Cuckoo Hashing with a Stash: Alternative Analysis, Simple Hash Functions

Martin Aumüller, Martin Dietzfelbinger

Technische Universität Ilmenau
Cuckoo Hashing

Maintain a dynamic dictionary for \(n\) keys

- lookups: \(\mathcal{O}(1)\)
- deletions: \(\mathcal{O}(1)\)
- insertions: \(\mathcal{O}(1)\) amortized expected
- space: \(2(1 + \varepsilon)n\) slots

Not so good: Insertion of a key set of size \(n\) fails and rebuilds the whole data structure with probability \(\mathcal{O}(n^{-1})\).

© Per H. Olsen
In some applications, e.g.,

- high-performance routing (packet statistics)
- database indexing

a failure probability of $O(n^{-3})$ could already lead to a failure rate that is too high.

$\Rightarrow$ **Cuckoo hashing not applicable**, although its performance is suitable for such applications.

**Task**
Preserve the performance and lower the failure probability.
Cuckoo Hashing with a Stash

Kirsch, Mitzenmacher and Wieder [KMW09]:

- add a small constant-sized piece of memory, the so-called stash
- move elements that cannot be inserted to this stash

They prove: Using a stash of size $s$ lowers failure probability from

\[ O(n^{-1}) \text{ to } O(n^{-(s+1)}) \).

Proof is technically involved (“Poissonization”, “Markov Chain coupling”). Assumes fully random hash functions.
The cuckoo graph $G(S, h_1, h_2)$:

- an undirected bipartite multigraph $(L, R, E)$ where $L$ and $R$ represent the table cells
- $E = \{(h_1(x), h_2(x)) \mid x \in S\}$
The Cuckoo Graph - Example
**Question:** Will all key insertions be successful?

**Lemma (Devroye, Morin [DM03])**

The hash functions $h_1$ and $h_2$ successfully insert all keys in $S$ if and only if each connected component of $G(S, h_1, h_2)$ is either a tree or unicyclic.

**Answer:** No.
How a Stash Helps

- resolves infinite loops by moving a key to the stash
- cuckoo graph contains only trees and unicyclic components if we remove stash keys

Important question

How many items are stored in the stash after a key set $S$ of size $n$ is inserted?

Can we find the answer for this in the cuckoo graph?
The Size of the Stash

Definition
The excess $\text{ex}(G)$ is the minimal number of edges we have to remove from $G$ such that all connected components in $G$ contain at most one cycle.

Proposition (Kirsch et al. [KMW09])
After the insertion of $S$ there are exactly $\text{ex}(G(S, h_1, h_2))$ keys in the stash.
The Size of the Stash – Example

We assume stash of small constant size $s$.

Central Question

How likely is it that more than $s$ keys are moved into the stash?

Equivalent question: $\Pr(\text{ex}(G) > s) = ?$
Part 1: New Proof (based on [DW03])

We know: If stash size $s$ is not sufficient, then $\text{ex}(G(S, h_1, h_2)) > s$.

Idea: Concentrate on subgraph with excess $s + 1$.

**Definition**

An *excess-$(s + 1)$ core structure* of $G = G(S, h_1, h_2)$ is a subgraph $G'$ of $G$ with the following properties:

1. $G'$ has excess exactly $s + 1$.
2. $G'$ has no leaf edges.
3. $G'$ contains only components with at least two cycles.

Pretty obvious: Stash of size $s$ overflows $\iff$ cuckoo graph contains an excess-$(s + 1)$ core structure.
Analysis of Stash Size – Example

Question: Stash size 2 sufficient?
Answer: No, we can find an excess-3 core structure.
Alternative Approach to Analysis
(used in D., Woelfel [DW03])

- count non-isomorphic graphs that form an excess-$(s + 1)$ core structure
- bound probability that one of the excess core structures is realized
Counting Excess Structures

**Theorem (Dietzfelbinger, Woelfel [DW03])**

Number of non-isomorphic connected graphs with \( k - \ell \) inner edges, \( \ell \) leaf edges and cyclomatic number \( q \) is bounded by

\[
k^{\mathcal{O}(\ell + q)}.
\]

**Problem:** Excess might be shared over more than one component.
**Solution:** Insert edges between components to connect the graph!
Counting Excess Structures

**Theorem (Dietzfelbinger, Woelfel [DW03])**

*Number of non-isomorphic connected graphs with* $k - \ell$ *inner edges,* $\ell$ *leaf edges and cyclomatic number* $q$ *is bounded by*

$$k^{\mathcal{O}(\ell + q)}.$$

**Problem:** Excess might be shared over more than one component.

**Solution:** Insert edges between components to connect the graph!
Counting Excess Structures

Theorem (Dietzfelbinger, Woelfel [DW03])

Number of non-isomorphic connected graphs with \( k - \ell \) inner edges, \( \ell \) leaf edges and cyclomatic number \( q \) is bounded by

\[ k^{O(\ell + q)}. \]

Problem: Excess might be shared over more than one component.
Solution: Insert edges between components to connect the graph!
The result

- matches the obtained upper bound in [KMW09]
- uses an intuitive approach to obtain the bound

\[ \Pr(\text{ex}(G) > s) = \mathcal{O}(n^{-(s+1)}) \]
Part 2: “Realistic” Hash Functions

Question

Analysis adaptable using hash functions with a bounded degree of independence, e.g., $d$-wise independent hash functions, which can be efficiently evaluated (like polynomials of degree $d - 1$)?
Class of Hash Functions [DW03]

- $g : U \rightarrow [r]$ from $d$-wise independent class
- $f_1, f_2 : U \rightarrow [m]$ from $d$-wise independent class
- $z_0^{(1)}, \ldots, z_{r-1}^{(1)}$ and $z_0^{(2)}, \ldots, z_{r-1}^{(2)}$ random from $[m]$, tabulated

Hash functions:

$$h_1(x) = \left( f_1(x) + z_{g(x)}^{(1)} \right) \mod m$$

$$h_2(x) = \left( f_2(x) + z_{g(x)}^{(2)} \right) \mod m$$

Evaluation in constant time! Class of these hash functions: $\mathcal{R}_{r,m}^d$. 

18 / 29
Theorem 2

Let $T \subseteq U$. Let $|g(T)| \geq |T| - \ell$ for $(h_1, h_2) \in \hat{R}_{r,m}^{2\ell}$.

Then all $(h_1(x), h_2(x)), x \in T$, are uniformly and independently distributed in $[m]^2$. 
Full Randomness on Excess Core Structures

We need full randomness on excess-$(s + 1)$ core structures to reuse previous analysis.

- Define $G(S, h_1, h_2)$ to be $\ell$-bad if there exists $T \subseteq S$ with $|g(T)| < |T| - \ell$ and $K(T) = G(T, h_1, h_2)$ forms an excess core structure for excess $s + 1$.

- Then: If $G(S, h_1, h_2)$ is “good” then hash function pair works fully randomly on all excess core structures of our interest $\Rightarrow$ can reuse analysis!

- Question: How likely is it that $G(S, h_1, h_2)$ is good?
Bounding Probability of $\ell$-bad Graphs

**Problem:** Pair of hash functions does not work fully randomly on bad graphs, because $|g(T)| < |T| - \ell$.

- works fully randomly on all $T' \subset T : |g(T')| \geq |T'| - \ell$
- extract subgraph $K(T')$ with $|g(T')| = |T'| - \ell$, so-called $2\ell$-reduced subgraph
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.
Goal: $|g(T)| = |T| - \ell$.
Status: $|g(T)| = |T| - \ell - 2$. 
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 2$.

1. Mark all keys in $K(T)$ that collide under $g$. 
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 2$.

1. Mark all keys in $K(T)$ that collide under $g$. 

![Diagram]
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 2$.

2. Remove unmarked components.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$.

2. Remove unmarked components.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$.

3. Remove components while $|g(T)| \leq |T| - \ell$. 
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 2$.

3. Remove components while $|g(T)| \leq |T| - \ell$. 

![Diagram of graphs](image-url)
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

3. Remove components while $|g(T)| \leq |T| - \ell$. 

\[ \text{Diagram showing} \]
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

4. Cannot remove any further components. Concentrate on one component from now on.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell - 1$.

4. Cannot remove any further components. Concentrate on one component from now on.
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

Approach: $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

Goal: $|g(T)| = |T| - \ell$.

Status: $|g(T)| = |T| - \ell - 1$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

5. Remove one marked edge.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

6. Remove edges until all leaf and cycle edges are marked.
Extracting $2\ell$-reduced subgraphs: Peeling

**Approach:** $G$ is $\ell$-bad $\Rightarrow$ graphs contains an excess core structure $K(T)$ with excess $s + 1$ and $|g(T)| < |T| - \ell$.

**Goal:** $|g(T)| = |T| - \ell$.

**Status:** $|g(T)| = |T| - \ell$.

6. Remove edges until all leaf and cycle edges are marked.
Lemma 3

If $G$ is $\ell$-bad, then there exists a subset $T \subseteq S$ such that $|g(T)| = |T| - \ell$ and $K(T)$ has the following properties:

1. There is one connected component in $K(T)$ that has at most $2\ell$ leaf and cycle edges.
2. All other connected components do not have leaves.
3. There are at most $2\ell$ connected components.

To bound probability for $\ell$-bad subgraphs: Can now re-use counting approach and have an extra factor $O(r^{-\ell})$ for the probability for the $g$-collisions to happen.
Cuckoo Hashing with a Stash and HF’s from class $\hat{R}$

**Theorem 3**

\[
\Pr(G(S, h_1, h_2) \text{ is } \ell\text{-bad}) = \mathcal{O}(n \cdot r^{-\ell}).
\]

For \( r = n^\beta, \frac{1}{2} < \beta < 1 \) and \( \ell = 2(s + 2) \).

**Corollary**

\[
\Pr(G(S, h_1, h_2) \text{ is } \ell\text{-bad}) = \mathcal{O}(n^{-(s+1)})
\]
Practical Stash Sizes

Success Rate of Cuckoo Hashing for Fixed Table Load of 49% and Different Table Sizes

Table Size
Success Rate in Percent

Stash 0
Stash 3
Stash 9
Conclusion

- Stash of size $s$ reduces failure probability drastically
  \[
  \mathcal{O}(n^{-1}) \rightarrow \mathcal{O}(n^{-(s+1)})
  \]: New proof.

- Analysis valid for constant-time, $o(n)$-space class $\hat{\mathcal{R}}$.

- A stash size of only 9 helps us to almost completely avoid rehashes in practical scenarios.
Kai-Min Chung and Salil P. Vadhan. 
Tight bounds for hashing block sources. 

Luc Devroye and Pat Morin. 
Cuckoo hashing: Further analysis. 

Martin Dietzfelbinger and Ulf Schellbach. 
On risks of using cuckoo hashing with simple universal hash classes. 
References II


Michael Mitzenmacher and Salil P. Vadhan.
Why simple hash functions work: exploiting the entropy in a data stream.