Maximizing the Throughput of Cuckoo Hashing in Network Devices

Yossi Kanizo, David Hay, and Isaac Keslassy

***Abstract*—Hash tables form a core component of network devices. Because of their large size, they are implemented using both fast on-chip SRAM and slow off-chip DRAM. However, this makes their implementation particularly delicate, as a suboptimal choice of the hashing scheme parameters may result in a higher average query time, and therefore in a lower throughput. Since hash tables are often on the critical packet path, this can therefore affect the throughput of the whole device.**

In this paper, we analyze the performance of network hash tables when using cuckoo hashing. First, we provide a complete tradeoff between the load of a hash-table and its average lookup time. The problem is solved by analyzing an equivalent problem: the expected maximum matching size of a random bipartite graph with a fixed left-side vertex degree. We provide exact results for any finite system, and also deduce asymptotic results as the SRAM memory size increases. In addition, we further consider other variants of this problem and model the impact of several parameters. Finally, we evaluate the performance of our models on Internet backbone traces, and illustrate the impact of the SRAM/DRAM access time ratio on the parameter choice. In particular, we show that the common intuition of avoiding DRAM accesses by using highly efficient schemes is not always correct. Sometimes, it is better to use a less efficient hashing method because it needs less SRAM accesses.

I. Introduction

A. Background

Network devices increasingly rely on hash tables to efficiently implement their algorithms, in fields as diverse as load-balancing, peer-to-peer, state management, monitoring, caching, routing, URL filtering, and security [1]–[5].

As a result, the device designers often implement a standard hash table structure that is re-used in several of those applications. Unfortunately, due to stringent *memory size constraints* in network devices, it is often impossible to fit the whole hash table within on-chip SRAM. Remaining elements are stored in off-chip DRAM, which is slower, but can also hold more elements [6]–[10]. This is illustrated in Figure 1.

In this paper, we are interested in *designing high-throughput schemes for network hash tables*. Unlike typical hash tables, network hash tables have two specificities. First, they are rebuilt infrequently. For all practical purposes, we can assume that they are built offline. Second, they require elements with query/modify requests to be processed *extremely fast*, using a small and bounded number of memory accesses. For example, a network hash table may store the states of a given number of flows, or the bills of a given number of customers. The set of flows or customers in the hash table is assumed to be predetermined. However, at each new packet arrival, the hash table needs to be accessed immediately.

Y. Kanizo is with the Dept. of Computer Science, Technion, Haifa, Israel. Email: ykanizo@cs.technion.ac.il.

D. Hay is with the School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel. Email: dhay@cs.huji.ac.il

I. Keslassy is with the Dept. of Electrical Engineering, Technion, Haifa, Israel. Email: isaac@ee.technion.ac.il.

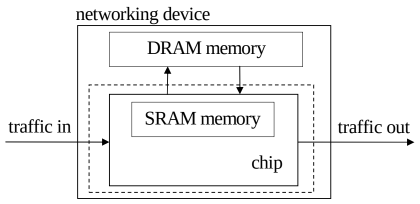


Fig. 1. Typical memory layout of a network device. The dashed rectangle delimits the chip's area. Most of the hash table is generally stored in the on-chip memory, while the remaining elements are stored in the slower off-chip memory.

Designing high-throughput schemes for network hash tables is non-trivial. For instance, assume that a network hash table needs to support n elements using an SRAM size of m bins. Further assume that it relies on multiple-choice hashing, such that each of the n elements can hash into d arbitrary bins using independent hash functions. Then the network hash table designer faces several fundamental tradeoffs. For example, if d is too small, i.e. each element can only hash into a few bins, the hashing scheme may not be efficient. Therefore, more elements may need to be placed in the slow DRAM. On the other hand, if d is too large, each element may take too long to check in which bin it actually resides out of the d potential SRAM bins.

As a result, incorrect hash-table settings can significantly *increase the average delay* needed to deal with each packet. Therefore, when incoming packets are processed sequentially, incorrect settings can also also significantly *decrease the throughput* of the network hash table. The goal of this paper is to help designers better understand the tradeoffs involved in the choice of the hashing parameters.

In this paper, we rely on *cuckoo hashing* to implement the network hash table. Cuckoo hashing guarantees that the element query/modify times are bounded and achieves an efficient space utilization (e.g., [6]–[9], [11]–[19] and references therein). When building the hash table offline in cuckoo hashing, each new element is placed in one of its d bins. If all bins are full, it displaces another element, which is then moved to one of its other $d-1$ bins. If all its $d-1$ other bins are full, it displaces yet another element, and so on. This process continues until either all elements are placed, or the process is stopped and the new element is placed in the DRAM (which is assumed infinite for simplicity).

Cuckoo hashing with $d = 2$ is especially interesting because of its high memory utilization. In fact, consider the bipartite graph formed by the n elements on one side, the m bins on the other, and $d = 2$ links leaving each element for 2 bins according to the hash values of the element. Then the number of elements that cuckoo hashing inserts successfully *is exactly the size of the maximum matching* [15]. Therefore, our results with $d = 2$ also provide an upper bound on the number of elements that can be inserted into the SRAM by *any alternative multiple-choice scheme with* $d = 2$. To further understand why cuckoo hashing has a high utilization, it is important to notice that *online* cuckoo hashing with $d = 2$, succeeds in inserting an element if and only if an augmenting path originating from the corresponding vertex exists [15]. This is because inserting an element into a cuckoo hash table is equivalent to finding an augmenting path in the corresponding graph (that is, a path that starts from the vertex corresponding to the considered element, and alternates between unmatched and matched edges until it ends at a right-side vertex all of whose edges are unmatched). Notice that for any sub-path (r_1, v, r_2), where v is a left-side vertex and r_1, r_2 are right-side vertices, (r_1, v) must be a matched edge and (v, r_2) an unmatched edge. Intuitively, this corresponds to moving element v from bin r_1 to bin r_2. Since maximum size matching can be computed by finding such augmenting paths, when considering each left-side vertex only once and in arbitrary order, we can immediately conclude that the number of elements that a cuckoo hashing inserts successfully *is exactly the size of the maximum matching*. For example, all n elements can be inserted if and only if the corresponding graph has a *perfect matching* (namely, a maximum matching of size n; see [15] for more details).

B. Contributions

Our first contribution is that we study the performance of cuckoo hashing with $d = 2$ *over all possible loads.* Past papers have focused on lower loads, and shown that up to a load $n/m = 0.5$, all elements could fit in the hash table with high probability [11], [13]–[15]. On the contrary, we also analyze the behavior of cuckoo hashing as the load gets beyond 0.5. To do so, we essentially transform the problem into a problem in graph theory, then provide a theoretical analysis of its performance, and later evaluate its real-life behavior by using Internet backbone traces.

Specifically, we study the *average performance* of cuckoo hashing by analyzing the *expected maximum matching size* in the bipartite graph introduced above. We decompose each random bipartite graph into *connected components*, and then separately analyze each component and evaluate the size of its local maximum bipartite match. The size of the maximum bipartite matching is the sum of the sizes of all local matches. Then, we count the number of connected components in the graph and thus derive the size of the maximum matching in the entire graph. Surprisingly, we can obtain an *exact expression* of the average performance of cuckoo hashing in any finite system.

We further show that the actual maximum matching size is sharply concentrated around its expected value. Thus, the difference between n and the expected maximum matching size provides, with high probability, *the number of elements that should be stored in the off-chip DRAM*.

Second, we provide an exact analysis of a common cuckoo hashing implementation in which the memory is (statically) partitioned into two segments, such that each segment corresponds to the image of one hash function; this implementation is particularly attractive when using *single-ported* memories.

Third, we further present exact analysis when, in order to minimize the number of memory accesses, the average number of choices is less than 2.

Fourth, we also obtain a lower bound on the required DRAM size when the number of hashes d exceeds 2.

We further evaluate our results on real-life Internet packet traces from an OC192 backbone link, using a real-life 64-bit mix hash function. We show when the load is 1, i.e. $n = m$, we can insert an average of 83.81% of the packets within the hash table. Likewise, when the load is 0.6, i.e. $n = 0.6m$, we can insert in average 99.38% of the packets, thus only storing 0.62% on off-chip DRAM. We further confirm our analytical models and show that our bounds for $d > 2$ are typically within 1% of the exact value.

Finally, we compare the network hash table throughput using different numbers d of hash functions. We first provide analytical results when the on-chip memory is partitioned into two (unequal) segments. Then, we run simulations and show that, unlike common belief, it is still worth using $d = 2$ hash functions beyond a load of 0.5, even though some of the packets are stored on DRAM. We also illustrate how the exact load at which cuckoo hashing with $d = 3$ hash functions outperforms traditional cuckoo hashing with $d = 2$ depends on the SRAM/DRAM access time ratio.

Incidentally, our paper can lead to two interesting contributions. First, the paper analysis also provides exact results when the numbers of elements n and buckets m are finite. This non-asymptotic analysis is particularly needed when n and m are known to be small. In addition, we note that for other multiple-choice hashing schemes, our results provide *a lower bound on the number of elements that should be stored off-chip*. This is because the maximum matching size of the graph is always an upper bound on the number of elements that can be inserted into the hash table. Moreover, since finding the maximum matching in bipartite graphs is a fundamental problem with a wide range of applications, we believe that our results also have a theoretical significance and may be used in other contexts.

C. Limits

In our paper, we make several key assumptions that may limit the reach of our results.

First, we rely on *cuckoo hashing* to implement the network hash table. While it is a leading state-of-the-art hashing option, we could have chosen to analyze alternative hashing schemes, which may have provided different results. As mentioned above, please note however that no multiple-choice algorithm with $d = 2$ can beat the SRAM utilization of cuckoo hashing, although it may theoretically yield a better throughput.

Our second assumption is that the hash table is accessed sequentially, such that each new packet needs to wait for the end of the former packet. As a result, throughput is inversely proportional to packet delay. This assumption is designed to cope with general hash tables, in which several applications may share the hash table, and therefore each packet may need to access and modify several elements in the hash table according to different application-based keys. Since the modifications of each packet may also affect the next packets, it is simpler to wait for its processing to end. The hash table may be made more efficient by processing packets of different application-based flows in parallel. But such a scheme may become too hard to implement for a large number of applications, because each key of each packet needs to be compared with the relevant keys of all previous packets currently accessing the hash table. In any case, our results can also be extended to such parallel accesses.

Our third assumption is that all element queries are for elements that are indeed stored in the hash-table. This assumption is common in several networking applications [10], while in others it requires a set membership query before actually accessing the hash-table. To obtain the expected latency, we further assume that all elements in the hash table are equally likely to be accessed.

Finally, our last assumption is that each access to DRAM is b times slower than an access to SRAM, i.e. the impact of DRAM is mainly through its access time. We do not take into account the chip in/out pin capacity, which may further reduce the range of hashing options available. We also do not consider the DRAM division into banks, and do not consider non-uniform DRAM access times as a first approximation.

D. Related Work

There is a large literature on multiple-choice hashing schemes for general hash tables [20], [21]. In particular, regarding the cuckoo hashing scheme [6]–[9], [11]–[19], the main effort has been to find a load threshold, such that for any load below the threshold a perfect matching exists with high probability. It is known that a cuckoo hashing scheme with $d = 2$ succeeds with high probability if the load is less than 0.5, but fails when the load is larger [11]. Recent works [13]–[15] have also settled the problem of finding the corresponding thresholds for $d > 2$. Moreover, [8] shows that cuckoo hashing with a stash (in our case, DRAM) of size s, $d = 2$, and a load less than 0.5 fails with probability $O\left(n^{-s}\right)$. Our paper differs in that we also consider the average efficiency of cuckoo hashing for load values beyond 0.5 for $d = 2$. Moreover, while most of the works investigate only the asymptotic behavior, we also present in our paper analytical expressions for finite random graphs along with the asymptotic ones. Last, we assume that the DRAM size is not a constraint.

We are particularly interested in schemes for network hash tables [10]. In such schemes, the lookup times are often assumed to be bounded. For instance, the multi-level hash table (MHT) [7], [22] scheme partitions the SRAM memory into subtables, with a single hash function per subtable. In addition, the cuckoo variant with one move enables applications that also need fast element insertions, which we do not consider in this paper [6]. Moreover, additional papers consider the problem of off-chip memory. When the SRAM is too small, an on-chip summary of the off-chip elements is used to reduce the average number of off-chip accesses to almost 1 per element query [23]. But note that as a consequence, an off-chip access is performed in any hashing operation. Another non-uniform memory model-based hashing scheme is the peacock hashing, which also stores clues in on-chip memory and improves upon MHT for deletions [24]. Additional references also focus on hash tables for specific applications, such as those based on Bloom filters [5], [10], [25]. However, all these papers do not focus on optimizing parameters in order to reduce the overall latency in a combined SRAM/DRAM system, which is the goal of this paper.

Finally, there are several related results in graph theory. [26], [27] provide the probability of a perfect matching in several random bipartite graph models. But they do not provide the expected maximum matching size when this probability is different from one. Also, several studies investigate the expected maximum matching size in other models of random graphs, and especially trees [28]–[31]. Our paper differs in that it considers a different model of random bipartite graphs, where each left-side vertex chooses a constant number of right-side vertices.

Paper Organization: We start by introducing the preliminary definitions in Section II. Then, Section III provides the expected maximum matching size of random bipartite graphs with left-side vertex degree 2, where a variation of the problem in which each left-side vertex degree is at most 2 is considered in Section IV. Next, in Section V, we solve the more appealing problem in which the right-side vertices are partitioned into two subsets, and each left-side vertex has exactly one edge to each of these subsets. Section VI provides an upper bound on the expected maximum matching size when the constant left-side vertex degree is at least three. Last, in Section VII we verify and evaluate our results, including by real-life trace-based experiments. For the sake of readability, most of the proofs are presented in Appendix A.

II. Bipartite Graph Model

In this section, we define cuckoo hashing using a bipartite graph, with the left-side vertices corresponding to elements and the right-side vertices to SRAM bins.

Formally, given two disjoint sets of vertices L and R of size n and m respectively, we consider a random bipartite graph $G = \langle L + R, E \rangle$, where each vertex $v \in L$ has $d = 2$ outgoing edges whose destinations are chosen independently at random among all vertices in R. We allow both choices to be the same vertex, implying that G might have parallel edges. For brevity, we sometimes say that $v \in L$ *chooses* a vertex $v' \in R$ if (v, v') is in E. The *load* of G is denoted by $\alpha = \frac{n}{m}$.

We further consider cases when the *average number of choices* is less than 2.

Definition 1. *Let d_v be the number of choices of each vertex $v \in L$. The* average number of choices a *is the average left-side vertex degree, i.e.* $a = \frac{\mathbb{E}\left(\sum_{v \in L} d_v\right)}{n} = \frac{\sum_{v \in L} \mathbb{E}(d_v)}{n}$.

First, in the deterministic case, we find the expected maximum matching size of the graph $G_a = \langle L + R, E \rangle$, where each vertex $v \in L$ independently chooses a predetermined number $d_v \in \{1, 2\}$ of random vertices in R, such that $a = \frac{d_1 + 2 \cdot d_2}{n}$.

Second, in the random case, we analyze the slightly different case of a random bipartite graph $G_p = \langle L + R, E \rangle$ where each vertex chooses two vertices with probability p and one vertex with probability $1 - p$. This implies that in G_p, the average number of choices $a = 1 + p$.

Finally, we also consider a *static partitioning of the choices*; the set R is partitioned into two disjoint sets R_u and R_d of sizes $\beta \cdot m$ and $(1 - \beta)\, m$. In that case, we consider a random bipartite graph $G_\beta = \langle L + (R_u \cup R_d), E \rangle$, where each vertex $v \in L$ chooses exactly one vertex in R_u and another vertex in R_d.

We want to find both the *expected maximum matching size* as well as the *normalized limit expected maximum matching size* for the above-mentioned graph models. To do so, we model our hash functions as fully random, which often yields an excellent approximation [10], [32].

Definition 2. *For any graph G, let $\mu(G)$ be the* expected size of the maximum size matching.

Notice that if G is a deterministic graph, then $\mu(G)$ is simply the size of its maximum size matching.

Definition 3. *The* normalized limit expected maximum matching size $\gamma = \lim_{n \to \infty} \frac{\mu(\cdot)}{n}$ *is the limit percentage of the expected maximum matching size (out of the number of the vertices in L).*

Note that often we are interested in $n - \mu(G)$, which corresponds to the expected number of unmatched left-side vertices in the graph. This corresponds to the number of elements that should be stored in the off-chip DRAM.

Finally, our goal is to model the throughput of the network hash table. To do so, we first assume that each access to on-chip SRAM takes a unit amount of time, while each access to off-chip DRAM takes a latency of b (where $b > 1$, e.g. $b = 10$). Also, all accesses are sequential. For instance, assume that we use cuckoo hashing with $d = 2$, and a given element is in the DRAM. Then a query for this element would first successively check the $d = 2$ SRAM bins, then the DRAM, for a total latency of $b + 2$.

We further assume that all the elements in the hash table are equally likely to be accessed. We define the *average latency* as the average total access latency over all elements in the hash table, including the elements in the SRAM as well as in the DRAM. We further define the *throughput* of the network hash table as the inverse of its average latency. For example, if it takes on average 2 accesses to query an element, then the hash table throughput is $\frac{1}{2}$. Our goal is to maximize this throughput.

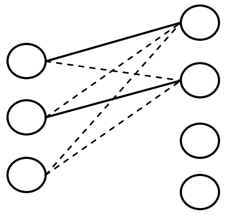


Fig. 2. Example of bipartite graph with left-side vertex degree 2

III. Expected Cuckoo Performance: Bipartite Graphs with $d = 2$

We are now interested in evaluating the expected performance of cuckoo hashing. As explained above, we approach the problem using a graph-theory perspective, since it is the same as evaluating the expected maximum matching size of the random bipartite graph G.

To do so, we consider the connected components of the random bipartite graph G. We start by stating some lemmas on these connected components, before establishing our main result on the expected matching size. Note that further evaluation of the results reported here appears in Section VII.

A. Expected Maximum Matching Size

We first consider an arbitrary bipartite graph $H = \langle L_H + R_H, E_H \rangle$, where each left-side vertex in L_H chooses $d = 2$ right-side vertices in R_H (parallel edges are allowed), with $|L_H| = s$ and $|R_H| = q$.

Figure 2 illustrates such a bipartite graph with $s = 3$, $q = 4$, and left-side vertex degree 2. Dashed lines represent edges not in the maximum size matching, while solid lines represent edges in the maximum size matching.

We start by quoting a few useful and straightforward lemmas (proved in Appendix A), before stating our result.

Lemma 1. *If $s \leq q - 2$, then H is not connected.*

Lemma 2. *If H is connected and $s \geq q$, then $\mu(H) = q$.*

Lemma 3. *If H is connected and $s = q - 1$ then $\mu(H) = s$.*

Lemma 4. *For any graph with $s = q - 1$, H is connected if and only if it is a tree.*

Lemma 5. *The number T_s of* labeled *connected bipartite graphs H whose $|L_H| = s$ and $|R_H| = s + 1$ is $T_s = (s + 1)^{s-1} s!$*

We can now prove the next theorem on our random bipartite graph G, which is the main theoretical result of this paper. This theorem provides the exact expected number of elements $\mu(G)$ that can fit the on-chip SRAM memory using standard cuckoo hashing with $d = 2$. Therefore, a corollary is that $n - \mu(G)$ also gives us the *expected number of elements that are left outside the chip*.

Theorem 1. *Let $d = 2$ and $\ell = \min(n, m - 1)$. The expected* *maximum matching size $\mu(G)$ is*

$$\mu(G) = m - \sum_{s=0}^{\ell} \left\{ \binom{n}{s} \cdot \binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(n-s)} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot \frac{2^s s!}{(s+1)^{s+1}} \right\}$$

Proof: Let M be a maximum matching of G. Our proof is based on counting the expected number of vertices in R that are not part of M, and on the decomposition of G into its connected components.

Lemma 1 yields that any connected component of G with s left-side vertices has at most $s+1$ right-side vertices. We call a connected component with s left-side vertices and $s+1$ right-side vertices *a deficit component of size* s. Lemma 3 implies that the maximum matching size of any such deficit component is s. Therefore, *exactly one* of its right-side vertices is not part of M. Notice that in all other connected components, where $q < s+1$, the maximum matching size of G is exactly q (Lemma 2), implying that all their right-side vertices are part of M.

Thus, in order to calculate the size of M, it suffices to count *the number of deficit components* x. The size of M is $m - x$ because exactly x right-side vertices do not participate in M, one for each deficit component.

Consider a random bipartite graph, with s left vertices, each of degree 2, and $s+1$ right vertices, and let $P_s = \frac{2^s T_s}{(s+1)^{2s}}$ be the probability that it is connected. Note that we multiply T_s by 2^s because T_s only counts connected bipartite graphs, which are necessarily trees (Lemma 4), with no distinction between the two edges connected to each left vertex, while in the denumerator we count all possible instances of random bipartite graphs as above, where we distinct between the two edges connected to each left vertex.

The expected number of deficit components of size s is $\binom{n}{s}\binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(n-s)} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot P_s$. The above expression consists of the following factors (in order):
(i) choosing the s vertices in L;
(ii) choosing the $s+1$ vertices in R;
(iii) the probability that all $s+1$ vertices in R may be connected only to the chosen s vertices in L;
(iv) the probability that all s vertices in L are only connected to the $s+1$ vertices in the right side; and,
(v) the probability that all chosen vertices are connected.

Finally, we calculate x by summing over all possible values on s. As mentioned before, the expected size of M is given by $m - x$. We get: $\mu(G) = m - \sum_{s=0}^{\ell} \binom{n}{s}\binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(n-s)} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot P_s$, where $\ell = \min(n, m-1)$, $P_s = \frac{2^s T_s}{(s+1)^{2s}}$, and $T_s = (s+1)^{s-2} \cdot (s+1)!$, as found in Lemma 5. ■

The following example provides a simple illustration when $n = m = 2$.

Example 1. *Consider the case $n = m = 2$ (and $d = 2$). Then in all random graphs the maximum matching size is 2, except for the two extreme cases where all 4 edges are connected to a specific vertex in R, and then the maximum matching size is 1. Each such case occurs with probability $\left(\frac{1}{2}\right)^4$. Hence, $\mu(G) = 2 - 2 \cdot \left(\frac{1}{2}\right)^4 = \frac{15}{8} = 1.875$. This is indeed precisely what we obtain using Theorem 1.*

B. Concentration Result

We next show that *the size of the maximum matching is highly concentrated around its expectation $\mu(G)$*. This implies that the number of off-chip elements will be close to its average value.

In order to prove this result, we apply Azuma's inequality to a Doob martingale (more specifically, the martingale is a vertex exposure martingale of the left-side vertices). Note that as long as all left-side vertices pick their edges independently, this concentration result holds regardless of the value of d, and more generally regardless of the specific distribution over which the hash functions are defined. Therefore, the concentration result also applies to the settings of Sections IV – VI.

Theorem 2. *Let H be a specific instance of the random graph G, as defined in Section II. For any $\lambda > 0$, $\Pr(|\mu(H) - \mu(G)| > \lambda\sqrt{n}) < 2e^{-\lambda^2/2}$.*

Proof: Our notations follow those of [33]. We first define an exposure martingale, which exposes one left-side vertex at a time, along with all its outgoing edges. This martingale is equivalent to a regular vertex exposure martingale, in which all right-side vertices are exposed first, and then left-side vertices are exposed one by one.

Specifically, let G be the probability space of all two-choice bipartite graphs as defined in Section II and f the size of the maximum size matching of a specific instance. Assume an arbitrary order of the left-side vertices $L = \{v_1, \ldots v_n\}$, and define $X_0, \ldots, X_n$ by $X_i(H) = E[f(G) \mid \forall x \le i, \forall v_y \in R, (v_x, v_y) \in G \text{ iff } (v_x, v_y) \in H]$. Note that $X_0(H) = \mu(G)$ since no edges were exposed, while $X_n(H) = \mu(H)$ as all edges are exposed.

Clearly, f satisfies the vertex Lipschitz condition since if two graphs H and H' differ at only one left-side vertex, $|f(H) - f(H')| \le 1$ (either that vertex is in the maximum matching or not). Thus, since each left-side vertex makes independent choices, [33, Theorem 7.2.3] implies that the corresponding vertex exposure martingale satisfies $|X_{i+1} - X_i| \le 1$. Hence, by applying Azuma's inequality, we immediately get the concentration result. ■

Notice that if we are interested only in one-sided bounds, we can get a slightly tighter result: $\Pr(\mu(G) - \mu(H) > \lambda\sqrt{n}) < e^{-\lambda^2/2}$. This is exploited in the following corollary, which shows that to obtain a given overflow fraction, the number of off-chip elements grows sub-linearly with n beyond its average value.

Corollary 3. *With probability at least $1 - \epsilon$, the number of elements that need to be stored in off-chip DRAM is less than $n - \mu(G) + \sqrt{2n \cdot \ln(1/\epsilon)}$, where $\mu(G)$ is as in Theorem 1.*

Proof: If a stash of size $n - \mu(G) + \sqrt{2n \cdot \ln 1/\epsilon}$ is used, cuckoo hashing fails if and only if $n - \mu(H) > n - \mu(G) + \sqrt{2n \cdot \ln 1/\epsilon}$, or by rewriting it, $\mu(G) - \mu(H) > \sqrt{2n \cdot \ln 1/\epsilon}$. By substituting $\lambda = \sqrt{2 \cdot \ln 1/\epsilon}$ in the above one-sided bound, we get the claimed result. ■

C. Limit Normalized Expected Maximum Matching Size

Our results above provide exact expressions, given n elements and m SRAM bins. We now want to study the scaling properties of the hash table, and are interested in the *asymptotic* expression where $n \to \infty$ with $\alpha = \frac{n}{m}$ constant.

The following results show an interesting connection between the limit normalized expected maximum matching size and the Lambert-W function, and even a connection between the perfect matching threshold and the radius of convergence of the Lambert-W function. For further details on the Lambert-W function, see also Appendix B.

Theorem 4. *Let $d = 2$. The limit normalized expected maximum matching size $\gamma = \lim_{n\to\infty} \frac{\mu(G)}{n}$ is given by:*

$$\gamma = \frac{1}{\alpha} + \frac{1}{2\alpha^2} \cdot W\left(-2\alpha \cdot e^{-2\alpha}\right) + \frac{1}{4\alpha^2} W^2\left(-2\alpha \cdot e^{-2\alpha}\right), \quad (1)$$

where the Lambert-W function is the inverse function of the function $\omega(x) = xe^x$.

Proof: We compute the limit of $\frac{\mu(G)}{n}$ as $n \to \infty$ such that $\alpha = \frac{n}{m}$:

$$\gamma = \lim_{n\to\infty} \frac{1}{n} \cdot \left(m - \sum_{s=0}^{\ell} \left\{ \binom{n}{s} \cdot \binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(n-s)} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot \frac{2^s s!}{(s+1)^{s+1}} \right\} \right)$$

We find through differentiation that $\left(1 - \frac{s+1}{m}\right)^{2(n-s)}$ is an increasing function with respect to n (where $m = \frac{n}{\alpha}$). Moreover, the expansion of $\frac{1}{n} \cdot \binom{n}{s}\binom{m}{s+1} \cdot \left(\frac{s+1}{m}\right)^{2s}$ shows that it is also an increasing function. Therefore, their product is also increasing and, by the monotone convergence theorem [34], we get

$$\gamma = \lim_{n\to\infty} \frac{m}{n} - \sum_{s=0}^{\infty} \lim_{n\to\infty} \left(\frac{1}{n} \cdot \binom{n}{s}\binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(n-s)} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot P_s \right)$$

where by convention $\binom{u}{v} = 0$ for $u < v$. By substituting the expression for P_s, and using the facts that $\binom{n}{s} = \frac{n^s}{s!} + O\left(n^{s-1}\right)$ and $\lim_{n\to\infty} (1 + a/n)^n = e^a$, we deduce:

$$\gamma = \lim_{n\to\infty} \frac{m}{n} - \sum_{s=0}^{\infty} \lim_{n\to\infty} \left(\frac{1}{n} \frac{n^s}{s!} \frac{m^{s+1}}{(s+1)!} e^{-2\alpha(s+1)} \cdot \frac{(s+1)^{2s}}{m^{2s}} \cdot \frac{2^s (s+1)^{s-1} s!}{(s+1)^{2s}} \right)$$

By substituting $m = \frac{n}{\alpha}$, and simplifying the above expression, we get:

$$\begin{aligned} \gamma &= \frac{1}{\alpha} - \frac{1}{\alpha} \cdot \sum_{s=0}^{\infty} \alpha^s \cdot 2^s \cdot \frac{(s+1)^{s-1}}{(s+1)!} \cdot e^{-2\alpha(s+1)} \\ &= \frac{1}{\alpha} - \frac{1}{2\alpha^2} \cdot \sum_{j=1}^{\infty} \left(-2\alpha \cdot e^{-2\alpha}\right)^j \cdot \frac{(-j)^{j-2}}{j!} \end{aligned}$$

Let $T(x) = \sum_{j=1}^{\infty} \frac{(-j)^{j-2}}{j!} \cdot x^j$ be a formal power series, where by substituting $x = -2\alpha \cdot e^{-2\alpha}$ we get the above expression. By differentiating $T(x)$ and multiplying by x, we get:

$$x \cdot \frac{d}{dx} T(x) = -\sum_{j=1}^{\infty} \frac{(-j)^{j-1}}{j!} \cdot x^j = -W(x),$$

where the Lambert-W function is the inverse function of the function $\omega(x) = xe^x$ [35], and the last equality follows from its known Taylor expansion that converges as long as x is within the radius of convergence with $|x| \leq e^{-1}$ [35].

Given that $x \cdot \frac{d}{dx} T(x) = -W(x)$, we compute $T(x)$:

$$T(x) = \int \frac{1}{x} \cdot (-W(x))\, dx = -W(x) - \frac{1}{2} W^2(x),$$

with convergence within $|x| \leq e^{-1}$.

Interestingly, the function $f(\alpha) = -2\alpha \cdot e^{-2\alpha}$ gets its minimum at $\alpha = 0.5$, where it precisely equals the radius of convergence $-e^{-1}$. Therefore, for all α we can substitute $x = -2\alpha \cdot e^{-2\alpha}$, since we are within the radius of convergence of $T(x)$, and we finally derive the result. ■

We note that this particular asymptotic result can be also achieved by the theory of giant components in random graphs [33], [36]. However, this technique is not applicable for finite n and m, and cannot be used to derive most of the other results in this paper. (A proof using this technique appears in Appendix A).

The following simple illustration of the result shows that standard cuckoo hashing can only reach about 84% of SRAM occupancy when the load is 1.

Example 2. *In case $\alpha = 1$, that is $n = m$, the normalized limit expected maximum matching size is*

$$\gamma = 1 + \frac{1}{2} \cdot W\left(-2 \cdot e^{-2}\right) + \frac{1}{4} W^2\left(-2 \cdot e^{-2}\right) \approx 0.8381.$$

The following corollary shows that when the load is below $\frac{1}{2}$, the probability for a right-side vertex to be part of a maximum matching goes to 1. This corollary also follows from the previously known result that there is a perfect matching with high probability in cuckoo hash tables with load $\alpha \leq \frac{1}{2}$ [11].

Corollary 5. *Let $d = 2$ and $\alpha = \frac{n}{m} \leq \frac{1}{2}$. Then the limit normalized expected maximum matching size is $\gamma = \lim_{n\to\infty} \frac{\mu(G)}{n} = 1$.*

Proof: In case $\alpha \leq \frac{1}{2}$, $W\left(-2\alpha \cdot e^{-2\alpha}\right)$ equals -2α, thus, $\gamma = \frac{1}{\alpha} + \frac{1}{2\alpha^2} \cdot (-2\alpha) + \frac{1}{4\alpha^2} (-2\alpha)^2 = 1$ ■

IV. Cuckoo with Low Memory Bandwidth: Bipartite Graphs With $d_v \leq 2$

In this section we are interested in a *low-memory-bandwidth version* of the cuckoo hash algorithm. We now let each element choose either 1 or 2 bins instead of only 2 bins, to force them to access less bins and use less memory I/O bandwidth.

The idea behind this cuckoo algorithm is that it may use less SRAM accesses than a full cuckoo algorithm. On the other hand, it will be less memory-efficient and therefore will also need to access the DRAM more often. We are interested in the tradeoff between these two considerations.

Formally, we relax the constraint that each vertex in L chooses exactly 2 vertices in R, and let each left-side vertex choose either 1 or 2 right-side vertices. Since we can divide the set of vertices either deterministically or randomly, we will discuss the results in both cases. See also [37] for a similar model.

Note that further evaluation of the results reported in this section can be found in Section VII-B.

A. Connected Components in Deterministic Graphs

As in Section III-A, we now consider a deterministic bipartite graph $H = \langle L_H + R_H, E_H \rangle$, with $|L_H| = s$ and $|R_H| = q$. We assume that the degree of each vertex in L_H is at most 2.

Proposition 1. *Lemmas 1, 2, and 3 hold also when the degree of each vertex in L_H is at most (but not necessarily) 2.*

Note that the proofs remain almost identical to the original proofs, replacing a few equalities with the corresponding inequalities.

Lemma 6. *Let $s + 1 = q$. If H is connected then the degree of each vertex in L_H is 2.*

B. Expected Maximum Matching Size

Predetermined Number of Choices—We assume that each vertex $v \in L$ independently chooses $1 \le d_v \le 2$ random vertices in R, where d_v is predetermined. The following result provides the expected maximum matching size in this case.

Theorem 6. *Given a predetermined average number of choices a, let $d_1 = (2 - a)\cdot n$ and $d_2 = n - d_1 = (a - 1)\cdot n$ be the number of vertices in L that choose one and two vertices in R, respectively. The expected maximum matching size $\mu(G_a)$ is given by:*

$$\mu(G_a) = m - \sum_{s=0}^{\ell} \left\{ \binom{d_2}{s} \cdot \binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(d_2 - s) + d_1} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot \frac{2^s s!}{(s+1)^{s+1}} \right\}$$

where $\ell = \min(d_2, m - 1)$.

Random Number of Choices—We assume that each vertex $v \in L$ independently chooses $1 \le d_v \le 2$ random vertices in R, where for each $v \in L$, d_v equals 2 with probability p, and it equals 1 with probability $1-p$. The following result reflects the expected maximum matching size in this case.

Theorem 7. *The expected maximum matching size $\mu(G_p)$ is given by*

$$\mu(G_p) = \sum_{d_2=0}^{n} \binom{n}{d_2} \cdot p^{d_2} \cdot (1-p)^{n-d_2} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right),$$

where $\mu(G_a)$ is given by Theorem 6.

C. Limit Normalized Expected Maximum Matching Size

Predetermined Number of Choices—We are also interested in the asymptotic expression, where $n \to \infty$, such that we fix both the load $\alpha = \frac{n}{m}$ and the average number of choices $a = \frac{d_1 + 2\cdot d_2}{n}$ of the vertices. This is reflected in the following theorem.

Theorem 8. *The limit normalized expected maximum matching size $\gamma_a = \lim_{n\to\infty} \frac{\mu(G_a)}{n}$ with average number of choices $a \in (1, 2]$ is given by:*

$$\gamma_a = \frac{1}{\alpha} + \frac{W\left(-2\alpha(a-1)\cdot e^{-a\alpha}\right)}{2\alpha^2\cdot(a-1)} + \frac{W^2\left(-2\alpha(a-1)\cdot e^{-a\alpha}\right)}{4\alpha^2\cdot(a-1)}.$$

For $a = 1$, it is $\gamma_a = \frac{1}{\alpha} - \frac{1}{\alpha}\cdot e^{-\alpha}$.

Interestingly, if even a small fraction of the elements do not have choice, then the limit normalized expected maximum matching size is not 1. This is reflected in the following corollary.

Corollary 9 ((No) Perfect Matching)**.** *If $1 \le a < 2$ then $\gamma_a < 1$.*

Random Number of Choices—We now study the case of the random bipartite graph $G_p = \langle L + R, E \rangle$, where each vertex chooses two vertices with probability p, and a single vertex with probability $1-p$. As we show in the next theorem, the asymptotic expression can be derived from γ_a.

Theorem 10. *The limit expected maximum matching size $\gamma_p = \lim_{n\to\infty} \frac{\mu(G_p)}{n}$ is $\gamma_p = \gamma_{a=1+p}$.*

V. Single-Ported Cuckoo: Static Partitioning of the Choices

We now consider a popular cuckoo-hashing implementation variant in which the bins are statically partitioned into two equal sets, and each element holds one hash function to each set. This variant is *easier to implement in hardware*, because it can be implemented using two simple *single-ported memories*, instead of a single dual-ported one.

Formally, we consider the random bipartite graph $G_\beta = \langle L + (R_u \cup R_d), E \rangle$, where R is now partitioned into two disjoint subsets R_u and R_d with $|R_u| = \beta \cdot m$ and $|R_d| = (1 - \beta)\, m$. Each vertex $v \in L$ independently chooses a single random vertex in R_u and another single random vertex in R_d. This corresponds, for example, to a hashing scheme that selects non-overlapping sets of bins as images of its hash functions (e.g., as in multilevel hashing scheme [22] or d-left [2]).

Note that further evaluation of the results reported in this section can be found in Section VII-C.

A. Connected Components in Deterministic Graphs

The following lemma counts all the possible bipartite graphs H_{ud} of the form $\langle L_H + (R_{H_u} \cup R_{H_d}), E_H \rangle$ with degree 2 for each vertex in L_H, where $|L_H| = s$, $|R_{H_u}| = i$ and $|R_{H_d}| = j$, such that each vertex $v \in L_H$ is connected using a single edge to some vertex in R_{H_u} and another single edge to some vertex in R_{H_d}.

Proposition 2. *Lemmas 1, 2, 3, and 4 hold for this case as well.*

Lemma 7. *Let $s = i+j-1$. The number $T_{i,j}$ of connected bipartite graphs is $T_{ij} = i^{j-1} \cdot j^{i-1} \cdot s! = i^{j-1} \cdot j^{i-1} \cdot (i+j-1)!$*

B. Expected Maximum Matching Size

In the next theorem we find the expected maximum matching size with a static partition of the right-side vertices.

Theorem 11. *Given the static partitioning of the bipartite graph G_β, the expected maximum matching size $\mu(G_\beta)$ is*

$$\mu(G_\beta) = m - \sum_{s=0}^{n} \binom{n}{s} \sum_{i=\ell_1}^{\ell_2} \binom{\beta \cdot m}{i} \binom{(1-\beta)\cdot m}{s+1-i} \left(1 - \frac{i}{\beta \cdot m}\right)^{n-s} \cdot \left(1 - \frac{s+1-i}{(1-\beta)\cdot m}\right)^{n-s} \left(\frac{i}{\beta \cdot m}\right)^{s} \left(\frac{s+1-i}{(1-\beta)\cdot m}\right)^{s} \cdot P_{i,s+1-i},$$

where $\ell_1 = \max\{0, s+1-(1-\beta)\cdot m\}$, $\ell_2 = \min(s+1, \beta \cdot m)$, $P_{ij} = \frac{T_{ij}}{(i \cdot j)^{i+j-1}}$, *and* $T_{ij} = i^{j-1} \cdot j^{i-1} \cdot (i+j-1)!$.

Proof: Similarly to the proof of Theorem 1, our proof is based on counting the expected number of vertices in L that are not in some specific maximum matching M of G_β, based on the decomposition of G into its connected components. As in the proof of Theorem 1, we consider the number of connected components with exactly s vertices in L and $q = s+1$ vertices in $R_u \cup R_d$, where we have to sum over all possible combinations $(i, s+1-i)$, where i corresponds to the number of vertices taken from R_u and $s+1-i$ corresponds to those taken from R_d.

Thus, the expected number of connected components in G_β with s vertices in L, i vertices in R_u and $s+1-i$ vertices in R_d is given by:

$$\binom{n}{s} \cdot \binom{\beta \cdot m}{i} \binom{(1-\beta)\cdot m}{s+1-i} \cdot \left(1 - \frac{i}{\beta \cdot m}\right)^{n-s} \cdot \left(1 - \frac{s+1-i}{(1-\beta)\cdot m}\right)^{n-s} \cdot \left(\frac{i}{\beta \cdot m}\right)^{s} \cdot \left(\frac{s+1-i}{(1-\beta)\cdot m}\right)^{s} \cdot P_{i,s+1-i},$$

The above expression consists of the following factors (in order):
(i) choosing the s vertices in L;
(ii) choosing the i vertices in R_u;
(iii) choosing the $s+1-i$ vertices in R_d;
(iv) the probability that all i vertices in R_u may be connected only to the chosen s vertices in L;
(v) the probability that all $s+1-i$ vertices in R_d may be connected only to the chosen s vertices in L;
(vi) the probability that all s vertices in L are only connected to the i vertices in R_u;
(vii) the probability that all s vertices in L are only connected to the $s+1-i$ vertices in R_d; and,
(viii) the probability that all chosen vertices are connected.

Finally, adding the expressions for all possible s's and i's and subtracting it from m yields the claimed result. ■

Up until now we were only interested in the maximum matching size. However, to determine the latency and throughput of our hash table, we also need to know how many elements are in each of the two partitions. Note that there are many matchings with the maximum matching size. Among those, we are interested in the matchings that maximize the expected number of elements in the first partition. This is reflected in the following theorem.

Theorem 12. *Given the static partitioning of the bipartite graph G_β, there is a maximum matching such that the expected number of elements in the first partition is*

$$\mu^1(G_\beta) = \beta m \cdot \left(1 - \left(1 - \frac{1}{\beta m}\right)^n\right).$$

Moreover, there is no other maximum matching with a higher expected number of elements in the first partition.

Proof: Since cuckoo hashing brings to a maximum matching, we consider the case where we always first try to insert an element into its first choice, i.e. into the first partition. Once an element is inserted there, the corresponding bucket would always stay occupied. This follows by the definition of cuckoo hashing.

Since the first partition size is βm, and there are n elements, the probability that no element hashes into some bucket in the first partition is $\left(1 - \frac{1}{\beta m}\right)^n$. It then follows that the expected number of occupied buckets in the first partition is as claimed in the theorem. ■

C. Limit Normalized Expected Maximum Matching Size

As in the previous sections, we are also interested in the asymptotic behavior of the partitioned cuckoo hashing scheme where $n \to \infty$ with both fixed load $\alpha = \frac{n}{m}$ and fixed partition β. We obtain the following theorem.

Theorem 13. *Given the static partitioning of the bipartite graph G_β, the limit normalized expected maximum matching size $\gamma_\beta = \lim_{n\to\infty} \frac{\mu(G_\beta)}{n}$ for $\beta \in (0,1)$ is given by:*

$$\gamma_\beta = \frac{1}{\alpha} - \frac{\beta \cdot (1-\beta)}{\alpha^2} \cdot (t_1 + t_2 - t_1 \cdot t_2),$$

where t_1, t_2 are provided by the following equations:

$$\frac{\alpha}{1-\beta} \cdot e^{-\frac{\alpha}{\beta}} = t_1 \cdot e^{-t_2} \quad , \quad \frac{\alpha}{\beta} \cdot e^{-\frac{\alpha}{1-\beta}} = t_2 \cdot e^{-t_1} \qquad (2)$$

and satisfy the condition $t_1 \cdot t_2 \leq 1$.

For $\beta \in \{0,1\}$ (namely, the trivial partitions), the limit normalized expected maximum matching size γ_β is $\frac{1}{\alpha} - \frac{1}{\alpha} \cdot e^{-\alpha}$.

We deduce the following two corollaries. The first one states that cuckoo hashing with equal partition is asymptotically equivalent to cuckoo hashing with no partition. The second one states how close partition needs to be to equal in order to reach an ideal average matching.

Corollary 14 (Asymptotic Equivalence)**.** *Let $d = 2$. The limit normalized expected maximum matching size of G_β with $\beta = 0.5$ is the same as the limit expected maximum matching size of G.*

Proof: We substitute $\beta = 0.5$ in the expression from Theorem 13, and get $\frac{\alpha}{0.5} \cdot e^{-\frac{\alpha}{0.5}} = t_1 \cdot e^{-t_2}$, $\frac{\alpha}{0.5} \cdot e^{-\frac{\alpha}{0.5}} = t_2 \cdot e^{-t_1}$. One of the solutions of the above equations is $t_1 =$

$t_2 = -W\left(-2\alpha e^{-2\alpha}\right)$. In the proof of Theorem 4, we showed that $-W\left(-2\alpha e^{-2\alpha}\right) \leq 1$. Thus, $t_1 \cdot t_2 < 1$. By substituting this solution in the expression for γ_β from Theorem 13 , we get the exact expression as in Equation (1). ■

Corollary 15. *Let $d = 2$, $\alpha \leq \frac{1}{2}$, and fix a partition β. The limit normalized expected maximum matching size $\gamma_\beta = \lim_{n\to\infty} \frac{\mu(G_\beta)}{n}$ is 1 whenever $\frac{1-\sqrt{1-4\alpha^2}}{2} \leq \beta \leq \frac{1+\sqrt{1-4\alpha^2}}{2}$.*

As in the last section, we are also interested in the limit normalized expected fraction of elements in each of the partitions. This following theorem corresponds to Theorem 12.

Theorem 16. *Given the static partitioning of the bipartite graph G_β, in the scaled system, there is a maximum matching such that the asymptotic expected fraction of elements in the first subtable is*

$$\mu^1_\beta = \frac{\beta}{\alpha} - \frac{\beta}{\alpha} e^{-\frac{\alpha}{\beta}}$$

Moreover, there is no maximum matching with a higher expected fraction.

Proof: The proof is obtained by taking the limit of the expression in Theorem 12, normalized by n. ■

Finally, given the off-chip memory access latency b, the following corollary shows the throughput of the hash table. It follows immediately from Theorems 13 and 16,

Corollary 17 (Asymptotic Equivalence)**.** *Given an on-chip memory with two partitions of sizes βm and $(1-\beta)\, m$, and assuming an SRAM of access latency* 1 *and an off-chip DRAM of access latency b, the hash table throughput is*

$$\left(\mu^1_\beta + 2 \cdot \left(\gamma_\beta - \mu^1_\beta\right) + (2+b) \cdot \gamma_\beta\right)^{-1} \tag{3}$$

Following Corollary 17, it is possible to compute the optimal partition β that maximizes the hash table throughput. We further evaluate this in Section VII-F.

VI. Super-Cuckoo: Bipartite Graphs with $d > 2$

We are now interested in checking how powerful cuckoo hashing can be when we allow more than 2 hash functions per element. Of course, using more hash functions will result in an increase in implementation complexity, and therefore one goal of this study is to point out the tradeoff between efficiency and complexity.

In this section we briefly show how our method can be applied to find an upper bound on the expected maximum size matching where each left-side vertex has $d > 2$ choices. Formally, we are given two disjoint sets of vertices L and R of size n and m, respectively, and a random bipartite graph $G^d = \langle L + R, E\rangle$, where each vertex $v \in L$ has d outgoing edges whose destinations are chosen independently at random (with repetition) among all vertices in R. We obtain the following upper bound on the maximum matching size of the bipartite graph G^d.

Theorem 18. *Let $\ell = \min\left(n, \left\lfloor \frac{m-1}{d-1} \right\rfloor\right)$ and $q = (d-1)\cdot s + 1$. Then, $\mu\left(G^d\right)$ is at most*

$$\min\left(n, m - \sum_{s=0}^{\ell} (q-s) \binom{n}{s}\binom{m}{q}\left(1-\frac{q}{m}\right)^{d(n-s)} \left(\frac{q}{m}\right)^{ds} \frac{d^s \cdot q!}{q^{(d-1)\cdot s+2}}\right).$$

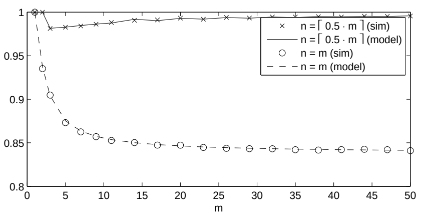


Fig. 3. Expected maximum matching size for various values of n and m (normalized by n).

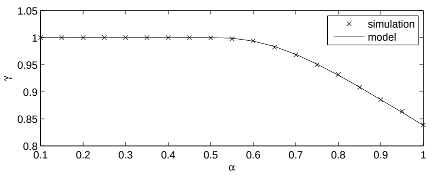


Fig. 4. Limit expected normalized maximum matching size for various values of load α.

An evaluation of the upper bound and a comparison to the simulated expected matching size is presented in Section VII-D.

VII. Evaluation and Experiments

We now evaluate our theoretical results, using both synthetic evaluations and trace-based experiments.

A. Expected Maximum Matching Size With $d = 2$

Figure 3 shows the expected maximum matching size normalized by n for various values of n and m. It compares simulation results with our analytical model from Theorem 1. For each instance of n and m, we randomized 10,000 bipartite graphs, then computed the average value. The results confirm that our model is fairly accurate, and also show the convergence of the expected maximum matching size to its limit.

Figure 4 shows the expected maximum matching size normalized by n as found in Theorem 4, for various values of load α, both via our analytical model and via simulations. The simulations were performed using $m = 1000$ and $n = \alpha \cdot m$. For each value of α, we randomized 100 bipartite graphs. Again, the model appears fairly accurate.

B. Expected Maximum Matching Size With $d_v \leq 2$

Figure 5 shows the normalized limit expected maximum matching size, for various values of load α and average number of choices a, both via our analytical model (from Theorem 8) as well as via simulations. The simulations were performed using $m = 1000$ and $n = \alpha \cdot m$, where for each instance of the simulation we randomized 100 bipartite graphs. Once again, the results confirm that our model is fairly accurate.

C. Expected Maximum Matching Size With Static Partition

Figure 6 shows the limit expected maximum matching size normalized by n, for various values of load α and

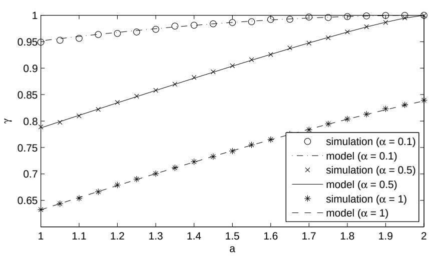


Fig. 5. *Low-choice cuckoo*: limit expected normalized maximum matching size as a function of the average number of choices a, for various values of the load α.

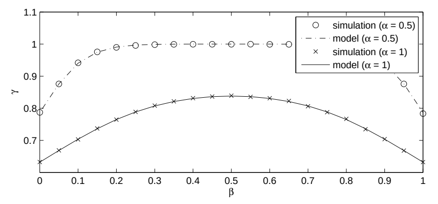


Fig. 6. *Partitioned cuckoo*: limit expected normalized maximum matching size as a function of the partition parameter β, for various values of the load α.

partition β, both via our analytical model (from Theorem 13) and via simulations. The simulations were performed using $m = 1000$ and $n = \alpha \cdot m$. For each pair of values of α and β, we randomized 100 bipartite graphs. The results confirm the accuracy of our model. They also illustrate how the limit expected maximum matching size is symmetric around $\beta = 0.5$, and the symmetric partition case reaches the optimum for this metric.

Note that in case $\alpha = 0.5$ and $\beta < 0.5$, while it seems that the normalized limit expected maximum matching size is 1, it is not the case. For instance, in case $\alpha = 0.5$ and $\beta = 0.45$, we get that $1 - \gamma_\beta \approx 1.675 \cdot 10^{-7}$. Referring to Corollary 15, this is because $\frac{1+\sqrt{1-4\alpha^2}}{2} = \frac{1}{2}$ in that extreme case. For strictly smaller loads, the imbalance in the partition sizes does not necessarily reduce γ_β and the plot becomes true flat in the middle, as illustrated in Corollary 15.

D. Expected Maximum Matching Size With $d > 2$

We evaluate the upper bound found for the expected matching size (Theorem 18). Figure 7 shows our upper bound as well as simulation results for various values of the number of choices d. We took $n = m = 100$, while for each instance of d, we randomized 10^5 bipartite graphs. In the case of $d = 2$, our upper bound matches the exact expression found in Theorem 1 and thus matches the simulation results. In addition, we can compare simulation results for higher values of d with our bounds. For instance, in the case of $d = 3$ the normalized expected maximum matching size via the simulation is 0.9402, while our upper bound is 0.9508. In case $d = 4$, we get a simulation value of 0.9795, while the corresponding upper bound is 0.9820.

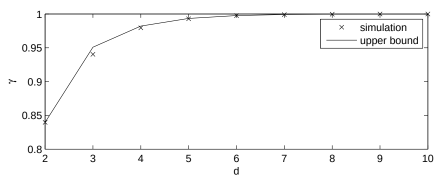


Fig. 7. *Super-cuckoo*: comparison of the simulation results and the theoretical upper bound for the normalized expected maximum matching size, as a function of the number of choices d in cuckoo hashing. The load is $\alpha = 1$.

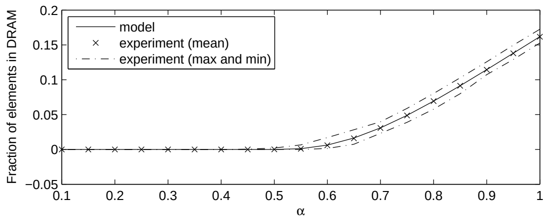


Fig. 8. Experiment using real-life traces and hash functions of the normalized number of elements in the DRAM, and comparison to the theoretical model.

E. Trace-Driven Experiments

We have also conducted experiments using real-life traces recorded on a single direction of an OC192 backbone link [38], where packets are hashed using a real 64-bit mix function [39]. Our goal is two-folded. First, we would like to verify that our analysis agrees with results of real-life traces. And second, we want to verify that the distribution of the overflow list size is highly concentrated around its mean, as stated in Theorem 2.

We took $m = 10{,}000$, and set a number of elements n as corresponding to various values of load α. We repeated each experiment 100 times. Fig. 8 shows that the results of our experiments are very close to our model. Furthermore, it also shows that the minimum and the maximum off-chip DRAM size are close to the mean.

F. Access Throughput

Finally, we compare the access throughputs of network hash tables based on several hashing schemes.

Figure 9 plots the access throughput when the off-chip memory is $b = 5$ times slower than the on-chip memory. In the case of the partitioned cuckoo hashing with $d = 2$, it assumes an optimal partitioning for each load, as provided by by Corollary 17. It is clear that that there is limited difference between the cases of $d = 2$ with partitioning and $d = 2$ without partitioning, and therefore by simplicity we only consider one of them below. More importantly,it shows that up to a load of approximately 0.7, and for loads above approximately 1, *it is better to use $d = 2$ than $d = 3$*. Intuitively, this is because the decrease in the number of needed SRAM accesses more than compensates for the resulting loss of efficiency and therefore the increase in the number of DRAM accesses.

Figure 10 further highlights this result. It plots the ratio of the throughput of several cuckoo and d-left [2] variants by the throughput of partitioned cuckoo with $d = 2$ and optimal partitioning. Therefore, it emphasizes the throughput gain (or loss) with respect to the partitioned cuckoo algorithm.

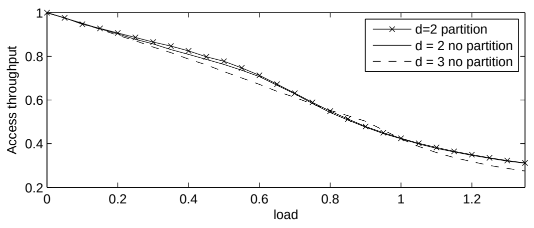
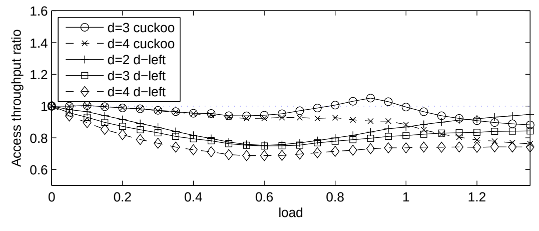
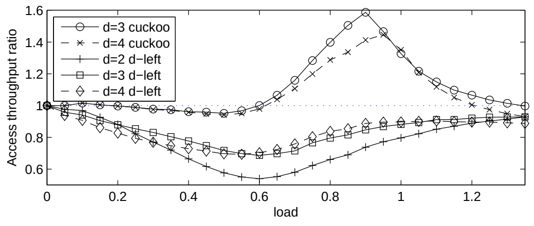


Fig. 9. Expected access throughput given that the off-chip memory is $b = 5$ times slower than the on-chip memory



(a) $b = 5$



(b) $b = 15$

Fig. 10. Ratio of the throughput of several cuckoo and d-left variants by the throughput of optimally-partitioned cuckoo hashing with $d = 2$ and optimal partitioning.

In Figure 10(a), when DRAM accesses are $b = 5$ slower than SRAM accesses, we can see more clearly the outperformance of partitioned cuckoo with $d = 2$ under most loads, and compared to most hashing algorithms. For instance, at load $\alpha = 1$, i.e. when the hash table is highly loaded, partitioned cuckoo hashing with $d = 2$ is (surprisingly) better then $d = 3$. By Theorem 13 and Theorem 16, we can find that the optimal memory partitioning is $\beta = 57.0\%$ for the first subtable and $1 - \beta = 43.0\%$ for the second subtable, and that 47.2% of the elements are stored in the first subtable, 36.2% in the second one, and 16.6% in the off-chip DRAM. The resulting throughput is $(1 \cdot 0.472 + 2 \cdot 0.362 + (2 + 5) \cdot 0.166)^{-1} \approx 0.4241$. This is indeed higher than the throughput of cuckoo hashing with $d = 3$ with no partitioning, which is found to be 0.4194 in simulations.

Finally, Figure 10(b) shows that for $b = 15$, the cost of DRAM accesses becomes higher, and therefore there is more incentive to be efficient in the SRAM. Cuckoo hashing with $d = 3$ outperforms then in most cases. Most often, the cost of DRAM is not high enough to justify $d = 4$.

VIII. Conclusion

In this work, we analyzed the throughput of a network hash table combined of on-chip SRAM and off-chip DRAM, and based on cuckoo hashing. We first provided an exact expression for the expected maximum matching size of a random bipartite graph with each left-side vertex picking $d = 2$ right-side vertices, for any number of left-side and right-side vertices. Then, we deduced asymptotic results as the memory size goes to infinity, and even showed a connection to the Lambert-W function. Both these results directly apply as exact results for cuckoo hashing, and also serve as upper bounds for any alternative multiple-choice hashing algorithm with $d = 2$ choices. We further analyzed several cuckoo hashing variants.

Our results illustrate the impact of the SRAM/DRAM access time ratio on the parameter choice. In particular, we show that the common intuition of avoiding DRAM accesses by using highly efficient schemes is not always correct. Sometimes, it is better to use a less efficient hashing method because it needs less SRAM accesses. In addition, while we focused in this paper on the throughput of an SRAM/DRAM system, these results can also be extended to evaluate the system power consumption. They can also apply to alternative systems, such as SRAM/CAM implementations.

Acknowledgment

This work was partly supported by the European Research Council Starting Grants n°210389 and n°259085, the Alon Fellowship, the ATS-WD Career Development Chair, and the Loewengart Research Fund.

References

[1] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, University of California at Berkley, 1996.
[2] B. Vöcking, "How asymmetry helps load balancing," in *IEEE FOCS*, 1999, pp. 131–141.
[3] J. W. Byers, J. Considine, and M. Mitzenmacher, "Geometric generalizations of the power of two choices," in *SPAA*. New York, NY, USA: ACM, 2004, pp. 54–63.
[4] G. R. Wright and W. Stevens, *TCP/IP Illustrated*. Addison-Wesley Publishing Co., 2005, vol. 2.
[5] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "Bloom filters via d-left hashing and dynamic bit reassignment," in *Allerton*, 2006.
[6] A. Kirsch and M. Mitzenmacher, "The power of one move: Hashing schemes for hardware," in *IEEE Infocom*, 2008, pp. 565–573.
[7] Y. Kanizo, D. Hay, and I. Keslassy, "Optimal fast hashing," in *IEEE Infocom*, 2009.
[8] A. Kirsch, M. Mitzenmacher, and U. Wieder, "More robust hashing: Cuckoo hashing with a stash," *SIAM Journal on Computing*, vol. 39, no. 4, pp. 1543–1561, 2009.
[9] R. Kutzelnigg, "A further analysis of cuckoo hashing with a stash and random graphs of excess r," *DMTCS*, vol. 12, no. 3, pp. 81–102, 2010.
[10] A. Kirsch, M. Mitzenmacher, and G. Varghese., *Hash-Based Techniques for High-Speed Packet Processing*. DIMACS, 2010.
[11] M. Drmota and R. Kutzelnigg, "A precise analysis of cuckoo hashing," *ACM Trans. Algorithms*, 2009.
[12] M. Naor, G. Segev, and U. Wieder, "History-independent cuckoo hashing," in *ICALP*, 2008.
[13] M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink, "Tight thresholds for cuckoo hashing via XORSAT," in *ICALP*, 2010.
[14] N. Fountoulakis and K. Panagiotou, "Sharp load thresholds for cuckoo hashing," *CoRR*, vol. abs/0910.5147, 2009.
[15] A. M. Frieze and P. Melsted, "Maximum matchings in random bipartite graphs and the space utilization of cuckoo hashtables," *CoRR*, vol. abs/0910.5535, 2009.
[16] Y. Arbitman, M. Naor, and G. Segev, "De-amortized cuckoo hashing: Provable worst-case performance and experimental results," in *ICALP*, 2009.

[17] A. Frieze, P. Melsted, and M. Mitzenmacher, "An analysis of random-walk cuckoo hashing," in *RANDOM*, 2009.
[18] M. Mitzenmacher, "Some open questions related to cuckoo hashing," in *European Symposium on Algorithms*, 2009, pp. 1–10.
[19] U. Ben-Porat, A. Bremler-Barr, H. Levy, and B. Plattner, "On the vulnerability of hardware hash tables to sophisticated attacks," in *Networking (1)*, 2012, pp. 135–148.
[20] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations," in *ACM STOC*, 1994, pp. 593–602.
[21] M. Mitzenmacher, A. Richa, and R. Sitaraman, *The power of two random choices: A survey of techniques and results*, 2000, vol. 1, pp. 255–312.
[22] A. Z. Broder and A. R. Karlin, "Multilevel adaptive hashing," in *ACM-SIAM SODA*, 1990, pp. 43–53.
[23] A. Kirsch and M. Mitzenmacher, "Simple summaries for hashing with choices," *IEEE/ACM Trans. Networking*, vol. 16, no. 1, pp. 218–231, 2008.
[24] S. Kumar, J. Turner, and P. Crowley, "Peacock hashing: Deterministic and updatable hashing for high performance networking," in *IEEE Infocom*, 2008, pp. 556–564.
[25] Y. Kanizo, D. Hay, and I. Keslassy, "The balanced Bloom filter," in *ICC*, 2012.
[26] W. W. David, "Matchings in random regular bipartite digraphs," *Discrete Mathematics*, vol. 31, no. 1, pp. 59–64, 1980.
[27] A. Frieze, "Perfect matchings in random bipartite graphs with minimal degree at least 2," *Random Struct Algor*, vol. 26, no. 3, pp. 319–358, 2005.
[28] J. H. Cho and E. M. Palmer, "On the asymptotic behavior of the independence number of a random (n, n)-tree," *Journal Graphs and Combinatorics*, vol. 12, no. 1, pp. 1–8, 1996.
[29] C. Banderier, M. Kuba, and A. Panholzer, "Analysis of three graph parameters for random trees," *Random Struct Algor*, vol. 35, no. 1, pp. 42–69, 2009.
[30] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman, "Algorithms for loop matchings," *SIAM J. on Applied Math.*, vol. 35, no. 1, pp. 68–82, 1978.
[31] H. Zhou and Z. Ou-Yang, "Maximum matching on random graphs," *CoRR*, vol. arXiv:cond-mat/0309348v1, 2003.
[32] M. Mitzenmacher and S. Vadhan, "Why simple hash functions work: Exploiting the entropy in a data stream," in *SODA*, 2008.
[33] N. Alon and J. H. Spencer, *The Probabilistic Method.* Wiley, 2000.
[34] J. Yeh, *Real analysis: theory of measure and integration*, 2nd ed. World Scientific Publishing Company, 2006.
[35] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996.
[36] R. M. Karp, "The transitive closure of a random digraph," *Random Struct Algor*, vol. 1, no. 1, pp. 73–93, 1990.
[37] Y. Peres, K. Talwar, and U. Wieder, "The $1+\beta$ choice process and weighted balls into bins," in *SODA*, 2010.
[38] C. Shannon, E. Aben, K. C. Claffy, and D. E. Andersen, "CAIDA Anonymized 2008 Internet Trace equinix-chicago 2008-03-19 19:00-20:00 UTC (DITL) (collection)," http://imdc.datcat.org/collection/.
[39] T. Wang, "Integer hash function," http://www.concentric.net/ Ttwang/tech/inthash.htm.
[40] R. Diestel, *Graph Theory*, 3rd ed. Springer-Verlag, 2005.
[41] H. I. Scoins, "The number of trees with nodes of alternate parity," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 58, no. 01, pp. 12–16, 1962.
[42] F. Bergeron, G. Labelle, and P. Leroux, *Combinatorial Species and Tree-Like Structures.* Cambridge University Press, 1998.

Appendix A
Omitted proofs

A. Proof of Lemma 1

The proof follows by induction on s. For $s = 1$, there are 2 edges in the graph and therefore every graph with $q \geq 3$ is not connected. Assume that the claim holds up until $s = s'$, we next prove that it holds for any bipartite graph H' such that $|L_{H'}| = s' + 1$ and $|R_{H'}| \geq s' + 3$. Assume towards a contradiction that there is a graph H' that is connected. We first show that there is a vertex in $R_{H'}$ with a degree 1: This follows from the fact that the average right-side degree is $\frac{2(s'+1)}{s'+3} < 2$, implying that there is at least one vertex with degree strictly less than 2; since the graph is connected, there are no right-side vertices with degree 0. Let v_r be such a vertex and let $v_\ell \in L_{H'}$ be the (only) left-side vertex to which it is connected. By the induction hypothesis, the graph induced by $L_{H'} \setminus \{v_\ell\}$ and $R_{H'} \setminus \{v_r\}$ is not connected, implying it has at least two connected components. In H', v_ℓ is connected to v_r and since its degree is 2 it can be connected only to one of these components. This implies that H' is also not connected, and the claim follows. ■

B. Proof of Lemma 2

We first consider the case where $s = q$. For $S \subseteq L_H$, let $d(S) \subseteq R_H$ be the set of vertices that are adjacent to any vertex in S. Hall's Theorem [40] implies that to prove that $\mu(H) = q$ (namely, there is a perfect matching in H) it suffices to prove that for every $S \subseteq L_H$, $|S| \leq |d(S)|$. Assume towards a contradiction that there is a subset $S \subseteq L_H$ such that $|S| > |d(S)|$, and denote $|d(S)|$ as ℓ. Furthermore, consider the bipartite graph $\hat{H} = \left\langle \hat{L}_H + \hat{R}_H, \hat{E}_H \right\rangle$, in which $\hat{L}_H = L_H \setminus S$, $\hat{R}_H = R_H \cup \{\hat{v}_R\} \setminus d(S)$ (where $\hat{v}_R$ is a newly-introduced vertex) and any edge in $E(H)$ of the form (v_ℓ, v_r) such that $v_\ell \in L_H \setminus S$ and $v_r \in d(S)$ is replaced with the edge $(v_\ell, \hat{v}_R)$ in $\hat{E}_H$. Notice that since H is connected, $\hat{H}$ must be connected as well. Recall that $|S| > \ell$, thus $\left|\hat{L}_H\right| = |L_H \setminus S| \leq s - \ell - 1$, while $\left|\hat{R}_H\right| = |R_H \cup \{\hat{v}_R\} \setminus d(S)| = |R_H| - |d(S)| + 1 = s - b + 1$. This contradicts Lemma 1, implying that for every $S \subseteq L_H$, $|S| \leq |d(S)|$ and by Hall's Theorem $\mu(H) = q$.

For $s > q$, trivially $\mu(H) \leq q$. Therefore, it suffices to show that there exists a subset $S \subseteq L_H$ of size q, such that the corresponding bipartite subgraph is connected (and hence has a perfect matching of size q). We construct S in q iterations such that at the end of iteration n we end up with some subsets $S_n \subseteq L_H$ and $Q_n \subseteq R_H$ of the same size n, whose corresponding subgraph is connected. We start by $n = 1$ and pick some vertex $v_R \in R_H$ and one of its adjacent vertices $v_L \in L_H$. Assuming that at the end of iteration n, sets S_n and Q_n were chosen (and their corresponding graph is connected), we next construct S_{n+1} and Q_{n+1}. Let v_1 be an arbitrary vertex in S_n and let v_2 be an arbitrary vertex in $L_H \setminus S_n$ (such a vertex always exists since $s > q > n$). Similarly, let v_1' be an arbitrary vertex in Q_n and let v_2' be an arbitrary vertex in $R_H \setminus Q_n$. Since H is connected there is a path between v_1 and v_2, and let v be the first vertex along this path that is not in S_n. Similarly, v' is the first vertex along the path between v_1' and v_2' that is not in Q_n. We differentiate between three cases: *(i)* v is adjacent to Q_n and v' is to S_n. In this case $S_{n+1} = S_n \cup \{v\}$ and $Q_{n+1} = Q_n \cup \{v'\}$ and the corresponding subgraph is connected; *(ii)* v is not adjacent to a Q_n. Let w be the vertex before v in the path between v_1 and v_2, and let w' be the vertex before w in the path. Note that $w' \in S_n$ by the choice of v, and that $w \notin Q_n$ (otherwise v is adjacent to a Q_n). Thus, for $S_{n+1} = S_n \cup \{v\}$ and $Q_{n+1} = Q_n \cup \{w\}$, the corresponding subgraph is connected; *(iii)* v' is not adjacent to a S_n. The claim holds similarly to case

(ii) by looking at the path between v_1' and v_2'. We continue this construction for q iterations, resulting in two subsets $S_q \subseteq L_H$ and $Q_q \subseteq R_H$ of size q each, whose corresponding subgraph is connected. ■

C. Proof of Lemma 3

Since each vertex in L_H has a degree of two, the sum of the degrees of all the vertices in R_H is $2s = 2q - 2$. Therefore, there must be at least one vertex $v_r \in R_H$ with degree 1 (there cannot be a vertex with degree 0 since H is connected). Let $v_L \in L_H$ be the (only) vertex that is connected to v_R and $\hat{v}_R \in R_H$ be the other vertex that is connected to v_L. Also consider the bipartite graph $\hat{H} = \left\langle \hat{L}_H + \hat{R}_H, \hat{E}_H \right\rangle$ that is given by removing v_R from H and adding a new edge $(v_L, \hat{v}_R)$. By the construction of $\hat{H}$, the degree of each vertex in $\hat{L}_H$ is exactly 2. Moreover, since H is connected, $\hat{H}$ is also connected. Hence, Lemma 2 implies that there is a matching of size s in $\hat{H}$. By the construction of $\hat{H}$, this is also a matching in graph H. ■

D. Proof of Lemma 4

First, if H is a tree then it is connected by definition. To show the other direction, we assume towards a contradiction that H is a connected graph with cycles; let C be a cycle in H, and consider an edge $e = (v_L, v_R)$ that resides at cycle C (where $v_L \in L_H$ and $v_R \in R_H$). We build the bipartite graph $\hat{H} = \left\langle \hat{L}_H + \hat{R}_H, \hat{E}_H \right\rangle$, such that $\hat{L}_H = L_H$, $\hat{R}_H = R_H \cup \{\hat{v}_R\}$, where $\hat{v}_R$ is a newly-introduced vertex, and $\hat{E}_H = E_H \setminus \{e\} \cup \{\hat{e}\}$, where $\hat{e} = (v_L, \hat{v}_R)$. Intuitively, we replace one of the edges in the cycle to reach for a newly-introduced vertex, and by that we increase the size of the connected component. Notice that $\hat{H}$ is connected and all vertices in $\hat{L}_H$ have a degree of 2. But, $\left|\hat{L}_H\right| < \left|\hat{R}_H\right| - 1$, thus contradicting Lemma 1 and the claim follows. ■

E. Proof of Lemma 5

We count the connected bipartite graphs with two disjoint sets L_H and R_H. By Lemma 4, we have to count the number of trees over the set $L_H \cup R_H$, where edges must be of the form (v_L, v_R), such that $v_L \in L_H$ and $v_R \in R_H$. We build (and count) the set as follows: The number of trees over the set R_H is $(s+1)^{s-1}$ (Cayleys formula). For each such tree instance, we put a new vertex (originally from L_H) between each pair of adjacent vertices. There are $s!$ possibilities to do so. ■

F. An Alternative Proof of Theorem 4

Considering the random graph with m vertices and n edges such that a vertex m_1 is connected to vertex m_2 if and only if there exists an element that hashes into m_1 and m_2. This random graph is called the cuckoo graph [13]. Neglecting the $O(1)$ loops, this graph is equivalent to the Erdös-Renyi random graph $G_{m,n}$ that assigns equal probability to all graphs with exactly n edges (and m vertices)

A matching in $G_{m,n}$ corresponds to directing some of the edges in the random graph such that the in-degree is at most 1. For each connected component C in $G_{m,n}$, if C is a tree we can direct all edges, while in all other cases we can direct as much edges as the number of vertices.

The number of such edges and vertices can be found in [33], [36], yielding the exact same result.

G. Proof of Lemma 7

The proof is identical to the proof of Lemma 4 with two modifications. First, instead of initially counting the number of trees over the set R_H, we count the number of parity trees [41] over the disjoint sets R_{H_u} and R_{H_d}. By [41] we are given that the number of parity trees is $i^{j-1} \cdot j^{i-1}$. Second, we do not have to color the edges because of the partition. ■

H. Proof of Theorem 13

As in the proof of Theorem 4, we compute the limit of $\frac{\mu(G)}{n}$ as $n \to \infty$. We consider the case where $\alpha = \frac{n}{m}$ and $0 \le \beta \le 1$ are fixed. So $\gamma_\beta = \lim_{n\to\infty} \frac{\mu(G_\beta)}{n}$, that is,

$$\begin{aligned}\gamma_\beta \quad = \quad & \lim_{n\to\infty} \frac{1}{n} \cdot \left(m - \sum_{s=0}^{n} \binom{n}{s} \cdot \sum_{i=b_1}^{b_2} \binom{\beta \cdot m}{i} \binom{(1-\beta)\cdot m}{s+1-i} \cdot \right.\\ & \left(1 - \frac{i}{\beta \cdot m}\right)^{n-s} \cdot \left(1 - \frac{s+1-i}{(1-\beta)\cdot m}\right)^{n-s} \cdot \left(\frac{i}{\beta \cdot m}\right)^{s} \cdot \\ & \left.\left(\frac{s+1-i}{(1-\beta)\cdot m}\right)^{s} \cdot P_{i,s+1-i}\right)\end{aligned}$$

By substituting the expression for $P_{i,s+1-i}$ from Theorem 11, and moving $\binom{n}{s}$ inside the second summation, we get:

$$\begin{aligned}\gamma_\beta = \lim_{n\to\infty} & \left(\frac{1}{\alpha} - \frac{1}{n} \sum_{s=0}^{n} \sum_{i=0}^{s+1} \binom{n}{s} \binom{\beta m}{i} \binom{(1-\beta)\, m}{s+1-i} \left(1 - \frac{i}{\beta m}\right)^{n-s} \cdot \right.\\ & \left(1 - \frac{s+1-i}{(1-\beta)\cdot m}\right)^{n-s} \cdot \left(\frac{i}{\beta \cdot m}\right)^{s} \cdot \left(\frac{s+1-i}{(1-\beta)\cdot m}\right)^{s} \cdot \\ & \left. \frac{i^{(s+1-i)-1} \cdot (s+1-i)^{i-1} \cdot (i + (s+1-i) - 1)!}{(i \cdot (s+1-i))^{i+(s+1-i)-1}} \right)\end{aligned}$$

By substituting $\alpha = \frac{n}{m}$, we get:

$$\begin{aligned}\gamma_\beta = \lim_{n\to\infty} & \left(\frac{1}{\alpha} - \frac{1}{n} \sum_{s=0}^{n} \sum_{i=0}^{s+1} \binom{n}{s} \binom{\frac{\beta}{\alpha} n}{i} \binom{\frac{1-\beta}{\alpha} n}{s+1-i} \left(1 - \frac{i}{\frac{\beta}{\alpha} n}\right)^{n-s} \cdot \right.\\ & \left(1 - \frac{s+1-i}{\frac{1-\beta}{\alpha} \cdot n}\right)^{n-s} \cdot \left(\frac{i}{\frac{\beta}{\alpha} \cdot n}\right)^{s} \cdot \left(\frac{s+1-i}{\frac{1-\beta}{\alpha} \cdot n}\right)^{s} \cdot \\ & \left. \frac{i^{(s+1-i)-1} \cdot (s+1-i)^{i-1} \cdot (i + (s+1-i) - 1)!}{(i \cdot (s+1-i))^{i+(s+1-i)-1}} \right)\end{aligned}$$

As in the proof of Theorems 4 and 8, using the monotone convergence theorem [34], we can put the limit inside the sum. By further simplifying the above expression with similar consideration to the proofs of Theorems 4 and 8, we get eventually:

$$\begin{aligned}\gamma_\beta = \frac{1}{\alpha} - \frac{\beta \cdot (1-\beta)}{\alpha^2} \quad & \sum_{s=0}^{\infty} \sum_{i=0}^{s+1} \frac{i^{(s+1-i)-1} \cdot (s+1-i)^{i-1}}{i! \cdot (s+1-i)!} \cdot \\ & \left(\frac{\alpha}{\beta} \cdot e^{-\frac{\alpha}{1-\beta}}\right)^{s+1-i} \cdot \left(\frac{\alpha}{1-\beta} \cdot e^{-\frac{\alpha}{\beta}}\right)^{i}\end{aligned}$$

We switch the order of summation and get that $i \in \{0, 1, \ldots\}$ and s goes from $\max\{0, i-1\}$ to ∞. We also substitute $j = s + 1 - i$ (or $s = i + j - 1$). Thus,

$$\gamma_\beta = \frac{1}{\alpha} - \frac{\beta \cdot (1-\beta)}{\alpha^2} \sum_{i=0}^{\infty} \sum_{j=\max\{0,i-1\}}^{\infty} \frac{i^{j-1} \cdot j^{i-1}}{i! \cdot j!} \cdot \left(\frac{\alpha}{1-\beta} \cdot e^{-\frac{\alpha}{\beta}}\right)^i \cdot \left(\frac{\alpha}{\beta} \cdot e^{-\frac{\alpha}{1-\beta}}\right)^j \tag{4}$$

Let $T(x,y) = \sum_{j+i \geq 1} \frac{i^{j-1} \cdot j^{i-1}}{i! \cdot j!} \cdot x^i \cdot y^j$. This expression has been previously found [11] to be the multivariate formal power series about the point $(x_0, y_0) = (0,0)$ of $t(x,y) = t_1(x,y) + t_2(x,y) - t_1(x,y) \cdot t_2(x,y)$ where $t_1(x,y)$ and $t_2(x,y)$ are given by the following implicit multivariate functions:

$$x = t_1(x,y) \cdot e^{-t_2(x,y)} \quad , \quad y = t_2(x,y) \cdot e^{-t_1(x,y)} \tag{5}$$

However, the mentioned range of convergence in [11] is insufficient for our case. (Note also that in [11] the sums should be over $i + j \geq 1$ and not over $i, j \geq 0$.)

Since we compute the limit normalized expected maximum matching, then the expression for γ_β in Equation (4) is bounded from below by 0, thus, by Equation (4) the double summation is bounded from above by a constant. On the other hand, all terms in the summation in Equation (4) are positive. Then, if we look at the partial-sum series (by defining an arbitrary order), we get an increasing series which is bounded. Thus, by the monotone convergence theorem the double series converges for any values x and y satisfying $x = \frac{\alpha}{1-\beta} \cdot e^{-\frac{\alpha}{\beta}}$ and $y = \frac{\alpha}{\beta} \cdot e^{-\frac{\alpha}{1-\beta}}$.

However, the multivariate functions in Equation (5) have multiple branches (as the Lambert-W function does [35]), that is, for a given x and y there is more than one solution. We aim to find this branch in terms of t_1 and t_2. We use the implicit function theorem to find the derivatives singularities. The Jacobian is given by

$$J = \begin{pmatrix} e^{-t_2(x,y)} & -t_1(x,y) \cdot e^{-t_2(x,y)} \\ -t_2(x,y) \cdot e^{-t_1(x,y)} & e^{-t_1(x,y)} \end{pmatrix},$$

and it is invertible wherever $|J| \neq 0$. Thus, there is a derivative singularity in case $t_1(x,y) \cdot t_2(x,y) = 1$, which is the only solution. Therefore, as the given formal power series in Equation (4) is about the point $(x_0, y_0) = (0,0)$ (which corresponds to $\alpha = 0$), where $t_1 = t_2 = 0$, it converges to the branch where $t_1(x,y) \cdot t_2(x,y) \leq 1$ (note that both $t_1(x,y)$ and $t_2(x,y)$ are always positive). ■

I. *Proof of Corollary 15*

One of the solutions to Equation (2) is given by: $t_1 = \frac{\alpha}{1-\beta}$, $t_2 = \frac{\alpha}{\beta}$. By substituting t_1 and t_2 in the expression for γ_β from Theorem 13, we get that the limit normalized expected maximum matching size is 1. We also have to verify that $t_1 \cdot t_2 \leq 1$. Since $\frac{\alpha}{1-\beta}$ and $\frac{\alpha}{\beta}$ are both positive, we are left with $\frac{\alpha}{1-\beta} \cdot \frac{\alpha}{\beta} < 1$. By solving the quadratic inequality, we get the claimed condition. Note that for $\alpha = 1/2$ the range reduces to $\beta = 1/2$. ■

J. *Proof of Lemma 6*

Assume on the contrary that H is connected but that there is (at least) a single vertex $v_L \in L_H$ with degree 1. Consider the bipartite graph $\hat{H} = \left\langle \hat{L}_H + \hat{R}_H, \hat{E}_H \right\rangle$, that is given by removing the vertex v_L (and its connected edge) from H . By the construction of $\hat{H}$, we get that $\hat{H}$ is connected, but $\left|\hat{L}_H\right| + 1 < \left|\hat{R}_H\right|$, which contradicts Lemma 1. ■

K. *Proof of Theorem 6*

As in the proof of Theorem 1, our proof is based on counting the expected number of vertices in L that are not in some specific maximum matching M of G, based on the decomposition of G into its connected components. The proof is almost identical, with the modification that, due to Lemma 6, we only take into account the d_2 vertices that have a degree of 2 (instead of all n vertices in the proof of Theorem 1).

Thus, the expected number of connected components in G with s elements in L and $s+1$ in R is given by:

$$\binom{d_2}{s}\binom{m}{s+1} \cdot \left(1 - \frac{s+1}{m}\right)^{2(d_2-s)+d_1} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot P_s,$$

where the above expression consists of the same considerations as in the proof of Theorem 1. Finally, as before, adding the expressions for all possible s's and subtracting the sum from m yields the claimed result. ■

L. *Proof of Theorem 7*

The number of vertices in L with degree 2 follows a Binomial distribution with n experiments and a probability of success p. In Theorem 6 we found the expected maximum matching size of each such instance. Thus, by the law of total expectation, the claimed result is given by computing the weighted average, where we compute a by the equations $d_1 + d_2 = n$ and $d_1 + 2 \cdot d_2 = a \cdot n$. ■

M. *Proof of Theorem 8*

We compute the limit of $\frac{\mu(G_a)}{n}$ as $n \to \infty$. We consider the case where $\alpha = \frac{n}{m}$ and $a = \frac{d_1 + 2 \cdot d_2}{n} > 1$ are fixed. So $\gamma_a = \lim_{n\to\infty} \frac{\mu(G_a)}{n}$, that is,

$$\gamma_a = \lim_{n\to\infty} \frac{1}{n} \left(m - \sum_{s=0}^{\ell} \binom{d_2}{s}\binom{m}{s+1} \left(1 - \frac{s+1}{m}\right)^{2(d_2-s)+d_1} \cdot \left(\frac{s+1}{m}\right)^{2s} \cdot P_s \right)$$

Given that $a = \frac{d_1 + 2 \cdot d_2}{n}$ and $n = d_1 + d_2$, we find that $d_2 = (a-1) \cdot n$ and $d_1 = (2-a) \cdot n$. Similarly to the proof of Theorem 4, we first have to find that each term in the summation is an increasing function with respect to n. We discover that $\left(1 - \frac{s+1}{m}\right)^{2(d_2-s)+d_1} = \left(1 - \frac{s+1}{m}\right)^{a \cdot n - s}$ is an increasing function (using differentiation), and also find that $\frac{1}{n} \cdot \binom{(a-1) \cdot n}{s}\binom{m}{s+1} \cdot \left(\frac{s+1}{m}\right)^{2s}$ is an increasing function as previously. Consequentially, each term in the sum is an increasing function and, by the monotone convergence theorem [34], we can put the limit inside the sum. By further simplifying the above expression as in the proof of Theorem 4 we eventually get:

$$\gamma_a = \frac{1}{\alpha} - \frac{1}{2\alpha^2 \cdot (a-1)} \cdot \sum_{j=1}^{\infty} \frac{(-j)^{j-2}}{j!} \cdot \left(-\alpha \cdot 2 \cdot (a-1) \cdot e^{-a\alpha}\right)^j$$

Let $T(x) = \sum_{j=1}^{\infty} \frac{(-j)^{j-2}}{j!} \cdot x^j$ be a Taylor expansion, where by substituting $x = -\alpha \cdot 2 \cdot (a-1) \cdot e^{-a\alpha}$ we get the above expression. Similarly to the proof of Theorem 4, we get that

$$T(x) = -W(x) - \frac{1}{2}W^2(x),$$

with convergence within $|x| \leq e^{-1}$ [35].

Since the function $f(\alpha) = -\alpha \cdot 2 \cdot (a-1) \cdot e^{-a\alpha}$ gets its minimum at $\alpha = a^{-1}$, where it equals $-\frac{2(a-1)}{a}e^{-1}$, and $\left|-\frac{2(a-1)}{a}e^{-1}\right| \leq e^{-1}$ for all $a \in [1,2]$, then for all α we can substitute $x = -\alpha \cdot 2 \cdot (a-1) \cdot e^{-a\alpha}$. Hence, it is within the radius of convergence of $T(x)$.

Finally, for the case where $a = 1$, then $d_2 = 0$ and $d_1 = n$. Therefore, the expression for the expected maximum matching size is reduced to $m - \left(m \cdot \left(1 - \frac{1}{m}\right)^n\right)$. Thus,

$$\begin{aligned} \gamma_a &= \lim_{n\to\infty} \frac{\mu(G_a)}{n} = \lim_{n\to\infty} \frac{1}{n} \cdot \left(m - \left(m \cdot \left(1 - \frac{1}{m}\right)^n\right)\right) \\ &= \frac{1}{\alpha} - \frac{1}{\alpha} \cdot e^{-\alpha}. \end{aligned}$$

■

N. *Proof of Corollary 9*

We show that γ_a is strictly monotonically increasing, thus $\gamma_a < 1$ for $1 \leq a < 2$, since $\gamma_a = 1$ for $a = 2$. This is shown by differentiating γ_a with respect to a:

$$\frac{d\gamma_a}{da} = -\frac{1}{4\alpha^2 (a-1)^2} \cdot \left(W\left(-2\alpha(a-1) \cdot e^{-a\alpha}\right) + 2\alpha(a-1)\right) \cdot W\left(-2\alpha(a-1) \cdot e^{-a\alpha}\right)$$

Both the first factor $-\frac{1}{4\alpha^2(a-1)^2}$ and the third factor $W\left(-2\alpha(a-1) \cdot e^{-a\alpha}\right)$ are negative. Thus, if the second factor is positive then $\frac{d\gamma_a}{da}$ is an increasing function with respect to $a \in [1,2)$.

If $\alpha > 0.5$, then $2\alpha(a-1) > 1$, and since $W(x)$ is minimized for $x = -\frac{1}{e}$ where it equals -1, the second factor is positive. On the other hand, consider that $\alpha \leq 0.5$. Since $W\left(-2\alpha(a-1) \cdot e^{-2\alpha(a-1)}\right) = -2(a-1)\alpha$ and $W(x)$ is an increasing function, then we have to show that $-2\alpha(a-1) \cdot e^{-2\alpha(a-1)} < -2\alpha(a-1) \cdot e^{-a\alpha}$, that is, $-2\alpha(a-1) > -a\alpha$. The last inequality can easily be shown for $1 \leq a < 2$. ■

O. *Proof of Theorem 10*

We compute the limit of $\frac{\mu(G_p)}{n}$ as $n \to \infty$.

$$\begin{aligned} \gamma_p &= \lim_{n\to\infty} \frac{\mu(G_p)}{n} \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{d_2=0}^{n} \binom{n}{d_2} \cdot p^{d_2} \cdot (1-p)^{n-d_2} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) \end{aligned}$$

Let $X \sim \text{Bin}(n,p)$ be the random variable counting the number of vertices in L that choose 2 vertices in R. By summing over three disjoint ranges of possible values for d_2, we get

$$\begin{aligned} \gamma_p = & \lim_{n\to\infty} \sum_{d_2=0}^{\lfloor np-n^{\frac{3}{4}}\rfloor} \Pr\{X=d_2\} \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) + \\ & \lim_{n\to\infty} \sum_{d_2=\lfloor np-n^{\frac{3}{4}}\rfloor+1}^{\lfloor np+n^{\frac{3}{4}}\rfloor-1} \Pr\{X=d_2\} \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) + \\ & \lim_{n\to\infty} \sum_{d_2=\lfloor np-n^{\frac{3}{4}}\rfloor}^{n} \Pr\{X=d_2\} \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) \end{aligned}$$

By Chebyshev's inequality we get that $\Pr\left\{|X-np| > n^{\frac{1}{4}}\sqrt{np(1-p)}\right\} \leq \frac{1}{n^{\frac{1}{4}}}$. Since $p(1-p) \leq 1$, we get that $\Pr\left\{|X-np| > n^{\frac{3}{4}}\right\} \leq \frac{1}{n^{\frac{1}{4}}}$. By the fact that $\frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) \leq 1$, we find that the first and the third limits go to zero.

Since the function $\mu(G_a)$ is increasing with respect to a (this can be shown by a simple combinatorial argument), we get the following lower bound:

$$\begin{aligned} \gamma_p &= \lim_{n\to\infty} \sum_{d_2=\lfloor np-n^{\frac{3}{4}}\rfloor+1}^{\lfloor np+n^{\frac{3}{4}}\rfloor-1} \Pr\{X=d_2\} \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) \\ &\geq \lim_{n\to\infty} \left(1 - \frac{1}{n^{\frac{1}{4}}}\right) \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{\lfloor np-n^{\frac{3}{4}}\rfloor+1}{n}}\right) \end{aligned}$$

as well as the following upper bound:

$$\begin{aligned} \gamma_p &= \lim_{n\to\infty} \sum_{d_2=\lfloor np-n^{\frac{3}{4}}\rfloor+1}^{\lfloor np+n^{\frac{3}{4}}\rfloor-1} \Pr\{X=d_2\} \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{d_2}{n}}\right) \\ &\leq \lim_{n\to\infty} 1 \cdot \frac{1}{n} \cdot \mu\left(G_{a=1+\frac{\lfloor np+n^{\frac{3}{4}}\rfloor-1}{n}}\right). \end{aligned}$$

By the squeeze theorem, we get the claimed result. ■

P. *Proof of Theorem 18*

We first establish a few lemmas before proving the result. As before, we start by considering a deterministic bipartite graph $H = \langle L_H + R_H, E_H\rangle$ with degree d of each vertex in L_H, where $|L_H| = s$ and $|R_H| = q$.

Lemma 8. *If $(d-1) \cdot s \leq q-2$, then H is not connected.*

Proof: As in the proof of Lemma 1, the proof follows by induction on s. For $s = 1$, there are d edges in the graph and therefore every graph with $q \geq d+1$ is not connected. Assuming that the claim holds up until $s = s'$, we next prove that it holds for any bipartite graph H' such that $|L_{H'}| = s'+1$ and $|R_{H'}| \geq (d-1) \cdot (s'+1) + 2$. Assume towards a contradiction that there is a graph H' which is connected.

We first show that there are $d-1$ vertices $v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}$ in $R_{H'}$, all of a degree 1 such that they are connected to the same vertex $v_\ell \in R_{H'}$: The sum of right-side vertex degree is $d \cdot (s'+1)$. Also, since the graph is connected there are no right-side vertices with degree 0. This implies that there are at least $(d-2)\cdot(s'+1)+2$ vertices of degree 1, thus there exists a vertex $v_\ell \in R_{H'}$ as claimed.

By the induction hypothesis, the graph induced by $L_{H'} \setminus \{v_\ell\}$ and $R_{H'} \setminus \{v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}\}$ is not connected, which implies that it has at least two connected components. In H', v_ℓ is connected to all vertices $v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}$. Since its degree is d it can be connected only to one of these components. This implies that H' is not connected as well, and the claim follows. ■

Lemma 9. *If H is connected and $(d-1)\cdot s = q-1$ then $\mu(H) = s$.*

Proof: Assume towards a contradiction that $\mu(H) < s$, and consider some maximum matching M. Let $v_\ell \in L_H$ be a vertex that is not in the maximum matching M, and $v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}$ be the vertices in R (which are not necessarily distinct) that are connected to v_ℓ. All vertices $v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}$ are connected also to another vertex in L_H, otherwise v_ℓ was in the maximum matching M.

Consider the bipartite graph $\hat{H} = \left\langle \hat{L}_H + \hat{R}_H, \hat{E}_H \right\rangle$, which is given by removing v_ℓ from H. Since the right-side vertices $v_{r_1}, v_{r_2}, \ldots, v_{r_{d-1}}$ are also connected to the other left-side vertices (except v_ℓ), the bipartite graph $\hat{H}$ is connected. However, we get that $\left|\hat{L}_H\right| = s-1$ and $\left|\hat{R}_H\right| = (d-1)\cdot s+1$, which contradicts with Lemma 8. ■

We note that in contrast to Lemma 2, the corresponding proposition is not true for $d > 2$; that is, if H is connected and $s \le q$, then the maximum matching size is not necessarily s. As a counter example, consider the case where $d = 3$ and $s = q = 3$, where two left-side vertices choose the same single right-side vertex (using all their 3 choices), and the other left-side vertex chooses all 3 right-side vertices. The resulting bipartite graph is clearly connected, but the maximum matching size is only 2 (only one of the first two left-vertices can be in the matching).

Lemma 10. *If $(d-1)\cdot s = q-1$ then H is connected if and only if it is a tree.*

Proof: The proof consists of the exact same construction $\hat{H}$ as in the proof of Lemma 4, where we eventually get a contradiction with Lemma 8. ■

Lemma 11. *The number T_s^d of connected bipartite graphs H whose $|L_H| = s$ and $|R_H| = 2(d-1)\cdot s+1$ is $T_s^d = \frac{((d-1)\cdot s+1)!}{((d-1)!)^s}((d-1)\cdot s+1)^{s-2}$.*

Proof: By Lemma 10, we have to count the number of bipartite trees over the two disjoint sets L_H and R_H of size s and $(d-1)\cdots+1$. Since H is a tree, then there are no cycles. Consequently, each one of the vertices in L_H is connected to d distinct vertices in R_H. Moreover, no two vertices in L_H share more than 1 vertex in R_H. For each vertex $v_\ell \in L_H$, let S_v be the set of the d right-side vertices that v_ℓ is connected to and also let the cycle C_{v_ℓ} be a cycle that consists of the d vertices of S_v.

Consider the graph $\hat{H} = \left\langle \hat{R}_H, \hat{E}_H \right\rangle$, which is given by connecting each cycle $C_{v_{\ell_1}}$ to $C_{v_{\ell_2}}$ using a common vertex v_r if and only if v_r is connected to both v_{ℓ_1} and v_{ℓ_2}. The resulting graph $\hat{H}$ is a Husimi graph over $(d-1)\cdot s+1$ vertices, where the number of such (labeled) graphs is $\frac{((d-1)\cdot s+1)!}{((d-1)!)^s\cdot s!}((d-1)\cdot s+1)^{s-2}$ [42].

Finally, each set S_v is determined by the (labeled) vertex in R_L. Thus, we multiply by $s!$ the above expression. ■

We are now able to prove the result.

Let M be a maximum matching of G. Similarly to the proof of Theorem 1, the proof is based on counting the expected number of vertices in R that are not part of M, and on the decomposition of G into its connected components.

We count the expected number of connected components with s left-side vertices and $q = (d-1)\cdot s+1$ right-side vertices. By Lemma 9, the maximum matching size of each such connected component is exactly s. Thus, there are $q-s$ right-side vertices that are not in M.

Let H be a bipartite graph $H = \langle L_H + R_H, E_H\rangle$, with degree d for all vertices in L_H, where $|L_H| = s$ and $|R_H| = q$. The probability P_s that H is connected is given by $P_s = \frac{(d!)^s T_s^d}{q^{d\cdot s}}$.

The remainder of the proof is similar to the proof of Theorem 1. ■

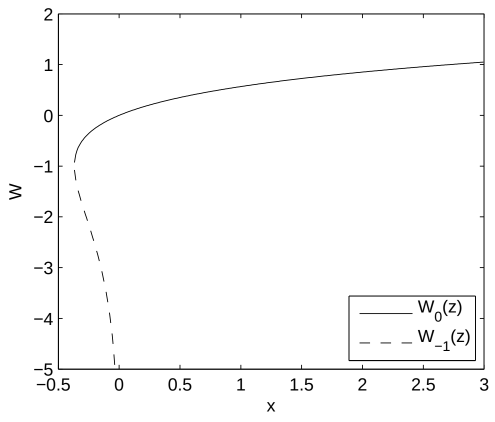


Fig. 11. The Lambert-W function.

Appendix B
The Lambert-W Function

The Lambert-W function, usually denoted by $W(\cdot)$, is given by the following implicit representation:

$$z = W(z)\cdot e^{W(z)},$$

where z is a complex number [35].

For real valued arguments, i.e. z is real valued, $W(z)$ has two real-valued branches: the principal branch, denoted by $W_0(\cdot)$ and the branch $W_{-1}(\cdot)$. Figure 11 shows the two real-valued branches. For instance, $W_0(-e^{-1}) = W_{-1}(-e^{-1}) = -1$ and $W_0(0) = 0$.

Note that the notation $W(\cdot)$ usually relates to the principle branch, i.e. $W_0(\cdot)$. Thus, although one would expect that for real-valued z, $W(z\cdot e^z) = z$, this is only the case for $z \ge -1$; in case $z < -1$, $W_{-1}(z\cdot e^z) = z \ne W(z\cdot e^z)$.