcal{O}(n/T_i)$ messages in round $i+1$. Algorithm $\mathcal{A}_{i+1}$ now lets *each* such ball contact *exactly* Δ_{i+1}^B new bins.[7] The algorithm stores what these additional edges are. Hence, in subsequent rounds, it can ignore them and perform the same steps as $\mathcal{A}_i$ and therefore $\mathcal{A}$. (If in $\mathcal{A}$ a ball later contacts a bin to which an "extra" edge has been formed, $\mathcal{A}_{i+1}$ simply marks it as known to $\mathcal{A}$ from that round on).

These considerations show that $\mathcal{A}_{i+1}$ satisfies all statements except possibly Statement (iii). Regarding this statement, recall that $\mathcal{A}$ sends $\mathcal{O}(n)$ messages in expectation. Hence, the same holds true for $\mathcal{A}_i$ in round $i+1$ when conditioning on the event that T_i disjoint i-trees are present in $G_{\mathcal{A}_i}(i)$. We added $\mathcal{O}(n/T_i)$ messages for each of the at most n/T_i balls in each of $\Theta(T_i)$ trees, resulting in $\mathcal{O}(n^2/T_i + in^2/T_{i-1}) \subseteq \mathcal{O}((i+1)n^2/T_i)$ (if $i = 0$ the second term is not present) expected messages in total. □

[7] Note that there is no need to execute $\mathcal{A}_{i+1}$ in a distributed fashion, as we only require it to prove statements about $\mathcal{A}$. Hence, $\mathcal{A}_{i+1}$ can gather all information to make the appropriate decision; for the purpose of the proof, we simply ignore the related communication.

We will use Lemma 4.11 to construct $\mathcal{A}_{i+1}$ out of $\mathcal{A}_i$. For the time being, assume that the requirements of the lemma are met and we can construct $\mathcal{A}_{i+1}$ with the stated properties, so that we can continue with the inductive step. Hence, we have a large number of i-trees we can use as building blocks to "piece together" $(i+1)$-trees in round $i+1$. However, we will also need an ample supply of bins that are isolated in $G_{\mathcal{A}_{i+1}}(i) = G_{\mathcal{A}_i}(i)$.

Lemma 4.12 *Assume that for $0 \le i < t$, $\mathcal{A}_{i+1}$ is an algorithm constructed using Lemma 4.11. Then $G_{\mathcal{A}_{i+1}}(i)$ contains $2^{-\mathcal{O}(tn/T_i)}n$ isolated bins w.h.p.*

Proof Observe that, since at most $\mathcal{O}(\log n + n/T_{\min}) = \mathcal{O}(\log n)$ messages are sent per round and node, degrees in $G_{\mathcal{A}_{i+1}}(i)$ are $\mathcal{O}(i \log n) \subset o(n)$. Hence, whenever a ball contacts a new bin by a random choice, the probability to contact a specific bin is at most $1/(n - o(n))$. By Statement (iii) of Lemma 4.11, $\mathcal{A}_{i+1}$ sends in expectation $\mathcal{O}(tn^2/T_i)$ messages in the first $i+1$ rounds. By Observation 4.9 and Theorem 4.8, this bound also applies w.h.p. Conditioning on the event that this bound is satisfied, the probability that a specific bin remains isolated until the end of round i is lower bounded by

$$\left(1 - \frac{2}{n}\right)^{\mathcal{O}(tn^2/T_i)} = e^{-\mathcal{O}(tn/T_i)}.$$

By linearity of expectation, another application of Theorem 4.8, and the union bound, the statement of the lemma follows. □

Having the building blocks in place, we need to show that indeed a lot of $(i+1)$-trees are assembled in round $i+1$.

Lemma 4.13 *Suppose that for some $0 \le i < t$, the preconditions of Lemma 4.11 are met, let $\mathcal{A}_{i+1}$ be the algorithm the lemma shows to exist, and let $\Delta^U_{i+1} := L\Delta^B_{i+1}$ for some natural $L \le n/T_i$. Then, w.h.p., $G_{\mathcal{A}_{i+1}}(i+1)$ contains at least $T_{i+1} \in 2^{-(n/T_i)^{\mathcal{O}(t)}}n$ disjoint $((\Delta^U_1, \ldots, \Delta^U_{i+1}), (\Delta^B_1, \ldots, \Delta^B_{i+1}), t+2)$-trees.*

Proof outline

1. Bound the size of $(i+1)$-trees by $(n/T_i)^{\mathcal{O}(t)}$.
2. Show that the probability for such a tree to occur is no smaller than $2^{-(n/T_i)^{\mathcal{O}(t)}}$. To see this, we add the tree edges from $E_{\mathcal{A}_{i+1}}(i+1)$ one by one. Lemmas 4.12 and 4.13 show that there are sufficiently many i-trees and isolated bins that each edge connects to a suitable node with probability at least $2^{-(n/T_i)^{\mathcal{O}(t)}}$. Similarly, the probability that no edges from balls outside the tree connect to tree bins is $2^{-(n/T_i)^{\mathcal{O}(t)}}$.
3. Since by Lemma 4.12 we have $2^{-(n/T_i)^{\mathcal{O}(t)}}n$ potential roots for $(i+1)$-trees at the beginning of round $i+1$, this implies that the expected number of $(i+1)$-trees is $2^{-(n/T_i)^{\mathcal{O}(t)}}n$. To obtain a lower bound on the expected number of *disjoint* trees, we randomly orient the edge set and observe that all edges of a tree point to the root with probability at least $2^{-(n/T_i)^{\mathcal{O}(t)}}$ (due to the bound on its size). Counting the roots of correctly oriented trees, any two roots are either in disjoint trees or one is part of the other's tree (but not vice versa). Hence, dividing the expected number of oriented trees by the tree size, we get a lower bound of $T_{i+1} \in 2^{-(n/T_i)^{\mathcal{O}(t)}}n$ on the expected number of disjoint trees.
4. Since $T_{i+1} \ge T_{\min}$, this bound is at least $n^{1-o(1)}$. Hence the claim follows from Observation 4.9 and Theorem 4.8. □

Proof Denote by $n_i \le n/T_i$ the number of nodes in an i-tree and by n_{i+1} the number of nodes in an $(i+1)$-tree. Clearly, the number of children of each bin in an $(i+1)$-tree is at most factor $\Delta^U_{i+1} + 1$ larger than in an i-tree. Analogously, ball degrees are at most a factor $\Delta^B_{i+1} + 1$ larger than in an i-tree. Therefore,

$$\begin{aligned}
n_{i+1} &\le (\Delta^U_{i+1} + 1)^{t+2}(\Delta^B_{i+1} + 1)^{t+2} n_i \\
&< \left(L(\Delta^B_{i+1} + 1)^2\right)^{t+2} n_i \\
&\in \left(\frac{n}{T_i}\right)^{\mathcal{O}(t)} &(4.1)\\
&\subseteq \left(\frac{n}{T_{\min}}\right)^{\mathcal{O}(\log^* n)} \\
&\subseteq o(T_i), &(4.2)
\end{aligned}$$

where in the third inequality we exploit that $L \le n/T_i$ by assumption.

Consider the following procedure, constructing an $(i+1)$-tree.

– Starting at the root (which can be any bin isolated at the beginning of round $i+1$), we iterate through the desired tree topology in a breadth-first-search fashion. In each step, we determine the actual ball or bin taking the respective place in the tree. When following edges from rounds $1, \ldots, i$, the nodes are predetermined by the topology of $G_{\mathcal{A}_{i+1}}(i)$. Otherwise, we choose as follows:
  - In case the node is a ball, we choose it to be adjacent to the root of a not yet involved i-tree.
  - In case of a bin, we choose a bin that is isolated in $G_{\mathcal{A}_{i+1}}(i)$.
– For each edge from $E_{\mathcal{A}}(i+1)$, we require that the respective ball indeed randomly contacted the respective bin in round $i+1$.

– No nodes from outside the tree contact a bin inside the tree by a random choice in round $i+1$.

Note that because we always pick balls adjacent to roots of i-trees when the topology of $G_{\mathcal{A}_{i+1}}(i)$ does not determine the choice, this procedure guarantees that the desired topology is constructed, i.e., any leaf bin of a j-tree for $1 \le j \le i$ is in distance at least $2(t+2)$ from the root.

We would like to lower bound the probability p that a bin that is isolated in $G_{\mathcal{A}_{i+1}}(i)$ becomes the root of a $(i+1)$-tree. To this end, we condition on the following event $\mathcal{E}$.

– $G_{\mathcal{A}_{i+1}}(i+1)$ contains at least T_i disjoint i-trees. This holds w.h.p. by assumption.
– $G_{\mathcal{A}_{i+1}}(i) = G_{\mathcal{A}_i}(i)$ contains $ne^{-\mathcal{O}(tn/T_i)}$ isolated bins. This holds w.h.p. by Lemma 4.12.
– In round $i+1$ of $\mathcal{A}_{i+1}$, in total $\mathcal{O}(n^2/T_i)$ messages are sent. Since $\mathcal{A}_{i+1}$ satisfies this in expectation by Lemma 4.11 and $\mathcal{O}(\log n + n/T_{\min}) = \mathcal{O}(\log n)$ messages are sent per ball, Observation 4.9 and Theorem 4.8 show that this holds w.h.p. as well.

As p is lower bounded by $P[\mathcal{E}]$ times the probability that R becomes the root of an $(i+1)$-tree conditional to $\mathcal{E}$, conditioning on $\mathcal{E}$ has negligible effect on the computed expectation.

Now we are ready to review the construction process for the $(i+1)$-tree laid out above. We add the edges of the tree that are not already determined by $G_{\mathcal{A}_{i+1}}(i)$ one by one, in each step multiplying with a lower bound on the probability to form a connection to a suitable node. Finally, we multiply with the probability that no ball from outside the tree contacts a bin in the tree. Leveraging the computed upper bound on n_{i+1} and conditioning on $\mathcal{E}$, this is straightforward.

– If for an edge the parent is a bin, we can choose from at least

$$T_i - n_{i+1} \overset{(4.2)}{\in} \Omega(T_i)$$

root bins of i-trees. Hence the probability of success is $\Omega(T_i/n) \subset e^{-\mathcal{O}(tn/T_i)}$ in each such step.
– If for an edge the parent is a ball we can choose from

$$ne^{-\mathcal{O}(tn/T_i)} - n_{i+1} \overset{(4.1)}{\subseteq} ne^{-\mathcal{O}(tn/T_i)}$$

bins, where we use that $(n/T_i)^{\mathcal{O}(t)} \subset e^{\mathcal{O}(tn/T_{\min})} \subseteq n^{o(1)}$. Hence, we have a probability of at least $e^{-\mathcal{O}(tn/T_i)}$ to succeed in each such step.
– The probability that an individual random contact of a ball outside the tree does *not* connect to a bin in the tree is lower bounded by $1-n_{i+1}/(n-o(n))$. The probability that none of the w.h.p. at most $\mathcal{O}(tn^2/T_i)$ such messages is received by a bin in the tree is thus at least

$$\left(1-\frac{2n_{i+1}}{n}\right)^{tn^2/T_i} \in 2^{-\mathcal{O}(tn_{i+1}n/T_i)} \overset{(4.1)}{\subseteq} 2^{-(n/T_i)^{\mathcal{O}(t)}}.$$

– Since there are $n_{i+1}-1$ edges in the tree, we can thus lower bound

$$p \in 2^{-(n/T_i)^{\mathcal{O}(t)}}\left(2^{-\mathcal{O}(tn/T_i)}\right)^{n_{i+1}} \overset{(4.1)}{\subseteq} 2^{-(n/T_i)^{\mathcal{O}(t)}}.$$

We conclude that the expected number of $(i+1)$-trees in $G_{\mathcal{A}_{i+1}}(i+1)$ is lower bounded by $pT_i \in 2^{-(n/T_i)^{\mathcal{O}(t)}}n$.

To lower bound the expected number of *disjoint* trees, choose a random orientation of the edge set of $G_{\mathcal{A}_{i+1}}(i+1)$ and count the expected number of trees for which all edges point to the root. The probability that this happens for a given tree is $2^{-n_{i+1}+1}$. Note that the nodes of a correctly oriented tree $\mathcal{T}$ cannot participate in any correctly oriented tree whose root lies outside the node set of $\mathcal{T}$. Thus, the expected number of disjoint $(i+1)$-trees is lower bounded by

$$\begin{aligned}\frac{2^{-n_{i+1}+1}}{n_{i+1}} \cdot 2^{-(n/T_i)^{\mathcal{O}(t)}}n &\overset{(4.1)}{=} 2^{-(n/T_i)^{\mathcal{O}(t)}}n\\ &\subseteq 2^{-(n/T_{\min})^{\mathcal{O}(\log^* n)}}n\\ &\subseteq n^{1-o(1)}.\end{aligned}$$

(The threshold $T_{\min} \in n/\log^{o(1/\log^* n)} n$ was chosen to guarantee the last inequality.) By Observation 4.9 and Theorem 4.8, we conclude that the probability that indeed $2^{-(n/T_i)^{\mathcal{O}(t)}}n$ disjoint $(i+1)$-trees are present in $G_{\mathcal{A}_{i+1}}(i+1)$ is at least $1-2e^{-n^{2-o(1)}/(n^{1+o(1)})} \subset 1-1/n^{\omega(1)}$. □

This lemma is the last piece we need for the induction step. It remains to determine when the induction halts, i.e., the maximal value of t such that $T_t \ge T_{\min}$.

Lemma 4.14 *Given $L, t \in \mathbb{N}$, consider any sequence $(T_i)_{i\in\mathbb{N}_0}$ satisfying that*

$$T_0 = \left\lfloor \frac{n}{L} \right\rfloor \quad \text{and} \quad \forall i \in \mathbb{N}_0 : \; T_{i+1} \in 2^{-(n/T_i)^{\mathcal{O}(t)}}n.$$

Then, there exist $T_{\min} \in n/\log^{o(1/\log^* n)}$ and

$$t \in (1-o(1))\log^* n - \log^* L$$

such that $T_t \ge T_{\min}$.

Proof Denote by

$${}^k a := \left.a^{a^{\cdot^{\cdot^{a}}}}\right\} k \in \mathbb{N} \text{ times}$$

the tetration (tower) with basis $2 \leq a \in 2^{\mathcal{O}(t)}$ (where the constants in the $\mathcal{O}$-term are the same as in the bound on T_{i+1} above), and by $\log_a^* x$ the smallest integer such that ${}^{(\log_a^* x)}a \geq x$. We bound

$$\begin{aligned}\frac{n}{T_t} &\leq 2^{(n/T_{t-1})^{\log(a)}}\\ &\leq 2^{\left(2^{(n/T_{t-2})^{\log a}}\right)^{\log a}}\\ &= 2^{2^{\log a (n/T_{t-2})^{\log a}}}\\ &= 2^{a^{(n/T_{t-2})^{\log a}}}.\end{aligned}$$

Repeating inductively yields

$$\begin{aligned}\log\left(\frac{n}{T_t}\right) &\leq \left({}^{t-1}a\right)^{(n/T_0)^{\log a}}\\ &\leq \left({}^{t-1}a\right)^{(n/T_0)^{a}}\\ &\leq {}^{t+\log_a^*(n/T_0)}a.\end{aligned}$$

Applying $\log^*$, we get the sufficient condition

$$\log^*\left({}^{t+\log_a^*(n/T_0)}a\right) \leq \log^* n - 4, \tag{4.3}$$

since then

$$\begin{aligned}\frac{n}{T_t} &\leq {}^{\log^*(n/T_t)}2\\ &= {}^{\log^*(\log(n/T_t))+1}2\\ &\overset{(4.3)}{\leq} {}^{\log^* n-3}2\\ &\leq \log\log n\\ &= (\log n)^{\log\log\log n/\log\log n}\\ &\in \log^{o(1/\log^* n)} n.\end{aligned}$$

Hence, there is a choice of $T_{\min} \in n/\log^{o(1/\log^* n)}$ so that $T_t \geq T_{\min}$.

It remains to show that (4.3) is satisfied for sufficiently large values of t. To this end, first recall that $a \geq 2$ and consider some integer $k \geq 2$. We estimate

$$\begin{aligned}&\log^*\left({}^{k}a(1+\log a)\right)\\ &= 1+\log^*\left(\log\left({}^{k}a(1+\log a)\right)\right)\\ &= 1+\log^*\left({}^{k-1}a\log a+\log(1+\log a)\right)\\ &\leq 1+\log^*\left({}^{k-1}a(1+\log a)\right).\end{aligned}$$

By induction on k, it follows that

$$\log^*\left({}^{k}a\right) \leq k-1+\log^*(a(1+\log a)) \leq k+\log^* a.$$

This implies

$$\begin{aligned}\log^*\left({}^{(t+\log_a^*(n/T_0))}a\right) &\leq t+\log_a^*(n/T_0)+\log^* a\\ &\leq t+\log^*(n/T_0)+\log^* a\\ &\in t+\log^* L+\log^*(\mathcal{O}(t))+2\\ &\subset (1+o(1))t+\log^* L+\mathcal{O}(1).\end{aligned}$$

Hence, the sufficient condition given in Inequality (4.3) can be satisfied for

$$t \in (1-o(1))\log^* n - \log^* L.$$

□

We remark that this interplay between L and t is by no means arbitrary. If for any $r \in o(\log^* n)$ one accepts a maximum bin load of $\log^{(r)} n/\log^{(r+1)} n + r + \mathcal{O}(1) \subset o(\log^{(r)} n)$, where $\log^{(r)} n$ denotes the r times iterated logarithm, Problem 4.3 can be solved in $r + \mathcal{O}(1)$ rounds by a variant of $\mathcal{A}_{\text{sym}}$ [24].

With the above lemmas in place, we can now stitch together the induction showing that, for $t \in (1-o(1))\log^* n - \log^* L$, there is an Algorithm $\mathcal{A}_t$ that simulates $\mathcal{A}$ and will produce many $(\Delta^U, \Delta^B, t+2)$-trees.

Lemma 4.15 *There exists $t \in (1-o(1))\log^* n - \log^* L$ such that for any symmetric algorithm $\mathcal{A}$ that sends $\mathcal{O}(n)$ messages in expectation, we can construct an Algorithm $\mathcal{A}_t$ with the following properties.*

1. *$\mathcal{A}_t$ simulates $\mathcal{A}$.*
2. *$G_{\mathcal{A}_t}(t)$ contains $n^{1-o(1)}$ disjoint $(\Delta^U, \Delta^B, t+2)$-trees w.h.p., for some vectors $\Delta^U, \Delta^B \in \mathbb{N}^t$ satisfying that $\Delta^U = L\Delta^B$.*

Proof Without loss of generality, suppose that $t > 0$; in particular $\lfloor n/L \rfloor \geq T_{\min}$. We prove the statement by induction on rounds $i \in \{0, \ldots, t\}$. The induction hypothesis is that there is an Algorithm $\mathcal{A}_i$ with the following properties.

(i) $\mathcal{A}_i$ simulates $\mathcal{A}$.
(ii) $\mathcal{A}_i$ sends the same messages as $\mathcal{A}$ in rounds $i+1, \ldots, t$.
(iii) If $i > 0$, $\mathcal{A}_i$ sends in total $\mathcal{O}(in^2/T_{i-1})$ messages in expectation in rounds $1, \ldots, i$.
(iv) For $\Delta^U, \Delta^B \in \mathbb{N}^i$ satisfying that $\Delta_j^U = L\Delta_j^B$ for all $j \in \{1, \ldots, i\}$, $G_{\mathcal{A}_i}(i)$ contains at least T_i disjoint $((\Delta_1^U, \ldots, \Delta_i^U), (\Delta_1^B, \ldots, \Delta_i^B), t+2)$-trees w.h.p.
(v) $T_i \geq T_{\min}$, $n/T_i \geq L$, and for $i \neq 0$ also $T_i \leq T_{i-1}$.

To anchor the induction at $i = 0$, set $T_0 := \lfloor n/L \rfloor$ and $\mathcal{A}_0 := \mathcal{A}$. Clearly, this choice satisfies Statements (i), (ii), (iii), and (v) for index $i = 0$. Since every bin is a $((), (), t+2)$-tree in $G_{\mathcal{A}}(0)$, statement (iv) holds as well.

To perform the induction step from $0 \le i < t$ to $i+1$, we apply Lemma 4.11 to construct $\mathcal{A}_{i+1}$, which is feasible by the induction hypothesis. According to the lemma, Statements (i), (ii), and (iii) are satisfied. By Lemma 4.13 (which also can be applied by the hypothesis), with $\Delta_{i+1}^B \in \mathcal{O}(n/T_i)$ given by Lemma 4.11 and $\Delta_{i+1}^U = L\Delta_{i+1}^B$, we have that $G_{\mathcal{A}_{i+1}}(i+1)$ contains $2^{-(n/T_i)^{\mathcal{O}(t)}}$ disjoint $(i+1)$-trees w.h.p., i.e., Statement (iv) is true. Since clearly $T_{i+1} \le T_i$ and therefore also $n/T_{i+1} \ge L$, the induction step succeeds provided that $T_{i+1} \ge T_{\min}$. According to Lemma 4.14, this is correct for all $i < t \in (1-o(1))\log^* n - \log^* L$.

Evaluating the hypothesis for index t, we obtain Algorithm $\mathcal{A}_t$ simulating $\mathcal{A}$. By statements (iv) and (v), $G_{\mathcal{A}_t}(t)$ contains $T_t \ge T_{\min} \in n^{1-o(1)}$ disjoint $(\Delta^U, \Delta^B, t+2)$-trees w.h.p., where $\Delta^U = L\Delta^B \in \mathbb{N}^t$, as claimed. □

In other words, there is an algorithm producing the same output distribution as $\mathcal{A}$ that guarantees that many highly symmetric structures are present in $G_{\mathcal{A}_t}(t)$. To bring the main result home, it thus remains to show that balls presented with the symmetric neighborhoods of $(\Delta^U, \Delta^B, t+2)$-trees risk to overload a bin if they all commit.

Lemma 4.16 *For an Algorithm $\mathcal{A}_t$ that terminates within t rounds, denote by $E(i)$ the expected load of a bin that is in distance 2 of the root of a $(\Delta^U, \Delta^B, t+2)$-tree in $G_{\mathcal{A}_t}(t)$ and has been added to the tree in round $i \in \{1,\dots,t\}$. If $\Delta^U = L\Delta^B$, then $\sum_{i=1}^t E(i) = L$.*

Proof Assume that ball b is within three hops of the root of a $(\Delta^U, \Delta^B, t+2)$-tree, but without fixing which of its 3-hop neighbors is the root bin. Label its $(2t)$-neighborhood by the random strings and port numberings nodes are given initially. Clearly, this information determines to which neighbor b commits, as $\mathcal{A}_t$ (up to the end of round t) can be interpreted as deterministic algorithm on the labeled graph $G_{\mathcal{A}_t}(t)$.

Now condition on the event $\mathcal{E}$ that this (fixed) neighborhood partakes in a $(\Delta^U, \Delta^B, t+2)$-tree with root within 3 hops of b and that b commits by the end of round t. Note that since the depth of the $(\Delta^U, \Delta^B, t+2)$-tree is $2(t+2)$ and b is within 3 hops of the root, the $(2t+1)$-neighborhood of b obeys the repetitive pattern of a t-tree. Now consider $i \in \{1,\dots,t\}$. For symmetry reasons, it is equally likely for each bin u that has been added to the tree in round t and is within three hops from b to become the root of a t-tree. Since this holds for *any* possible $(2t)$-neighborhood of b (conditioned on $\mathcal{E}$), the previous statement implies the following: If u is a bin in distance exactly 2 from the root of a t-tree that has been added in round i, the probability that an adjacent ball b commits to it equals $p(i)/\Delta_i^B$, where $p(i)$ is the probability that a ball b within 3 hops of the root commits to one of its neighboring bins added to the tree in round i.

We have that $\sum_{i=1}^t p(i) = 1$, as by assumption the algorithm terminates within t rounds. By linearity of expectation,

$$\sum_{i=1}^t E(i) = \sum_{i=1}^t \Delta_i^U \frac{p(i)}{\Delta_i^B} = \sum_{i=1}^t Lp(i) = L.$$

□

We are now in the position to prove our lower bound on the trade-off between maximum bin load and running time of symmetric algorithms.

Theorem 4.1 *For each $L \in \mathbb{N}$, there exists $t \in (1-o(1))\log^* n - \log^* L$ with the following property. If any symmetric Algorithm $\mathcal{A}$ that sends in total $\mathcal{O}(n)$ messages in expectation terminates within t rounds, then w.h.p. $n^{1-o(1)}$ bins have load larger than L.*

Proof Set $L' := 2tL$. By Lemma 4.15, there is an Algorithm $\mathcal{A}_t$ simulating $\mathcal{A}$ which satisfies that $G_{\mathcal{A}_t}(t)$ contains $n^{1-o(1)}$ disjoint $(\Delta^U, \Delta^B, t+2)$-trees w.h.p., for some $\Delta^U = L'\Delta^B \in \mathbb{N}^t$. Since $\mathcal{A}$ terminates by the end of round t, so does $\mathcal{A}_t$.

Consider a $(\Delta^U, \Delta^B, t+2)$-tree in $G_{\mathcal{A}_t}(t)$. According to Lemma 4.16, $\sum_{i=1}^t E(i) = L'$, where $E(i)$ is the expected load of a bin in distance 2 from the root that has been added to the tree in round i. Therefore, there is an $i_0 \in \{1,\dots,t\}$ such that $E(i_0) \ge L'/t = 2L$. On the other hand, the maximum possible load of such a bin is $\Delta_{i_0}^U \in n^{o(1)}$. Denoting by X the random variable counting the number of balls committing to such a bin, it must hold that

$$P[X > L] = P[X > E(i_0)/2] \in 1/n^{o(1)},$$

since any faster asymptotic decrease of this probability would lead to the contradiction

$$\begin{aligned} 2L &\le E(X) \\ &\le (1 - P[X > L])L + P[X > L]\Delta_{i_0}^U \\ &\in L + o(1). \end{aligned}$$

As the number of disjoint $(\Delta^U, \Delta^B, t+2)$-trees is $n^{1-o(1)}$ w.h.p., the expected number of bins with load larger than L is therefore $n^{1-o(1)}$. By Observation 4.9 and Theorem 4.8, we conclude that the number of bins with load larger than L is also $n^{1-o(1)}$ w.h.p. □

4.3 Generalizations

There is a number of ways in which Theorem 4.1 can be strengthened.

Probabilistic termination guarantee: The symmetric algorithm in question may terminate within t rounds with some probability $p < 1$. The same proof essentially works for such algorithms as well, where the statement of Lemma 4.16

is weakened to guaranteeing $\sum_{i=1}^{L} E(i) \geq pL$. Accordingly, one chooses $L' = 2tL/p$ in the proof of Theorem 4.1, implying that the theorem still applies provided that $p \in 1/^{o(\log^* n)}2$. In particular, any symmetric algorithm runs for more than $(1-o(1))\log^* n - \log^* L$ rounds with probability $1-o(1)$ or with high probability suffers a bin load larger than L whenever it terminates faster.

Larger degrees: We assumed that balls never contact more than $\mathcal{O}(\log n)$ bins. This can be relaxed to $n^{o(1/\log^* n)}$ contacted bins per round, since Observation 4.9 still applies for $t \in \mathcal{O}(\log^* n)$ rounds. This can be generalized to a degree bound of λn for any constant $\lambda < 1$ for algorithms that send $\mathcal{O}(n)$ messages in total w.h.p. [24]. However, this requires a more careful information theoretic argument, exploiting that balls sufficiently far from leafs in disjoint trees do not communicate, and *conditional to this* their randomness is independent also in later rounds; this permits to use Chernoff's or Azuma's bound on the random variables describing the random bits of the nodes conditional to participating in i-trees in round $i \in \{1,\ldots,t-1\}$.

Direct communication between bins: One can generalize the tree structures to account for bins contacting other bins in the same way as balls (i.e., according to u.i.r. port numberings). Analogous reasoning shows that the asymptotic bound $T_i \in 2^{-(n/T_{i-1})^{\mathcal{O}(t)}} T_{i-1}$ holds in this case as well.

Address forwarding: Consider the following additional ability of the algorithm/system. Each bins has a unique identifier initially only known to itself, and any node learning this identifier can contact the bin directly in future rounds (irrespective of the port numbering). This means that nodes now can obtain information from up to distance 2^{2t} in $G_{\mathcal{A}}(t)$. Arguing about sufficiently deep t-trees, our reasoning still applies, and Lemma 4.14 is easily adapted by choosing $a \in 2^{2^{\mathcal{O}(t)}}$.

5 Constant-time solutions

Considering Theorems 3.2 and 4.1, two questions come to mind.

– 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?

In this section, we will discuss these issues.

5.1 An asymmetric algorithm

In order to answer the first question, we need to specify precisely what dropping the assumption of symmetry means.

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

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

We will now show that asymmetric algorithms can indeed obtain constant bin loads in constant time, at asymptotically optimal communication costs. Note that for asymmetric algorithms, we can w.l.o.g. assume that n is a multiple of some number $l \in o(n)$, since we may opt for ignoring negligible $n - l\lfloor n/l \rfloor$ bins. We will use this observation in the following. We start by presenting a simple algorithm demonstrating the basic idea of our solution. Fix some $l \in \mathcal{O}(\log n)$ that is a factor of n and consider Algorithm $\mathcal{A}_{\text{asym}}(l)$.

Algorithm $\mathcal{A}_{\text{asym}}(l)$: Simple solution to Problem 5.1.

```
1 Each ball contacts one bin chosen uniformly at random from the
  set $\{il \mid i \in \{1,\ldots,n/l\}\}$.
2 Bin $il$, $i \in \{1,\ldots,n/l\}$, assigns up to $3l$ balls to the bins
  $(i-1)l+1,\ldots,il$, such that each bin gets at most three balls.
  (That is, it informs the respective balls, which then commit to the
  respective bin.)
3 The remaining balls (and the bins) proceed as if executing the
  symmetric Algorithm $\mathcal{A}_{\text{sym}}$ starting in round 2, however, with
  $k(2)$ initialized to $4 \cdot 2^{\lfloor \alpha l \rfloor}$ 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. Small values of l suffice for this algorithm to terminate quickly.

Theorem 5.2 *Algorithm $\mathcal{A}_{\text{asym}}(l)$ solves Problem 5.1 with a maximum bin load of three. It terminates within $\log^* n - \log^* l + \mathcal{O}(1)$ rounds w.h.p.*

Proof For $i \in \mathbb{N}_0$, we denote by Y^i the random variable counting the number of bins receiving at least i messages in Step 1 of the algorithm. We can apply Chernoff's bound to these variables [11], showing that $|Y^i - E(Y^i)| \in \mathcal{O}\left(\log n + \sqrt{E(Y^i)\log n}\right)$ w.h.p. Consequently, we have that the number $Y^i - Y^{i+1}$ of bins receiving exactly i messages differs by at most $\mathcal{O}\Big(\log n + \sqrt{\max\{E(Y^i), E(Y^{i+1})\}\log n}\Big)$ from its expectation w.h.p. Moreover, Chernoff's bound states that these bins receive at most $l + \mathcal{O}(\sqrt{l \log n} + \log n) \subset \mathcal{O}(\log n)$ messages w.h.p., i.e., we need to consider only values of $i \in \mathcal{O}(\log n)$.

Thus, the number of balls that are not accepted in the first round is bounded by

$$\begin{aligned}
&\sum_{i=3l+1}^{n} (i-3l)\left(Y^i - Y^{i+1}\right)\\
&\in \sum_{i=3l+1}^{\mathcal{O}(\log n)} (i-3l)\, E\left(Y^i - Y^{i+1}\right) + \mathcal{O}\left(\sqrt{n \log n}\right)\\
&\subseteq \frac{n}{l} \sum_{i=3l+1}^{\mathcal{O}(\log n)} (i-3l) \binom{n}{i} \left(\frac{l}{n}\right)^i \left(1-\frac{l}{n}\right)^{n-i} + o(n)\\
&\subseteq \frac{n}{l} \sum_{i=3l+1}^{\mathcal{O}(\log n)} (i-3l) \left(\frac{el}{i}\right)^i + o(n)\\
&\subseteq \frac{n}{l} \sum_{j=1}^{\infty} jl \left(\frac{e}{3}\right)^{(j+2)l} + o(n)\\
&\subseteq \mathcal{O}\left(\left(\frac{e}{3}\right)^{2l} n\right) \subseteq \left(\frac{3}{e}\right)^{-(2-o(1))l} n
\end{aligned}$$

w.h.p., where in the third step we used the inequality $\binom{n}{i} \leq (en/i)^i$.

Thus, w.h.p. at most $2^{-\Omega(l)}n$ balls are not assigned in the first two steps. Hence, we can deal with the remaining balls within $\log^* n - \log^* l + \mathcal{O}(1)$ rounds by "kick-starting" $\mathcal{A}_{\text{sym}}$ with a larger initial value of $k = 4 \cdot 2^{\lfloor \alpha l \rfloor}$ for some $\alpha \in \Omega(1)$ (cf. [24]). We conclude that $\mathcal{A}_{\text{asym}}(l)$ will terminate after $\log^* n - \log^* l + \mathcal{O}(1)$ rounds w.h.p. as claimed. □

In particular, if we set $l := \log^{(r)} n$, for any $r \in \mathbb{N}$, the algorithm terminates within $r + \mathcal{O}(1)$ rounds w.h.p.

We remark that Algorithm $\mathcal{A}_{\text{asym}}(l)$ 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 [24], we provide a more involved algorithm for which this expectation is constant.

5.2 A symmetric algorithm using $\omega(n)$ messages

A constant round complexity can also be achieved by symmetric algorithms if we permit $\omega(n)$ messages in total. Also here the key idea is to organize bins into groups of size $\omega(1)$, in order to assign a non-constant fraction of the balls in the first round, and then deal with the remaining balls by the "kick-started" variant of Algorithm $\mathcal{A}_{\text{sym}}$.

Given an integer $l \leq n/\log n$, consider the simple Algorithm $\mathcal{A}_\omega(l)$, which guarantees that a constant fraction of the bins will be assigned to *coordinators* of $\Omega(l)$ bins.

Algorithm $\mathcal{A}_\omega(l)$**:** Helper routine for solving Problem 4.3 with a superlinear number of messages.

```
1 With probability $n/l$, a ball contacts a uniformly random subset
  of $l$ bins.
2 Each bin receiving at least one message responds to one of these
  messages, choosing arbitrarily. The respective ball is the
  coordinator of the bin.
```

Lemma 5.3 *When executing $\mathcal{A}_\omega(l)$, bins receive at most $\mathcal{O}(\log n)$ messages w.h.p. In total $\mathcal{O}(n)$ messages are sent w.h.p. W.h.p., a constant fraction of the bins is assigned to coordinators of $\Omega(l)$ bins.*

Proof Chernoff's bound shows that in Step 1 w.h.p. $\Theta(n/l)$ balls decide to contact bins, i.e., $\Theta(n)$ messages are sent, and a second application of the bound shows that bins receive $\mathcal{O}(\log n)$ messages w.h.p. A third application (utilizing that the indicator variables for bins being non-empty are negatively associated [11]) shows that w.h.p. a constant fraction of the bins receives at least one message. Thus, $\Theta(n/l)$ balls coordinate in total $\Theta(n)$ bins, implying that also $\Theta(n)$ bins must be coordinated by balls that are responsible for $\Omega(l)$ bins. □

Permitting communication exceeding n messages by more than a constant factor, all but a small fraction of the balls can find a bin coordinated by a ball responsible for $\Omega(l)$ bins. Most of these balls can be distributed to the bins such that the maximum load remains constantly bounded; subsequently, Algorithm $\mathcal{A}_{\text{sym}}$ finishes the job.

Algorithm $\mathcal{A}_{\text{sym}}(l)$**:** Symmetric algorithm using $\omega(n)$ messages; l_0 and C are suitable values, see Theorem 5.4.

```
1 Run Algorithm . Coordinators inform their bins about the number
  of bins they supervise.
2 Each ball contacts $l$ u.i.r. bins. The bins respond with the number
  of bins their coordinator supervises (0 if they have no
  coordinator).
3 If a ball receives a maximal value of at least $l_0 \in \Omega(l)$, it
  responds to a randomly chosen bin sending a value of at least $l_0$.
  This message contains a random string of $\mathcal{O}(\log n)$ bits to
  identify the ball.
4 The bins forward the received bit strings to their coordinators,
  who assign up to $C \in \mathcal{O}(1)$ of them to each of their supervised
  bins. This information is forwarded back to the respective balls,
  which then commit to the assigned bins.
5 The remaining balls (and the bins) proceed as if executing the
  symmetric Algorithm $\mathcal{A}_{\text{sym}}$ starting in round 2, however, with
  $k(2)$ initialized to $4 \cdot 2^{\lfloor \alpha l \rfloor}$ for an appropriately chosen constant
  $\alpha > 0$.
```

Theorem 5.4 *For $l \in \mathcal{O}(\log n)$ and suitable choices $l_0 \in \Theta(l)$, $C \in \Theta(1)$, $\mathcal{A}_{\text{sym}}(l)$ sends $\mathcal{O}(ln)$ messages w.h.p. and solves Problem 4.3 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.*

Proof W.h.p., Algorithm $\mathcal{A}_\omega(l)$ assigns coordinators to a constant fraction of the bins such that these coordinators control $l_0 \in \Omega(l)$ bins. Thus, the probability that a ball contacts only bins coordinated by balls supervising fewer than l_0 is smaller than $2^{-\Omega(l)}$; Chernoff's bound therefore states that w.h.p. $(1 - 2^{-\Omega(l)})n$ balls contact a bin b with $\ell(b) \geq l_0$.

Next, these balls contact a bin b from which they received a value $\ell(b) \geq l_0$, where they choose uniformly at random among all such bins. Note that (for sufficiently large constants) all random bit strings chosen by the balls are distinct w.h.p., so the coordinators can identify all balls contacting bins they supervise. The coordinators assign (at most) constantly many of the respective balls to each of their bins. Similarly to the proof of Theorem 5.2, we can see that (if the constant C was sufficiently large) all but $2^{-\Omega(l)}n$ balls commit to a bin. Afterwards, we again proceed as in Algorithm $\mathcal{A}_{\text{sym}}$, with k initialized to $4 \cdot 2^{\lfloor \alpha l \rfloor}$ for an appropriate constant $\alpha > 0$; analogously to Theorem 5.2, we obtain the claimed running bound. The bounds on message complexity can be deduced from Chernoff's bound as usual.

Again, choosing $l = \log^{(r)} n$ for any $r \in \mathbb{N}$, Problem 4.3 can be solved within $r + \mathcal{O}(1)$ rounds using $\mathcal{O}(n \log^{(r)} n)$ messages w.h.p.

6 Conclusions

We presented tight bounds for the asymptotic performance of adaptive balls-into-bins algorithms. Our results demonstrate that adaptivity enables substantial improvements on previous parallel balls-into-bins algorithms in terms of the trade-offs between time, maximal load, and communication.

Given that in a totally anonymous setting it is possible to achieve a bin load of 2 within $\log^* n + \mathcal{O}(1)$ rounds, we hope that the proposed techniques may serve to improve future load balancing primitives for decentralized systems. We therefore believe that it is of practical interest to determine the optimal choices of $k(i)$ in Algorithm $\mathcal{A}_{\text{sym}}$ and quantify the resulting performance. Another open question is whether asymmetry and adaptivity can be combined in a simple manner, yielding algorithms that perform well for realistic values of n.

Acknowledgments We would like to thank Mohsen Ghaffari, Fabian Kuhn, Thomas Locher, and Reto Spöhel. This work was partly supported by the Swiss National Fund (SNF), the Society of Swiss Friends of the Weizmann Institute of Science, and the German Research Foundation (DFG, reference number Le 3107/1-1).

References

1. Adler, M., Chakrabarti, S., Mitzenmacher, M., Rasmussen, L.: Parallel randomized load balancing. In: Proc. 27th Symposium on Theory of Computing (STOC), pp. 238–247 (1995)
2. auf der Heide, F.M., Scheideler, C., Stemann, V.: Exploiting storage redundancy to speed up randomized shared memory simulations. Theor. Comput. Sci. **162**(2), 245–281 (1996)
3. Awerbuch, B., Azar, Y., Grove, E.F., Kao, M.Y., Krishnan, P., Vitter, J.S.: Load balancing in the L_p norm. In: Proc. 36th Symposium on Foundations of Computer Science (FOCS), pp. 383–391 (1995)
4. Azar, Y., Broder, A.Z., Karlin, A.R., Upfal, E.: Balanced allocations. SIAM J. Comput. **29**(1), 180–200 (1999)
5. Bast, H., Hagerup, T.: Fast and reliable parallel hashing. In: Proc. 3rd Symposium on Parallel Algorithms and Architectures (SPAA), pp. 50–61 (1991)
6. Berenbrink, P., Friedetzky, T., Goldberg, L.A., Goldberg, P.W., Hu, Z., Martin, R.: Distributed selfish load balancing. SIAM J. Comput. **37**(4), 1163–1181 (2007)
7. Berenbrink, P., Friedetzky, T., Hu, Z., Martin, R.: On weighted balls-into-bins games. Theor. Comput. Sci. **409**(3), 511–520 (2008)
8. Berenbrink, P., auf der Heide, F.M., Schröder, K.: Allocating weighted jobs in parallel. In: Proc. 9th Symposium on Parallel Algorithms and Architectures (SPAA), pp. 302–310 (1997)
9. Czumaj, A., Stemann, V.: Randomized allocation processes. Random Struct. Algorithms **18**(4), 297–331 (2001)
10. Sarma, A.D., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proc. 43rd Symposium on Theory of Computing (STOC) (2011)
11. Dubhashi, D., Ranjan, D.: Balls and bins: a study in negative dependence. Random Struct. Algorithms **13**, 99–124 (1996)
12. Even, G., Medina, M.: Revisiting randomized parallel load balancing algorithms. In: Proc. 16th Colloquium on Structural Information and Communication Complexity (SIROCCO), pp. 209–221 (2009)
13. Even, G., Medina, M.: Parallel randomized load balancing: a lower bound for a more general model. In: Proc. 36th Conference on Theory and Practice of Computer Science (SOFSEM), pp. 358–369 (2010)
14. Fich, F., Ruppert, E.: Hundreds of impossibility results for distributed computing. Distrib. Comput. **16**(2–3), 121–163 (2003)
15. Gil, J., auf der Heide, F.M., Wigderson, A.: The tree model for hashing: lower and upper bounds. SIAM J. Comput. **25**(5), 936–955 (1996)
16. Gonnet, G.H.: Expected length of the longest probe sequence in hash code searching. J. ACM **28**(2), 289–304 (1981)
17. Hagerup, T.: The log-star revolution. In: Proc. 9th Symposium on Theoretical Aspects of Computer Science (STACS), pp. 259–278 (1992)
18. Karp, R.M., Luby, M., auf der Heide, F.M.: Efficient PRAM simulation on a distributed memory machine. Algorithmica **16**, 517–542 (1996)
19. Kenthapadi, K., Panigrahy, R.: Balanced allocation on graphs. In: Proc. 7th Symposium on Discrete Algorithms (SODA), pp. 434–443 (2006)
20. Kleinberg, R., Piliouras, G., Tardos, E.: Load balancing without regret in the bulletin board model. In: Proc. 28th Symposium on Principles of Distributed Computing (PODC), pp. 56–62 (2009)
21. Koutsoupias, E., Mavronicolas, M., Spirakis, P.G.: Approximate equilibria and ball fusion. Theory Comput. Syst. **36**(6), 683–693 (2003)
22. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot Be computed locally! In: Proc. 23rd Symposium on the Principles of Distributed Computing (PODC) (2004)
23. Lenzen, C.: Optimal deterministic routing and sorting on the congested clique. In: Proc. 32nd Symposium on Principles of Distributed Computing (PODC), pp. 42–50 (2013)
24. Lenzen, C., Wattenhofer, R.: Tight bounds for parallel randomized load balancing. Comput. Res. Repos. arXiv:1102.5425 (2011)

25. Lenzen, C., Wattenhofer, R.: Tight bounds for parallel randomized load balancing: extended abstract. In: Proc. 43rd Symposium on Theory of Computing (STOC), pp. 11–20 (2011)
26. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
27. Lotker, Z., Patt-Shamir, B., Peleg, D.: Distributed MST for constant diameter graphs. Distrib. Comput. **18**(6), 453–460 (2006)
28. Lynch, N.: A hundred impossibility proofs for distributed computing. In: Proc. 8th Symposium on Principles of distributed computing (PODC), pp. 1–28 (1989)
29. Matias, Y., Vishkin, U.: Converting high probability into nearly-constant time—with applications to parallel hashing. In: Proc. 23rd Symposium on Theory of Computing (STOC), pp. 307–316 (1991)
30. Mitzenmacher, M.: The Power of Two Choices in Randomized Load Balancing. Ph.D. thesis, University of California, Berkeley (1996)
31. Mitzenmacher, M.: How Useful is Old Information? Tech. rep., Systems Research Center, Digital Equipment Corporation (1998)
32. Mitzenmacher, M.: On the analysis of randomized load balancing schemes. In: Proc. 10th Symposium on Parallel Algorithms and Architectures (SPAA), pp. 292–301 (1998)
33. Mitzenmacher, M., Prabhakar, B., Shah, D.: Load balancing with memory. In: Proc. 43th Symposium on Foundations of Computer Science (FOCS), pp. 799–808 (2002)
34. Mitzenmacher, M., Richa, A., Sitaraman, R.: Handbook of Randomized Computing, vol. 1, chap. The Power of Two Random Choices: A Survey of the Techniques and Results, pp. 255–312. Kluwer Academic Publishers, Dordrecht (2001)
35. Patt-Shamir, B., Teplitsky, M.: The round complexity of distributed sorting: extended abstract. In: Proc. 30th Symposium on Principles of Distributed Computing (PODC), pp. 249–256 (2011)
36. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed mst construction. In: Proc. 40th Symposium on Foundations of Computer Science (FOCS), pp. 253–261 (1999)
37. Peres, Y., Talwar, K., Wieder, U.: The $(1+\beta)$-choice process and weighted balls-into-bins. In: Proc. 21th Symposium on Discrete Algorithms (SODA), pp. 1613–1619 (2010)
38. Raman, R.: 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), pp. 161–175 (1990)
39. Schneider, J., Wattenhofer, R.: A new technique for distributed symmetry breaking. In: 29th Symposium on Principles of Distributed Computing (PODC) (2010)
40. Stemann, V.: Parallel balanced allocations. In: Proc. 8th Symposium on Parallel Algorithms and Architectures (SPAA), pp. 261–269 (1996)
41. Talwar, K., Wieder, U.: Balanced allocations: the weighted case. In: Proc. 39th Symposium on Theory of Computing (STOC), pp. 256–265 (2007)
42. Vöcking, B.: How asymmetry helps load balancing. J. ACM **50**(4), 568–589 (2003)
43. Wieder, U.: Balanced allocations with heterogenous bins. In: Proc. 19th Symposium on Parallel Algorithms and Architectures (SPAA), pp. 188–193 (2007)