智能网络与优化实验室
Intelligent Network and Optimization Laboratory, Renmin University

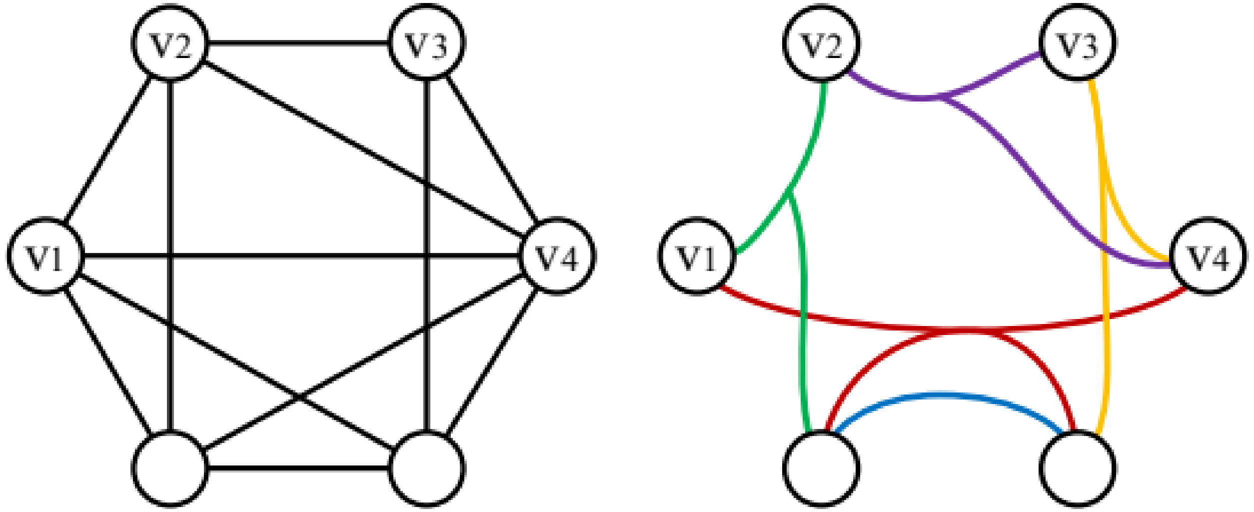
Hypercore Maintenance in Dynamic Hypergraphs

Author: Qi Luo, Dongxiao Yu, Zhipeng Cai

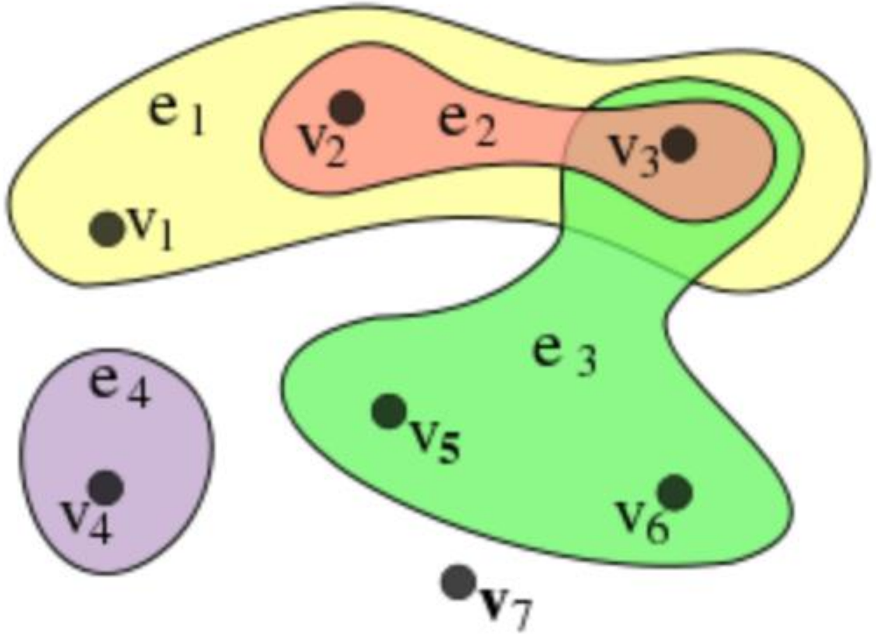
2021 IEEE 37th International Conference on Data Engineering (ICDE)

Xiaojia Xu

Background



(a) pairwise graph (b) hypergraph



Hypergraphs in various domains

- social networks
- Knowledge graphs
- VLSI
- ecommerce

[2] D. Yang, B. Qu, J. Yang, and P. Cudré-Mauroux, "Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach," in *The World Wide Web Conference, WWW*. ACM, 2019, pp. 2147–2157.

[3] Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He, "Heterogeneous hypergraph embedding for document recommendation," *Neurocomputing*, vol. 216, pp. 150–162, 2016.

[4] B. Fatemi, P. Taslakian, D. Vázquez, and D. Poole, "Knowledge hypergraphs: Extending knowledge graphs beyond binary relations," *CoRR*, vol. abs/1906.00137, 2019.

[5] T. Hwang, Z. Tian, R. Kuang, and J. A. Kocher, "Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction," in *Proceedings of the 8th IEEE International Conference on Data Mining ICDM*, 2008, pp. 293–302.

[6] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in VLSI domain," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[7] Q. Liu, Y. Huang, and D. N. Metaxas, "Hypergraph with sampling for image retrieval," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2255–2262, 2011.

[8] J. Li, J. He, and Y. Zhu, "E-tail product return prediction via hypergraph-based local graph cut," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining, KDD*, Y. Guo and F. Farooq, Eds. ACM, 2018, pp. 519–527.

- recommendation
- bioinformatics
- multimedia

Learning tasks based on hypergraphs

- clustering
- classification
- hyperedge prediction

[9] J. Huang, R. Zhang, and J. X. Yu, "Scalable hypergraph learning and processing," in *2015 IEEE International Conference on Data Mining, ICDM*. IEEE Computer Society, 2015, pp. 775–780.
[10] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*. MIT Press, 2006, pp. 1601–1608.
[11] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *The Thirty-First Innovative Applications of Artificial Intelligence Conference*. AAAI Press, 2019, pp. 3558–3565.
[12] D. Arya and M. Worring, "Exploiting relational information in social networks using geometric deep learning on hypergraphs," in *Proceedings of the International Conference on Multimedia Retrieval, ICMR*. ACM, 2018, pp. 117–125.
[13] D. Li, Z. Xu, S. Li, and X. Sun, "Link prediction in social networks based on hypergraph," in *22nd International World Wide Web Conference, WWW Companion Volume*. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 41–42.

Related cohesive subgraphs

- quasi-clique
- k-truss
- k-core

[14] J. Abello, M. G. C. Resende, and S. Sudarsky, "Massive quasi-clique detection," in *5th Latin American Symposium of Theoretical Informatics Proceedings,LATIN*, ser. Lecture Notes in Computer Science, vol. 2286. Cancun, Mexico: Springer, 2002, pp. 598–612.

[15] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.

[16] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," in *National Security Agency Technical report*, vol. 16, 2008, pp. 3–29.

Corresponding indexes for reflexing cohesiveness

- **coreness and trussness**

[17] Q. Luo, D. Yu, F. Li, Z. Dou, Z. Cai, J. Yu, and X. Cheng, "Distributed core decomposition in probabilistic graphs," in *8th International Conference Computational Data and Social Networks Proceedings*, ser. Lecture Notes in Computer Science, vol. 11917. Ho Chi Minh City, Vietnam: Springer, 2019, pp. 16–32.

[18] Q. Hua, Y. Shi, D. Yu, H. Jin, J. Yu, Z. Cai, X. Cheng, and H. Chen, "Faster parallel core maintenance algorithms in dynamic graphs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1287–1300, 2020.

[19] H. Jin, N. Wang, D. Yu, Q. Hua, X. Shi, and X. Xie, "Core maintenance in dynamic graphs: A parallel approach based on matching," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2416–2428, 2018.

[20] N. Wang, D. Yu, H. Jin, C. Qian, X. Xie, and Q. Hua, "Parallel algorithm for core maintenance in dynamic graphs," in *37th IEEE International Conference on Distributed Computing Systems, ICDCS*, K. Lee and L. Liu, Eds. IEEE Computer Society, 2017, pp. 2366–2371.

Coreness: the most commonly adopted index

- the coreness of vertices

- low computation cost [21]
- effectiveness in studying the properties of networks [22]
- large-scale network fingerprinting and visualization [23].

[21] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," *CoRR*, vol. cs.DS/0310049, 2003.

[22] A. Das, M. Svendsen, and S. Tirthapura, "Incremental maintenance of maximal cliques in a dynamic graph," *VLDB J.*, vol. 28, no. 3, pp. 351–375, 2019.

[23] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, "Large scale networks fingerprinting and visualization using the k-core decomposition," in *Neural Information Processing Systems*, 2005, pp. 41–50.

The core notation in hypergraph

- The core notation in hypergraph was firstly proposed in [24].
- [25] proposed a matching strategy based on the hypercore property of vertices in the coarsening phase.
- [26] showed how to obtain the empty k-core for r-uniform hypergraphs with a high probability
- [27] presented an efficient parallel core decomposition algorithm in hypergraphs.

[24] M. Leng, L. Sun, J. Bian, and Y. Ma, "An $o(m)$ algorithm for cores decomposition of undirected hypergraph," *Journal of Chinese Computer Systems*, vol. 34, no. 11, pp. 2568–2573, 2013.
[25] M. Leng and L. Sun, "Comparative experiment of the core property of weighted hyper-graph based on the ispd98 benchmark," *Journal of Information and Computational ence*, vol. 10, no. 8, pp. 2279–2290, 2013.
[26] J. Jiang, M. Mitzenmacher, and J. Thaler, "Parallel peeling algorithms," *ACM Trans. Parallel Comput.*, vol. 3, no. 1, pp. 7:1–7:27, 2016.
[27] J. Shun, "Practical parallel hypergraph algorithms," in *PPoPP '20: 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, 2020, pp. 232–249.

Fundamentals

The challenges of exact hypercore maintenance in hypergraphs include:

- (1) how to determine the hypercore number change after a graph change;
- (2) how to reduce the number of hyperedges and vertices traversed in the process of identifying those hyperedges and vertices that change the hypercore number;
- (3) how to finally identify the vertices and hyperedges whose hypercore numbers will change.

Fundamentals

- 1) We propose the concept of hypercore number on hyperedges, and reveal the relationship of the hypercore number between vertices and hyperedges.
- 2) We present algorithms for exact hypercore maintenance in large-scale dynamic hypergraphs. Rigorous theoretical analysis ensures that our algorithm can update the hypercore numbers of vertices and hyperedges efficiently

Fundamentals

$G(V, E)$: a simple and undirected hypergraph

$E \subset 2^V$: a set of hyperedges

$e \in E$: a set of $|e|$ vertices that take interaction

$e(v)$: a hyperedge e that v belongs to

$E(v)$: the set of hyperedges that v belongs to

$N_G(v)$: the neighborhood of a vertex $v \in V$ consists of all vertices in the hyperedges that contain v

$$N_G(v) = \cup_{e \in E} e(v)$$

$d_G(v)$: the hyper-degree of a vertex v

Fundamentals

*Definition 1 (**k-hypercore**):* A k-hypercore is a connected maximal sub-hypergraph $H = (V', E')$ of $G = (V, E)$, such that $\forall v \in V'$, $d_H(v) \geq k$.

*Definition 2 (**Hypercore number of vertex**):* For a given vertex v in the hypergraph G, the hypercore number of vertex v, denoted as $coreV(v)$, equals to k if there exists a k-hypercore containing vertex v, but there is not any $(k+1)$-hypercore containing v.

Fundamentals

Definition 3 (***Hypercore number of hyperedge***)*:* For a given hyperedge e in the hypergraph G, the hypercore number of hyperedge e, denoted as $coreE(e)$, equals to k if the hypercore number of each vertex in e is not smaller than k.

$$coreE(e) = \min\{coreV(v) : v \in e\}. \tag{1}$$

$$coreV(v) = arg \max_{K \geq 0}\{|\{e : e \in E(v), coreE(e) \geq K\}| \geq K\}. \tag{2}$$

Fundamentals

Lemma 1: If a hyperedge e_0 is deleted from hypergraph $G = (V, E)$, then the hypercore number of every vertex v and every hyperedge e can decrease by at most 1.

Fundamentals

Lemma 1: If a hyperedge e_0 is deleted from hypergraph $G = (V, E)$, then the hypercore number of every vertex v and every hyperedge e can decrease by at most 1.

- **Prove this lemma by contradiction**

$G' = G \backslash \{e_0\}$

Assume that there exists a vertex v whose hypercore number decreases by more than 1.

$coreV\ (v)\ =\ k - x$ **after deleting hyperedge e_0, where $x\ \geq\ 2$**

H: **the k-hypercore containing v in G. Let $H_0 = H \backslash e_0$**

for each vertex $u\ \in H_0$,$d'_H(v) \geq\ k-1 \Rightarrow\ coreV\ (v)\ =\ k-1$

Fundamentals

Lemma 2: Let $G' = (V, E')$ denote the hypergraph obtained by inserting a hyperedge e_0 into the hypergraph $G = (V, E)$. Then the hypercore number of every vertex v and every hyperedge e in G can increase by at most 1.

Fundamentals

Lemma 2: Let $G' = (V, E')$ denote the hypergraph obtained by inserting a hyperedge e_0 into the hypergraph $G = (V, E)$. Then the hypercore number of every vertex v and every hyperedge e in G can increase by at most 1.

- **Prove this lemma by contradiction**

$coreV(v) = k$ **in** G

Assume that there exists a vertex v **whose hypercore number changes by more than 1.**

$coreV(v) = k + x$ **in** G'**, where** $x \geq 2$

After deleting e_0 **from** G'**, the hypercore number of** v **can decrease by at most 1 by Lemma 1.**

$coreV(v) \geq k + x - 1 > k$

Fundamentals

*Definition 4 (**Pre-core number**):* After inserting a hyperedge e_0 into hypergraph G, the *pre-core number* of e, denoted by $\overline{coreE}(e)$, is defined as $\overline{coreE}(e) = \min\{coreV(v) : v \in e_0\}$.

Lemma 3: If a hyperedge e_0 is inserted into $G = (V, E)$, the pre-core number will increase by at most 1.

Fundamentals

Lemma 4: If a hyperedge e_0 is inserted into $G = (V, E)$, for any vertex $v \in V$, v may increase its hypercore number only if $coreV(v) = \overline{coreE}(e_0)$.

Lemma 5: If a hyperedge e_0 is deleted from $G = (V, E)$, for any vertex $v \in V$, v may decrease its hypercore number only if $coreV(v) = coreE(e_0)$.

Lemma 6: If a hyperedge e_0 is inserted into $G = (V, E)$, for any hyperedge $e \in E$, e may increase its hypercore number only if $coreE(e) = \overline{coreE}(e_0)$.

Lemma 7: If a hyperedge e_0 is deleted from $G = (V, E)$, for any hyperedge $e \in E \setminus \{e_0\}$, e may decrease its hypercore number only if $coreE(e) = coreE(e_0)$.

Fundamentals

Theorem 1: If a hyperedge e_0 is inserted into hypergraph $G = (V, E)$, then only the vertices v and the hyperedges e, which satisfy $coreV(v) = \overline{coreE}(e_0)$, $coreE(e) = \overline{coreE}(e_0)$, and are reachable from e_0 via a path that consists of vertices and hyperedges with hypercore number equal to $\overline{coreE}(e_0)$, may increase the hypercore number.

Fundamentals

Theorem 2: If a hyperedge e_0 is deleted from hypergraph $G = (V, E)$, then only the vertices v and hyperedges e, which satisfy $coreV(v) = coreE(e_0)$, $coreE(e) = coreE(e_0)$, and are reachable from e_0 via a path that consists of vertices and hyperedges with hypercore number equal to $coreE(e_0)$, may decrease the hypercore number.

Fundamentals

*Definition 5 (**Support Degree**):* The *support degree* of a vertex v, denoted as $sup(v)$, is defined as the number of hyperedges containing v and satisfying $coreE(e) \geq coreV(v)$. Each hyperedge e with $coreE(e) \geq coreV(v)$ is a called a *support hyperedge* of v.

Fundamentals

Theorem 3:

1) After a hyperedge e_0 is inserted into $G = (V, E)$, where $v \in V$ and $sup(v) \le coreV(v)$, then $coreV(v)$ will not increase.
2) After a hyperedge e_0 is deleted from $G = (V, E)$, where $v \in V$ and $sup(v) < coreV(v)$, then $coreV(v)$ will decrease by 1.

Algorithms

Algorithm 1: Incremental hypercore maintenance

Input : $G = (V, E)$, coreV, coreE, e_0
Output: $coreV, coreE$

$G \leftarrow G \cup \{e_0\}$;
$k \leftarrow \min\{\text{coreV}(v) : v \in e_0\}$; `// pre-core of` e_0
coreE$(e_0) \leftarrow k$;
exclude $\leftarrow \emptyset$;
sup $\leftarrow$ ComputeSupport$(G, coreE, coreV, e_0)$;
while $\exists sup(v) \leq k$ **do**
    exclude.$add(v)$;
    foreach $e \in E(v)$ ***and*** $coreE(e) = k$ **do**
        foreach $u \in e$ **do**
            if $sup(u) \neq null$ ***and*** $u \notin$ exclude **then**
                $sup(u) \leftarrow sup(u) - 1$;
foreach v *that* $sup(v) \neq null$ ***and*** $u \notin$ exclude **do**
    foreach $e \in E(v)$ ***and*** $coreE(e) = k$ **do**
        Update coreE(e) by Equation 1;
    coreV$(v) \leftarrow k + 1$;
return coreV, coreE ;

Algorithms

Algorithm 2: ComputeSupport($G, coreV, coreE, e_0$)

Input : G, coreV, coreE, e_0
Output: sup

```
visit ← ∅ ;
sup ← ∅ ;
stack ← ∅ ;
k ← coreE(e0) ;
foreach v ∈ V do visit(v) ← false;
foreach v ∈ e0 and coreE(v) = k do
    stack.push(v);
    visit(v) ← true ;
while stack ≠ ∅ do
    v ← stack.pop() ;
    foreach e ∈ E(v) do
        if coreE(e) ≥ coreV(v) then
            sup(v) ← sup(v) == null?1 : sup(v) + 1 ;
        if coreE(e) = k then
            foreach u ∈ e do
                if visit(u) = false and coreV(u) = k then
                    stack.push(u) ;
                    visit(u) ← true ;
return sup;
```

Algorithms

Theorem 4: Algorithm 1 can correctly update the hypercore numbers of vertices and hyperedges in $O(|\hat{V}| \cdot d_{max} + |\hat{E}|s)$ time, after inserting a hyperedge e_0 into G.

s	**: the maximum cardinality of hyperedges**
C	**: the set of hypercore numbers of vertices in** G
V_K E_K	**: the set of vertices and hyperedges whose hypercore numbers equal to** k
$\hat{V}$	: $\hat{V} = max_{k \in C}\{V_K\}$
$\hat{E}$	: $\hat{E} = max_{k \in C}\{E_K\}$
d_{max}	**: the maximum degree of vertices**

Algorithms

Theorem 4: Algorithm 1 can correctly update the hypercore numbers of vertices and hyperedges in $O(|\hat{V}| \cdot d_{max} + |\hat{E}|s)$ time, after inserting a hyperedge e_0 into G.

For the running time, there are three processes in the algorithm. At first the hypercore number of each vertex in e_0 is computed, which takes $O(s)$ time. Then identifying the potential vertices using the DFS process takes $O(|\hat{V}| \cdot d_{max} + |\hat{E}|s)$ time, and distinguishing the vertices that will not increase the hypercore number takes $O(|\hat{V}| + |\hat{E}|s)$. Finally, it takes $O(|\hat{V}| + |\hat{E}|)$ time to update the hypercore numbers of vertices and hyperedges. Combining all together, the running time is $O(|\hat{V}| \cdot d_{max} + |\hat{E}|s)$. ■

Algorithms

Algorithm 3: Hypercore Decomposition

Input : A hypergraph $G = (V, E)$
Output: Hypercore number of vertices and hyperedges
compute $d(v)$ for $v \in V$;
$k \leftarrow 1$;
while G *is not empty* **do**
    while $\exists v \in V$ ***such that*** $d(v) \leq k$ **do**
        foreach $e \in E(v)$ **do**
            foreach $u \in e$ **do**
                $d(u) \leftarrow d(u) - 1$;
            delete e from E ;
            coreE$(e) \leftarrow k$;
        delete v from V;
        coreV$(v) \leftarrow k$;
    $k \leftarrow k + 1$;
return coreE, coreV;

Evaluation

TABLE I
THE STATISTICS OF REAL-WORLD HYPERGRAPHS.

Dataset	$\|V\|$	$\|E\|$	c_{max}	**Ins.**(ns)	**Del.**(ns)
tags-math	1K	174K	5	0.374	0.313
DAWN	2K	143K	16	0.657	0.448
tags-ask-ubuntu	3K	151K	5	0.254	0.247
NDC-substances	5K	10K	25	0.567	0.121
threads-ask-ubuntu	125K	167K	14	0.163	0.037
threads-math	176K	595K	21	0.383	0.131
coauth-History	1.1M	895K	25	47.101	0.336
coauth-Geology	1.2M	1.2M	25	78.826	9.945

TABLE II
THE STATISTICS OF TEMPORAL HYPERGRAPHS.

Dataset	$\|V\|$	**#TS.**	**#Uniq.**	**Ins.**	**Del.**
tags-stack-overflow	50K	14.4M	5.6M	45	142
coauth-DBLP	1.9M	3.7M	2.6M	1.5K	5.6K
threads-stack-overflow	2.6M	11.3M	9.7M	1.9K	8.5K

Conclusion

- This paper proposed the algorithms for exact hypercore maintenance in large-scale dynamic hypergraphs.
- Extensive experiments demonstrate our algorithms can significantly speed up the hypercore update process.
- Due to the ubiquitous applications of hypergraphs, studies on hypergraphs have attracted much attention recently. However, due to complex representations and lack of adequate tools, efficient solutions to some fundamental problems in hypergraphs are still not derived.
- Their work shows that it is possible to design efficient dense subgraph mining algorithms by deeply investigating the structural properties of hypergraphs. Hence, it deserves to making more efforts on mining tasks in hypergraphs.

THANK YOU

Xiaojia Xu