Choice-memory tradeoff in allocations

The balls-and-bins paradigm describes the process where \( b \) balls are placed independently and uniformly at random in \( n \) bins. Many variants of this classical occupancy problem were intensively studied, having a wide range of applications in Computer Science.

It is well-known that when \( b = \lambda n \) for \( \lambda \) fixed and \( n \to \infty \), the load of each bin tends to Poisson with mean \( \lambda \) and the bins are asymptotically independent. In particular, for \( b = n \), the typical number of empty bins at the end of the process is \( (1/e + o(1))n \). The maximal load in that case is \( \text{whp} \left(1 + o(1)\right) \frac{\log n}{\log \log n} \) (the notation \( \text{whp} \) denotes a probability that tends to 1 as \( n \to \infty \)).

The extensive study of this model in the context of Load Balancing was pioneered by the celebrated paper of Azar, Broder, Karlin and Upfal (1999) that analyzed the effect of a choice between \( k \) independent uniform bins on the maximal load, in an online allocation of \( n \) balls to \( n \) bins. It was shown in Azar et al that the Greedy algorithm (choose the least loaded bin of the \( k \)) is optimal and achieves a maximal-load of \( \log_k \log n \) \( \text{whp} \), compared to a load of \( \frac{\log n}{\log \log n} \) for the original case \( k = 1 \). Thus, \( k = 2 \) random choices already significantly reduce the maximal load, and as \( k \) further increases, the maximal load drops until it becomes constant at \( k = \Omega(\log n) \).

In the context of online bipartite matchings (e.g., Hashing), the process of dynamically matching each client in a group \( A \) of size \( n/2 \) with one of \( k \) independent uniform resources in a group \( B \) of size \( n \) precisely corresponds to the above generalization of the balls-and-bins paradigm: Each ball has \( k \) options for a bin, and is assigned to one of them by an online algorithm that should avoid collisions (no two balls can share a bin). It is well known that the threshold for achieving a perfect matching in this case is \( k = \log_2 n \): For \( k \geq (1 + \varepsilon) \log_2 n \), \( \text{whp} \) every client can be exclusively matched to a target resource, and if \( k \leq (1 - \varepsilon) \log_2 n \) then \( \Omega(n) \) requests cannot be satisfied.

In this work, we study the above models in the presence of a constraint on the memory that the online algorithm has at its disposal. For example, in Hashing, access to the hash table may be significantly slower than to the internal memory of the algorithm (e.g., cache as opposed to disk), while this internal memory is considerably smaller.

We find that a tradeoff between the choice and the memory governs the ability to achieve a perfect allocation as well as a constant maximal load. Surprisingly, the threshold separating the subcritical regime from the supercritical regime takes a simple form, in terms of the product of the number of choices \( k \), and the size of the memory in bits \( m \):

- If \( km \gg n \) then one can allocate \((1 - \varepsilon)n\) balls in \( n \) bins without any collisions \( \text{whp} \), and consequently achieve a load of 2 for \( n \) balls.
- If \( km \ll n \) then any algorithm for allocating \( \varepsilon n \) balls \( \text{whp} \) creates \( \Omega(n) \) collisions and an unbounded maximal load.

Roughly put, when \( km \gg n \) the amount of choice and memory at hand suffices to guarantee an essentially best-possible performance. On the other hand, when \( km \ll n \), the memory is too limited to enable the algorithm to make use of the extra choice it has, and no substantial improvement can be gained over the case \( k = 1 \), where no choice is offered whatsoever.

Note that rigorous lower bounds for space, specifically tradeoffs between space and performance (time, communication, etc.), have been studied intensively in the literature of Algorithm Analysis, and are usually highly nontrivial.
Our first main result establishes the exact threshold of the choice-memory tradeoff for achieving a constant maximal-load. As mentioned above, one can verify that when there is unlimited memory, the maximal load is $\text{whp}$ uniformly bounded iff $k = \Omega(\log n)$. Thus, assuming that $k = \Omega(\log n)$ is a prerequisite for discussing the effect of limited memory on this threshold.

**Theorem 1.** Consider $n$ balls and $n$ bins: Each ball has $k = \Omega(\log n)$ uniform choices for bins, and $m = \Omega(\log^2 n)$ bits of memory are available. If $km = \Omega(n)$, one can achieve an $O(1)$ maximal-load $\text{whp}$. Conversely, if $km = o(n)$, any algorithm $\text{whp}$ creates a load that exceeds any constant.

Consider the case $k = \Theta(\log n)$. The naïve algorithm for achieving a constant maximal-load in this setting requires roughly $n$ bits of memory ($2^n$ bits of memory always suffice). Surprisingly, the above theorem implies that $O(n/\log n)$ bits already suffice, and this is tight.

Again consider the case of $k = \Theta(\log n)$, where an online algorithm with unlimited memory can achieve an $O(1)$ load $\text{whp}$. While the above theorem settles the memory threshold for achieving a constant load in this case, one can ask what the optimal maximal load would be below the threshold. This is answered by the next theorem, which shows that in this case, e.g., $m = n^{1-\delta}$ bits of memory yield no significant improvement over an algorithm which makes random allocations.

**Theorem 2.** Consider $n/k$ balls and $n$ bins, where each ball has $k$ uniform choices for bins, and $m \geq \log n$ bits of memory are available. Then for any algorithm, the maximal load is at least $(1 + o(1))\frac{\log(n/m)}{\log\log(n/m) + \log k}$ $\text{whp}$. In particular, if $m = n^{1-\delta}$ for $\delta > 0$ fixed and $2 \leq k \leq \text{polylog}(n)$, then the maximal load is $\Theta(\frac{\log n}{\log\log n})$ $\text{whp}$.

Recall that a load of order $\log n / \log\log n$ is what one would obtain using a random allocation of $n$ balls in $n$ bins. The above theorem states that, when $m = n^{1-\delta}$ and $k \leq \text{polylog}(n)$, any algorithm would create such a load already after $n/k$ rounds. This improves by a factor of 2 a recent lower bound of Benjamini and Makarychev, who considered the case $k = 2$ and obtained independently a similar bound for this special case. Our result also extends this bound to any $k \leq \text{polylog}(n)$.

We further establish the threshold for achieving a perfect matching when allocating $(1-\delta)n$ balls in $n$ bins for $0 < \delta < 1$ fixed. The upper and lower bounds obtained for this threshold are tight up to a multiplicative constant, and again pinpoint its location at $km = \Theta(n)$. Hence, for any value of $k$, the online allocation algorithm we present is optimal with respect to its memory requirements. Finally, we analyze non-adaptive allocation algorithms and give tight upper and lower bounds for their performance.

The key argument in the lower bounds is analyzing the expected number of new collisions that a given step introduces. We wish to estimate this value with an error probability smaller than $2^{-m}$, so it would hold $\text{whp}$ for all $2^m$ possible allocation strategies for this step. To this end, we introduce a Bernstein-Kolmogorov type large deviation inequality, and apply it as part of a delicate martingale analysis. This inequality relates the sum of a sequence of random variables to the sum of its conditional expectations, and crucially does not involve the length of the sequence. We believe that this inequality may have other applications in Combinatorics and the Analysis of Algorithms.

For the upper bounds, our online algorithm essentially partitions the bins into blocks, where for different blocks it maintains an accounting of the occupied bins with varying resolution. Once a block exceeds a certain threshold of occupied bins, it is discarded and a new block takes its place.