Balanced Allocations: Balls-into-Bins Revisited and Chains-into-Bins

Tuğkan Batu* Petra Berenbrink† Colin Cooper‡

21 January 2008
CDAM Research Report LSE-CDAM-2007-34

Abstract

The study of *balls-into-bins games* or *occupancy problems* has a long history since these processes can be used to translate realistic problems into mathematical ones in a natural way. In general, the goal of a balls-into-bins game is to allocate a set of independent objects (tasks, jobs, balls) to a set of resources (servers, bins, urns) and, thereby, to minimize the maximum load.

In this paper we show two results. First, we analyse the maximum load for the *chains-into-bins* problem where we have n bins and the balls are connected in n/ℓ chains of length ℓ. In this process, the balls of one chain have to be allocated to ℓ consecutive bins. We allow each chain d i.u.r. bin choices. The chain is allocated using the rule that the maximum load of any bin receiving a ball of that chain is minimized. We show that, for $d \geq 2$, the maximum load is $(\ln\ln(n/\ell))/\ln d + O(1)$ with probability $1 - O(1/\ln\ln(n/\ell))$. This shows that the maximum load is decreasing with increasing chain length. Secondly, we analyse for which number of random choices d and which number of balls $m < n$, the maximum load of an *off-line assignment* can be upper bounded by one. This holds, for example, for $m < 0.97677 \cdot n$ and $d = 4$.

*Department of Mathematics, London School of Economics, London WC2A 2AE, UK. Email: T.Batu@lse.ac.uk.

†School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. Email: petra@cs.sfu.ca.

‡Department of Computer Science, King's College, University of London, London WC2R 2LS, UK. Email: colin.cooper@kcl.ac.uk.

1 Introduction

The study of *balls-into-bins games* or *occupancy problems* has a long history. These models are commonly used to derive results in probability theory. Furthermore, balls-into-bins games can be used as a means of translating realistic problems into mathematical ones in a natural way. In general, the goal of a balls-into-bins game is to allocate a set of independent objects (tasks, jobs, balls) to a set of resources (servers, bins, urns). It is assumed that the balls are independent and do not know anything about the other balls. Each ball is allowed to choose independently and uniformly at random (i.u.r.) a subset of the bins in order to be allocated into one of the bins. The performance of these processes is usually analyzed in terms of the maximum load of any bin.

One extreme approach for such a balls-into-bins game is to allow each ball to communicate with every bin. Thus, it is possible to query the load of every bin and to place the ball into the bin that is least loaded. This allocation process always yields an optimum allocation of the balls. However, the time and number of communications needed to allocate the balls is extremely large. The opposite approach is to allow every ball to communicate with one bin only. In this case, every ball is thrown into one of the bins, chosen independently and uniformly at random. For the case of m balls and n bins it is well known that a bin exists which receives $\Theta\left(m/n + \sqrt{2(m \log n)/n}\right)$ balls, with high probability (w.h.p.).[1]

An alternative approach which lies between these two, is to allow every ball to select one of $d \geq 2$ i.u.r. chosen bins. The GREEDY[d] process studied by Azar et al. [1], chooses d i.u.r. bins per ball, and the ball is allocated into the least loaded among the chosen bins. For this process, the maximum number of balls found in any bin, i.e., the *maximum load*, decreases to $m/n + \ln\ln n + O(1)$ (see [1], [2]). Thus, even a small amount of additional random choice can decrease the maximum load drastically. This phenomenon is often referred to as the "power of two random choices" (see [11]). It is also well known that each additional constant factor beyond two choices per ball decreases the maximum load by only an additional constant factor (see [17]).

In this paper we show two results. First, we analyse the maximum load for the *chains-into-bins* problem, where balls are connected to n/ℓ chains of length ℓ. In this process, the balls of one chain have to be allocated to consecutive bins. We allow each chain d i.u.r. bin choices. The chain is allocated using the rule that the maximum load of any bin receiving a ball of that chain is minimized. We show that, for $d \geq 2$, the maximum load is $\ln\ln(n/\ell)/\ln d + O(1)$, with probability $1 - O(1/\ln\ln(n/\ell))$. Secondly, we study off-line allocation of m balls into n bins with d choices per ball. In particular, we analyse for which values of d and m, the maximum load of an off-line assignment, using $d \geq 2$ bins chosen i.u.r. without replacement, can be upper bounded by one. This holds, for example, for $m < 0.97677 \cdot n$ and $d = 4$, and extends the work of Sanders et al. [13].

2 Known Results

Azar et al. [1] introduced Greedy[d] to allocate n balls into n bins. Greedy[d] chooses d bins i.u.r. for each ball and allocates the ball into a bin with the minimum load. They show that after placing n balls, the maximum load is $\Theta((\ln\ln(n))/\ln(d) + 1)$, w.h.p. Compared to single-choice games, this is an *exponential* decrease in the maximum load. We note that, for the case where $m < n/e$, their results can be extended to show a maximum load of at most $(\ln\ln n - \ln\ln(n/m))/\ln d + O(1)$, w.h.p.

Vöcking [17] introduced the Always-Go-Left protocol, yielding a maximum load of $(\ln\ln n)/d + O(1)$, w.h.p. The Always-Go-Left protocol clusters the bins into d clusters of n/d consecutive bins each. Every ball now chooses i.u.r. one bin from every cluster is allocated into a bin with the minimum load. If several of the chosen bins have the same minimum load, it is allocated into the "leftmost" bin. In [9], Kenthapadi and Panigrahy suggest an alternative protocol yielding the same maximum load. They cluster the bins into $2n/d$ clusters of $d/2$ consecutive bins each. Every ball now randomly chooses 2 of these clusters and it is allocated into the cluster with the smallest total load. In the chosen cluster, the ball is then allocated into the bin with minimum load again. The authors also argue in that paper that clustering is essential to reduce the

[1] A sequence of events A_n occurs with high probability if $\lim_{n\to\infty} P(A_n) = 1$.

load to $(\ln\ln n)/d + O(1)$. In [2], the authors analyse Greedy[d] for $m \gg n$. They show that the maximum load is $m/n + \ln\ln(n)$, w.h.p. Finally, Mitzenmacher et al. [12] show that a similar performance gain occurs if the process is allowed to memorise a constant number of bins with small load.

In [15], Sanders and Vöcking consider the *random arc allocation problem*, which is closely related to the chains-into-bins problem. In their model, they allocate arcs of an arbitrary length to a cycle. Every arc is assigned an i.u.r. position on the cycle. The chains-into-bins problem with $d = 1$ can be regarded as a special discrete case of their problem, where the cycle represents the n bins and the arcs represent the chains (in [15], different arc lengths are allowed). Translated into the chains-into-bins setting, the authors show the following result. If $m = n/\ell$ chains of length ℓ are allocated to n bins ($m \to \infty$) then the maximum load is at most $(\ln(n/\ell))/(\ln\ln(n/\ell))$, w.h.p. Note that their result is asymptotically the same as that for allocating n/ℓ balls into n/ℓ bins, provided that $n/\ell \to \infty$. In [7], the author shows that the expected maximum load is smaller if we allocate $n/2$ chains of length 2 with one random choice per chain, compared to n balls into n bins with $d = 2$.

Sanders et al. [13] consider an *off-line version* of the balls-into-bins problem where the balls still have d random choices, but the final allocation is computed by a centralized algorithm once the choices available to each ball are known. In [13], Sanders et al. show that it is possible to achieve a maximum load of $\lceil m/n \rceil + 1$, w.h.p., using a *centralised* flow algorithm. The authors also argue that there exists values of m for which this allocation is tight. In [14], Sanders further studies the values of m for which an optimal maximum load of $\lceil m/n \rceil$ is possible. To give some examples, let us define $k = m/n$. Then, Sanders shows that for $\lceil k \rceil \leq 1$ and $k/\lceil k \rceil \leq 0.49$ the load is w.h.p. at most 1. For $\lceil k \rceil \leq 2$ and $k/\lceil k \rceil \leq 0.83$ the load is at most 2, and for $\lceil k \rceil \leq 8$ and $k/\lceil k \rceil \leq 0.98$, the load is at most 8. Analysis similar to ours also appear in [8, 5].

3 Our Results

In this paper we consider two different balls-into-bins problems. The first problem is the *chains-into-bins* problem where we allocate m chains of length ℓ into n bins. We assume $m = \lceil n/\ell \rceil$, so that the total bin occupancy is (about) n. The more general case of sparse but arbitrary m is considered in the full version of this paper. Assume the balls are allocated uniformly at random (i.u.r.) to bins wrapped cyclically around $1, ..., n$. If chain i is allocated to bin b_i then the chain occupies bins $b_i, b_{i+1}, ..., b_{i+\ell-1}$ where the counting is modulo n. Thus the load in bin b_i is the number of chains crossing the bin, i.e., the number of chains allocated to bins in $b_{i-\ell+1}, ..., b_{i-1}, b_i$. We consider the case where every chain randomly chooses d bins $b_{i_1}, \ldots, b_{i_d}$. For a random choice b_{i_j}, we compute the maximum load of bins $b_{i_j}, b_{i_j+1}, ..., b_{i_j+\ell-1}$. The chain is now allocated to a bin b_{i_j} such that the maximum load is minimized. In this paper we show that, for $d \geq 2$, the maximum load for this algorithm is $(\ln\ln(n/\ell))/\ln d + O(1)$, with probability $1 - O(1/\ln\ln(n/\ell))$. This shows that the maximum load decreases with increasing chain length. Note that, if ℓ is fixed, our result is asymptotically the same as that for allocating n/ℓ balls into n/ℓ bins with d random choices.

To discuss our results, let us compare the maximum load we get to the maximum that a process that clusters the bins achieves. In the latter process we cluster the bins in n/ℓ clusters of ℓ consecutive bins. Each chain now i.u.r. chooses d clusters, and it is allocated into every bin of the cluster with minimum load. (Note, in this process all bins of the same cluster have the same load.) The maximum load of this process is nothing else than the maximum load of the greedy process for n/ℓ balls and n/ℓ bins, which is $\Theta((\log\log(n/\ell))/\log d + 1)$ (see [1]) with a probability of $1 - 1/(n/\ell)$. The results of Vöcking [17] and also Kenthapadi and Panigrahy [9] suggest that clustering tends to be quite helpful for balls-into-bins games such that the authors believe that the maximum load $\Theta((\ln\ln(n/\ell))/\ln d + 1)$ is tight. Note that only our probability bound is weaker than what one might expect. For a discussion of our probability guarantee, we refer the reader to the beginning of Section 4.

We also revisit the balanced allocation problem on the off-line setting introduced by Sanders et al. [13]. Let $\lambda(d) = \lambda(d, m, n)$ denote the best (minimal) maximum load per bin for the offline allocation of m balls into n bins, with d i.u.r. choices per ball (choices without replacement). The following results hold w.h.p.:

- For $m < n/2$, $\lambda(2) = 1$;

- For $m < 0.816 \cdot n$, $\lambda(3) = 1$;
- For $m < 0.97677 \cdot n$, $\lambda(4) = 1$.

We will also show (Theorem 4) that for $m > n$, $d = 2$ is optimal in the sense that with increasing d, the probability that no optimal assignment (with load $\lceil m/n \rceil$) can be found increases.

4 A Chains-into-Bins Game

The case considered here is $m = \lceil n/\ell \rceil$ chains of length ℓ. To keep things simple, we assume ℓ divides n and suppress all further rounding. Assume $m = n/\ell$ chains are allocated i.u.r. to bins wrapped cyclically round $1, ..., n$. The chains contain ℓ balls each. The first ball of a chain is called *header* in the following, the remaining balls comprise the *tail*. If chain j (meaning the header of chain j) is allocated to bin b_i, then the balls of the chain occupy bins $b_i, b_{i+1}, ..., b_{i+\ell-1}$, where counting is modulo n. In the following the *load* in bin b is the total number of balls allocated to the bin, i.e., the number of chains allocated to bins $b_{i-\ell+1}, ..., b_{i-1}, b$. We define the *h-load* of a bin as the number of headers allocated to the bin.

We consider the case where every chain randomly chooses d bins $b_{i_1}, \ldots, b_{i_d}$. For random choice b_{i_j} it computes the maximum load of bins $b_{i_j}, b_{i_j+1}, ..., b_{i_j+\ell-1}$. The chain is now allocated to b_{i_j} such that the maximum load in any affected bin is minimized. This allocation process is called *GREEDY[d] for chains*. We show the following result.

Theorem 1 *Let $m = n/\ell$ chains length ℓ be allocated to n bins, with $d \geq 2$ i.u.r. bin choices per ball. Let $\xi(d) = \xi(d, m, n, \ell)$ be an upper bound on the maximum load of any bin, obtained by GREEDY[d] for chains. Then,*

$$\xi(d) = \frac{\ln \ln(n/\ell)}{\ln d} + O(1)$$

with probability $1 - O(1/\ln \ln(n/\ell))$.

Note the following extension of [1] to the case where $m \ll n$.

Theorem 2 (ABKU99) *Provided that $m < n/e$ and $d \geq 2$, the maximum load achieved by GREEDY[d] is at most*

$$L(d) = \frac{\ln \ln n - \ln \ln(n/m)}{\ln d} + O(1)$$

with high probability.

For any ℓ the approximation ratio $\xi(d)/L(d)$ is

$$\frac{\xi(d)}{L(d)} = 1 + \frac{\log \log \ell + \log\left(1 - \frac{\log \ell}{\log n}\right)}{\log \log n - \log \log \ell} \leq 1 + \frac{\log \log \ell}{\log \frac{\log n}{\log \ell}}.$$

The worst case approximation ratio is $O(\log \log n)$, which is still better than $\xi(1)/L(d) = \Omega(\log(n/\ell))$. We remark that once $\ell = \Omega(n/\log \log n)$, then $m = n/\ell$ and $\xi(1)$ are upper bounds which improve on $\xi(d)$ for $d \geq 2$.

4.1 Analysis of GREEDY[d] for Chains

In this section, we prove Theorem 1. Recall that in this problem, the bins are arranged cyclically in the order $1, 2, ..., n$. Each chain consists of ℓ balls. When a chain is *allocated* to bin v, the chain covers bins $v, v+1, v+\ell-1$; that is, each ball of the chain occupies one of these bins. We use the following definitions:

- $\eta(v) = \eta(v, t)$ is the h-load of bin v (number of headings allocated to v) at (the end of) step $t = 0, 1, ..., n/\ell$.

- For given $A \subseteq [n]$, define $\lambda(A,t) = \sum_{v \in A} \eta(v,t)$ as the total h-load of the bins of A.
- Let $R(v) = \{v - \ell + 1, ..., v-1, v\}$.
- Let $\lambda(v) = \lambda(v,t) = \lambda(R(v),t)$ be the load of v. Thus, $\lambda(v,t)$ is the *derived load* of bin v at step t, arising from chain headers allocated to the ℓ bins of $R(v)$.
- Let $Q(i,t) = \{v : \lambda(v,t) \geq i\}$ give the labels of bins whose derived load is $\geq i$ at (the end of) step t.
- Let $S(i,t) = \cup_{v \in Q(i,t)} R(v)$. Thus $S(i,t)$ gives the labels of those bins whose occupancy by a chain header could contribute to bins with derived load $\geq i$.
- Let $\theta_{\geq i}(t) = |S(i,t)|$.

Note that even if some bins of $R(v)$ do not contribute in a given instance, we include them in $S(i,t)$ anyway. For example, suppose at step t bin v contains i ball headers and bins $v-\ell+1, ..., v-1$ are empty, and bins $v+1, ...v+\ell-1$ do not contain chain headers. Then, $\{v, v+1, ...v+\ell-1\} \subseteq Q(i,t)$ and $\{v-\ell+1, ..., v, v+1, ...v+\ell-1\} \subseteq S(i,t)$. Our proof method follows that of [1] and, in particular, Theorem 3.2 of that paper. For consistency, we have preserved their proof structure and notation as far as possible, and in some places reproduced their arguments verbatim. We begin by stating a lemma given in [1] (Lemma 3.1).

Lemma 1 (ABKU99) *Let $X_1, X_2, ..., X_n$ be a sequence of random variables with values in an arbitrary domain, and let $Y_1, Y_2, ..., Y_n$ be a sequence of binary random variables with the property that $Y_i = Y_i(X_1, ..., X_i)$. If*

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

then

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

where $B(n,p)$ denotes a binomially distributed random variable with parameters n and p.

Let h_t be the height of ball t, i.e., the maximum derived load resulting from the chosen placement of ball t. Greedy[d] allocates the ball to the bin which minimizes the maximum load. Hence,

$$h_t = \min_{j=1,...,d} \ \max \ \{\lambda(v_j + r), \ r = 0, ..., \ell - 1\},$$

where $v_1, ..., v_d$ are the u.a.r. bin choices offered at step t.

The *components* $C_j, j = 1, ..., s$, of $S(i,t)$ consist of subsets of consecutive bins contained in $S(i,t)$. These components are of length at least ℓ as the component containing v also contains $R(v)$. If we consider a component C, then the length c of this component can be written as

$$c = q\ell + r \qquad 0 \leq r \leq \ell - 1,$$

and, thus, C contains at least iq chain headers. For any component

$$\lambda(C) \geq iq = \frac{iqc}{q\ell + r} = \frac{ic}{\ell + r/q} \geq \frac{ic}{2\ell - 1}.$$

Thus,

$$\lambda(S(i,t)) \geq \frac{i}{2\ell - 1} \sum_{j=1,...,s} |C_j| = \frac{i\theta}{2\ell - 1} = \frac{i|S(i,t)|}{2\ell - 1}. \tag{1}$$

We define

$$\beta_i = \begin{cases} n & i = 1, ..., 21 \\ \frac{n}{4e} & i = 22 \\ n2e\left(\frac{\beta_{i-1}}{n}\right)^d & i > 22. \end{cases}$$

For $j = i + 22,\ i \geq 1$ we have

$$\beta_{j+1} = n2e\left(\frac{\beta_j}{n}\right)^d = n\frac{(2e)^{1+d+\cdots+d^i}}{(4e)^{d^{i+1}}}.$$

It follows that

$$\frac{n}{(4e)^{d^i}} \leq \beta_j = n\frac{(2e)^{1+d+\cdots+d^{i-1}}}{(4e)^{d^i}} = n\frac{(2e)^{\frac{d^i-1}{d-1}}}{(4e)^{d^i}} \leq \frac{n}{2^{d^i}}. \tag{2}$$

Let $m = n/\ell$. Define $\mathcal{E}_i(t, \beta_i) = \{\theta_{\geq i}(t) \leq \beta_i\}$ and let $\mathcal{E}_i = \mathcal{E}_i(m, \beta_i)$ be the event that the number of bins contributing to derived loads $\geq i$ is bounded by β_i throughout the process. From (1), we have that

$$\frac{i\theta_{\geq i}}{2\ell - 1} \leq \lambda(S(i, t)) \leq \frac{n}{\ell},$$

and thus $\mathcal{E}_{22}$ holds with certainty.

Since the d bin choices for a chain are independent we have

$$\mathbf{Pr}\left(h_t \geq i + 1 \mid \theta_{\geq i}(t-1)\right) = \left(\frac{\theta_{\geq i}(t-1)}{n}\right)^d.$$

Let $Y_t = Y(i+1, t)$ be given by

$$Y_t = 1 \iff \left\{h_t \geq i + 1, \quad \theta_{\geq i}(t-1) \leq \beta_i\right\}.$$

Let X_j denote the bin choices available to the j-th chain. Then,

$$\mathbf{Pr}(Y_t = 1 \mid X_1, ..., X_{t-1}) \leq \left(\frac{\beta_i}{n}\right)^d = p_i,$$

We apply Lemma 1 to conclude that

$$\mathbf{Pr}(\sum_{t=1}^{m} Y_t \geq k) \leq \mathbf{Pr}(B(m, p_i) \geq k). \tag{3}$$

Each $Y_t = 1$ event adds at most an extra component of size $2\ell - 1$ to $S(i+1, t)$ so that

$$\theta_{\geq i+1}(m) \leq (2\ell - 1)\sum Y_t. \tag{4}$$

Let $k = emp_i$, then, provided that $\sum Y_t \leq k$, we have

$$\theta_{\geq i+1}(m) \leq 2\ell k = 2\ell emp_i = n2e\left(\frac{\beta_i}{n}\right)^d = \beta_{i+1}.$$

From (3) and (4), we have

$$\mathbf{Pr}(\theta_{\geq i+1}(m) \geq (2\ell - 1)k \mid \mathcal{E}_i) \leq \mathbf{Pr}(\sum Y_t \geq k \mid \mathcal{E}_i) \leq \frac{\mathbf{Pr}(B(m, p_i) \geq k)}{\mathbf{Pr}(\mathcal{E}_i)}.$$

Let ω satisfy

$$\ln\ln\ln(n/\ell) \leq \ln\omega = o\left(\left(\frac{n}{\ell}\right)^{(d-1)/d}\right). \tag{5}$$

Provided that $mp_i \geq 2\ln\omega$, we have

$$\mathbf{Pr}(B(m,p_i) \geq emp_i) \leq e^{-mp_i} = \frac{1}{\omega^2}$$

by the Chernoff Inequality. Since

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

and assuming inductively that $\mathbf{Pr}(\neg\mathcal{E}_i) \leq i/\omega^2$, we have

$$\mathbf{Pr}(\neg\mathcal{E}_{i+1}) \leq \frac{i+1}{\omega^2}.$$

As in [1], choose i^* as the smallest i such that $p_{i+1} = \left(\frac{\beta_i}{n}\right)^d \leq \frac{2\ln\omega}{m}$ and, thus, from (2),

$$i^* = \frac{\ln\ln(m/2\ln\omega)}{\ln d} + O(1).$$

We have that

$$\begin{aligned}\mathbf{Pr}(\theta_{\geq i^*+1}(m) \geq (2\ell)\,6\ln\omega \mid \mathcal{E}_{i^*}) &\leq \mathbf{Pr}(\sum Y_t \geq 6\ln\omega \mid \mathcal{E}_{i^*}) \\ &\leq \frac{\mathbf{Pr}(B(m,(2\ln\omega)/m) \geq 6\ln\omega)}{\mathbf{Pr}(\mathcal{E}_{i^*})} \\ &\leq \frac{1}{\omega^2\mathbf{Pr}(\mathcal{E}_{i^*})},\end{aligned}$$

and, thus,

$$\mathbf{Pr}(\theta_{\geq i^*+1}(m) \geq (2\ell)6\ln\omega) \leq (i^*+1)/\omega^2. \tag{6}$$

Finally, for derived loads of height i^*+2,

$$\mathbf{Pr}(\sum Y_t \geq 1 \mid \theta_{\geq i^*+1}(m) \leq (2\ell)6\ln\omega) \leq \frac{m\left(\frac{(2\ell)6\ln\omega}{n}\right)^d}{\mathbf{Pr}(\theta_{\geq i^*+1} \leq (2\ell)6\ln\omega)} = o(1).$$

Using (5) and (6), the probability of a derived load of height i^*+2 is bounded by

$$\frac{i^*+1}{\omega^2} + o(1) \leq O\left(\frac{1}{\ln\ln(n/\ell)}\right).$$

5 Off-line Allocation for $m \leq n$

In this section we consider off-line allocation of balls into bins. The final allocation of balls into bins is computed by a centralized algorithm once the choices available to each ball are known. The off-line algorithm has to use the random bin choices of the balls. If ball b has chosen bin $b_1, \ldots b_d$, the off-line algorithm has to allocate b into one of these bins.

Let $\lambda(d)$ be the maximum load resulting from the allocation of m balls into n bins with d random choices each. Then, [13], together with a simple majorisation result similar to the one in [1], shows that for $d \geq 2$, $\lceil m/n \rceil \leq \lambda(2) \leq \lceil m/n \rceil + 1$. Our main focus (Section 5.1) here is to find values of d such that $\lambda(d) = \lambda(d,m,n) = \lceil m/n \rceil$. In particular, there are values of d such that, when $m \leq n$, the maximum load is 1. Curiously, we will also show that, once $m > n$, $d = 2$ is best (and also cheapest) although values of $\lambda(d,m,n) = \lceil m/n \rceil$ may still be obtained in certain ranges (see [14]).

The choice of bins available to each ball can be modelled by a hyper-edge of size d on the vertex set $[n]$. Thus the sets of d choices available to the m balls is equivalent to a d-uniform hypergraph $G = (V(G), E(G))$, with n vertices and m edges. In the following we define $c = dm/n$. The optimal allocation and value of $\lambda(d)$ can be found by a reduction to bipartite matching as explained below. We also give a simpler algorithm (for suitably small values of m) based on stripping leaves of the corresponding hypergraph.

Bipartite Matching Algorithm. Let F be a bipartite graph with bipartition (X, Y), where $X = E(G)$, $Y = V(G)$. The edges of F are defined as follows: If $x \in E(G)$ is an edge of G and $y \in V(G)$ is a vertex of x, then there is an edge (x, y) in F. Apply a bipartite matching algorithm to F. If there is X-saturating matching, then the maximum loading of any bin is 1. If not, then remove an X-maximal matching M, deleting the vertices M from X, and repeat the process. If L repeats were made, the maximum loading of any bin is $\lambda(d) = L$. This approach is optimal in the off-line context.

Halls condition for the existence of an X-saturating matching is that $|A| \leq |N(A)|$ for all $A \subseteq X$ and $N(A) \subseteq Y$. Thus the maximum load $\lambda(d)$ can be found directly on the hypergraph G by

$$\lambda(d) = \max_{S \subseteq [n], S \neq \emptyset} \lceil |E[S]|/|S| \rceil,$$

where $E[S]$ are the edges induced by vertex set S.

This approach is similar to the one used in [13], which uses a reduction of the problem (in the case $d = 2$) to network flow problem, a standard method for solving bipartite matching. In the notation of that paper, the authors define

$$L^*_{\max} = \max_{\Delta \subseteq D} \lceil |L_\Delta|/\Delta \rceil,$$

where D is the number of disks (bins), and L_Δ are the edges induced by Δ. This also explains the 'saw-tooth' graph in Figure 1 of [14]. For certain values of $c = 2m/n$ (i.e., N/D, $D = n, N = m$), the edge–vertex ratio of the graph increases to the next integer, and the average maximum load jumps up by 1.

5.1 Maximum Load

We assume that each ball chooses d bins *without replacement*. Allowing d bins selected i.u.r. *with* replacement may increase the maximum load in some cases by one. In particular, when $d = 2$ some vertices may have loops which will definitely increase the maximum load for $m < n/2$ from 1 to 2. The probability that some two balls make the same choice of bins is $O(m/n^{d-1})$. Thus, for $d \geq 3$ we obtain (w.h.p.) a simple d-uniform hypergraph $G = (V(G), E(G)$, chosen uniformly from $G_{n,m,d}$, the space of all simple hypergraphs with $|V(G)| = n$ vertices and $|E(G)| = m$ edges. For $d = 2$, the situation is less clear, as parallel edges occur with constant probability, hence, the result of [13] that shows $\lceil m/n \rceil \leq \lambda(2) \leq \lceil m/n \rceil + 1$. We prove the following theorem.

Theorem 3 *The following results hold w.h.p.*

(i) For $m < n/2$, and $d = 2$ choices, $\lambda(2) = 1$.

(ii) For $m < 0.816 \cdot n$ and $d = 3$ choices, $\lambda(3) = 1$.

(iii) For $m < 0.97677 \cdot n$ and $d = 4$ choices, $\lambda(4) = 1$.

Proof For certain cases which we consider, the maximum load $\lambda(d)$ can w.h.p. be deduced directly from the structure of the hypergraph. Selecting a bin from the given choice of d corresponds to orienting the edge towards that bin (vertex). The occupancy of a bin, is the in-degree of the vertex in the corresponding orientation of the hypergraph G.

If an edge of G is incident with a vertex of degree 1 (a leaf edge), we can orient the edge towards this vertex, as part of any optimal off-line allocation. After all leaf edges have been recursively removed, the remaining graph will either be empty (in which case $\lambda(d) = 1$) or contain a non-empty subgraph $C_2 = (V(C), E(C))$ of minimum degree 2 (the 2-core). Subsequent recursive removal of vertices of degree at most 2 either gives an empty graph or a 3-core C_3 and so on. Properties of cores of random hypergraphs are given in the Appendix.

Let $C = C_2$ denote the 2-core, and suppose the edge density m/n has been chosen so that there is no 3-core w.h.p. In order to make an optimal allocation to the vertices (bins) of C, we must orient the edges to minimize the maximum in-degree. The average in-degree per vertex (occupancy per bin) is $\theta = |E(C)|/|V(C)|$ and thus some vertex must have in-degree at least $\lceil \theta \rceil$. However, provided $\lceil \theta \rceil = 1$ we may still be able to make an allocation $\lambda(d)$ with maximum occupancy 1, depending on the (w.h.p.) structure of the hypergraph.

Result (i). For random graphs ($d = 2$) and for $m < n/2$ the 2-core (if any) consists w.h.p. of $O(\log n)$ vertices on isolated cycles of length at least 3 (see e.g., [4]). There may also be $O(\log n)$ parallel edges disjoint w.h.p. from each other and the cycles above, which we include in the 2-core.

The 2-core is easily dealt with; and the following trivial algorithm finds an optimal allocation with $\lambda(2) = 1$.

- Recursively direct all leaf edges (incident with vertices of degree 1) to the leaf node and remove leaf edges until the graph is empty.
- Orient any remaining cycles so that each vertex on a cycle has in-degree 1.

When $m > n/2$, a 2-core C of order n emerges (see [4]) and $|E(C)|/|V(C)| > 1$. There is no orientation of the 2-core which can achieve maximum occupancy 1.

Result (ii). If we allow $d \geq 3$ choices then there is no 2-core below some critical value $m^*(d)$, whereas above m^* there is a 2-core of order n. Thus the 2-core emerges suddenly around m^*. The threshold value $c^* = dm^*/n$ for the emerging 2-core of d-uniform hypergraph is tabulated below. The value $\widetilde{c}$ given in Row 3 of the table is the value of c for which $\theta = |E(C)|/|V(C)| = 1$ (the last possible value of m for which $\lambda(d) = 1$), and $\widetilde{m}$ is the corresponding value of m.

d	2	3	4	5	6	7
m^*/n	–	0.816	0.775	0.7	0.633	0.571
c^*	–	2.455	3.09	3.51	3.82	4.08
$\widetilde{c}$ at $\theta = 1$	1	2.755	3.91	4.964	5.985	6.994
$\widetilde{m}/n$	0.5	0.9183	0.97677	0.9928	0.9975	0.9991

Thus for $m \leq 0.816n$, a choice of $d = 3$ bins per ball gives maximum allocation $\lambda(3) = 1$. The algorithm to find an optimal assignment is the leaf stripping described above.

Result (iii). Let C be the 2-core of a 4-uniform random hypergraph. Bohman and Kim [3] prove that provided $|E(C)|/|V(C)| < 1$ then w.h.p. the ratio

$$\lambda(4) = \max_{S \subseteq C, S \neq \emptyset} \lceil \tfrac{|E(S)|}{|V(S)|} \rceil$$

is maximized by the 2-core C itself (i.e., C contains no denser subgraph S). Thus, for $d = 4$ and $m < 0.97677n$, $\lambda(4) = 1$ w.h.p. (see the $\widetilde{m}/n$ row in the table above). The Bipartite Matching algorithm will give the optimal allocation. □

We remark that by increasing the value of d an allocation of one ball per bin can probably be obtained for any $m < n$ (see the table above). However we do not prove this here, but rather return to this topic in a later publication.

Provided that there is no 3-core, the stripping algorithm allows an occupancy of at most 2 balls per bin. For there is always a vertex of degree at most 2 remaining during any recursive stripping. However, the threshold value m_3 for the emergence of a 3-core decreases with increasing d.

Theorem 4 *For $m > n$, $d = 2$ gives the best guarantee for not having a 3-core in the sense that $\lambda(2) \leq 2$ for $m < 1.67$.*

Proof As before let $c = dm/n$. Using the results of the appendix we have:

d (choices per ball)	2	3	4	5	6
c at 3-core threshold	3.35	4.66	5.34	5.79	6.13
m_3/n edge-vertex ratio	1.67	1.55	1.33	1.15	1.02

□

The Leaf Stripping algorithm can easily be generalized to deal with k-cores. In that case, all balls directed to a bin with a load of at most k are allocated to the respective bins. If there is no $k+1$-core, an occupancy of at most k is obtained this algorithm. The thresholds for the k-cores are given in the appendix (8). We prove in the extended version of this paper that for $a > 1$, the value $m(a, d)$ for which $\lambda(d) \leq a$ is maximized at $d = 2$.

6 Conclusions and Open Problems

In this paper we analyse the maximum load for the *chains-into-bins* problem where balls are connected in n/ℓ chains of length ℓ. We show that, for $d \geq 2$, the maximum load is $\ln\ln(n/\ell)/\ln d + O(1)$ with probability $1 - O(1/\ln\ln(n/\ell))$. This shows that the maximum load is going down with increasing chain length. We also analyse for which number of random choices d and which number of balls $m < n$, the maximum load of an *off-line assignment* can be upper bounded by one.

Surprisingly, there are many open questions in the area of balls-into-bins games. Only very few results are known for weighted balls-into-bins games, where the balls come with weights and the load of a bin is the sum of the weights of the balls allocated to it. Here, it is even not known if two or more random choices improve the maximum load, compared to the simple process where every ball is allocated to a randomly chosen bin (see [16]). Also, it would be interesting to get tight results for the maximum load and results specifying "worst-case" weight distributions for the balls. Something in the flavor "given that the total weight of the balls is fixed, it is better to allocate lots of small balls, compared to fewer bigger ones." Another interesting problem is to show results relating the maximum load to the order in which the balls are allocated. For example, is it always better to allocate balls in the order of decreasing ball weight, compared to the order of increasing ball weight?

For chains-into-bins problem, a nice open question is to prove Knuth's [10] conjecture stating that breaking chains into two parts only increases the maximum load. This question still open for a single choice and also for several random choices per ball. See [7] for a first progress in this direction. Another question is if similar results to the one we showed in this paper for GREEDY[d] applied to chains also holds for the Always-Go-Left protocol from [17] applied on chains.

References

[1] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, Eli Upfal: Balanced Allocations. *SIAM J. Computing* **29** (1999), pp. 180–200.

[2] Petra Berenbrink, Artur Czumaj, Angelika Steger, Berthold Vöcking: Balanced Allocations: The Heavily Loaded Case. *Proc. of the 30th Annual ACM Symposium on Theory of Computing (STOC 2000)*, pp. 745–754.

[3] T. Bohman, J.H. Kim. A phase transition for avoiding a giant component. *In Random Structures and Algorithms*, pp 195-214 (2005).

[4] Béla Bollobás. *Random Graphs.* Cambridge University Press.

[5] Fabiano C. Botelho and Rasmus Pagh and Nivio Ziviani. Simple and Space-Efficient Minimal Perfect Hash Functions. 10th Workshop on Algorithms and Data Structures (WADS), 2007.

[6] Colin Cooper. *The size of the cores of a random graph with a given degree sequence.* Random Structures and Algorithms, 25: 353-375 (2004).

[7] Matthias Englert: Chains of Length Two into Bins. *Manuscript, University of Aachen.*

[8] Dimitris Fotakis and Rasmus Pagh and Peter Sanders and Paul G. Spirakis. Space Efficient Hash Tables with Worst Case Constant Access Time. Theory of Computing Systems, 38 (2), pp. 229–248, 2005.

[9] Krishnaram Kenthapadi, Rina Panigrahy: Balanced allocation on graphs. *Proc. of 17th Annual Symposium on Discrete Algorithms (SODA 06)*, pp. 434-443.

[10] D.E. Knuth: Sorting and Searching, vol. 3 of "The Art of Computer Programming", Addison-Wesley, 2nd edition, 1998.

[11] Michael Mitzenmacher, Andrea W. Richa, Ramesh Sitaraman: The Power of Two Random Choices: A Survey of Techniques and Results. *Handbook of Randomized Computing*, 2000.

[12] Michael Mitzenmacher, Balaji Prabhakar, Devarat Shah: Load Balancing with Memory. *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pp. 799–808.

[13] Peter Sanders, Sebastian Egner, Jan H. M. Korst: Fast Concurrent Access to Parallel Disks. *Algorithmica* **35** (2003), pp. 21–55.

[14] Peter Sanders: Reconciling simplicity and realism in parallel disk models. *Parallel Computing*, 28(5): pp. 705-723 (2002).

[15] Peter Sanders, Berthold Vcking: Tail Bounds And Expectations For Random Arc Allocation And Applications. *In Combinatorics, Probability & Computing* 12(3), 2003.

[16] Kunal Talwar, Udi Wieder: Balanced allocations: the weighted case. *Proc. of the 39th Symposium on Theory of Computing* (STOC), pp 256-265, 2007.

[17] Berthold Vöcking: How Asymmetry Helps Load Balancing. *Proc. of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999)*, pp. 131-140.

A Properties of Random Hypergraphs

The cores in $G_{n,p,d}$ hypergraphs can be found as follows (see e.g., [6]). The degree sequence is close to Poisson. We use that terminology as shorthand, and speak of Poisson hypergraphs. Let $dm = cn$. Then, the degree distribution of individual vertices (bins) is close to a Poisson distribution with parameter c. This approximation is very good and enables us to estimate the vertices and edges in the cores of $G_{n,p,d}$ up to $An + o(n)$, where An is the core size in the Poisson hypergraph.

For $d \geq 2$ and $k \geq 2$ (and with the exception of the case $d = 2, k = 2$) the threshold for the emergence of the k-core ($k \geq 2$) of $G_{n,p,d}$ is determined by the smallest c for which a solution $\widehat{x}$ in $[0, 1]$ of

$$x^{\frac{1}{d-1}} = f(x) = (d-1)xf'(x) \tag{7}$$

occurs. Here $cn = dm$ and

$$f(x) = 1 - e^{-cx}\left(1 + \cdots + \frac{(cx)^{k-2}}{(k-2)!}\right).$$

Above this threshold, the largest solution $\widehat{x}$ in $[0, 1]$ of

$$x^{\frac{1}{d-1}} = f(x)$$

is used. The expected number of vertices in the k-core is νn where

$$\nu = 1 - e^{-cx}\left(1 + \cdots + \frac{(cx)^{k-1}}{(k-1)!}\right),$$

and the expected number of edges in the k-core is ηn, where

$$\eta = \frac{c}{d}x^{d/(d-1)}.$$

2-cores of $G_{n,p,d}$ hypergraphs. Let $f(x) = 1 - e^{-cx}$. Then, for $d \geq 3$, the threshold for the emergence of the 2-core of $G_{n,p,d}$ is determined by the largest solution $\widehat{x}$ in $[0, 1]$ to

$$x^{\frac{1}{d-1}} = f(x) = (d-1)cxe^{-cx}. \tag{8}$$

3-cores of $G_{n,p,d}$ hypergraphs. For $k = 3$, $f(x) = 1 - e^{-cx}(1 + cx)$ and the threshold for the emergence of the 3-core is

$$x^{\frac{1}{d-1}} = 1 - e^{-cx}(1 + cx) = (d-1)c^2xe^{-cx}.$$