# Generalized cuckoo hashing with a stash, revisited

Brice Minaud, Charalampos Papamanthou

# Generalized Cuckoo Hashing with a Stash, Revisited

Brice Minaud[1] and Charalampos Papamanthou[2]

[1]Ecole Normale Supérieure, PSL University, CNRS, Inria, France
[2]Yale University, USA

**Abstract**

Cuckoo hashing is a common hashing technique, guaranteeing constant-time lookups in the worst case. Adding a stash was proposed by Kirsch, Mitzenmacher, and Wieder at SICOMP 2010, as a way to reduce the probability of failure (i.e., the probability that a valid Cuckoo assignment fails to exist). It has since become a standard technique in areas such as cryptography, where a negligible probability of failure is often required. We focus on an extension of Cuckoo hashing that allows multiple items per bucket, which improves the load factor. That extension was also analyzed by Kirsch *et al.* in the presence of a stash. In particular, letting $d$ be the number of items per bucket, and $s$ be the stash size, Kirsch *et al.* showed that, for constant $d$ and $s$, the failure probability is $\mathcal{O}(n^{(s+1)(1-d)})$. In this paper, we first report a bug in the analysis by Kirsch *et al.* by showing a counter-example leading to an asymptotically-larger probability of failure ($\Omega(n^{-d-s-1})$). Then we provide a general analysis and upper bound of the failure probability for (almost) arbitrary $d$ and $s$, instead of just constant, which is useful for applications in cryptography. We finally deduce from the general analysis a tight bound $\Theta(n^{-d-s})$ for the probability of failure, for constants $d$ and $s$.

## 1 Introduction

Cuckoo hashing was introduced by Pagh and Rodler [PR04], and proceeds as follows. We wish to allocate a set $S$ of $n$ items into two tables $T_1, T_2$ of size $m = (1+\varepsilon)n$ each, where $\varepsilon > 0$ is an arbitrary constant. The construction is parametrized by two uniformly random hash functions $h_1 : S \to T_1$ and $h_2 : S \to T_2$. Each item $x \in S$ may be allocated either to $h_1(x)$, or to $h_2(x)$. Pagh and Rodler prove that a valid Cuckoo assignment (where no two items are assigned to the same location) exists with probability $1 - \mathcal{O}(n^{-1})$ over the randomness of the hash functions. Cuckoo hashing guarantees constant-time lookups: Any item $x$ can be retrieved by visiting two memory locations. This makes Cuckoo hashing attractive in real-time systems, where worst-case performance is critical as well as in cryptography, where it serves as a core component for certain oblivious algorithms [CGLS17]. Two variations of the above simple Cuckoo hashing construction have been considered in the literature, which are relevant to our paper.

**Large buckets for improved load factor** Dietzfelbinger and Weidling [DW07] consider a simple tweak of Cuckoo hashing, where up to $d > 1$ items can be assigned to the same location in the table. In that setting, there is only a single table $T$ consisting of $m = (1 + \varepsilon)n/d$ buckets, each of size $d$. With this tweak, the load factor (ratio of occupied space in the table after allocating all items) improves to $1/(1 + \varepsilon)$ and therefore can be made arbitrarily close to the optimum 1 (In the basic construction the load factor is bounded by 1/2.)

**Adding a stash to handle failures** Kirsch, Mitzenmacher, and Wieder [KMW10] study Cuckoo hashing in the presence of a *stash*. The stash is used to store items that the Cuckoo insertion algorithm would otherwise fail to assign (and thus trigger a rehash). They show that when allowing for a stash of constant size $s$, the probability of failure of Cuckoo hashing becomes $\mathcal{O}(n^{-s})$. If $s$ is allowed to vary, a similar bound $\mathcal{O}(n^{-s/2})$ was proven in [ADW14], assuming $s = \mathcal{O}(n^c)$ for sufficiently small constant $c$. In either case, the failure probability decreases exponentially with $s$. Note that having a stash is crucial for both real-time and cryptographic applications, where rehashing can be undesirable. In real-time systems, the rehash approach can be unsatisfactory, since it offers poor worst-case performance guarantees. In some cryptographic applications, rehashes are forbidden entirely,

because security proofs rely on the fact that the hash functions are independent of the data being hashed, and rehashing breaks that property [HFNO21][1].

**Cuckoo graph** For the formal treatment of our results, we use the well-established notion of a *Cuckoo graph*: A Cuckoo hashing instance of a set $S$ of $n$ items stored in a table $T$ of $n(1+\varepsilon)/d$ buckets of size $d$ each in the presence of a stash of size $s$ is associated with the *Cuckoo graph* $G(n, \varepsilon, d, s)$, with $n(1+\varepsilon)/d$ vertices and $n$ edges

$$E = \{(h_1(x), h_2(x)) : x \in S\}.$$

This graph is undirected, may contain multiple copies of the same edge ($E$ is a multiset), and may contain loops. Computing a valid Cuckoo hashing can be translated into operations on the Cuckoo graph, with the correspondence of Table 1. If the stash is allowed to be of size up to $s$, the existence of a valid Cuckoo assignment for hash functions $h_1$ and $h_2$ can be translated into purely graph-theoretic terms (Specifically, it translates to a graph orientability problem with deletions [HKL$^+$18].) We say that the Cuckoo graph is *suitable* if and only if it is possible to remove at most $s$ edges, and orient the remaining edges, such that every vertex has outdegree at most $d$. We will refer to the *failure probability* as the probability that $G(n, \varepsilon, d, s)$ is not suitable, over the choice of the hash functions $h_1$ and $h_2$.

**Our starting point** In this work, we are interested in studying the failure probability of Cuckoo hashing in the presence of a stash and when buckets of size $d > 1$ are used. Kirsch, Mitzenmacher, and Wieder [KMW10] conclude their seminal paper by claiming the following (We paraphrase here to adjust to our notation.)

**Proposition 1** ([KMW10] Proposition 4.3)**.** *For any constants $\varepsilon > 0$, $d \geq 1 + \ln(1/\varepsilon)/(1 - \ln 2)$, and $s \geq 0$, the probability that $G(n, \varepsilon, d, s)$ is not suitable is $\mathcal{O}(n^{(s+1)(1-d)})$.*

Our starting point in this work is showing Proposition 4.3 from [KMW10] cannot hold, by considering a "bad event" (i.e., one that causes $G(n, \varepsilon, d, s)$ not to be suitable) that has probability $\Omega(n^{-d-s-1})$. Define $\mathcal{E}_v$ to be the event that there exists a subset of items $X \subseteq S$ of cardinality $d+s+1$, such that $\forall x \in X, h_1(x) = h_2(x) = v$. Clearly, if $\mathcal{E}_v$ holds, there can be no valid Cuckoo assignment, since the bucket $v$ can hold at most $d$ items, and at most $s$ can be relocated to the stash, and therefore $G(n, \varepsilon, d, s)$ is not suitable. So it is enough to compute a lower bound on $\mathcal{E}_v$, which will also be a lower bound for the failure probability. We give our result directly here. (A tighter bound will be proved in Section 3.)

**Lemma 1** (Lower bound of failure probability)**.** *Assume $\varepsilon > 0$, $d \geq 1$ and $s \geq 0$ are constant. The probability that $G(n, \varepsilon, d, s)$ is not suitable is $\Omega(n^{-d-s-1})$.*

*Proof.* Since the hash functions are uniform, the probability that a given $x \in S$ satisfies $h_1(x) = h_2(x) = v$ is $m^{-2}$. $\mathcal{E}_v$ may be viewed as the event that there are at least $d+s+1$ successes in a binomial experiment with $n$

[1]The issue is subtle: If rehashes are allowed, the hash functions are no longer uniformly random; instead, they are uniformly random conditioned on the event that a rehash is not necessary, and that event depends on the data being hashed.

Table 1: From Cuckoo hashing to Cuckoo graph.

| Cuckoo hashing | Cuckoo graph |
|---|---|
| Assigning item $x$ to cell $h_1(x)$ | Orienting edge $(h_1(x), h_2(x))$ towards $h_2(x)$ |
| Assigning item $x$ to cell $h_2(x)$ | Orienting edge $(h_1(x), h_2(x))$ towards $h_1(x)$ |
| Assigning item $x$ to the stash | Removing edge $(h_1(x), h_2(x))$ |
| Number of items stored in cell $c$ | Outdegree of vertex $c$ |

trials, each with probability of success $m^{-2} = 1/c^2n^2$ (where $c = (1+\epsilon)/d$ is a constant). Therefore

$$\begin{aligned}
\Pr(\mathcal{E}_v) &= \sum_{k \geq d+s+1} \binom{n}{k} m^{-2k} (1-m^{-2})^{n-k} \\
&\geq (1-m^{-2})^n \sum_{k \geq d+s+1} \left(\frac{n}{k}\right)^k m^{-2k} \\
&= (1-1/(c^2n^2))^n \sum_{k \geq d+s+1} \left(\frac{n}{kc^2n^2}\right)^k \\
&\geq (1-1/(c^2n^2))^n \cdot \left(\frac{1}{(d+s+1)c^2n}\right)^{d+s+1} \\
&= \Omega(n^{-d-s-1}) \,,
\end{aligned}$$

since $(1-1/(c^2n^2))^n \to_{n\to\infty} 1$. □

**Origin of the flaw.** Lemma 1 implies that Proposition 4.3 from [KMW10] is flawed. Looking at the proof of Proposition 4.3 in [KMW10], the issue stems from the expression $F \leq \sum_{s+1 \leq j \leq m/(1+\varepsilon)} F(j)$, near the beginning of the proof. The sum should start from $j = 1$, not $j = s+1$. This has a large impact on the final result, because the $F(j)$'s are (roughly) exponentially decreasing; in particular, $F(1)$ dominates the sum.

The argument in [KMW10] is, roughly speaking, that each term $F(j)$ can be upper-bounded by $O(n^{j(1-d)})$. Because the authors assume $j \geq s+1$, they get a bound of the form $O(n^{(s+1)(1-d)})$ for the overall sum. Since in reality, the sum must start at 1, this line of reasoning can only yield a bound $O(n^{1-d})$, which does not depend on the stash size $s$, defeating the point of the analysis.

For that reason, we set out in this article to repair Proposition 4.3, using a different proof technique. Our analysis follows the same structure as the one in [DW07], which previously studied the same problem without a stash. In the end, the new analysis mainly comes down to a finer upper bound for the terms $F(j)$'s, that properly accounts for the presence of a stash.

**Our main results.** Our main results are organized as follows.

- In Section 2, we repair Proposition 4.3 from [KMW10], as explained above. In particular we compute an upper bound on the probability of $G(n, \varepsilon, d, s)$ not being suitable for *(almost) arbitrary values* of $d$ and $s$, instead of just constants as in [KMW10]. The only requirements we have is that $d \geq 1 + \ln(1/\varepsilon)/(1-\ln 2)$ (this condition is tight by [KMW10, Proposition 4.4]) and $1 \leq s \leq m/(10e^4)$. We show the failure probability is
$$\mathcal{O}\left(\left(\frac{e}{m}\right)^{d+s} + \left(\frac{2e}{m}\right)^{2(d-1)} \left(\frac{\alpha s}{m}\right)^{s+1} + m\gamma^m\right),$$
for some constants $\alpha > 0$ and $\gamma < 1$. See Theorem 1. Note that the above is negligible in $n$ when $s$ and $d$ are, for example, $\log n$, which is important for applications in cryptography (e.g., [BBF+21]).
- In Section 3, we focus on the case that $d$ and $s$ are constants, and give a tight bound $\Theta(n^{-d-s})$ for the probability that $G(n, \varepsilon, d, s)$ is not suitable. We leave as open problem to derive a tight bound for arbitrary $d$ and $s$.
- Finally, in Section 4, we examine how our updated result affects the guarantees of the algorithm that actually does the final assignment (and which was presented in [KMW10]), showing that the assignment algorithm from [KMW10] will output a stash whose size exceeds $s$ with probability $\mathcal{O}(n^{-d-s})$—see Corollary 6.

Throughout this paper, $\varepsilon$ is viewed as a constant, as is standard in the analysis of Cuckoo hashing schemes. In other words, the failure probability is regarded as a function of $n$, $d$, $s$, for fixed $\varepsilon > 0$. Concretely, this means that the hidden constants in the $\mathcal{O}()$, $\Omega()$, $\Theta()$ notation may depend on $\varepsilon$. In [KMW10], $d$ and $s$ are also regarded as constants, with the same implication. Parts of our analysis (e.g., in Section 3) will also view $d$ and $s$ as constant. Which quantities are constant will be explicit in theorem statements.

# 2 Failure probability upper bound for general $d$ and $s$

In this section, we repair Proposition 4.3 from [KMW10] and prove a generalized result for the failure probability. We stress that unlike Proposition 4.3 from [KMW10], we do not require $d$ and $s$ to be constants.

In what follows, it will be convenient to use both cuckoo hashing terminology, and cuckoo graph terminology, via the correspondance discussed in the introduction, depending on the situation. In particular, we will sometimes identify an item $x \in S$ with the corresponding edge $(h_1(x), h_2(x)) \in E$. (This is a one-to-one mapping: Even if two items $x \neq y \in S$ are such that $h_1(x) = h_1(y)$ and $h_2(x) = h_2(y)$, recall that $E$ is a multiset, and each item gives rise to a distinct edge.)

Our proof follows the same approach as [DW07, Proof of Theorem 1], but has to account for the addition of a stash. We first introduce some necessary notation: For $X \subseteq E$ being a subset of edges, let $\Gamma(X)$ be the set of endpoints of edges in $X$ (E.g., if $X = \{(1,2),(2,3),(1,3),(1,1)\}$ then $\Gamma(X) = \{1,2,3\}$.) Without a stash (*i.e.,* $s = 0$), [DW07] observes that $G(n, \varepsilon, d, s)$ is suitable iff $\forall X \subseteq S, |\Gamma(X)| \geq |X|/d$. If the condition fails, *i.e.,* $\exists X, |\Gamma(X)| < |X|/d$, suitability fails "trivially" because there is not enough room to store the $|X|$ edges in $\Gamma(X)$. So the previous equivalence may be understood as saying: suitability holds iff it does not fail trivially on any subset of edges. This is still the case when adding a stash. We present the following claim.

**Lemma 2.** *$G(n, \varepsilon, d, s)$ is suitable iff for all $X \subseteq S$, $d|\Gamma(X)| + s \geq |X|$.*

*Proof.* If the condition $d|\Gamma(X)| + s \geq |X|$ fails for some subset $X$ of edges, then there is not enough room in $\Gamma(X)$ and the stash to store the edges in $X$. Indeed, each vertex in $|\Gamma(X)|$ can hold at most $d$ items, and the stash can hold at most $s$ items. This shows that if $G$ is suitable, then the condition must hold for all $X$. Conversely, assume the graph is not suitable. We are going to build an $X$ such that the condition fails.

Define the *overflow* $\mathsf{ov}(D)$ of a directed graph $D$ as the minimal number of edges to remove such that every vertex has outdegree at most $d$. Let $s'$ be the smallest integer such that the following statement holds: there exists a directed graph $D$ arising from orienting the edges of $G$ such that $\mathsf{ov}(D) = s'$ (in other words, $s'$ is the minimum stash size, across all possible orientations of $G$). The fact that $G$ is not suitable translates to $s' > s$. Fix $D$ witnessing the previous statement. Consider the subset $Y$ of vertices of $D$ that have outdegree strictly more than $d$. Let $Y' \supseteq Y$ be the set of vertices that can be reached from $Y$ by following a directed path. Observe that the outdegree of every vertex $v$ in $Y'$ must be at least $d$. Otherwise, there would exist a directed path from a vertex $v$ in $Y$ to some $w$ in $Y'$ with outdegree $< d$. Flipping the direction of every edge along this path decreases the outdegree of $v$ by 1, increases the outdegree of $w$ by 1, and does not change the outdegree of intermediate vertices. Because $v$ was over capacity (outdegree $> d$) and $w$ was under capacity (outdegree $< d$), this means that after flipping the edges of the path, the overflow has decreased by 1. This would contradict the minimality of $s'$. Let $D' = (Y', X)$ be the subgraph of $D$ induced by $Y'$. Here, $X$ is the set of edges of $D$ whose endpoints are both in $Y'$. Note that $Y' = \Gamma(X)$. By construction, we have $s' = \mathsf{ov}(D) = \mathsf{ov}(D') = \sum_{v \in Y'} (\mathsf{out}_{D'}(v) - d) = |X| - d|Y'| = |X| - d|\Gamma(X)|$. Since $s' > s$, $X$ witnesses $d|\Gamma(X)| + s < |X|$. □

Lemma 2 will be useful in proving the main theorem, presented next.

**Theorem 1.** *Fix a constant $0 < \varepsilon \leq 0.25$. There exist constants $\alpha > 0$ and $\gamma < 1$, such that for every $d \geq 1 + \ln(1/\varepsilon)/(1 - \ln 2)$, and $1 \leq s \leq m/(10e^4)$, the probability that $G(n, \varepsilon, d, s)$ is not suitable is*

$$\mathcal{O}\left(\left(\frac{e}{m}\right)^{d+s} + \left(\frac{2e}{m}\right)^{2(d-1)} \left(\frac{\alpha s}{m}\right)^{s+1} + m\gamma^m\right).$$

*If $d$ and $s$ are constants, that expression is $\mathcal{O}(n^{-d-s})$.*

*Proof.* By Lemma 2, the probability that $G$ is not suitable is equal to

$$F := \Pr(\exists X \subseteq S : d|\Gamma(X)| + s < |X|)\,.$$

Now, let $F(j)$ be the probability that there exists a set $Y$ of $j$ vertices and a set $X$ of $dj + s + 1$ edges that satisfy $\Gamma(X) \subseteq Y$. Note that $F(j)$ must be zero when $dj + s + 1 > n$, since $X$ must be a subset of $S$ with $|S| = n$. Therefore, letting $J = (n - s - 1)/d$, by the union bound we have

$$F \leq \sum_{1 \leq j \leq J} F(j)\,, \tag{1}$$

Note also that

$$F(j) \leq \binom{m}{j} \Pr[I_j \geq dj + s + 1]\,, \tag{2}$$

where $I_j$ is the number of edges from $S$ whose endpoints both fall into a fixed set $Y$ of size $j$. Since $I_j$ follows a binomial distribution with $\mathbb{E}[I_j] = n(j/m)^2$, we can use the Chernoff–Hoeffding bound, as in [DW07], to bound $\Pr[I_j \geq jd + s + 1]$, and therefore for $j < J$, Eq. (13) from [DW07], in the presence of the stash, becomes

$$F(j) \leq \binom{m}{j} \left(\frac{nj}{m^2(d + s'/j)}\right)^{jd+s'} \left(\frac{n(m^2 - j^2)}{m^2(n - jd - s')}\right)^{n-jd-s'}, \tag{13'}$$

where we set for convienience $s' = s + 1$ (Some equation indices follow [DW07] when possible for ease of comparison, adding a prime, so (13') is the counterpart of (13) in [DW07].) Continuing from there and replacing $n$ by $dm/(1 + \varepsilon)$ we have

$$\begin{aligned}
F(j) &\leq \frac{m^m}{j^j(m-j)^{m-j}} \left(\frac{d}{d + s'/j} \cdot \frac{j}{(1+\varepsilon)m}\right)^{jd+s'} \left(\frac{d(m^2 - j^2)}{(1+\varepsilon)m(n - jd - s')}\right)^{dm/(1+\varepsilon)-jd-s'} \\
&= \frac{m^m}{j^j(m-j)^{m-j}} \left(\frac{d}{d + s'/j} \cdot \frac{j}{(1+\varepsilon)m}\right)^{jd+s'} \left(\frac{m^2 - j^2}{(1+\varepsilon)m(m/(1+\varepsilon) - j - s'/d)}\right)^{dm/(1+\varepsilon)-jd-s'} \\
&< (1+\varepsilon)^{-jd-s'} m^{m\frac{1+\varepsilon-d}{1+\varepsilon}} j^{j(d-1)+s'} (m-j)^{j-m} \left(\frac{m^2 - j^2}{m - (1+\varepsilon)(j + s'/d)}\right)^{d\frac{m-(j+s'/d)(1+\varepsilon)}{1+\varepsilon}} \\
&= f(j, \varepsilon)\,.
\end{aligned} \tag{14'}$$

We split the sum from Equation 1 into three cases, $j = 1$, $1 < j < 5s$, and $5s \leq j \leq J$.

**Case 1:** $j = 1$. By (13') we get:

$$\begin{aligned}
F(1) &\leq m \left(\frac{d}{d + s'} \cdot \frac{1}{(1+\varepsilon)m}\right)^{d+s'} \left(\frac{n - n/m^2}{n - d - s'}\right)^{n-d-s'} \\
&< m \left(\frac{1}{(1+\varepsilon)m}\right)^{d+s'} e^{d+s'-n/m^2} \\
&= \mathcal{O}\left(\left(\frac{e}{m}\right)^{d+s}\right).
\end{aligned} \tag{15'}$$

Note that if $s$ and $d$ are constant, this is $\mathcal{O}(n^{-d-s})$.

**Case 2:** $1 < j \leq 5s$. Recall that the upper-bound $F(j) \leq f(j, \varepsilon)$ in Equation (14') was computed as a union bound as follows

$$F(j) \leq \binom{m}{j} \Pr[I_j \geq dj + s + 1] \leq f(j, \varepsilon)\,.$$

It is clear that $\Pr[I_j \geq dj + s + 1]$, a function of $\varepsilon$, is decreasing while $\varepsilon$ is increasing because $\Pr[I_j \geq dj + s + 1]$ is the upper tail of a binomial experiment, and $\varepsilon$ only decreases the probability of success of each trial in the experiment, while it does not affect any other parameter. It follows that $F(j) \leq f(j, 0)$. Using (14') we get

$$\begin{aligned}
F(j) < f(j,0) &= m^{m(1-d)} j^{j(d-1)+s'} (m-j)^{j-m} \left(\frac{m^2 - j^2}{m - j - s'/d}\right)^{d(m-j-s'/d)} \\
&= m^{m(1-d)} j^{j(d-1)+s'} m^{j-m} (1 - j/m)^{j-m} m^{d(m-j-s'/d)} \cdot \left(1 + \frac{j}{m} + \frac{s'}{dm} \cdot \frac{1 + j/m}{1 - j/m - s'/(dm)}\right)^{d(m-j-s'/d)} \\
&< m^{j(1-d)-s'} j^{j(d-1)+s'} e^{j(m-j)/m} e^{d(m-j-s'/d)\left(j/m + \frac{s'}{dm} \cdot \frac{1+j/m}{1-j/m-s'/(dm)}\right)} \\
&= \left(\frac{j}{m}\right)^{(d-1)j+s'} e^{j(m-j)/m} e^{d(m-j-s'/d)\left(j/m + \frac{s'}{dm} \cdot \frac{1+j/m}{1-j/m-s'/(dm)}\right)} \\
&= \left(\frac{j}{m}\right)^{(d-1)j+s'} e^{j(m-j)/m} e^{dj(m-j-s'/d)/m} e^{d(m-j-s'/d) \cdot \left(\frac{s'}{dm} \cdot \frac{1+j/m}{1-j/m-s'/(dm)}\right)} \\
&< \left(\frac{j}{m}\right)^{(d-1)j+s'} e^{(d+1)j} e^{s'\left(\frac{1+j/m}{1-j/m-s'/(dm)}\right)}.
\end{aligned}$$

Because $j < 5s$, $s \leq m/(10e^4)$ and $d$ is bounded below by a constant, $e^{(1+j/m)/(1-j/m-s'/(dm))}$ is upper-bounded by an absolute constant $C$. Reinjecting $C$ we have:

$$\begin{aligned} F(j) &< \left(\frac{j}{m}\right)^{(d-1)j+s'} e^{(d+1)j} C^{s'} \\ &= \left(\left(\frac{j}{m}\right)^{d-1} e^{d+1}\right)^{j} \left(\frac{Cj}{m}\right)^{s'}. \end{aligned} \tag{17'}$$

In the end, the expression is the same as in [DW07], with an additional term $(Cj/m)^{s'}$. Denote the rest of the expression by $g(j)$, so that (17') becomes $F(j) < g(j)(Cj/m)^{s'}$. It was already observed in [DW07] that in the relevant range $1 < j \leq 5s \leq m/(2e^4)$, $g(j)$ is geometrically decreasing. More precisely, the ratio of one term to the previous term is upper-bounded by a constant strictly less than 1. It follows that $g(2)$ dominates $\sum_{2\leq j<5s} g(j)$. This yields

$$\sum_{2\leq j<5s} F(j) = \mathcal{O}\left(g(2)\left(\frac{C\cdot 5s}{m}\right)^{s'}\right) = \mathcal{O}\left(\left(\frac{2e}{m}\right)^{2(d-1)}\left(\frac{\alpha s}{m}\right)^{s+1}\right),$$

for some constant $\alpha > 0$. Note that if $s$ and $d$ are constant, this is $\mathcal{O}(n^{1-2d-s}) = \mathcal{O}(n^{-d-s})$.

**Case 3:** $5s < j \leq J = (n-s-1)/d$. The case $5s < j \leq m/(2e^4)$ is handled in Case 2. The case $j > m/(2e^4)$ was already shown to be $\mathcal{O}(m\gamma^m)$ for some $\gamma < 1$ in [DW07]. Note that if $s$ and $d$ are constant, this is $\mathcal{O}(n^{-d-s})$. Therefore overall, if $s$ and $d$ are constant, the probability of failure is $\mathcal{O}(n^{-d-s})$. This concludes the proof. □

# 3 Failure probability tight bound for constant $d$ and $s$

In this section we prove a tight bound for the failure probability, for constant $d$ and $s$.

**Theorem 2.** *Fix a constant $0 < \varepsilon \leq 0.25$. For every constants $d \geq 1 + \ln(1/\varepsilon)/(1-\ln 2)$ and $s \geq 1$, the probability that $G(n, \varepsilon, d, s)$ is not suitable is $\Theta(n^{-d-s})$.*

*Proof.* By Theorem 1, we know that for constants $d$ and $s$, the probability of failure is $\mathcal{O}(n^{-d-s})$. It is enough to show that the failure probability is $\Omega(n^{-d-s})$. To do that, we will first show that $\Pr(\mathcal{E}_v) = \mathcal{O}(n^{-d-s-1})$, where $\mathcal{E}_v$ is the event defined in the introduction. Indeed,

$$\begin{aligned} \Pr(\mathcal{E}_v) &= \sum_{k\geq d+s+1} \binom{n}{k} m^{-2k}(1-m^{-2})^{n-k} \\ &\leq \sum_{k\geq d+s+1} \frac{n^k}{k!} m^{-2k} \\ &= \sum_{k\geq d+s+1} \frac{n^{-k}}{k!}\left(\frac{1+\varepsilon}{d}\right)^{-2k} \\ &\leq n^{-d-s-1} \sum_{k\geq d+s+1} \frac{1}{k!}\left(\frac{1+\varepsilon}{d}\right)^{-2k} \\ &\leq n^{-d-s-1} e^{\left(\frac{1+\varepsilon}{d}\right)^{-2}} \\ &= \mathcal{O}(n^{-d-s-1}). \end{aligned}$$

Therefore, by Lemma 1, we have $\Pr(\mathcal{E}_v) = \Theta(n^{-d-s-1})$. We continue by showing that the failure probability is bounded from below by $m\Pr(\mathcal{E}_v)/2$, thus completing the proof.

Indeed, consider the binary random variable $I_v$ that is equal to 0 iff $\mathcal{E}_v$ occurs. Likewise, let $I$ denote the binary random variable that is equal to 0 iff $\mathcal{E} = \bigvee_v \mathcal{E}_v$ occurs. Observe $I = \prod_v I_v$.

**Claim 1.** *Variables $I_v$ are negatively associated.*

*Proof.* For $v \in V$, let $B_v = |\{x \in S : h_1(x) = h_2(x) = v\}|$ denote the number of edges that loop in on $v$ in the Cuckoo graph. Let $B' = |\{x \in S : h_1(x) \neq h_2(x)\}|$ denote the number of edges that are not loops, so that $B' + \sum_v B_v = n$. By [DR96, Theorem 13], the $B_v$'s together with $B'$ are negatively associated. A fortiori the $B_v$'s are negatively associated. We have $I_v = f(B_v)$ where $f$ is the non-increasing function that is equal to 1 if its input is strictly less than $d + s + 1$, and 0 otherwise. Hence by [DR96, Proposition 7.2], random variables $I_v$ are negatively associated. ■

By [DR96, Proposition 3] and Claim 1, we have $\mathbb{E}(\prod_v I_v) \leq \prod_v \mathbb{E}(I_v)$. We therefore get

$$\begin{aligned}
\Pr(\mathcal{E}) &= 1 - \mathbb{E}(I) \\
&= 1 - \mathbb{E}\Big(\prod_v I_v\Big) \\
&\geq 1 - \prod_v \mathbb{E}(I_v) \\
&= 1 - (1 - \Pr(\mathcal{E}_v))^m && \text{for any fixed } v \\
&\geq 1 - e^{-m \Pr(\mathcal{E}_v)} && \text{using } 1 - x \leq e^{-x}.
\end{aligned}$$

We have already established that $m \Pr(\mathcal{E}_v) = \Theta(n^{-d-s})$ (since $\Pr(\mathcal{E}_v) = \Theta(n^{-d-s-1})$), so we can freely assume $m \Pr(\mathcal{E}_v) < 1$ for $n \geq n_0$ where $n_0$ is a fixed constant. Using the fact that $1 - e^{-x} > x/2$ for $x$ in $[0, 1]$, we get

$$\Pr(\mathcal{E}) \geq m \Pr(\mathcal{E}_v)/2 = \Omega(n^{-d-s}),$$

which concludes the proof, since the the failure probability is bounded from below by $\Pr(\mathcal{E})$. □

# 4 Algorithmic results

In the previous sections, we have identified a mistake in [KMW10, Proposition 4.3], and repaired the proposition by proving a new result, with a corrected bound (Theorem 1). [KMW10, Proposition 4.3] is later used to prove [KMW10, Theorem 4.2]. As a result, [KMW10, Theorem 4.2] becomes invalid—both the proof, and the bound in the theorem statement are incorrect. In this section, we repair [KMW10, Theorem 4.2]. No other result in [KMW10] depends on [KMW10, Proposition 4.3], and to the best of our knowldege, the rest of the article is correct.

Theorem 1 is a statement about the *existence* of a solution to a problem. It provides an upper bound on the probability that a certain allocation problem fails to have a solution. A natural question is how to find such a solution algorithmically. Kirsch *et al.* propose an algorithm for that purpose. Using the terminology from Section 1, the algorithm can be formulated as follows.

We start from a graph with vertex set $V = T_1 \cup T_2$, and no edge. Each item $x \in S$ is inserted in turn. To insert item $x$, the directed edge $(h_1(x), h_2(x))$ is added to the graph. The algorithm then looks for a directed path starting from $h_1(x)$, and ending in any vertex with outdegree strictly less than $d$. Such a path is called an *augmenting path*. To find an augmenting path, the algorithm performs a breadth-first search starting from $h_1(x)$. If an augmenting path is found, the direction of every edge along the path is flipped. If the breadth-first search exhausts the vertices reachable from $h_1(x)$ without finding an augmenting path, the item $x$ is added to the stash; in that case, the edge $(h_1(x), h_2(x))$ is removed from the graph. Observe that this insertion algorithm preserves the invariant that every vertex has outdegree at most $d$. As a consequence, the graph $G$ output by the overall algorithm sastisfies $\Delta^+(G) \leq d$. Let us call that algorithm $\mathcal{A}$.

As noted in [KMW10], results about the insertion time of $\mathcal{A}$ can be deduced directly from the analysis in [DW07]. For that reason, Kirsch *et al.* focus their analysis of $\mathcal{A}$ on the distribution of the stash size (*i.e.* the distribution of the number of items that are sent to the stash by $\mathcal{A}$). The following result is claimed.

**Theorem 3** ([KMW10], Theorem 4.2)**.** *Les $s_{\mathcal{A}}$ denote the size of the stash after all $n$ items have been inserted using algorithm $\mathcal{A}$. For small enough $\varepsilon > 0$ and $d \geq 15.8 \cdot \ln(1/\varepsilon)$, and any integer $s \geq 2$, we have $\Pr(s_{\mathcal{A}} \geq s) = O(n^{sd(D-1)})$, where $D = d^{-1/3} + d^{-1}$. For $s = 1$, we have $\Pr(s_{\mathcal{A}} \geq s) = O(n^{1-d})$.*

As noted earlier, the proof of Theorem 3 relies on [KMW10, Proposition 4.3], which we have shown to be false. In fact, the bound $O(n^{sd(D-1)})$ given in the theorem cannot hold, because it is supposed to bound to the probability that the algorithm fails to find a solution; but after correcting [KMW10, Proposition 4.3], it is now asymptotically lower than the probability $\Theta(n^{-d-s})$ that a solution exists in the first place (Theorem 2).

To repair Theorem 4.2, we use a different approach from [KMW10]: We show that algorithm $\mathcal{A}$ is *optimal*, in the sense that it minimizes the number of items sent to the stash. More precisely, given any fixed pair of hash functions $h_1, h_2$ (to avoid cluttering notation, we leave the hash functions as implicit parameters), let $s_{\mathcal{A}}$ denote the size of the stash at the outcome of algorithm $\mathcal{A}$. On the other hand, let $s_{\text{opt}}$ denote the smallest stash size such that a solution exists; formally, using the notation from Section 1:

$$s_{\text{opt}} = \min\{s : G(n, \varepsilon, d, s) \text{ is suitable}\}.$$

**Theorem 4.** *For any any pair of hash functions $(h_1, h_2)$, it holds that $s_{\mathcal{A}} = s_{\text{opt}}$.*

*Proof.* Since $\mathcal{A}$ outputs a valid solution to the cuckoo allocation problem, $s_{\mathcal{A}} \geq s_{\text{opt}}$ holds trivially. We now prove the converse.

First, let us introduce some notation. Given a directed graph $G = (V, E)$, let $G^u = (V, E^u)$ denote the undirected graph obtained by forgetting edge orientations in $G$. Given a graph $G = (V, E)$, and $V' \subseteq V$ a subset of vertices, let $G[V'] = (V', E[V'])$ denote the subgraph of $G$ induced by $V'$, that is: $E[V'] = \{(u, v) \in E : u, v \in V'\}$. Finally, a vertex $v$ in a directed graph $G$ is said to be *full* if there is no augmenting path from $v$, *i.e.* no directed path from $v$ to any vertex $w$ with $\deg^+(w) < d$. (Paths of length zero are allowed, hence $v$ being full implies $\deg^+(v) \geq d$.) The key lemma is the following.

**Lemma 3.** *Let $G = (V, E)$ be a directed graph such that $\Delta^+(G) \leq d$, and let $V' \subseteq V$. The following three properties are equivalent.*

*(1) For all $v \in V'$, $v$ is full in $G$.*

*(2) There exists $V'' \supseteq V'$ such that $|E[V'']| \geq d|V''|$.*

*(3) There exists $V'' \supseteq V'$ such that $|E[V'']| = d|V''|$.*

*Proof.* (2) $\Rightarrow$ (1). Assume (2). We have:

$$d|V''| \leq |E[V'']| = \sum_{v \in V''} \deg^+_{G[V'']}(v) \leq \sum_{v \in V''} \deg^+_G(v) \leq \Delta^+(G)|V''| \leq d|V''|.$$

Hence all inequalities are equalities, which implies $\deg^+_{G[V'']}(v) = \deg^+_G(v) = d$ for all $v \in V''$. Hence, there is no edge from $V''$ to $V \setminus V''$, and all vertices in $V''$ have outdegree $d$. Hence all vertices in $V'$ are full.

(1) $\Rightarrow$ (3). Assume (1). Let $V''$ be the set of vertices reachable from $V'$ in $G$. All vertices in $V''$ must have outdegree at least $d$, since the vertices of $V'$ are full. Since $\Delta^+(G) \leq d$, all vertices in $V''$ have outdegree exactly $d$. On the other hand, by construction, there is no edge from $V''$ to $V \setminus V''$. In conclusion:

$$|E[V'']| = \sum_{v \in V''} \deg^+_{G[V'']}(v) = \sum_{v \in V''} \deg^+_G(v) = d|V''|.$$

(3) $\Rightarrow$ (2) is trivial. This concludes the proof. ∎

Intuitively, Lemma 3 says that, assuming $\Delta^+(G) \leq d$, whether $v$ is full in $G$ is entirely determined by the underlying undirected graph $G^u$. Indeed, observe that property (2) in Lemma 3 depends only on $G^u$.[2]

**Corollary 5.** *Let $G_0 = (V, E_0)$, $G_1 = (V, E_1)$ be two directed graphs such that $\Delta^+(G_i) \leq d$, and $E_0^u \subseteq E_1^u$. If a vertex $v$ is full in $G_0$, then it is full in $G_1$.*

*Proof.* This follows directly from the equivalence (1) $\Leftrightarrow$ (2) in Lemma 3. ∎

We are now ready to prove $s_{\mathcal{A}} \leq s_{\text{opt}}$. Let $G = G(n, \varepsilon, d, s)$ be the cuckoo graph defined in Section 1. That is, $G = (V, E)$ with $V = T_1 \cup T_2$, and $E = \{\{h_1(x), h_2(x)\} : x \in S\}$.

Recall that algorithm $\mathcal{A}$ starts from an empty graph $G_0 = (V, \varnothing)$. Assume any fixed insertion order $x_1, \ldots, x_{|S|}$ on the set $S$ of items. Let $G_i = (V, E_i)$ denote the graph obtained after algorithm $\mathcal{A}$ has inserted items $x_1, \ldots, x_i$. Let $s = |S|$, so that $G_s$ is the graph at the output of algorithm $\mathcal{A}$. Let $S_{\mathcal{A}} \subseteq S$ be

[2]The fact that having no augmenting path, ostensibly a property of a directed graph, is entirely determined by the underlying undirected graph, may appear surprising, but it is a common occurence in combinatorial optimization: see [Sch03, Chapter 61], and the discussion in Section 5.

the set of items sent to the stash by $\mathcal{A}$. Note that $G_s^u$ is almost the same as $G$, except it is missing the edges corresponding to items in $S_{\mathcal{A}}$; that is:

$$E = E_s^u \cup \{\{h_1(x), h_2(x)\} : x \in S_{\mathcal{A}}\}. \qquad (*)$$

By construction of $\mathcal{A}$, whenever an item $x_i \in S_{\mathcal{A}}$ is sent to the stash, it must be the case that $h_1(x_i)$ and $h_2(x_i)$ are full in $G_{i-1}$. By Corollary 5, it follows that $h_1(x_i)$ and $h_2(x_i)$ are also full in $G_s$. Letting $V' = \{h_1(x), h_2(x) : x \in S_{\mathcal{A}}\}$, the vertices of $V'$ are full in $G_s$. Using Lemma 3 again, there exists $V'' \supseteq V'$ such that $|E_s[V'']| = d|V''|$. Using $(*)$, this yields :

$$|E[V'']| = |E_s[V'']| + s_{\mathcal{A}} = d|V''| + s_{\mathcal{A}}.$$

On the other hand, all edges in $E[V'']$ correspond to items that must be stored in $V''$, since both of their endpoints are in $V''$. But the vertices of $V''$ can accomodate at most $d|V''|$ items. It follows that at least $s_{\mathcal{A}}$ items cannot be stored, and must be sent to the stash. Hence $s_{\text{opt}} \geq s_{\mathcal{A}}$. □

As a direct corollary, the probability that algorithm $\mathcal{A}$ outputs a solution with a stash of size at most $s$ is equal to the probability that such a solution exists. Hence, the bound from Theorem 1 applies directly to algorithm $A$. For simplicity, we only write the corollary in the case that $s$ and $d$ are constant.

**Corollary 6.** *Les $s_{\mathcal{A}}$ denote the size of the stash after all $n$ items have been inserted using algorithm $\mathcal{A}$. Assume $0 < \varepsilon \leq 0.25$, $d \geq 1 + \ln(1/\varepsilon)/(1 - \ln 2)$, and $1 \leq s \leq m/(10e^4)$ are constant. Then $\Pr[s_{\mathcal{A}} > s] = \mathcal{O}(n^{-d-s})$.*

# 5 Related work

Cuckoo hashing was introduced by Pagh and Rodler [PR04]. A variant where each table location can receive $d > 1$ items was studied by Dietzfelbinger and Weidling [DW07], and a variant with $k > 2$ hash functions was analyzed by Fotakis *et al.* [FPSS05]. Later, Kirsch, Mitzenmacher and Wieder introduced cuckoo hashing with a stash, proving that for all three aforementioned variants of cuckoo hashing, the failure probability decreases exponentially with the stash size [KMW10]. The present work corrects the results of [KMW10] pertaining to the variant with $d > 1$ items per cell. As discussed in Section 2, our main proof owes much to the analysis in [DW07]. In a different direction, Aumüller *et al.* have shown that the results of [KMW10] regarding cuckoo hashing with a stash can be modified to work with explicit hash function families, rather than uniformly random hash functions [ADW14].

Cuckoo hashing relates to balls-and-bins games, and particularly to the so-called two-choice process [ABKU94, RMS01]. In the simplest version of the two-choice process, $n$ balls are thrown into $m$ bins, one after the other. For each ball, two bins are selected uniformly at random. The ball is inserted into whichever bin holds the fewest balls at insertion time. Cuckoo hashing follows a similar premise, with the difference that whenever a new ball is inserted, the choices made for previous balls can be revisited. That is, each ball can be moved at any time between its two possible locations, rather than this being decided only at insertion time. Our main theorem (Theorem 1) can be reinterpreted from that perspective: it provides a bound on the probability that the latter variant of the two-choice process admits a solution where the most loaded bin contains at most $d$ items, if we are allowed to "skip" inserting at most $s$ balls. The result of Dietzfelbinger and Weidling cited earlier is written from that perspective [DW07].

As noted in the introduction, Cuckoo hashing can also be cast as a graph orientability problem. In that light, Lemma 2 and Theorem 4 are variants of a result due to Hakimi [Hak65], which gives a similar equivalent condition for a graph to admit an orientation where the indegree of each vertex $v$ is at least equal to a prescribed value $\ell(v)$. Our own lemmas correspond to the setting where $\ell(v)$ is constant, but we allow for a stash (i.e. at most $s$ edges can be deleted). A result of Frank and Gyárfás gives a similar condition when one wishes to prescribe both a lower and upper bound on the indegree of each vertex [GF78]. This line of results belongs to the family of so-called "min-max" theorems in combinatorial optimization. In particular, they are closely related to linear programming duality. We refer the reader to [Sch03] for an excellent exposition of those relationships.

## Acknowledgments

The authors would like to thank Adam Kirsch, Michael Mitzenmacher, Udi Wieder, as well as Martin Dietzfelbinger, for their helpful comments. This work was supported by the ANR JCJC project SaFED and by the National Science Foundation.

# References


[ABKU94] Yossi Azar, Andrei Z Broder, Anna R Karlin, and Eli Upfal. Balanced allocations. In *Proceedings of the twenty-sixth annual ACM symposium on theory of computing*, pages 593–602, 1994.

[ADW14] Martin Aumüller, Martin Dietzfelbinger, and Philipp Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. *Algorithmica*, 70(3):428–456, 2014.

[BBF+21] Angèle Bossuat, Raphael Bost, Pierre-Alain Fouque, Brice Minaud, and Michael Reichle. SSE and SSD: page-efficient searchable symmetric encryption. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021*, pages 157–184. Springer International Publishing, 2021.

[CGLS17] T.-H. Hubert Chan, Yue Guo, Wei-Kai Lin, and Elaine Shi. Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 660–690. Springer International Publishing, 2017.

[DR96] Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996.

[DW07] Martin Dietzfelbinger and Christoph Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theoretical Computer Science*, 380(1-2):47–68, 2007.

[FPSS05] Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul Spirakis. Space efficient hash tables with worst case constant access time. *Theory of Computing Systems*, 38(2):229–248, 2005.

[GF78] András Gyárfás and András Frank. How to orient the edges of a graph. *Combinatorics*, 18:353–362, 1978.

[Hak65] S. Louis Hakimi. On the degrees of the vertices of a directed graph. *Journal of the Franklin Institute*, 279(4):290–308, 1965.

[HFNO21] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Alibi: A flaw in cuckoo-hashing based hierarchical ORAM schemes and a solution. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 338–369. Springer International Publishing, 2021.

[HKL+18] Tesshu Hanaka, Ioannis Katsikarelis, Michael Lampis, Yota Otachi, and Florian Sikora. Parameterized orientable deletion. In *SWAT*, 2018.

[KMW10] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. *SIAM Journal on Computing*, 39(4):1543–1561, 2010.

[PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[RMS01] Andrea W Richa, M Mitzenmacher, and R Sitaraman. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization*, 9:255–304, 2001.

[Sch03] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.