Sampling zeros of a sparse tensor

Bora Uçar

CNRS and LIP, ENS Lyon, France

Molière Workshop 6-8 July 2021

Joint work with:

Jules Bertrand
ENS de Lyon

Fanny Dufossé
DataMove, Inria

Problem

Given a d-dimensional sparse tensor $\mathcal{T}$, set up data structures to answer queries of the form

"is $t_{i_1,i_2,\ldots,i_d}$ zero or nonzero?"

Specifications/Requirements:

- Small memory overhead
- Fast construction
- Query response in $O(d)$ time.

Context and motivation

Kolda and Hong propose a stochastic, iterative method for efficient GCP decomposition of both dense and sparse tensors.

Stratified sampling

The nonzeros and the zeros are sampled separately in each iteration.

How to sample zeros: Sort, then use binary search for queries.

Other sampling approaches investigated, and the stratified sampling approach is demonstrated to be more useful numerically.

Motivation

Increase efficiency of the stratified sampling approach by developing data structures and algorithms for quickly detecting whether a given position in a tensor is zero or not.

What is available?

Coordinate format, each nonzero is listed with its d indices (and value). Z nonzeros.

Radix-sort

- $O(dZ)$ construction time ✓
- $O(Z)$ storage ✓
- $O(d \log Z)$ query response time in the worst case ✗

Static hashing

Minimal perfect hash functions (MPHFs): static data structures that map a given set of Z elements to $\{0, \ldots, Z-1\}$.

Use of an MPHF: store an id for a nonzero in a space of size Z and answer queries in constant time in the worst case.

The efficient ones fit the bill: fast construction, small memory overhead, and $O(d)$ response time.

Our approach

Adapt static hashing of Fredman, Komlós, & Szeremédi,'84 (FKS).

Key properties:

- two computations of the form

$$(\mathrm{k}^T \mathrm{x} \bmod p) \bmod Z$$

  to hash a nonzero (a d-tuple).

- $O(Z)$ memory
- Worst case $O(d)$ query response time
- Fast construction

Review of FKS static hashing method

Universe (of integers) $U = \{0, \ldots, u-1\}$, and $S \subseteq U$ with $|S| = n$ be the set to be represented. A prime number $p > u - 1$.

The first level hash function for a k, $h_k(x) = (kx \bmod p) \bmod n$.

(assigns x to bucket $B_{h_k(x)}$.)

The second level hash function $h_{k^{(i)}}(x) = (k^{(i)}x \bmod p) \bmod b_i^2$ for B_i with $b_i > 0$ elements.

- $\sum_i b_i^2$ should be $O(n)$ so that the method uses linear space (the first level hash function defines)
- Each $k^{(i)}$ should be an injection for the respective bucket (the second level hash function)

FKS: Construction and query

The values k and $k^{(i)}$ are found by random sampling and trials.

- First level: Sample k, if $\sum_i b_i^2 \leq 3n$, accept it. Otherwise, retry.
- B_i: Sample $k^{(i)}$; if $h_{k^{(i)}}(\cdot)$ is an injection accept. Otherwise, retry.

Efficient construction: $O(n)$ in expectation

Accept $\sum_i b_i^2 \leq 5n$ and use $2b_i^2$ space for each bucket B_i.

At least one half of the potential k and $k^{(i)}$ values guarantee that the two requirements are met.

Query: Compute the bucket with the first level hash, then within a bucket check the unique possible location with the second level hash.

Direct adaptations of FKS

Each nonzero is a d-tuple: linearize and use a unique integer for each nonzero (e.g., $[x_0, x_1, x_2]$ is $x_0 + s_0 \times (x_1 + s_1 \times x_2)$, where s_i is the size of the tensor on dimension i).

For large tensors, the numbers are too big

Direct adaptations of FKS

The universe U: all d-tuples $[x_0, \ldots, x_{d-1}]$ where $0 \leq x_i < p$.

The first level hash function for a k: $h(\mathrm{x}) = (\mathrm{k}^T\mathrm{x} \bmod p) \bmod Z$. The nonzero x is in bucket B_i where $i = h(\mathrm{x})$.

Lemma

For any given set $E \subseteq U$ of Z nonzeros, there is a $\mathrm{k} \in U$ such that with the hash function $(\mathrm{k}^T\mathrm{x} \bmod p) \bmod Z$, we have $\sum_{i=0}^{Z-1} b_i^2 < 4Z$.

Corollary

For any $E \subseteq U$, a set of Z nonzeros, with at least half of the potential $\mathrm{k} \in U$, we have $\sum_{i=0}^{Z-1} b_i^2 < 7Z$.

The linear space requirement is met for storing ids.

Direct adaptations of FKS

Lemma

For each bucket B_i with $b_i > 0$ elements, there is a $\mathrm{k}' \in U$ such that the function $(\mathrm{k}'^T \mathrm{x} \bmod p) \bmod b_i^2$ is an injection for $p \gg b_i^2$.

Corollary

Let B_i be a bucket with $b_i > 0$ elements. For at least half of the d-tuples $\mathrm{k}' \in U$, it holds that the function $(\mathrm{k}'^T \mathrm{x} \bmod p) \bmod 2b_i^2$ defines an injection for the elements of B_i for $p \gg b_i^2$.

$O(d)$ look-up time is met.

In expected linear time, $O(dZ)$, we can find k and all $\mathrm{k}^{(i)}$.

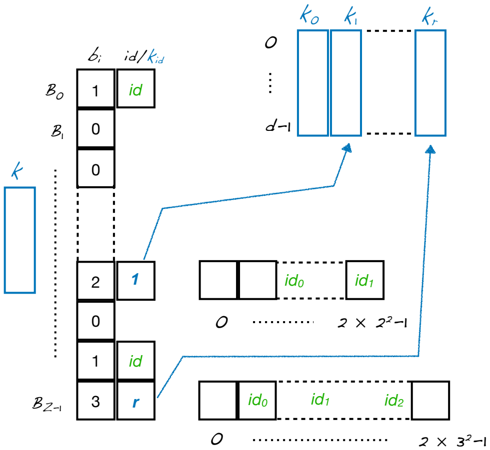
A lean variant of FKS

Previous lemma sand corollaries show that we can use the FKS method.

There is catch: there are $\Omega(Z)$ buckets, and storing a d-tuple $\mathrm{k}^{(i)}$ per bucket means $\Omega(dZ)$ storage, equivalent to the tensor itself. This is too much, and grows with d.

A lean variant of FKS

Share the second level hash functions among the buckets.



For each bucket B_i:

- the number b_i of nonzeros.
- If b_i is 1, the id of the nonzero mapping to i is stored.
- If b_i is larger than 1, then the index of a d-tuple in K is stored, and a space of size $2b_i^2$ to hold the ids of b_i nonzeros assigned to B_i.

A lean variant of FKS

Theorem

In expectation $O(\log_2 Z)$ different $k' \in K$ suffice.

Construction: $O(dZ \log_2 Z)$ a worst-case bound

Memory: $2Z - m + \sum_{i,b_i>1} 2b_i^2$. The set K adds another $O(d \log_2 Z)$ space. Provable $O(Z)$.

Query response: $O(d)$ in the worst case.

Experiments

Compare the proposed FKSlean with

- **HashàlaFKS:** the standard average-case constant time hashing method available in `C++std` as `unordered_map`, with which we propose to use the first level hash function $(\mathrm{k}^T\mathrm{x} \bmod p) \bmod Z$ of FKSlean. HashàlaFKS will be used as the baseline.
- **BBH:** a minimal perfect hash function (2017).
- **RecSplit:** another minimal perfect hash function (2020).
- **PTHash:** the most recent minimal perfect hash function to the best of our knowledge (2021).

Tensors from FROSTT, random Erdös–Renyi-like random tensors, $\mathcal{R}(d, s, Z)$. Also matrices from The SuiteSparse Matrix Collection.

Construction time: Real-life data

name	HashàlaFKS	uRecSplit (8,100)	uRecSplit (5,5)	uBBH (1)	uBBH (5)	uPTHash (1)	uPTHash (2)	uPTHash (3)	uPTHash (4)	FKSlean
kmer_A2a	591.28	1.64	0.47	1.74	0.71	0.66	0.42	0.93	0.53	0.63
queen_4147	495.18	1.76	0.46	1.72	0.71	0.64	0.40	0.88	0.52	0.73
com-Orkut	355.32	1.73	0.45	1.53	0.69	0.61	0.39	0.84	0.51	0.61
nell-1	213.19	1.75	0.42	1.28	0.75	0.62	0.41	0.81	0.51	0.73
delicious-3d	196.96	1.83	0.43	1.28	0.74	0.63	0.40	0.82	0.51	0.64
delicious-4d	201.39	1.81	0.44	2.00	1.07	0.70	0.47	0.89	0.59	0.78
flickr-3d	163.38	1.78	0.40	1.22	0.71	0.61	0.39	0.78	0.50	0.61
flickr-4d	163.54	1.78	0.41	1.84	1.01	0.66	0.45	0.84	0.56	0.61
nell-2	111.59	1.74	0.38	1.18	0.62	0.58	0.38	0.73	0.48	0.59
enron	72.88	1.90	0.41	1.80	0.89	0.64	0.44	0.79	0.55	0.68
vast-2015-mc1-3d	34.71	1.87	0.38	1.20	0.49	0.52	0.36	0.65	0.44	0.74
vast-2015-mc1-5d	33.40	1.97	0.41	1.99	0.84	0.60	0.43	0.73	0.52	0.72
chicago-crime	5.77	2.29	0.43	1.99	0.76	0.62	0.46	0.72	0.56	0.68
uber	3.17	2.57	0.48	2.21	0.84	0.66	0.50	0.75	0.60	0.75
lbnl-network	1.29	3.23	0.59	2.99	1.15	0.67	0.54	0.76	0.62	0.87
geo-mean		1.94	0.43	1.67	0.78	0.63	0.43	0.79	0.53	**0.69**

Table: The construction time. HashàlaFKS in seconds, others as ratios to that of HashàlaFKS.

Query response time: Real-life data

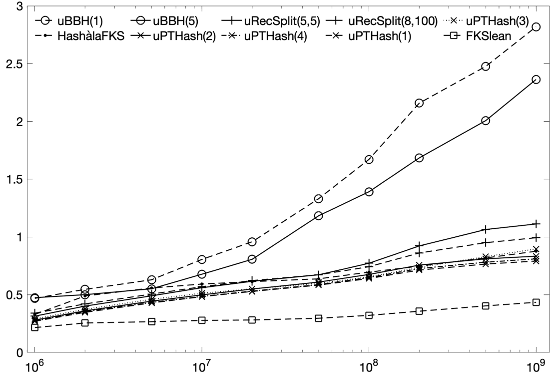
name	HashàlaFKS	uRecSplit (8,100)	uRecSplit (5,5)	uBBH (1)	uBBH (5)	uPTHash (1)	uPTHash (2)	uPTHash (3)	uPTHash (4)	FKSlean
kmer_A2a	0.78	1.08	1.17	2.45	1.82	0.54	0.60	0.74	0.56	0.49
queen_4147	0.74	1.13	1.14	2.61	1.99	0.54	0.61	0.73	0.56	0.50
com-Orkut	0.73	1.03	1.07	1.70	1.31	0.51	0.56	0.71	0.53	0.47
nell-1	0.72	1.03	1.08	2.61	2.01	0.66	0.77	0.77	0.68	0.48
delicious-3d	0.71	1.08	1.11	2.63	2.07	0.59	0.69	0.67	0.62	0.45
delicious-4d	0.71	1.12	1.19	2.76	2.21	0.93	1.00	0.97	0.95	0.48
flickr-3d	0.71	0.97	0.99	2.14	1.69	0.62	0.73	0.72	0.64	0.46
flickr-4d	0.70	1.11	1.17	2.63	2.14	0.94	0.99	0.95	0.95	0.47
nell-2	0.66	0.98	0.98	2.22	1.86	0.65	0.76	0.76	0.67	0.46
enron	0.64	1.06	1.07	2.13	1.90	0.92	0.97	0.95	0.93	0.47
vast-2015-mc1-3d	0.61	0.95	0.92	1.54	1.36	0.66	0.75	0.76	0.67	0.48
vast-2015-mc1-5d	0.63	1.03	1.01	1.67	1.54	0.89	0.92	0.92	0.89	0.45
chicago-crime	0.56	0.88	0.86	1.16	0.94	0.75	0.78	0.80	0.75	0.47
uber	0.53	0.91	0.86	1.12	1.00	0.75	0.77	0.80	0.75	0.49
lbnl-network	0.46	0.90	0.85	1.20	1.13	0.76	0.79	0.81	0.77	0.54
geo-mean		1.01	1.02	1.94	1.61	0.70	0.77	0.80	0.71	**0.48**

Table: The query response time for 10^6 queries. HashàlaFKS in seconds, others as ratio to that of HashàlaFKS.

Random data (to see the trend with increasing Z)

Query response time (10^6 queries) on the $\mathcal{R}(d = 4, s = 10^6, Z)$ family.

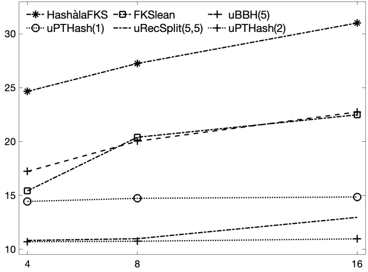
The same ranking of the methods: FKSlean is the fastest.



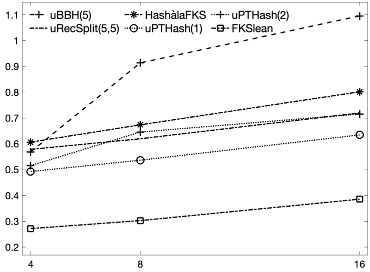
The query response time of all methods increases with Z.

Random data (to see the trend with increasing d)

The construction and query response time (10^6 queries) on $\mathcal{R}(d, s, Z)$ where $d = \{4, 8, 16\}$, $s = 10^6$, and $Z = 2 \times 10^7$.



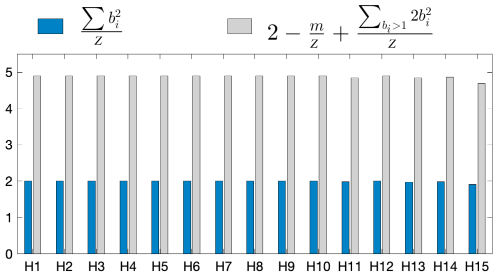
(a) Construction time



(b) Query response time

The same ranking of the methods (FKSlean middle of the pack for construction, and the fastest for query response).

Space utilization of FKSlean



- m: # empty buckets: $0.36 \leq m \leq 0.37$.
- $\sum b_i^2$ observed to be ≤ 2.
- The total space requirement $2Z - m + \sum_{i,b_i>1} 2b_i^2 \leq 5Z$.
- Maximum value of a bucket size is 13 in all experiments.
- The number of d-tuples in K is between $0.36 \log_2 Z$ and $0.42 \log_2 Z$.

Summary of comparisons

Construction time: the proposed method is in the middle. uRecSplit-(5,5) and uPTHash-(2) have the shortest construction time.

Query response time: The proposed FKSlean has the shortest time.

In the tensor application: FKSlean becomes the method of choice, especially for large d, Z, or the number of queries q.

One aspect we did not look (while MPHFs strive to reduce): the bits-per-key complexity (storage of hash functions). FKSlean is much worse than others, $\Omega(\log_2 \log_2 Z)$ vs around 3 bits for others.

Concluding remarks

FKSlean: a perfect hashing method with provable space bounds.

Experimental results: less than $5Z$ plus an additional $O(d \log_2 Z)$ term for storing the shared hash functions.

Comparisons with recent MPHFs: FKSlean has the shortest query response time among all alternatives while having a construction time in the middle of the pack.

Future work:

- The dynamic case where nonzeros come and go.
- Tighter analysis of the memory requirements.

Further information

Thank you for your attention.

Technical report and codes available:
https://hal.inria.fr/hal-03127673

More information: http://perso.ens-lyon.fr/bora.ucar

References I

- J. Bertrand, F. Dufossé, Bora Uçar. Algorithms and data structures for hyperedge queries. [Research Report] RR-9390, Inria Grenoble Rhône-Alpes. 2021.
- T. A. Davis and Y. Hu. 2011. The University of Florida sparse matrix collection. ACM Trans. Math. Software 38, 1 (2011), pp. 1:1–1:25.
- E. Esposito, T. Mueller Graf, and S. Vigna, RecSplit: Minimal Perfect Hashing via Recursive Splitting, in ALENEX) 2020, pp. 175–185.
- L. Fredman, J. Komlós, and E. Szemerédi. Storing a Sparse Table with $O(1)$ Worst Case Access Time, J. ACM 31, 3 (1984), pp. 538–544.
- T. G. Kolda and D. Hong, Stochastic Gradients for Large-Scale Tensor Decomposition, SIAM Journal on Mathematics of Data Science 2 (2020), pp. 1066–1095.
- A. Limasset, G. Rizk, R. Chikhi, andP. Peterlongo. Fast and Scalable Minimal Perfect Hashing for Massive Key Sets, in SEA 2017, pp. 25:1–25:16.

References II

G. E. Pibiri and R. Trani. 2021. PTHash: Revisiting FCH Minimal Perfect Hashing, In 44th SIGIR, International Conference on Research and Development in Information Retrieval (to appear), 2021.

S. Smith, J. W. Choi, J. Li, R. Vuduc, J. Park, X. Liu, and G. Karypis. 2017. FROSTT: The Formidable Repository of Open Sparse Tensors and Tools. Available at http://frostt.io/.

Dataset

name	d	size in each dimension	Z
kmer_A2a	2	170,728,175 ×170,728,175	360,585,172
queen_4147	2	4,147,110 × 4,147,110	329,499,288
com-Orkut	2	3,072,441 × 3,072,441	234,370,166
nell-1	3	2,902,330 × 2,143,368 × 25,495,389	143,599,552
delicious-3d	3	532,924 × 17,262,471 × 2,480,308	140,126,181
delicious-4d	4	532,924 × 17,262,471 × 2,480,308 × 1,443	140,126,181
flickr-3d	3	319,686 × 28,153,045 × 1,607,191	112,890,310
flickr-4d	4	319,686 × 28,153,045 × 1,607,191 × 731	112,890,310
nell-2	3	12,092 × 9,184 × 28,818	76,879,419
enron	4	6,066 × 5,699 × 244,268 × 1,176	54,202,099
vast-2015-mc1-3d	3	165,427 ×11,374 × 2	26,021,854
vast-2015-mc1-5d	5	165,427 × 11,374 × 2 × 100 × 89	26,021,945
chicago_crime	4	6,186 × 24 × 77 × 32	5,330,673
uber	4	183 ×24 × 1,140 × 1,717	3,309,490
lbnl-network	5	1,605 × 4,198 × 1,631 × 4,209 × 868,131	1,698,825

Table: Real-life test data used in the experiments.