An Analysis of Random-Walk Cuckoo Hashing

Alan Frieze[*], Páll Melsted
Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh PA 15213
U.S.A.

Michael Mitzenmacher[†]
School of Engineering and Applied Sciences
Harvard University
Cambridge MA 02138
U.S.A.

Abstract

In this paper, we provide a polylogarithmic bound that holds with high probability on the insertion time for cuckoo hashing under the random-walk insertion method. Cuckoo hashing provides a useful methodology for building practical, high-performance hash tables. The essential idea of cuckoo hashing is to combine the power of schemes that allow multiple hash locations for an item with the power to dynamically change the location of an item among its possible locations. Previous work on the case where the number of choices is larger than two has analysed breadth-first search, which is both inefficient in practice and currently has only a polynomial upper bound on the insertion time that holds with high probability. On the other hand it does have expected constant amortized insertion time. Here we significantly advance the state of the art by proving a polylogarithmic bound that holds with high probability on the more efficient random-walk method, where items repeatedly kick out random blocking items until a free location for an item is found.

1 Introduction

Cuckoo hashing [13] provides a useful methodology for building practical, high-performance hash tables by combining the power of schemes that allow multiple hash locations for an item (e.g., [1, 2, 3, 14]) with the power to dynamically change the location of an item among its possible locations. Briefly (more detail is given in Section 2), each of n items x has d possible locations $h_1(x), h_2(x), \ldots, h_d(x)$, where d is typically a small constant and the h_i are hash functions, typically assumed to behave as independent fully random hash functions. (See [12] for some theoretical justification of this assumption, as well as [13] for related experimental results.) We assume each location can hold only one item. When an item x is inserted into the table, it can be placed immediately if one of its d locations is currently empty. If not, one of the items in its d locations must be displaced and moved to another of its d choices to make room for x. This item in turn may need to displace another item out of one its d locations. Inserting an item may require a sequence of moves, each maintaining the invariant that each item remains in one of its d potential locations, until no further evictions are needed. Further variations of cuckoo hashing, including possible implementation designs, are considered in for example [5, 6, 8, 9, 10].

It is often helpful to place cuckoo hashing in a graph theoretic setting, with each item corresponding to a node on one side of a bipartite graph, each location corresponding to a node on the other side of a bipartite graph, and an edge between an item x and a location b if b is one of the d

[*]Supported in part by NSF Grant CCF-0502793.
[†]Supported in part by NSF Grant CNS-0721491 and grants from Cisco Systems Inc., Yahoo!, and Google.

locations where x can be placed. In this case, an assignment of items to locations forms a matching and a sequence of moves that allows a new item to be placed corresponds to a type of augmenting path in this graph. We call this the cuckoo graph (and define it more formally in Section 2).

The case of $d = 2$ choices is notably different than for other values of d. When $d = 2$, after the first choice of an item to kick out has been made, there are no further choices as one walks through the cuckoo graph to find an augmenting path. Alternatively, in this case one can think of the cuckoo graph in another form, where the nodes represent locations and items correspond to edges between the locations, with each item connecting the two locations corresponding to it. Because of these special features of the $d = 2$ case, its analysis appears much simpler, and the theory for the case where there are $d = 2$ location choices for each item is well understood at this point [4, 11, 13].

The case where $d > 2$ remains less well understood, although values of d larger than 2 rate to be important for practical applications. The key question is which item to move if the d potential locations for a newly inserted item x are occupied. A natural approach in practice is to pick one of the d locations randomly, replace the item y at that location with x, and then try to place y in one of its other $d - 1$ location choices [6]. If all of the locations for y are full, choose one of the other $d-1$ locations (other than the one that now contains x, to avoid the obvious cycle) randomly, replace the item there with y, and continue in the same fashion. At each step (after the first), place the item if possible, and if not randomly exchange the item with one of $d - 1$ choices. We refer to this as the random-walk insertion method for cuckoo hashing.

There is a clear intuition for how this random walk on the locations should perform. If a fraction f of the items are adjacent to at least one empty location in the corresponding graph, then we might expect that each time we place one item and consider another, we should have approximately a probability f of choosing an item adjacent to an empty location. With this intuition, assuming the load of the hash table is some constant less than 1, the time to place an item would be at most $O(\log n)$ with high probability and $O(1)$ in expectation.[1]

Unfortunately, it is not clear that this intuition should hold true; the intuition assumes independence among steps when the assumption is not necessarily warranted. Bad substructures might arise where a walk could be trapped for a large number of steps before an empty location is found. Indeed, analyzing the random-walk approach has remained open, and is arguably the most significant open question for cuckoo hashing today.

Because the random-walk approach has escaped analysis, thus far the best analysis for the case of $d > 2$ is due to Fotakis et al. [6], and their algorithm uses a breadth-first search approach. Essentially, if the d choices for the initial item x are filled, one considers the other choices of the d items in those locations, and if all those locations are filled, one considers the other choices of the items in those locations, and so on. They prove a constant expected time bound for an insertion for a suitably sized table and a constant number of choices, but to obtain a high probability bound under their analysis requires potentially expanding a logarithmic number of levels in the breadth-first search, yielding a polynomial bound ($O(n^{\zeta})$, for some constant $\zeta < 1$), on the time to find an empty location with high probability. It was believed this should be avoidable by analyzing the random-walk insertion method. Further, in practice, the breadth-first search would not be the choice for most implementations because of its increased complexity and memory requirements over the random-walk approach.

In this paper, we demonstrate that, with high probability, for sufficiently large d the cuckoo graph has certain structural properties that yield that on the insertion of any item, the time re-

[1]Generally, an event $\mathcal{E}_n$ occurs *with high probability* if $\mathsf{P}(\mathcal{E}_n) = 1 - O(1/n^{\alpha})$ for some constant $\alpha > 0$. However, please see the discussion in Section 2, just before Theorem 1, for more details on our usage here.

quired by the random-walk insertion method is polylogarithmic in n also with high probability. The required properties and the intuition behind them are given in subsequent sections. Besides providing an analysis for the random-walk insertion method, our result can be seen as an improvement over [6] in that the bound holds for every possible starting point for the insertion (with high probability). The breadth-first search of [6] gives constant expected time, implying polylogarithmic time with probability $1 - o(1)$. However when inserting $\Omega(n)$ elements into the hash table, the breadth-first search algorithm cannot guarantee a sub-polynomial running time for the insertion of each element. This renders the breadth-first search algorithm unsuitable for many applications that rely on guarantees for individual insertions and not just expected or amortized time bounds.

While the results of [6] provide a starting point for our work, we require further deconstruction of the cuckoo graph to obtain our bound on the performance of the random-walk approach.

Simulations in [6] (using the random-walk insertion scheme), indicate that constant expected insertion time is possible. While our guarantees do not match the running time observed in simulations, they give the first clear step forward on this problem for some time.

We note that since the publication of our results in RANDOM 2009, Fountoulakis, Panagioutou and Steger [7] have been able to show polylogarithmic insertion time with high probability for all $d \geq 3$, for any number of locations above the threshold necessary for the existence of a matching from elements into locations. Their results offer an improved analysis of the methodology set out in this paper.

2 Definitions and Results

We begin with the relevant definitions, followed by a statement of and explanation of our main result.

Let $h_1, \ldots h_d$ be independent fully random hash functions $h_i : [n] \rightarrow [m]$ where $m = (1 + \varepsilon)n$. This is clearly sufficient to cover the general case where $h : U \rightarrow [m]$ for some universe of keys U. The necessary number of choices d will depend on ϵ, which gives the amount of extra space in the table. We let the *cuckoo graph* G be a bipartite (multi-)graph with a vertex set $L \cup R$ and an edge set $\bigcup_{x \in L} \{(x, h_1(x)), \ldots (x, h_d(x))\}$, where $L = [n]$ and $R = [m]$. We refer to the left set L of the bipartite graph as *items* and the right set R as *locations*. We use $\log n$ for $\log_e n$ throughout.

An assignment of the items to the locations is a left-perfect matching M of G such that every item $x \in L$ is incident to a matching edge. The vertices $F \subseteq R$ not incident to the matching M are called *free* vertices. For a vertex v the *M-distance* to a free vertex is the length of the shortest M-alternating path from v to a free vertex. We do not include matching edges in this count.

We present the algorithm for insertion as Algorithm 1 below. The algorithm augments the current matching M with an augmenting path P. An item is assigned to a free neighbor if one exists; otherwise, a random neighbor is chosen to displace from its location, and this is repeated until an augmenting path is found. (In some rare cases, the path P will contain cycles. These will be ignored in the augmentation). In practice, one generally sets an upper bound on the number of moves allowed to the algorithm, and a failure occurs if there remains an unassigned item after that number of moves. Such failures can be handled by additional means, such as stashes to store items that cause failures [9]. Note that a good estimate of the appropriate upper bound on the number of moves before declaring a failure is key for keeping the stash reasonably sized (preferably constant-sized). Also, because we do not augment the matching until an appropriate free cell is found, the algorithm may experience cycles as we perform the random walk. Because of this, edges are exclusive-ored into the path, so that repeated edges can be removed.

We note that our analysis that follows also holds when the table experiences deletions (and even

Algorithm 1 Insert-node

```
procedure INSERT-NODE($G$,$M$,$u$)
    $P \leftarrow ()$
    $v \leftarrow u$
    $i \leftarrow d + 1$
    loop
        if $h_j(v)$ is not covered by $M$ for some $j \in \{1, \ldots, d\}$ then
            $P \leftarrow P \oplus (v, h_j(v))$
            return Augment($M$,$P$)
        else
            Let $j \in_R \{1, \ldots, d\} \setminus \{i\}$ and $w$ be such that $(h_j(v), w) \in M$
            $P \leftarrow P \oplus (v, h_j(v)) \oplus (h_j(v), w)$
            $v \leftarrow w$
            $i \leftarrow j$
        end if
    end loop
end procedure
```

Algorithm 2 Simple-Insert-node

```
procedure SIMPLE-INSERT-NODE($G$,$M$,$u$)
    $v \leftarrow u$
    loop
        if $h_j(v)$ is not covered by $M$ for some $j \in \{1, \ldots, d\}$ then
            $M \leftarrow M \cup \{(v, h_j(v))\}$
            return $M$
        else
            Let $j \in_R \{1, \ldots, d\}$ and $(w, h_j(v)) \in M$
            $M \leftarrow (M \setminus \{(w, h_j(v))\}) \cup \{(v, h_j(v))\}$
            $v \leftarrow w$
        end if
    end loop
end procedure
```

re-insertion of deleted items). This is because our result is based on the structure of the underlying graph G, and not on the history that led to the specific current matching. The statement of the main result is that given that G satisfies certain conditions, which it will with high probability, the insertion time is polylogarithmic with high probability. It is important to note that we have two distinct probability spaces, one for the hash functions which induce the graph G, and another for the randomness employed by the algorithm. For the probability space of hash functions, we say that an event $\mathcal{E}_n$ occurs *with high probability* if $\mathsf{P}(\mathcal{E}_n) = 1 - O(n^{-2d})$. For the probability space of randomness used by the algorithm we use a common definition of *with high probability*, namely that the event occurs with probability $1 - O(n^{-\alpha})$ for some $\alpha > 0$. In the following, log refers to natural logarithms, i.e. to the base e.

Although the analysis is focused on Algorithm 1, which keeps track of the path used, our analysis goes through similarly for the simpler Algorithm 2, which instead modifies the matching at each step and can immediately reverse the last move made. This is because our argument works regardless of the initial vertex of the random walk or the current matching and is based on showing that at any point there are many simple paths the random walk can take to reach a free vertex. Since the two algorithms behave the same way for simple walks, our results follow for this algorithm as well.

Theorem 1 *Let ϵ and d be such that if $\varepsilon < \frac{1}{6}$, then $d \geq 4 + 2\varepsilon - 2(1+\varepsilon)\log\left(\frac{\varepsilon}{1+\varepsilon}\right)$, and if $\varepsilon \geq \frac{1}{6}$, then $d \geq 8$. Let $\gamma_0 = \frac{d+\log d}{(d-1)\log(d/3)}$ and $\gamma_1 = \frac{d+\log d}{(d-1)\log(d-1)}$. Conditioned on an event of probability $1 - O(n^{4-2d})$ regarding the structure of the cuckoo graph G, the expected time for insertion into a cuckoo hash-table using Algorithms 1 and 2 is $O\left(\log^{1+\gamma_0+2\gamma_1} n\right)$. Furthermore, the insertion time is $O\left(\log^{2+\gamma_0+2\gamma_1} n\right)$ with high probability.*

We have worked reasonably hard so that the constants γ_0, γ_1 are small for large d.

The algorithm will fail if the graph does not have a left-perfect matching, which happens with probability $O(n^{4-2d})$ [6]. We show that all necessary structural properties of G hold with probability $1 - O(n^{-2d})$, so that the probability G fails to have the right structure is dominated by the probability that G has no left-perfect matching.

At a high level, our argument breaks down into a series of steps. First, we show that the cuckoo graph expands suitably so that most vertices are within $O(\log\log n)$ M-distance from a free vertex. Calling the free vertices F and this set of vertices near to the free vertices S, we note that if we reach a vertex in S, then the probability of reaching F from there over the next $O(\log\log n)$ steps in the random walk process is inverse polylogarithmic in n, so we have a reasonable chance of getting to a free vertex and finishing. We next show that if the cuckoo graph has a suitable expansion property, then from any starting vertex, we reach a vertex of S through the random walk in only $O(\log n)$ steps, with probability $\Omega(\log^{-\gamma_1} n)$. This second part is the key insight into this result; instead of trying to follow the intuition to reach a free vertex in $O(\log n)$ steps, we aim for the simpler goal of reaching a vertex S close to F and then complete the argument.

We prove the necessary lemmas below. We start with some basic structural lemmas on the cuckoo graph that we require. Lemma 2 shows that large subsets of R have large neighborhoods in L. Lemma 3 shows that at least half the vertices of L are at a constant M-distance from F. This is an improvement over similar previous results that we utilize subsequently.

From here we prove the key points. First, we show that the number of R-vertices at M-distance k from F shrinks geometrically with k, which proves that the number of vertices at M-distance $\Omega(\log\log n)$ from F is sufficiently small. Next we show that successive levels in a breadth first search started at the vertex corresponding to an item to be inserted expands very fast i.e. at a rate close to $d-1$. This will allow us to conclude that a random walk will quickly reach a vertex within M-distance $O(\log\log n)$ from F. Putting it all together then yields our main result.

3 Expansion and Related Cuckoo Graph Structure

We first show that large subsets of R have corresponding large neighborhoods in L. For $S \subseteq L \cup R$ we let $N(S) = \{u \notin S : \exists v \in S \text{ such that } (u,v) \in E(G)\}$.

Lemma 2 *If $1/2 \le \beta \le 1 - \frac{2d^3 \log n}{n}$ and $\alpha = d - 1 - \frac{d+\log d}{1-\log(1-\beta)} > 0$ then with high probability for every subset $Y \subseteq R$ of size $|Y| = (\beta+\varepsilon)n$ we have $X = N(Y) \subseteq L$ is of size at least $n\left(1 - \frac{1-\beta}{\alpha}\right)$.*

For some intuition as to the use of this lemma, consider the set X' of vertices of L at M-distance at most i, with $|X'| = \beta n$ where $\beta \ge 1/2$, and let Y be the set of vertices in R matched with X' along with the free vertices in R. Then Y has size $(\beta+\varepsilon)n$, and it's neighborhood has size at least $n\left(1 - \frac{1-\beta}{\alpha}\right)$. This means that $R \setminus X'$, the set of vertices at M-distance more than $i+1$ is α times smaller than $R \setminus X'$, the set of vertices at M-distance more than i.

Proof: We show that with high probability, there does not exist a pair X, Y such that $|Y| = (\beta+\varepsilon)n$, $|X| < n - \frac{(1+\varepsilon)n - |Y|}{\alpha}$ and X is the neighborhood of Y in G.

Let $S = L \setminus X$ and $T = R \setminus Y$. Then our assumptions on X, Y imply that $|S| \ge \frac{(1+\varepsilon)n-|Y|}{\alpha} = \frac{n(1-\beta)}{\alpha}$ and $|T| = (1+\varepsilon)n - (\beta+\varepsilon)n = (1-\beta)n$. Each vertex in L has all of its edges in T with probability $\left(\frac{1-\beta}{1+\varepsilon}\right)^d$ independently of other vertices. Thus for any T the size of S is a binomially distributed random variable, $\mathrm{Bin}(n, \left(\frac{1-\beta}{1+\varepsilon}\right)^d)$. Thus the probability of the existence of a pair X, Y is at most

$$\binom{(1+\varepsilon)n}{(\beta+\varepsilon)n} \mathsf{P}\left(|S| \ge \frac{(1-\beta)n}{\alpha}\right) = \binom{(1+\varepsilon)n}{(1-\beta)n} \mathsf{P}\left(\mathrm{Bin}\left(n, \left(\frac{1-\beta}{1+\varepsilon}\right)^d\right) \ge \frac{(1-\beta)n}{\alpha}\right)$$
$$\le \left(e\frac{1+\varepsilon}{1-\beta}\right)^{(1-\beta)n} \left(e\frac{\left(\frac{1-\beta}{1+\varepsilon}\right)^d}{\frac{1-\beta}{\alpha}}\right)^{\frac{1-\beta}{\alpha}n} = \left(\frac{\alpha e^{1+\alpha}(1+\varepsilon)^{\alpha-d}}{(1-\beta)^{\alpha-d+1}}\right)^{\frac{1-\beta}{\alpha}n} \qquad (1)$$

where we have used the inequality $\mathsf{P}\left(\mathrm{Bin}(n,p) \ge \rho p n\right) \le \left(\frac{e}{\rho}\right)^{\rho p n}$. (While there are tighter bounds, this is sufficient for our purposes.)

Taking logarithms, dropping the $1+\varepsilon$ factor, and letting $D = d + \log d$ and $B = \log(1-\beta)$ and using the definition of α gives

$$\frac{\log(RHS(1))}{(1-\beta)n/\alpha} \le \log\left(\frac{d-1-D/(1-B)}{d}\right) + D - \frac{D}{1-B} + \frac{DB}{1-B}$$
$$= \log\left(\frac{d-1-D/(1-B)}{d}\right) \le \log\left(\frac{d-1}{d}\right).$$

Then we can upper bound (1) by

$$\left(\frac{d-1}{d}\right)^{\frac{1-\beta}{\alpha}n} \le \exp\left(-\frac{1}{d}\frac{2d^3\log n}{d}\right) \le n^{-2d}$$

□

The following Lemma corresponds to Lemma 8 in [6]. We give an improved bound on an important parameter k^* which gives an improvement for the running time of the breadth-first search algorithm. We require this improvement over the analysis of [6] for our result.

Lemma 3 *Assume $d \geq 8$, and furthermore if $\varepsilon \leq \frac{1}{6}$ also assume $d \geq 4 + 2\varepsilon - 2(1+\varepsilon)\log\left(\frac{\varepsilon}{1+\varepsilon}\right)$. Then with high probability the number of vertices in L at M-distance at most k^* from F is at least $\frac{n}{2}$, where $k^* = 4 + \frac{\log\left(\frac{1}{6\varepsilon}\right)}{\log\left(\frac{d}{6}\right)}$ if $\varepsilon \leq \frac{1}{6}$ and $k^* = 5$ if $\varepsilon \geq \frac{1}{6}$.*

Proof

Let M be any left-perfect matching of the cuckoo graph G. Let $Y_0 = F$ be the free vertices in R and $X_1 = N(Y_0)$. The vertices of X_1 are adjacent to free vertices, and thus are at augmentation M-distance 1. Let Y_1 be the matching neighbors of vertices in X_1 in addition to Y_0. In general, given X_i we let

$$Y_i = Y_0 \cup \{y \in R : (x,y) \in M \text{ for some } x \in X_i\} \qquad \text{and} \qquad X_{i+1} = N(Y_i) \tag{2}$$

Note that $|Y_0| = \varepsilon n$ and $|Y_i| = |X_i| + \varepsilon n$ for $i \geq 1$. Thus to show that $|X_i| \geq \frac{n}{2}$ for some i it is enough to concentrate on expansion properties of the Y sets in R.

Claim 4 *With high probability any set $Y \subseteq R$ of size $\xi(1+\varepsilon)n$ has a neighborhood of size at least*

$$\begin{cases} \frac{d}{6}\xi n & \text{if } \xi \leq \frac{6}{d^2} & \textbf{Case 1} \\ \frac{1}{d}n & \text{if } \frac{6}{d^2} \leq \xi \leq \frac{1}{d} & \textbf{Case 2} \\ \frac{1}{5}n & \text{if } \frac{1}{d} \leq \xi \leq \frac{1}{5} & \textbf{Case 3} \\ \frac{3}{10}n & \text{if } \frac{1}{5} \leq \xi \leq \frac{3}{10} & \textbf{Case 4} \\ \frac{2}{5}n & \text{if } \frac{3}{10} \leq \xi \leq \frac{2}{5} & \textbf{Case 5} \\ \frac{1}{2}n & \text{if } \frac{2}{5} \leq \xi & \textbf{Case 6} \end{cases}$$

Using Claim 4 we see that it will take at most $\log_{\frac{d}{6}}\left(\frac{\frac{6}{d^2}}{\varepsilon}\right)$ steps to go from size εn to at least $\frac{6}{d^2}n$ and 5 additional steps to go from $\frac{6}{d^2}n$ to $\frac{1}{2}n$. This makes for a total of at most

$$\begin{aligned} i^* &\leq \log_{\frac{d}{6}}\left(\frac{\frac{6}{d^2}}{\frac{\varepsilon}{1+\varepsilon}}\right) + 6 \\ &= \frac{-\log(\varepsilon) + \log 6}{\log\left(\frac{d}{6}\right)} + 6 - 2 \cdot \frac{\log d - \frac{1}{2}\log(1+\varepsilon)}{\log d - \log 6} \\ &\leq 4 + \frac{\log\left(\frac{1}{6\varepsilon}\right)}{\log\left(\frac{d}{6}\right)} \end{aligned}$$

steps to go from Y_0 of size εn to X_{i^*} of size at least $\frac{1}{2}n$. Note that if $\varepsilon \geq \frac{1}{6}$ we can use $i^* \leq 5$, since then $\xi \geq \frac{1}{7} \geq \frac{1}{d}$. This concludes the proof of Lemma 3 assuming Claim 4, which we now prove below. □

Proof of Claim 4

Assume $d \geq 8$ and if $\varepsilon \leq \frac{1}{6}$ we require $d \geq 4 + 2\varepsilon - 2(1+\varepsilon)\log\left(\frac{\varepsilon}{1+\varepsilon}\right)$. For $\varepsilon \leq 1$ this implies $\log\left(\frac{\varepsilon}{1+\varepsilon}\right) \geq 1 - \frac{d-2}{2(1+\varepsilon)}$. Now let $Y \subseteq R$, $|Y| = \xi(1+\varepsilon)n$ and assume $|Y| \geq \varepsilon n$. So $\frac{\varepsilon}{1+\varepsilon} \leq \xi$ and for $\varepsilon \geq 1$ we have $\xi \geq \frac{1}{2}$.

We first upper bound the probability that any set Y has exactly sn neighbors in L. We then argue from the fact that our bound is monotone increasing in s that this will suffice to eliminate the possibility for fewer than sn neighbors. Our bound is

$$\begin{aligned}&\binom{(1+\varepsilon)n}{\xi(1+\varepsilon)n}\binom{n}{sn}\left((1-\xi)^d\right)^{(1-s)n}(1-(1-\xi)^d)^{sn} \\ &\leq \exp\left(n\left((1+\varepsilon)H(\xi)+H(s)+(1-s)d\log(1-\xi)+s\log(1-(1-\xi)^d)\right)\right)\end{aligned} \tag{3}$$

where $H(x) = -\xi\log(\xi)-(1-\xi)\log(1-\xi)$ is the standard entropy (expressed in natural logarithms, as log represents $\log_e$) and $\binom{n}{k} \leq e^{nH(\frac{k}{n})}$. We will assume that

$$s \leq 1-(1-\xi)^d. \tag{4}$$

We observe that the product of the terms involving s in (3) is monotone increasing in s up to this point.

It is enough now to show that

$$\Phi_1 = (1+\varepsilon)H(\xi)+H(s)+(1-s)d\log(1-\xi)+s\log(1-(1-\xi)^d) \tag{5}$$

is strictly negative to obtain a bound of $e^{-\Omega(n)}$ on the probability that Y has sn neighbors. If we show that this holds for some particular value $s^* \leq 1-(1-\xi)^d$, then by a simple union bound the probability that Y has s^*n neighbors or fewer is bounded by $O(ne^{-\Omega(n)}) = e^{-\Omega(n)}$ as well.

Case 1: $\frac{\varepsilon}{1+\varepsilon} \leq \xi \leq \frac{6}{d^2}$.

We start with (5) and write $s = \gamma d\xi$ and assume $s \leq \frac{1}{d}$. We also use the upper bound $H(\xi) \leq \xi(1-\log\xi)$ and get

$$\Phi_1 \leq (1+\varepsilon)\xi(1-\log\xi)+\gamma d\xi(1-\log(\gamma d\xi))+(1-\gamma d\xi)d\log(1-\xi)+\gamma d\xi\log(1-(1-\xi)^d) \tag{6}$$

Now use $\log(1-\xi) \leq -\xi-\frac{\xi^2}{2}$, $(1-\gamma d\xi)d \geq (1-\frac{1}{d})d = d-1$ and $\log(1-(1-\xi)^d) \leq \log d\xi$ to obtain

$$\Phi_1 \leq \Phi_2 = \xi\left((1+\varepsilon)(1-\log\xi)+\gamma d(1-\log\gamma)-(d-1)\right)-(d-1)\frac{\xi^2}{2} \tag{7}$$

Since $\xi \geq \frac{\varepsilon}{1+\varepsilon}$ and $(1+\varepsilon)\left(1-\log\left(\frac{\varepsilon}{1+\varepsilon}\right)\right) \leq \frac{d-2}{2}$, we see that using $\gamma = 6$ gives $\gamma(1-\log\gamma) < 1/2$ and so the coefficient of ξ in (7) is negative.

Case 2: $\frac{6}{d^2} \leq \xi \leq \frac{1}{d}$.

If $|Y| \in \left[\frac{6m}{d^2}, \frac{m}{d}\right]$ we can choose a subset Y' of Y of size exactly $\frac{6m}{d^2}$ and use $|N(Y)| \geq |N(Y')| \geq \frac{n}{d}$ from Case 1.

Case 3: $\frac{1}{d} \leq \xi$.

Φ_1 is increasing with respect to ε and $\xi \geq \frac{\varepsilon}{1+\varepsilon}$. So we can take $\varepsilon = \frac{\xi}{1-\xi}$ in order to bound Φ_1.

$$\Phi_1 \leq \Phi_3 = \frac{1}{1-\xi}H(\xi)+H(s)+(1-s)d\log(1-\xi)+s\log(1-(1-\xi)^d) \tag{8}$$

The derivative of the RHS of (8) with respect to ξ is given by

$$d\left(\frac{s(1-\xi)^{d-1}}{1-(1-\xi)^d}-\frac{1-s}{1-\xi}\right)-\frac{\log\xi}{(1-\xi)^2} = \frac{-d(1-\xi)\left(1-\frac{s}{1-(1-\xi)^d}\right)-\log\xi}{(1-\xi)^2} \tag{9}$$

Note that both $-d(1-\frac{s}{1-(1-\xi)^d})$ and $-\frac{\log \xi}{1-\xi}$ are decreasing in ξ, so it is enough to verify that the numerator in the RHS of (9) is negative at the left endpoint $\xi = 1/d$.

For $s \leq \frac{1}{5}$ we get

$$\begin{aligned} &- d\left(1-\frac{1}{d}\right)\left(1-\frac{s}{1-(1-\frac{1}{d})^d}\right) + \log d \\ \leq &- (d-1)\left(1-\frac{0.2}{1-e^{-1}}\right) + \log d \quad \leq -0.68(d-1) + \log d \end{aligned}$$

which is negative for $d \geq 8$.

So for $s \leq \frac{1}{5}$ we see that Φ_3 is decreasing in ξ. Since $\xi \geq \frac{1}{d}$, the maximum is obtained at $\xi = \frac{1}{d}$ and plugging in $\xi = \frac{1}{d}$ yields

$$\Phi_3 \leq \Phi_4 = \frac{d}{d-1} H\left(\frac{1}{d}\right) + H(s) + (1-s)d\log\left(1-\frac{1}{d}\right) + s\log\left(1-\left(1-\frac{1}{d}\right)^d\right). \tag{10}$$

Taking the derivative of Φ_4 with respect to d gives

$$\frac{(d-1)(1-(1-\frac{1}{d})^d - s)(1+(d-1)\log(1-\frac{1}{d})) - (\log d)(1-(1-\frac{1}{d})^d)}{(d-1)^2(1-(1-\frac{1}{d})^d)}. \tag{11}$$

Note that

$$0 \leq (d-1)\left(1+(d-1)\log\left(1-\frac{1}{d}\right)\right) \leq (d-1)/d.$$

We want to show that $\Phi_4' < 0$ and so we can drop the contribution from s. Factoring $1-(1-1/d)^d$ from the remainder of the numerator gives at most $(d-1)/d - \log d < 0$.

For an upper bound on Φ_4 we plug in $d = 8$ into equation (10) and get

$$\Phi_4 \leq \frac{8}{7} H\left(\frac{1}{8}\right) + H(s) + (1-s)8\log\left(\frac{7}{8}\right) + s\log\left(1-\left(\frac{7}{8}\right)^8\right). \tag{12}$$

The derivative of the RHS of (12) with respect to s is

$$\log\left(\frac{1-s}{s}\right) + s\log\left(\frac{1-(\frac{7}{8})^8}{(\frac{7}{8})^8}\right)$$

which is positive for $s \leq 1/5$. Thus plugging in $s = \frac{1}{5}$ into the RHS of (12) we get

$$\frac{8}{7} H\left(\frac{1}{8}\right) + H\left(\frac{1}{5}\right) + \frac{32}{5}\log\left(\frac{7}{8}\right) + \frac{1}{5}\log\left(1-\left(\frac{7}{8}\right)^8\right) < -0.006$$

which is strictly negative.

Case 4: $1/5 \leq \xi \leq 3/10$.

Starting from (8) we see that the derivative of Φ_3 with respect to d is

$$\left(1-\frac{s}{1-(1-\xi)^d}\right)\log(1-\xi)$$

which is negative since we are assuming $s \leq 1-(1-\xi)^d$. So we will use $d=8$ from now on to get an upper bound on Φ_3 and obtain

$$\Phi_3 \leq \Phi_5 = \frac{1}{1-\xi}H(\xi) + H(s) + 8(1-s)\log(1-\xi) + s\log(1-(1-\xi)^8). \tag{13}$$

The derivative of Φ_5 with respect to ξ is

$$\Phi_5' = \frac{-8(1-\xi)\left(1-\frac{s}{1-(1-\xi)^8}\right) - \log\xi}{(1-\xi)^2}. \tag{14}$$

If $\frac{1}{5} \leq \xi \leq 1$ and $s \leq \frac{1}{2}$, then we can upper bound the numerator by

$$-8(1-\xi)\left(1-\frac{\frac{1}{2}}{1-\left(\frac{4}{5}\right)^8}\right) - \log\xi \leq -3(1-\xi) - \log\xi,$$

which is convex and negative at both endpoints, $\xi = 1/5$ and $\xi = 1$. Thus Φ_5 is decreasing in ξ on $[\frac{1}{5}, 1]$. Evaluating Φ_5 with $\xi = \frac{1}{5}$ and $s = \frac{3}{10}$ gives $\Phi_5 < -0.06$.
Case 5: $3/10 \leq \xi \leq 2/5$.
We evaluate Φ_5 at $\xi = \frac{3}{10}$ and $s = \frac{2}{5}$, and get $\Phi_5 < -0.19$.
Case 6: $2/5 \leq \xi \leq 1$.
Evaluating Φ_5 with $\xi = \frac{2}{5}$ and $s = \frac{1}{2}$ gives $\Phi_5 < -.23$.
This completes the proof of Lemma 3 □

Let $k^* = \max\{4 + \frac{\log\left(\frac{1}{6\varepsilon}\right)}{\log\left(\frac{d}{6}\right)}, 5\}$ and for notational convenience now let Y_k be the vertices in R at M-distance at most k^*+k from F (as opposed to Y_{k+k^*} as in (2)) and let $|Y_k| = (\beta_k+\varepsilon)n$. We note that Lemma 3 guarantees that with high probability at most $\frac{n}{2}$ vertices in R are at M-distance more than k^* from F and so with high probability $\beta_k \geq 1/2$ for $k \geq 0$.

Indeed, we further note that as a byproduct of our Lemma 3, we obtain an improved bound on the expected running time for the breadth-first variation on cuckoo hashing from [6], of $\left(\frac{1}{\varepsilon}\right)^{O(1)}$ instead of $\left(\frac{1}{\varepsilon}\right)^{O(\log d)}$. Although it is something of an aside from our main argument, we present the result below for completeness.

Theorem 5 *The breadth-first search insertion procedure given in [6] runs in* $O\left(\max\left\{d^4\left(\frac{1}{6\varepsilon}\right)^{\frac{1}{1-\frac{\log 6}{\log d}}}, d^5\right\}\right)$ *expected time, provided* $d \geq 8$*. If* $\varepsilon \leq \frac{1}{6}$ *then* $d \geq 4 + 2\varepsilon - 2(1+\varepsilon)\log\left(\frac{\varepsilon}{1+\varepsilon}\right)$ *suffices.*

Proof of Theorem 5
We follow the proof of Theorem 1 in [6]. The breadth-first search insertion procedure takes time $O(|T_v|)$ where T_v is a BFS tree rooted at the newly inserted vertex v, which is grown until a free vertex is found.

The expected size of T_v is bounded above by d^{k^*}, which is at most d^5 for $\varepsilon \geq \frac{1}{6}$ and at most

$$d^{4+\log\left(\frac{1}{6\varepsilon}\right)/\log\left(\frac{d}{6}\right)} = d^4\left(\frac{1}{6\varepsilon}\right)^{\frac{\log d}{\log d - \log 6}} = d^4\left(\frac{1}{6\varepsilon}\right)^{\frac{1}{1-\frac{\log 6}{\log d}}}$$

for $\varepsilon \leq \frac{1}{6}$. □

4 Vertices Near Free Vertices

We now move to showing that for sufficiently large k of size $O(\log\log n)$, a large fraction of the R-vertices are within M-distance k of the free vertices F with high probability. This provides one of the cornerstones of our main result.

Our next lemma might look circular, but what it is saying is that if β_k satifies a certain upper bound then **whp** it will satify an even stronger one.

Lemma 6 *Suppose that $d \geq 8$ and if $\varepsilon \leq \frac{1}{6}$ assume $d \geq 4 + 2\varepsilon - 2(1+\varepsilon)\log\left(\frac{\varepsilon}{1+\varepsilon}\right)$. Then with high probability*

$$1 - \beta_k = O\left(\frac{\log^{\gamma_0} n}{(d-1)^k}\right) \quad \textit{whenever } 1 - \beta_k \geq 2d^3 \log n/n. \tag{15}$$

where $\gamma_0 = \frac{d+\log d}{(d-1)\log(d/3)}$ is defined in Theorem 1.

The reader should not be put off by the fact that the lemma only has content for $k = \Omega(\log\log n)$. It is only needed for these values of k.

Proof: Assume that the high probability event in Lemma 2 occurs and $1 - \beta_k \geq \log n/n$ and the neighborhood X_k of Y_k in L has size at least $n - \frac{(1-\beta_k)n}{\alpha_k}$ where $\alpha_k = d - 1 - \frac{d+\log d}{1-\log(1-\beta_k)}$. Note that for $\beta_k \geq \frac{3}{4}$ and $d \geq 8$ this implies

$$\alpha_k \geq \frac{d}{3} \text{ and } \frac{(d+\log d)/(d-1)}{1-\log(1-\beta_k)} \leq 0.9. \tag{16}$$

First assume that $\beta_0 \geq 3/4$, we will deal with the more general case of $\beta_0 \geq \frac{1}{2}$ later. Recall that $Y_{k+1} = F \cup M(X_k)$ where $M(X_k) = \{y : (x,y) \in M \text{ for some } x \in X_k\}$. Thus $|Y_{k+1}| = (\beta_{k+1} + \varepsilon)n \geq \varepsilon n + n - \frac{(1-\beta_k)n}{\alpha_k}$. This implies that

$$\begin{aligned} 1 - \beta_{k+1} &\leq \frac{1-\beta_k}{\alpha_k} && (17)\\ &= \frac{1-\beta_k}{d-1}\left(1 - \frac{(d+\log d)/(d-1)}{1-\log(1-\beta_k)}\right)^{-1} \\ &\leq \frac{1-\beta_k}{d-1}\exp\left(h\left(\frac{(d+\log d)/(d-1)}{1-\log(1-\beta_k)}\right)\right) && (18)\end{aligned}$$

where $h(t) = t + 3t^2/2$. NWe ote that $(1-t)^{-1} \leq \exp(h(t))$ for $t \in [0, .9]$.

$$(19)$$

$$\leq \frac{1-\beta_k}{d-1}\exp\left(h\left(\frac{(d+\log d)/(d-1)}{1-\log(1-\beta_0)+k\log(d/3)}\right)\right). \tag{20}$$

For (20) we have assumed that $1 - \beta_k \leq 3^k(1-\beta_0)/d^k$, which follows from (16) and (17) provided $\beta_k \geq 3/4$.

For $\beta_0 \in [\frac{1}{2}, \frac{3}{4}]$ note that α_k is increasing in d and β_k. Also starting with $\beta_0 = \frac{1}{2}$ and using $d = 8$ we see numerically that $1 - \beta_3 \leq \frac{\frac{1}{2}}{\alpha_0\alpha_1\alpha_2} \leq \frac{1}{4}$. Thus after at most 3 steps we can assume $\beta_3 \geq 3/4$. To simplify matters we will assume $\beta_0 \geq 3/4$, since doing this will only "shift" the indices by at most 3 and distort the equations by an $O(1)$ factor.

Using inequality (20) repeatedly gives

$$1 - \beta_{k+1} \leq$$

$$\begin{aligned}
&\frac{1-\beta_0}{(d-1)^{k+1}} \exp\left(\frac{d+\log d}{(d-1)\log(d/3)} \sum_{j=0}^{k} \frac{1}{j+\frac{1-\log(1-\beta_0)}{\log(d/3)}} + O\left(\sum_{j=0}^{k} \frac{1}{\left(j+\frac{1-\log(1-\beta_0)}{\log(d/3)}\right)^2}\right)\right) \\
&\leq \frac{1-\beta_0}{(d-1)^{k+1}} \exp\left(\frac{d+\log d}{(d-1)\log(d/3)} \log\left(\frac{1-\log(1-\beta_k)}{1-\log(1-\beta_0)}\right) + O(1)\right) \qquad (21) \\
&= O\left(\frac{\log^{\gamma_0} n}{(d-1)^{k+1}}\right).
\end{aligned}$$

Note that (21) is obtained as follows:

$$\sum_{j=0}^{k} \frac{1}{j+\frac{1-\log(1-\beta_0)}{\log(d/3)}} = \log\left(\frac{k+\zeta}{\zeta}\right) + O(1) \leq \log\left(\frac{1-\log(1-\beta_k)}{1-\log(1-\beta_0)}\right) + O(1) \qquad (22)$$

where $\zeta = \frac{1-\log(1-\beta_0)}{\log(d/3)}$. Now $1-\beta_k \leq 3^k(1-\beta_0)/d^k$ implies that $k \leq \log((1-\beta_0)/(1-\beta_k))/\log(d/3)$. Substituting this upper bound for k into the middle term of (22) yields (21). □

5 BFS Expansion from Any Starting Point

We now require some additional structural lemmas regarding the graph G in order to show that a breadth first search on the graph expands suitably. Proving this expansion it what allows us to show that from any starting point, a random walk on the cuckoo graph quickly reaches a vertex that is near a free vertex.

Lemma 7 *If $k \leq \frac{1}{6}\log_d n$, then with high probability G does not contain a connected subgraph H on $2k+1$ vertices with $2k+3d$ edges, where $k+1$ vertices come from L and k vertices come from R.*

Proof: We put an upper bound on the probability of the existence of such a subgraph using the union bound. Given the vertices of H, H can be constructed by taking a bipartite spanning tree on the $k+1$ vertices from L and k vertices from R and adding $j = 3d$ edges. There are

$$\binom{n}{k+1}\binom{(1+\varepsilon)n}{k}$$

ways of choosing the vertices of H,

$$k^{(k+1)-1}(k+1)^{k-1}$$

possible spanning trees, and

$$(k(k+1))^j$$

ways of choosing the configuration for the j other edges. All $2k+j$ edges occur with probability at most

$$\left(\frac{d}{(1+\varepsilon)n}\right)^{2k+j},$$

giving us that the probability of such a subgraph is at most

$$\binom{n}{k+1}\binom{(1+\varepsilon)n}{k} k^{(k+1)-1}(k+1)^{k-1}\,(k(k+1))^j\left(\frac{d}{(1+\varepsilon)n}\right)^{2k+j}$$
$$\leq \left(\frac{en}{k+1}\right)^{k+1}\left(\frac{e(1+\varepsilon)n}{k}\right)^{k} k^k(k+1)^{k-1}\left(\frac{dk(k+1)}{(1+\varepsilon)n}\right)^{j} d^{2k}\left(\frac{1}{(1+\varepsilon)n}\right)^{2k}$$
$$\leq n\,(ed)^{2k}\left(\frac{dk(k+1)}{n}\right)^{j}. \tag{23}$$

For $k \leq \frac{1}{6}\log_d n$ and $d \geq 4$ and n sufficiently large we have $(ed)^{2k} \leq \exp\left(2\log(d)\frac{\log n}{3\log d}\right) = n^{2/3}$ and $\frac{dk(k+1)}{n}$ is $O(n^{-1+\frac{1}{d}})$. Hence for $j = 3d$ we have the right hand side of (23) is $O(n^{2/3}n^{-3d+3}) = O(n^{-2d})$. □

Lemma 8 *With high probability there do not exist $S \subseteq L$, $T \subseteq R$ such that $N(S) \subseteq T$, $2d^2\log n \leq s = |S| \leq n/d$, $t = |T| \leq (d-1-\theta_s)s$ and*

$$\theta_s = \frac{d+\log d}{\log(n/((d-1)s))} \geq \frac{d+\log d}{\log(n/t)}. \tag{24}$$

Proof: The expected number of pairs S, T satisfying the given conditions can be bounded by

$$\sum_{s=2d^2\log n}^{n/d}\binom{n}{s}\binom{(1+\varepsilon)n}{t}\left(\frac{t}{(1+\varepsilon)n}\right)^{ds} \leq \sum_{s=2d^2\log n}^{n/d}\left(\frac{ne}{s}\right)^{s}\left(\frac{(1+\varepsilon)ne}{t}\right)^{t}\left(\frac{t}{(1+\varepsilon)n}\right)^{ds}$$
$$\leq \sum_{s=2d^2\log n}^{n/d}\left(\frac{ne}{s}\right)^{s} e^{(d-1-\theta_s)s}\left(\frac{t}{(1+\varepsilon)n}\right)^{ds-(d-1-\theta_s)s}$$
$$\leq \sum_{s=2d^2\log n}^{n/d}\left(\frac{t}{s}\frac{e^{d-\theta_s}}{(1+\varepsilon)^{1+\theta_s}}\left(\frac{t}{n}\right)^{\theta_s}\right)^{s} \leq \sum_{s=2d^2\log n}^{n/d}\left((d-1)e^{d-\theta_s}\left(\frac{t}{n}\right)^{\theta_s}\right)^{s}$$
$$\leq \sum_{s=2d^2\log n}^{n/d}\left(\frac{d-1}{d}\right)^{s} = O\left(n^{-\frac{2d^2\log n}{d}}\right) = O\left(n^{-2d}\right).$$

□

Suppose now that we are in the process of adding u to the hash table. For our analysis, we consider exploring a subgraph of G using breadth-first search, starting with the root $u \in L$ and proceeding until we reach F. We emphasize that this is not the behavior of our algorithm; we merely need to establish some properties of the graph structure, and the natural way to do that is by considering a breadth-first search from u.

Let $L_1 = \{u\}$. Let the R-neighbors of u be $w_1, w_2, \ldots, w_d$ and suppose that none of them are in F. Let $R_1 = \{w_1, w_2, \ldots, w_d\}$. Let $L_2 = \{v_1, v_2, \ldots, v_d\}$ where v_i is matched with w_i in M, for $i = 1, 2, \ldots, d$. In general, suppose we have constructed L_k for some k. R_k consists of the R-neighbors of L_k that are not in $R_{\leq k-1} = R_1 \cup \cdots \cup R_{k-1}$ and L_{k+1} consists of the M-neighbors of R_k. Note that we assign an (arbitrary) ordering to the vertices each L_k. An edge (x, y) from L_k is *wasted* if either $y \in R_j$ for some $j < k$ or if there exists $x' \in L_k$ with x' prior to x in the ordering of L_k such that the edge $(x', y) \in G$. We let

$$k_0 = \lfloor \log_{d-1}(n) - 1 \rfloor$$

and $\rho_k = |R_k|, \lambda_k = |L_k|$ for $1 \leq k \leq k_0$. Assume for the moment that

$$|R_k| \cap F = \emptyset \text{ for } 1 \leq k \leq k_0. \tag{25}$$

Lemma 9 *Assume that* (25) *and the high probability events of Lemma 7 and Lemma 8 hold. Then*

$$\rho_{k_0} = \Omega\left(\frac{n}{\log^{\gamma_1} n}\right), \tag{26}$$

where $\gamma_1 = \frac{d+\log d}{(d-1)\log(d-1)}$.

Proof: We can assume that $1-\beta_{k_0} \geq 2d^3 \log n/n$. Otherwise we have that $|R_{\leq k_0-1}| \geq n-2d^3\log n$. But

$$|R_{\leq k_0-1}| \leq \sum_{k=1}^{k_0-1} |R_k| \leq \sum_{k=1}^{k_0-1} (d-1)^{k-1} \leq \frac{(d-1)^{\lfloor \log_{d-1}(n)-1\rfloor -1}}{d-2} \leq \frac{n}{(d-1)(d-2)}$$

and we have a contradiction.

For $1 \leq k \leq k_1 = \left\lfloor \frac{\log_d n}{6} \right\rfloor$ Lemma 7 implies that we generate at most $3d$ wasted edges in the construction of $L_j, R_j, 1 \leq j \leq k$. If we consider the full BFS path tree, where vertices can be repeated, then each internal vertex of the tree L has $d-1$ children. For every wasted edge we cut off a subtree of the full BFS tree. What remains when all the wasted edges have been cut is the regular BFS tree. Clearly the worst case is when all the subtrees cut off are close to the root, in which case the $3d$ wasted edges can at most stunt the growth of the tree for four levels. In this case $d-2$ edges cut at the first three levels, so that there is just one node on each level, and six edges cut off at the fourth level. That is, the worst case in terms of the expansion of the BFS tree is that the first 4 levels offer no expansion, and subsequent levels expand by a factor of $d-1$. This means that in the worst case we have

$$\rho_k > (d-1)^{k-5} \text{ for } 1 \leq k \leq k_1. \tag{27}$$

In particular $\rho_{k_1} = \Omega\left((d-1)^{\frac{\log_d n}{6}}\right) > 2d^2 \log n$ for large enough n and so Lemma 8 applies to the BFS tree at this stage. In general Lemma 8 implies that for $k_1 \leq j \leq k_0$

$$\rho_1 + \rho_2 + \cdots + \rho_j \geq (d-1-\theta_s)s \tag{28}$$

where

$$s = \lambda_1 + \lambda_2 + \cdots + \lambda_j = 1 + \rho_1 + \rho_2 + \cdots + \rho_{j-1}. \tag{29}$$

This follows from the fact that $\lambda_1 = 1$ and (25) implies $\lambda_j = \rho_{j-1}$ for $j \geq 2$.

Now $\lambda_j \leq (d-1)\lambda_{j-1}$ for $j \geq 3$ and so s in (29) satisfies $s \leq 1+d+d(d-1)+\cdots+d(d-1)^{j-2} < (d-1)^j$. Thus θ_s in (24) and (28) satisfies

$$\theta_s \leq \phi_j = \frac{d+\log d}{\log n - j\log(d-1)}.$$

Thus, by (28) and (29) we have (after dropping a term)

$$\rho_j \geq (d-2-\phi_j)(\rho_1 + \rho_2 + \cdots + \rho_{j-1}). \tag{30}$$

An induction then shows that for $\ell \geq 1$,

$$\rho_{k_1+\ell} \geq (\rho_1 + \cdots + \rho_{k_1})(d-2-\phi_{k_1+\ell}) \prod_{k=1}^{\ell-1}(d-1-\phi_{k_1+k}). \tag{31}$$

Indeed the case $\ell = 1$ follows directly from (30). Then, by induction,

$$\begin{aligned}\rho_{k_1+\ell+1} &\geq (\rho_1 + \cdots + \rho_{k_1})(d-2-\phi_{k_1+\ell+1})\left(1+\sum_{k=1}^{\ell}(d-2-\phi_{k_1+k})\prod_{i=1}^{k-1}(d-1-\phi_{k_1+i})\right) &(32)\\ &= (\rho_1 + \cdots + \rho_{k_1})(d-2-\phi_{k_1+\ell+1})\prod_{k=1}^{\ell}(d-1-\phi_{k_1+k}). &(33)\end{aligned}$$

To check (33) we can use induction on ℓ. Re-write

$$1+\sum_{k=2}^{\ell+1}(d-2-\phi_{k_1+k})\prod_{i=2}^{k-1}(d-1-\phi_{k_1+i})$$
$$= 1+\sum_{k'(=k-1)=1}^{\ell}(d-2-\phi_{(k_1+1)+k'})\prod_{i'=(i-1)=1}^{k'-1}(d-1-\phi_{(k_1+1)+i'}) \quad (34)$$

Assume inductively that

$$RHS(34) = \prod_{k'=1}^{\ell}(d-1-\phi_{(k_1+1)+k'}). \quad (35)$$

Then re-write the RHS of (35) as

$$\prod_{k=2}^{\ell+1}(d-1-\phi_{k_1+k}). \quad (36)$$

Then multiply LHS(34) by $d-1-\phi_{k_1+1}$ and tidy up to get the expression in brackets in the RHS of (32) with ℓ replaced by $\ell+1$. Finally observe that multiplying (36) by $d-1-\phi_{k_1+1}$ gives us the required expression for $\ell+1$.

We deduce from (27) and (31) that provided $k_1+\ell \leq k_0$ (which implies $\frac{\phi_{k_1+\ell}}{d-1} \leq \frac{1}{2}$),

$$\begin{aligned}\rho_{k_1+\ell} &\geq ((d-1)^{k_1-5}-1)(d-2-\phi_{k_1+\ell})\prod_{k=1}^{\ell-1}(d-1-\phi_{k_1+k})\\ &= ((d-1)^{k_1-5}-1)\left(1-\frac{1}{d-1-\phi_{k_1+\ell}}\right)(d-1-\phi_{k_1+\ell})\prod_{k=1}^{\ell-1}(d-1-\phi_{k_l+k})\\ &= \left(1-\frac{1}{(d-1)^{k_1-5}}\right)\left(1-\frac{1}{d-1-\phi_{k_1+\ell}}\right)(d-1)^{k_1+l-5}\prod_{k=1}^{\ell}\left(1-\frac{\phi_{k_l+k}}{d-1}\right)\\ &\geq \frac{1}{2}(d-1)^{k_1+\ell-5}\exp\left\{-\frac{1}{d-1}\sum_{k=1}^{\ell}\phi_{k_1+k}-\frac{1}{(d-1)^2}\sum_{k=1}^{\ell}\phi_{k_1+k}^2\right\} &(37)\end{aligned}$$

where we have used the fact that $1-\frac{1}{(d-1)^{k_1-5}} = o(1)$ and $1-\frac{1}{d-1-\phi_{k_1+\ell}} \geq 1-\frac{1}{(d-1)/2} \geq 1-\frac{2}{7}$. To simplify the product we use the inequality $1-x \geq e^{-x-x^2}$ which is valid for $x \in [0, \frac{1}{2}]$ and note that $\frac{\phi_{k_1+k}}{d-1} \leq \frac{\phi_{k_1+l}}{d-1} \leq \frac{1}{2}$.

Note next that

$$\begin{aligned}
\sum_{k=1}^{\ell} \phi_{k_1+k} &= \sum_{k=1}^{\ell} \frac{d+\log d}{\log n - (k_1+k)\log(d-1)} \\
&= \sum_{k=1}^{\ell} \frac{d+\log d}{\log(d-1)} \frac{1}{\frac{\log n}{\log(d-1)} - k_1 - k} \\
&= \frac{d+\log d}{\log(d-1)} \left(\log\left(\frac{\log n}{\log(d-1)} - k_1 \right) - \log\left(\frac{\log n}{\log(d-1)} - k_1 - \ell \right) + O(1) \right) \\
&= \frac{d+\log d}{\log(d-1)} \left(\log\left(\frac{\log n - k_1 \log(d-1)}{\log n - (k_1+\ell)\log(d-1)} \right) + O(1) \right) \\
&\le \frac{d+\log d}{\log(d-1)} (\log\log n + O(1))
\end{aligned}$$

where the sum is estimated in the same way as in (22).

Similarly we find that

$$\begin{aligned}
\sum_{k=1}^{\ell} \phi_{k_1+k}^2 &\le \sum_{k=1}^{\ell} \left(\frac{d+\log d}{\log(d-1)} \right)^2 \frac{1}{\left(\frac{\log n}{\log(d-1)} - k_1 - k \right)^2} \\
&\le \left(\frac{d+\log d}{\log(d-1)} \right)^2 \sum_{k=1}^{\infty} \frac{1}{k^2} \\
&= O(1).
\end{aligned}$$

Thus, putting $\ell = k_0 - k_1$ we get

$$\rho_{k_0} = \Omega\left(\frac{(d-1)^{k_0}}{(\log n)^{(d+\log d)/((d-1)\log(d-1))}} \right)$$

and the lemma follows. □

6 Proof of Theorem 1

Assume that all high probability events identified so far occur. Let S denote the set of vertices $v \in R$ at M-distance at most $\Delta = k^* + (\gamma_0 + \gamma_1)\log_{d-1}\log n + 2K$ from F, where K is a large constant and k^* is given in Lemma 3. Then by Lemma 6

$$|R \setminus S| \le \frac{n}{(d-1)^K \log^{\gamma_1}(n)}.$$

We have used $(d-1)^K$ to "soak up" the hidden constant in the statement of Lemma 6. The requirement $1 - \beta_k \ge 2d^2 \log n/n$ in Lemma 6 can be assumed. Indeed, if it fails then at most $O(\log n)$ vertices are at M-distance greater than Δ from F.

If K is sufficiently large then Lemma 9 implies that

$$|R \setminus S| \le \rho_{k_0}/2. \tag{38}$$

Every vertex $v \in S$ has a path of length $l \le \Delta$ to a free vertex. The probability that the random walk follows this path is $\left(\frac{1}{d-1}\right)^l \ge \left(\frac{1}{d-1}\right)^\Delta$. This is a lower bound on the probability the algorithm finds a free vertex within Δ steps, starting from $v \in S$. We now split the random walk into rounds, and each round into two phases.

The first phase starts when the round starts and ends when the random walk reaches a vertex of S or after k_0 steps (possibly the first phase is empty). Then, the second phase starts and ends either when the random walk reaches a free vertex or after Δ steps, finishing this round. We omit the second phase if the first phase fails to reach S. The length of the first phase is at most k_0 and the second phase takes at most Δ steps. The following claim is immediate from our previous discussion. (It is here that we use (38)).

Claim 10 *Starting from a vertex $v \notin S$ the expected number of rounds until the random walk is in S is at most $O(\log^{\gamma_1} n)$.*

Proof Indeed the probability that a random walk of length k_0 passes through S is at least $\frac{\rho_{k_0} - |R \setminus S|}{(d-1)^{k_0}} = \Omega(\log^{-\gamma_1} n)$. $\square$

By Claim 10 we have a $\Omega(\log^{-\gamma_1} n)$ chance of reaching S at the end of the first phase. When we start the second phase we have at least a $\left(\frac{1}{d-1}\right)^{\Delta}$ probability of reaching a free vertex, thus ending the random walk. Then the number of rounds until we reach a free vertex is dominated by a geometric distribution with parameter $\Omega\left(\left(\frac{1}{d-1}\right)^{\Delta} \log^{-\gamma_1} n\right)$ and thus the expected number of rounds is $O((d-1)^{\Delta} \log^{\gamma_1} n)$. Since both Lemma 6 and Claim 10 apply regardless of the starting vertex, this shows that the expected number of steps until we reach a free vertex is at most

$$\begin{aligned} O\left(k_0 \log^{\gamma_1} n (d-1)^{\Delta}\right) &= O\left((\log n)\,(\log^{\gamma_1} n)(d-1)^{(\gamma_0+\gamma_1)\log_{d-1}\log n + O(1)}\right) \\ &= O\left(\log^{1+\gamma_0+2\gamma_1} n\right). \end{aligned}$$

There is still the matter of Assumption (25). This is easily dealt with. If we find $v \in R_k \cap F$ then we are of course delighted. So, we could just add a dummy tree extending $2(k_0 - k)$ levels from v where each vertex in the last level is in F. The conclusion of Claim 10 will remain unchanged. This completes the proof of Theorem 1.

7 Conclusion

We have demonstrated that for sufficiently large d with high probability the graph structure of the resulting cuckoo graph is such that, regardless of the starting vertex, the random-walk insertion method will reach a free vertex in polylogarithmic time with high probability. Obvious directions for improvement include reducing the value of d for which this type of result holds, and reducing the exponent in the time bound. The ultimate goal here would be to prove the random walk approach or a similarly simple approach could achieve logarithmic insertion time with high probability and constant expected insertion time. See [7] for subsequent results that move forward in these directions. A further open direction would be to expand the result to cuckoo hashing variants where more than one item can be stored in a location.

Ideally, we would hope to prove a logarithmic time bound that holds with high probability for the random-walk insertion algorithm. Such a result would require a greater understanding of the cuckoo graph that would allow one to improve over our two-phase analysis.

Acknowledgement: We thank the referees for their help in improving the presentation in this paper.

References

[1] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced Allocations. *SIAM Journal on Computing*, 29(1):180-200, 1999.

[2] A. Broder and A. Karlin. Multilevel Adaptive Hashing. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms* (SODA), pp. 43-53, 1990.

[3] A. Broder and M. Mitzenmacher. Using Multiple Hash Functions to Improve IP Lookups. *Proceedings of the 20th IEEE International Conference on Computer Communications* (INFOCOM), pp. 1454-1463, 2001.

[4] L. Devroye and P. Morin. Cuckoo Hashing: Further Analysis. *Information Processing Letters*, 86(4):215-219, 2003.

[5] M. Dietzfelbinger and C. Weidling. Balanced Allocation and Dictionaries with Tightly Packed Constant Size Bins. *Theoretical Computer Science*, 380(1-2):47-68, 2007.

[6] D. Fotakis, R. Pagh, P. Sanders, and P. Spirakis. Space Efficient Hash Tables With Worst Case Constant Access Time. *Theory of Computing Systems*, 38(2):229-248, 2005.

[7] N. Fountoulakis, K. Panagioutou and A. Steger, On the Insertion Time of Cuckoo Hashing, arXiv:1006.1231.

[8] A. Kirsch and M. Mitzenmacher. Using a Queue to De-amortize Cuckoo Hashing in Hardware. In *Proceedings of the Forty-Fifth Annual Allerton Conference on Communication, Control, and Computing*:751-758, 2007.

[9] A. Kirsch, M. Mitzenmacher, and U. Wieder. More Robust Hashing: Cuckoo Hashing with a Stash. In *Proceedings of the 16th Annual European Symposium on Algorithms*, pp. 611-622, 2008.

[10] A. Kirsch and M. Mitzenmacher. The Power of One Move: Hashing Schemes for Hardware. In *Proceedings of the 27th IEEE International Conference on Computer Communications* (INFOCOM), pp. 565-573, 2008.

[11] R. Kutzelnigg. Bipartite Random Graphs and Cuckoo Hashing. In *Proceedings of the Fourth Colloquium on Mathematics and Computer Science*:403-406, 2006.

[12] M. Mitzenmacher and S. Vadhan. Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), pp. 746-755, 2008.

[13] R. Pagh and F. Rodler. Cuckoo Hashing. *Journal of Algorithms*, 51(2):122-144, 2004.

[14] B. Vöcking. How Asymmetry Helps Load Balancing. *Journal of the ACM*, 50(4):568-589, 2003.