The Randomness in Randomized Load Balancing[*]

Chandra Nair
Dept. of Electrical Engineering
Stanford University
Stanford, CA 94305.
mchandra@stanford.edu

Balaji Prabhakar
Depts. of EE & CS
Stanford University
Stanford, CA 94305
balaji@isl.stanford.edu

Devavrat Shah
Dept. of Computer Science
Stanford University
Stanford, CA 94305
devavrat@cs.stanford.edu

Abstract

Load balancing is a classical and important problem arising in a number of application scenarios. Consider the following canonical abstraction: Jobs arrive according to an arrival process at a bank of N identical servers each having a separate queue. The arriving jobs need to be assigned to the servers so that load on the servers is well balanced. The policy "join the server with shortest remaining work" is known to be an optimal policy (Winston, 1977). When N is large, the complexity of implementing this optimal scheme becomes very high. Therefore, many simple randomized approximations have been proposed in the recent literature. These various algorithms trade-off performance for implementation simplicity. In this paper, we wish to understand this trade-off by using the entropy rate of the induced queue-size process as a metric; i.e., the lower the entropy rate of the queue-size process, the better the load-balancing.

1 Introduction

Load balancing has received considerable attention in the classical literature [4, 5]. We consider the following canonical model, some times called the "supermarket model". Jobs arrive according to a rate $N\lambda$ (where $\lambda < 1$) Poisson process at a bank of N independent rate 1 exponential server queues. The arriving jobs are to be assigned to the servers so as to balance the load; or, more specifically, to minimize the expected delay or the expected queue size. It is well-known that the assignment policy "join the shortest queue" is optimal [12]. But, when N is large it becomes expensive to determine the shortest queue for each arriving job.

This has recently motivated several randomized algorithms, all aimed at simplifying the implementation of the assignment process. Azar *et. al.* [1] considered the following static version of the load balancing problem: Drop N balls into N bins so as to minimize

[*]This research is supported by a Stanford Graduate Fellowship, and by grants from a Alfred P. Sloan Fellowship and a Terman Fellowship

the maximum loading. The policy "join the least loaded bin" results in an optimal loading of one ball per bin. In contrast, consider the class of randomized algorithms: join the shortest of $d \geq 1$ *randomly chosen* bins. Azar *et. al.* [1] show that for $d \geq 2$ the maximum load is $\frac{\ln \ln n}{\ln d} + O(1)$ with a high probability, as compared to $\frac{\ln n}{\ln \ln n}(1 + o(1))$ for $d = 1$.

The dynamic supermarket model mentioned above has been studied by Mitzenmacher [6] and Vvedenskaya *et.al.* [11]. For $d = 1$ the supermarket model reduces to N independent M/M/1 queues with arrival rate λ and service rate 1, and the queue-size distribution is geometric; that is, $P(Q \geq i) = \lambda^i$. For $d \geq 2$, the following rather remarkable fact is established in [6] and [11]. The tail of the queue-size distributions is super-geometric; that is, $P(Q \geq i) = \lambda^{\frac{d^i - 1}{d-1}}$ as $N \to \infty$.

More recently, Shah and Prabhakar [9] consider randomized load balancing algorithms which use *memory*. Specifically, they consider the static problem of dropping N balls into N bins by defining the following "(d,1) system". At the beginning of the k^{th} iteration the $(d, 1)$ system has stored in its memory the identity of the least loaded bins at the end of the $(k-1)^{th}$ iteration. During the k^{th} iteration it selects d new bins uniformly at random and assigns the k^{th} ball to the least loaded of these d bins and the one bin retained in memory. It concludes the k^{th} iteration by writing the identity of the least loaded of the $d + 1$ bins into memory. In this language [1] studies the $(d, 0)$ system. It is shown in [9] that the maximum load in the $(d, 1)$ system is bounded above $\frac{\ln \ln n}{\ln(2d-1)} + O(1)$ with a high probability. Thus, the $(d, 1)$ system performs no worse than the $(2d - 1, 0)$ system so far as minimizing the maximum load is concerned.

Clearly, one can define the $(d, 1)$ version of the supermarket model in a similar way. For the remainder of the paper we will be concerned exclusively with the dynamic supermarket model. Due to Poisson arrivals and independent exponential services, the joint queue-size process at time t, $Q(t) = (q_1(t), ..., q_N(t))$, corresponding to each of the policies introduced above is an irreducible, aperiodic and ergodic continuous-time Markov chain. A major thrust of this paper is to compute the entropy rate of these Markov chains. Our interest for doing this rests on the following large conclusion obtained in the paper: As d increases the entropy rate of the $(d, 0)$ *decreases*, and is the smallest for the join the shortest queue system (which corresponds to the $(N, 0)$ system when sampling is done without replacement). Since systems are also better load-balanced as d increases, we hope to connect the goodness of the load-balancing of an assignment policy with the smallness of the entropy rate of the queue-size process it induces.

Now, the entropy rate of a discrete-time Markov chain living on a countable state space is given by $-\sum_{ij} \pi_i p_{ij} \log p_{ij}$, where π_i is the invariant distribution and $\{p_{ij}\}$ is the transition matrix (see Chapter 4 of [2], for example). Unfortunately, except for the $(1, 0)$ system, the invariant distribution of all the systems mentioned above is difficult, if not impossible, to compute explicitly. Further, if the arrival and service processes are allowed to be arbitrarily distributed, then explicit formulas for the entropy rate of the queue-size process are not known.

Nevertheless, we shall be interested in computing the entropy rate of such arbitrary load-balancing systems. We do this by using the method employed in Prabhakar and Gallager [8]. The method of [8] requires a discrete-time formulation which we shall adopt in Section 1.1 and the main theorems concerning entropy rate of the $(d, 0)$ systems are established in Section 2. The computation of entropy rates for the $(d, 1)$ system, which uses memory, seems more complicated. In particular, this makes it difficult to compare the performance of systems using memory with that of systems which use no memory with entropy as the metric. As mentioned above, [9] shows that the $(d, 1)$ system is at least as good as the $(2d-1, 0)$ system in the metric of minimizing the maximum load. In Section 3.1 we use simulations to further compare systems which use memory with systems that do not by introducing another metric, motivated by the entropy approach. Finally, in Section 4 we conclude the paper by discussing the lessons learnt about the relevance of entropy as a metric for load balancing.

1.1 The Model and Notation

Consider a system of N first-come-first-served (FCFS) queues arranged in parallel each with an independent rate 1 server providing i.i.d. service times. The arrival process to the system of queues, A, is assumed to be stationary and ergodic. We assume that time is slotted and that there is at most one arrival per time slot. The arrivals occur at the beginning of the time slots and departures occur just before the end of the time slots. The service time distribution, S, is arbitrary and, for convenience, we assume that $P(S=0)=0$. This ensures that at most one departure can occur at each queue in each time slot.

Let $Q(k^+) = (q_1(k^+), ..., q_N(k^+))$ be the queue size vector at time k^+, where k^+ is the time just after the occurrence of possible arrivals in time slot k. Similarly, define $Q(k^-) = (q_1(k^-), ..., q_N(k^-))$, where k^- is the time just after departures in time slot $k-1$. Write $Q(k) = (Q(k^-), Q(k^+))$.

Let $\sigma(k)$ be the permutation of numbers $1, \dots, N$ which arranges the queues according to their size at time k^-; i.e. $q_{\sigma_1(k)}(k^-) \leq \dots \leq q_{\sigma_N(k)}(k^-)$. To disambiguate between several possible permutations when there are ties in queue sizes, assume that ties are broken in a deterministic manner; for example, according to port numbers.

Let $p = (p_1, \dots, p_N)$ be a probability vector representing the probabilities of the outcome of the toss of a coin with N sides, and let $p_1 \geq p_2 \geq \cdots \geq p_N$. With a given vector p, we may identify an assignment policy as follows. If a packet arrives in time slot k, we toss an N-sided coin distributed according to p. If the outcome of the coin toss is C, $1 \leq C \leq N$, then the packet joins the queue $\sigma_C(k)$. Let $\mathcal{T}$ be the class of algorithms describable by such a coin toss model.

We are now ready to identify the systems $(d, 0)$ as algorithms in the above class. The extreme case $(1, 0)$ is identified with the loading probability vector $(\frac{1}{N}, ..., \frac{1}{N})$. The other extreme case $(N, 0)$ (join the shortest queue) corresponds to the loading probability vector $(1, ..., 0)$. Observe that ties are broken in a deterministic fashion in the $(N, 0)$ system. The set of algorithms corresponding to the $(d, 0)$, $1 < d < N$, also belong to class $\mathcal{T}$ with corresponding loading probability vectors

$$p_i^d = \frac{\binom{N-i+1}{d} - \binom{N-i}{d}}{\binom{N}{d}}. \tag{1}$$

2 Entropy Rate

The development in this section closely parallels the arguments in [8]. We borrow the arguments used in [8] for the single queue system and apply them to the N-queue system with the small changes that the increase in queues warrants. Let $\dots, a_{-2}, a_{-1}, a_0, a_1, a_2, \dots$ be the doubly-infinite ordered sequence of time slots in which arrivals occur. We adopt the convention, $a_0 < 0 \leq a_1$. Let s_i be the service time obtained by the packet which arrived in the slot a_i. Let $A_i = a_{i+1} - a_i$ be the sequence of inter-arrival times. Let C_i denote the outcome of the toss of the N-sided coin when the ith packet arrives. We also assume that the arrivals, services and the coin tosses are independent of each other.

Definition 1 *The bank of N parallel $\cdot$/GI/FCFS queues is said to satisfy the Q-condition if the number of packets in each of the queues in equilibrium has a finite first moment; i.e. $E(q_i(0^-)) < \infty, \quad$ for all $i = 1, ..., N$.*

Conditions: First of all, we observe that even for a single queue system it is necessary for the service distribution to have finite second moment, i.e $E(s_1^2) < \infty$. Further, it is sufficient for the arrival process to be strongly mixing [8]. Now, we wish to find sufficient conditions which guarantee that the Q-condition holds for our system of N servers. Consider a particular algorithm: join a queue chosen uniformly at random. For this case it is easy to see that if the arrival process to the entire bank of queues is strongly mixing, then the property is preserved for each of the arrival processes to the N queues. In fact, if the overall arrival process is geometric, so are the individual arrival processes.

Let T be any algorithm in $\mathcal{T}$ and let p be its associated coin toss vector. We have, by definition of $\mathcal{T}$, $p_1 \geq \cdots, p_N$. Let $\tilde{q}(j)$ be the total number of packets present in the least loaded j queues. Let $\hat{q}(N-j)$ be the total queue size in the most loaded $N-j$ queues. Observe that, for every j, we have, $\sum_{i=1}^{j} p_i \geq \sum_{i=1}^{j} \frac{1}{N}$. This implies that, in algorithm T it is more likely for any packet to go into $\tilde{q}(j)$, than in algorithm $(1, 0)$. From, this it not difficult to see that T is better load balanced than $(1, 0)$. Specifically, it is easy to see that the total amount of service rendered by all the queues in T upto to any time t, will stochastically dominate the same in the $(1, 0)$ algorithm. Given this fact, it is not difficult to conclude that if the $(1, 0)$ system satisfies the Q-condition, so does all other algorithms in $\mathcal{T}$. Therefore, it is sufficient that: (a) arrivals are strongly mixing or renewal, and (b) service times have finite second moment.

Lemma 1 *Let $\{Q_k, k \in \mathbb{Z}\}$ be the equilibrium queue-size process of the parallel bank of N $\cdot$/GI/1-FCFS queue satisfying the Q-condition. Then, $H(Q_k) < \infty$ or equivalently $H(q_i(k)) < \infty$ for all $i = 1, ..., N$.*

Proof The random variables $q_i(k^-)$ and $q_i(k^+)$ are non-negative, integer-valued and have finite means. Therefore, their entropies are lesser than geometric random variables with means equal to $E(q_i(k^-))$ and $E(q_i(k^+))$, respectively. It follows that $H(q_i(k^-)) + H(q_i(k^+)) < \infty$. ■

Residual services: Let $v_i(k^+)$ denote the ordered vector of packets in queue $q_i(k^+)$ along with the amount of service each has *yet to receive.*

Definition 2 *The bank of N parallel $\cdot$/GI/FCFS queues is said to satisfy the V-condition if the entropy of the residual services are finite; i.e. $H(v_i(k^+)) < \infty$ for all $i = 1, ..., N$.*

It is demonstrated in [8] that if the Q-condition is satisfied then the V-condition will be satisfied if the services have bounded support.

The condition on services having bounded support is not necessary. For example, if the services are i.i.d geometric, the residual services, by the memory-less property, will also be geometric. In this case, it is easy to see that as long as the Q-condition is satisfied, $H(V(k^+)) \leq H(s)E(\sum_{i=1}^{N} q_i(k^+)) + \sum_{i=1}^{N} H(q_i(k^+)) < \infty$. Here, $H(s)$, denotes the entropy of the service distribution.

Theorem 1 *Suppose the arrival process to the N-queue system is stationary, ergodic and renewal. Furthermore, let the service distribution be independent of the arrival process and i.i.d. Let both the Q-conditions and the V-conditions be satisfied. Then the entropy rate of the queue-size process of any algorithm which belongs to $\mathcal{T}$ is equal to $\lambda(H_{ER}(A) + H(S) + H(C))$.*

Proof For $K > 0$ consider the queue-size process restricted to $[0, K]$: $\{Q_0, \ldots, Q_K\}$. Let $N(K) = \max\{n : a_n \leq K\}$ be the number of arrivals in $[0, K]$. Given the vector of residual services in each queue at time zero, all the arrival instances in $[0, K]$, the outcomes of the coin tosses and the services times of the packets, one can run the queue in forwards time to get the queue-size at all times $[0, K]$. One could also obtain the vector of residual services in each queue at time K^+.

Now given the queue-size at all times $[0, K]$ and the vector of residual services in each queue at time K^+, one can find the arrival instances in $[0, K]$ by looking for an upward jump in any of the queues. One can also determine the outcome of the coin tosses during this interval by looking at the particular queue in which an upward jump occurs. From the downward jumps of the queue, one can compute the departure instances and so using these arrivals and departures one can compute the services of the packets which departed (residual services for all the packets that were already present in the queue at time zero) during this period in all of the FCFS queues. With the help of the residual services in each queue at time K^+, we could also compute the services of the packets in the system at time K. Thus, we have the following bijection:

$$(Q_0, \ldots, Q_K; V_{K^+}) \leftrightarrow (Q_0, V_{0^+}, a_1, A^{N(K)-1}, S^{N(K)}, p^{N(K)}). \tag{2}$$

By Lemma 1 it follows that $H(Q_0) < \infty$. This and the ergodicity of the process $\{Q_k, k \in \mathbb{Z}\}$ imply that it has a finite entropy rate. We have also seen that both $H(V_{0^+})$ and $H(V_{K^+})$ are finite.

Now taking entropies[1] at (2), dividing both sides by K and letting K go to infinity we get

$$\begin{aligned} H_{ER}(\mathbf{Q}) \triangleq \lim_{K\to\infty} \frac{H(\{Q_k; 0 \leq k \leq K\})}{K} &= \lim_{K\to\infty} \frac{H(A^{N(K)-1}, S^{N(K)}, C^{N(K)})}{K}. \\ &\overset{a}{=} \lambda(H_{ER}(A) + H(S) + H(C)). \end{aligned}$$

The paper by Prabhakar and Gallager [8], gives a detailed argument establishing (a). ■

3 Loading probability vector: Majorization, Load Balancing and Entropy Rate

This section deals with the relationship of the loading probability vectors to load balancing and entropy rate. All the loading probability vectors we consider satisfies $p_1 \geq \cdots \geq p_N$. Let T_1, T_2 be two algorithms which belong to $\mathcal{T}$ and let p^{T_1} and p^{T_2} be their loading probability vectors. We ask the question: Is it possibble to look at p^{T_1}, p^{T_2} and determine which system balances the load better?

[1] This is precisely where the discrete time setting is required, since in continuous time, entropies are not always preserved by bijections.

A vector p^{T_1} is said to majorize p^{T_2}, ($p^{T_1} \gg p^{T_2}$), with the elements of the vectors arranged in decreasing order, if

$$\sum_{i=1}^{k} p_i^{T_1} \leq \sum_{i=1}^{k} p_i^{T_2}, \quad \text{for} \quad k = 1, \ldots, N.$$

Here, since both p^{T_1} and p^{T_2} are probability vectors, we have equality holding for $k = N$. So, for such systems, it is quite clear that the algorithm T_2, which is represented by p^{T_2}, is a better load balancer than the algorithm T_1 represented by p^{T_1}. Let the loading probability vector for the $(k, 0)$ system be denoted by p^k.

Lemma 2 $p^d \gg p^{d+1}$.

Proof From the definition of p^d in equation (1), we have

$$\sum_{i=1}^{k} p_i^d = \sum_{i=1}^{k} \frac{\binom{N-i+1}{d} - \binom{N-i}{d}}{\binom{N}{d}} = \frac{\binom{N}{d} - \binom{N-i}{d}}{\binom{N}{d}}.$$

Therefore, it is sufficient to show that,

$$\frac{\binom{N}{d} - \binom{N-i}{d}}{\binom{N}{d}} \leq \frac{\binom{N}{d+1} - \binom{N-i}{d+1}}{\binom{N}{d+1}} \quad \text{for} \quad i = 1, \ldots, N.$$

This reduces to showing

$$\frac{(N-i)!(N-d-1)!}{N!(N-i-d-1)!} \leq \frac{(N-i)!(N-d)!}{N!(N-i-d)!} \quad \Leftrightarrow \quad N - i - d \leq N - d.$$

Since this is true for all $i \geq 1$ and hence the proof is complete. ■

Let two systems, T_1 and T_2 be such that $p^{T_1} \gg p^{T_2}$. From Theorem 1, it is possible to compute the entropy rates of the two systems, T_1 and T_2. It is interesting that whenever $p^{T_1} \gg p^{T_2}$, then we have from the following lemma that the entropy rate of queue-size process of T_1 is greater than that of T_2.

Lemma 3 *If $p^{T_1} \gg p^{T_2}$, then $H(p^{T_1}) \geq H(p^{T_2})$.*

Proof: We know from Theorem 4.3.33, page 197 of [3], that if $p^{T_1} \gg p^{T_2}$, then there exists a doubly stochastic matrix, Γ, such that $p^{T_1} = \Gamma p^{T_2}$. Given this and the concavity of the entropy, the result readily follows from Jensen's inequality. ■

3.1 Systems with memory

Consider the (d,1) system. Let us look at the bijection in equation (2). Given all the arrivals, coin toss outcomes, services and the initial conditions at time zero, we can compute the queue-size process until time K by running the queues in forwards time. But given the queue-size vector, we cannot compute the outcome of the coin tosses at precisely those instances when the memory bin was loaded. Thus, it is quite easy to see that the entropy rate of the queue-size process for the $(d, 1)$ system is lesser than that of the $(d, 0)$ system.

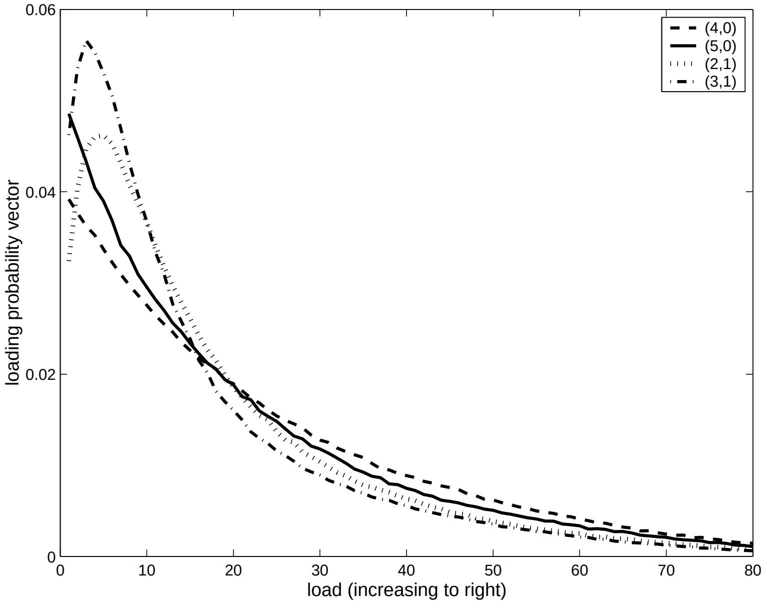

Figure 1: Loading probability vectors for different systems

It is difficult, if not impossible, to calculate the loading probability vector for the $(d, 1)$ system analytically. Therefore, we use simulations to emperically determine the loading probability vector for systems with memory. Theoretically, even if we calculate the loading probability vectors, one would still not be able to calculate the entropy rate of the $(d, 1)$ system by the Theorem 1. This is due to the fact that the loads of the bins chosen at any time k are dependent on the load of the bin in memory at time k, and the latter is not independent from time to time. Therefore the entropy rate of systems with memory is difficult to compute using the method of bijections. However, we note that it is possible to obtain upper bounds on the entropy rate of systems with memory by considering systems in which have the memory effects last only over a bounded duration of time. We do not pursue this further here, but instead use simulations to study the $(d, 1)$ systems.

Simulation setup: We consider a bank of 100 parallel $\cdot$/M/1-FCFS queues.The arrival rate to the system is 0.99*100 packets per second, and each of the hundred queues is served by a unit rate exponential server. Every arriving packet is loaded to the least loaded of the d randomly sampled queues and the queue in memory. Ties in queue sizes are broken uniformly at random. The simulations are run for 10 million packet arrivals. This number of iterations was observed to be sufficient for the emperical loading probability vectors of the $(d, 0)$ systems (for $d = 4, 5$) to converge to their theoretical values determined at equation (1). Further, it is noted in [1] that the mixing time of Markov chains with variations of the static $(d, 0)$ systems is of the order of N^3 (equal to 100^3 here).

Using this setup for simulation, we obtain the loading probability vectors for the (2,1) system and the (3,1) system. The simulations were done in MATLAB and the random number generator used is that of MATLAB version 6.0.

Figure 1 plots the loading probability vectors of the (2,1) and the (3,1) systems along with those of the (4,0) and the (5,0) systems. From the figure we observe that the loading probability vectors of the $(d, 1)$ systems are not necessarily monotonically decreasing.

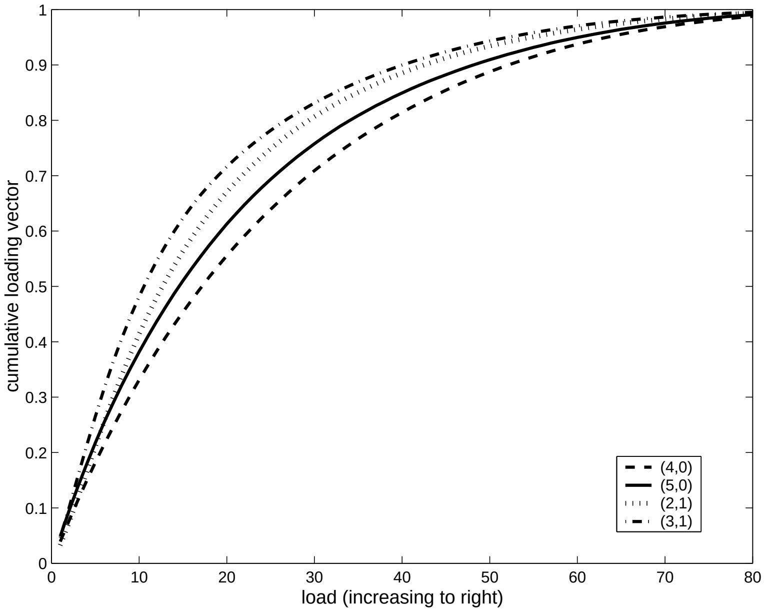

Figure 2: Cumulative loading vectors for different systems

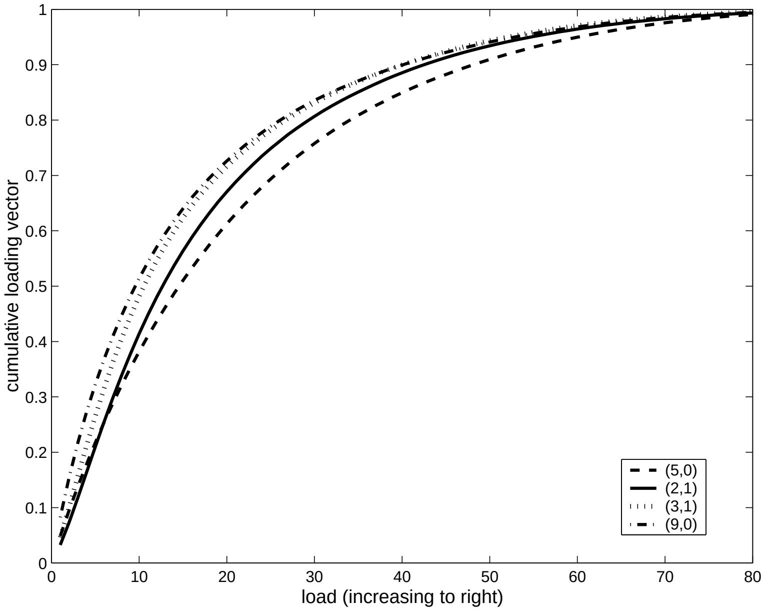

Figure 3: Cumulative loading vectors for different systems

Hence, they do not belong to the class of algorithms $\mathcal{T}$ in general. Note that if we look at the least loaded 20 bins, the systems with memory load these bins with greater probability than the (4,0) and the (5,0) systems. This is better illustrated in Figure 2, which plots the cumulative loading vectors of the various systems. The cumulative loading vector is the cumulative distribution obtained from the loading probability distribution.

From Figure 2, one can expect that the (2,1) and the (3,1) systems load lesser loaded queues more frequently than the (4,0) and (5,0) systems. Thus, it is possible that the $(d, 1)$ system with memory might perform much better than the $(2d-1, 0)$ lower bound proved in [9]. To get an upper bound on the performance of the (3,1) system, we simulated various $(k, 0)$ systems and found that the (9,0) system is the smallest value for k that majorizes the (3,1) system as shown in Figure 3.

4 Conclusions

The paper considered different load balancing algorithms and the queue-size processes generated by these algorithms. The entropy rate of the queue-size processes was computed using the method of bijections introduced in [8] for a class of such algortihms, which includes the (d,0) systems. We defined the notion of a loading probability vector associated with every algorithm in this class, and found that if the loading probability vectors of two algorithms T_1 and T_2 are such that p^{T_1} majorizes p^{T_2}, then T_1 has a worse load balancing performance than T_2. Moreover, the entropy rate of the queue-size process under T_1 is shown to be higher than that under T_2.

The recent work of Vöcking [10] obtains a rather puzzling result. He considers loading N balls into N bins in the following fashion: Divide the N bins into d groups and take one sample at random from each group. Load the ball into the least loaded bin, breaking ties in of two ways (i) always to the left, and (ii) uniformly at random. The surprising conclusion is that breaking ties to the left leads to better load balancing. Mitzenmacher and Vöcking [7] generalize this result to the continuous supermarket version. It is possible to easily show that the entropy rate of the queue-size process corresponding to the break ties to the left system is smaller than that of the break ties at random system, the extra entropy being injected by the coins tossed to break ties. Based on the initial work of this paper for relating the smallness of entropy rate to the goodness of load balancing, we hope it is possible to better understand the phenomenon discovered by Vöcking [10] on how "asymmetry helps load balancing".

References

[1] Y. Azar, A. Broder, A. Karlin, and E. Upfal, "Balanced allocations", *Proc. of the 26th ACM Symp. on Theory of Computing (STOC)*, pp. 593-602, 1994.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[3] R. A. John and C. R. Johnson, *Matrix Analysis*, (1st paperback edition), Cambridge: Cambridge University Press, 1991.

[4] N. Jonson, and S. Kotz, *Urn Models and their Applications*, John Wiley and Sons, 1977.

[5] V. F, Kolchin, B. A. Sevasťyanov, and V. P. Chistyakov, *Random Allocations*, V. H. Winston and Sons, 1978.

[6] M. Mitzenmacher, "The power of two choices in Randomized load balancing," *PhD thesis, University of California at Berkeley*, 1996.

[7] M. Mitzenmacher and B. Vöcking, "The Asymptotics of Selecting the Shortest of Two, Improved," *Proc. of the Allerton Conference on Communication, Control and Computing*, Urbana, Illinois, 1999.

[8] B. Prabhakar and R. Gallager, "Entropy and the Timing Capacity of Discrete Queues," *In preparation*, 2001.

[9] D. Shah and B. Prabhakar, "The use of memory in randomized load balancing," *In preparation*, 2001.

[10] B. Vöcking, "How Asymmetry helps load balancing", *FOCS*, 1999.

[11] 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*, vol. 32, pp. 15-27, 1996.

[12] W. Winston, "Optimality of the shortest line discipline," *Jour. of Appl. Probability*, vol. 14, pp. 181-189, 1977.