Combinatorics, Probability and Computing (1999) 8, 473–482. Printed in the United Kingdom

Studying Balanced Allocations with Differential Equations†

MICHAEL MITZENMACHER§

Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA
(e-mail: michaelm@pa.dec.com)

Received 4 November 1997; revised 25 June 1998

Using differential equations, we examine the GREEDY algorithm studied by Azar, Broder, Karlin and Upfal for distributed load balancing [1]. This approach yields accurate estimates of the actual load distribution, provides insight into the exponential improvement GREEDY offers over simple random selection, and allows one to prove tight concentration theorems about the loads in a straightforward manner.

1. Introduction

Suppose that n balls are placed into n bins, with each ball being placed into a bin chosen independently and uniformly at random. Let the *load* of a bin be the number of balls in that bin after all balls have been thrown. It is well known that, with high probability, the maximum load upon completion will be approximately $\frac{\log n}{\log \log n}$ [8]. (We use *with high probability* to mean with probability at least $1 - O(1/n)$, where n is the number of balls. Also, log will always mean the natural logarithm, unless otherwise noted.)

Azar, Broder, Karlin and Upfal considered how much more evenly distributed the load would be if each ball had two (or more) choices [1]. Suppose that the balls are placed sequentially, and each ball is placed into the less full of *two* bins chosen independently and uniformly at random with replacement (breaking ties arbitrarily). In this case, they showed that the maximum load drops to $\frac{\log \log n}{\log 2} + O(1)$ with high probability. If each ball instead

† A preliminary version of this work appeared in the *Proceedings of the 37th Annual IEEE Symposium on the Foundations of Computer Science*, October 1996.
§ Much of this work was done at U.C. Berkeley, supported by a fellowship from the Office of Naval Research and by NSF grant CCR-9505448.

has d choices, then the maximum load will be $\frac{\log\log n}{\log d} + O(1)$ with high probability. Having two choices hence yields a qualitatively different type of behaviour from the single choice case, leading to an exponential improvement in the maximum load; having more than two choices further improves the maximum load by only a constant factor. This result has important implications for distributed load balancing, hashing, and PRAM simulation [1].

Following Azar, Broder, Karlin and Upfal, we refer to the algorithm in which each ball has d random choices as GREEDY(d). In this paper, we develop an alternative method of studying the performance of GREEDY(d) using differential equations. The differential equations describe the limiting performance of GREEDY(d) as the number of balls and bins tends to infinity. As we will demonstrate, the description of the limiting performance proves highly accurate, even when n is relatively small. In particular, we note that from our results one can determine the fraction of bins of any fixed load at the end of the process for the limiting case as n goes to infinity. These limiting quantities provide accurate estimates for the fraction of bins of each fixed load for sufficiently large systems. This analysis therefore adds significantly more detail to the previous analysis of [1]. Moreover, besides giving more insight into the actual performance of GREEDY, our methods provide a great deal of intuition behind the behavioural difference between one and two choices.

Our motivation in studying this problem is twofold. First, we wish to demonstrate and highlight this methodology, and encourage its use for studying random processes. While this methodology is by no means new, its uses have been surprisingly limited. The technical results justifying the relationship between families of Markov processes and differential equations date back at least to Kurtz [13, 14]. Karp and Sipser provided an early use of this technology to analyse an algorithm for finding maximum matchings in sparse random graphs [11]. Other past applications in the analysis of algorithms include [9, 12], and more recently many more have been found (see, for example, [2, 3, 15, 17, 21], to name a few).

Our second motivation is to demonstrate the power of using two choices. This idea dates back at least as far as the work of Eager, Lazowska and Zahorjan [7], who examined a dynamic load balancing model based on viewing processors as single server queues. In the static setting, this idea was also studied by Hajek [9], who used the same approach we undertake to determine the fraction of empty bins. The aforementioned exponential improvement in behaviour was noted and proved first in a paper by Karp, Luby and Meyer auf der Heide [10]. The work by Azar, Broder, Karlin and Upfal examined a simpler model that clarified the argument and provided many new results. Related work by the author [16, 17], as well as by others [20], examines the power of two choices in dynamic settings. Continued work in the area includes recent work by Stemann [19] and Czumaj and Stemann [6].

In the rest of the paper, we explain the derivation of the differential equations that describe the GREEDY strategy of [1] and compare the results from the differential equations with simulations. We also demonstrate how the equations give more insight into the behaviour of GREEDY and how the equations relate to the work in [1].

2. The differential equations

In this section, we demonstrate how to establish a family of differential equations that can be used to model the behaviour of the GREEDY strategy of [1]. We begin the process with m balls and n bins. We shall require for our analysis that $m = cn$ for some constant c. Balls arrive sequentially, and, upon arrival, each ball chooses d bins independently and uniformly at random (with replacement); the ball is then placed in the least loaded of these bins (with ties broken arbitrarily).

We first ask how many bins remain empty after the protocol GREEDY(d) terminates. This question has a natural interpretation in the task-processor model: how many of our processors are not utilized? The question can also be seen as a matching problem on random bipartite graphs: given a bipartite graph with n vertices on each side such that each vertex on the left has d edges to vertices chosen independently and uniformly at random on the right, what is the expected size of the greedy matching obtained by sequentially matching vertices on the left to a random unmatched neighbour? Our attack, again, is to consider this system as $n \to \infty$. This question has been previously solved by Hajek using entirely similar techniques [9]. We shall briefly repeat his argument with some additional insights. Once we show how to answer the question of the number of empty bins, we shall extend it to the more general load balancing problem.

2.1. The empty bins problem

We set up the problem of the number of empty bins 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 non-empty bins after T balls have been thrown. Then $\{Y(i)\}, i = 0 \ldots m$, is clearly a Markov chain. Moreover,

$$\mathsf{E}[Y(T+1) - Y(T)] = 1 - \left(\frac{Y(T)}{n}\right)^d, \tag{2.1}$$

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

The notation becomes somewhat more convenient if we scale by a factor of n. We let t be the time at which exactly nt balls have been thrown, and we let $y(t)$ be the fraction of non-empty bins. Then equation (2.1) becomes

$$\frac{\mathsf{E}[y(t+1/n) - y(t)]}{1/n} = 1 - (y(t))^d. \tag{2.2}$$

We claim the random process described by equation (2.2) is well approximated by the trajectory of the differential equation

$$\frac{dy}{dt} = 1 - y^d, \tag{2.3}$$

where this equation has been obtained from equation (2.2) by replacing the right-hand side with the appropriate limiting value as n tends to infinity, dy/dt. That the random

process given by the Markov chain closely follows the trajectory given by the differential equation follows easily from known techniques, such as, for example, Kurtz's theorem, or the similar work on random graphs by Wormald [21]. (As previously mentioned, the balls and bins process has a natural interpretation in terms of random bipartite graphs.)

To clarify this connection, here we state a form of Kurtz's theorem, as given in [18, Theorem 5.3]. We provide the necessary notation, and then relate the notation back to the underlying balls and bins process. Suppose we are given a finite set of vectors $\{\vec{e}_1, \ldots, \vec{e}_k\}$ in $\mathbf{R}^d$. We consider an initial process $\vec{x}(t)$ with generator

$$Lf(\vec{x}) = \sum_{i=1}^{k} \lambda_i(\vec{x})(f(\vec{x} + \vec{e}_i) - f(\vec{x}))$$

and a scaled process $\vec{z}_n(t)$ with generator

$$L_n f(\vec{x}) = \sum_{i=1}^{k} n\lambda_i(\vec{x}) \left(f\left(\vec{x} + \frac{\vec{e}_i}{n}\right) - f(\vec{x}) \right).$$

The limiting operator L_∞ satisfies

$$L_\infty f(\vec{x}) = \sum_{i=1}^{k} n\lambda_i(\vec{x}) \langle \nabla f(\vec{x}), \vec{e}_i \rangle,$$

and corresponds to the deterministic solution $\vec{z}_\infty$ of the equation

$$\frac{d}{dt}\vec{z}_\infty(t) = \sum_{i=1}^{k} \lambda_i(\vec{z}_\infty(t))\vec{e}_i. \tag{2.4}$$

We relate the above notation to the problem of keeping track of the fraction of non-empty bins. For convenience, think of the times balls enter the system as being determined by a Poisson arrival process at a rate of one per unit time. (This does not affect the result; it merely simplifies the discussion. See, for example, [13].) In the scaled process with n balls and n bins, n balls arrive per unit time. The process is one-dimensional, and hence $\vec{e}_1 = (1)$. The rate function λ_1 satisfies $\lambda_1(\vec{x}) = 1 - x_1^d$: this is just the probability that an incoming ball lands in an empty bin. The limiting process is then given by the differential equation (2.4), which in this case is equivalent to equation (2.3); note that for convenience we have dropped the vector notation and simply used the variable y.

Kurtz's theorem states that the scaled processes approach the limiting process, with error bounds similar to Chernoff-like bounds.

Theorem 2.1 (Kurtz's theorem [18], Theorem 5.3). *Let $\lambda_i(\vec{x}) : \mathbf{R}^d \to \mathbf{R}^+$ be uniformly bounded and Lipschitz continuous, and let $\vec{z}_\infty$ be the unique solution of* (2.4) *with $\vec{z}_\infty(0) = \vec{x}(0)$. For each finite T there exist a positive constant C_1 and a function C_2 with*

$$\lim_{\epsilon \downarrow 0} \frac{C_2(\epsilon)}{\epsilon^2} \in (0, \infty) \quad \text{and} \quad \lim_{\epsilon \uparrow \infty} \frac{C_2(\epsilon)}{\epsilon} = \infty$$

such that, for all $n \geqslant 1$ *and* $\epsilon > 0$,

$$\Pr\left(\sup_{0 \leqslant t \leqslant T} |\vec{z}_n(t) - \vec{z}_\infty(t)| \geqslant \epsilon\right) \leqslant C_1 e^{-nC_2(\epsilon)}.$$

Moreover, C_1 *and* C_2 *can be chosen independently of* $\vec{x}$.

The connection between the balls and bins process and the differential equation (2.3) yields the following theorem.

Theorem 2.2. *Suppose* cn *balls are thrown into* n *bins according to the* GREEDY(d) *protocol for some constant* c. *Let* Y_{cn} *be the number of non-empty bins when the process terminates. Then* $\lim_{n\to\infty} \mathsf{E}[\frac{Y_{cn}}{n}] = y_c$, *where* $y_c < 1$ *satisfies*

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

Proof. The preconditions for Kurtz's theorem (the above or [14, Chapter 8]) are easily checked for the one-dimensional system described by (2.3), so by Kurtz's theorem we have that this differential equation is the correct limiting process.† Instead of solving (2.3) for y in terms of t, we solve for t in terms of y: $\frac{dt}{dy} = \frac{1}{1-y^d} = \sum_{i=0}^{\infty} y^{id}$. We integrate up to some time t, yielding

$$t = \sum_{i=0}^{\infty} \frac{y(t)^{id+1}}{(id+1)}. \tag{2.5}$$

From equation (2.5), given d we can solve for $y(t)$ for any value of t using, for example, binary search. One can also attempt to find an equation for y in terms of d and t; standard integral tables give such equations when $d = 2, 3$ and 4, for example. When $t = c$, all of the balls have been thrown, and the process terminates. Plugging $t = c$ into equation (2.5) yields the theorem, with $y_c = y(c)$. □

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

Theorem 2.3. *In the notation of Theorem 2.2,* $|\frac{Y_{cn}}{n} - y_c|$ *is, with high probability,*

$$O\left(\sqrt{\frac{\log n}{n}}\right),$$

where the constant depends on c. □

† Again, it appears that there might be a problem here since we consider events occurring at discrete time-steps, instead of according to random times from a Poisson process. One can always adopt the convention that each discrete time-step corresponds to an amount of time given by an exponentially distributed random variable. In the limiting case, this distinction disappears.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.