Balanced Allocation and Dictionaries with Tightly Packed Constant Size Bins

(Extended Abstract)

Martin Dietzfelbinger[*] Christoph Weidling[†]

July 18, 2005

Abstract

We study a particular aspect of the balanced allocation paradigm (also known as the "two-choices paradigm"): constant sized bins, packed as tightly as possible. Let $d \geq 1$ be fixed, and assume there are m bins of capacity d each. To each of $n \leq dm$ balls two possible bins are assigned at random. How close can $dm/n = 1+\varepsilon$ be to 1 so that with high probability each ball can be put into one of the two bins assigned to it, without any bin overflowing? We show that $\varepsilon > (2/e)^{d-1}$ is sufficient. If a new ball arrives with two new randomly assigned bins, we wish to rearrange some of the balls already present in order to accommodate the new ball. We show that on average it takes constant time to rearrange the balls to achieve this, for $\varepsilon > \gamma \cdot \beta^d$, for some constants $\gamma > 0$, $\beta < 1$. An alternative way to describe the problem is in data structure language. Generalizing cuckoo hashing (Pagh and Rodler, 2001), we consider a hash table with m positions, each representing a bucket of capacity $d \geq 1$. Keys are assigned to buckets by two fully random hash functions. How many keys can be placed in these bins, if key x may go to bin $h_1(x)$ or to bin $h_2(x)$? Our results lead to an implementation of a dynamic dictionary that accommodates n keys in $m = (1+\varepsilon)n/d$ buckets of size $d = O(\log(1/\varepsilon))$, so that key x resides in bucket $h_1(x)$ or $h_2(x)$. If $d \geq 1 + 3.26 \cdot \ln(1/\varepsilon)$, then for a lookup operation only two hash functions have to be evaluated and two contiguous segments of d memory cells have to be inspected. The expected time for inserting a new key is constant, for some $d = O(\log(1/\varepsilon))$.

1 Introduction: Bounded Balanced Allocation, d-Orientability, and Blocked Cuckoo Hashing

In this paper, we study a data allocation problem that can be described in different terminologies, and aspects of which have been considered in different contexts.

In the **"balanced allocation paradigm"** (also known as the "two-choices paradigm") we have n balls and m bins. To each ball two bins are assigned at random. Each ball is to be placed into one of the two bins assigned to it; the aim is to keep the maximum load in the bins small. Much work has been devoted to analyzing the online version of this experiment, where the balls arrive one after the other and are put into a bin upon arrival, and this placement can never be changed later. Not that much is known about the offline version, where all balls with their two choices are given as

[*]Corresponding author, Technische Universität Ilmenau, 98684 Ilmenau, Germany, email: martin.dietzfelbinger@tu-ilmenau.de

[†]sd&m AG, 63065 Offenbach am Main, Germany (affiliated with the Technische Universität Ilmenau while this work was done), email: christoph.weidling@sdm.de

input. In particular, it has been known for quite a while that for $n = O(m)$ with high probability[1] constant bin size is enough to place all balls. Other authors have considered "dynamic" situations in which the positions of balls may change in the course of a random process.

In this paper, we ask how well we can utilize the space in the bins if we fix the capacity of the bins to some number $d \geq 1$. That means, given n, we wish to keep the space overhead $\varepsilon = \frac{dm-n}{n}$ as small as possible and still be able to place the balls w.h.p. We show that $\varepsilon > (2/e)^{d-1}$ is sufficient to guarantee this. We also consider a dynamic version of the problem. Assume that n balls are placed and a new ball arrives, with two randomly assigned bins. We show that the expected time needed to rearrange the balls so that the new ball can be accommodated as well is constant, as long as $\varepsilon > \gamma \cdot \beta^d$, for some constants $\gamma > 0$, $\beta < 1$. This implies that the expected time to place n balls is $O(n)$. No estimates for the density that can be achieved asymptotically for fixed bin size d larger than some small constant have been known before, although the question arises in several applications. (Details below.)

It is easy to see that the allocation problem with bounded bin size d is equivalent to a version of the ***d*-orientability problem** [12] for random graphs. We say an undirected graph $G = (V, E)$ of m nodes and n edges is *d-orientable* if the edges can be directed in such a way that every node has outdegree not larger than d. Given a constant d, we ask for upper and lower bounds on the edge density $\frac{n}{m}$ (certainly $< d$) so that a graph with n randomly placed edges (including loops and multiple edges) is d-orientable w.h.p. Also, we ask how long it takes to adapt a given edge orientation when a new random edge arrives.

Yet another formulation of the same problem can be given in **data structure** language. We wish to implement dynamic dictionaries so that constant lookup time is guaranteed. Dynamic dictionaries store keys from a universe U (possibly together with satellite data) and support the operations *insert*, *delete*, and *lookup*. Pagh and Rodler's **"cuckoo hashing"** method [18] assumes that each one of n keys is assigned to two locations $h_1(x)$ and $h_2(x)$ in a hash table of size m, and can be stored in one of the two locations. Each location has capacity 1. We generalize this approach by considering buckets of capacity d, for some arbitrary constant $d \geq 1$. Our construction results in the following. Assuming that fully random hash functions are available, we obtain an implementation of a dynamic dictionary that for given $\varepsilon > 0$ stores n keys in space $(1 + \varepsilon)n$ in such a way that a lookup for x requires evaluating two hash functions and probing two contiguous blocks of d memory cells. The expected cost of inserting a new key is $(1/\varepsilon)^{O(\log\log(1/\varepsilon))}$. This compares favorably with the performance of "*d-ary cuckoo hashing*", a different generalization of cuckoo hashing by Fotakis, Pagh, Sanders, and Spirakis [8]. There $d = O(\log(1/\varepsilon))$ independent hash functions are used to achieve a similar space utilization. The access procedure of our scheme is more local and hence more suited for cache architectures. Experiments support the hope that the new scheme is competitive with d-ary cuckoo hashing [18] as far as space utilization is concerned, and allows faster accesses. (The experiments are not part of this submission. The interested reader may find a description of some of the results in Appendix E.)

Remark 1. For reference, we describe the connection between the hashing formulation and the d-orientability formulation from [2, 12, 20] in detail. Given is a set S of n keys and two random hash functions h_1, h_2. We consider a random (multi)graph $G_u = (V, E_u)$ with labeled edges. The node set is $V = [m] = \{0, \ldots, m-1\}$, the set of labeled edges is $E_u = E_u(S, h_1, h_2) = \{\{h_1(x), h_2(x)\} \mid x \in S\}$. We say that G_u is *d-orientable* if the n edges in E_u can be directed in such a way that each node has outdegree at most d. Assigning such directions to edges is equivalent to storing the keys in a table with m blocks with maximal load d, as follows: the edge $\{y, y'\} = \{h_1(x), h_2(x)\}$ is directed from y to y' if and only if x is stored in block y. Below, any directed version of G_u with

[1]In the sequel, we write "w.h.p." for "with high probability", which means "with probability $1 - \frac{1}{\mathrm{poly}(n)}$".

outdegree bounded by d will be called $G = (V, E)$.

Terminology in this paper. We have chosen to describe the algorithms and the analysis in the language of implementing a dictionary as a hash table with two functions. This also makes it possible to discuss a new solution to the problem caused by the assumption that fully random hash functions are available for free, see Section 2.3. However, all results readily translate into the terminology of the balanced allocation paradigm or the graph orientation paradigm.

1.1 Background and related work

Early contributions to our version of the allocation problem (fixed sized bins) were made in connection with the balanced allocation paradigm. (For a survey of this area, see [14].) In the seminal paper [1] by Azar, Broder, Karlin, and Upfal it was noted [9] (also see [2]) that if a set of $n \leq 1.67m$ balls is allocated to m cells, with two choices per ball, then with high probability the keys can be placed so that no bin holds more than two balls. This immediately extends to a scheme for storing n balls in m bins with a maximum load of $d = 2 \cdot \lceil n/(1.67m) \rceil \leq 2 + 1.2\frac{n}{m}$, which for m, n large corresponds to a space overhead of $\varepsilon = 0.2$ in our notation.

Simultaneously, observations concerning the existence of such placements were made in papers on the simulation of parallel random access machines on distributed memory machines by redundantly storing data (e.g., [3, 13]). In particular, it was shown there that a maximum load of $O(1 + n/m)$ is achievable with high probability, even if for allocating the bins hash functions from classes described in [23] are used.

Sanders et al. [21, 22] studied the static allocation problem with fixed bin sizes as the combinatorial abstraction of a scheme called "Randomized Duplicate Allocation (RDA)", used for storing data blocks on disks. In [22] it was shown that with high probability a bin size of $d = 1 + \lceil n/m \rceil$ is sufficient (this would correspond to a bound $d \geq \frac{1}{\varepsilon}$ in our notation). In [21] the question was asked how close n/m might be to $d = \lceil n/m \rceil$ so that still block size d is sufficient. For this, the criterion given as (2) below was derived and limiting values were determined by inspecting the corresponding functions for $d = 2, 3, \ldots, 9$ separately. No asymptotic relation between ε and d was derived. We close this gap, and moreover show how to calculate an assignment in expected linear time where [21, 22] suggested using more expensive maxflow computations.

In [2] the online version of the case of heavily loaded bins (i.e., $\frac{n}{m} \to \infty$) was studied. In [4] it was demonstrated that perfect balance ($n = dm$, with no slack at all) is impossible w.h.p. if $d < \gamma_1 \ln n$ for a suitable constant γ_1, while perfect balance is possible w.h.p. if $d > \gamma_2 \ln n$ for some larger constant γ_2. For constant d, no better bound than in [22] was given. Also in [4], a randomized rebalancing procedure was described and analyzed, with running time polynomial in n.

On the data structures side, the paper by Pagh and Rodler [18] showed that $m = (2 + \varepsilon)n$ cells are enough if each cell may have load 1 and a key x may be stored in one of the locations given by two hash functions. It is not hard to see, using the threshold for the appearance of other than unicyclic components in random graphs, that it is impossible to decrease the space below $2n$ and still run cuckoo hashing in this original form.

As a remedy for this situation, Fotakis et al. [8] suggested "d-ary cuckoo hashing". The scheme they study (in the language of implementing dictionaries) amounts to the balanced allocation problem with bin size 1 for the case where each ball has d random bins it can go to. Fotakis et al. [8] show that with n balls and $m = (1 + \varepsilon)n$ bins it is necessary to have $d = \Omega(\log(1/\varepsilon))$ and sufficient to have $d = O(\log(1/\varepsilon))$ for it being possible to place the keys w.h.p., and that some $d = O(\log(1/\varepsilon))$ is sufficient to arrive at an insertion procedure that adds one new ball with d new random locations in expected constant time. This leads to an implementation of a dictionary for

n keys in space $(1+\varepsilon)n$, where a lookup requires evaluating d hash values and probing d random locations in the worst case, where $d = O(\log(1/\varepsilon))$. Inserting a key takes expected constant time. From the point of view of efficient implementation of data structures, our result leads to a comparable space utilization, but has the advantage that only two hash functions have to be evaluated and two contiguous blocks of d memory cells must be probed in a search. (This approach is advantageous in architectures with caches; which could be observed also in experiments that we have conducted. The different time requirements for searches would become clear if we counted the number of cache faults in the cache-oblivious model [10]. In comparision to d-ary cuckoo hashing we get a reduction in the number of cache faults by a factor of B where B is the cache block size. We do not give the obvious details.)

Recently, Panigrahy [19] studied the dynamic version of the allocation problem (in the formulation for dynamic hash tables with insertions) for two choices and bin size $d = 2$. He established, by analyzing related branching processes, that inserting keys is possible in expected constant time as long as $n \leq 1.67m$ — the same bound as given for the static case in [1]. Our results are derived by totally different methods; they are not quite as tight for $d = 2$, but achieve a much better space utilization already if d is chosen only a little larger.

In the analysis of the static case we start with the condition (2) on d and ε that has been noted already in [21]. The transformation of this condition into the general relation $d > 1 + \frac{\ln(1/\varepsilon)}{1-\ln 2}$ by means of calculus has escaped other researchers until this date. Concerning the dynamic case, where we ask for the cost of adding one ball (or inserting one key), the basic structure of our analysis is the same as in [8], which is based upon ideas from [15]. We show that the random graph generated by the n bins as nodes and the two possible locations for the n keys as undirected edges has certain expansion properties, and derive upper bounds on the probability that a random node is at a great distance to all nonfull nodes in the directed version of the graph. In comparison to the analysis in [8], quite a few extra technical obstacles have to be overcome. (For lack of space, not all details of this analysis are given in this extended abstract; they can be found in the appendix.)

2 Detailed Overview

In this section we describe the data structures, the algorithms, and our results.

2.1 The Static Case

A set S of n keys from the universe U is to be stored. We use an array $T[0..m-1]$ consisting of $m = n(1+\varepsilon)/d$ blocks (subarrays) of d cells each. (For convenience we assume that $n(1+\varepsilon)/d$ is an integer.) Inside each block we store up to d keys sequentially. Given two hash functions $h_1, h_2\colon U \to [m]$, we say that h_1, h_2 are *suitable* for S and d if it is possible to store each key x from S in one of the blocks $h_1(x), h_2(x)$ without any block receiving more than d keys. If the keys from S are stored according to h_1, h_2, a *lookup procedure* is obvious, which involves evaluating two hash values and searching two blocks. — Our first result:

Theorem 1. *Let $\varepsilon > 0$ be arbitrary. Assume that $d \geq 1 + \frac{\ln(1/\varepsilon)}{1-\ln 2}$. Let n be sufficiently large, let $S \subseteq U$ be an arbitrary set of n keys, and let T be a table with m blocks of size d each, where $dm \geq (1+\varepsilon)n$. Further assume that $h_1, h_2\colon U \to [m]$ are fully random hash functions. Then with probability $1 - O(1/m^{d-1})$ the functions h_1, h_2 are suitable for S and d.*

Remark. The bound on d stated in the theorem is not very far away from the true threshold. Indeed, using standard methods the following can be shown by estimating the number of buckets hit by exactly $d-1$ keys: There is a constant c such that if $d + c\ln d < \frac{\ln(1/\varepsilon)}{1-\ln 2}$ then w.h.p. h_1, h_2 are not suitable for S.

The proof of Theorem 1 is outlined in Section 3. It starts from a known characterization of h_1, h_2 being suitable [21, 22], but in contrast to all earlier approaches carries through a full analysis of the resulting estimates to obtain a closed bound valid for all d.

2.2 Updates: The Cuckoo Insertion Procedure

Assume n keys are stored in the table T according to h_1, h_2, with blocks of size d.

Inserting a new key x can best be described in terms of the directed graph G from Remark 1. In G, find a directed path $y_0, y_1, \ldots, y_\ell$ with $y_0 \in \{h_1(x), h_2(x)\}$ and y_ℓ a node that is "*free*", i.e., has outdegree smaller than d. (This means that block y_ℓ contains an empty cell.) The edges that form the path correspond to keys $x_1, \ldots, x_\ell$ such that x_i is stored in y_{i-1}, but may be stored in y_i. After moving x_i from y_{i-1} to y_i, for $1 \le i \le \ell$ (this corresponds to flipping the edges on the path), node (block) y_0 is free, and hence we can store x there.

We call a path $y_0, y_1, \ldots, y_\ell$ as described an "augmenting path" for G and x.

It is very easy to see that if the keys from $S \cup \{x\}$ can be stored according to h_1, h_2 at all, then there is an augmenting path. So the problem is to find a short augmenting path fast. As proposed in [8], a simple approach for this is *breadth-first-search* (*BFS*) in G, starting from $\{h_1(x), h_2(x)\}$. The time for this is proportional to the number of edges probed before a free node is found. Since the nodes in the part of G that is searched have outdegree d, this number is not larger than $2(d + d^2 + \cdots + d^\ell) < 4d^\ell$, where ℓ is the length of a *shortest* path from $\{h_1(x), h_2(x)\}$ to a free node. Thus we will have to analyze (the distribution of) the distance between $\{h_1(x), h_2(x)\}$ and the set of free nodes.

Our second main result runs as follows:

Theorem 2. *Let $\varepsilon > 0$ be arbitrary. Assume that $d \ge 90.1 \cdot \ln(1/\varepsilon)$. Let n be sufficiently large, let S an arbitrary set of n keys, let $x \in U - S$, and let T be a table with m blocks of size d each, where $dm \ge (1+\varepsilon)n$. Assume that $h_1, h_2 \colon U \to [m]$ are fully random hash functions, and that the keys from S have been stored in T by an algorithm that is ignorant of $h_1(x), h_2(x)$. Then the expected time to insert x by the BFS procedure is $(1/\varepsilon)^{O(\log d)}$.*

The proof is outlined in Section 4. The constants in the bound are certainly not optimal. In this extended abstract, we do not address the issue of extra space needed by the BFS, for the following reasons: First, it is possible by simple and not very interesting modifications to the straightforward BFS to get by with $O(n^{1-\zeta})$ extra space in the worst case, for $\zeta > 0$ constant. Even a full rehashing (choosing new hash functions h_1 and h_2 and rearranging all keys) can be done "in place". Second, if the technique described in Section 2.3 below is used, no more than $n^{2/3}$ keys have to be handled at any time, so the scratch space problem vanishes altogether.

An alternative approach to insertion (also suggested in [8]) is to search an augmenting path by a certain kind of random walk in G, as follows: Assume x is to be inserted. Repeat the following, starting with $z := x$:

> Calculate $h_1(z)$ and $h_2(z)$. If one of the two blocks $h_1(z)$ or $h_2(z)$ is not full, store z in one such block, and stop. (Ties are broken arbitrarily.) If both blocks are full, randomly choose one of the keys stored in these blocks, call it z', kick z' out from its block (this is the "cuckoo step") and insert z in its place. Let $z := z'$, and start again.

Many variations of this procedure are possible, e.g., one might try to insert z in the block it was not ejected from in the round before. Also, rules for stopping the loop have to be incorporated.

All implementations used in our experiments are based on this random walk idea, not on the BFS procedure. It is an intriguing open problem to provide an analysis of the random walk insertion procedure, if possible establishing a bound of $O(1/\varepsilon)$ on the expected number of blocks probed in the course of an insertion. (The same question is open for d-ary cuckoo hashing.)

2.3 Sharing Fully Random Hash Functions

For the analysis to carry through, we assume that the hash functions h_1, h_2 behave fully randomly on S. If in the course of inserting a key it turns out that h_1, h_2 are not suitable, we might want to rehash the whole set, using new hash functions h_1, h_2. We cannot rely on the set S of keys to provide this degree of randomness. Note that although in the balanced allocation literature the full randomness assumtion is routinely used, this is not the case in the hashing literature. Earlier work on hashing (e.g., [3, 8, 13, 18]) has, very carefully, pointed out ways of working around this problem, for example by using functions from high-performance universal classes like in [23]. (This would not be sufficient for d-ary cuckoo-hashing, though.) In [7, 17] it was demonstrated that full randomness can be simulated by universal hashing at the cost of $O(n)$ words of extra space. However, using such a construction would be unsatisfactory in our context, since we aim at getting by with εn extra space.

We propose the following workaround, which might be helpful also in other contexts. Let $\varepsilon > 0$ and n be given. Using high-performance hash classes [6, 23] we may choose a function $h\colon U \to [n^{1/3}]$ so that with probability $1 - n^{-c}$ (for some constant c) the set S is split into $n^{1/3}$ pieces $S_i = \{x \in S \mid h(x) = i\}$ of size $\le (1 + \frac{\varepsilon}{2})n^{2/3}$. (There is no need that S is known or the pieces are listed.) For each of the pieces S_i we run cuckoo hashing with blocks of size d in a separate table T_i of size $(1+\varepsilon)n^{2/3}$. It is an easy exercise, using polynomial hash functions and techniques described in [6], to provide a pair h_1, h_2 of hash functions that with high probability behaves fully randomly on one S_i, if we are allowed to use space $n^{5/6}$ for storing h_1, h_2. Since we may use the *same* hash function pair h_1, h_2 for all pieces S_i, $i = 0, \ldots, n^{1/3} - 1$, the overall space needed for the fully random functions is $O(n^{5/6}) = o(n)$. The algorithms described in the present paper then has to be applied to each of the pieces separately. (Details will be given in a forthcoming paper.)

3 Analysis for the Static Case

In this section we analyze the size of d needed for storing a set S of n keys in $m = (1+\varepsilon)n/d$ blocks of size d, if $h_1, h_2\colon U \to [m]$ are fully random, thus proving Theorem 1. We set out as in [8, 21, 22]: A set S of n keys can be stored in $m = (1+\varepsilon)n/d$ blocks of size d if and only if for every $X \subseteq S$ it holds that $|\Gamma(X)| \ge \frac{1}{d}|X|$, where $\Gamma(X) = \{h_1(x), h_2(x) \mid x \in X\}$. (This may be seen by applying Hall's marriage theorem [5, p. 31] to the bipartite graph $(S, [dm], E)$, $E = \{(x, d \cdot h_i(x) + j) \mid x \in S, i \in \{1,2\}, 0 \le j < d\}$.) Thus, to prove Theorem 1 it is sufficient to establish an upper bound on the probability F of the event that there is some $X \subseteq S$ such that $|\Gamma(X)| < \frac{1}{d}|X|$.

Lemma 1. *If $\varepsilon \le 0.25$ and $d > 1 + \frac{\ln(1/\varepsilon)}{1-\ln 2}$, then $F = O(m^{1-d})$.*

Outline. For $1 \le j \le m/(1+\varepsilon)$, let $F(j)$ be the probability that there is a set Y of j blocks such that some set $X \subseteq S$, $|X| = dj$, satisfies $\Gamma(X) \subseteq Y$. Clearly, $F \le \sum_{1 \le j \le m/(1+\varepsilon)} F(j)$. Using the

Chernoff-Hoeffding bound (13) and the binomial bound (11), we get

$$F(j) \leq \binom{m}{j}\left(\frac{n(j/m)^2}{jd}\right)^{jd}\left(\frac{n-n(j/m)^2}{n-jd}\right)^{n-jd}$$
$$\leq (1+\varepsilon)^{-jd} m^{\frac{m(1+\varepsilon-d)}{1+\varepsilon}} j^{j(d-1)} (m-j)^{j-m}\left(\frac{m^2-j^2}{m-j(1+\varepsilon)}\right)^{d\frac{m-j(1+\varepsilon)}{1+\varepsilon}}, \quad (1)$$

which was already observed in [21, 22]. We examine the expression on the right-hand side of (1), which we abbreviate by $f(j,\varepsilon)$, in different ranges of j. For $j=1$ we find $f(1,\varepsilon)=O(m^{1-d})$. — Now assume that j, $2 \leq j < e^{-4}m$, is fixed. We prove that $f(j,\varepsilon)$ is a decreasing function of ε; thus, we can concentrate on the case $\varepsilon=0$. The sequence $f(j,0)$, $j=2,\ldots,\lfloor e^{-4}m\rfloor$ turns out to be geometrically decreasing; hence we get $\sum_{2\leq j\leq e^{-4}m} F(j) = O(m^{2-2d})$. — The most involved calculation concerns the range $e^{-4}m \leq j \leq (1-2\varepsilon)m$. Here we read off from (1) by substituting $j=\alpha m$, that $f(j,\varepsilon)=O(c^m)$ for a constant $c<1$, if

$$d > \frac{\alpha\ln\alpha+(1-\alpha)\ln(1-\alpha)}{\alpha(\ln\alpha-\ln(1+\varepsilon))+\frac{(1-\alpha(1+\varepsilon))(\ln(1-\alpha^2)-\ln(1-\alpha(1+\varepsilon)))}{1+\varepsilon}}, \quad (2)$$

for $e^{-4} \leq \alpha \leq 1-2\varepsilon$. We prove that the right-hand side of (2) is bounded by

$$g(\alpha) := \frac{\alpha\ln\alpha+(1-\alpha)\ln(1-\alpha)}{\alpha\ln\alpha+(1-\alpha)\ln(1+\alpha)} \quad (3)$$

By calculus we show that g is an increasing function and that $g(1-2\varepsilon) < 1-\frac{\ln\varepsilon}{1-\ln 2}$. — For the range $j > (1-2\varepsilon)m$, we observe that for each $\varepsilon \leq 0.25$ the right-hand side of (2) is decreasing in α, if $1-2\varepsilon \leq \alpha \leq 1/(1+\varepsilon)$. (Details may be found in the appendix, Section B.) □

4 The Expected Insertion Time

We want to examine the expected time of the BFS algorithm for inserting a new key x in T, as described in Section 2.2 (also recall Remark 1). Just before x is inserted, a set S of n keys is stored in T. We assume that the directed graph G determined by this placement is independent of the hash values $h_1(x), h_2(x)$ — this is the case if x was never inserted before. We start a BFS in G from $\{h_1(x), h_2(x)\}$ with the aim of finding a shortest path to a node in

$$Y_0 := \{y \in V \mid y \text{ is free in } G\}.$$

The time for the BFS in G is $O(\min\{n, d^{\ell+1}\})$, where ℓ is the length of such a shortest path. Our aim is to see that the expectation of the number of edges we have to inspect before a free node is found is $O(1)$. For this, we analyze the distribution of the number of nodes at different distances to Y_0. (The analysis runs along the lines of that of [8], but we are dealing with quite a different graph.) — Recursively define, for $k \geq 1$:

$$\begin{aligned} X_k &:= \{x \in S \mid h_1(x) \in Y_{k-1} \text{ or } h_2(x) \in Y_{k-1}\} \text{ and} \\ Y_k &:= \{y \in [m] \mid \exists x \in X_k \colon x \text{ is stored in } y\}. \end{aligned}$$

We say that the keys of X_k "*hit*" Y_{k-1}. It is easy to see that Y_k is the set of nodes from which Y_0 can be reached in at most k steps in G. By the definitions, $Y_{k-1} \subseteq Y_k$, for $k=1,2,\ldots$. We will establish in a series of lemmas that with high probability the cardinalities $|Y_0|, |Y_1|, |Y_2|, \ldots$ grow at certain minimal rates in different ranges of k. These properties of G, which hold regardless of the decisions made by the algorithm that has stored S, are derived from expansion properties of the undirected graph G' (see Remark 1), which follow from h_1, h_2 being fully random. The full proofs of Lemmas 2, 4, and 6 may be found in the appendix.

Lemma 2. *Let $\varepsilon \leq 0.1$ and $d \geq 90.1 \cdot \ln(\frac{1}{\varepsilon})$. Then there is a constant $\beta < 1$ such that with probability $1 - O(m\beta^m)$ each set of blocks Y that satisfies $\frac{\varepsilon}{1+\varepsilon} m \leq r = |Y| \leq \frac{5}{13} m$ is hit by at least $\frac{4}{3} rd$ keys from S.*

Outline. A set of blocks Y, $|Y| = r$, is hit by $\frac{4}{3}rd$ keys with h_1 or h_2 if at most $n - \frac{4}{3}rd$ keys *avoid* Y with both hash functions h_1 and h_2. Let $F(r)$ denote the probability that there is a set Y of size r such that there are more than $n - \frac{4}{3}rd$ keys that avoid Y with both hash functions. Employing the Chernoff-Hoeffding bound (13), we obtain:

$$F(r) \leq \binom{m}{r} \left(\frac{n\left(1 - \frac{r}{m}\right)^2}{n - \frac{4}{3}rd} \right)^{n - \frac{4}{3}rd} \left(\frac{n - n\left(1 - \frac{r}{m}\right)^2}{\frac{4}{3}rd} \right)^{\frac{4}{3}rd}. \tag{4}$$

By $n = dm/(1+\varepsilon)$, the binomial bound (11), and the substitution $r = \alpha m$ we observe that $F(r) < \beta^m$ for a suitable $0 < \beta < 1$, provided that

$$d > \frac{\alpha \ln \alpha + (1-\alpha)\ln(1-\alpha)}{\left(\frac{1}{1+\varepsilon} - \frac{4}{3}\alpha\right)\left(2\ln(1-\alpha) - \ln\left(1 - \frac{4(1+\varepsilon)\alpha}{3}\right)\right) + \frac{4\alpha}{3}\ln\left(\frac{3(2-\alpha)}{4(1+\varepsilon)}\right)} \tag{5}$$

for all α between $\frac{\varepsilon}{1+\varepsilon}$ and $\frac{5}{13}$. By a detailed analysis, the right-hand side of (5) can be shown to be bounded by $90.1\ln(1/\varepsilon)$, for $0 \leq \varepsilon \leq 0.1$ and $\frac{\varepsilon}{1+\varepsilon} \leq \alpha \leq \frac{5}{13}$. This implies the lemma. (The constant 90.1 may be replaced by smaller numbers if the range of ε is reduced. For details, see the proof in Appendix C.) $\square$

Lemma 3. *If the digraph G induced by S being stored meets the conclusion of Lemma 2, then there is some $k^* \leq 2 + \log_{\frac{4}{3}}\left(\frac{5(1+\varepsilon)}{13\varepsilon}\right)$ such that $|Y_{k^*}| > \frac{m}{2}$.*

Proof. Assume $k \geq 1$ and $|Y_{k-1}| \leq \frac{5}{13}m$. Consider the set X_k of keys that hit Y_{k-1}. By the assumption, $|X_k| \geq \frac{4}{3}d|Y_{k-1}|$. By definition, all keys $x \in X_k$ are stored in blocks in Y_k. Only $d|Y_{k-1}|$ of them can be stored in Y_{k-1}, so at least $\frac{1}{3}d|Y_{k-1}|$ of them must be stored in $Y_k - Y_{k-1}$, which implies that $|Y_k - Y_{k-1}| \geq \frac{1}{3}|Y_{k-1}|$; hence $|Y_k| \geq \frac{4}{3}|Y_{k-1}|$.

Now let k' be minimal with $|Y_{k'}| > \frac{5}{13}m$. By the above, it is easy to see that either $|Y_{k'}| > \frac{1}{2}m$ (then we let $k^* = k'$) or $|Y_{k'+1}| \geq \frac{4}{3} \cdot \frac{5}{13}m > \frac{1}{2}m$ (then we let $k^* = k'+1$). (This holds even if $5m/13 < |Y_{k'}| \leq m/2$, since then we apply the conclusion of Lemma 2 to a subset of $Y_{k'}$ of size $\lfloor 5m/13 \rfloor$.)

Because there are exactly εn free cells in the table, we have $|Y_0| \geq \frac{\varepsilon n}{d} = \frac{\varepsilon}{1+\varepsilon}m$. Thus, $|Y_k| \geq \frac{\varepsilon}{1+\varepsilon}m\left(\frac{4}{3}\right)^k$ for $0 \leq k < k'$, whence we get $k^* \leq 1 + k' \leq 2 + \left\lfloor \log_{\frac{4}{3}}\left(\frac{5(1+\varepsilon)}{13\varepsilon}\right)\right\rfloor$. $\square$

Lemma 4. *Let $d \geq 8$. Then there is some $\beta < 1$ such that with probability $1 - O(\beta^m)$ we have that each set Y of blocks with $\frac{m}{2} \leq |Y| \leq m - \frac{4m}{e^4 d^3}$ is hit by at least $n - \frac{9}{10}d(m - |Y|)$ keys.*

Outline. Let $\overline{Y} = [m] - Y$ and $r = |\overline{Y}|$. Again by the Chernoff-Hoeffding bound (13) and the binomial bound (11) we find that the probability that there is a set Y with $r = m - |Y|$ in $[\frac{4m}{e^4d^3}, \frac{m}{2}]$ such that more than $\frac{9}{10}dr$ keys hit $\overline{Y}$ with both hash functions, is bounded by

$$\frac{m^m}{r^r(m-r)^{m-r}}\left(\frac{10r}{9(1+\varepsilon)m}\right)^{\frac{9}{10}rd}\left(\frac{10(m^2-r^2)}{(10m-9r(1+\varepsilon))m}\right)^{\frac{dm}{1+\varepsilon}-\frac{9}{10}rd}. \tag{6}$$

We denote the expression in (6) by $f(r,\varepsilon)$. It is not hard to see that $f(r,\varepsilon)$ is decreasing in ε, thus we can concentrate on the case $\varepsilon = 0$. After replacing ε by 0 in (6), and substituting $\alpha = r/m$, an expression for a function $g(\alpha,d)^m$ results. By analyzing the partial derivative of g w.r.t. d we note that for each fixed α the function $g(\alpha,d)$ is decreasing in d. A look at a Maple plot (alternatively: an involved analysis) reveals that $g(\alpha,8)$ has exactly one minimum point with respect to α in the interior of the interval $[\frac{4}{e^4d^3}, \frac{1}{2}]$, so it does not exceed $\max\{g(\frac{4}{e^4d^3},8), g(0.5,8)\} = g(\frac{4}{e^4d^3},8) < 1$. (For the details see the full proof in Appendix D.) □

Lemma 5. *If G meets the conclusions of Lemmas 2 and 4, and k^* satisfies $|Y_{k^*}| \geq \frac{m}{2}$, then there is some ℓ^*, $\ell^* = O(\log d)$ such that $m - |Y_{k^*+\ell^*}| \leq \frac{4m}{e^4d^3}$.*

Proof. The straightforward induction proof, which uses Lemma 4, is omitted. □

The following lemma states a standard expansion property of bipartite random graphs, see [16, p. 109].

Lemma 6. *Let $d \geq 20$, and $\gamma = \frac{4}{e^4d^3}$. With probability $1 - O(m^{-d/2})$ we have that each set $X \subseteq S$ of keys with $d \leq |X| \leq \gamma dm$ hits more than $\Delta|X|$ different blocks, where $\Delta = \frac{1}{d^{1/3}} + \frac{1}{d}$.*

Proof. The probability that there is a set X of j keys, $d \leq j \leq \gamma dm$, that hits no more than Δj blocks can be bounded by

$$\sum_{d \leq j \leq \gamma dm}\binom{n}{j}\binom{m}{\lfloor \Delta j \rfloor}\left(\frac{\lfloor \Delta j \rfloor}{m}\right)^{2j} \leq \sum_{d \leq j \leq \gamma dm}\left(\frac{en}{j}\right)^{j}\left(\frac{em}{\Delta j}\right)^{\Delta j}\left(\frac{\Delta j}{m}\right)^{2j}. \tag{7}$$

(We used the simple binomial bound (12).) Straightforward simplifications, using that $d \geq 20$ implies that $\Delta < \frac{1}{2}$, lead to the bound $\sum_{d \leq j \leq \gamma dm}\left((e/2)^{3/2} \cdot d \cdot \sqrt{j/m}\right)^j$ for the right hand side in (7). For $j = d, d+1, \ldots, \lfloor \gamma dm \rfloor$ the terms in this sum are geometrically decreasing by a factor smaller than $\frac{1}{2}$, hence the sum is bounded by $O(m^{-d/2})$. □

We conclude that for $k \geq k^* + \ell^*$ the complements of the sets Y_k shrink fast.

Lemma 7. *Assume that $|[m] - Y_{k^*+\ell^*}| \leq \gamma m$ and that the hash functions h_1, h_2 meet the conclusion of Lemma 6. Then the cardinalities $a_j = |[m] - Y_{k^*+\ell^*+j}|$, $j = 0, 1, 2, \ldots$, satisfy $a_j \leq d^{-2/3} \cdot a_{j-1}$ for $j = 1, 2, 3, \ldots$. Hence,*

$$|[m] - Y_{k^*+\ell^*+j}| \leq \gamma d^{-2j/3}m, \textit{ for } j = 0, 1, 2, \ldots.$$

(In particular, there is some L with $Y_{k^*+\ell^*+L} \neq [m] = Y_{k^*+\ell^*+L+1}$.)

Proof. Fix $j \geq 1$. If $a_j = 0$, there is nothing to prove; thus assume $a_j \geq 1$. Then by the definitions, all a_j nodes in $[m] - Y_{k^*+\ell^*+j}$ are full. That means that the set E_j of edges in G with tails in $[m] - Y_{k^*+\ell^*+j}$ has cardinality da_j. By the assumption that the conclusion of Lemma 6 is satisfied the edges in E_j touch at least $\Delta da_j = (d^{2/3} + 1)a_j$ nodes overall. Only a_j of these nodes are in $[m] - Y_{k^*+\ell^*+j}$. By the definition of the sets Y_k, no edge in G can run from a node in $[m] - Y_{k^*+\ell^*+j}$ to a node in $Y_{k^*+\ell^*+(j-1)}$. Hence the heads of the edges in E_j hit at least $(d^{2/3} + 1)a_j - a_j = d^{2/3}a_j$ distinct nodes in $[m] - Y_{k^*+\ell^*+(j-1)}$, which implies that $a_{j-1} \geq d^{2/3}a_j$. □

Lemma 8. *Assume $Y_0, Y_1, \ldots, Y_{k^*}, \ldots, Y_{k^*+\ell^*}, \ldots, Y_{k^*+\ell^*+L}$ are fixed and fulfill the expansion properties from Lemmas 3, 5, and 7. Assume further that $h_1(x), h_2(x)$ are random values in $[m]$. Then the expected number of edges probed in the BFS insertion procedure for x is $(1/\varepsilon)^{O(\log d)}$.*

Proof. Let N_x be the number of edges of G probed when x is inserted. Let $\sigma_k = \sum_{0<\kappa\le k} d^\kappa < 2d^k$, for $k \ge 0$, and $k_x = \min\{k \mid h_1(x) \in Y_k \text{ or } h_2(x) \in Y_k\}$. Then the number of edges of G probed when inserting x is not larger than $2\sigma_{k_x}$. Thus, it is sufficient to estimate $\mathbf{E}(\sigma_{k_x})$. We have

$$\mathbf{E}(\sigma_{k_x}) = \sum_{q\ge1} \mathbf{Prob}(\sigma_{k_x} \ge q) = \sum_{k\ge1} \mathbf{Prob}(k_x \ge k) \cdot d^k. \tag{8}$$

The last sum in (8) is estimated in two pieces. We have

$$\sum_{1\le k\le k^*+\ell^*} \mathbf{Prob}(k_x \ge k) \cdot d^k \le (k^* + \ell^*) d^{k^*+\ell^*}. \tag{9}$$

For the rest of the sum in (8), we notice that by Lemma 7

$$\begin{aligned}
\sum_{k^*+\ell^*<k\le k^*+\ell^*+L} \mathbf{Prob}(k_x \ge k) \cdot d^k &= d^{k^*+\ell^*} \cdot \sum_{1\le j\le L} \mathbf{Prob}(k_x \ge k^* + \ell^* + j) \cdot d^j \\
&\le \; d^{k^*+\ell^*} \cdot \sum_{1\le j\le L} \mathbf{Prob}(h_1(x), h_2(x) \in [m] - Y_{k^*+\ell^*+(j-1)}) \cdot d^j \\
&\le \; d^{k^*+\ell^*} \cdot \sum_{1\le j\le L} (d^{-2(j-1)/3})^2 \cdot d^j < 2d^{k^*+\ell^*+1}. \qquad (10)
\end{aligned}$$

The sum of the parts in (9) and (10) is bounded by $(k^* + \ell^* + 2)d^{k^*+\ell^*+1} = O(d)^{O(\log(1/\varepsilon))} = (1/\varepsilon)^{O(\log d)}$. This shows that the expected number of edges probed in inserting x is bounded by $(1/\varepsilon)^{O(\log d)}$. $\square$

To prove Theorem 2 we note that with probability $1 - O(m^{-d/2})$ the graph G satisfies the conclusions of Lemmas 2, 4, and 6. If this is the case, then the expansion properties of Lemmas 3, 5, and 7 hold, and we may apply Lemma 8 to obtain the claimed bound on the expected insertion time. If G does not have the expansion properties from Lemmas 3, 5, and 7, and Y_0 is reachable from $\{h_1(x), h_2(x)\}$, the BFS will find an augmenting path in time $O(n)$ — this gives a contribution of $O(m^{1-d/2})$ to the expected insertion time. In case Y_0 is not reachable from $\{h_1(x), h_2(x)\}$, the functions h_1, h_2 are not suitable for $S \cup \{x\}$, and we must perform a total rehashing for all these keys. By Theorem 1 this happens with probability $O(m^{1-d})$. It is easily seen that even if we simply insert the keys by the BFS procedure, and rehash again if necessary, the expected time for rebuilding the table is $O(n)$. Hence, this last case contributes $O(m^{2-d})$ to the expected insertion cost.

Conclusion. We obtained new results for a natural data allocation problem arising in different contexts: balanced allocation with two choices, edge orientation in random graphs, dynamic dictionaries with worst case constant access time. The constant given in Theorem 2 is certainly not optimal, and should be improved. It is an intriguing open problem to analyze at least one variant of the random-walk insertion procedure, if possible establishing a bound of $O(\ln(1/\varepsilon))$ on the expected number of blocks probed (while maintaining the bound $d = O(\log(1/\varepsilon))$).

References

[1] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29:180–200, 2000.

[2] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking. Balanced allocations: The heavily loaded case. In *32nd STOC*, pp. 745–754. ACM, 2000.

[3] A. Czumaj, F. Meyer auf der Heide, and V. Stemann. Contention resolution in hashing based shared memory simulations. *SIAM J. Comput.*, 29:1703–1739, 2000.

[4] A. Czumaj, Ch. Riley, and Ch. Scheideler. Perfectly balanced allocation. In *RANDOM-APPROX*, LNCS 2764, pp. 240–251. Springer, 2003.

[5] R. Diestel. *Graph Theory.* Springer, New York, 1997.

[6] M. Dietzfelbinger and F. Meyer auf der Heide. Dynamic hashing in real time. In Buchmann, J., et al., editor, *Informatik · Festschrift zum 60. Geburtstag von Günter Hotz*, pp. 95–119. B. G. Teubner, 1992.

[7] M. Dietzfelbinger and P. Woelfel. Almost random graphs with simple hash functions. In *35th STOC*, pp. 629–638. ACM, 2003.

[8] D. Fotakis, R. Pagh, P. Sanders, and P. Spirakis. Space efficient hash tables with worst case constant access time. *Theory of Computing Systems*, 38:229–248, 2005.

[9] A. Frieze. Personal communication in [1]. 1990.

[10] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *40th FOCS*, pp. 285–298. IEEE, 1999.

[11] T. Hagerup and Ch. Rüb. A guided tour of Chernoff bounds. *Inf. Process. Lett.*, 33:305–308, 1990.

[12] R. Karp. Random graphs, random walks, differential equations and the probabilistic analysis of algorithms. In *15th STACS*, LNCS 1373, pp. 1–2. Springer, 1998.

[13] R. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulations on a distributed memory machine. *Algorithmica*, 16:517–542, 1996.

[14] M. Mitzenmacher, A. W. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results, vol. 1, pp. 255–312. In Rajasekaran et al., editor, *Handbook of Randomized Computing.* Kluwer Academic Press, 2001.

[15] R. Motwani. Average-case analysis of algorithms for matchings and related problems. *J. of the ACM*, 41(6):1329–1356, 1994.

[16] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[17] A. Östlin and R. Pagh. Uniform hashing in constant time and linear space. In *35th STOC*, pp. 622–628. ACM, 2003.

[18] R. Pagh and F. F. Rodler. Cuckoo hashing. *J. Algorithms*, 51:122–144, 2004.

[19] R. Panigrahy. Efficient hashing with lookups in two memory accesses. In *16th SODA*. ACM-SIAM, 2005.

[20] P. Sanders. Fast priority queues for cached memory. In *1st Workshop ALENEX*, LNCS 1619, pp. 312–327. Springer, 1999.

[21] P. Sanders. Reconciling simplicity and realism in parallel disk models. In *12th SODA*, pp. 67–76. ACM-SIAM, 2001.

[22] P. Sanders, S. Egner, and J. Korst. Fast concurrent access to parallel disks. In *11th SODA*, pp. 849–858. ACM-SIAM, 2000.

[23] A. Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In *30th FOCS*, pp. 20–25. IEEE, 1989.

A Some Inequalities

We will be using the following upper bound for binomial coefficients:

$$\binom{n}{k} \le \frac{n^n}{k^k(n-k)^{n-k}} = \left(\frac{1}{\mu^\mu(1-\mu)^{1-\mu}}\right)^n \text{, for } 0 \le k \le n, \tag{11}$$

where $\mu = k/n$. If k is small in comparison to n, the following weaker inequality is already helpful:

$$\binom{n}{k} \le \left(\frac{e \cdot n}{k}\right)^k. \tag{12}$$

Further, a standard version of the Chernoff-Hoeffding bounds is used repeatedly: If $X_1, \ldots, X_n$ are independent 0-1-valued random variables and $X = X_1 + \cdots + X_n$, then for $\mathbf{E}(X) \le a \le n$ we have

$$\mathbf{Prob}(X \ge a) \le \left(\frac{\mathbf{E}(X)}{a}\right)^a \left(\frac{n - \mathbf{E}(X)}{n-a}\right)^{n-a}. \tag{13}$$

(For a proof, see e.g. [11].)

B Proof of Lemma 1

Proof. For $1 \le j \le m/(1+\varepsilon)$, let $F(j)$ be the probability of the event that there is a set Y of j blocks such that there is a set $X \subseteq S$ of dj keys which satisfies $\Gamma(X) \subseteq Y$. Clearly,

$$F \le \sum_{1 \le j \le m/(1+\varepsilon)} F(j). \tag{14}$$

To bound $F(j)$, we apply the Chernoff-Hoeffding bound (13) as follows. For a fixed $Y \subseteq [m]$ of size j define random variables I_x, $x \in S$, as follows:

$$I_x := \begin{cases} 1, & \text{if } h_1(x), h_2(x) \in Y, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Further let $I = \sum_{x \in S} I_x$. We have $\mathbf{Prob}(I_x = 1) = (j/m)^2$ and $\mathbf{E}(I) = \sum_{x \in S} \mathbf{E}(I_x) = n\frac{j^2}{m^2}$. Because the I_x are independent, by (13) we obtain the following bound:

$$\mathbf{Prob}(I \ge jd) \le \left(\frac{nj^2}{m^2 jd}\right)^{jd} \left(\frac{n(m^2 - j^2)}{m^2(n - jd)}\right)^{n-jd}. \tag{16}$$

Because there are $\binom{m}{j}$ sets Y of size j, we get

$$F(j) \le \binom{m}{j} \left(\frac{nj^2}{m^2 jd}\right)^{jd} \left(\frac{n(m^2 - j^2)}{m^2(n - jd)}\right)^{n-jd}. \tag{17}$$

We apply (11) and substitute $n = \frac{dm}{1+\varepsilon}$ to obtain

$$\begin{aligned} F(j) &\le \frac{m^m}{j^j(m-j)^{m-j}} \left(\frac{j}{(1+\varepsilon)m}\right)^{jd} \left(\frac{d(m^2 - j^2)}{(1+\varepsilon)m(n - jd)}\right)^{dm/(1+\varepsilon) - jd} \\ &= \frac{m^m}{j^j(m-j)^{m-j}} \left(\frac{j}{(1+\varepsilon)m}\right)^{jd} \left(\frac{m^2 - j^2}{(1+\varepsilon)m(m/(1+\varepsilon) - j)}\right)^{dm/(1+\varepsilon) - jd} \\ &= (1+\varepsilon)^{-jd} m^{\frac{m(1+\varepsilon-d)}{1+\varepsilon}} j^{j(d-1)} (m-j)^{j-m} \left(\frac{m^2 - j^2}{m - j(1+\varepsilon)}\right)^{d\frac{m - j(1+\varepsilon)}{1+\varepsilon}}. \end{aligned} \tag{18}$$

We abbreviate the expression of the right-hand side of (18) by $f(j,\varepsilon)$ and estimate this bound in different ranges of j, $1 \le j \le m/(1+\varepsilon)$.

Case 1: $j = 1$. — By (17) we get:

$$\begin{aligned} F(1) &\le m\left(\frac{1}{(1+\varepsilon)m}\right)^{d}\left(\frac{n-n/m^2}{n-d}\right)^{n-d} < m\left(\frac{1}{(1+\varepsilon)m}\right)^{d} e^{d-n/m^2} \\ &= O\left(\frac{1}{m^{d-1}}\right). \end{aligned} \tag{19}$$

Case 2: $2 \le j < e^{-4}m$. — We first note that $f(j,\varepsilon)$ is decreasing in ε. For this, we differentiate:

$$\frac{\partial \ln f}{\partial \varepsilon} = -d(1+\varepsilon)^{-2} m \ln\left(\frac{1-(j/m)^2}{1-j(1+\varepsilon)/m}\right). \tag{20}$$

Since $j/m < 1$, we have $1-(j/m)^2 > 1-(1+\varepsilon)j/m$, hence $\frac{\partial \ln f}{\partial \varepsilon} < 0$. This means that $f(j,\varepsilon) < f(j,0)$, and hence we continue from (18) as follows:

$$\begin{aligned} F(j) &< m^{m(1-d)} j^{j(d-1)} (m-j)^{j-m} (m+j)^{d(m-j)} \\ &= m^{m(1-d)} j^{j(d-1)} m^{j-m} \left(1-\frac{j}{m}\right)^{j-m} m^{d(m-j)} \left(1+\frac{j}{m}\right)^{d(m-j)} \\ &< m^{j(1-d)} j^{j(d-1)} e^{\frac{j(m-j)}{m}} e^{\frac{dj(m-j)}{m}} < \left(\left(\frac{j}{m}\right)^{d-1} e^{d+1}\right)^{j}. \end{aligned} \tag{21}$$

The terms $\left(\left(\frac{j}{m}\right)^{d-1} e^{d+1}\right)^{j}$ are geometrically decreasing for $2 \le j < e^{-4}m$, hence

$$\sum_{2 \le j < e^{-4}m} F(j) = O\left(\left(\left(\frac{2}{m}\right)^{d-1} e^{d+1}\right)^{2}\right) = O\left(\frac{1}{m^{2d-2}}\right). \tag{22}$$

Case 3: $e^{-4}m \le j \le (1-2\varepsilon)m$. — If we substitute $\alpha = j/m$ in the right-hand side of (18) and take logarithms, we get $\ln(F(j)) \le mR_\varepsilon(\alpha)$, where

$$\begin{aligned} R_\varepsilon(\alpha) &:= -\alpha\ln\alpha - (1-\alpha)\ln(1-\alpha) + \alpha d(\ln\alpha - \ln(1+\varepsilon)) \\ &\quad + \frac{d(1-\alpha(1+\varepsilon))(\ln(1-\alpha^2) - \ln(1-\alpha(1+\varepsilon)))}{1+\varepsilon}. \end{aligned} \tag{23}$$

If d is chosen so that $R_\varepsilon(\alpha) < 0$ for $e^{-4} \le \alpha \le 1-2\varepsilon$, we will have $F(j) \le c^m$ for a constant $c < 1$. To find a suitable d we shuffle the expression for $R_\varepsilon(\alpha)$ a little and get the following sufficient condition for $R_\varepsilon(\alpha) < 0$:

$$d > \frac{\alpha\ln\alpha + (1-\alpha)\ln(1-\alpha)}{\alpha(\ln\alpha - \ln(1+\varepsilon)) + \frac{(1-\alpha(1+\varepsilon))(\ln(1-\alpha^2)-\ln(1-\alpha(1+\varepsilon)))}{1+\varepsilon}} =: r_\varepsilon(\alpha) \tag{24}$$

Our aim is to find an upper bound for $r_\varepsilon(\alpha)$, in the range $0 \le \varepsilon \le 0.25$, $e^{-4} \le \alpha \le 1-2\varepsilon$. First we show that

$$g(\alpha) := \frac{\alpha\ln\alpha + (1-\alpha)\ln(1-\alpha)}{\alpha\ln\alpha + (1-\alpha)\ln(1+\alpha)} \tag{25}$$

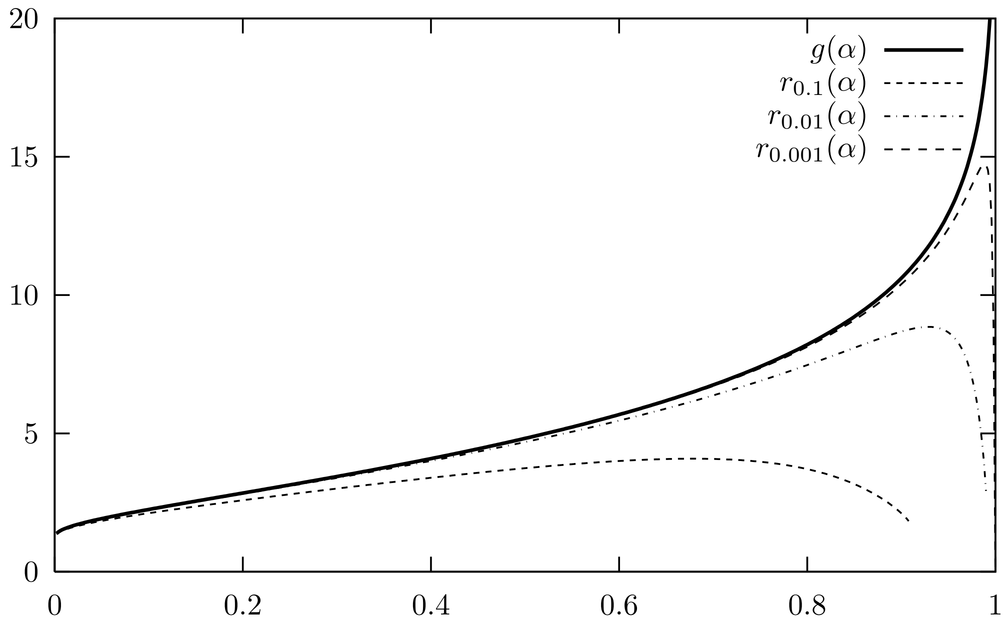

Figure 1: The graphs of g and r_ε for several ε

is an upper bound for r_ε, for all $\varepsilon \le 0.25$ (Fig. 1).

The expressions r_ε and $g(\alpha)$ are fractions of the form A/B and A/B', where $A, B, B' < 0$. This implies that $A/B' \ge A/B$ if and only if $B' \ge B$. That means that we must prove the following inequality:

$$\begin{aligned} 0 \le\ & \alpha \ln \alpha + (1-\alpha)\ln(1+\alpha) - \\ & \left(\alpha(\ln\alpha - \ln(1+\varepsilon)) + \frac{(1-\alpha(1+\varepsilon))(\ln(1-\alpha^2) - \ln(1-\alpha(1+\varepsilon)))}{1+\varepsilon}\right) \\ & = (1-\alpha)\ln(1+\alpha) + \alpha\ln(1+\varepsilon) - \left(\frac{1}{1+\varepsilon} - \alpha\right)\ln\left(\frac{1-\alpha^2}{1-\alpha(1+\varepsilon)}\right). \end{aligned} \tag{26}$$

This is easy: The derivative of the last expression in (26) with respect to ε is

$$\frac{\ln(1-\alpha^2) - \ln(1-\alpha(1+\varepsilon))}{(1+\varepsilon)^2} = \frac{\ln\left(\frac{1-\alpha^2}{1-\alpha(1+\varepsilon)}\right)}{(1+\varepsilon)^2} > 0. \tag{27}$$

Therefore

$$\begin{aligned} & (1-\alpha)\ln(1+\alpha) + \alpha\ln(1+\varepsilon) - \left(\frac{1}{1+\varepsilon} - \alpha\right)\ln\left(\frac{1-\alpha^2}{1-\alpha(1+\varepsilon)}\right) \\ & \qquad \ge (1-\alpha)\ln(1+\alpha) - (1-\alpha)\left(\ln(1-\alpha^2) - \ln(1-\alpha)\right) = 0, \end{aligned} \tag{28}$$

which shows that (26) is true.

Next we need to analyze the function $g(\alpha)$ in the interval $[e^{-4}, 1-2\varepsilon]$. Now g is a fixed function (see Fig. 1) and we abbreviate a somewhat tedious exercise in calculus looking at

$$g'(\alpha) = \frac{\ln(\alpha)((1+\alpha)\ln(1+\alpha) - 2\alpha) + \ln(1-\alpha)(2\alpha - (1+\alpha)\ln(\alpha) - 2)}{(\alpha\ln(\alpha) + \ln(1+\alpha) - \ln(1+\alpha)\alpha)^2(1+\alpha)}. \tag{29}$$

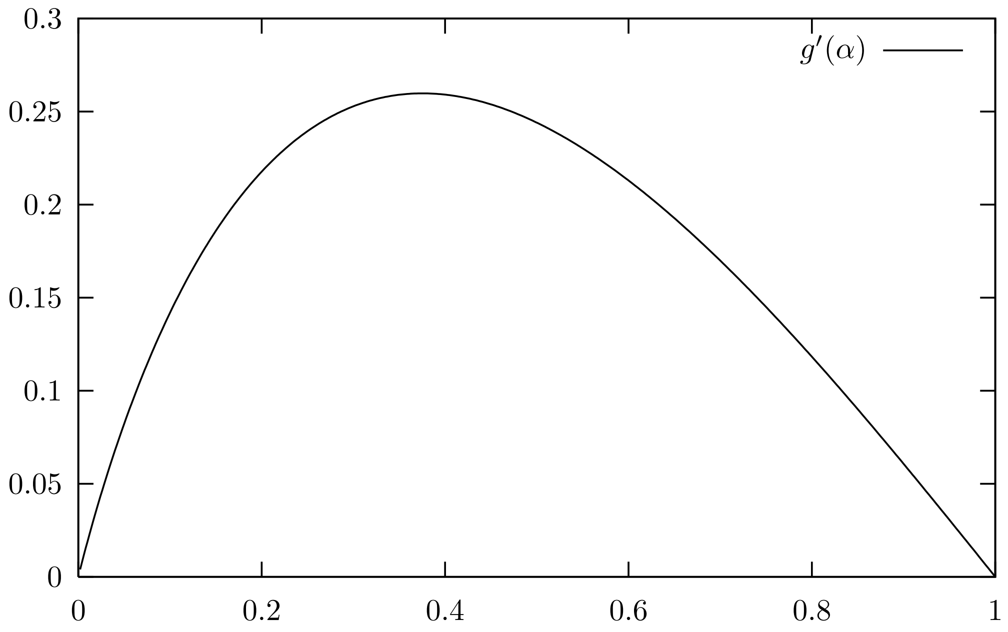

Figure 2: The graph of the numerator of g'

Both numerator (see Fig. 2) and denominator of the latter function turn out to be positive for $0 < \alpha < 1$. This means that $g(\alpha)$ is strictly increasing and hence

$$g(\alpha) \le g(1-2\varepsilon) = 1 + \frac{\ln\varepsilon - \ln(1-\varepsilon)}{\frac{(1-2\varepsilon)\ln(1-2\varepsilon)}{2\varepsilon} + \ln 2 + \ln(1-\varepsilon)}. \tag{30}$$

Our next aim is to bound $g(1-2\varepsilon)$ in (30).

We want to show that $g(1-2\varepsilon) \le 1 + \frac{-\ln\varepsilon}{1-\ln 2}$. This is true if and only if

$$\left(\frac{(1-2\varepsilon)\ln(1-2\varepsilon)}{2\varepsilon} + 1 + \ln(1-\varepsilon)\right)\ln\varepsilon - (1-\ln 2)\ln(1-\varepsilon) \ge 0. \tag{31}$$

To prove (31) symbolically, we study the Taylor series of $\frac{(1-2\varepsilon)\ln(1-2\varepsilon)}{2\varepsilon} + 1 + \ln(1-\varepsilon)$. However, in this extended abstract we omit this tedious calculation and refer to a plot (see Fig. 3), which makes it clear that $1 + \frac{-\ln\varepsilon}{1-\ln 2} - g(1-2\varepsilon) > 0$ for $0 < \varepsilon < 0.5$. We summarize: If we choose

$$d \ge 1 + \frac{1}{1-\ln 2}\ln\left(\frac{1}{\varepsilon}\right) = 1 + 3.25889\ldots\ln\left(\frac{1}{\varepsilon}\right), \tag{32}$$

then $F(j) = O(c^n)$ for a constant $c < 1$, for $e^{-4} \le j \le (1-2\varepsilon)m$.

Case 4: $(1-2\varepsilon)m \le j \le m/(1+\varepsilon)$. — Let $0 \le \varepsilon \le 0.25$ be fixed. Looking at the derivative of $r_\varepsilon = r_\varepsilon(\alpha)$ at $\alpha = 1-2\varepsilon$, we get

$$r'_\varepsilon(1-2\varepsilon) = \frac{((1-2\varepsilon)\ln(1-2\varepsilon) + 2\varepsilon\ln\varepsilon)(2\ln 2 + \ln(1-\varepsilon) - \ln(1+2\varepsilon))}{\left((1-\varepsilon-\varepsilon^2)\ln\left(\frac{1+\varepsilon}{1-2\varepsilon}\right) - (2\varepsilon^2+\varepsilon)\ln\left(\frac{4(1-\varepsilon)}{1+2\varepsilon}\right)\right)^2}. \tag{33}$$

We show that the numerator of the fraction in (33) is negative, hence that $r'_\varepsilon(1-2\varepsilon) < 0$.

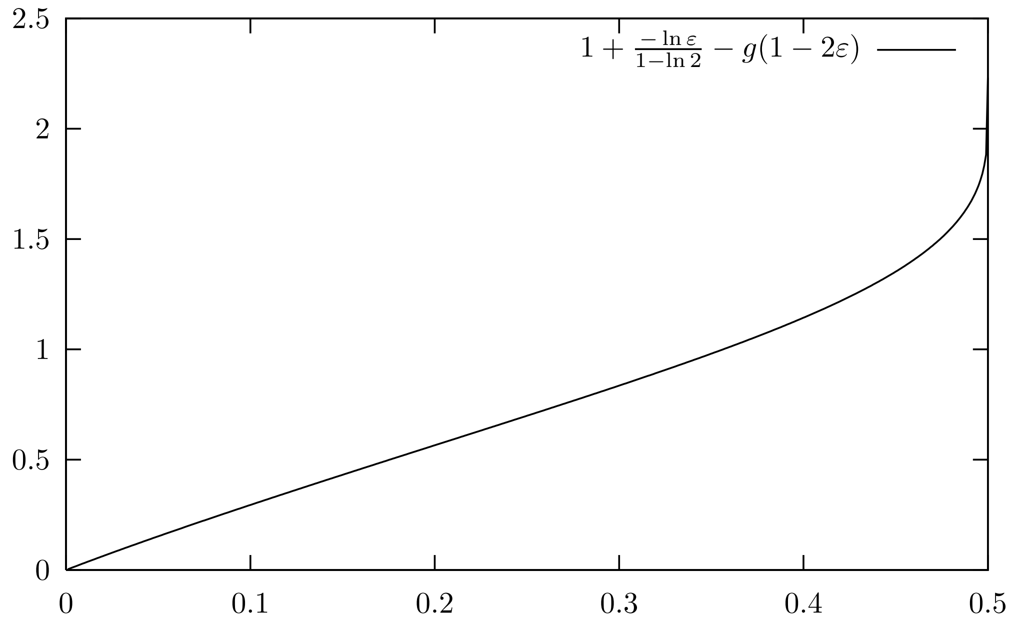

Figure 3: The graph of $1+\frac{-\ln\varepsilon}{1-\ln 2}-g(1-2\varepsilon)$

To prove that the numerator is negative, we examine its two factors. Obviously, $(1-2\varepsilon)\ln(1-2\varepsilon)+2\varepsilon\ln\varepsilon<0$. By $\ln(1-x)\geq -x/(1-x)$, $0\leq x<1$, we get for $\varepsilon\leq 0.25$:

$$2\ln 2+\ln(1-\varepsilon)-\ln(1+2\varepsilon)=2\ln 2+\ln\left(1-\frac{3\varepsilon}{1+2\varepsilon}\right)$$
$$>2\ln 2-\frac{3\varepsilon}{1-\varepsilon}>0. \quad (34)$$

Let us denote the denominator of $r_\varepsilon(\alpha)$ from (24) by u. We differentiate twice to obtain

$$u''=\frac{2\alpha(1+\alpha^2)(2\varepsilon+\varepsilon^2+2)-(1+\varepsilon)(6\alpha^2+1+\alpha^4)}{\alpha(\alpha^2-1)^2(\varepsilon\alpha+\alpha-1)(1+\varepsilon)}. \quad (35)$$

The numerator of u'' is a polynomial of degree four. We discuss the position of its roots. Since $\varepsilon<0.25$, there is only one root in $(0,1/(1+\varepsilon)]$, which is $1+\varepsilon-\sqrt{\varepsilon^2+2\varepsilon}$. Further we have

$$1+\varepsilon-\sqrt{\varepsilon^2+2\varepsilon}<1-2\varepsilon, \quad (36)$$

therefore u'' does not have a root in $[1-2\varepsilon,1/(1+\varepsilon)]$. At $\alpha=1-2\varepsilon$ we find

$$u''(1-2\varepsilon)=-\frac{(1-4\varepsilon)(2\varepsilon^2+1)}{4\varepsilon(1-\varepsilon)^2(1+\varepsilon)(1-4\varepsilon^2)}<0. \quad (37)$$

Hence u'' is negative if $1-2\varepsilon\leq\alpha\leq 1/(1+\varepsilon)$.

The second derivative $\frac{1}{\alpha(1-\alpha)}$ of the numerator v of r_ε is positive. Both u and v are negative. This implies

$$(v'u-vu')'=v''u-vu''<0, \quad (38)$$

and hence $v'u-vu'$ is decreasing. Together with the fact that $v'u-vu'$ is negative in $\alpha=1-2\varepsilon$, this shows that $v'u-vu'<0$ in $[1-2\varepsilon,1/(1+\varepsilon)]$. Thus

$$r'_\varepsilon=\left(\frac{v}{u}\right)'=\frac{v'u-vu'}{u^2}<0, \quad (39)$$

and hence r_ε is decreasing. Hence $r_\varepsilon(\alpha) \le r_\varepsilon(1-2\varepsilon)$ in this range, and we get $F(j) = O(c^n)$ as in Case 3. $\square$

C Proof of Lemma 2

A set of blocks Y, $|Y| = r$, is hit by $\frac{4}{3}rd$ keys with h_1 or h_2 if and only if at most $n - \frac{4}{3}rd$ keys hit $\overline{Y} = [m] - Y$ with both hash functions h_1 and h_2. By $F(r)$ we denote the probability that there exists a set Y of size r such that more than $n - \frac{4}{3}rd$ keys hit $\overline{Y}$ with both hash functions. To bound $F(r)$ we argue as in the proof of Lemma 1, using the Chernoff-Hoeffding bound (11). This yields

$$F(r) \le \binom{m}{r} \left(\frac{n\left(1-\frac{r}{m}\right)^2}{n - \frac{4}{3}rd} \right)^{n-\frac{4}{3}rd} \left(\frac{n - n\left(1-\frac{r}{m}\right)^2}{\frac{4}{3}rd} \right)^{\frac{4}{3}rd}. \tag{40}$$

We substitute $n = dm/(1+\varepsilon)$ and $r = \alpha m$, use (11), and simplify to get

$$F(r) \le \left[\frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(\frac{(1-\alpha)^2}{1-(1+\varepsilon)\frac{4}{3}\alpha} \right)^{\frac{d}{1+\varepsilon}-\frac{4}{3}\alpha d} \left(\frac{2-\alpha}{\frac{4}{3}(1+\varepsilon)} \right)^{\frac{4}{3}\alpha d} \right]^m. \tag{41}$$

To prove the lemma it is sufficient to show

$$\frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(\frac{(1-\alpha)^2}{1-(1+\varepsilon)\frac{4}{3}\alpha} \right)^{\frac{d}{1+\varepsilon}-\frac{4}{3}\alpha d} \left(\frac{2-\alpha}{\frac{4}{3}(1+\varepsilon)} \right)^{\frac{4}{3}\alpha d} \tag{42}$$

is smaller than 1 for $\varepsilon/(1+\varepsilon) \le \alpha \le 5/13$. This is the case if and only if d is at least

$$\frac{\alpha \ln \alpha + (1-\alpha)\ln(1-\alpha)}{\left(\frac{1}{1+\varepsilon} - \frac{4}{3}\alpha\right)\left(2\ln(1-\alpha) - \ln\left(1 - \frac{4(1+\varepsilon)\alpha}{3}\right)\right) + \frac{4\alpha}{3}\left(\ln(2-\alpha) - \ln\left(\frac{4(1+\varepsilon)}{3}\right)\right)}. \tag{43}$$

We wish to find an upper bound for the expression in (43). We call its numerator u and its denominator v and show that v'' if positive for $0 < \alpha \le \frac{5}{13}$, if $\varepsilon \le 0.1$. Then we can replace v by a secant $\tilde{v}$ through the points of v at $\alpha = 0$ and $\alpha = \frac{5}{13}$, and get $u/\tilde{v} > u/v$ (Fig. 4).

The second derivative of v is

$$v'' = \frac{12\alpha - 16\varepsilon - 8 + 84\alpha\varepsilon + 10\alpha^2 - 40\alpha^2\varepsilon - 32\alpha^2\varepsilon^2 + 64\varepsilon^2 - 12\alpha^3\varepsilon - 12\alpha^3}{3(2-\alpha)^2(1+\varepsilon)(-3+4\alpha+4\alpha\varepsilon)(1-\alpha)^2}. \tag{44}$$

The denominator of the latter term is negative if $\alpha < \frac{5}{13}$. Therefore it is sufficient to consider the roots of the numerator of v''. Although the numerator of v'' is a polynomial of degree 3 we do not want to bother with its exact roots. Instead we investigate its derivative, which is

$$12 + 84\varepsilon + 20\alpha - 80\alpha\varepsilon - 64\varepsilon^2\alpha - 36\alpha^2\varepsilon - 36\alpha^2. \tag{45}$$

The roots of the latter term are

$$\alpha_{1,2} = \frac{5 - 20\varepsilon - 16\varepsilon^2 \pm \sqrt{133 + 664\varepsilon + 996\varepsilon^2 + 640\varepsilon^3 + 256\varepsilon^4}}{18(1+\varepsilon)}. \tag{46}$$

It is easy to see that $\alpha_1, \alpha_2 \notin [0, \frac{5}{13}]$ if $0 < \varepsilon \le 0.1$. With $\alpha = \frac{5}{13}$ the expression (45) reduces to

$$\frac{2428}{169} + \frac{8096}{169}\varepsilon - \frac{320}{13}\varepsilon^2, \tag{47}$$

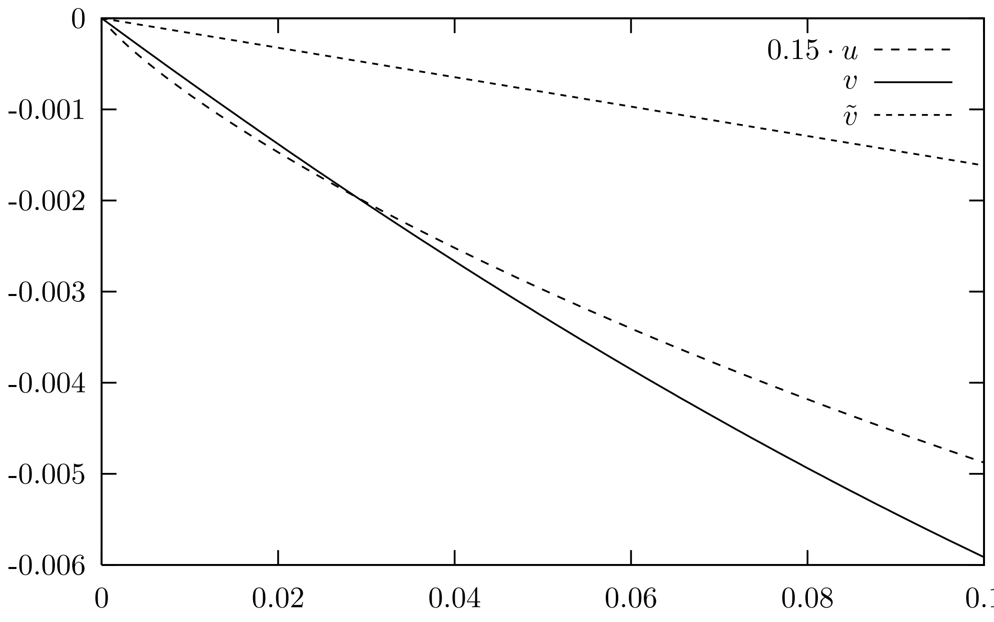

Figure 4: Numerator u, denominator v and secant $\tilde{v}$ of v in the function from (43), for $\varepsilon = 0.01$

which is positive if $\varepsilon \leq 0.1$. Therefore the numerator of v'' is increasing. Further, at $\alpha = \frac{5}{13}$ the numerator of v'' is

$$-\frac{5686}{2197} + \frac{21328}{2197}\varepsilon + \frac{10016}{169}\varepsilon^2, \tag{48}$$

which is negative, if $\varepsilon \leq 0.1$; because it is increasing it is negative for each $0 \leq \alpha \leq \frac{5}{13}$. Because the denominator of v'' is negative, v'' itself is positive.

The secant $\tilde{v}$ that intersects v at $\alpha = 0$ and $\alpha = \frac{5}{13}$, is

$$\tilde{v}(\alpha) = \frac{13}{5} v\left(\frac{5}{13}\right)\alpha. \tag{49}$$

If we replace v by $\tilde{v}$ in (43), we get

$$\frac{5(\alpha\ln(\alpha) + (1-\alpha)\ln(1-\alpha))}{13\alpha\left(\left(\frac{1}{1+\varepsilon} - \frac{20}{39}\right)\ln\left(\frac{192}{13(19-20\varepsilon)}\right) + \frac{20}{39}\ln\left(\frac{63}{52(1+\varepsilon)}\right)\right)}. \tag{50}$$

The expression

$$\left(\left(\frac{1}{1+\varepsilon} - \frac{20}{39}\right)\ln\left(\frac{192}{13(19-20\varepsilon)}\right) + \frac{20}{39}\ln\left(\frac{63}{52(1+\varepsilon)}\right)\right) \tag{51}$$

is negative if $\varepsilon \leq 0.1$. The expression

$$\frac{\alpha\ln(\alpha) + (1-\alpha)\ln(1-\alpha)}{\alpha} \tag{52}$$

has a positive derivative. So (50) is decreasing and is maximal at the left edge. Because $\alpha \geq \frac{\varepsilon}{1+\varepsilon}$, the upper bound for (50) is

$$\frac{5(\varepsilon\ln\varepsilon - (1+\varepsilon)\ln(1+\varepsilon))}{13\varepsilon\left(\left(\frac{1}{1+\varepsilon} - \frac{20}{39}\right)\ln\left(\frac{192}{13(19-20\varepsilon)}\right) + \frac{20}{39}\ln\left(\frac{63}{52(1+\varepsilon)}\right)\right)} =: g_\varepsilon \tag{53}$$

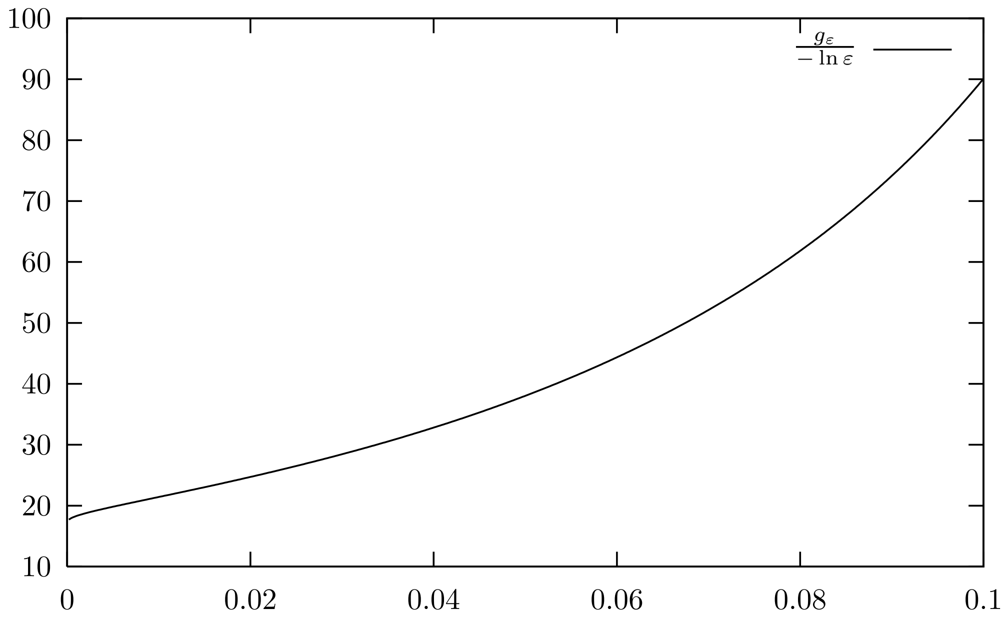

Figure 5: The function $\frac{g_\varepsilon}{-\ln\varepsilon}$

By choosing $d > g_\varepsilon$, we get $F(r) = O(\beta^m)$ for a constant $\beta < 1$.

Now let us take a closer look at g_ε. First we see that

$$\lim_{\varepsilon\to 0}\frac{g_\varepsilon}{-\ln\varepsilon} = -\frac{15}{74\ln 2 - 39\ln 13 - 19\ln 19 + 59\ln 3 + 20\ln 7} = 15.82\ldots \tag{54}$$

We want to avoid a comprehensive analysis of $-g_\varepsilon/\ln\varepsilon$. Instead we refer to Fig. 5. Obviously, $-g_\varepsilon/\ln\varepsilon$ is a continous, strictly increasing function. For $\varepsilon = 0.1$ we get a value of $90.08177\ldots$. This finishes the proof of the lemma.

Comment: For tighter bounds on ε, the constant 90.1 in Lemma 2 may be replaced by some smaller constant. For example, for $0 \le \varepsilon \le 0.01$, we have $g_\varepsilon < 25\ln(1/\varepsilon)$.

D Proof of Lemma 4

Proof. The set Y is hit by $n - \frac{9d}{10}(m - |Y|)$ keys with one hash function if and only if at most $\frac{9d}{10}(m - |Y|) = \frac{9d}{10}|\overline{Y}|$ keys hit the set $\overline{Y}$ with *both* hash functions.

As before, we bound the probability $F_\varepsilon(r)$ that there is a set $\overline{Y}$ of size r, $\frac{4}{e^4 d^3} \le r \le \frac{m}{2}$, which is hit by more than $\frac{9d}{10}|\overline{Y}|$ keys with both hash functions by applying the Chernoff-Hoeffding-bound (13) and the binomial bound (11):

$$F_\varepsilon(r) \le \frac{m^m}{r^r(m-r)^{m-r}}\left(\frac{10r}{9(1+\varepsilon)m}\right)^{\frac{9}{10}rd}\left(\frac{10(m^2-r^2)}{(10m-9r(1+\varepsilon))m}\right)^{\frac{dm}{1+\varepsilon}-\frac{9}{10}rd} \tag{55}$$

It is easy to see the right hand side of (55) is decreasing in ε. Indeed, the derivative of $\ln(F_\varepsilon(r))$ with respect to ε is

$$-\frac{dm}{(1+\varepsilon)^2}\ln\left(\frac{10(m^2-r^2)}{10m^2-9(1+\varepsilon)rm}\right), \tag{56}$$

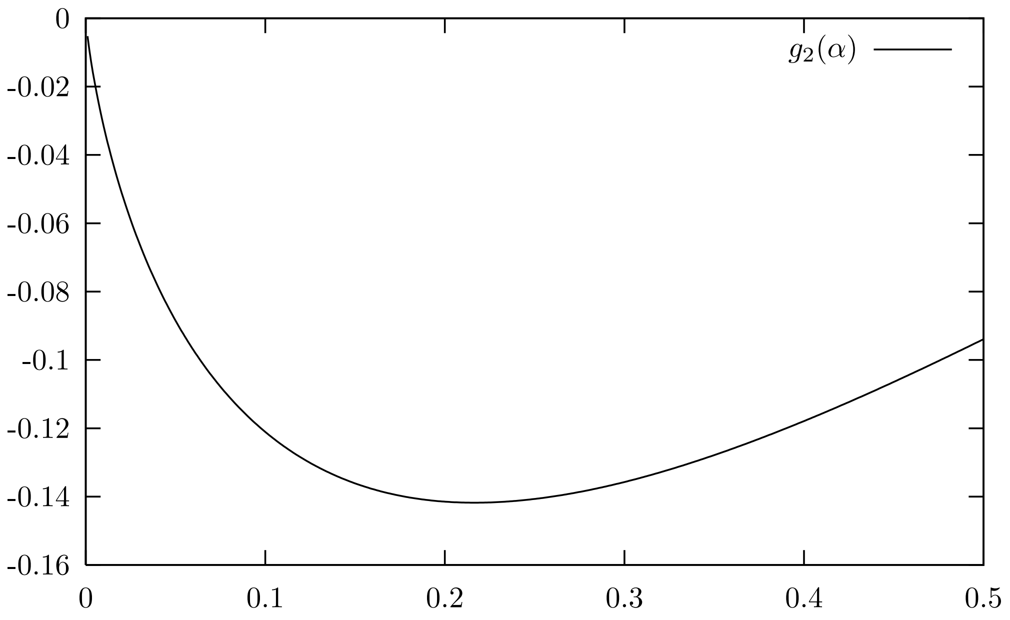

Figure 6: The graph of g_2

which is negative for $r \leq m/2$, because $\frac{r}{m} \leq \frac{1}{2} < \frac{9(1+\varepsilon)}{10}$ and hence

$$10r^2 < 9(1+\varepsilon)rm \quad \Rightarrow \quad 10(m^2 - r^2) > 10m^2 - 9(1+\varepsilon)rm. \tag{57}$$

This leads us to

$$F_\varepsilon(r) < \frac{m^m}{r^r(m-r)^{m-r}} \left(\frac{10r}{9m}\right)^{\frac{9}{10}rd} \left(\frac{10(m^2-r^2)}{(10m-9r)m}\right)^{dm-\frac{9}{10}rd} =: g_1(r). \tag{58}$$

Next we establish that $g_1(r)$ decreases if d increases: We substitute $r = \alpha m$ and examine

$$\frac{1}{m} \cdot \frac{\partial \ln g_1(r)}{\partial d} = \ln 10 + \frac{9}{10}\alpha \ln\left(\frac{\alpha}{9}\right) + \left(1 - \frac{9}{10}\alpha\right) \ln\left(\frac{1-\alpha^2}{10-9\alpha}\right) =: g_2(\alpha) \tag{59}$$

Again, to spare the reader a tedious discussion of this function, we read off from a plot (see Fig. 6) that this function is negative for $0 < \alpha < 1/2$, and hence that $g_1(r)$ decreases if d increases.

Now let us look at $g_1(r)$ for $d = 8$. With $r = \alpha m$ we get the following representation: $g_1(r) = \tilde{g}(\alpha)^m$, where

$$\tilde{g}(\alpha) = \frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}} \cdot \left(\frac{10\alpha}{9}\right)^{7.2\alpha} \cdot \left(\frac{10(1-\alpha^2)}{10-9\alpha}\right)^{8-7.2\alpha} \tag{60}$$

Again, we shortcut a tedious proof by calculus by looking at a plot of $\tilde{g}(\alpha)$ (see Fig. 7), which reveals that this function attains the value 1 for $\alpha = 0$ and has exactly one minimum point at about 0.14. This means that $\tilde{g}(\alpha)$ attains its maximum in $[\frac{4}{e^4 d^3}, \frac{1}{2}]$ at one of the border points, and hence $\tilde{g}(\alpha) \leq \max\{g(\frac{4}{e^4}d^3), g(\frac{1}{2})\} < 1$, for all $d \geq 8$, $\frac{4}{e^4 d^3} \leq \alpha \leq \frac{1}{2}$. This implies that $F_\varepsilon(r) \leq \tilde{g}(\frac{4}{e^4 d^3})^m$ for all r in $[\frac{4m}{e^4 8^3}, m/2]$, and hence

$$\sum_{\frac{4m}{e^4 d^3} \leq r \leq m/2} F_\varepsilon(r) < m \cdot \beta^m,$$

for some constant $\beta < 1$.

□

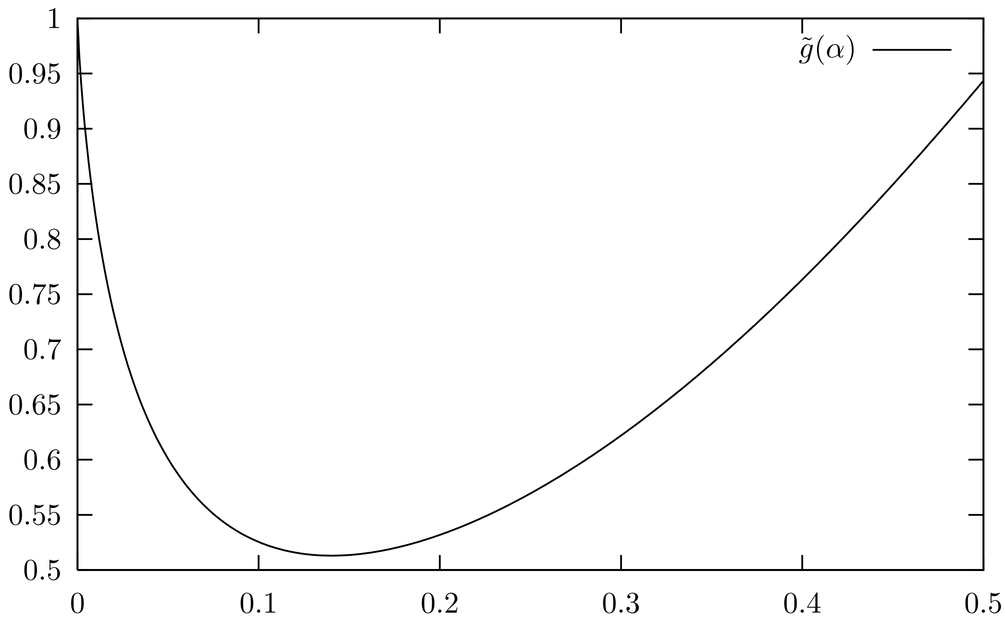

Figure 7: The graph of $\tilde{g}$

E Experiments

In two experiments we compared the performance of four methods for implementing a dynamic dictionary:

- blocked cuckoo hashing (*cuckoo-block*), as described in the paper;
- a cuckoo-linear-probing scheme (*cuckoo-lp*), a variant in which key x is assigned to two positions $h_1(x)$ and $h_2(x)$ in a table of size $m = (1+\varepsilon)n$ and may be placed in one of the cells $(h_j(x) + r) \bmod m$, for $j = 1, 2$ and $0 \le r < d$;
- linear probing (*lp*) with one hash function;
- cuckoo hashing with d functions (*cuckoo-d*).

In the first experiment, $n = 10^6$ distinct keys were inserted into an empty table of size $(1+\varepsilon)n$. Then 10^6 keys, all different from the first ones, were searched. Thus the time for building a dictionary and the time for a negative lookup were measured.

The keys were randomly chosen from $[2^{28}]$; the hash functions were random polynomials from $\{h\colon x \mapsto (\sum_{i=0}^{3} a_i x^i) \bmod p \bmod m \mid a_0, \ldots, a_3 \in [2^{28}]\}$, where p is a large prime number and m is the number of blocks for *cuckoo-block* and the size of the array otherwise.

Insertion for all cuckoo hashing variants was implemented in the random-walk fashion, see Section 2.2. The experiments[2] were executed for different values of ε and d. For each of the cuckoo hashing variants, each task, and for each ε we noted the "best" d, which minimized the measured time for the respective operation. Further, the time needed by linear probing was measured (tables 1 and 2).

[2]Processor: Intel(R) Pentium(R) 4 CPU 2.40 GHz stepping 07 (8 K L1 cache, 512 K L2 cache), board: GA-8PE800 i845PE ATX, RAM: DDR-333 512 MB, environment: SuSE Linux 9.1 (kernel 2.6.5), compiler: Intel C Compiler for Linux, version 8.0, optimization options: -O3 -xN -ipo

ε	cuckoo-d		cuckoo-block		cuckoo-lp		lp
	time	d	time	d	time	d	time
0.5	1.618	3	0.924	5	0.908	5	0.38
0.2	2.142	3	0.938	8	0.94	5	0.388
0.1	2.678	4	0.952	16	0.98	8	0.4
0.05	3.376	5	0.966	16	1.024	11	0.418
0.02	4.46	6	1.002	32	1.094	15	0.466
0.01	5.354	7	1.02	32	1.152	18	0.546

Table 1: Average insertion time in μs and the "best" d for a key when filling an empty array

ε	unsuccessful search							successful search						
	cuckoo-d		cuckoo-block		cuckoo-lp		lp	cuckoo-d		cuckoo-block		cuckoo-lp		lp
	time	d	time	d	time	d	time	time	d	time	d	time	d	time
0.5	1.286	3	0.814	2	0.816	2	0.378	0.846	3	0.708	2	0.794	2	0.354
0.2	1.278	3	0.82	3	0.806	2	0.43	0.838	3	0.76	3	0.788	2	0.36
0.1	1.272	3	0.82	3	0.814	3	0.548	0.834	3	0.766	3	0.806	3	0.368
0.05	1.678	4	0.826	4	0.81	3	0.934	1.048	4	0.79	4	0.814	5	0.384
0.02	2.086	5	0.834	5	0.81	3	3.012	1.24	5	0.802	5	0.82	6	0.416
0.01	2.092	5	0.834	6	0.808	3	10.114	1.24	5	0.81	6	0.82	6	0.462

Table 2: Average lookup time for a search in μs and the "best" d

Looking at Tables 1 and 2 one notes that decreasing ε forces that larger d are chosen but affects the insertion time and the negative lookup time of *cuckoo-block* and *cuckoo-lp* only a little. On the other hand, *cuckoo-d* is affected more: each additional hash function needed to make up for a smaller ε increases the cost for a negative lookup operation by about $0.4\,\mu$s. Moreover, we observe a dramatic increase of the negative lookup time for *lp* if ε goes below 0.02. For positive lookups, linear probing cannot be beaten on average for sets of the size considered here; but one should bear in mind that the worst-case lookup time is as slow as an unsuccessful search.

In a second experiment, randomly chosen keys were inserted into an empty table with $2 \cdot 10^7$ cells as long as possible for the cuckoo hashing variants with two hash functions. An overall time bound Kn for some constant $K = K(\varepsilon)$ was fixed beforehand. This gives estimates on the maximal possible space utilization for a given d (Table 3). The experiment shows that *cuckoo-lp* packs the keys more tightly than *cuckoo-block* at least in the static case and for sets of the size considered here. For comparison, we have listed the bounds on ε for a given d that result from numerically optimizing with bound (2) and from applying Theorem 1.

d	cuckoo-lp	cuckoo-block	A	B	d	cuckoo-lp	cuckoo-block	A	B
2	0.038394	0.115584	0.535156	0.7358	7	0.000178	0.003828	0.022396	0.1586
3	0.007117	0.043228	0.208261	0.5413	8	0.000113	0.002393	0.014370	0.1167
4	0.001975	0.02061	0.105351	0.3983	9	0.000076	0.001551	0.009402	0.0859
5	0.000724	0.01102	0.059569	0.2931	10	0.000062	0.001024	0.006234	0.0632
6	0.000332	0.006375	0.035834	0.2156	11	0.000048	0.000686	0.004176	0.0465

Columns labeled with A denote the smallest ε satisfiying the relation (2) and columns labeled with B denote the smallest ε satisfying the conclusion of Lemma 1.

Table 3: Smallest ε for given d; $n \approx 2 \cdot 10^7$