Load Thresholds for Cuckoo Hashing with Overlapping Blocks

Stefan Walzer
Technische Universität Ilmenau, Germany
stefan.walzer@tu-ilmenau.de
https://orcid.org/0000-0002-6477-0106

Abstract

2012 ACM Subject Classification Theory of computation → Data structures design and analysis, Mathematics of computing → Probabilistic algorithms, Theory of computation → Bloom filters and hashing

Keywords and phrases Cuckoo Hashing, Unaligned Blocks, Hypergraph Orientability, Load Thresholds, Randomised Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.102

Related Version Due to size constraints, many technical details are omitted. These details will be included in a full version and can already be found in the preprint [22] (`https://arxiv.org/abs/1707.06855`) of this article.

Acknowledgements I am indebted to my advisor Martin Dietzfelbinger for drawing my attention to this problem as well as for providing a constant stream of useful literature recommendations. When discussing a preliminary version of this work at the Dagstuhl Seminar 17181 on Theory and Applications of Hashing, Michael Mitzenmacher and Konstantinos Panagiotou provided useful comments.

1 Introduction

In **standard cuckoo hashing** [20], a set $X = \{x_1, \dots, x_{cn}\}$ of objects (possibly with associated data) from a universe $\mathcal{U}$ is to be stored in a hash table indexed by $V = \{0, \dots, n-1\}$ of size n such that each object x_i resides in one of two associated memory locations $h_1(x_i), h_2(x_i)$, given by hash functions $h_1, h_2 : \mathcal{U} \to V$. In most theoretic works, these functions are modelled as fully random functions, selected uniformly and independently from $V^{\mathcal{U}}$.

The load parameter $c \in [0, 1]$ indicates the desired space efficiency, i.e. the ratio between objects and allocated table positions. Whether or not a valid placement of the objects in the table exists is well predicted by whether c is above or below the *threshold* $c^* = \frac{1}{2}$: If $c \leq c^* - \varepsilon$ for arbitrary $\varepsilon > 0$, then a placement exists with high probability (whp), i.e. with probability approaching 1 as n tends to infinity, and if $c \geq c^* + \varepsilon$ for $\varepsilon > 0$, then no placement exists whp.

If a placement is found, we obtain a dictionary data structure representing $X \subseteq \mathcal{U}$. To check whether an object $x \in U$ resides in the dictionary (and possibly retrieve associated data), only the memory locations $h_1(x)$ and $h_2(x)$ need to be computed and searched for x. Combined with results facilitating swift creation, insertion and deletion, standard cuckoo hashing has decent performance when compared to other hashing schemes at load factors around $\frac{1}{3}$ [20].

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 102; pp. 102:1–102:10
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Several generalisations have been studied that allow trading rigidity of the data structure – and therefore performance of lookup operations – for load thresholds closer to 1.

- In **k-ary cuckoo hashing**, due to Fotakis et al. [8], a general number $k \geq 2$ of hash functions is used.
- Dietzfelbinger and Weidling [6] propose partitioning the table into $\frac{n}{\ell}$ contiguous **blocks of size ℓ** and assign two random blocks to each object via the two hash functions, allowing an object to reside anywhere within those blocks.
- By **windows of size ℓ** we mean the related idea – called "cuckoo-lp" in [6] – where x may reside anywhere in the intervals $[h_1(x), h_1(x) + \ell)$ and $[h_2(x), h_2(x) + \ell)$ (all indices understood modulo n). Compared to the block variant, the values $h_1(x), h_2(x) \in V$ need not be multiples of ℓ, so the possible intervals do not form a partition of V.

The overall performance of a cuckoo hashing scheme is a story of multidimensional trade-offs and hardware dependencies, but based on experiments in [6, 17] roughly speaking, the following empirical claims can be made:

- k-ary cuckoo hashing for $k > 2$ is slower than the other two approaches. This is because lookup operations trigger up to k evaluations of hash functions and k random memory accesses, each likely to result in a cache fault. In the other cases, only the number of key comparisons rises, which are comparatively cheap.
- Windows of size ℓ offer a better tradeoff between worst-case lookup times and space efficiency than blocks of size ℓ.

Although our results are oblivious of hardware effects, they support the second empirical observation from a mathematical perspective.

1.1 Previous Work on Thresholds

Precise thresholds are known for k-ary cuckoo hashing [4, 12, 10], cuckoo hashing with blocks of size ℓ [7, 3], and the combination of both, i.e. k-ary cuckoo hashing with blocks of size ℓ with $k \geq 3, \ell \geq 2$ [9]. The techniques in the cited papers are remarkably heterogeneous and often specific to the cases at hand. Lelarge [18] managed to unify the above results using techniques from statistical physics that, perhaps surprisingly, feel like they grasp more directly at the core phenomena. Generalising further, Leconte, Lelarge, and Massoulié [15] solved the case where each object must occupy $j \in \mathbb{N}$ incident table positions, $r \in \mathbb{N}$ of which may lie in the same block (see also [13]).

Lehman and Panigrahy [17] showed that, asymptotically, the load threshold is $1 - (2/e + o_\ell(1))^\ell$ for cuckoo hashing with blocks of size ℓ and $1 - (1/e + o_\ell(1))^{1.59\ell}$ in the case of windows, with no implication for small constant ℓ. Beyer [2] showed in his master's thesis that for $\ell = 2$ the threshold is at least 0.829 and at most 0.981. To our knowledge, this is an exhaustive list of published work concerning windows.

In a spirit similar to cuckoo hashing with windows, Porat and Shalem [21] analyse a scheme where memory is partitioned into pages and a bucket of size k is a choice of k memory positions from the same page (not necessarily contiguous). The authors provide rigorous lower bounds on the corresponding thresholds as well as empirical results.

1.2 Our Contribution

We provide precise thresholds for k-ary cuckoo hashing with windows of size ℓ for all $k, \ell \geq 2$. In particular this solves the case of $k = 2$ left open in [6, 17]. Note the pronounced improvements in space efficiency when using windows over blocks, for instance in the case of $k = \ell = 2$, where the threshold is at roughly 96.5% instead of roughly 89.7%.

Formally, for any $k, \ell \geq 2$, we identify real analytic functions $f_{k,\ell}$, $g_{k,\ell}$, such that for $\gamma_{k,\ell} = \inf_{\lambda>0}\{f_{k,\ell}(\lambda) \mid g_{k,\ell}(\lambda) < 0\}$ we have

▶ **Main Theorem.** *The threshold for k-ary cuckoo hashing with windows of size ℓ is $\gamma_{k,\ell}$, in particular for any $\varepsilon > 0$,*

(i) *if $c > \gamma_{k,\ell} + \varepsilon$, then no valid placement of objects exists whp and*

(ii) *if $c < \gamma_{k,\ell} - \varepsilon$, then a valid placement of objects exists whp.*

While $f_{k,\ell}$ and $g_{k,\ell}$ are very unwieldy, with ever more terms as ℓ increases, numerical aproximations of $\gamma_{k,\ell}$ can be attained with mathematics software. We provide some values in Table 1.

1.3 Methods

The obvious methods to model cuckoo hashing with windows either give probabilistic structures with awkward dependencies or the question to answer for the structure follows awkward rules. Our first non-trivial step is to transform a preliminary representation into a hypergraph with n vertices, cn uniformly random hyperedges of size k, an added deterministic cycle, and a question strictly about the orientability of this hypergraph.

In the new form, the problem is approachable by a combination of belief propagation methods and the objective method [1], adapted to the world of hypergraph orientability by Lelarge [18] in his insightful paper. The results were further strengthened by a Theorem in [15], which we apply at a critical point in our argument.

As the method is fundamentally about approximate sizes of incomplete orientations, it leaves open the possibility of $o(n)$ unplaced objects; a gap that can be closed in an afterthought with standard methods.

Table 1 Some thresholds $c_{k,\ell}$ as obtained by [20, 3, 7, 4, 12, 11, 9] and values of $\gamma_{k,\ell}$ as obtained from our main theorem.
In both tables, the line for $\ell = 1$ corresponds to plain k-ary cuckoo hashing, reproduced here for comparison.

Thresholds $c_{k,\ell}$ for k-ary cuckoo hashing with blocks of size ℓ:

$\ell \backslash k$	2	3	4	5	6	7
1	0.5	0.9179352767	0.9767701649	0.9924383913	0.9973795528	0.9990637588
2	0.8970118682	0.9882014140	0.9982414840	0.9997243601	0.9999568737	0.9999933439
3	0.9591542686	0.9972857393	0.9997951434	0.9999851453	0.9999989795	0.9999999329
4	0.9803697743	0.9992531564	0.9999720661	0.9999990737	0.9999999721	0.9999999992

Thresholds $\gamma_{k,\ell}$ for k-ary cuckoo hashing with windows of size ℓ:

$\ell \backslash k$	2	3	4	5	6	7
1	0.5	0.9179352767	0.9767701649	0.9924383913	0.9973795528	0.9990637588
2	0.964994923	0.9968991072	0.9996335076	0.9999529036	0.9999937602	0.9999991631
3	0.994422754	0.9998255112	0.9999928198	0.9999996722	0.9999999843	0.9999999992
4	0.998951593	0.9999896830	0.9999998577	0.9999999977	≈ 1	≈ 1

1.4 Further Discussion

In the full version of this paper, we touch on three further issues that complement our results but are somewhat detached from our main theorem.

Numerical approximations of the thresholds. We explain how mathematics software can be used to get approximations for the values $\gamma_{k,\ell}$, which have been characterised only implicitely.

Speed of convergence. We provide experimental results with finite table sizes to demonstrate how quickly the threshold behaviour emerges.

Constructing orientations We examine the LSA algorithm by Khosla for insertion of elements, adapted to our hashing scheme. Experiments suggest an expected constant runtime per element as long as the load is bounded away from the threshold, i.e. $c < \gamma_{k,\ell} - \varepsilon$ for some $\varepsilon > 0$.

2 Definitions and Notation

A cuckoo hashing scheme specifies for each object $x \in X$ a set $A_x \subset V$ of table positions that x may be placed in. For our purposes, we may identitfy x with A_x. In this sense, $H = (V, X)$ is a hypergraph, where table positions are vertices and objects are hyperedges. The task of placing objects into admissible table positions corresponds to finding an *orientation* of H, which assigns each edge $x \in X$ to an incident vertex $v \in x$ such that no vertex has more than one edge assigned to it. If such an orientation exists, H is *orientable*.

We now restate the hashing schemes from the introduction in this hypergraph framework, switching to letters e (and E) to refer to (sets of) edges. We depart in notation, but not in substance, from definitions given previously, e.g. [8, 5, 17]. Illustrations are available in Figure 1.

Concerning ***k*-ary cuckoo hashing** the hypergraph is given as:

$$H_n = H^k_{n,cn} := (\mathbb{Z}_n, E = \{e_1, e_2, \dots, e_{cn}\}), \quad \text{for } e_i \leftarrow [{\textstyle\frac{\mathbb{Z}_n}{k}}], \tag{1}$$

where $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$ and for a set S and $k \in \mathbb{N}$ we write $e \leftarrow [\frac{S}{k}]$ to indicate that $e = \{s_1, s_2, \dots, s_k\}$ is obtained by picking $s_1, \dots, s_k$ independently and uniformly at random from S.

There is a subtle difference to picking e uniformly at random from $\binom{S}{k}$, the set of all k-subsets of S, as the elements $s_1, \dots, s_k$ need not be distinct. We therefore understand e as a multiset. Also, we may have $e_i = e_j$ for $i \neq j$, so E is a multiset as well.[1]

Assuming the table size n is a multiple of ℓ, ***k*-ary cuckoo hashing with blocks of size $\boldsymbol{\ell}$** is modelled by the hypergraph

$$B_n = B^{k,\ell}_{n,cn} := (\mathbb{Z}_n, \{e'_1, e'_2, \dots, e'_{cn}\}), \quad \text{where } e'_i = \bigcup_{j \in e_i} [j\ell, (j+1)\ell) \text{ and } e_i \leftarrow [{\textstyle\frac{\mathbb{Z}_{n/\ell}}{k}}], \tag{2}$$

that is, each hyperedge is the union of k blocks chosen uniformly at random from the set of all blocks, which are the n/ℓ intervals of size ℓ in $\mathbb{Z}_n$ that start at a multiple of ℓ. Note that for $\ell = 1$ we recover H_n.

Similarly, ***k*-ary cuckoo hashing with windows of size $\boldsymbol{\ell}$** is modelled by

$$W_n = W^{k,\ell}_{n,cn} := (\mathbb{Z}_n, \{e'_1, e'_2, \dots, e'_{cn}\}), \quad \text{where } e'_i = \bigcup_{j \in e_i} [j, j+\ell) \text{ and } e_i \leftarrow [{\textstyle\frac{\mathbb{Z}_n}{k}}], \tag{3}$$

that is, each hyperedge is the union of k windows chosen uniformly at random from the set of all windows, which are the n intervals of size ℓ in $\mathbb{Z}_n$, this time without alignment

[1] While our choice for the probability space is adequate for cuckoo hashing and convenient in the proof, such details are inconsequential. Choosing H_n uniformly from the set of all hypergraphs with cn *distinct* edges all of which contain k *distinct* vertices would be equivalent for our purposes.

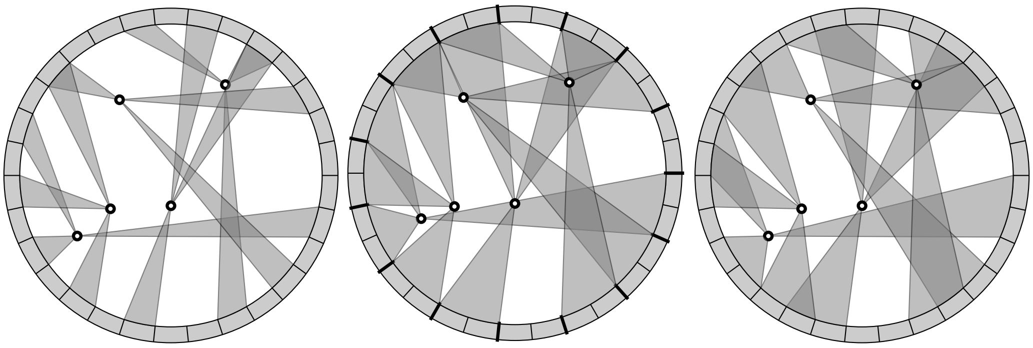

Figure 1 Drawing of possible outcomes for the hypergraphs H_n, B_n and W_n (modelling k-ary cuckoo hashing plain / with blocks / with windows) for $n = 30$, $c = \frac{1}{6}$, $k = 3$ and $\ell = 2$ (ℓ only for B and W). Each edge is drawn as a point and connected to all incident table cells, which are arranged in a circle. In the case of B, thick lines indicate the borders between blocks.

restriction. Note that intervals wrap around at the ends of the set $\{0, \ldots, n-1\}$ with no awkward "border intervals". Again, for $\ell = 1$ we recover H_n.

3 Outline of the Proof

Step 1: A tidier problem. The elements of an edge e of B_n and W_n are not independent, as e is the union of k intervals of size ℓ. This poorly reflects the actual tidiness of the probabilistic object. We may obtain a model with independent elements in edges, by switching to a more general notion of what it means to orient a hypergraph.

Formally, given a weighted hypergraph $H = (V, E, \eta)$ with weight function $\eta : V \cup E \to \mathbb{N}$, an *orientation* μ of H assigns to each pair (e, v) of an edge and an incident vertex a number $\mu(e, v) \in \mathbb{N}_0$ such that

$$\forall e \in E\colon \sum_{v \in e} \mu(e, v) = \eta(e), \text{ and } \forall v \in V\colon \sum_{e \ni v} \mu(e, v) \le \eta(v). \tag{4}$$

We will still say that an edge e *is oriented to* a vertex v (possibly several times) if $\mu(e, v) > 0$. One may be inclined to call $\eta(v)$ a *capacity* for $v \in V$ and $\eta(e)$ a *demand* for $e \in E$, but we use the same letter in both cases as the distinction is dropped later anyway.

Orientability of H, B and W from earlier is also captured in the generalised notion with implicit vertex weights of $\eta \equiv 1$.

A simplified representation of B_n is straightforward to obtain. We provide it mainly for illustration purposes, see Figure 2(a):

$$\hat{B}_n := \hat{B}^{k,\ell}_{n,cn} := (\mathbb{Z}_{n/\ell}, \{e_1, e_2, \ldots, e_{cn}\}, \eta), \quad \text{where } e_i \leftarrow \left[{\mathbb{Z}_{n/\ell} \atop k}\right] \tag{5}$$
$$\text{and } \eta(v) = \ell \text{ for } v \in \mathbb{Z}_{n/\ell} \text{ and } \eta(e_i) = 1 \text{ for } 1 \le i \le cn.$$

In $\hat{B}_n$, each group of ℓ vertices of B_n representing one block is now contracted into a single vertex of weight ℓ and edges contain k independent vertices representing blocks instead of $k\ell$ dependent vertices. It is clear that B_n is orientable if and only if $\hat{B}_n$ is orientable.

In a similar spirit we identify a transformed version $\hat{W}_n$ for W_n, but this time the details are more complicated as the vertices have an intrinsic linear geometry, whereas B_n featured essentially an unordered collection of internally unordered blocks. The *ordinary edges* in $\hat{W}_n$

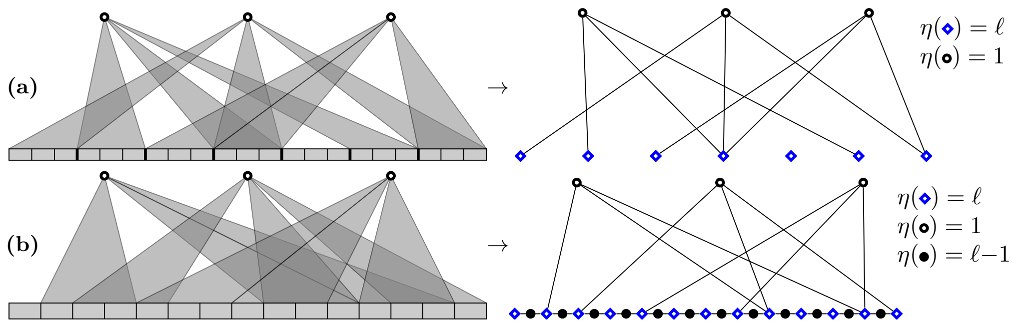

Figure 2 (a) In k-ary cuckoo hashing with blocks of size ℓ (here $k = \ell = 3$), we can contract each block into a single vertex of weight ℓ to obtain a simpler but equivalent representation of the orientation problem.
(b) In k-ary cuckoo hashing with windows of size ℓ, a similar idea can be made to work, but additional helper edges (drawn as ●) of weight $\ell - 1$ are needed (see Proposition 1).

also have size k instead of size $k\ell$, but we need to introduce additional *helper edges* that capture the linear geometry of $\mathbb{Z}_n$, see Figure 2(b). We define:

$$\begin{aligned} &\hat{W}_n := \hat{W}^{k,\ell}_{n,cn} := (\mathbb{Z}_n, C_n \cup \{e_1, \dots, e_{cn}\}, \eta) \qquad (6)\\ &\text{with ordinary edges } e_i \leftarrow \left[\tbinom{\mathbb{Z}_n}{k}\right], \text{ helper edges } C_n = \{c_i := (i, i+1) \mid i \in \mathbb{Z}_n\},\\ &\text{and weights } \eta(w) = \ell,\ \eta(h) = \ell - 1,\ \eta(e) = 1 \ \text{ for } \ w \in \mathbb{Z}_n, h \in C_n, e \in \{e_1, \dots, e_{cn}\}. \end{aligned}$$

Note that formally the graphs W_n and $\hat{W}_n$ are random variables on a common probability space. An outcome $\omega = (e_i)_{1 \le i \le cn}$ from this space determines both graphs.

The following proposition justifies the definition and is proved in the full version of this paper.

▶ **Proposition 1.** *$\hat{W}_n$ is orientable if and only if W_n is orientable.*[2]

An important merit of $\hat{W}_n$ that will be useful in Step 3 is that it is *locally tree-like*, meaning each vertex has a probability of $o(1)$ to be involved in a constant-length cycle. Here, by a cycle in a hypergraph we mean a sequence of distinct edges $e_1, e_2, \dots, e_j$ such that successive edges share a vertex and e_j and e_1 share a vertex.

Note the interesting special case $\hat{W}^{2,2}_{n,cn}$, which is a cycle of length n with cn random chords, unit edge weights and vertices of weight 2. Understanding the orientability thresholds for this graph seems interesting in its own right, not just as a means to understand $W^{2,2}_{n,cn}$.

Step 2: Incidence Graph and Allocations. The next step is by no means a difficult or creative one, we merely perform the necessary preparations needed to apply [15], introducing their concept of an allocation in the process.

This will effectively get rid of the asymmetry between the roles of vertices and edges in the problem of orienting $\hat{W}_n$, by switching perspective in two simple ways. The first is to consider the incidence graph G_n of $\hat{W}_n$ instead of $\hat{W}_n$ itself, i.e. the bipartite graph

$$G_n = G^{k,\ell}_{n,cn} = (\underbrace{C_n}_{A_C} \cup \underbrace{\{e_1, \dots, e_{cn}\}}_{A_R}, \underbrace{\mathbb{Z}_n}_{B}, \underbrace{\text{“}\ni\text{”}}_{E(G_n)}). \qquad (7)$$

[2] Formally this should read: The events $\{W_n \text{ is orientable}\}$ and $\{\hat{W}_n \text{ is orientable}\}$ coincide.

We use $A = A_C \cup A_R$ to denote those vertices of G_n that were edges in $\hat{W}_n$ and B for those vertices of G_n that were vertices in $\hat{W}_n$. Vertices $a \in A$ and $b \in B$ are adjacent in G_n if $b \in a$ in $\hat{W}_n$. The weights η on vertices and edges in $\hat{W}_n$ are now vertex weights with $\eta(a_C) = \ell - 1$, $\eta(a_R) = 1$, $\eta(b) = \ell$ for $a_C \in A_C$, $a_R \in A_R$, $b \in B$. The notion of μ being an orientation translates to μ being a map $\mu : E(G_n) \to \mathbb{N}_0$ such that $\sum_{b \in N(a)} \mu(a, b) = \eta(a)$ for all $a \in A$ and $\sum_{a \in N(b)} \mu(a, b) \le \eta(b)$ for all $b \in B$. Note that vertices from A need to be *saturated* ("$= \eta(a)$" for $a \in A$) while vertices from B need not be ("$\le \eta(b)$" for $b \in B$). This leads to the second switch in perspective.

Dropping the saturation requirement for A, we say μ is an *allocation* if $\sum_{u \in N(v)} \mu(u, v) \le \eta(v)$ for all $v \in A \cup B$.

Clearly, any orientation is an allocation, but not vice versa; for instance, the trivial map $\mu \equiv 0$ is an allocation. Let $|\mu|$ denote the size of an allocation, i.e. $|\mu| = \sum_{e \in E} \mu(e)$. By bipartiteness, no allocation can have a size larger than the total weight of A, i.e.

$$\text{for all allocations } \mu\colon |\mu| \le \eta(A) = \sum_{a \in A} \eta(a) = |A_C| \cdot (\ell - 1) + |A_R| \cdot 1 = (\ell - 1 + c)n$$

and it is precisely the orientations of G_n that have size $\eta(A)$. We conclude:

▶ **Proposition 2.** *Let $M(G_n)$ denote the maximal size of an allocation of G_n. Then*

$$\frac{M(G_n)}{n} = \ell - 1 + c \quad \textit{if and only if} \quad G_n \textit{ is orientable} \quad \textit{if and only if} \quad \hat{W}_n \textit{ is orientable.}$$

Step 3: The Limit T of G_n. Reaping the benefits of step 1, we find G_n to have $O(1)$ cycles of length $O(1)$ whp. To capture the local appearance of G_n even more precisely, let the r*-ball* around a vertex v in a graph be the subgraph induced by the vertices of distance at most r from v. Then the r-ball around a random vertex of G_n is distributed, as n gets large, more and more like the r-ball around the root of a random infinite rooted tree $T = T_c^{k,\ell}$. It is distributed as follows, with nodes of types A_C, A_R or B.

- The root of T is of type A_C, A_R or B with probability $\frac{1}{2+c}$, $\frac{c}{2+c}$ and $\frac{1}{2+c}$, respectively.
- If the root is of type A_C, it has two children of type B. If it is of type A_R, it has k children of type B. If it is of type B, it has two children of type A_C and a random number X of children of type A_R, where $X \sim \text{Po}(kc)$. Here $\text{Po}(\lambda)$ denotes the Poisson distribution with parameter λ.
- A vertex of type A_C that is not the root has one child of type B. A vertex of type A_R that is not the root has $k - 1$ children of type B.
- A vertex of type B that is not the root has a random number X of children of type A_R, where $X \sim \text{Po}(kc)$. If its parent is of type A_C, then it has one child of type A_C, otherwise it has two children of type A_C.
- Vertices of type A_C, A_R and B have weight $\ell{-}1$, 1 and ℓ, respectively.

All random decisions should be understood to be independent. A type is also treated as a set containing all vertices of that type. In the full version of this paper we briefly recall the notion of *local weak convergence* and argue that the following holds:

▶ **Proposition 3.** $(G_n)_{n \in \mathbb{N}} = (G_{n,cn}^{k,\ell})_{n \in \mathbb{N}}$ *converges locally weakly to* $T = T_c^{k,\ell}$.

Step 4: The Method of [15]. We are now in a position to apply a powerful Theorem due to Leconte, Lelarge, and Massoulié [15] that characterises $\lim_{n \to \infty} \frac{M(G_n)}{n}$ in terms of solutions to belief propagation equations for T. Put abstractly: The limit of a function of G_n is a function of the limit of G_n. We elaborate on details and deal with the equations in the full version of this paper. After condensing the results into a characterisation of $\gamma_{k,\ell} \in (0,1)$ in terms of "well-behaved" functions we obtain:

▶ **Proposition 4.**

$$\lim_{n\to\infty} \tfrac{M(G_{n,cn})}{n} \begin{cases} = \ell - 1 + c & \textit{almost surely} & \textit{if } c < \gamma_{k,\ell} \\ < \ell - 1 + c & \textit{almost surely} & \textit{if } c > \gamma_{k,\ell} \end{cases}$$

Step 5: Closing the Gap. It is important to note that we are not done, as

$$\lim_{n\to\infty} \tfrac{M(G_{n,cn})}{n} = \ell - 1 + c \text{ a.s.} \quad \text{does not imply}^3 \quad M(G_{n,cn}) = n \cdot (\ell - 1 + c) \text{ whp.} \tag{8}$$

We still have to exclude the possibility of a gap of size $o(n)$ on the right hand side, imagine for instance $M(G_{n,cn}) = (\ell - 1 + c)n - \sqrt{n}$ to appreciate the difference. In the setting of cuckoo hashing with double hashing (see [16]), it is actually the analogue of this pesky distinction that seems to be in the way of proving precise thresholds for perfect orientability, so we should treat this seriously.

Luckily the line of reasoning by Lelarge [18] can be adapted to our more general setting. The key is to prove that if not all objects can be placed into the hash table, then the configuration causing this problem has size $\Theta(n)$ (and those large overfull structures do not go unnoticed on the left side of (8)).

▶ **Lemma 1.** *There is a constant $\delta > 0$ such that whp no set of $0 < t < \delta n$ vertices in $\hat{W}_n$ (of weight ℓt) induces edges of total weight ℓt or more, provided $c \leq 1$.*

The proof of this Lemma (using first moment methods) and the final steps towards our main theorem are found in the full version of this paper.

4 Conclusion and Outlook

We established a method to determine load thresholds $\gamma_{k,\ell}$ for k-ary cuckoo hashing with (unaligned) windows of size ℓ. In particular, we resolved the cases with $k = 2$ left open in [6, 17], confirming corresponding experimental results by rigorous analysis.

The following four questions may be worthwhile starting points for further research.

Is there more in this method? It is conceivable that there is an insightful simplification of our calculations that yields a less unwieldy characterisation of $\gamma_{k,\ell}$. We also suspect that the threshold for the appearance of the $(\ell+1)$-*core* of $\hat{W}_n$ can be identified with some additional work (for cores see e.g. [19, 14]). This threshold is of interest because it is the point where the simple peeling algorithm to compute an orientation of $\hat{W}_n$ breaks down.

Can we prove efficient insertion? Given our experiments concerning the performance of Khosla's LSA algorithm for inserting elements in our hashing scheme (for details refer to the full version), it seems likely that its runtime is linear. But one could also consider approaches that do not insert elements one by one but build a hash table of load $c = \gamma_{k,\ell} - \varepsilon$ given all elements at once. Something in the spirit of the selfless algorithm [3] or excess degree reduction [4] may offer linear runtime with no performance degradation as ε gets smaller, at least for $k = 2$.

How good is it in practice? This paper does not address the competitiveness of our hashing scheme in realistic practical settings. The fact that windows give higher thresholds than (aligned) blocks for the *same* parameter ℓ may just mean that the "best" ℓ for a particular use case is lower, not precluding the possibility that the associated performance benefit is outweighed by other effects. [6] provide a few experiments in their appendix suggesting slight advantages for windows in the case of unsuccessful searches and slight disadvantages for successful searches and insert operations, in one very particular setup with $k = 2$. Further research could take into account precise knowledge of cache effects on modern machines, possibly using a mixed approach, respecting alignment only insofar as it is favoured by the caches. Ideas from Porat and Shalem [21] could prove beneficial in this regard.

What about other geometries? We analysed linear hash tables where objects are assigned random intervals. One could also consider a square hash table $(\mathbb{Z}_{\sqrt{n}})^2$ where objects are assigned random squares of size $\ell \times \ell$ (with no alignment requirement). We suspect that understanding the thresholds in such cases would require completely new techniques.

References

1 David Aldous and J. Michael Steele. *The Objective Method: Probabilistic Combinatorial Optimization and Local Weak Convergence*, pages 1–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-662-09444-0_1.

2 Stephan Beyer. Analysis of the Linear Probing Variant of Cuckoo Hashing. Master's thesis, Technische Universität Ilmenau, 2012. URL: http://gso.gbv.de/DB=2.1/PPNSET?PPN=685166759.

3 Julie Anne Cain, Peter Sanders, and Nicholas C. Wormald. The Random Graph Threshold for k-orientiability and a Fast Algorithm for Optimal Multiple-Choice Allocation. In *Proc. 18th SODA*, pages 469–476, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283433.

4 Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight Thresholds for Cuckoo Hashing via XORSAT. In *Proc. 37th ICALP (1)*, pages 213–225, 2010. doi:10.1007/978-3-642-14165-2_19.

5 Martin Dietzfelbinger and Christoph Weidling. Balanced Allocation and Dictionaries with Tightly Packed Constant Size Bins. In *Proc. 32nd ICALP*, pages 166–178, 2005. doi:10.1007/11523468_14.

6 Martin Dietzfelbinger and Christoph Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci.*, 380(1-2):47–68, 2007. doi:10.1016/j.tcs.2007.02.054.

7 Daniel Fernholz and Vijaya Ramachandran. The k-orientability Thresholds for $G_{n,p}$. In *Proc. 18th SODA*, pages 459–468, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283432.

8 Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul G. Spirakis. Space Efficient Hash Tables with Worst Case Constant Access Time. *Theory Comput. Syst.*, 38(2):229–248, 2005. doi:10.1007/s00224-004-1195-x.

9 Nikolaos Fountoulakis, Megha Khosla, and Konstantinos Panagiotou. The Multiple-Orientability Thresholds for Random Hypergraphs. In *Proc. 22nd SODA*, pages 1222–1236, 2011. URL: http://www.siam.org/proceedings/soda/2011/SODA11_092_fountoulakisn.pdf.

10 Nikolaos Fountoulakis and Konstantinos Panagiotou. Orientability of Random Hypergraphs and the Power of Multiple Choices. In *Proc. 37th ICALP (1)*, pages 348–359, 2010. doi:10.1007/978-3-642-14165-2_30.

11 Nikolaos Fountoulakis and Konstantinos Panagiotou. Sharp Load Thresholds for Cuckoo Hashing. *Random Struct. Algorithms*, 41(3):306–333, 2012. doi:10.1002/rsa.20426.

12 Alan M. Frieze and Páll Melsted. Maximum Matchings in Random Bipartite Graphs and the Space Utilization of Cuckoo Hash Tables. *Random Struct. Algorithms*, 41(3):334–364, 2012. doi:10.1002/rsa.20427.

13 Pu Gao and Nicholas C. Wormald. Load Balancing and Orientability Thresholds for Random Hypergraphs. In *Proc. 42nd STOC*, pages 97–104, 2010. doi:10.1145/1806689.1806705.

14 Svante Janson and Malwina J. Luczak. A simple solution to the k-core problem. *Random Struct. Algorithms*, 30(1-2):50–62, 2007. doi:10.1002/rsa.20147.

15 M. Leconte, M. Lelarge, and L. Massoulié. Convergence of multivariate belief propagation, with applications to cuckoo hashing and load balancing. In *Proc. 24th SODA*, pages 35–46, 2013. URL: http://dl.acm.org/citation.cfm?id=2627817.2627820.

16 Mathieu Leconte. Double hashing thresholds via local weak convergence. In *51st Annual Allerton Conference on Communication, Control, and Computing*, pages 131–137, 2013. doi:10.1109/Allerton.2013.6736515.

17 Eric Lehman and Rina Panigrahy. 3.5-Way Cuckoo Hashing for the Price of 2-and-a-Bit. In *Proc. 17th ESA*, pages 671–681, 2009. doi:10.1007/978-3-642-04128-0_60.

18 Marc Lelarge. A New Approach to the Orientation of Random Hypergraphs. In *Proc. 23rd SODA*, pages 251–264, 2012. URL: http://dl.acm.org/citation.cfm?id=2095139.

19 Michael Molloy. Cores in random hypergraphs and boolean formulas. *Random Struct. Algorithms*, 27(1):124–135, 2005. doi:10.1002/rsa.20061.

20 Rasmus Pagh and Flemming Friche Rodler. Cuckoo Hashing. *J. Algorithms*, 51(2):122–144, 2004. doi:10.1016/j.jalgor.2003.12.002.

21 Ely Porat and Bar Shalem. A Cuckoo Hashing Variant with Improved Memory Utilization and Insertion Time. In *Proc. 22nd DCC*, 2012. doi:10.1109/DCC.2012.41.

22 Stefan Walzer. Load thresholds for cuckoo hashing with overlapping blocks. *CoRR*, abs/1707.06855, 2017. arXiv:1707.06855.