FairHash:
A Fair and Memory/Time-efficient Hashmap

Nima Shahbazi, Stavros Sintos, Abolfazl Asudeh

Department of Computer Science
University of Illinois Chicago

ACM SIGMOD, Santiago Chile
June 2024

Santiago
Chile

SIGMOD
PODS
2024

Motivation

Little attention to **fairness-aware data structures**

Hashmaps are the founding block of many applications

- Bloom filter, Count sketches, Min-wise hashing, etc.

This paper:

- Revisits hashmaps through the lens of group fairness

Review: Traditional Hash Functions

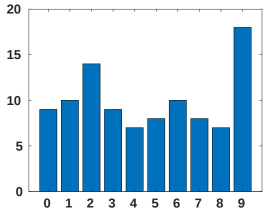
Traditional k-wise independent hashing [Sie89]

- Randomly map a key to a random value in a specific output range
- Unlikely that independent random value assignment distribute points uniformly in the buckets
- (Related topic: The Occupancy Problem [MR95])

Example

100 iid integers in range $[0, 9]$

- Not uniformly distributed within the buckets
- Number of collisions minimized when uniform distribution is satisfied



Review: Data-informed Hashmaps

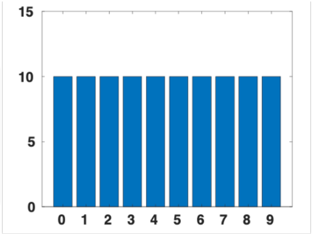
Learn a hash function that uniformly distributes the data across different buckets [KBC+18][1]

- CDF of data is constructed
- Range of values are partitioned into equi-size buckets

Example

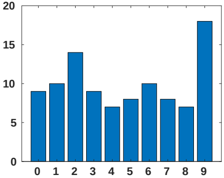
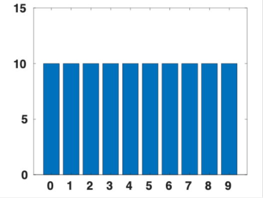
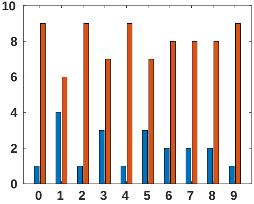
Data-informed Hashmap learned over 100 integers in range $[0, 9]$

- Uniformly distributed within the buckets

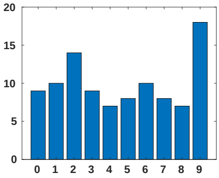
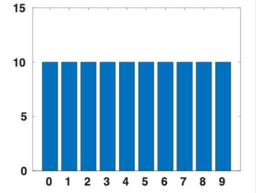
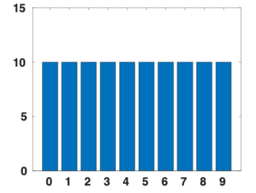
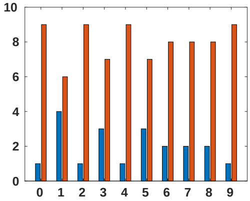
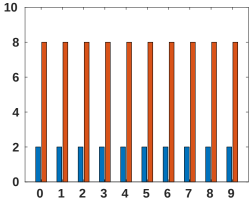


[1] We refer to [KBC+18] as CDF-based hashmap.

Motivation at a Glance

	Traditional hashmaps	Data-informed hashmaps	
Overall Dist.			
Group-level Dist.			

Motivation at a Glance

	Traditional hashmaps	Data-informed hashmaps	(0-unfair) **FairHash**
Overall Dist.			
Group-level Dist.			

Fairness Definitions

Given

1. A set P of n points in $\mathbb{R}^d$
   - Each point belong to one of the k demographic groups $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_k\}$
2. A hashmap $\mathcal{H}$ with
   - m buckets, $b_1, \ldots, b_m$
   - a hash function $h : \mathbb{R}^d \to [1, m]$

   that maps each point $p \in P$ to one of the m buckets.

Fairness Definitions

Collision Probability

- $\forall$ random pairs $p \in P, q \in P$:
  - $Pr[h(p) = h(q)] = \frac{1}{m}$

Single fairness

- $\forall$ random points $p_i \in \mathbf{g}_i$:
  - $Pr[h(p_i) = h(x)] = \ldots = Pr[h(p_k) = h(x)] = \frac{1}{m}$

Pairwise fairness

- $\forall$ random pairs $p_i \in \mathbf{g}_i$ and $q_i \in \mathbf{g}_i$:
  - $Pr[h(p_i) = h(q_i)] = \ldots = Pr[h(p_k) = h(q_k)] = \frac{1}{m}$
- The **strongest notion of fairness**: if satisfied, the other two are also satisfied.

(ε, α)-hashmap

ε-unfairness

A hashmap is ε**-unfair**, if and only if

$$\frac{\max_{\mathbf{g}\in\mathcal{G}}(Pr_{\mathbf{g}})}{1/m} \leq (1+\varepsilon) \;\Rightarrow\; \max_{\mathbf{g}\in\mathcal{G}}(Pr_{\mathbf{g}}) \leq \frac{1}{m}(1+\varepsilon) \tag{1}$$

α-mermory

A We say a hashmap with m buckets satisfies α**-memory**, if and only if it stores at most $\alpha(m-1)$ boundary points.

(ε, α)-hashmap

A hashmap that is ε**-unfair** and satisfies α**-memory**.

Comparisons

Hashmap	Architecture	Query time	Collision probability	Single fairness	Pairwise fairness
traditional	data-independent	$O(1)$	✗	✗	✗
CDF-based	data-dependent	$O(\log m)$	✓	✓	✗
FairHash	data-dependent	$O(\log m)$	✓	✓	✓

Summary of algorithmic results

Algorithm	Assumptions		Performance[a]		
	No. Attributes	No. Groups	(ε, α)-hashmap	Query time	Pre-processing time
Ranking	$d \geq 2$	$k \geq 2$	$(\varepsilon_R, 1)$	$O(\log m)$	$O(n^d \log n)$
Sweep&Cut	$d \geq 1$	$k \geq 2$	$(0, \frac{n}{m})$	$O(\log n)$	$O(n \log n)$
Necklace_{2g}	$d \geq 1$	2	$(0, 2)$	$O(\log m)$	$O(n \log n)$
Necklace_{kg}	$d \geq 1$	$k > 2$	$(0, k(4 + \log n))$	$O(\log(km \log n))$	$O(mk^3 \log n + knm(n + m))$

[a] The approximate collision results are provided in the paper [SSA24].

Observation: Only the *ordering* between the tuples specify the buckets in the CDF-based hashmap.

Idea:

- Combine the attribute of a point $p \in P$ into a single score $f(p)$, using a (ranking) function $f : \mathbb{R}^d \to \mathbb{R}$
- Construct the hashmap on $f(p)$.
- Objective: Find the function f, according to which the unfairness is minimized.

Algorithm Overview. Use computational geometry concepts and Linear functions $f(p) = w^\top p$

- Consider points as hyperplanes in the dual space
- Design a Ray-sweeping algorithm to efficiently find all possible orderings $\rightarrow$ return a function that minimizes unfairness.

Cut-based Algorithms

Observation: Buckets do not necessarily need to be continuous!

Idea:

- Partition the values into more than m "*bins*".
- Many-to-one mapping: Several bins are assigned to each bucket.

Theorem

Independent of how the points are distributed and their orders, there *always* exists a cut-based hashmap that is **0-unfair**.

Cut-based Algorithms: SWEEP&CUT $(0, \frac{n}{m})$-hashmap

Algorithm Overview. Make two sorted passes over P

1. First pass: (knowing the number of tuples each bucket should contain from each group) For every tuple record the bucket it should belong
2. Second pass: add a cut between each pair of points that belong to different buckets.

Cut-based Algorithms: $\textsc{Necklace}_{2g}$ $(0, 2)$-hashmap

Necklace Splitting Problem [AG21]

Divide a necklace of T beads of n' types between k' agents, such that

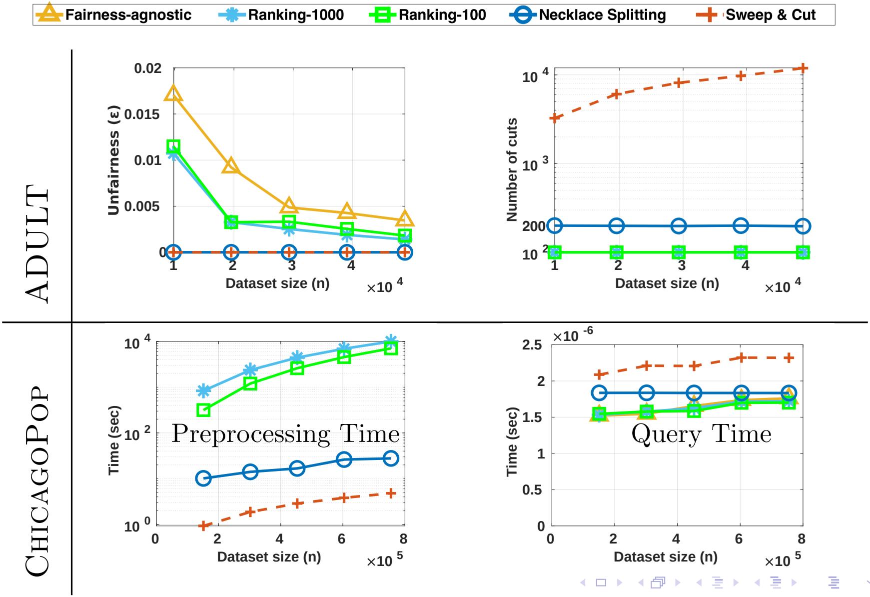
1. all agents receive the same amount of beads from each type.
2. the number of splits to the necklace is minimized.

Reduction: points $\rightarrow$ beads; group $\rightarrow$ bead type; buckets $\rightarrow$ agents

Algorithm Overview (2-groups): Iterative Algorithm

- Consider sorted P as a circle (p_n comes before p_1)
- Key idea: The circle *always* has at least one consecutive window of size $\frac{n}{m}$ that contains $\frac{|\mathbf{g}_1|}{m}$ points from $\mathbf{g}_1$ (and hence $\frac{|\mathbf{g}_2|}{m}$ points from $\mathbf{g}_2$).
- At each iteration, *efficiently* find such a window; carve it out of the circle; connect the two ends to form the circle for the next iteration

Highlighted Experiment Results



Thank you!

Figure: Github Repository

References

- Noga Alon and Andrei Graur, *Efficient splitting of necklaces*, ICALP 2021, 2021.
- Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis, *The case for learned index structures*, SIGMOD, 2018, pp. 489–504.
- Rajeev Motwani and Prabhakar Raghavan, *Randomized algorithms*, Cambridge university press, 1995.
- Alan Siegel, *On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications*, FOCS, 1989, pp. 20–25.
- Nima Shahbazi, Stavros Sintos, and Abolfazl Asudeh, *Fairhash: A fair and memory/time-efficient hashmap*, SIGMOD **2** (2024), no. 3, 1–29.