The Power of Two Random Choices: A Survey of Techniques and Results

Michael Mitzenmacher * Andréa W. Richa † Ramesh Sitaraman ‡

1 Introduction

To motivate this survey, we begin with a simple problem that demonstrates a powerful fundamental idea. Suppose that n balls are thrown into n bins, with each ball choosing a bin independently and uniformly at random. Then the *maximum load*, or the largest number of balls in any bin, is approximately $\log n / \log \log n$ with high probability.[1] Now suppose instead that the balls are placed sequentially, and each ball is placed in the least loaded of $d \geq 2$ bins chosen independently and uniformly at random. Azar, Broder, Karlin, and Upfal showed that in this case, the maximum load is $\log \log n / \log d + \Theta(1)$ with high probability [ABKU99].

The important implication of this result is that even a small amount of choice can lead to drastically different results in load balancing. Indeed, having just two random choices (i.e., $d = 2$) yields a large reduction in the maximum load over having one choice, while each additional choice beyond two decreases the maximum load by just a constant factor. Over

*Computer Science, Harvard University, Cambridge, MA 02138, michaelm@eecs.harvard.edu. Supported in part by NSF CAREER Award CCR-9983832 and an Alfred P. Sloan Research Fellowship.

†Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287–5406, aricha@asu.edu. Supported in part by NSF CAREER Award CCR-9985284, NSF Grant CCR-9900304, and ASU Faculty-Grant-In-Aid CRR-L032.

‡Department of Computer Science, University of Massachusetts, Amherst, MA 01003, and Akamai Technologies Inc., 500 Technology Square, Cambridge, MA 02139, ramesh@cs.umass.edu. Supported in part by an NSF CAREER Award CCR-9703017.

[1] We use *with high probability* to mean with probability at least $1 - O(1/n^{\alpha})$ for some constant α; generally this α will be 1. A precise analysis shows that the expected maximum load is $\Gamma^{-1}(n) - 3/2 + o(1)$ [Gon81].

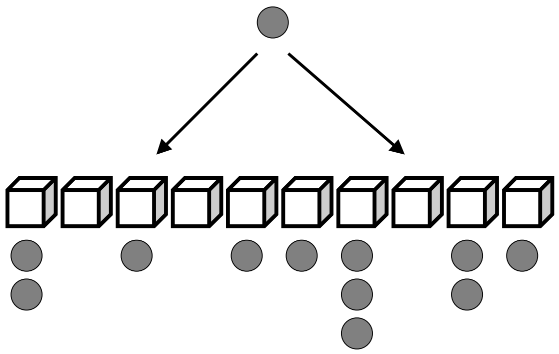

Figure 1: Two choices, the balls-and-bins model.

the past several years, there has been a great deal of research investigating this phenomenon. The picture that has emerged from this research is that the power of two choices is not simply an artifact of the simple balls-and-bins model, but a general and robust phenomenon applicable to a wide variety of situations. Indeed, this *two-choice paradigm* continues to be applied and refined, and new results appear frequently.

1.1 Applications of the two-choice paradigm

The two-choice paradigm and balls-and-bins models have several interesting applications. We outline a few here and we point out more applications in the succeeding sections.

1.1.1 Hashing

Although the balls-and-bins models we discuss may appear simplistic, they have many interesting applications to hashing. In particular, the two-choice paradigm can be used to reduce the maximum time required to search a hash table. The standard hash table implementation [Knu73] uses a single hash function to map keys to entries in a table. If there is a collision, i.e., if two or more keys map to the same table entry, then all the colliding keys are stored in a linked list called a *chain*. Thus, each table entry is the head of a chain and the maximum time to search for a key in the hash table is proportional to the length of the longest chain in the table. If the hash function is *perfectly random* — i.e., if each key is mapped to an entry of the table independently and uniformly at random, and n keys are sequentially inserted into a table with n entries — then the length of the longest chain is $\Theta(\log n/ \log\log n)$ with high probability. This bound follows from the analogous bound on the maximum load in the classical balls-and-bins problem where each ball chooses a single bin independently and uniformly at random.

Now suppose that we use *two* perfectly random hash functions. When inserting a key, we apply both hash functions to determine the two possible table entries where the key can be inserted. Then, of the two possible entries, we add the key to the shorter of the two chains. To search for an element, we have to search through the chains at the two entries given by both hash functions. If n keys are sequentially inserted into the table, the length of the longest chain is $\Theta(\log\log n)$ with high probability, implying that the maximum time needed to search the hash table is $\Theta(\log\log n)$ with high probability. This bound also follows from the analogous bound for the balls-and-bins problem where each ball chooses two bins at random.

In general, using multiple hash functions can be advantageous when the key parameter is the maximum number of keys located on a chain at a table entry. Such situations also arise naturally in practice when one hopes to fit each chain in a single cache line, as described for example in [BM00].

The two-choice paradigm applied to hashing has several advantages over other proposed hashing techniques (e.g., [BK90, DKM+88, FKS84]) in that it uses only two hash functions, it is easy to parallelize, and it is on-line (i.e., it does not involve re-hashing of data). Furthermore, it is not necessary to have perfectly random hash functions: similar results hold by choosing our hash functions randomly from smaller families of hash functions; see [KLM96].

1.1.2 Shared memory emulations on DMMs

One of the earliest applications of the two-choice paradigm is in the study of algorithms to emulate shared memory machines (as, for example, PRAMs) on distributed memory machines (DMMs) [CMS95, KLM96, MSS96]. In such emulations, the processors and the memory cells of the shared memory machine are distributed to the processors and memory modules of the DMM using appropriately chosen (universal) hash functions. Typically, the goal of the emulation algorithm is to minimize slowdown, or delay, of the emulation, which is the time needed by the DMM to emulate one step of the shared memory machine. Several of the balls-and-bins ideas and analysis are relevant in this context since minimizing slowdown involves orchestrating the communication between the processors (the balls) and the memory modules (the bins) so as to avoid memory contention, caused by several processors attempting to access the same memory module.

1.1.3 Load balancing with limited information

Another area where the two-choice paradigm has proven useful is the problem of dynamically assigning tasks to servers (e.g., disk servers or network servers). For simplicity, suppose that all the servers and all the tasks are identical, and that any task can be assigned to any server. Furthermore, suppose that the tasks arrive sequentially and need to be assigned to a server. Clearly, in the interest of response time of the tasks, we would like to keep the maximum load (where here load refers to the number of tasks) of any server as small as possible. Ideally, when a task arrives requesting a server, we would like to assign it to the least loaded server. However, complete information about the loads of all the servers may be expensive to obtain. For instance, querying a server for its load may involve sending a message and waiting for a response, processing an interrupt at the server, etc. An alternative approach that requires no information about the server loads is to simply allocate each task to a random server. If there are n tasks and n servers, using the balls-and-bins analogy, some server is assigned $\Theta(\log n / \log \log n)$ tasks with high probability. If instead each task obtains limited information by querying the load of two servers chosen independently and uniformly at random, and allocates itself to the least loaded of these two servers, then the maximum load on the n servers is only $\Theta(\log \log n)$ with high probability.

1.1.4 Low-congestion circuit routing

Many of the early applications of the two-choice approach have a distinct load balancing flavor. Cole et al. [CMM$^+$98] show that the two-choice paradigm can be applied effectively in a different context, namely, that of routing virtual circuits in interconnection networks with low congestion. They show how to incorporate the two-choice approach to a well-studied paradigm due to Valiant for routing virtual circuits to achieve significantly lower congestion, as we discuss in Section 3.

1.2 A brief history

We now provide a brief history of research on the two-choice paradigm. In the sections that follow, we discuss the results in more detail.

The earliest work we know that applies the two-choice paradigm to load balancing is that of Eager, Lazowska, and Zahorjan [ELZ86a]. The authors provide empirical evidence that a load balancing system based on allowing tasks to migrate to the least loaded of randomly selected processors improves performance. They also derive analytical results based on an appropriate Markov model. Their approach is related to fluid limit models. Recent work by Vvedenskaya, Dobrushin, and Karpelevich [VDK96] and Mitzenmacher [Mit96b, Mit96a] has led to an enduring technique for analysis of these load balancing systems based on fluid limit models, as described in Section 4.

The first rigorous analytical demonstration of the power of two choices is due to Karp, Luby, and Meyer auf der Heide [KLM92, KLM96], who considered the possibility of using two hash functions in the context of PRAM emulation by DMMs. Subsequent work on shared memory emulations on DMMs [CMS95, MSS96] has given rise to a powerful technique for analysis called the witness tree method. (See Section 3 for more details on this technique.)

The balls-and-bins problem has proven to be a fertile ground for investigating the power of two choices. The *classical balls-and-bins problem*, where each ball is thrown into a bin chosen independently and uniformly at random, has been studied for several decades [JK77]. Azar, Broder, Karlin, and Upfal [ABKU99] first considered the *sequential multiple-choice* balls-and-bins problem, where each ball chooses $d \geq 2$ bins independently and uniformly at random, and the balls are thrown sequentially into the least loaded of its d bin choices. This seminal paper introduced an important and intuitive technique for analyzing algorithms that use the two-choice paradigm, known as the layered induction method. In Section 2, we present this technique in more detail. Adler et al. [ACMR95] introduced the *parallel multiple-choice balls-and-bins problem* where each ball chooses $d \geq 2$ random bins independently and uniformly at random, but the balls must be assigned to bins in parallel by performing a limited number of rounds of communication. Since this survey focuses on results that show the power of having multiple choices in balls-and-bins problems, whenever we refer to a balls-and-bins problem in the remainder of this chapter, we will implicitly be referring to a *multiple-choice* balls-and-bins problem (i.e., a problem where each ball is assigned $d \geq 2$ random bin choices), unless we clearly state otherwise.

A balls-and-bins problem (such as those described above) where balls are inserted but never deleted from the system is referred to as a *static* problem. In a *dynamic problem*, balls can also be deleted from the system. Azar et al. [ABKU99] introduced a simple dynamic model for the sequential balls-and-bins problem, in which at each step a random ball is deleted and a new ball is inserted into the system; each time a ball is inserted, it is placed in the least loaded of two bins chosen independently and uniformly at random. An alternative to *random deletions* is *adversarial deletions* where an oblivious adversary decides on the sequence of insertions and deletions of the balls. (In this context, an *oblivious adversary* is one that specifies the sequence of insertions and deletions of the balls in advance, without knowledge of the random bin choices of the balls.) Only recently, this more powerful dynamic model has been analyzed [CMM+98, CFM+98]. Dynamic models have also been explored in connection with the parallel balls-and-bins problem. For instance, Adler et al. [ABS98] consider the situation where the balls are queued in first-in, first-out (FIFO) order at each bin and the first ball in each queue is deleted at each time step.

Finally, a *uniform balls-and-bins problem* (again, such as the ones described above) is a d-choice balls-and-bins problem where the d random choices assigned to a ball are independent and uniform. Vöcking [Vöc99] was the first to show how nonuniform ball placement strategies can help reduce the maximum bin load.

1.3 The three major techniques

One interesting aspect of the work in this rich area is that several different techniques have proven useful. The main techniques used to analyze balls-and-bins problems are layered induction, witness trees, and fluid limits via differential equations. Our survey is organized by successively discussing these three techniques, so that our focus is at least as much on the techniques as on the results obtained by using them. In fact, we demonstrate all of these approaches with examples. In presenting our survey in this manner, we hope to provide the reader with the necessary tools to pursue further research in this area.

In Section 2, we discuss the *layered induction* technique pioneered by Azar, Broder, Karlin, and Upfal [ABKU99]. In this approach, we bound the maximum load by bounding the number of bins with k or more balls via induction on k. The layered induction approach provides nearly tight results and a straightforward attack for handling balls-and-bins problems. It has proven effective for much more than the original problem studied in [ABKU99]. For example, as we explain, the layered induction approach can be used in a dynamic setting where an adversary inserts and deletes balls from the system over time.

In Section 3, we discuss an alternative technique for handling these problems called the *witness tree* method. The key idea of this approach is to show that if a "bad event" occurs — in our case, if some bin is heavily loaded — one can extract from the history of the process a suitable "tree of events" called the witness tree. The probability of the bad event can then be bounded by the probability of occurrence of a witness tree. Generally, witness tree arguments involve the most complexity, and they have proven to be the most challenging in terms of obtaining tight results. This complexity, however, yields strength: witness tree arguments tend to provide the strongest results, especially for dynamic settings that include both deletions and re-insertions of items. Furthermore, the more sophisticated uses of the two-choice paradigm in the design of communication protocols are more easily amenable to witness tree analyses.

In Section 4, we discuss a final technique, which studies algorithms that use the two-choice paradigm via *fluid limit models*. If one pictures the size (in this case, the number of bins) of the system growing to infinity, the resulting system can then be described by an appropriate family of differential equations. The fluid limit approach is more standard in the queueing theory literature, where it is used more widely, and it has proven especially useful for studying variations of the balls-and-bins problems that map naturally to queueing problems. A major weakness of the fluid limit approach is that even minor dependencies in the system can make the approach untenable. Also, its theoretical basis is at times incomplete; one must often return to a more concrete probabilistic argument to obtain the desired results. In many ways, however, it is the simplest and most flexible of the three methods. Moreover, when a problem can be put in this framework, the differential equations generally yield extremely accurate numerical results.

2 The Layered Induction Approach

In this section, we address the results in the balls-and-bins literature that follow the layered induction approach introduced by Azar, Broder, Karlin, and Upfal in [ABKU99]. In this approach, we inductively bound the number of bins that contain at least j balls conditioned on the number of bins that contain at least $j-1$ balls. Azar et al. show that, in the sequential d-choice balls-and-bins problem, the maximum load of a bin is $\log\log n/\log d + \Theta(1)$ with high probability. They also show that this bound is optimal (up to an additive constant term) among all the uniform multiple-choice placement strategies.

The layered induction approach in fact proved to be also useful in dynamic scenarios. For example, Azar et al. analyze the situation where at each step a random ball is deleted and a new ball is inserted in the system using layered induction [ABKU99]. Recently some progress was made in analyzing the behavior of the balls-and-bins problem under more realistic deletion scenarios. In [CFM$^+$98], Cole et al. consider two natural situations: the particular case where balls that have been in the system for the longest time are deleted, and the more general case where an adversary specifies the sequence of insertions and deletions in advance. They show how to use layered induction arguments to provide simple proofs of upper bounds on the maximum bin load for these two scenarios. One of the main contributions of Cole et al. in [CFM$^+$98] was to demonstrate how the layered induction techniques can yield interesting results in realistic deletion scenarios.

This section is organized as follows: in Section 2.1, we show the layered induction approach introduced by Azar et al. for the sequential balls-and-bins problem, and show how this approach can be modified to handle some extensions of the sequential problem; Section 2.2 shows how the layered induction approach can be adapted to actually prove lower bound results.

2.1 The approach

In this section, we describe the main results in the balls-and-bins literature that use layered induction for placing an upper bound on the maximum bin load. Our main goal is to make the reader understand the basic layered induction techniques in detail, so we start by presenting the simple layered induction argument for the sequential balls-and-bins problem due to Azar et al. [ABKU99]. Then we present other balls-and-bins results obtained by layered induction, and show how to modify the original argument of Azar et al. to hold for these results. The proof and notation we present here are very close to the original paper by Azar et al. We have made some minor changes in the interest of clarity.

Theorem 1 *Suppose that n balls are sequentially placed into n bins. Each ball is placed in the least full bin at the time of the placement, among d bins, $d \geq 2$, chosen independently and uniformly at random. Then after all the balls are placed, with high probability the number of balls in the fullest bin is at most* $\log \log n / \log d + O(1)$.

Azar et al. also show in [ABKU99] that the maximum bin load is at least $\log \log n / \log d - O(1)$ (see Section 2.2), proving that the maximum bin load for this problem is in fact equal to $\log \log n / \log d + \Theta(1)$.

Before presenting the proof, which is somewhat technical, we briefly sketch an intuitive analysis. For any given i, instead of trying to determine the number of bins with load *exactly* i, it is easier to study the number of bins with load *at least* i. The argument proceeds via what is, for the most part, a straightforward induction. Let the *height* of a ball be one more than the number of balls already in the bin in which the ball is placed. That is, if we think of balls as being stacked in the bin by order of arrival, the height of a ball is its position in the stack. Suppose we know that the number of bins with load at least i, over the entire course of the process, is bounded above by β_i. We wish to find a β_{i+1} such that, with high probability, the number of bins with load at least $i+1$ is bounded above by β_{i+1} over the course of the entire process with high probability. We find an appropriate β_{i+1} by bounding the number of balls of height at least $i+1$, which gives a bound for the number of bins with at least $i+1$ balls.

A ball has height at least $i+1$ only if, for each of the d times it chooses a random bin, it chooses one with load at least i. Conditioned on the value of β_i, the probability that each choice finds a bin of load at least i is $\frac{\beta_i}{n}$. Therefore the probability that a ball thrown any time during the process joins a bin already containing i or more balls is at most $\left(\frac{\beta_i}{n}\right)^d$. For $d \geq 2$, we can conclude that the sequence β_i/n drops at least quadratically at each step in the following manner. The number of balls with height $i+1$ or more is stochastically dominated by a Bernoulli random variable, corresponding to the number of heads with n (the number of balls) flips, with the probability of a head being $\left(\frac{\beta_i}{n}\right)^d$ (the probability of a ball being placed in a bin with i or more balls). We can find an appropriate β_{i+1} using standard bounds on Bernoulli trials, yielding $\beta_{i+1} \leq cn\left(\frac{\beta_i}{n}\right)^d$, for some constant c. The fraction $\frac{\beta_i}{n}$ therefore drops at least quadratically at each step, so that after only $j = O(\log\log n)$ steps the fraction drops below $1/n$, and we may conclude that $\beta_j < 1$. The proof is technically challenging primarily because one must handle the conditioning appropriately.

We shall use the following notation: the state at time t refers to the state of the system immediately after the tth ball is placed. $B(n,p)$ is a Bernoulli random variable with parameters n and p. The variable $h(t)$ denotes the height of the tth ball, and $\nu_i(t)$ and $\mu_i(t)$ refer to the number of bins with load at least i and the number of balls with height at least i at time t, respectively. We use ν_i and μ_i for $\nu_i(n)$ and $\mu_i(n)$ when the meaning is clear.

In preparation for the detailed proof, we make note of two elementary lemmas. The first statement can be proven by standard coupling methods:

Lemma 2 *Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables in an arbitrary domain, and let $Y_1, Y_2, \ldots, Y_n$ be a sequence of binary random vari-*

ables, with the property that $Y_i = Y_i(X_1, \ldots, X_{i-1})$. *If*

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

then

$$\mathbf{Pr}(\sum_{i=1}^{n} Y_i \geq k) \leq \mathbf{Pr}(B(n,p) \geq k);$$

and similarly, if

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

then

$$\mathbf{Pr}(\sum_{i=1}^{n} Y_i \leq k) \leq \mathbf{Pr}(B(n,p) \leq k).$$

■

The second lemma presents some useful Chernoff-type bounds; proofs may be found in [HR90].

Lemma 3 *If* X_i $(1 \leq i \leq n)$ *are independent binary random variables,* $\mathbf{Pr}[X_i = 1] = p$, *then the following hold:*

$$\text{For } t \geq np, \qquad \mathbf{Pr}\left(\sum_{i=1}^{n} X_i \geq t\right) \leq \left(\frac{np}{t}\right)^t \mathrm{e}^{t-np}. \tag{1}$$

$$\text{For } t \leq np, \qquad \mathbf{Pr}\left(\sum_{i=1}^{n} X_i \leq t\right) \leq \left(\frac{np}{t}\right)^t \mathrm{e}^{t-np}. \tag{2}$$

In particular, we have

$$\mathbf{Pr}\left(\sum_{i=1}^{n} X_i \geq \mathrm{e}np\right) \leq \mathrm{e}^{-np}, \quad \textit{and} \tag{3}$$

$$\mathbf{Pr}\left(\sum_{i=1}^{n} X_i \leq np/\mathrm{e}\right) \leq \mathrm{e}^{(\frac{2}{e}-1)np}. \tag{4}$$

■

Proof:[Proof of Theorem 1:] Following the earlier sketch, we shall construct values β_i so that $\nu_i(n) \leq \beta_i$, for all i, with high probability. Let $\beta_6 = \frac{n}{2\mathrm{e}}$, and $\beta_{i+1} = \frac{\mathrm{e}\beta_i^d}{n^{d-1}}$, for $6 \leq i < i^*$, where i^* is to be determined.

We let $\mathcal{E}_i$ be the event that $\nu_i(n) \leq \beta_i$. Note that $\mathcal{E}_6$ holds with certainty. We now show that, with high probability, if $\mathcal{E}_i$ holds then $\mathcal{E}_{i+1}$ holds, for $6 \leq i \leq i^* - 1$.

Fix a value of i in the given range. Let Y_t be a binary random variable such that

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

That is, Y_t is 1 if the height of the tth ball is at least $i+1$ and at time $t-1$ there are fewer than β_i bins with load at least i.

Let ω_j represent the bins selected by the jth ball. Then

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

Thus, from Lemma 2, we may conclude that

$$\mathbf{Pr}(\textstyle\sum_{i=1}^{n} Y_t \geq k) \leq \mathbf{Pr}(B(n, p_i) \geq k).$$

Conditioned on $\mathcal{E}_i$, we have $\sum Y_t = \mu_{i+1}$. Thus

$$\begin{aligned} \mathbf{Pr}(\nu_{i+1} \geq k \mid \mathcal{E}_i) &\leq \mathbf{Pr}(\mu_{i+1} \geq k \mid \mathcal{E}_i) \\ &= \mathbf{Pr}(\textstyle\sum Y_t \geq k \mid \mathcal{E}_i) \\ &\leq \frac{\mathbf{Pr}(\sum Y_t \geq k)}{\mathbf{Pr}(\mathcal{E}_i)} \\ &\leq \frac{\mathbf{Pr}(B(n, p_i) \geq k)}{\mathbf{Pr}(\mathcal{E}_i)} \end{aligned}$$

We bound the tail of the binomial distribution using Equation (3). Letting $k = \beta_{i+1}$ in the above, we have that

$$\mathbf{Pr}(\nu_{i+1} \geq \beta_{i+1} \mid \mathcal{E}_i) \leq \frac{\mathbf{Pr}(B(n, p_i) \geq enp_i)}{\mathbf{Pr}(\mathcal{E}_i)} \leq \frac{1}{\mathrm{e}^{p_i n} \mathbf{Pr}(\mathcal{E}_i)},$$

or that

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

whenever $p_i n \geq 2 \log n$.

Hence, whenever $p_i n \geq 2 \log n$, we have that if $\mathcal{E}_i$ holds with high probability, then so does $\mathcal{E}_{i+1}$. To conclude we need to handle the case where $p_i n \leq 2 \log n$ separately: we shall show that if this is the case, then with high probability there are no balls of height at least $i+2$. Let i^* be the smallest value of i such that $\frac{\beta_i^d}{n^d} \leq \frac{2\log n}{n}$. It is easy to check inductively that $\beta_{i+6} \leq n/2^{d^i}$, and hence that $i^* \leq \frac{\log\log n}{\log d} + O(1)$.

We have

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

where the second inequality again follows from Equation (3). Also,

$$\mathbf{Pr}(\mu_{i^*+2} \geq 1 | \mu_{i^*+1} \leq 6\log n) \leq \frac{\mathbf{Pr}(B(n, (6\log n/n)^d) \geq 1)}{\mathbf{Pr}(\mu_{i^*+1} \leq 6\log n)} \leq \frac{n(6\log n/n)^d}{\mathbf{Pr}(\mu_{i^*+1} \leq 6\log n)},$$

where the second inequality comes from applying the crude union bound. We remove the conditioning using the fact that

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

to obtain that

$$\mathbf{Pr}(\mu_{i^*+2} \geq 1) \leq \frac{(6\log n)^d}{n^{d-1}} + \frac{i^*+1}{n^2} = O\left(\frac{1}{n}\right),$$

which implies that with high probability the maximum bin load is less than $i^* + 2 = \log\log n/\log d + O(1)$. ■

We now present some of the extensions of the sequential balls-and-bins problem which were analyzed using layered induction. For each of these extensions, we give a brief sketch on how to modify the argument in the proof of Theorem 1 to hold for the new balls-and-bins problem. We refer to the respective papers for the complete proofs.

We start by considering the extensions of the sequential problem which appear in [ABKU99]. Azar et al. consider the case when the number of balls may not be equal to the number of bins in the system. Let m denote the number of balls to be sequentially inserted into the n bins, where each ball makes d bin choices independently and uniformly at random, and is placed in the least filled of the d bins. Azar et al. show that the maximum bin load is now $(\log\log n/\log d)(1 + o(1)) + \Theta(m/n)$ with high probability. The major changes in the proof of Theorem 1 in order to hold for this case are in the definition of the values β_i and in the choice of the base case for our inductive process (in Theorem 1, we chose the base case to be $i = 6$). Here we let $\beta_x = n^2/(2em)$, for some convenient choice of base case x, and we require that $\mathbf{Pr}(\nu_x \geq \frac{n^2}{2em})$ holds with high probability. Then we define the variable β_{i+x} so as to be less than or equal to $\frac{n}{2^{d^i}}$, for all i, thus obtaining (using the same analysis as in the proof of Theorem 1) that

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

The main challenge needed to complete the proof is to show that x can be chosen to be $O(m/n) + o(\log\log n/\log d)$. Note that when $m >> n$, the bound on the maximum bin load is asymptotically optimal. The heavily loaded case where $m >> n$ was also recently studied in more detail in [BCSV00].

Azar et al. also consider a dynamic extension of the sequential problem in [ABKU99], as described in the following theorem:

Theorem 4 *Consider the infinite process where at each step, a ball is chosen independently and uniformly at random to be removed from the system, and a new ball is inserted in the system. Each new ball inserted in the system chooses $d \geq 2$ possible destination bins independently and uniformly at random, and is placed in the least full of these bins. This process may start at any arbitrary state, provided we have at most n balls in the system. For any fixed $T > n^3$, the fullest bin at time T contains, with high probability, fewer than $\log\log n/\log d + O(1)$ balls.*

The analysis of the case $d = 1$ for the infinite stochastic process defined in Theorem 4 is simple, since the location of a ball does not depend on the locations of any other balls in the system. Thus for $d = 1$, in the stationary distribution, with high probability the fullest bin has $\Theta(\log n/\log\log n)$ balls. The analysis of the case $d \geq 2$ is significantly harder, since the locations of the current n balls might depend on the locations of balls that are no longer in the system. By the definition of the process, the number of balls of height i cannot change by more than 1 in a time step. Hence the variable $\mu_{\geq i}(t)$ can be viewed as a random walk on the integers ℓ, $0 \leq \ell \leq n$. The proof of Theorem 4 is based on bounding the maximum values taken by the variables $\mu_{\geq i}(t)$ by studying the underlying process.

Only recently, Cole et al. [CFM+98] showed how to use layered induction to address the more realistic deletion scenarios in Theorem 5 below.

Theorem 5 *Consider the polynomial time process where in the first n steps, a new ball is inserted into the system, and where at each subsequent time step, either a ball is removed or a new ball is inserted in the system, provided that the number of balls present in the system never exceeds n. Each new ball inserted in the system chooses $d \geq 2$ possible destination bins independently and uniformly at random, and is placed in the least full of these* *bins. Suppose that an adversary specifies the full sequence of insertions and deletions of balls in advance, without knowledge of the random choices of the new balls that will be inserted in the system (i.e., suppose we have an oblivious adversary). If this process runs for at most* n^c *time steps, where c is any positive constant, then the maximum load of a bin during the process is at most* $\log\log n/\log d + O(1)$*, with high probability.*

Cole et al. show that the original argument of Azar et al. for the sequential balls-and-bins problem can in fact be made to hold in this dynamic scenario: the key difference between this result and that of [ABKU99] is that Azar et al. find a dominating distribution of heights on one set of n balls, whereas Cole et al. use a distribution that applies to every set of n balls present in the system as it evolves. As it happens, the bounds and the proof are essentially the same; the most significant changes lie in the end game, where we must bound the number of bins containing more than $\log\log n/\log d$ balls.

Cole et al. also consider a situation where items that have been in the system for the longest time are deleted, again using a variant of the layered induction argument in [ABKU99]. In this case initially $2n$ balls are inserted, and then repeatedly the oldest n balls are deleted and n new balls are inserted. This argument makes use of a two-dimensional family of random variables, similar in spirit to the work of [Mit00] (which we address in Section 4). The bounds are the same as in Theorem 5, and hence the results are actually already implied by this theorem. However, the approach used in the proof for this specialized case may provide interesting results when applied to other problems, not only in the balls-and-bins domain. See [CFM+98] for the complete proofs.

2.1.1 Bounds on the recovery time

Suppose we start with a situtation where n balls are allocated to n bins in some *arbitrary* fashion. Now, consider the infinite process of Theorem 4 where at each time step a ball chosen independently and uniformly at random is deleted, and a new ball is inserted into the least loaded of d bins chosen independently and uniformly at random. How many time steps does it take for the system to approach steady-state (i.e., typical) behavior? More specifically, how many time steps does it take for the maximum load to be $\log\log n/\log d + O(1)$ with high probabilty? This quantity that is related to the mixing time of the underlying Markov process is called the *recovery time.* The recovery time quantifies the transient behavior of the system and is a useful measure of how quickly the system recovers from an arbitrarily bad configuration. It turns out that the bound of n^3 time steps for the recovery time in Theorem 4 is not tight. Czumaj and Stemann [CS97] provide a tight bound using a standard probabilistic tool called the coupling method [Lin92]. Specifically, they show that after $(1+o(1))n \ln n$ steps the maximum load is $\log\log n/\log d + O(1)$ with high probability. The proof of this result was later simplified significantly by Czumaj [Czu98] via the use of the path coupling method[2] of Bubley and Dyer [BD97]. Czumaj also considers a variation of the infinite process in Theorem 4 where deletions are performed differently. In the new process instead of deleting a random ball, each deletion is performed by choosing a non-empty bin independently and uniformly at random and deleting a ball from the chosen bin. He shows that even though the new deletion process does not significantly affect steady-state behavior, the recovery time of the new process is at least $\Omega(n^2)$ and at most $O(n^2 \ln n)$, i.e., the recovery time of the new process is significantly larger.

2.2 How to use layered induction to prove lower bounds

In this section, we illustrate how to use layered induction to prove lower bounds. We show how we can adapt the argument in the proof of Theorem 1 to provide a lower bound on the maximum number of balls in a bin for the sequential balls-and-bins problem. More specifically, a corresponding lower bound of $\log\log n/\log d - O(1)$ is presented, based on the following idea: first we bound the number of bins with load at least 1 after the $(n/2)$th ball in inserted, then we bound the number of bins of height 2 after the $(3n/4)$th ball, etc. This lower bound, combined with the results in Theorem 1, demonstrates that the maximum bin load for the sequential d-choice balls-and-bins problem is in fact $\log\log n/\log d + \Theta(1)$ with high probability. The proof is taken from [ABKU99].

Before proving this result, we note that Azar et al. actually proved that the greedy strategy is stochastically optimal among all possible multiple-choice uniform placement strategies [ABKU99]. (Recall that a d-choice uniform placement strategy is a placement strategy where all d random bin choices assigned to a ball are independent and uniform). Equivalently, the probability that the maximum height exceeds any value z for any uniform placement strategy based on d choices is smallest when the bin with the least number of balls is chosen. Hence their result is the best possible, for

[2] The path coupling technique when applicable, is easier to use than standard coupling; see [Jer98] for a good survey of this technique.

uniform placement strategies. (It is worth noting that Vöcking [Vöc99] uses a placement strategy that is not uniform to beat this lower bound, as we discuss in Section 3.) They show this by establishing a one-to-one correspondence between the possible results under the proposed greedy strategy and any other fixed strategy. This one-to-one correspondence matches results so that the maximum load for each possible result pair is smaller using Azar et al.'s greedy placement strategy. This is an example of a simple stochastic comparison; for more on this area, see [Sto83, SS94].

Theorem 6 *Suppose that n balls are sequentially placed into n bins. Each ball is placed in the least full bin at the time of the placement, among d bins, $d \geq 2$, chosen independently and uniformly at random. Then after all the balls are placed the number of balls in the fullest bin is at least $\log \log n / \log d - O(1)$ with high probability.*

Proof: Let $\mathcal{F}_i$ be the event that $\nu_{\geq i}(t) \geq \gamma_i$, where the variables γ_i are such that $\gamma_{i+1} < \gamma_i/2$ (the variables γ_i will be revealed shortly). In fact, each $\gamma_i < n/2^{2^i}$. We want to upper bound

$$\mathbf{Pr}(\neg \mathcal{F}_{i+1} \mid \mathcal{F}_i).$$

Our goal is to show that, given $\mathcal{F}_i$, $\mathcal{F}_{i+1}$ holds with high probability.

We fix $i > 0$ and define the binary random variables Z_t for t in the range $R = [(1 - 1/2^i)n, (1 - 1/2^{i+1})n)$ so that

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

That is, the value Z_t is 1 if and only if the height of the tth ball equals $i+1$ or there are already γ_{i+1} bins with load at least $i+1$ at time $t-1$. Note that, as i increases, we consider the values of Z_t over shorter but further out time intervals. The intuition here is that in order to show that there are at least so many bins with load $i+1$ at time $(1 - 1/2^{i+1})n$, we start counting balls with that height from time $(1 - 1/2^i)n$; we wait until that point in time in order to ensure that there are sufficiently many bins with load i to make counting balls with height $i+1$ worthwhile. We can get away with decreasing the amount of time we count balls as i increases, since the values γ_i decrease so fast.

Our definition of Z_t implies that as long as $\nu_{\geq i+1}(t-1) \leq \gamma_{i+1}$, then $Z_t = 1$ precisely when all d choices have load at least i, and at least one of the d choices for the tth ball has load exactly i. Let ω_j represent the choices available to the jth ball. Then

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

Hence

$$\mathbf{Pr}\left(\sum_{t\in R} Z_t \le k \mid \mathcal{F}_i\right) \le \mathbf{Pr}(B(n/2^{i+1}, p_i) \le k).$$

By choosing

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

we may conclude that

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

as long as $p_i n/2^{i+1} \ge 17 \ln n$ by using a tail bound such as [AS92]

$$\mathbf{Pr}(B(N,p) < Np/2) < e^{-Np/8}.$$

Let i^* be the largest integer for which the tail bound holds. Clearly $i^* = \ln\ln n/\ln d - O(1) = \log\log n/\log d - O(1)$.

Now by the definition of Z_t, the event $\{\sum_{t\in R} Z_t \ge \gamma_{i+1}\}$ implies $\mathcal{F}_{i+1}$. Hence

$$\mathbf{Pr}(\neg\mathcal{F}_{i+1} \mid \mathcal{F}_i) \le \mathbf{Pr}(\sum_{t\in R} Z_t < \gamma_{i+1} \mid \mathcal{F}_i) = o(1/n^2).$$

Thus for sufficiently large n

$$\begin{aligned}\mathbf{Pr}(\mathcal{F}_{i^*}) &= \mathbf{Pr}(\mathcal{F}_{i^*} \mid \mathcal{F}_{i^*-1})\cdot\mathbf{Pr}(\mathcal{F}_{i^*-1} \mid \mathcal{F}_{i^*-2})\cdot\ldots\cdot\mathbf{Pr}(\mathcal{F}_1 \mid \mathcal{F}_0)\cdot\mathcal{F}_0\\ &\ge (1-1/n^2)^{i^*} = 1-o(1/n).\end{aligned}$$

■

3 The Witness Tree Method

Another powerful technique for analyzing balls-and-bins problems is the witness tree method. Suppose that we would like to bound the probability of the occurrence of some "bad event", such as the probability of the occurrence of a "heavily-loaded" bin. The key idea is to show that the occurrence of the bad event implies the occurrence of a "tree of events" called the witness tree. Thus, the probability that the bad event occurs is at most the probability that some witness tree occurs. The latter probability can in turn be bounded by enumerating all possible witness trees and summing their individual probabilities of occurrence.[3]

One of the earliest uses of the witness tree method occurs in the study of algorithms to emulate shared memory machines (as for example, PRAMs) on distributed memory machines (DMMs) [CMS95, MSS96]. Besides shared memory emulations, witness trees were independently discovered and used in the context of the parallel balls-and-bins problem [ACMR95].

3.1 The sequential balls-and-bins problem

We start by providing a simple analysis of a variant of the sequential balls-and-bins problem using the witness tree technique. The proof provided here is adapted from [CFM+98], but all the essential ideas in the proof were used earlier in the analysis of randomized circuit-switching algorithms [CMM+98].

The problem that we wish to study can be described formally as a random process $Q_d(\vec{v}, \vec{w})$, where $\vec{v} = (v_1, v_2, \cdots)$, and $\vec{w} = (v_1, v_2, \cdots)$ are (infinite) vectors that specify the identity of the balls to be deleted and inserted respectively. The process begins with n insertions, where n is the total number of bins, followed by an alternating sequence of deletions and insertions specified by $\vec{v}$ and $\vec{w}$ respectively.[4] We assign each ball a unique ID number, and without loss of generality we assume the first n balls have ID numbers 1 through n. At time $n + j$, the ball with ID number v_j is deleted and then the ball with ID number w_j is inserted. If ball w_j has never been inserted before, then it is placed in the least loaded of d bins chosen independently and uniformly at random. If the ball has been inserted before, it is placed in the least loaded (at time $n + j$, after the deletion of ball v_j) of the d bins chosen when it was first inserted; that is, the bin choices of a ball are fixed when it is first inserted in the system. We assume that $\vec{v}$ and $\vec{w}$ are consistent, so there is only one ball with a given ID number in the system at a time. Note also that $\vec{v}$ and $\vec{w}$ must be chosen by the adversary before the process begins, without reference to the random choices made during the course of the process. For simplicity, we now consider only the special case $d = 2$.

Theorem 7 *At any time t, with probability at least $1 - 1/n^{\Omega(\log\log n)}$, the maximum load of a bin achieved by process $Q_2(\vec{v}, \vec{w})$ is $4 \log\log n$.*

[3] The witness tree method is similar in spirit to the delay sequence method used to bound the message latencies of routing algorithms [Upf84, Ale82, LMRR94, MS92].

[4] The fact that insertions and deletions alternate is not crucial except to ensure that the total number of balls in the system at any given time is at most n.

Proof: We prove the theorem in two parts. First, we show that if there is a bin r at time t with 4ℓ balls, where $\ell = \log\log n$, then there exists a degree ℓ pruned witness tree. Next, we show that with high probability, no degree ℓ pruned witness tree exists.

Constructing a witness tree. A witness tree is a *labeled* tree in which each node represents a bin and each edge (r_i, r_j) represents a ball whose two bin choices are r_i and r_j. Suppose that some bin r has load 4ℓ at time t. We construct the witness tree as follows. The root of the tree corresponds to bin r. Let $b_1, \ldots, b_{4\ell}$ be the balls in r at time t. Let r_i be the other bin choice associated with ball b_i (one of the choices is bin r). The root r has 4ℓ children, one corresponding to each bin r_i. Let $t_i < t$ be the last time b_i was (re-)inserted into the system. Without loss of generality, assume that $t_1 < t_2 < \ldots < t_{4\ell}$. Note that the height of ball b_i when it was inserted at time t_i is at least i (since balls $b_1, \ldots, b_{i-1}$ were already in bin r at time t_i). Therefore, the load of bin r_i, the other choice of b_i, is at least $i-1$ at time t_i. We use this fact to recursively grow a tree rooted at each r_i.

The witness tree we have described is irregular. However, it contains as a subgraph an ℓ-ary tree of height ℓ such that

- The root in level 0 has ℓ children that are internal nodes.
- Each internal node on levels 1 to $\ell-2$ has two children that are internal nodes and $\ell-2$ children that are leaves.
- Each internal node on level $\ell-1$ has ℓ children that are leaves.

For convenience we refer to this subtree as the actual witness tree henceforth.

Constructing a pruned witness tree. If the nodes of the witness tree are guaranteed to represent distinct bins, proving our probabilistic bound is a relatively easy matter. However, this is not the case; a bin may reappear several times in a witness tree, leading to dependencies that are difficult to resolve. This makes it necessary to *prune* the tree so that each node in the tree represents a distinct bin. Consequently, the balls represented by the edges of the pruned witness tree are also distinct. In this regard, note that a ball appears at most once in a pruned witness tree, even if it was (re-)inserted multiple times in the sequence.

We visit the nodes of the witness tree iteratively in *breadth-first* search order starting at the root. As we proceed, we remove (i.e., prune) some nodes of the tree and the subtrees rooted at these nodes – what remains is the pruned witness tree. We start by visiting the root. In each iteration, we visit the next node v in breadth-first order that has not been pruned. Let $B(v)$ denote the set of nodes visited *before* v.

- If v represents a bin that is different from the bins represented by nodes in $B(v)$, we do nothing.
- Otherwise, prune all nodes in the subtree rooted at v. Then, we mark the edge from v to its parent as a *pruning edge*.

Note that the pruning edges are not part of the pruned witness tree. The procedure continues until either no more nodes remain to be visited or there are ℓ pruning edges. In the latter case, we apply a final pruning by removing all nodes that are yet to be visited. (Note that this final pruning produces no new pruning edges.) The tree that results from this pruning process is the pruned witness tree. After the pruning is complete, we make a second pass through the tree and construct a set C of *pruning balls*. Initially, C is set to $\emptyset$. We visit the pruning edges in BFS order and for each pruning edge (u, v) we add the ball corresponding to (u, v) to C, if this ball is distinct from all balls currently in C and if $|C| \leq \lceil p/2 \rceil$, where p is the total number of pruning edges.

Lemma 8 *The pruned witness tree constructed above has the following properties.*

1. *All nodes in the pruned witness represent distinct bins.*
2. *All edges in the pruned witness tree represent distinct balls. (Note that pruning edges are not included in the pruned witness tree.)*
3. *The pruning balls in C are distinct from each other, and from the balls represented in the pruned witness tree.*
4. *There are $\lceil p/2 \rceil$ pruning balls in C, where p is the number of pruning edges.*

Proof: The first three properties follow from the construction. We prove the fourth property as follows. Let b be a ball represented by some pruning edge, and let v and w be its bin choices. Since v and w can appear at most once as nodes in the pruned witness tree, ball b can be represented by at most two pruning edges. Thus, there are $\lceil p/2 \rceil$ distinct pruning balls in C. ∎

Enumerating pruned witness trees. We bound the probability that a pruned witness tree exists by bounding both the number of possible pruned witness trees and the probability that each such tree could arise. First, we choose the shape of the pruned witness tree. Then, we traverse the tree in breadth-first order and bound the number of choices for the bins for each tree node and the balls for each tree edge; we also bound the associated probability that these choices came to pass. Finally, we consider the number of choices for pruning balls in C and the corresponding probability that they arose. Multiplying these quantities together yields the final bound – it is important to note here that we can multiply terms together only because all the balls in the pruned witness tree and the pruning balls in C are all distinct.

Ways of choosing the shape of the pruned witness tree. Assume that there are p pruning edges in the pruned tree. The number of ways of selecting the p pruning edges is at most

$$\binom{\ell^2 2^\ell}{p} \leq \ell^{2p} 2^{\ell p},$$

since there are at most $\ell^2 2^\ell$ nodes in the pruned witness tree.

Ways of choosing balls and bins for the nodes and edges of the pruned witness tree. The enumeration proceeds by considering the nodes in BFS order. The number of ways of choosing the bin associated with the root is n. Assume that you are considering the ith internal node v_i of the pruned witness tree whose bin has already been chosen to be r_i. Let v_i have δ_i children. We evaluate the number of ways of choosing a distinct bin for each of the δ_i children of v_i and choosing a distinct ball for each of the δ_i edges incident on v_i and weight it by multiplying by the appropriate probability. We call this product E_i.

There are at most $\binom{n}{\delta_i}$ ways of choosing distinct bins for each of the δ_i children of v_i. Also, since there are at most n balls in the system at any point in time, the number of ways to choose distinct balls for the δ_i edges incident on v_i is also at most $\binom{n}{\delta_i}$. (Note that the n balls in the system may be different for each v_i; however, there are still at most $\binom{n}{\delta_i}$ possibilities for the ball choices for any vertex.) There are $\delta_i!$ ways of pairing the balls and the bins, and the probability that a chosen ball chooses bin r_i and a specific one of δ_i bins chosen above is $2/n^2$. Thus,

$$E_i \leq \binom{n}{\delta_i}\binom{n}{\delta_i}\delta_i!\left(\frac{2}{n^2}\right)^{\delta_i} \leq (2e)^{\delta_i}/\delta_i!. \qquad (5)$$

Let m be the number of internal nodes v_i in the pruned witness tree such that $\delta_i = \ell$. Using the bound in Equation 5 for only these m nodes, the number of ways of choosing the bins and balls for the nodes and edges respectively of the pruned witness tree weighted by the probability that these choices occurred is at most $n \cdot ((2e)^\ell/\ell!)^m$.

Ways of choosing the pruning balls in C. Using Lemma 8, we know that there are $\lceil p/2 \rceil$ distinct pruning balls in C. The number of ways of choosing the balls in C is at most $n^{\lceil p/2 \rceil}$, since at any time step there are at most n balls in the system to choose from. Note that a pruning ball has both its bin choices in the pruned witness tree. Therefore, the probability that a given ball is a pruning ball is at most

$$\binom{\ell^2 2^\ell}{2} \frac{2}{n^2} \leq \ell^4 2^{2\ell}/n^2.$$

Thus the number of choices for the $\lceil p/2 \rceil$ pruning balls in C weighted by the probability that these pruning balls occurred is at most

$$n^{\lceil p/2 \rceil} (\ell^4 2^{2\ell}/n^2)^{\lceil p/2 \rceil} \leq (\ell^4 2^{2\ell}/n)^{\lceil p/2 \rceil}.$$

Putting it all together. The probability at time t that there exists a pruned witness tree with p pruning edges, and m internal nodes with $\ell = \log\log n$ children each, is at most

$$\begin{aligned} \ell^{2p} 2^{\ell p} \cdot n \cdot ((2e)^\ell/\ell!)^m \cdot (\ell^4 2^{2\ell}/n)^{\lceil p/2 \rceil} &\leq n \cdot ((2e)^\ell/\ell!)^m \cdot (\ell^8 2^{4\ell}/n)^{\lceil p/2 \rceil} \\ &\leq n \cdot (2e^2/\log\log n)^{m \log\log n} \cdot (\log\log^8 n \log^4 n/n)^{\lceil p/2 \rceil}. \quad (6) \end{aligned}$$

Observe that either the number the pruning edges, p, equals ℓ or the number of internal nodes with ℓ children, m, is at least $2^{\ell-2} = \log n/4$. Thus, in either case, the bound in Equation 6 is $1/n^{\Omega(\log\log n)}$. Furthermore, since there are at most $\ell^2 2^\ell$ values for p, the total probability of a pruned witness tree is at most $\ell^2 2^\ell \cdot 1/n^{\Omega(\log\log n)}$ which is $1/n^{\Omega(\log\log n)}$. This completes the proof of the theorem. ∎

A similar approach can be used to show that the maximum load of $Q_d(\vec{v}, \vec{w})$ is $O(\log\log n/\log d)$, with high probability, for arbitrary values of d. The witness tree method can be used to analyze several complex problems that are not easily amenable to layered induction or fluid limit models. The analysis presented above of the sequential balls-and-bins problem with adversarial insertions and deletions is a good example of such a problem. However, due to their enumerative nature, it is difficult (though often possible) to obtain the best constants using witness tree arguments. For instance, the layered induction technique can be used to provide a tighter high-probability bound of $\log\log n/\log d + O(1)$ on the maximum load of $Q_d(\vec{v}, \vec{w})$, even though the analysis holds only when deletions are performed by removing a random ball currently in the system [ABKU99].

3.1.1 Extensions

The basic sequential balls-and-bins problem and the multiple-choice approach has been extended in several natural ways. We review two such extensions that are insightful and perhaps counter-intuitive.

Czumaj and Stemann [CS97] consider the multiple-choice approach with a small twist. Suppose you throw n balls into n bins sequentially, and each ball chooses d bins independently, uniformly and at random. Now, suppose that when a ball is thrown into one of its d chosen bins you are allowed to reallocate the balls in the d chosen bins so that the loads in these bins are as evenly balanced as possible, i.e., their loads differ at most by 1 after the rebalancing. Does this rebalancing decrease the maximum load? If so, by how much? Czumaj and Stemann show that even though the maximum load of the rebalancing algorithm is smaller than that of the original multiple-choice algorithm, the difference is no more than an additive constant! In particular, the maximum load of the rebalancing algorithm is also $\log \log n / \log d + \Omega(1)$, with high probability. Thus, rebalancing produces no significant additional benefit.

Vöcking [Vöc99] considers a variation of the multiple-choice method where each ball makes d independent but *nonuniform* choices. In particular, the bins are divided into d groups with n/d bins each, and each ball makes its ith choice uniformly from the bins in the ith group, for $1 \leq i \leq d$. As before, the ball is placed in a bin with the smallest number of balls. (If there are several bins with the smallest number of balls, we choose one of them randomly.) Does this make any difference to the minimum load? One can show that the maximum load is still $\Theta(\log \log n / \log d)$, with high probability, using a witness tree argument that is similar to the proof of Theorem 7.

Now, Vöcking considers an additional twist. Suppose the balls choose bins independently in the nonuniform manner described above, and in addition, we introduce the following tie-breaking rule called "always-go-left". The always-go-left rule states that a ball must be placed in the bin with the minimum load of its d choices, and if there are several bins with the smallest load it must be placed in the *leftmost* of these bins. Now, what happens to the maximum load? At first glance, it may appear that the tie-break rule should not make a big difference, and it should if anything increase the load. But, surprisingly, the combination of the nonuniform choices and always-go-left rule actually *decreases* the maximum load to $\frac{\ln \ln n}{d \cdot \ln \phi_d} + O(1)$ with high probability, where here ϕ_d corresponds to the exponent of growth for a generalized Fibonacci sequence. (For reference,

$\phi_2 = 1.61 < \phi_3 = 1.83 < \phi_4 = 1.92\ldots < 2$.) It should be pointed out that if the balls make independent uniform choices, any tie-breaking rule including always-go-left, does not make a difference; i.e., the maximum load is still $\Theta(\log\log n/\log d)$ with high probability [ABKU99].

In view of these results, it is natural to ask if there is a method of choosing the d bins and a rule for allocating each ball to one of its chosen bins that provides an even smaller maximum load. Vöcking shows that no significant decrease in the maximum load is possible. In particular, he shows that if each ball chooses its d bins according to an *arbitrary* distribution on $[n]^d$ and the ball is placed in one of its chosen bins using an *arbitrary* rule, the maximum load is $\frac{\ln\ln n}{d\cdot\ln\phi_d} - O(1)$, with high probability.

3.2 The parallel balls-and-bins problem

In this section, we illustrate how to use the witness tree approach to analyze collision protocols for the parallel balls-and-bins problem. The parallel version of the balls-and-bins problem was first studied by Adler et al. [ACMR95]. Unlike the sequential case where balls are thrown into bins one after another, we consider the situation when n balls choose each d bins independently and uniformly at random, in parallel. The balls choose their final destinations by performing ρ rounds of communication. Each round consists of two stages. In the first stage each ball can send messages, in parallel, to any of its d chosen bins. In the second stage, each bin can respond by sending messages, in parallel, to any of the balls from which it received a message.

A natural class of protocols for the parallel balls-and-bins problem is the class of *collision protocols*. Collision protocols have been used widely for contention resolution in message routing [KLM96, GMR94, MPR98, CMS95, MSS96]. Such protocols were first used for the parallel balls-and-bins problem in [ACMR95]. The algorithm we present here is due to Stemann [Ste96] and can be described as follows. We set a threshold τ such that each bin can accept no more than a total of τ balls during the entire process — i.e., τ is the maximum load of any bin. The collision protocol proceeds as follows. (For simplicity, we study the case when $d = 2$.)

- In parallel each ball picks two bins independently and uniformly at random.
- While there is a ball that has not been allocated, do the following.
  - In parallel, each unallocated ball sends a request to its two chosen bins.

- In parallel, each bin that would have load at most τ if it accepted all balls requesting the bin in that round sends an acknowledgment to all the requesting balls. (A bin that would achieve a load greater than τ does nothing.)
- Each ball that receives an acknowledgment is allocated to the respective bin (ties are broken by randomly selecting one of the bins that sent an acknowledgment).

We illustrate how we can analyze this simple protocol using the witness tree method.

Theorem 9 *For any $1 \leq \rho \leq \log\log n$, the collision protocol described above with threshold $\tau = O(\sqrt[\rho]{\frac{\log n}{\log\log n}})$ finishes after ρ rounds with probability at least $1 - \frac{1}{n^{\Omega(1)}}$.*

Proof Sketch: As in the proof of Theorem 7, we start by building a witness tree. Suppose that there are unallocated balls at the end of round ρ. This implies that some bin r received more than τ requests in the ρ^{th} round. The root of the tree corresponds to bin r. Let $b_1, \ldots, b_{\tau+1}$ be the balls that sent a request to r in round ρ. (We assume that the balls b_i are ordered in the ascending order of their IDs.) For each $1 \leq i \leq \tau+1$, both of the bin choices of b_i received at least $\tau + 1$ requests in round $\rho - 1$. Let r_i be the other bin choice associated with ball b_i (one of the choices is bin r). The root r has $\tau + 1$ children, one corresponding to each bin r_i. Now, we use that fact that each bin r_i had $\tau + 1$ requests in round $\rho - 1$ to recursively grow a depth-$(\rho - 1)$ tree rooted at each r_i. Thus, we have constructed a complete $\tau + 1$-ary tree of depth ρ as our witness tree.

The next step is to enumerate all possible witness trees and prove that the probability that some witness tree occurs is at most $1/n^{\Omega(1)}$. It is instructive to first consider the situation where all the nodes in the witness tree represent distinct bins. In this situation, the enumeration proceeds as follows. Let m be the number of nodes in the witness tree.

- The number of ways of choosing a distinct bin for each node of the tree is at most $n \cdot (n-1) \cdots (n-m+1) = n^{\underline{m}} \leq n^m$.
- The number of ways of choosing distinct balls for each of the $\tau + 1$ edges of an internal node of the tree is $\binom{n}{\tau+1} \leq n^{\tau+1}/(\tau+1)!$. Note that once the balls are chosen, they are paired up in the ascending order (from left to right) of their IDs, i.e., there is only one way of pairing them up with the bins. Since at least $\frac{m-1}{\tau+1}$ nodes of the tree

are internal nodes, the total number of ways of labeling each edge of the tree with balls is $n^{m-1}/((\tau+1)!)^{\frac{m-1}{\tau+1}}$.

- Once the entire tree is labeled with balls and bins, the probability that the labeled tree occurs is at most $\left(\frac{2}{n^2}\right)^{m-1}$, since there are $m-1$ edges and since each edge corresponds to the event that a particular ball chooses two particular bins.

Putting it all together, the probability of occurrence of a witness tree (provided each node represents a distinct bin) is at most

$$n^m \cdot \frac{n^{m-1}}{((\tau+1)!)^{\frac{m-1}{\tau+1}}} \cdot \left(\frac{2}{n^2}\right)^{m-1} \leq \frac{n \cdot 2^{m-1}}{((\tau+1)!)^{\frac{m-1}{\tau+1}}}.$$

Observing that $m = \frac{(\tau+1)^{\rho+1}-1}{\tau}$, and setting $\tau = c\sqrt[\rho]{\frac{\log n}{\log\log n}}$, for a suitably large constant c, the above bound is at most $1/n^{\Omega(1)}$.

Unfortunately, the above simplified analysis does not always hold since the bins in the witness tree may not be distinct. We resolve this problem in a manner similar to the proof of Theorem 7. We prune the witness tree so that the resulting tree contains only bins that are distinct (and, hence, the balls are distinct also). If there are are too "few" pruning edges, then there exists a "large" subtree where the bins are distinct. In this case, we perform an analysis similar to the one outlined above to derive the bound. Otherwise, if there are a "large" number of pruning edges, both the random choices of the balls corresponding to the pruning edges fall within the set of bins in the witness tree. Since the size of the witness tree is small compared to n, i.e., the bins in the witness tree are a small subset of all the bins, it is unlikely that there are a "large" number of pruning edges. Thus, the probability that a witness tree exists is small in either case. ■

A consequence of Theorem 9 is that the collision protocol achieves a maximum load of $O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ in two rounds, and a maximum load of a constant in $O(\log\log n)$ rounds, with high probability.

The basic parallel balls-and-bins problem can be extended in various natural ways. We now look at some of these extensions.

3.2.1 Weighted Balls

Berenbrink et al. [BMS97] generalize the parallel balls-and-bins problem to the situation where m balls are thrown into n bins in parallel, and each ball has a *weight* associated with it. The load of any bin in the weighted version of the problem is the sum of the weights of the balls allocated to that bin. They show that a more sophisticated collision protocol achieves a maximum load of $O\left(\frac{m \cdot w_{avg}}{n} + w_{max}\right)$ in $O\left(\frac{\log\log n}{\log\left(\frac{m \cdot w_{avg}}{n \cdot w_{max}}+1\right)}\right)$ rounds, with high probability, where w_{avg} and w_{max} are the average and maximum weight of a ball respectively. Note that by the pigeonhole principle, some bin receives a load of at least $\frac{m \cdot w_{avg}}{n}$, and the bin with the maximum-weighted ball has load at least w_{max}. Thus, the protocol of Berenbrink et al. achieves the optimal maximum load to within constant factors, with high probability.

3.2.2 Dynamic Arrivals

Adler et al. [ABS98] consider a natural generalization of the parallel balls-and-bins problem. In their model, m balls arrive in *each round* and must be allocated in parallel to n bins. (This model should be distinguished from the dynamic but sequential arrival of balls considered in Sections 4.2 and 3.1.) Each bin has a first-in, first-out (FIFO) queue where the balls wait to be "served". In each round, each bin *serves* and *removes* the ball at the head of its queue. The goal is to allocate balls in a manner that minimizes the number of rounds that a ball spends waiting to be served by a bin. Adler at al. study a natural protocol for allocating the balls. Each arriving ball chooses two bins independently and randomly, and adds itself to the queues of *both* bins. When a ball is served and removed by one of its queues, the ball is also deleted from the other queue.

Theorem 10 *For the protocol outlined above, any given ball waits at most $O(\log\log n)$ rounds before being served, with high probability, provided $m \leq \frac{n}{6e}$.*

Proof Sketch: The proof of the result uses the classical witness tree method except that we view the nodes of the tree as representing balls (instead of bins). Suppose that a ball b arrives at time T and waits more than τ rounds before being served. A depth-τ witness tree can be constructed as follows. The root of the witness tree is associated with ball b. Consider the two bins r' and r'' chosen by ball b. Since b is not served by either bin at time $T+\tau$, there must exist two balls b' and b'' that are served by r' and r'' respectively at time $T+\tau$. We make balls b' and b'' the two children of ball b in the witness tree. Since each queue uses the FIFO protocol, b' and b'' arrived at time T or earlier. Hence balls b' and b'' waited more than $\tau-1$ rounds before being served. Thus, we can recursively grow depth-$(\tau-1)$ trees rooted at b' and b''.

We now enumerate and bound the probability of occurrence of a depth-τ witness tree, for $\tau = O(\log \log n)$. As always, one has to deal with the fact that a ball can appear multiple times in the tree. But, perhaps the greater technical difficulty is that, unlike our other examples of witness tree proofs, we have no a priori bound on the number of balls present in the system at a given time. Therefore, we need to explicitly characterize the balls that can appear in the witness tree. The reader is referred to [ABS98] for further details. ■

3.2.3 Load Balancing in Parallel Environments

Much of the recent interest in the balls-and-bins problem is due to its applicability to scheduling tasks in a parallel or distributed environment. The examples we have seen so far apply to the so-called *client-server* paradigm where the clients generate tasks (i.e., balls), sequentially or in parallel, and the tasks must be allocated to servers (i.e., bins) so as to balance the load on each server.

In this section, we explore a somewhat different paradigm that is relevant to load balancing in a parallel computer. In a parallel computer, unlike the client-server model, the processors play a *dual role* in that they both generate and execute tasks. We seek distributed algorithms that ensure that the maximum load of any processor remains "small", i.e., we would like the tasks to be as evenly distributed among the processors as possible. In addition, we would like to avoid excessive communication between processors and would like to execute the tasks in the processors where they are generated as much as possible. This additional locality constraint is an important distinguishing feature that is absent in the client-server model.

The load balancing problem can be classified according to the nature of the tasks themselves. The problem is more complex if the tasks have explicit dependencies; for instance, the tasks may represent a multi-threaded computation modeled as a directed acyclic graph [BL94, ABP98], or the tasks may be generated by a backtrack search or branch-and-bound algorithm [KZ93]. The situation where the tasks are independent is somewhat simpler and several models for generating and consuming independent tasks are considered in the literature [RSAU91, BFM98, BFS99]. In the *random load model* each processor in each step generates a task with a fixed probability λ and consumes a task with a fixed probability μ, where $0 \leq \lambda < \mu \leq 1$. Whereas in the *adversarial load model*, the load of each processor at each time can be modified arbitrarily by an adversary, provided the net change in load is at most a given parameter δ.

The literature in this area can also be classified by the nature of the proposed algorithm. In a *work sharing* algorithm, a "heavily-loaded" processor seeks to donate some of its excess tasks to a suitable processor [BFM98, BFS99]. A key issue in work sharing is how a heavily-loaded processor finds one or more "lightly-loaded" processors to which to donate its excess tasks. The matching of the heavily-loaded processors to lightly-loaded processors must be performed efficiently and in a distributed manner. In a *work stealing* algorithm, a "lightly-loaded" processor "steals" tasks from a suitable processor (e.g., [ELZ86b, FMM91, FM87, HZJ94, Mit98, BL94, ABP98]). A particular example of this approach is idle-initiated work stealing where a processor that becomes idle seeks to obtain tasks from nonidle processors.

Randomized algorithms have proven to be a critical tool in this matching process since the earliest investigations in this area. More recently, there has been an effort to use collision algorithms and related ideas to perform this matching [BFM98, BFS99]. Berenbrink et al. [BFS99] show how to maintain a load of $O(\log\log n)$ on an n-processor parallel machine, with high probability, in the random load model. Collision protocols are used to construct a tree rooted at each heavily-loaded processor, and each such processor communicates down its tree to search for an "unattached" lightly-loaded processor. Note that one can easily achieve the same bound on the load by migrating each task as soon as it is generated using a variant of the algorithm described in Section 3.2.2. However, such an algorithm would entail a large amount of communication. The primary contribution of Berenbrink et al. is that their work sharing algorithm ensures that processors send tasks only if they are heavily-loaded, reducing the total communication performed by a factor of $\Theta(\log\log n)$ with high probability.

3.3 A Lower Bound Using Witness Trees

We have seen how to use witness trees for proving upper bounds on the maximum load of a bin. However, witness trees are useful in proving lower bounds as well. In the upper bound proofs of Theorem 7 and 9 we observed that if there is a "heavily-loaded" bin there exists a witness tree whose node-degree and height are "large". The key idea in deriving lower bounds using witness trees is that, in some circumstances, the converse is also true: if a witness tree with "large" degree and height occurs then some bin is expected to receive a "large" number of balls. We illustrate this technique by proving a lower bound on the maximum load for the parallel balls-and-bins problem.

The collision protocol that we outlined in Section 3.2 is *nonadaptive* in that the possible destinations for the balls are chosen *before* any communication takes place. Furthermore, the protocol is *symmetric* in that all balls and bins perform the same algorithm and the bins are chosen independently and at random. A natural question to ask is if there exists a nonadaptive and symmetric algorithm that achieves a smaller expected maximum load than the collision protocol outlined in Section 3.2. Adler et al. [ACMR95] show that the expected maximum load of a bin is $\Omega\left(\sqrt[\rho]{\frac{\log n}{\log\log n}}\right)$ for any protocol that performs ρ rounds of communication, provided that ρ is a constant and the protocol belongs to a natural *subclass* of nonadaptive and symmetric protocols. (Most known nonadaptive and symmetric protocols belong to this subclass. We describe the restrictions that define this subclass in Theorem 11 below.) This lower bound result was later extended to all values of ρ by Berenbrink et al. [BMS97], which we state below.

Theorem 11 *The expected maximum load of a bin is* $\Omega\left(\sqrt[\rho]{\frac{\log n}{\log\log n}}\right)$ *for any protocol that performs* ρ *rounds of communication,* $1 \leq \rho \leq \log\log n$*, provided that the protocol satisfies the following conditions:*

- *the protocol is nonadaptive and symmetric,*
- *removing a set of balls before the protocol begins cannot increase the expected maximum load achieved by the protocol, and*
- *if a ball cannot distinguish between its two bin choices after* ρ *rounds of communication, the protocol allocates the ball randomly with probability* $1/2$ *to either of its choices.*

Proof Sketch: Let a (T, r)-ary tree be a tree of depth r such that the root has T children, and every other internal node has $T-1$ children. The first step in our proof is to show that with constant probability there exists a $(\tau+1, \rho+1)$-ary witness tree W such that all the nodes of the tree represent distinct bins, for some $\tau = \Theta\left(\sqrt[\rho]{\frac{\log n}{\log\log n}}\right)$. It is easy to show that the expected number of such witness trees is $\Omega(n)$ using an enumeration very similar to the one in the proof of Theorem 9. (The primary difference is that we now seek a lower bound on the expectation.) The number of ways of labeling a $(\tau+1, \rho+1)$-ary tree with distinct balls and bins is at least $n^{\underline{m}} \cdot n^{\underline{m-1}} \geq (n-m)^{2m-1}$, where m is the number of nodes in the tree. The probability of occurrence of each labeled tree is $\left(\frac{2}{n^2}\right)^{m-1}$. Thus the expected number of $(\tau+1, \rho+1)$-ary witness trees is (using linearity of expectation) at least

$$(n-m)^{2m-1} \cdot \left(\frac{2}{n^2}\right)^{m-1} = \Omega(n),$$

since $m = (\tau+1)\frac{\tau^{\rho+1}-1}{\tau-1} = o(n)$. The fact that the expected number of witness trees is $\Omega(n)$ does not immediately imply that there exists such a witness tree with constant probability. We need to show that the variance of the number of witness trees is "small". The lower and upper bounds on the expectation and variance respectively, in conjunction with Chebyshev's inequality, yields the fact that a $(\tau+1, \rho+1)$-ary witness tree exists with constant probability. We refer to Czumaj et al [CMS95] and Stemann [Ste96] for the details of the existence proof.

Let W be the $(\tau+1, \rho+1)$-ary witness tree with distinct bins that we have shown exists with constant probability. The initial random choices made by the balls can be represented by an *access graph* with n nodes and n edges. Each of the n nodes represent a distinct bin. Each of the n edges represents the pair of bin choices made by a distinct ball. Note that the witness tree W is a subgraph of the access graph. In addition to assuming that the protocol is nonadaptive and symmetric, we now use our second additional assumption that *removing a set of balls, i.e., deleting some edges of the access graph, cannot increase the expected maximum load achieved by the protocol*. More precisely, given two access graphs G and G' such that the edges of G are a superset of the edges of G', the expected[5] maximum load achieved by the protocol on G' is at most the expected maximum load achieved on G. This assumption is intuitively reasonable and holds for most natural algorithms considered in the literature, including the collision protocol outlined in Section 3.2. We utilize this assumption to remove all edges, i.e., balls, in the access graph that do not belong to W.

Henceforth, we consider only tree W and any lower bound we derive on the expected maximum load of the protocol on W is a lower bound on the expected maximum load for the original access graph. Let b be the ball that corresponds to an edge incident on root r of W. We argue that ball b has probability $\frac{1}{2}$ of being allocated to r.

The key to the argument is quantifying the amount of knowledge b can acquire by performing ρ rounds of communication. Initially, ball b knows nothing about the choices made by any other ball. After the first round, assuming in the worst case that the two bins chosen by b convey all their information to b, b knows about the choices of all the balls that chose one of its bins, i.e., b knows about the edges in the neighborhood $N(b)$, where $N(b)$ is the set of all edges of W incident to an endpoint of b. (We define $N(S)$, where S is a set of edges, to be $\cup_{b \in S} N(b)$.) Inductively, after $\rho > 1$

[5]The expectation is taken over different runs of the protocol on a *given* access graph, i.e., the bins chosen by the balls are fixed.

rounds, ball b knows about the ρ-neighborhood $N_\rho(b)$ which is recursively defined to equal $N(N_{\rho-1}(b))$.

Let ball b correspond to edge (r, r'), where r is the root of W and r' is its child. Removing edge (r, r') from the neighborhood set $N_\rho(b)$ splits it into two connected components, $N_{\rho,r}(b)$ that contains r and $N_{\rho,r'}(b)$ that contains r'. Since W is a $(\tau+1, \rho+1)$-ary tree, both $N_{\rho,r}(b)$ and $N_{\rho,r'}(b)$ are both complete τ-ary trees of depth ρ, i.e., they are *identical.* Since the neighborhood set containing *either* bin is identical, there is no reason for ball b to choose one bin over another. We now use our third additional assumption that in this case the *protocol must choose a bin randomly with probability* $\frac{1}{2}$. Thus ball b chooses root r with probability $\frac{1}{2}$. Since this holds for each of the $\tau+1$ children of r, the expected load of r is at least $\frac{\tau+1}{2} = \Omega\left(\sqrt[\rho]{\frac{\log n}{\log\log n}}\right)$.

Since a $(\tau+1, \rho+1)$-ary tree W occurs with constant probability, it follows that the expected maximum load is $\Omega(\sqrt[\rho]{\frac{\log n}{\log\log n}})$ for any nonadaptive symmetric protocol that obeys the two additional assumptions stated in the theorem. ■

3.4 Randomized Protocols for Circuit Routing

Much of the recent interest in the balls-and-bins problem derives from the straightforward analogy of scheduling tasks (i.e., balls) on multiple servers or processors (i.e., bins). Can the multiple-choice approach be used effectively in problem domains other than task scheduling? In this section, we show how we can use a variant of the multiple-choice method to perform low-congestion circuit routing in multistage interconnection networks. The results presented in this section are based on Cole et al. [CMM$^+$98] and represent some of the more sophisticated applications of witness tree arguments. It is worth pointing out that these results do not appear to be amenable to either layered induction or fluid limit models.

Several modern high-speed multimedia switches and ATMs utilize (virtual) circuit-switching to route communication requests [RCG94, TY97]. In a circuit-switched network, requests arrive at the input nodes of the network and require a path to some output node of the network. A *circuit-routing algorithm* allocates a path through the network for each request. When the request is completed, the path allocated to it is freed up. The goal is to devise routing algorithms that minimize congestion, where *congestion* is the maximum number of paths that must be simultaneously supported by a link of the network.

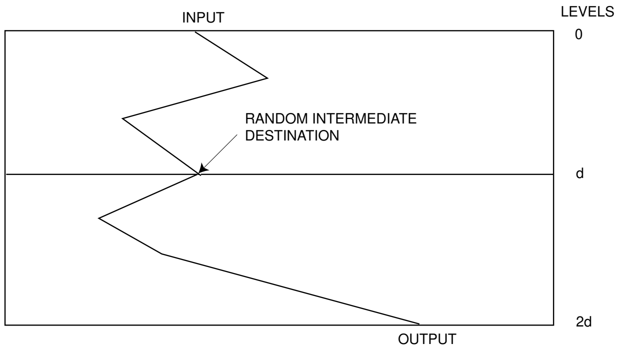

Figure 2: Valiant's paradigm

A canonical circuit routing problem often studied in the literature is the permutation routing problem. In a *permutation routing problem* at most one request originates at each input of the network and at most one request is destined for each output of the network. Furthermore, we distinguish between two kinds of permutation routing problems: static and dynamic. In a *static problem*, all the requests that constitute a permutation routing problem are present at time 0, before the routing begins. The routing algorithm constructs paths for all of the requests in a "batch" mode. All of the requests in a batch complete before routing of the next batch of requests begins. In contrast, in a *dynamic problem*, requests arrive and leave over time, according to a sequence constructed by an oblivious adversary. The routing algorithm routes a path for each arriving request in an on-line fashion with no knowledge of future request arrivals. We assume that at any time, the requests being routed form a partial permutation; that is, each input and output node correspond to at most one routed request.

The results in this section apply to variants of a popular type of multistage interconnection network called the *butterfly network* (See [Lei92] For a description of its structure.). An n-input butterfly B_n has $n(\log n + 1)$ nodes arranged in $\log n + 1$ *levels* of n nodes each.

Furthermore, there is a unique path of length $\log n$ in B_n from each input node to each output node. A *two-fold butterfly* BB_n consists of two copies of B_n placed one after the other such that each output node in the first copy is identified with the corresponding input node of the second copy. The inputs of BB_n are at level 0 while the outputs are at level $2d$, where $d = \log n$. (See Figure 2).

An early example of the use of randomized algorithms for communication problems is the work of Valiant [Val82, VB81]. Valiant showed that any permutation routing problem can be transformed into two random problems by first routing a path for each request to a random intermediate destination, chosen independently and uniformly, and then on to its true destination (See Figure 2). This routing technique known as *Valiant's paradigm* is analogous to the classical balls-and-bins problem where each request (i.e., ball) chooses a random intermediate destination (i.e., bin). It follows from this analogy that the congestion achieved by Valiant's paradigm corresponds to the maximum load of the classical balls-and-bins problem, and is $\Theta(\log n/\log\log n)$, with high probability.

3.4.1 Valiant's Paradigm with Two Random Choices

A question that begs asking is if the two-choice method can be incorporated into Valiant's paradigm to significantly reduce the congestion, just as two random choices can be used to significantly reduce the maximum load of a bin in the balls-and-bins problem. The simplest way to incorporate the two-choice approach into Valiant's algorithm is to let each request choose *two* random intermediate destinations (instead of one), and choose the path that has the smaller congestion. (The congestion of a path is the maximum congestion of all its edges.) But this simple approach fails to decrease the congestion significantly. The problem lies in the fact that even though a request can choose any of the n intermediate nodes in level d to be on its path, it has very few nodes that it can choose in levels that are close to its input or output. Therefore, it is quite likely that there exists a set of $m = \Theta(\log n/\log\log n)$ requests such that all the paths chosen by these requests intersect at some node at level $\log m$ of BB_n. Thus, any allocation of requests to paths causes congestion of at least $m = \Omega(\log n/\log\log n)$.

The key to applying the two-choice approach to circuit routing is to avoid creating hot spots of congestion near the inputs and outputs of the network. To achieve this we select two random paths for each request as follows. The nodes on levels $0,\ldots,d/2-1$ and $d+d/2+1,\ldots,2d$ are flipped randomly, where $d=\log n$. In particular, each input and output node maps the *first path* p of a request to its straight edge and its *second path* p' to its cross edge with probability $\frac{1}{2}$,and with probability $\frac{1}{2}$ the order is reversed. Similarly, each node on levels $1,\ldots,d/2-1$ and $d+d/2+1,\ldots,2d-1$ with probability $\frac{1}{2}$ connects its input straight edge with its output straight edge and its input cross edge with its output cross edge, and with probability $\frac{1}{2}$ the connections are reversed. (See Figure 3.) Note that these random choices completely determine the two paths p and p' of each request, because there is exactly

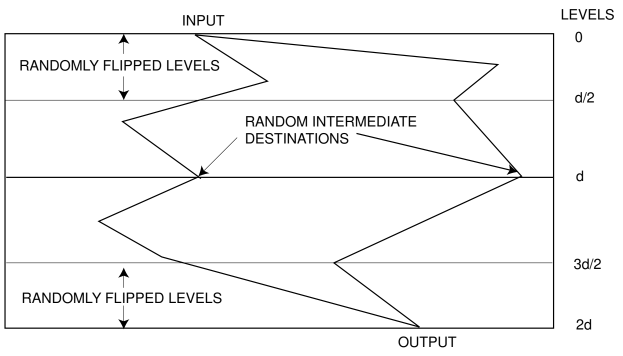

Figure 3: Valiant's paradigm with two random destinations

one path connecting a node on level $d/2$ with a node on level $d + d/2$ in the BB_n network. For a path p, the other path p' connecting the same input and output nodes is called the *buddy* of p. The random switching ensures that any edge on the levels $1, \ldots, d/2$ and $d + d/2 + 1, \ldots, 2d$ is traversed by at most one of the randomly-generated paths.[6] However, each edge that originates at a node in an interior level, i.e., levels $d/2 + 1, \ldots, d + d/2$, is potentially traversed by several of these paths. Note that the random paths chosen for distinct requests are no longer independent, because the paths of the two requests may share one or more randomly-flipped switches. This is a key technical difficulty that complicates the witness tree analyses in this section.

3.4.2 Permutation Routing

Recall that in a dynamic permutation routing problem, the requests arrive and leave according to a sequence constructed by an oblivious adversary. When a request r arrives, our routing algorithm chooses two random paths as described in Section 3.4.1, evaluates the congestion of the two paths, and allocates r to the path with smaller congestion.

Theorem 12 *The routing algorithm described above finds paths for every request of a dynamic permutation routing problem in network BB_n such that the congestion at any given time t is $O(\log \log n)$, with high probability.*

[6] The idea of using randomly-flipped switches to control congestion was first used by Ranade [Ran87] for packet routing and was later adapted to circuit-switching algorithms by Maggs and Sitaraman [MS92].

Proof Sketch: The overall structure of the proof is similar to that of Theorem 7, though the details are much more complicated. First, we fix the settings of the randomly-flipped switches. This determines two choices of paths for each request. Assume that there is an edge e with congestion larger than $4c$ at some time t, where $c = \lceil \log \log n \rceil$. Let p denote the last path mapped to edge e on or before time t. When p was mapped to e there were already $4c$ other paths present at this edge. Let $p_1, \ldots, p_{4c}$ denote these paths such that p_i was mapped to e at time step t_i with $t_i < t_{i+1}$. The root of the tree is the request corresponding to p and the requests corresponding to $p_1, \ldots, p_{4c}$ are its children. Now we consider the buddies $p'_1, \ldots, p'_{4c}$ of these paths. Path p'_i traverses an edge with congestion at least $i-1$ at time step t_i, because the congestion of p_i is not larger than the congestion of p'_i at time i, and when p_i was mapped to e there were already $i-1$ other paths present at this edge. As a consequence, we can construct a tree by applying the argument above recursively to $p'_2, \ldots, p'_{4c}$. The tree constructed in this fashion is the witness tree.

The technical challenge is performing the next step of enumerating and bounding the probability that a witness tree occurs. Recollect that the paths are not chosen independently for each request, since paths belonging to distinct requests may share one or more randomly-flipped switches in the first or last $d/2$ levels. Therefore, when the witness tree is pruned, besides ensuring that the requests in the pruned tree are distinct, it is also necessary to ensure that the paths of the requests in the pruned tree share only a "limited" number of randomly-flipped switches, i.e., the paths in the pruned tree represent "almost" independent random choices. The bound follows by enumerating the paths in the witness trees and their respective probabilities of occurrence. ■

The results in this section show that the two-choice method can be used to significantly reduce the congestion of Valiant's algorithm for dynamic permutation routing. The two-choice approach can also be adapted to route any static permutation routing problem on BB_n with congestion $O(\log \log n / \log \log \log n)$, with high probability [CMM$^+$98].

4 A Differential Equations Approach: Fluid Limits

We now describe a third technique that has proven useful for analyzing randomized load balancing schemes based on the two-choice paradigm. This technique was developed in parallel in the computer science and queueing theory communities. The approach relies on determining the behavior of the system as its size grows to infinity. For example, in the balls-and-bins model, the size of the system naturally corresponds to the number of bins, so we consider what happens as the number of bins grows towards infinity. Often we can describe the behavior naturally as a differential equation or a family of differential equations.

Once we have a family of differential equations that describe the process, we can try to solve it to obtain a closed form solution. Then, in many cases, we can relate this solution to the behavior of a system of finite size using a concentration result. That is, we may show that the behavior of a system of finite size is close to the behavior given by the solutions of the differential equations, with high probability.

Sometimes we cannot obtain a closed form from the differential equations. In such cases, however, we can often solve the differential equations using numerical methods.

4.1 Balls and bins

4.1.1 Empty bins

To introduce the approach, let us consider the sequential multiple-choice balls-and-bins problem, where $m = cn$ balls are thrown into n bins with each ball having two bin choices. We first ask what fraction of the bins remain empty. This question was first considered by Hajek [Haj88].

The problem can be solved by developing a Markov chain with a simple state that describes the balls-and-bins process. We first establish a concept of time. Let $Y(T)$ be the number of nonempty bins after T balls have been thrown. Then $\{Y(i)\}$, $0 \leq i \leq m$, is a Markov chain, since the choices for each ball are independent of the state of the system. Moreover

$$\mathbf{E}[Y(T+1) - Y(T)] = 1 - \left(\frac{Y(T)}{n}\right)^d, \tag{7}$$

since the probability that a ball finds all nonempty bins among its d choices is $(Y(T)/n)^d$.

The notation becomes somewhat more convenient if we scale by a factor of n. If t is the time at which exactly nt balls have been thrown, and $y(t)$ is the fraction of nonempty bins, then equation (7) becomes

$$\frac{\mathbf{E}[y(t+1/n) - y(t)]}{1/n} = 1 - (y(t))^d. \tag{8}$$

We claim the random process described by equation (8) is well approximated by the trajectory of the differential equation

$$\frac{dy}{dt} = 1 - y^d. \tag{9}$$

This equation has been obtained from equation (8) by replacing the left hand side with the appropriate limiting value as n grows to infinity, dy/dt. That is, we think of each ball as being thrown during a small interval of time dt of duration $1/n$. In equation (8) we replace the expected change in y over this interval by dy, with the intuition that the behavior of the system tends to follow its expectation over each step.

This claim, that the system behaves close to what we might expect simply by looking at the expected change at each step, follows from a theorem similar to the law of large numbers for special types of Markov chains. The important feature is that the differential equation obtained is independent of the number of bins n; such Markov chains are called *density dependent*, as their behavior depends essentially on the density (in this case y) of objects in the state rather than the total number of objects. Here, the objects are the nonempty bins. For such a system there are bounds similar to Azuma's inequality for martingales [MR95]. Indeed, the derivation of the theorem is based on a suitable martingale. For example, for the balls-and-bins problem above, if y^* is the solution of the differential equation, then

$$\mathbf{Pr}\left(\sup_{0 \le t \le T} |y(t) - y^*(t)| \ge \epsilon\right) \le C_1 e^{-nC_2(\epsilon)}$$

for constants C_1 and C_2 that may depend on T. This approach is also often referred to as the *fluid limit* approach, since the discrete process is replaced by an often simpler continuous process reminiscent of the behavior of physical fluid models.

These theorems apparently first appeared in the work of Kurtz [EK86, Kur70, Kur81], and were eventually applied to algorithmic problems related to random graphs [Haj88, KS81, KVV90] as well as to queueing problems [CH91]. Recently these techniques have resurged in the random graph community, initiated by the work of Wormald [Wor95]. The text by Shwartz and Weiss on large deviations provides a solid introduction into the entire area of large deviations, including Kurtz's work [SW95]. There are by now many examples of works that use large deviation bounds and differential equations for a variety of problems, including but in no way limited to [AM97, AH90, AFP98, KMPS95, LMS+97].

Given this framework, it is easy to find the limiting fraction of empty bins after $m = cn$ balls have been thrown, by solving the differential equation $\frac{dy}{dt} = 1 - y^d$ with the initial condition $y(0) = 0$ at time c. This can be done by using the nonrigorous high school trick of writing the equation as $\frac{dy}{1-y^d} = dt$ and integrating both sides.

Theorem 13 *Let c and d be fixed constants. Suppose cn balls are sequentially thrown into n bins, each ball being placed in the least full of d bins chosen independently and uniformly at random. Let Y_{cn} be the number of nonempty bins when the process terminates. Then $\lim_{n\to\infty} \mathbf{E}[\frac{Y_{cn}}{n}] = y_c$, where $y_c < 1$ satisfies*

$$c = \sum_{i=0}^{\infty} \frac{y_c^{id+1}}{(id+1)}.$$

Using this we may solve for y_c. (Closed form expressions exist for some values of d, but there does not seem to be a general way to write y_c as a function of c.)

We may actually use Kurtz's Theorem to obtain a concentration result.

Theorem 14 *In the notation of Theorem 13, $|\frac{Y_{cn}}{n} - y_c|$ is $O\left(\sqrt{\frac{\log n}{n}}\right)$ with high probability, where the constant depends on c.*

One can also obtain entirely similar bounds for Y_{cn} using more straightforward martingale arguments; however, the martingale approach does not immediately lead us to the value to which Y_{cn}/n converges. This is a standard limitation of the martingale approach: in contrast, the fluid limit model allows us to find the right limiting value.

4.1.2 Nonempty bins

The previous analysis can be extended to find the fraction of bins with load at least (or exactly) k for any constant k as $n \to \infty$. To establish the appropriate Markov chain, let $s_i(t)$ be the fraction of bins with load *at least* i at time t, where again at time t exactly nt balls have been thrown. Note that this chain deals with the tails of the loads, rather than the loads themselves. This proves more convenient, as we found in Section 2. The differential equations regarding the growth of the s_i (for $i \geq 1$) are easily determined [Mit96b, Mit99b]:

$$\begin{cases} \dfrac{ds_i}{dt} &= (s_{i-1}^d - s_i^d) \quad \text{for } i \geq 1\,; \\ s_0 &= 1. \end{cases} \tag{10}$$

The differential equations are easily interpreted as denoting that an increase in the number of bins with at least i balls occurs when the d choices of a ball about to be placed must all be bins with load at least $i-1$, but not all bins with load at least i.

The case of n balls and n bins corresponds to time 1. Interestingly, the differential equations reveal the double exponential decrease in the tails in this situation quite naturally:

$$\begin{aligned}
s_i(1) &= \int_{t=0}^{1} \left[(s_{i-1}(t))^d - (s_i(t))^d \right] dt \\
&\leq \int_{t=0}^{1} (s_{i-1}(t))^d \, dt \\
&\leq \int_{t=0}^{1} (s_{i-1}(1))^d \, dt \\
&= (s_{i-1}(1))^d.
\end{aligned}$$

Hence

$$s_i(1) \leq (s_1(1))^{d^{i-1}}.$$

The above argument shows that the tails $s_i(1)$ in the limiting case given by the differential equations decrease doubly exponentially; of course, since the results for finite n are tightly concentrated around these s_i, the implication is that the behavior also occurs in finite systems. The astute reader will notice that the steps taken to bound the integral expression for $s_i(1)$ above entirely mimics the original proof of Azar, Broder, Karlin, and Upfal. That is, we bound $s_i(1)$ based only on $s_{i-1}(1)$. Indeed, the differential equations provide an appealing natural intuition for their proof, and a similar approach can be used to mimic their lower bound argument as well. Although this intuition implies a maximum load of $\log \log n / \log d + \Theta(1)$ in the case of n balls and n bins, the general theory for density dependent chains does not seem to get one there immediately. An immediate problem is that Kurtz's theorem, as generally stated, requires a fixed number of dimensions. That is, we can only work with s_i for $i \leq K$ for some fixed constant K. Hence considering loads of up to $O(\log \log n)$ requires more than a simple application of the theorem. A further problem is that as the tail gets small, the probabilistic bounds are not strong enough. Hence one apparently needs an explicit argument, such as that given by Azar et al., to achieve such an explicit bound.

While the fluid limit approach does not obviate the need for detailed probabilistic arguments, it provides a remarkably useful tool. In particular, when applicable it generally provides natural intuition and remarkable accuracy in predicting the behavior of even moderate sized systems. (See, e.g., [Mit96b].) Moreover, it offers tremendous flexibility. Many variations on the balls-and-bins problem can be placed into the differential equations framework. We provide a short introduction to some of the more interesting ones.

4.2 Queueing theory models

We consider the fluid limit model of a natural queueing system that generalizes the static multiple-choice balls-and-bins problem. Suppose we think of the bins as FIFO (First-In, First-Out) servers and the balls as tasks that enter and leave after being processed. In this case, we assume that tasks arrive as a Poisson process of rate λn proportional to the number of servers n, with $\lambda < 1$. We also assume that tasks require an amount of service distributed exponentially with mean 1. This model, a generalization of the natural M/M/1 queueing model to many servers, has been widely studied in the case where incoming customers are placed at the server with the shortest queue. (See, for example, the essential early work of Weber [Web78], Whitt [Whi86], and Winston [Win77], as well as the more recent work by Adan and others [AWZ90, Ada94].) Of course, such a load balancing scheme requires some means of centralization. In a completely decentralized environment, attempting to determine the shortest queue might be expensive, in terms of time or other overhead. Rather than just assigning tasks randomly, we might consider having each task examine a small number of servers and go to the least loaded of the queues examined. This idea appeared in early work by Eager, Lazowska, and Zahorjan [ELZ86a].

We consider the case where each task chooses $d \geq 2$ servers at random. The fluid limit analysis for this setting and its surprising implications were found independently by Vvedenskaya, Dobrushin, and Karpelevich [VDK96] and Mitzenmacher [Mit96b, Mit96a].

As previously, we let $s_i(t)$ be the fraction of queues with load *at least* i at time t. The differential equations describing the fluid limit process are easily established.

$$\left\{ \begin{array}{lcl} \dfrac{ds_i}{dt} & = & \lambda(s_{i-1}^d - s_i^d) - (s_i - s_{i+1}) \quad \text{for} \quad i \geq 1\,; \\ s_0 & = & 1. \end{array} \right. \tag{11}$$

Let us explain the reasoning behind the system in (11). Consider a system with n queues, and determine the expected change in the number of servers with at least i customers over a small period of time of length

dt. The probability a customer arrives during this period is $\lambda n\, dt$, and the probability an arriving customer joins a queue of size $i-1$ is $s_{i-1}^d - s_i^d$. (This is the probability that all d servers chosen by the new customer are of size at least $i-1$ but not all are of size at least i.) Thus the expected change in the number of queues with at least i customers due to arrivals is exactly $\lambda n(s_{i-1}^d - s_i^d)dt$, and the expected change in the fraction of queues with at least i customers due to arrivals is therefore $\lambda(s_{i-1}^d - s_i^d)dt$. Similarly, as the number of queues with i customers is $n(s_i - s_{i+1})$, the probability that a customer leaves a server of size i in this period is $n(s_i - s_{i+1})dt$. Thus the expected change in s_i due to departures is $-(s_i - s_{i+1})dt$. Putting it all together, and replacing the expected change by ds_i, we obtain the system (11).

To determine the long range behavior of the system above requires looking for a *fixed point*. A fixed point (also called an *equilibrium point* or a *critical point*) is a point where for all i, $\frac{ds_i}{dt} = 0$. Intuitively, if the system reaches its fixed point, it will stay there.

Lemma 15 *The system (11) with $d \geq 2$ and $\lambda < 1$ has a unique fixed point with $\sum_{i=1}^{\infty} s_i < \infty$ given by*

$$s_i = \lambda^{\frac{d^i - 1}{d-1}}.$$

Proof: It is easy to check that the proposed fixed point satisfies $\frac{ds_i}{dt} = 0$ for all $i \geq 1$. Conversely, from the assumption $\frac{ds_i}{dt} = 0$ for all i we can derive that $s_1 = \lambda$ by summing the equations (11) over all $i \geq 1$. (Note that we use $\sum_{i=1}^{\infty} s_i < \infty$ here to ensure that the sum converges absolutely. The condition corresponds to the natural condition that expected number of tasks in the system is finite at the fixed point. That $s_1 = \lambda$ at the fixed point also follows intuitively from the fact that at the fixed point, the rate at which customers enter and leave the system must be equal.) The result then follows from (11) by induction. ∎

Intuitively, we would expect a well-behaved system to converge to its fixed point. In fact one can show that the trajectory of the fluid limit process given by the system (11) indeed converges to its fixed point [Mit96a, VDK96]; in fact, it does so exponentially quickly [Mit96a]. That is, the L_1 distance to the fixed point decreases like $c_1 e^{-c_2 t}$ for some constants c_1 and c_2. These results imply that in the limit as n gets large, the equilibrium distribution of a system with n queues is tightly concentrated around the fixed point [VDK96]. In fact, this behavior is readily seen even when the number of servers n is around 100.

Looking at the fixed point, we see that the tails decrease doubly exponentially in d. Hence we expect to see much shorter queues when $d \geq 2$ as opposed to when a server is chosen uniformly at random, i.e. $d = 1$. In fact, both the maximum queue length and the expected time in the system decrease exponentially! More formally, we know that the expected time in the system for an M/M/1 queue in equilibrium is $\frac{1}{1-\lambda}$, and hence this is the expected time in the setting with n servers when tasks choose servers uniformly at random. If we let $T_d(\lambda)$ be the expected time for a task in the system corresponding to the fluid limit model (i.e. as n grows to infinity) for $d \geq 2$, then

$$\lim_{\lambda \to 1^-} \frac{T_d(\lambda)}{\log \frac{1}{1-\lambda}} = \frac{1}{\log d}.$$

That is, as the system is saturated, the average time a task spends in the system when it queues at the shortest of $d \geq 2$ choices grows like the logarithm of the average time when just one choice is made. The result is remarkably similar in flavor to the original result by Azar et. al. [ABKU99]; a more compelling, simple explanation of this connection would certainly be interesting.

It is worth emphasizing, however, the importance of this result in the queueing model. In the case of the static balls-and-bins problem, the difference between one choice and two choices is relatively small, even when the number of balls and bins is large: with one million balls and one million bins, when the balls are distributed randomly, the maximum load is generally at most 12, while using two choices the maximum load drops to 4. Because the average time spent waiting before being served depends on the load λ, even with a small number of servers, having two choices can have a great effect under high load. For example, with one hundred servers at an arrival rate of $\lambda = 0.99$ per server, randomly choosing a server leads to an average time in the system of 100 time units; choosing the shortest of two reduces this to under 6!

4.2.1 Simple variations

The flexibility of the fluid limit approach allows many variations of this basic scheme to be examined. For example, suppose there are two classes of tasks entering the system. High priority tasks choose the shortest of two servers, while low priority tasks choose a random server; the servers, however, are still FIFO. In this case the corresponding fluid limit model is governed by the following set of differential equations:

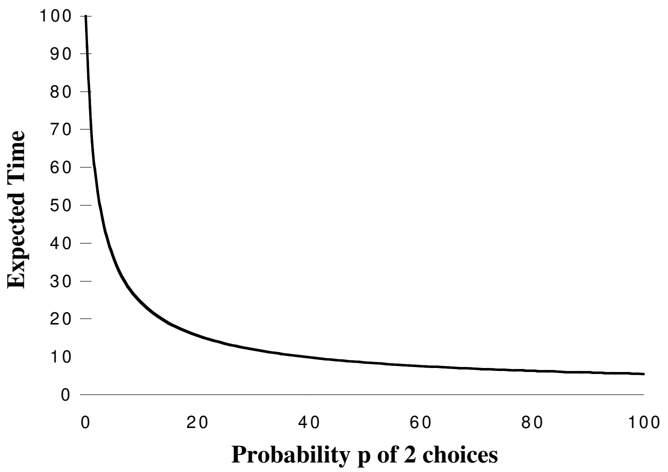

Figure 4: Expected time in the system versus probability (p) of that a customer chooses two locations ($\lambda = 0.99$).

$$\begin{cases} \dfrac{ds_i}{dt} & = & \lambda p(s_{i-1}^2 - s_i^2) + \lambda(1-p)(s_{i-1} - s_i) - (s_i - s_{i+1}) \text{ for } i \geq 1\,; \\ s_0 & = & 1. \end{cases} \tag{12}$$

The fixed point is given by $s_1 = \lambda$, $s_i = \lambda s_{i-1}(1 - p + ps_{i-1})$. There does not appear to be a convenient closed form for the fixed point for general values of p. Note that at the fixed point, it is easy to determine the distribution of the queue length customers of each priority join.

Surprisingly, the effect of increasing the fraction of customers with two choices has a nonlinear effect on the average time a customer spends in the system that is dramatic at high loads. Figure 4, which was obtained by numerically solving for the fixed point, demonstrates this phenomenon at $\lambda = 0.99$; most of the gain occurs when only 20% of the customers have two choices. Simulation results verify this behavior. The intuition for this effect is that the average queue length is not linear in the load; at high loads small increases in the load can dramatically increase average queue lengths. Hence, even giving a small fraction of the incoming tasks additional information greatly reduces the average queue length. This example demonstrates how the fluid limit approach can be used to gain significant insights into the original problem by studying variations in a simple, natural way.

As another example, we consider the variation considered by Vöcking described earlier in Section 3.1.1 [Vöc99]. We simplify by focusing on the case where $d = 2$. In this case, there are two sets of servers, with half of the bins on the left and half on the right. An incoming task chooses two servers, one uniformly at random from the left and one from the right, and queues at the server with fewer customers. Ties are broken always in favor of the bins on the left. Let $y_i(t)$ be the fraction of the n servers that have load at least i and are in the group on the left. Similarly, let $z_i(t)$ be the fraction of the n bins that have load at least i and are in the group on the right. Note $y_i(t), z_i(t) \leq 1/2$, and $y_0(t) = z_0(t) = 1/2$ for all time. Also, if we choose a random bin on the left, the probability that it has load at least i is $\frac{y_i}{1/2} = 2y_i$. The differential equations describing the limiting process are thus

$$\frac{dy_i}{dt} = 4\lambda\,(y_{i-1} - y_i)\,z_{i-1} - (y_i - y_{i+1})\ ; \tag{13}$$

$$\frac{dz_i}{dt} = 4\lambda\,(z_{i-1} - z_i)\,y_i - (z_i - z_{i+1}). \tag{14}$$

That is, for y_i to increase, our choice on the left must have load $i-1$, and the choice on the right must have load at least $i-1$. For z_i to increase, our choice on the right must have load $i-1$, but now the choice on the right must have load at least i. For y_i to decrease, a task must leave a server on the left with load i, and similarly z_i decreases when a task leaves a server on the right with load i. This system appears substantially more complex than the standard dynamic two-choice model; in fact, there is as yet no proof that there is a unique fixed point, although experiments suggest that this is the case. Indeed, even if the fixed point is unique, it does not appear to have a simple closed form. We shall assume that the fixed point for this system is unique from here on. An argument in [MV98] demonstrates that such a fixed point must be strictly better than the fixed point for uniform selection of two servers, in the following sense. If u_i represents the fraction of servers with load at least i at the fixed point for the system given by (13) and (14), then $u_i \leq \lambda^{2^i-1}$ for all i.

Using the fluid limit, it is simple to consider the following natural variation: suppose we split the left and the right sides unevenly. That is, suppose the left contains $\alpha \cdot n$ bins, and the right contains $(1-\alpha)\cdot n$ bins. Then $y_0 = \alpha$, $z_0 = 1-\alpha$ for all time, and by the same reasoning as for equations (13) and (14),

$$\frac{dy_i}{dt} = \frac{1}{\alpha(1-\alpha)}\,(y_{i-1} - y_i)\,z_{i-1} - (y_i - y_{i+1}).\ ; \tag{15}$$

i	s_i	u_i $\alpha = 0.5$	u_i $\alpha = 0.53$
1	0.9000	0.9000	0.9000
2	0.7290	0.7287	0.7280
3	0.4783	0.4760	0.4740
4	0.2059	0.1998	0.1973
5	0.0382	0.0325	0.0315
6	0.0013	0.0006	0.0005

Figure 5: The tails of the distribution at $\lambda = 0.9$.

$$\frac{dz_i}{dt} = \frac{1}{\alpha(1-\alpha)}(z_{i-1} - z_i)\, y_i - (z_i - z_{i+1}). \tag{16}$$

Interestingly, an even split is not best! As shown in Figure 4.2.1, for $\lambda = 0.9$, the tails fall slightly faster using a somewhat uneven split. In general the right value of α depends on λ; as λ increases to 1, the best value of α approaches 1/2. Increasing the fraction of processors on the left appears to mitigate the tendency of processors on the left to be more heavily loaded than those on the right.

Other variations that can be easily handled include constant service times; other service distributions can also be dealt with, although with more difficulty [Mit99a, VS97]. Threshold-based schemes, where a second choice is made only if a first choice has high load, are easily examined [Mit99a, VS97]. Closed models where customers recirculate require only minor changes [Mit96b]. Similar simple load stealing models, such as those those developed in [ELZ86b], can also be attacked using the fluid limit approach, as in [Mit98]. Tackling these variations with differential equations allows insight into how the changes affect the problem and yields a simple methodology for generating accurate numerical results quickly.

We now consider two variations on the basic queueing model that appear both particularly interesting and open for further research.

4.2.2 Dealing with stale information

Thus far, we have assumed that the load information obtained by a task when deciding at which server to queue is completely accurate. This may not always be the case. For example, there may be an *update delay*, if load information may be updated infrequently. Alternatively, if there is some delay in transferring a task to its queue choice, the load information it obtained will necessarily not reflect the load when it actually joins the queue. If these delays are of the same order as the processing time for a job, it can have dramatic effects. This model was considered by Mirchandaney, Towsley, and Stankovic in [MTS89]. Further work using fluid limit models appeared in [Mit00], which we follow. Additional simulation studies and novel models appear in [Dah99].

For example, let us again consider a system of n FIFO servers and Poisson arrivals at rate λn of tasks with exponentially distributed service requirements. Suppose that queue length information is available on a global bulletin board, but it is updated only periodically, say every T units of time. We might choose to ignore the bulletin board and simply have each task a choose a random server. We might allow a task to peek at the bulletin board at a few random locations, and proceed to the server with the shortest posted queue from these random choices. Or a task might look at the entire board and proceed to the server with the shortest posted queue. How does the update delay effect the system behavior in these situations?

In this case, an appropriate limiting system utilizes a two-dimensional family of variables to represent the state space. We let $P_{i,j}(t)$ be the fraction of queues at time t that have true load j but have load i posted on the bulletin board. We let $q_i(t)$ be the rate of arrivals at a queue of size i at time t; note that, for time-independent strategies (that is, strategies that are independent of the time t) , the rates $q_i(t)$ depend only on the load information at the bulletin board and the server selection strategy used by the tasks. In this case, if we denote the time that the last phase began by T_t, then $q_i(t) = q_i(T_t)$, and the rates q_i change only when the bulletin board is updated.

We first consider the behavior of the system during a phase, or at all times $t \neq kT$ for integers $k \geq 0$. Consider a server showing i customers on the bulletin board, but having j customers: we say such a server is in state (i,j). Let $i,j > 1$. What is the rate at which a server leaves state (i,j)? A server leaves this state when a customer departs, which happens at rate $\mu = 1$, or a customer arrives, which happens at rate $q_i(t)$. Similarly, we may ask the rate at which customers enter such a state. This can happen if a customer arrives at a server with load i posted on the bulletin board but having $j-1$ customers, or a customer departs from a server with load i posted on the bulletin board but having $j+1$ customers. This description naturally leads us to model the behavior of the system by the following set of differential equations:

$$\frac{dP_{i,0}(t)}{dt} = P_{i,1}(t) - P_{i,0}(t)q_i(t)\,; \tag{17}$$

$$\frac{dP_{i,j}(t)}{dt} = (P_{i,j-1}(t)q_i(t) + P_{i,j+1}(t)) - (P_{i,j}(t)q_i(t) + P_{i,j}(t)),\ j \geq 1 \quad (18)$$

These equations simply measure the rate at which servers enter and leave each state. (Note that the case $j = 0$ is a special case.)

At times t where the board is updated, a discontinuity occurs. At such t, necessarily $P_{i,j}(t) = 0$ for all $i \neq j$, as the load of all servers is correctly portrayed by the bulletin board. If we let $P_{i,j}(t^-) = \lim_{z \to t^-} P_{i,j}(z)$, so that $P_{i,j}(t^-)$ represents the state just before an update, then

$$P_{i,i}(t) = \sum_j P_{j,i}(t^-).$$

Experiments with these equations suggest that instead of converging to a fixed point, because of the discontinuity at update times, the system converges to a *fixed cycle*. That is, there is a state such that if the limiting system begins a phase in that state, then it ends the phase in the same state, and hence repeats the same cycle for every subsequent phase. Currently, however, there is no known proof of conditions that guarantee this cyclic behavior.

The age of the load information can have dramatic effect on the performance of the system. We provide a representative example that demonstrates the issues that arise. Figure 6 presents simulation results for $n = 100$ server at $\lambda = 0.9$. (The case of one random choice was not simulated; since each server in this case acts as an M/M/1 queue, the equilibrium distribution is fully known.) The results from numerically solving the fluid limit model are not included simply because they would be difficult to distinguish from the simulation results; the simulation results are within 1-2% of the results obtained from the fluid limit model, except in the case of choosing the shortest queue, where the simulations are within 8-17% of the fluid limit model. (Modeling the shortest queue system requires an additional approximation that causes some inaccuracy; see [Mit00] for details.) Simulations were performed for 50,000 time steps, with data collected only after the first 5,000 steps to allow the dependence on the initial state to not affect the results. The values of T simulated were $T = 0, 0.1, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 10.0, 15.0, 20.0, 25.0, 50.0$. The results presented are the average of three separate simulations.

An interesting, and at first glance counter-intuitive, behavior that immediately manifests is that going to the apparently shortest queue can be a terribly bad strategy. The intuition explaining this phenomenon becomes clear when we recall that the board contains out of date information about

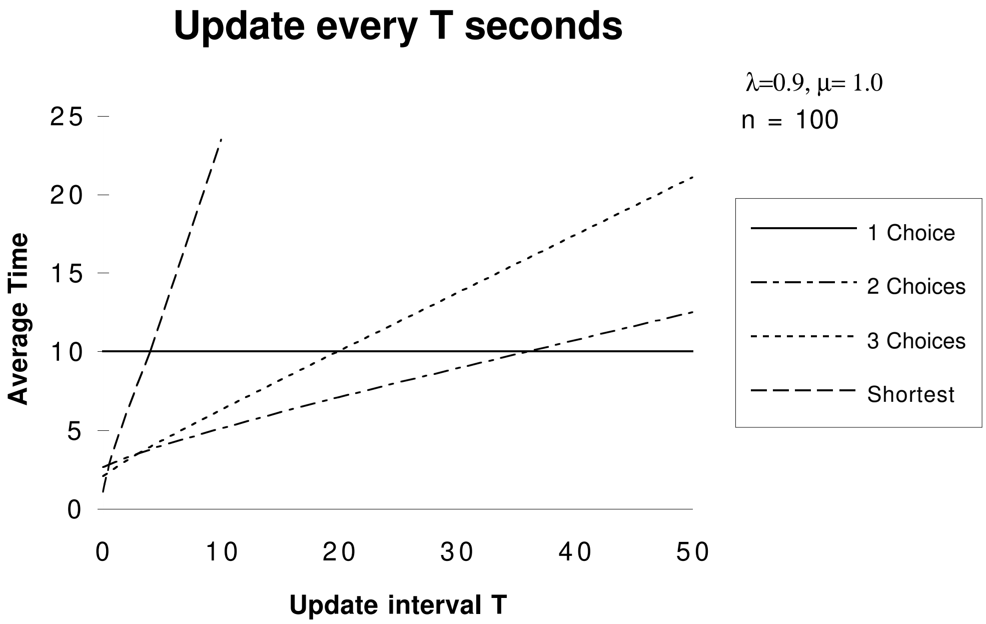

Figure 6: Comparing different strategies at $\lambda = 0.90$, 100 queues.

the loads. The problem is that all of the incoming tasks seek out the small number of queues with small load, so that a rush of tasks all head to a processor until the board later appropriately updates its load. The tasks essentially exhibit a herd behavior, moving together in the same fashion, due to the unfortunate feedback given by the system.

Another way to describe this intuition is to consider what happens at a market when it is announced that "Aisle 7 is now open." Very often Aisle 7 quickly becomes the longest queue. This herd behavior has been noticed in real systems that use old information in load balancing; for example, in a discussion of the TranSend system, Fox et al. note that initially they found "rapid oscillations in queue lengths" because their system updated load information periodically [FGC+97][Section 4.5].

Even given this intuition, it is still rather surprising that even for reasonably small delays, choosing the shortest of two randomly selected processors is a better global strategy than having all tasks choose the shortest from three! The reasoning remains the same: choosing the shortest from three processors skews the distribution towards a smaller set of processors, and when the updates are not quick enough to reflect this fact, poor balance ensues. Of course, as the delay between updates reaches infinity, even two choices performs worse than simple random selection!

Of course, whenever there is more information, better performance can result. For example, suppose that whenever a task is placed on a server the board is appropriately marked. In this case the periodic updates simply provide information about how many tasks have finished. In this setting, going to the apparently shortest queue again becomes a worthwhile strategy. This solution was the one adopted by Fox et al. [FGC+97]; note, however, that it requires that tasks be able to update the board.

Although these initial results provide some insight into the problems that arise in using stale information about queue lengths, the general problem of how to cope with incomplete or inaccurate load information and still achieve good load balancing performance appears to be an area with a great deal of research potential.

4.2.3 Networks of Server Banks

Once the case of a bank of servers has been handled, one might ask about the case of networks of such systems. In particular, we review the well-understood case of Jackson networks. A standard Jackson network consists of J servers labeled $1, \ldots, J$. Each server has an infinite buffer and services tasks at rate μ_j. For each server j there is an associated Poisson arrival process of rate λ_j. A task repeatedly queues for service, according to the rule that when it finishes at server j, it moves to server k with probability p_{jk} and leaves the system with probability $1 - \sum_k p_{jk}$. (For convenience we assume that the p_{jk} are such that the expected number of servers visited is always finite.)

The state of a Jackson network can be given by a vector representing the queue lengths, $\vec{n}(t) = (n_1(t), \ldots, n_J(t))$, where $n_i(t)$ is the length of the queue at the ith server at time t. Under this formulation the state $\vec{n}(t)$ is a Markov process. The stationary distribution may be described as follows. We consider the *effective arrival rate* ρ_j at each server j. This is a combination of the external arrival rate plus the arrival rate from nodes within the network. These effective arrival rates ρ_j satisfy

$$\rho_j = \lambda_j + \sum_k \rho_k p_{kj}.$$

The tails of the queue lengths for the system of queues satisfy

$$\mathbf{Pr}(n_j \geq r_j \text{ for all } j) = \prod_j \left(\frac{\rho_j}{\mu_j} \right)^{r_j}.$$

The standard interpretation for this result is that in equilibrium each server looks like an independent M/M/1 queue with the appropriate arrival and service rates.

Suppose instead of a standard Jackson network of single servers we have a similar network consisting of server banks, with each bank having a large number of servers n. Of course the arrival rates are scaled so that the arrival rate at each server bank is $n\lambda_j$, and the effective arrival rate at each server bank is $n\rho_j$. A job seeking service at a bank of servers selects a server from that bank to queue at by taking the shortest of $d \geq 2$ random choices. (When $d = 1$, the system is easily seen to be equivalent to a standard Jackson network.) Given our previous results on server systems and the results for standard Jackson networks one might hope that in the limit as n grows to infinity, the load n_{ij} at the ith server of the jth bank would have the distribution:

$$\mathbf{Pr}(n_{ij} \geq r_j) = \left(\frac{\rho_j}{\mu_j}\right)^{\frac{d^{r_j}-1}{d-1}}.$$

Moreover, we would hope that the Jackson-like network continues to have the property that in equilibrium each bank of servers appears to be an independent bank of servers with the appropriate arrival and service rates.

Indeed, Martin and Suhov prove this to be the case in [MS99]. They call the resulting system a *Fast Jackson network*, since the queue lengths decrease doubly exponentially, and hence the expected time in a system is naturally faster than in a standard Jackson network. Further work by Martin examines the stochastic processes that arise from such networks in greater detail [Marb, Mara].

Given the results of Vöcking [Vöc99], it seems clear that a Jackson-like network of servers using the tie-breaking scheme leads to *Faster Jackson Networks*. A full proof of this fact, however, will require extending the work of [MS99] to this situation. Another direction to take is to try to extend these results to broader classes of networks, or use these results to bound the performance of networks that may not exhibit such pleasant properties. Given the wealth of results showing that two choices are substantially better than one in both static and dynamic situations, it seems that extending this idea in more directions in network scenarios would be worthwhile. Although the results for butterfly-like networks presented in Section 3.4 suggest that the analysis of the two-choice paradigm can be significantly more complex, these results suggest that the fluid limit approach still has potential in this area.

References

[ABKU99] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29:180–200, 1999. A preliminary version of this paper appeared in *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, 1994.

[ABP98] N. S. Arora, R. D. Blumofe, and C. G. Plaxton. Thread scheduling for multiprogrammed multiprocessors. In *Proceedings of the Tenth ACM Symposium on Parallel Algorithms and Architectures*, pages 119–129, June 1998.

[ABS98] M. Adler, P. Berenbrink, and K. Schröder. Analyzing an infinite parallel job allocation process. *Lecture Notes in Computer Science (Proceedings of ESA 1998)*, 1461:417–428, 1998.

[ACMR95] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 238–247, May 1995.

[Ada94] I. J. B. F. Adan. *A compensation approach for queueing problems.* CWI (Centrum voor Wiskunde en Informatica), 1994.

[AFP98] J. Aronson, A. Frieze, and B. Pittel. Maximum matchings in sparse random graphs: Karp-Sipser revisited. *Random Structures and Algorithms*, 12(2):111–177, March 1998.

[AH90] M. Alanyali and B. Hajek. Analysis of simple algorithms for dynamic load balancing. *Mathematics of Operations Research*, 22(4):840–871, 1990.

[Ale82] R. Aleliunas. Randomized parallel communication. In *Proceedings of the ACM SIGACT–SIGOPS Symposium on Principles of Distributed Computing*, pages 60–72, August 1982.

[AM97] D. Achlioptas and M. Molloy. The analysis of a list-coloring algorithm on a random graph. In *Proceedings of the Thirty-Eighth Annual Symposium on Foundations of Computer Science*, pages 204–212, 1997.

[AS92] N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley and Sons, 1992.

[AWZ90] I. J. B. F. Adan, J. Wessels, and W. H. M. Zijm. Analysis of the symmetric shortest queue problem. *Stochastic Models*, 6:691–713, 1990.

[BCSV00] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking. Balanced allocations: The heavily loaded case. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 745–754, 2000.

[BD97] R. Bubley and M. Dyer. Path coupling: a technique for proving rapid mixing in Markov chains. In *Proceedings of the Thirty-Eighth Annual Symposium on Foundations of Computer Science*, pages 223–231, 1997.

[BFM98] P. Berenbrink, T. Friedetzky, and E. W. Mayr. Parallel continuous randomized load balancing. In *Proceedings of the Tenth ACM Symposium on Parallel Algorithms and Architectures*, pages 192–201, 1998.

[BFS99] P. Berenbrink, T. Friedetzky, and A. Steger. Randomized and adversarial load balancing. In *Proceedings of the Eleventh ACM Symposium on Parallel Algorithms and Architectures*, pages 175–184, 1999.

[BK90] A. Z. Broder and A. Karlin. Multi-level adaptive hashing. In *Proceedings of the First Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 43–53, 1990.

[BL94] R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. In *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, pages 356–368, November 1994.

[BM00] A. Broder and M. Mitzenmacher. Using multiple hash functions to improve ip lookups. Technical Report TR–03–00, Department of Computer Science, Harvard University, Cambridge, MA, 2000.

[BMS97] P. Berenbrink, F. Meyer auf der Heide, and K. Schröder. Allocating weighted jobs in parallel. In *Proceedings of the Ninth ACM Symposium on Parallel Algorithms and Architectures*, pages 302–310, June 1997.

[CFM+98] R. Cole, A. Frieze, B.M. Maggs, M. Mitzenmacher, A. W. Richa, R. K. Sitaraman, and E. Upfal. On balls and bins with deletions. In *Second International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, number 1518 in Lecture Notes in Computer Science, pages 145–158. Springer–Verlag, October 1998.

[CH91] J. P. Crametz and P. J. Hunt. A limit result respecting graph structure for a fully connected loss network with alternative routing. *The Annals of Applied Probability*, 1(3):436–444, 1991.

[CMM+98] R. Cole, B. M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A. W. Richa, K. Schröder, R. K. Sitaraman, and B. Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 378–388, May 1998.

[CMS95] A. Czumaj, F. Meyer auf der Heide, and V. Stemann. Shared memory simulations with triple-logarithmic delay. *Lecture Notes in Computer Science*, 979:46–59, 1995.

[CS97] A. Czumaj and V. Stemann. Randomized allocation processes. In *Proceedings of the Thirty-Eighth Annual Symposium on Foundations of Computer Science*, pages 194–203, October 1997.

[Czu98] A. Czumaj. Recovery time of dynamic allocation processes. In *10th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 202–211, 1998.

[Dah99] M. Dahlin. Interpreting stale load information. In *Proceedings of the Nineteenth Annual IEEE International Conference on Distributed Computing Systems*, 1999. To appear in *IEEE Transactions on Parallel and Distributed Systems*.

[DKM+88] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. Tarjan. Dynamic perfect hashing — upper and lower bounds. In *Proceedings of the Twenty-Ninth Annual Symposium on Foundations of Computer Science*, 1988.

[EK86] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and convergence.* John Wiley and Sons, 1986.

[ELZ86a] D. L. Eager, E. D Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12:662–675, 1986.

[ELZ86b] D. L. Eager, E. D. Lazowska, and J. Zahorjan. A comparison of receiver-intiated adaptive load sharing. *Performance evaluation Review*, 16:53–68, March 1986.

[FGC+97] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Extensible cluster-based scalable network services. In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, October 1997.

[FKS84] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $o(1)$ worst-case access time. *Journal of the ACM*, 31:538–544, 1984.

[FM87] R. Finkel and U. Manber. DIB – A distributed implementation of backtracking. *ACM Transactions on Programming Languages and Systems*, 9(2):235–256, April 1987.

[FMM91] R. Feldmann, P. Mysliwietz, and B. Monien. A fully distributed chess program. In *Advances in Computer Chess 6*, pages 1–27, Chichester, UK, 1991. Ellis Horwood.

[GMR94] L. A. Goldberg, Y. Matias, and S. Rao. An optical simulation of shared memory. In *Proceedings of the Sixth ACM Symposium on Parallel Algorithms and Architectures*, pages 257–267, June 1994.

[Gon81] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28(2):289–304, 1981.

[Haj88] B. Hajek. Asymptotic analysis of an assignment problem arising in a distributed communications protocol. In *Proceedings of the 27th Conference on Decision and Control*, pages 1455–1459, 1988.

[HR90] T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33:305–308, February 1990.

[HZJ94] M. Halbherr, Y. Zhou, and C. F. Joerg. MIMD-style parallel programming based on continuation-passing thread. Memo CSG

Memo-335, Massachussets Institute of Technology,Laboratory for Computer Science, Computation Structures Group, March 1994. Also appears in *Proceedings of Second International Workshop on Massive Parallelism: Hardware, Software and Applications*, October 1994.

[Jer98] M. Jerrum. Mathematical foundations of the markov chain monte carlo method. In *Probabilistic Methods for Algorithmic Discrete Mathematics, Algorithms and Combinatorics*, pages 116–165. Springer–Verlag, 1998.

[JK77] N. Johnson and S. Kotz. *Urn Models and Their Application.* John Wiley and Sons, 1977.

[KLM92] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 318–326, May 1992.

[KLM96] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. *Algorithmica*, 16:245–281, 1996.

[KMPS95] A. Kamath, R. Motwani, K. Palem, and P. Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Structures and Algorithms*, 7(1):59–80, August 1995.

[Knu73] D. E. Knuth. *The Art of Computer Programming I–III.* Addison–Wesley, Reading, MA, second edition, 1973.

[KS81] R. M. Karp and M. Sipser. Maximum matchings in sparse random graphs. In *Proceedings of the Twenty-Second Annual Symposium on Foundations of Computer Science*, pages 364–375, 1981.

[Kur70] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, 7:49–58, 1970.

[Kur81] T. G. Kurtz. *Approximation of Population Processes.* SIAM, 1981.

[KVV90] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[KZ93] R. M. Karp and Y. Zhang. Randomized parallel algorithms for backtrack search and branch-and-bound computation. *Journal of the ACM*, 40(3):765–789, July 1993.

[Lei92] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

[Lin92] T. Lindvall. *Lectures on the Coupling Method*. John Wiley and Sons, 1992.

[LMRR94] F. T. Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, July 1994.

[LMS+97] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 150–159, 1997.

[Mara] J. B. Martin. Point processes in fast Jackson networks. Submitted for publication.

[Marb] J. B. Martin. Stochastic bounds for fast Jackson networks. Submitted for publication.

[Mit96a] M. Mitzenmacher. Load balancing and density dependent jump Markov processes. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 213–222, 1996.

[Mit96b] M. D. Mitzenmacher. *The power of two choices in randomized load balancing*. PhD thesis, University of California at Berkeley, Department of Computer Science, Berkeley, CA, 1996.

[Mit98] M. Mitzenmacher. Analyses of load stealing models using differential equations. In *Proceedings of the Tenth ACM Symposium on Parallel Algorithms and Architectures*, pages 212–221, 1998.

[Mit99a] M. Mitzenmacher. On the analysis of randomized load balancing schemes. *Theory of Computing Systems*, 32:361–386, 1999.

[Mit99b] M. Mitzenmacher. Studying balanced allocations with differential *Combinatorics, Probability, and Computing*, 8:473–482, 1999.

[Mit00] M. Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 11(1):6–20, 2000.

[MPR98] P. D. Mackenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. *Journal of the ACM*, 45(2):324–378, March 1998.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[MS92] Bruce M. Maggs and Ramesh K. Sitaraman. Simple algorithms for routing on butterfly networks with bounded queues (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 150–161, May 1992. To appear in SIAM Journal on Computing.

[MS99] J. B. Martin and Y. M Suhov. Fast Jackson networks. *Annals of Applied Probability*, 9:854–870, 1999.

[MSS96] F. Meyer auf der Heide, C. Scheideler, and V. Stemann. Exploiting storage redundancy to speed up randomized shared memory. *Theoretical Computer Science*, 162(2):245–281, 1996.

[MTS89] R. Mirchandaney, D. Towsley, and J. A. Stankovic. Analysis of the effects of delays on load sharing. *IEEE Transactions on Computers*, 38:1513–1525, 1989.

[MV98] M. Mitzenmacher and B. Vöcking. The asymptotics of selecting the shortest of two, improved. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pages 326–327, 1998. Full version available as Harvard Computer Science TR-08-99.

[Ran87] A. G. Ranade. Constrained randomization for parallel communication. Technical Report YALEU/DCS/TR-511, Department of Computer Science, Yale University, New Haven, CT, 1987.

[RCG94] R. Rooholamini, V. Cherkassky, and M. Garver. Finding the right ATM switch for the market. *Computer*, 27(4):16–28, April 1994.

[RSAU91] L. Rudolph, M. Slivkin-Allalouf, and E. Upfal. A simple load balancing scheme for task allocation in parallel machines. In *Proceedings of the Third Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 237–245, July 1991.

[SS94] M. Shaked and J. Shantikumar. *Stochastic Orders and Their Applications*. Academic Press, Inc., 1994.

[Ste96] V. Stemann. Parallel balanced allocations. In *Proceedings of the Eighth ACM Symposium on Parallel Algorithms and Architectures*, pages 261–269, June 1996.

[Sto83] D. Stoyan. *Comparison Methods for Queues and Other Stochastic Models*. John Wiley and Sons, 1983.

[SW95] A. Shwartz and A. Weiss. *Large Deviations for Performance Analysis*. Chapman & Hall, 1995.

[TY97] J. Turner and N. Yamanaka. Architectural choices in large scale ATM switches. Technical Report WUCS–97–21, Department of Computer Science, Washington University, St. Louis, MO, 1997.

[Upf84] E. Upfal. Efficient schemes for parallel communication. *Journal of the ACM*, 31(3):507–517, July 1984.

[Val82] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, May 1982.

[VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 263–277, May 1981.

[VDK96] N.D. Vvedenskaya, R.L. Dobrushin, , and F.I. Karpelevich. Queueing system with selection of the shortest of two queues: an asymptotic approach. *Problems of Information Transmission*, 32:15–27, 1996.

[Vöc99] B. Vöcking. How asymmetry helps load balancing. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, pages 131–140, 1999.

[VS97] N.D. Vvedenskaya and Y.M. Suhov. Dobrushin's mean-field approximation for a queue with dynamic routing. *Markov Processes and Related Fields*, 3(4):493–526, 1997.

[Web78] R. Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probabiblities*, 15:406–413, 1978.

[Whi86] W. Whitt. Deciding which queue to join: Some counterexamples. *Operations Research*, 34:55–62, 1986.

[Win77] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probabilities*, 14:181–189, 1977.

[Wor95] N. C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probabilities*, 5:1217–1235, 1995.