Data Mining and Knowledge Discovery (2023) 37:2389–2437
https://doi.org/10.1007/s10618-023-00956-2

Hypercore decomposition for non-fragile hyperedges: concepts, algorithms, observations, and applications

Fanchen Bu[1] · **Geon Lee**[2] · **Kijung Shin**[1,2]

Received: 8 February 2023 / Accepted: 6 July 2023 / Published online: 8 August 2023

Abstract

Hypergraphs are a powerful abstraction for modeling high-order relations, which are ubiquitous in many fields. A hypergraph consists of nodes and hyperedges (i.e., subsets of nodes); and there have been a number of attempts to extend the notion of k-cores, which proved useful with numerous applications for pairwise graphs, to hypergraphs. However, the previous extensions are based on an unrealistic assumption that hyperedges are *fragile*, i.e., a high-order relation becomes obsolete as soon as a single member leaves it.In this work, we propose a new substructure model, called (k,t)-hypercore, based on the assumption that high-order relations remain as long as at least t fraction of the members remains. Specifically, it is defined as the maximal subhypergraph where (1) every node is contained in at least k hyperedges in it and (2) at least t fraction of the nodes remain in every hyperedge. We first prove that, given t (or k), finding the (k,t)-hypercore for every possible k (or t) can be computed in time linear w.r.t the sum of the sizes of hyperedges. Then, we demonstrate that real-world hypergraphs from the same domain share similar (k,t) -hypercore structures, which capture different perspectives depending on t. Lastly, we show the successful applications of our model in identifying influential nodes, dense substructures, and vulnerability in hypergraphs.

Keywords Hypergraphs mining · **k**-cores · Cohesive substructure models · Real-world hypergraph analysis

Responsible editor: Charalampos Tsourakakis.

✉ Kijung Shin
kijungs@kaist.ac.kr

Fanchen Bu
boqvezen97@kaist.ac.kr

Geon Lee
geonlee0325@kaist.ac.kr

[1] School of Electrical Engineering, KAIST, Daejeon, South Korea

[2] Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea

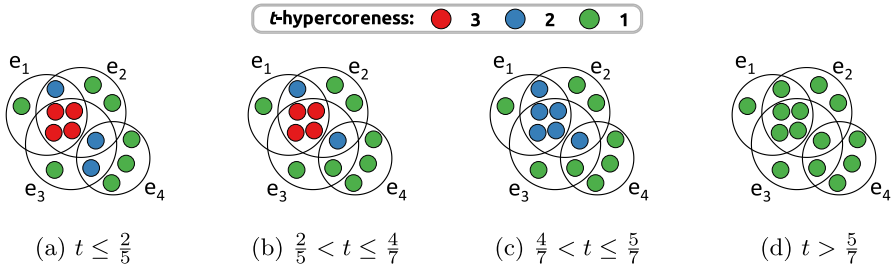
1 Introduction

Graphs are a powerful model for representing pairwise relations, and they have been used for recommendation systems (Silva et al. 2010; Debnath et al. 2008), information retrieval (Blanco and Lioma 2012; Mihalcea and Radev 2011), knowledge representation (Chein and Mugnier 2008), and many more. However, graphs are limited to pairwise relations and thus fail to precisely describe high-order (i.e., group-wise) relations among more than two nodes.

Hypergraphs, where each hyperedge consists of an arbitrary number of nodes, break the limitation by describing high-order relations precisely (Benson et al. 2018a; Yin et al. 2017) and contain graphs as special cases. Hypergraphs have been successful in modeling real-life processes in diverse fields, including chemical reactions (Konstantinova and Skorobogatov 2001), epidemic spread (Bodó et al. 2016), and blockchain economy (Qu et al. 2018).

For a given pairwise graph, the *k-core* (Seidman 1983) is a cohesive substructure that is defined as the maximal subgraph where each node has degree at least k (i.e., each node is incident to at least k edges) within it. Extensive research has been conducted to show its linear-time computability (Batagelj and Zaversnik 2003) and successful applications to k-cores, including graph visualization (Alvarez-Hamelin et al. 2006), community detection (Corominas–Murtra et al. 2014), anomaly detection (Shin et al. 2018a), and biological process modeling (Luo et al. 2009).

There have been attempts to generalize the notion of k-cores to hypergraphs (Hua et al. 2023; Luo et al. 2021, 2022; Gabert et al. 2021a, b; Sun et al. 2020), and the generalized notations, called *hypercores*, commonly assume that hyperedges are *fragile*. That is, a hyperedge (i.e., a group relation) becomes obsolete as soon as *any* constituent node opts out of it. Specifically, an entire hyperedge is ignored as soon as any node in it is removed during hypercore computation. However, such an assumption is unrealistic and potentially leads to much information loss. For example, an online group chat may remain active even if someone leaves it; and a recipe (i.e., a group of ingredients) may still produce a delicious result even if some ingredients are unavailable. As another example, the hypergraph shown in Fig. 1(d)



(a) $t \leq \frac{2}{5}$ (b) $\frac{2}{5} < t \leq \frac{4}{7}$ (c) $\frac{4}{7} < t \leq \frac{5}{7}$ (d) $t > \frac{5}{7}$

Fig. 1 An example of (k, t)-hypercores. Assuming more robust hyperedges (i.e., decreasing the hyperedge-fraction threshold t) reveals cohesive substructures that are overlooked when fragile hyperedges are assumed. Notably, when fragile hyperedges are assumed (i.e., when $t = 1$), every node has the same t-hypercoreness, as shown in (**d**)

cannot be decomposed into k-hypercores with different k, although the cohesiveness of subhypergraphs varies, since fragile hyperedges are assumed.

In order to better reveal the structural information in hypergraphs, we propose the notion of (k, t)-hypercores. In addition to the node-degree threshold k, we introduce the *hyperedge-fraction threshold* t that determines how many constituent nodes suffice to maintain a hyperedge. Specifically, given a hypergraph and thresholds k and t, the (k, t)-*hypercore* is defined as the maximal subhypergraph where (1) every node is contained in at least k hyperedges in it and (2) at least t fraction of the constituent nodes (i.e., the nodes constituting the original hyperedge) remain in every remaining hyperedge. The larger the value of t is, the more fragile the hyperedges are. Based on the concept, we define the *t-hypercoreness* of a node as the maximum k such that the node is in the (k, t)-hypercore, and *the k-fraction* of a node as the maximum t such that the node is in the (k, t)-hypercore. In Fig. 1, we show an example where the (k, t)-hypercore structures change with t. Notably, some other variants of hypercores have been considered. The concept of $(k;\ell)$-hypercores has been considered by Limnios et al. (2021), where the $(k;\ell)$-hypercore requires that at least ℓ constituent nodes (instead of t fraction required in the (k, t)-hypercore) remain in every remaining hyperedge. The concept of neighbor-k-hypercores has been considered by Arafat et al. (2023), which focuses on the number of neighbors (i.e., nodes coexisting in at least one hyperedge) of each node, and the concept is further extended to (neighbor, degree)-(k, d)-hypercores. Compared to the existing concepts, our proposed concepts provide unique information on hypergraphs, as theoretically proven and empirically demonstrated.

We first show that the proposed concepts are well-defined and have containment properties w.r.t both k and t, then propose peeling-like computation algorithms for computing all the proposed concepts and show their correctness and time complexity. In particular, we show that both the t-hypercoreness for given k and the k-fraction for given t of each node can be computed in time proportional to the sum of the sizes of hyperedges.

In order to demonstrate the usefulness of the proposed concepts, we investigate the (k, t)-hypercore structures of fourteen real-world hypergraphs in six different domains (Sinha et al. 2015; Mastrandrea et al. 2015; Leskovec et al. 2007) while varying t. The examination leads to the following observations from different perspectives:(1) *domain-based patterns of* (k, t)*-hypercore sizes*: hypergraphs in the same domain show similar patterns of the (k, t)-hypercore sizes with different k and t values; (2) *heavy-tailed distributions of t-hypercoreness*: in most investigated real-world hypergraphs, the t-hypercoreness of nodes consistently follows heavy-tailed distributions regardless of t; (3) *heterogeneity of t-hypercoreness*: in the same real-world hypergraph, the t-hypercoreness with different t provides statistically and information-theoretically distinct information.

We also utilize some properties of the proposed concepts in three applications: (1) *influential-node identification*: we generalize the SIR model in hypergraphs and use the model to show that t-hypercoreness is a reliable indicator to node-influence; (2) *dense substructure discovery*: we show that (k, t)-hypercores generally have much higher density than the whole hypergraph and consider a generalized vertex cover problem to demonstrate that t-hypercoreness can be used to find dense substructures;

(3) *vulnerability detection*: we generalize the core minimization problem to detect vulnerabilities in hypergraphs by finding the nodes whose removal reduces the size of the (k, t)-hypercore (for given k and t) most, and to this end, we propose an efficient and effective algorithm.

In short, our contributions are three-fold:

- *New concepts.* We propose the (k, t)-hypercore, a new substructure model for hypergraphs, together with t-hypercoreness and k-fraction (Defs. 5 to 7). In addition to the node-degree threshold k, the proposed concepts incorporate the hyperedge-fraction threshold t to provide more comprehensive information.
- *Properties and algorithms.* We show some theoretical properties of the proposed concepts, and computation algorithms (Algs. 1 to 3) for the proposed concepts with analyses of the correctness and time complexity (Thorems. 1 to 3).
- *Observations and applications.* We investigate 14 real-world hypergraphs, which leads to interesting observations (Sect. 5), including a surprising similarity in the (k, t)-hypercores of hypergraphs in the same domain. We also show successful applications (Sect. 6) of the proposed concepts to influence estimation, dense-substructure detection, and vulnerability detection.

Reproducibility The code and datasets are available at https://github.com/bokveizen/non-fragile-hypercore (Bu et al. 2023).

2 Preliminaries

In this section, we provide the mathematical background and preliminaries that are used throughout this paper.

Hypergraphs. A *hypergraph* $H = (V, E)$ consists of a node set V and a hyperedge multiset E.[1] Given a hypergraph $H = (V, E)$, we associate each hyperedge with a distinct positive integer in $\mathbb{N}$,i.e., $E = \{e_i : i \in I_E\}$, where I_E is called the *index set* of E. The *degree* $d(v; H)$ of a node v is the number of hyperedges that contain v, i.e., $d(v;H) = |\{i \in I_E : v \in e_i\}|$. The set $N(v; H)$ of *neighbors* of a node v is the number of nodes coexisting with v in at least one hyperedge, i.e., $N(v;H) = \{u \in V : u \neq v, \exists e \in E \text{ s.t. } u, v \in e\}$. The *constituent nodes* of a hyperedge $e \in E$, is the nodes in e. The *size* of a hyperedge $e \in E$, denoted by $|E|$, is the cardinality of E (i.e., the number of constituent nodes of e). The *size* of H, denoted by $|H|$, is the number of nodes in H, i.e., $|H| = |V|$. The *total size* of H, denoted by $TS(H)$, is the sum of the size of each hyperedge in H (i.e., $TS(H) = \sum_{i \in I_E} |e_i|$). All hypergraphs in this paper are finite, undirected, and unweighted; and in them, each node has degree at least 1, i.e., $d(v;H) \geq 1, \forall v \in V$, and each hyperedge is of cardinality at least two, i.e., $|e_i| \geq 2, \forall i \in I_E$. If in a hypergraph $H = (V, E)$, each hyperedge is of cardinality exactly two, i.e., $|e_i| = 2, \forall i \in I_E$, then H is also called a (pairwise) graph.

[1] A multiset is a set al.lowing duplicate elements.

Table 1 Notations

Notation	Definition
$H = (V, E)$	A hypergraph with nodes V and hyperedges E
$d(v; H)$	The degree of v in H
I_E	The index set of E
k, t	The degree and hyperedge-fraction thresholds
$C_{k,t}(H)$	The (k, t)-hypercore of H
$c_t(v;H), c_t^*(H)$	The t-hypercoreness of v in H, and that of H
$f_k(v;H), f_k^*(H)$	The k-fraction of v in H, and that of H

Definition 1 (Subhypergraph) A hypergraph $H' = (V', E')$ is a **subhypergraph** of $H = (V, E)$ if each hyperedge in H' is a subset of the hyperedge with the same index in H, i.e., $e'_i \subseteq e_i, \forall i \in I_{E'} \subseteq I_E$. If $e'_i = e_i, \forall i \in I_{E'}$, we call H' a **complete subhypergraph** of H.

Note that a subhypergraph should be a hypergraph, and thus each hyperedge in a subhypergraph should also be of cardinality at least two.

We summarize the notations in Table 1. In the notations, the input hypergraph H may be omitted when the context is clear.

Hypercores. In pairwise graphs, the concept of k-cores (Seidman 1983) is widely used. Given a pairwise graph G and $k \in \mathbb{N}$, the *k-core* of G is the maximal subgraph where each node has degree at least k within it.

Definition 2 (k-core) Given a pairwise graph $G = (V, E)$ and $k \in \mathbb{N}$, the ***k*-core** of H, denoted by $C_k(G) = (V', E')$, is the maximal subgraph of G where each node has degree at least k (i.e., is incident to at least k edges) within C_k.[2]

It is naturally generalized to hypergraphs (Hua et al. 2023; Luo et al. 2021, 2022; Gabert et al. 2021a, b; Sun et al. 2020), as follows.

Definition 3 (k-hypercore) Given a hypergraph $H = (V, E)$ and $k \in \mathbb{N}$, the ***k*-hypercore** of H, denoted by $C_k(H) = (V', E')$, is the maximal *complete subhypergraph* of H where each node has degree at least k (i.e., is contained in at least k hyperedges) within C_k.

Some variants of hypercores have been considered. See Sect. 3 for some related discussions.

Clique expansion. One of the most common ways to convert hypergraphs into pairwise graphs is the clique expansion, where each hyperedge $e \in E$ is converted to a clique consisting of the nodes in e. Given a hypergraph $H = (V, E)$, its unweighted

[2] In this work, the maximal subgraph (subhypergraph) satisfying some conditions means that every other graph (hypergraph) satisfying such conditions is a subgraph (subhypergraph) of the maximal one.

clique expansion is $G_{\text{uc}}(H) = (V, \mathcal{E})$, and its weighted clique expansion is $G_{\text{wc}}(H) = (V, \mathcal{E}, \omega)$, where the edge set $\mathcal{E} = \{(u, v) \in \binom{V}{2} : \exists e \in E \; s.t. \; \{u, v\} \subseteq e\}$, and the weight function $\omega((u, v)) = |\{i \in I_E : \{u, v\} \subseteq e_i\}|$. Clique expansion provides an approach to make the hypergraphs easier to analyze, but the information on the higher-order interactions is lost, which is natural since for a set of nodes V, there are $O(|V|^2)$ possible pairs in V, while there are $O(2^{|V|})$ possible subsets. Two hypergraphs with obviously different structures may have the same clique expansions.

Star expansion. Each hypergraph can be represented as a bipartite graph, which is called its star expansion (Zien et al. 1999). The star expansion of a hypergraph H is the bipartite graph whose node set is the union of V and E and whose edge set consists of the incidence relations in H.

Definition 4 (Star expansion) Given a hypergraph $H = (V, E)$, its **star expansion** (i.e., bipartite-graph representation) is $G_{se}(H) = (V \cup E, E_{se}(H))$, where $E_{se}(H) = \{(v, e) : v \in V, e \in E, v \in e\} \subseteq V \times E$.[3]

As Yang et al. (2022) pointed out, although a star expansion contains all the incidence information in hypergraphs, the remaining heterogeneous structure has no explicit edges between nodes and is unsuitable for many well-studied graph algorithms designed for simple homogeneous graphs.

3 Concepts

In this section, we introduce the proposed concepts and show some theoretical properties of them. Moreover, we discuss the connections and differences between the proposed concepts and some existing related concepts.

3.1 Proposed concepts

In pairwise graphs, each edge represents a connection between two nodes, and thus the removal of either node naturally results in the complete nullification of the edge. In contrast, a hyperedge with three or more nodes still represents the interactions among the remaining nodes even when some constituent nodes are removed. As we have discussed and shown in Fig. 1, the straightforward generalization in Definition 3 groundlessly assumes fragile hyperedges and suffers from information loss. We seek to better reveal the structure of hypergraphs by considering *non-fragile hyperedges*.

[3] Similar to clique expansion, we can also have weighted star expansion, which is, however, not used in this work.

Therefore, we introduce the hyperedge-fraction threshold t that determines the minimum proportion of constituent nodes required to maintain a hyperedge, which leads to Definition 5.

Definition 5 ($(k,\ t)$-hypercore) Given $H = (V, E)$, $k \in \mathbb{N}$, and $t \in [0, 1]$, the **(k,t)-hypercore** of H, denoted by $C_{k,t}(H) = (V', E')$, is the maximal (in terms of total size) subhypergraph of H where (1) every node in has degree at least k (i.e., is contained in at least k hyperedges) within $C_{k,t}$ and (2) at least t proportion of the constituent nodes remain in every hyperedge of $C_{k,t}(H)$. Formally, $d(v;C_{k,t}(H)) \geq k, \forall v \in V'$ and $|e'_i \cap e_i| \geq t|e_i|, \forall i \in I_{E'} \subseteq I_E$.

Note that the definition of $(k,\ t)$-hypercore requires that at least two nodes remain in each hyperedge because of the definition of subhypergraphs (see Sect. 2). See also Line 6 in Algorithm 1.

Definition 6 (t-hypercoreness) Given $H = (V, E)$ and $t \in [0, 1]$, the ***t*-hypercoreness** of $v \in V$, denoted by $c_t(v;H)$, is the maximum positive integer such that v is in the $(c_t(v), t)$-hypercore, i.e., $c_t(v) = \max\{k \in \mathbb{N} : v \in V(C_{k,t})\}$. We call $c^*_t(H) := \max\{c_t(v) : v \in V\}$ the t-hypercoreness of H.

Definition 7 (k-fraction) Given $H = (V, E)$ and $k \in \mathbb{N}$, the ***k*-fraction** of $v \in V$, denoted by $f_k(v;H)$, is the maximum real number in $[0,\ 1]$ such that v is in the $(k, f_k(v))$-hypercore, i.e., $f_k(v) = \max\{t \in [0, 1] : v \in V(C_{k,t})\}$. For the completeness of definition, if $\{t \in [0, 1] : v \in V(C_{k,t})\} = \emptyset$, we let $f_k(v;H) = -1$. We call $f^*_k(H) := \max\{f_k(v) : v \in V\}$ the k-fraction of H.

Note that the proposed concepts are extendable to weighted hypergraphs. Specifically, as long as we have rigorous definitions of node degrees and hyperedge fractions on weighted hypergraphs, the extensions are straightforward.

Example. In Fig. 1, the t-hypercoreness of each node changes when the t changes. Specifically, four nodes have t-hypercoreness 3 when $t \leq \frac{4}{7}$. They have t-hypercoreness 2 when $\frac{4}{7} < t \leq \frac{5}{7}$, and have t-hypercoreness 1 when $t > \frac{5}{7}$, which means that their 3-fraction is $\frac{4}{7}$ and 2-fraction is $\frac{5}{7}$.

The following propositions show that the $(k,\ t)$-hypercores are well-defined and have two-way containment properties.

Proposition 1 (Existence and uniqueness) *Given any hypergraph H, $k \in \mathbb{N}$, and $t \in [0, 1]$, $C_{k,t}$ uniquely exists and is possibly empty.*

Proof See Appendix A.1. □

Proposition 2 (*Two-way containment*) *Let H be any hypergraph. Fix any $k \in \mathbb{N}$, for any $0 \leq t_1 < t_2 \leq 1$, $C_{k,t_2}(H)$ is a subhypergraph of $C_{k,t_1}(H)$. Similarly, fix any $t \in [0, 1]$, for any $k_1 < k_2 \in \mathbb{N}$, $C_{k_2,t}(H)$ is a subhypergraph of $C_{k_1,t}(H)$.*

Proof See Appendix A.2. □

3.2 Related concepts

Below, we discuss some existing related concepts, especially the connections and differences between them and our proposed concepts.

Existing variants of hypercores. As mentioned in Sect. 2 (see Definition 3), most previous works (Hua et al. 2023; Luo et al. 2021, 2022; Gabert et al. 2021a, b; Sun et al. 2020) are based on the straightforward generalization of k-cores to hypergraphs assuming fragile hyperedges (i.e., a hyperedge is removed when *any* node leaves it), which is equivalent to the (k, t)-hypercore with $t = 1$ (i.e., a special case of (k, t)-hypercore). Limnios et al. (2021) defined the $(k;\ell)$-hypercore of a given hypergraph H as the maximal subhypergraph of H where each node has degree at least k (i.e., is contained in at least k hyperedges) within the subhypergraph and each hyperedge contains at least ℓ nodes. Based on the concept, we can define ℓ-hypercoreness.

Definition 8 ($(k;\ell)$-hypercore) Given a hypergraph $H = (V, E)$ and $k, \ell \in \mathbb{N}$, the **$(k;\ell)$-hypercore** of H, denoted by $\tilde{C}_{k;\ell}(H)$, is the maximal subhypergraph of H such that each node has degree at least k (i.e., is contained in at least k hyperedges) in within $\tilde{C}_{k;\ell}(H)$ and each hyperedge contains at least ℓ nodes.

Definition 9 (ℓ-hypercoreness) Given $H = (V, E)$ and $\ell \in \mathbb{N}$, the **ℓ-hypercoreness** of $v \in V$, denoted by $\tilde{c}_\ell(v)$, is the maximum positive integer such that v is in the $(\tilde{c}_\ell(v);\ell)$-hypercore, i.e., $\tilde{c}_\ell(v) = \max\{k \in \mathbb{N} : v \in V(\tilde{C}_{k;\ell})\}$.

The concept of $(k;\ell)$-hypercores is equivalent to a k-core-like concept on bipartite graphs called (α, β)-cores (Liu et al. 2020; Sarıyüce and Pinar 2018).

Definition 10 ($(\alpha;\beta)$-core) Given a bipartite graph $G_B = (V_1 \cup V_2, E)$ and $\alpha, \beta \in \mathbb{N}$, the **$(\alpha;\beta)$-core** of G_B, denoted by $\hat{C}_{\alpha;\beta}(G_B) = (V_1' \cup V_2', E')$ where $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$, is the maximal subgraph of G_B such that each node in V_1' has degree at least α within $\hat{C}_{\alpha;\beta}(G_B)$, and each node in V_2' has degree at least β within $\hat{C}_{\alpha;\beta}(G_B)$.

Lemma 1 *Given H, k, and ℓ, the $(k;\ell)$-hypercore of H is equivalent to the $(\alpha = k, \beta = \ell)$-core of $G_{bp}(H)$, the star expansion of H (see Sect. 2).*

Proof See Appendix A.3. □

Notable, only the special case with $\ell = 2$ was actually used by Limnios et al. (2021), and such a special case (i.e., $(k;\ell = 2)$-hypercore) was also previously considered by Vogiatzis (2013). Also, $(k;\ell = 2)$-hypercore is equivalent to the proposed (k, t)-hypercore with $t = 0$.

Lemma 2 *Given a hypergraph $H = (V, E)$ and $k \in \mathbb{N}$, $\tilde{C}_{k;\ell=2}(H) = C_{k;t=0}(H)$.*

Proof See Appendix A.4. □

Essential differences exist between the concept of $(k;\ell)$-hypercores and the concept of (k, t)-hypercores proposed by us. In Appendix A.5, we theoretically analyze the limitations of the $(k;\ell)$-hypercores and the superiority of the proposed (k, t)-hypercores with empirical comparisons. For example, Lemma 3 below tells us that the proposed concept of (k, t)-hypercores can provide unique information of a hypergraph, which is not contained in the existing concept of $(k;\ell)$-hypercores for any ℓ.

Lemma 3 *There exist H, k, t such that $C_{k,t}(H) \neq \tilde{C}_{k;\ell}(H)$ for any ℓ.*

Proof See Appendix A.5. □

Recently, Arafat et al. (2023) proposed a variant of hypercores, where for each node, the number of neighbors (i.e., nodes coexisting in at least one hyperedge) of this node (instead of the degree of this node) is considered, which leads to the concept of neighbor-k-hypercores. Based on the concept, we can define neighbor-hypercoreness.

Definition 11 (neighbor-k-hypercores) Given a hypergraph $H = (V, E)$, and $k \in \mathbb{N}$, the **neighbor-k -hypercore** of H, denoted by $C_k^{nbr}(H)$, is the maximal complete subhypergraph of H such that each node in $C_k^{nbr}(H)$ has at least k neighbors (i.e., $|N(v;C_k^{nbr}(H))| \geq k, \forall v \in C_k^{nbr}(H)$).

Definition 12 (neighbor-hypercoreness) Given $H = (V, E)$, the **neighbor-hypercoreness** of $v \in V$, denoted by $c^{nbr}(v)$, is the maximum positive integer such that v is in the neighbor-$c^{nbr}(v)$-hypercore, i.e., $c^{nbr}(v) = \max\{k \in \mathbb{N} : v \in V(C_k^{nbr})\}$.

Arafat et al. (2023) further extended the concept of neighbor-k-hypercores by incorporating the information of the degree of each node, which leads to the concept of (neighbor, degree)-(k, d)-hypercores.

Definition 13 ((neighbor, degree)-(k, d)-hypercores) Given a hypergraph $H = (V, E)$, and $k, d \in \mathbb{N}$, the **(neighbor, degree)-(k, d) -hypercore** of H, denote by $C_{k,d}^{nd}(H)$, is the maximal complete subhypergraph of H such that each node in $C_{k,d}^{nd}(H)$ has at least k neighbors and has degree at least d (i.e., $|N(v;C_{k,d}^{nd}(H))| \geq k \wedge d(v;C_{k,d}^{nd}(H)) \geq d, \forall v \in C_{k,d}^{nd}(H)$).

Since two parameters are involved, we can have multiple ways to define the hypercoreness w.r.t (neighbor, degree)-(k, d)-hypercores, and an intuitive and straightforward way is as follows.

Definition 14 (neighbor-degree-hypercoreness) Given $H = (V, E)$, the **neighbor-degree-hypercoreness** of $v \in V$, denoted by $c^{nd}(v)$, is the maximum positive integer such that v is in the neighbor-degree-$(c^{nd}(v), c^{nd}(v))$-hypercore, i.e., $c^{nd}(v) = \max\{k \in \mathbb{N} : v \in V(C_{k,k}^{nd})\}$.

Notably, the above two concepts consider only complete subhypergraphs, i.e., they still assume fragile hyperedges.

Simplicial complexes. Another way to take the subsets of hyperedges into consideration is to use simplicial complexes (Torres et al. 2021). For example, Preti et al. (2021) considered the computation of k-trusses in simplicial complexes. Similar to clique expansion, converting hypergraphs into simplicial complexes also brings information loss. Our work shows that considering the subsets of relations is meaningful also when the data is modeled as hypergraphs; when the data is modeled as hypergraphs, considering the subsets of the relations is also meaningful; and we provide a way to do so.

Algorithm 1 (k,t)-Hypercore

Input: $H=(V,E)$, k, t, and original hyperedge sizes $\mathcal{D}$
Output: $C_{k,t}(H)$: the (k,t)-hypercore of H

$\mathcal{R} \leftarrow \{v \in V : d(v;\ H) < k\}$ ▷ Nodes to remove
while $\mathcal{R} \neq \varnothing$ **do**
  $\mathcal{R}' \leftarrow \varnothing$ ▷ Nodes to remove in next round
  for each $e_i \in E$ **s.t.** $e_i \cap \mathcal{R} \neq \varnothing$ **do**
    $e_i \leftarrow e_i \setminus \mathcal{R}$ ▷ Remove nodes
    if $|e_i| < t\mathcal{D}(i)$ **or** $|e_i| < 2$ **then**
      $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{v \in e_i : d(v;\ H) = k\}$
      $E \leftarrow E \setminus \{e_i\}$ ▷ Remove hyperedge
    end if
  end for
  $V \leftarrow V \setminus \mathcal{R}$
  $\mathcal{R} \leftarrow \mathcal{R}'$
end while
return H

4 Computation algorithms

In this section, we provide the computation algorithms of the proposed concepts: (k, t)-hypercore, t-hypercoreness, and k-fraction. We also show their correctness and time complexity.

Algorithm 2 t-Hypercoreness

Input: $H = (V, E)$ and t
Output: t-hypercoreness $c_t(v)$ for each node $v \in V$

1: $\mathcal{R} \leftarrow \varnothing$
2: **while** $H \neq \varnothing$ **do**
3: **if** $\mathcal{R} = \varnothing$ **then**
4: $k \leftarrow \min_{v \in V} d(v;\ H) + 1$
5: $\mathcal{R} \leftarrow \{v \in V : d(v;\ H) = k - 1\}$
6: **else**
7: $c_t(v) \leftarrow k - 1, \forall v \in \mathcal{R}$
8: $\mathcal{R}' \leftarrow \varnothing$ ▷ Nodes to remove in next round
9: **for each** $e_i \in E$ **s.t.** $e_i \cap \mathcal{R} \neq \varnothing$ **do**
10: $e_i \leftarrow e_i \setminus \mathcal{R}$ ▷ Remove nodes
11: **if** $|e_i| < t\mathcal{D}(i)$ **or** $|e_i| < 2$ **then**
12: $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{v \in e_i : d(v;\ H) = k\}$
13: $E \leftarrow E \setminus \{e_i\}$ ▷ Remove hyperedge
14: **end if**
15: **end for**
16: $V \leftarrow V \setminus \mathcal{R}$
17: $\mathcal{R} \leftarrow \mathcal{R}'$
18: **end if**
19: **end while**
20: **return** $c_t(v)$ for each $v \in V$

4.1 Computation of (k, t)-hypercore

Algorithm 1 shows the process of finding a (k, t)-hypercore, where $\mathcal{D}$ maps the index of a hyperedge to the original size of the hyperedge (in the original hypergraph $H = (V, E)$, $\mathcal{D}(i) = |e_i|, \forall i \in I_E$). During the process, we remove each node with degree less than k from all its incident hyperedges (Line 5) and delete each hyperedge with the number of remaining nodes below the threshold (Lines 6 to 8). Notably, in the threshold for hyperedges (Line 6), we also require the cardinality to be at least 2 because of the definition of hypergraphs. When the degree of a node decreases from k to $k - 1$, it is added to the set of nodes to be removed in the next round (Line 7).

Theorem 1 *Given $H = (V, E)$, $k \in \mathbb{N}$, and $t \in [0, 1]$, Algorithm 1 returns $C_{k,t}(H)$ in $O(|V| + |E| + (1 - t)\sum_{e \in E} |e|)$ time.*[4]

Proof See Appendix A.6. □

[4] We assume that the input hypergraph is in the memory and thus do not count the complexity of loading the hypergraph, which is $O(\sum_{e \in E} |e|)$.

4.2 Computation of *t*-hypercoreness

Algorithm 2 describes the process of computing t-hypercoreness. Essentially, by the containment property w.r.t k, we repeatedly find the (k, t)-hypercore, while increasing k until the remaining hypergraph becomes empty; and thus Algorithm 2 can also output the (k, t)-hypercores for the given t and all possible k with the same time complexity as shown in Thm. 2.

Theorem 2 *Given $H = (V, E)$ and $t \in [0, 1]$, Algorithm 2 returns $c_t(v)$ for all $v \in V$ in $O(c_t^*|V| + |E| + (1-t)\sum_{e\in E}|e|)$ time.*

Proof See Appendix A.7. □

Algorithm 3 k-Fraction

Input: $H = (V, E)$ and k
Output: k-fraction $f_k(v)$ for each $v \in V$

1: $\mathcal{D}(i) \leftarrow |e_i|, \forall i \in I_E$ ▷ Record original sizes
2: $H' = (V', E') \leftarrow$ Alg. 1 with H, k, 0 and $\mathcal{D}$
3: $t \leftarrow 0$
4: $\mathcal{R} \leftarrow \varnothing$
5: **while** $H' \neq \varnothing$ **do**
6: **if** $\mathcal{R} = \varnothing$ **then**
7: $t \leftarrow \min_{e'_i \in E'} |e'_i|/\mathcal{D}(i)$
8: **for each** $e'_i \in E'$ **s.t.** $|e'_i| = t\mathcal{D}(i)$ **do**
9: $\mathcal{R} \leftarrow \mathcal{R} \cup \{v \in e : d(v;\ H') = k\}$
10: $E' \leftarrow E' \setminus \{e\}$
11: **end for**
12: **else**
13: $f_k(v) \leftarrow t, \forall v \in \mathcal{R}$
14: $\mathcal{R}' \leftarrow \varnothing$ ▷ Nodes to remove in next round
15: **for each** $e_i \in E$ **s.t.** $e_i \cap \mathcal{R} \neq \varnothing$ **do**
16: $e_i \leftarrow e_i \setminus \mathcal{R}$ ▷ Remove nodes
17: **if** $|e_i| \leq t\mathcal{D}(i)$ **or** $|e_i| < 2$ **then**
18: $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{v \in e_i : d(v;\ H) = k\}$
19: $E \leftarrow E \setminus \{e_i\}$ ▷ Remove hyperedge
20: **end if**
21: **end for**
22: $V \leftarrow V \setminus \mathcal{R}$
23: $\mathcal{R} \leftarrow \mathcal{R}'$
24: **end if**
25: **end while**
26: **return** $f_k(v)$ for each $v \in V$

4.3 Computation of *k*-fraction

Algorithm 3 shows the process of computing k-fraction. Similar to Algorithm 2, we repeatedly find the (k, t)-hypercore while increasing t until an empty hypergraph remains. We first find the minimum fraction t for the remaining hyperedges (Line 7), i.e., at least one hyperedge will be totally removed if we use any fraction strictly larger than t. We check the hyperedges that will be

immediately removed and collect the nodes that will consequently be removed (Lines 8–11). Notably, Algorithm 3 can output the (k, t)-hypercores for the given k and all possible t with the same time complexity in Thm. 3.

Theorem 3 *Given $H = (V, E)$ and $k \in \mathbb{N}$, Algorithm 3 returns $f_k(v)$ for all $v \in V$ in $O(\sum_{e \in E} |e|)$ time.*

Proof See Appendix A.8. □

There are some existing works on improving the efficiency of the computation of some related hypercore concepts (Luo et al. 2021, 2022; Arafat et al. 2023). We leave potential improvements of our computation algorithms as future directions.

5 Observations

In this section, we present observations with regard to our proposed concepts, on real-world hypergraphs, from various perspectives. In particular, we show empirical properties and patterns that are pervasive or shared within each domain.

Datasets In Tbl. 2, we report the basic statistics of the fourteen real-world hypergraph datasets in six different domains used in this work (source: cs.cornell.edu/ arb/data).

For each dataset, we remove the hyperedges of cardinality 1. Although parallel hyperedges are allowed in our framework, we only keep one copy of each group of parallel hyperedges as in previous studies (Ko et al. 2022; Lee et al. 2020; Do et al. 2020; Lee et al. 2021).

5.1 Patterns of (k, t)-hypercore sizes

Due to the newly introduced parameter t, we have hypercores of different sizes for different (k, t) pairs. In Fig. 2, we report the hypercore sizes (i.e., the number of nodes in the hypercore) for different k and t, where the color represents the size of the (k, t)-hypercore. Specifically, the color of the position (k, t) is the color assigned to $\tilde{n}_{k,t} := \log_{|V|} |V(C_{k,t})| \in [0, 1]$, for all (k, t) such that $C_{k,t} \neq \emptyset$. Fig. 2 also shows f_k^* for each k (see the boundary between the colored and empty regions in each subfigure).

Similarity within each domain is observed in Fig. 2. To numerically measure the similarity, we need to compare the size of all (k, t)-hypercores in different hypergraphs. Since different hypergraphs may have different absolute sizes and thus have different ranges of (k, t) pairs, normalization is needed. Given any hypergraph $H = (V, E)$, by the containment properties (Proposition 2), $1 \leq c_t^* \leq c_0^*, \forall t$. Therefore, we can use the normalizer $\mathcal{N}_H : [0, 1] \to \{1, 2, \ldots, c_0^*\}$ defined by $\mathcal{N}_H(x) = \lceil (c_0^*)^x \rceil$. We then define the dissimilarity between two hypercore sizes by their difference in log scale (as in Fig. 2), which is also normalized in [0, 1]. Formally, the dissimilarity between two hypergraphs H_1, H_2

Table 2 The basic statistics of the 14 real-world datasets from 6 domains used in our empirical evaluations.See Table 5 in Appendix C for the number of hyperedges of different cardinality in each dataset

Dataset	$\|V\|$	$\|E\|$	max./avg. $d(v)$	max./avg. $\|e\|$
Coauth-DBLP	1,831,126	2,169,663	846 / 4.06	25 / 3.42
Coauth-Geology	1,087,111	908,516	716 / 3.21	25 / 3.84
NDC-classes	1149	1047	221 / 5.57	24 / 6.11
NDC-substances	3438	6264	578 / 14.51	25 / 7.96
Contact-high	327	7818	148 / 55.63	5 / 2.33
Contact-primary	242	12,704	261 / 126.98	5 / 2.42
Email-Enron	143	1457	116 / 31.43	18 / 3.09
Email-Eu	979	24,399	910 / 86.93	25 / 3.49
Tags-ubuntu	3021	145,053	12,930 / 164.56	5 / 3.43
Tags-math	1627	169,259	13,949 / 363.80	5 / 3.50
Tags-SO	49,945	5,517,054	520,468 / 427.77	5 / 3.87
Threads-ubuntu	90,054	115,987	2170 / 2.97	14 / 2.31
Threads-math	153,806	535,323	11,358 / 9.08	21 / 2.61
Threads-SO	2,321,751	8,589,420	34,925 / 9.75	25 / 2.64

at the normalized point (x, t) with $x, t \in [0, 1]$ is $\tilde{d}(x,t;H_1,H_2) := \min(|\tilde{n}_{\mathcal{N}_{H_1}(x),t}(H_1) - \tilde{n}_{\mathcal{N}_{H_2}(x),t}(H_2)|, 1)$, where we let $\tilde{n}_{k,t} = -1$ if $C_{k,t}$ is empty. This dissimilarity can also be understood as the difference between the same position of two subfigures in Fig. 2. Finally, we define the hypercore-size-mean-difference (HSMD) distance, which lies between 0 and 1, as follows:

Definition 15 (Hypercore-size-mean-difference (HSMD) distance) Given two hypergraphs H_1 and H_2, the hypercore-size-mean-difference (HSMD) distance between H_1 and H_2 is defined as

$$\mathrm{HSMD}(H_1, H_2) := \sqrt{\int_0^1 \int_0^1 (\tilde{d}(x,t;H_1,H_2))^2 \, dx \, dt}.$$

See Fig. 3 for the HSMD distance between each pair of datasets, where the domain-based patterns are clearly shown by the small distance between those datasets in the same domain.

Observation 1 (Domain-based patterns of (k, t)-hypercore sizes) Real-world hypergraphs in the same domain usually have similar patterns of the hypercore sizes with different k and t values, and the patterns vary from domain to domain.

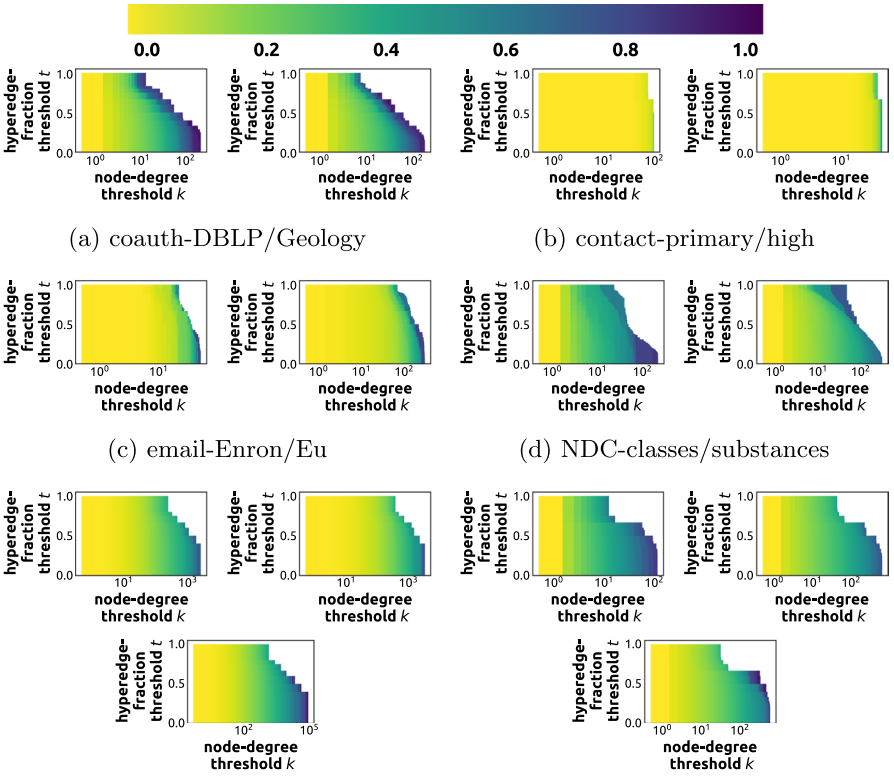


Fig. 2 Domain-based patterns of (k, t)-hypercore sizes. The (k, t)-hypercore sizes vary depending on the node-degree threshold k and the hyperedge-fraction threshold t with datasets grouped by domains. The color indicates the size of the corresponding (k, t)-hypercore. The size and k are in a log scale

5.2 Distributions of *t*-hypercoreness

We now investigate the distributions of the t-hypercoreness of nodes with different t values, which show common patterns. Heavy-tailed distributions, especially power-law distributions, are observed in real-world (hyper)graphs w.r.t many different quantities (McGlohon et al. 2008; Watts and Strogatz 1998; Albert and Barabási 2002; Adamic et al. 2001; Ko et al. 2022; Lee and Shin 2021). In Fig. 4, for the t-hypercoreness sequences of each dataset with $t \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, we report the log-likelihood ratio (R-value) of heavy-tailed distributions against the exponential distribution, where a positive R-value indicates that heavy-tailed distributions are more promising. In particular, we compute the log-likelihood ratio for two heavy-tailed distributions (power-law and log-normal) and take the maximum. In most cases, the log-likelihood ratio is positive, which supports the possibility that the t-hypercoreness follows heavy-tailed distributions consistently regardless of the value of t. Notably, regarding the

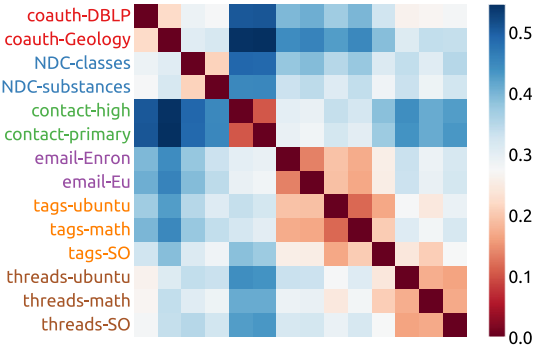


Fig. 3 Datasets in the same domain tend to have small HSMD distance, while ones in different domains usually have large HSMD distance. The average within-domain distance 0.166 and the global average distance 0.323 are significantly different with $p = 8.6e{-}10$ in the t-test

distributions of k-fraction, we could not find any systematic pattern. Moreover, strong power-law distributions are observed in some datasets. In Fig. 5, for two datasets, we show the numbers of nodes with t-hypercoreness at least k with different k values with different t values, together with the results of power-law fitting, i.e., linear regression in log-log scale; and consistent power-law distributions of the t-hypercoreness sequences are observed. In Table 3, we provide the full results of the heavy-tailed distribution tests. Specifically, we report the log-likelihood ratio (R-value) of heavy-tailed distributions against the exponential distribution, where a positive R-value indicates that heavy-tailed distributions are more promising; and the p-values, where a small p-value indicates that the heavy-tailed or exponential distribution is significant.

Observation 2 (Heavy-tailed distributions of t-hypercoreness) In most real-world hypergraphs, t-hypercoreness follows heavy-tailed distributions regardless of t. In particular, in some datasets, the t-hypercoreness strongly follows a power law.

5.3 Heterogeneity of t-hypercoreness

We show that t-hypercoreness is statistically different from several existing centrality measures, and t-hypercoreness provides significantly different information depending on t.

Correlations. To show (a) the distinctiveness of t-hypercoreness from existing centrality measures, and (b) the dissimilarity between t-hypercoreness with different t values, we first measure the Pearson correlation coefficients. In Fig. 6, we report Pearson's r between the t-hypercoreness sequences with different t values and each of the **degree** and **coreness** sequences in the *unweighted* and *weighted* clique expansions. We also report Pearson's r between each pair of t-hypercoreness sequences. It is observed that even for the same hypergraph, the hypercoreness sequences with different t values can be fairly dissimilar.

Information gain. We also show from the perspective of information theory that hypercoreness sequences with different t values contain different information. To this end, we define the *information gain*.

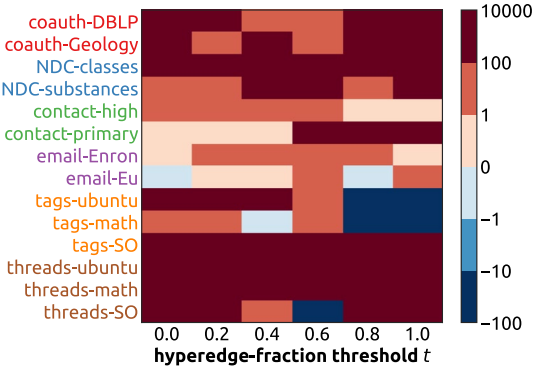
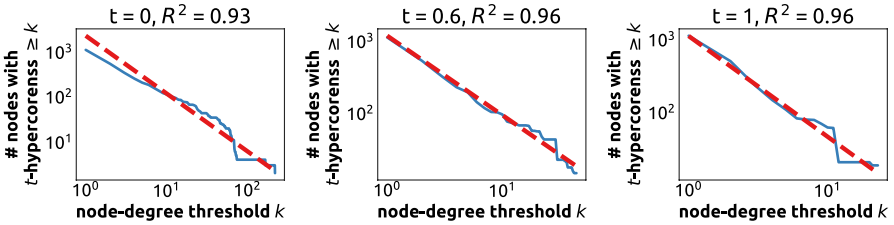
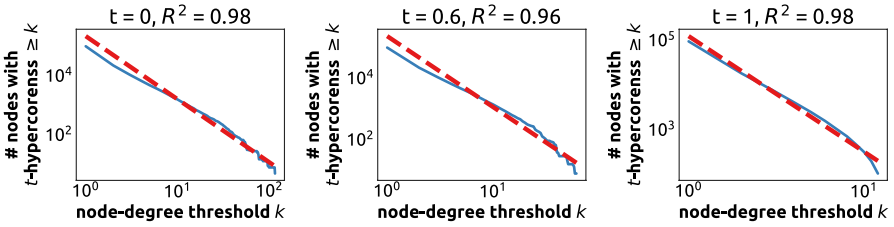


Fig. 4 t-Hypercoreness follows heavy-tailed distributions consistently. The maximum log-likelihood ratio of two heavy-tailed distributions (power-law and log-normal) against the exponential distribution for the t-hypercoreness sequences with different t values



(a) NDC-classes



(b) threads-ubuntu

Fig. 5 t-Hypercoreness consistently follows power-law distributions in some datasets. For the NDC-classes and threads-ubuntu datasets with $t \in \{0, 0.6, 1\}$, we show the numbers of nodes with t-hypercoreness at least k with different k values. Each red dashed line represents the result of power-law fitting, i.e., the linear regression in log-log scale, with the R^2 value above each subfigure. In the two datasets, t-hypercoreness consistently and strongly follows a power law

Definition 16 (Information gain (Quinlan 1986)) Given $H = (V, E)$, for $i \in \mathbb{N}$, define $V_i := \{v \in V : d(v) = i\}$ for $i \in \mathbb{N}$, and $V_i^t := \{v \in V : c_t(v) = i\}$. The **information gain** of the t-hypercoreness sequence over the degree sequence is

$$\mathcal{H}^t(H) := -\sum_{i,j\in\mathbb{N}} \frac{|V_i \cap V_j^t|}{n} \log_2 \frac{|V_i \cap V_j^t|}{n} + \sum_{i\in\mathbb{N}} \frac{|V_i|}{n} \log_2 \frac{|V_i|}{n}.$$

Table 3 The detailed statistics on the heavy-tailed distribution tests. For each dataset and each $t \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, we report the log-likelihood ratio (R-value) of heavy-tailed distributions against the exponential distribution with its p-value. In most cases, the R-value is positive and the p-value is small, which implies the significance of the heavy-tailed distributions

	$t = 0$		$t = 0.2$		$t = 0.4$		$t = 0.6$		$t = 0.8$		$t = 1$	
Dataset	R-value	p-value	R-value	p-value	R-value	p-value	R-value	p-value	R-value	p-value	R-value	p-value
Coauth-DBLP	156.75	7.24e−13	184.96	7.28e−16	139.27	3.36e−13	50.80	0.001	1685.14	1.45e−40	117.76	4.75e−56
Coauth-Geology	106.38	3.06e−11	83.21	7.70e−8	31.52	6.86e−8	17.80	9.57e−5	1049.01	0.0	989.44	0.0
NDC-classes	45.32	4.38e−6	364.85	5.80e−24	103.93	4.90e−14	282.71	1.06e−42	290.16	4.21e−45	242.25	1.28e−41
NDC-substances	30.90	2.34e−5	26.95	0.00061	2608.99	4.33e−208	1884.07	6.54e−171	1175.06	8.51e−91	221.15	2.78e−24
Contact-high	16.15	3.20e−20	16.15	3.20e−20	16.15	3.20e−20	16.76	0.0040	0.70	0.48	0.70	0.48
Contact-primary	0.19	0.23	0.19	0.23	0.19	0.23	136.51	1.75e−16	127.81	6.41e−13	127.81	6.41e−13
Email-Enron	0.29	0.73	2.05	0.24	2.67	0.063	8.43	0.024	1.55	2.4e−267	0.22	0.76
Email-Eu	−0.47	0.60	0.05	0.97	2.40	3.77e−9	83.69	2.05e−11	−0.28	0.36	11.26	0.005
Tags-ubuntu	201.24	4.28e−21	201.24	4.28e−21	201.24	4.28e−21	83.69	2.05e−11	−14.68	5.76e−6	−17.40	1.30e−32
Tags-math	8.81	0.06	8.81	0.06	8.81	0.06	15.31	0.027	−17.96	1.03e−9	−14.34	0.00052
Tags-SO	616.59	2.41e−29	616.59	2.41e−29	3617.24	8.07e−222	2189.25	5.27e−234	−17.40	1.30e−32		
Threads-ubuntu	279.41	2.09e−22	278.50	2.71e−22	259.53	8.96e−22	130.95	1.15e−14	119.14	8.15e−14	226.74	6.06e−41
Threads-math	226.30	1.45e−23	225.66	1.68e−23	5192.50	2.08e−282	11461.10	0.0	3305.64	0.0	6632.53	0.0
Threads-SO	444.93	3.47e−57	436.37	4.84e−56	153.14	6.50e−19	−23.83	7.24e−8	6002.46	0.0	2682.90	4.64e−102

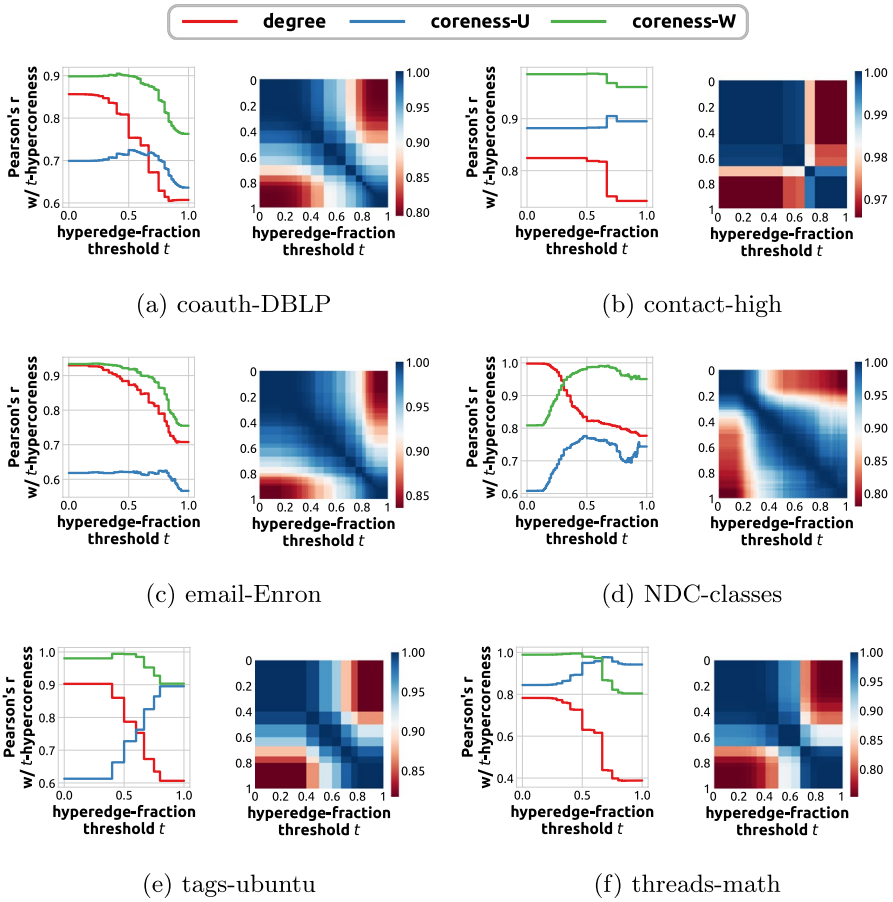


Fig. 6 Statistical difference exists between t-hypercoreness and other centrality measures, as well as among t-hypercoreness with different t. Left: the Pearson correlation coefficients between the t-hypercoreness sequences with different t and each of the degree and coreness sequences in the unweighted (coreness-U) and weighted (coreness-W) clique expansions. Right: the Pearson correlation coefficient between each pair of t-hypercoreness sequences. See Fig. 15 in Appendix C for the results on other datasets

The higher the information gain a hypercoreness sequence has, the more finely the nodes can be divided by the corresponding degree-hypercoreness pairs. In Fig. 7, we report the information gain for different t values. The highest information gain is achieved by different t values in different datasets, and hypergraphs in the same domain show similar patterns. In summary:

Observation 3 (Heterogeneity of t-hypercoreness) In real-world hypergraphs, the t-hypercoreness of nodes provides statistically and information-theoretically distinct information depending on t.

6 Applications

In this section, we present some successful applications of our proposed concepts to demonstrate their usefulness.

6.1 Influential-node identification

It is well-known that in pairwise graphs, coreness is a good indicator of influential nodes (Kitsak et al. 2010). However, influential-node identification in hypergraphs is still underexplored, while some trials have been done (Zhu et al. 2018a; Antelmi et al. 2021). We use the SIR model, a widely-used epidemic model. The model is straightforwardly generalized so that it can be used on hypergraphs, where the probability of a susceptible node being infected by the infected nodes in a hyperedge is proportional to the proportion of infected nodes in the hyperedge. At each time step, each infected node recovers with a given probability (γ) independently. We simulate the HYPERSIR (see Algorithm 4) process assuming a single initially infected node. In Algorithm 4, we show the process of HYPERSIR. In the relatively large datasets (coauth-DBLP, coauth-Geology, and threads-SO), we randomly draw 10% of the nodes, and perform the simulation 100 times for each seed node. In the other datasets, we simulate 10, 000 times for each node as the seed.

Algorithm 4 HYPERSIR

Input: $H = (V, E)$, seed node v^*, transmission rate β, and recovery rate γ
Output: number of ever-infected nodes $|R|$

$S \leftarrow V \setminus \{v^*\}$; $I \leftarrow \{v^*\}$; $R \leftarrow \varnothing$
while $I \neq \varnothing$ **do**
  $P_s(v_s) \leftarrow 1, \forall v_s \in S$
  for each $e \in E$ **s.t.** $e \cap I \neq \varnothing \wedge e \cap S \neq \varnothing$ **do**
    $I_e \leftarrow e \cap I$; $S_e \leftarrow e \cap S$
    $P_s(u) \leftarrow P_s(u)(1 - 2\beta|I_e|/|e|), \forall u \in S_e$
  end for
  v_i moves to R with probability γ, $\forall v_i \in I$
  v_s moves to I with probability $1 - P_s(v_s)$, $\forall v_s \in S$
end while
return $|R|$

We investigate the relations between the average number of ever-infected nodes and the following quantities of the seed node in addition to ***t* -hypercoreness** and **degree**:

- *Neighbor-hypercoreness* (Arafat et al. 2023): see Definition 12 (abbreviation: nbr-hypercoreness);
- *Neighbor-degree-hypercoreness* (Arafat et al. 2023): see Definition 14 (abbreviation: nd-hypercoreness);

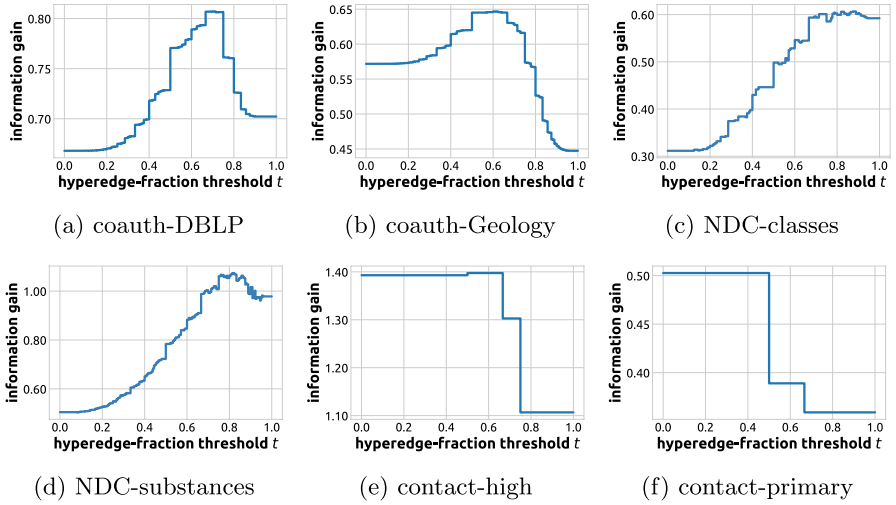


Fig. 7 t-Hypercoreness has substantial information gain over degree, and it provides distinct information depending on t. The average Pearson's r between the information gain sequences is 0.232 overall and 0.890 within domains. The two values are significantly different with 5.0e−9 as the p-value of the t-test. See Appendix C for the results on other datasets and the results using other quantities

- *Coreness* in the unweighted (coreness-U) / weighted (coreness-W) clique expansion;
- *Eigencentrality* in unweighted (eigencentrality-U)/ weighted (eigencentrality-W) clique expansion;
- *Hyper-eigencentrality* (Tudisco and Higham 2021): three different versions, linear (hyperEC-L), log-exp (hyperEC-LE), and max (hyperEC-M);
- ℓ **-hypercoreness** (Limnios et al. 2021): see Definition 9.[5]

We also consider two supervised machine-learning methods. Specifically, we apply **node2vec** (Grover and Leskovec 2016) to the unweighted clique expansion of each dataset, and we apply a self-supervised hypergraph learning method **TriCL** (Lee and Shin 2023) (which is based on the architecture proposed by Feng et al. (2019)) directly to the original hypergraphs. Both additional baseline methods output node embeddings of dimension 128. For each dataset, we sample 10% of the nodes (for the three relatively large datasets where we only use 10% of the nodes, we sample 1% of the total nodes, i.e., 10% of the 10%) uniformly at random and provide the ground-truth influence of the sampled nodes.[6] For both methods, we apply linear regression using the node embeddings as the features, and then we use the fitted

[5] Recall that ℓ-hypercoreness with $\ell = 2$ is include in t-hypercoreness with $t = 0$. For each dataset, we apply min-max normalization to all the possible ℓ values with $\ell \geq 3$ so that t-hypercoreness and ℓ -hypercoreness can fit in the same x-axis with the range [0, 1].

[6] The average performance over five independent trials is reported.

linear regression model to predict the influence of the nodes. Due to the scalability issues, results of them are unavailable on some large datasets.

We take the largest connected component of each dataset, as in previous works on pairwise graphs (Kitsak et al. 2010). For simplicity, we use $\gamma = 1$, and choose $\beta \in \{0.05, 0.025, 0.01, 0.005, 0.0025\}$ to avoid the cases when almost all seed nodes lead to similar results. For the *email-Eu* dataset, Fig. 9 shows the detailed relations between the average number of ever-infected nodes (i.e., final $|R|$) and each of the aforementioned quantities, with the best-fitted lines. Figure 8 shows the Pearson correlation coefficient between the average number of ever-infected nodes and each quantity. The comparison between t-hypercoreness and the coreness in clique expansions validates the information loss brought by the clique expansions. On most of the datasets, at least one among the t-hypercoreness with $t \in \{0, \frac{1}{2}, \frac{2}{3}, 1\}$ works better than all the other baseline methods. On the remaining datasets, t-hypercoreness with a proper t value ranks second. Moreover, even if we always use the t-hypercoreness with $t = \frac{1}{2}$, t-hypercoreness still outperforms all the baseline methods on 10 out of 14 datasets. In practice, we may sample a small number of nodes and choose the t value that is most influence-indicative (w.r.t the Pearson correlation coefficient) on the sampled set of nodes. For this purpose, we use the same 10% nodes (or 1% for some large datasets) that are used as a training set for the machine-learning methods. In Table 4, for each dataset, we show (1) the most indicative t value in each of the five trials, (2) the performance and rank of t-hypercoreness averaged on the five trials,[7] and (3) the performance and rank of t-hypercoreness with the best t values among the four candidate values. We can observe that a well-performing t value is always found (although the chosen t values may vary), and t-hypercoreness performs well and stably, almost always outperforming all the baselines.

Observation 4 (Influence indicativeness of t-hypercoreness) In real-world hypergraphs, t-hypercoreness identifies influential nodes well. In most cases, t-hypercoreness with a proper t is the best indicator of influential nodes among all considered centrality measures. In different hypergraphs, the t value maximizing the correlation between t-hypercoreness and node influence varies, and in most cases, such t is neither 0 nor 1.

6.2 Dense substructure discovery

Intuitively, (k, t)-hypercores are not limited to complete subhypergraphs. Thus, they can be denser than complete subhypergraphs, which previous works (Hua

[7] We count ℓ-hypercoreness with each ℓ value as a separate method ($\ell = 2$ is not counted since it is already included in the concept of t-hypercoreness with $t = 0$).

et al. 2023; Luo et al. 2021, 2022; Gabert et al. 2021a, b; Sun et al. 2020) focus on.

Given $H = (V, E)$, we define its density as $\delta(H) = |E|/|V|$. In Fig. 10, for each dataset and each $t \in [0, 1]$, we show the relative density of the (c_t^*, t)-hypercore, which is defined as $\tilde{\delta}_t = \delta(C_{c_t^*,t})/\delta(H)$. Note that the hypercores are significantly denser than the whole hypergraph, especially when t is small. In addition, except for the *tags-SO* dataset, the similarity between hypergraphs in the same domain is observed. Similar to the normalized hypercore-size-mean-difference (HSMD) distance used in Sect. 5.1, we define the relative-density-mean-difference (RDMD) distance between two hypergraphs to measure the similarity of the patterns.

Definition 17 (Relative-density-mean-difference (RDMD) distance) Given two hypergraphs H_1 and H_2, the relative-density-mean-difference (RDMD) distance between H_1 and H_2 is defined as

$$\mathrm{RDMD}(H_1, H_2) := \sqrt{\int_0^1 (\log \tilde{\delta}_t(H_1) - \log \tilde{\delta}_t(H_2))^2 \, dt}.$$

See Fig. 11 for the RDMD distance between each pair of datasets.

Observation 5 (Density of (k, t)-hypercores) In real-world hypergraphs, (k, t)-hypercores are dense, and the density tends to decrease as t increases. The relative density with respect to t tends to be similar in hypergraphs in the same domain.

We utilize the high density of (k, t)-hypercores for the *max* (k_c, t_c)-*vertex cover problem* below, where we say a hyperedge e is t_c-*covered* by a set of nodes V' if $|e \cap V'| \geq t_c|e|$.

Problem 1 (max (k_c, t_c)-vertex cover problem) given a hypergraph $H = (V, E)$, $k_c \in \mathbb{N}$ and $t_c \in (0, 1]$, the **max** (k_c, t_c)**-vertex cover problem** aims to find $V^* \in \binom{V}{k_c} := \{V' \subset V : |V'| = k_c\}$ such that the number of hyperedges t_c-covered by V^* is maximized.

In our experiments, we compare three different algorithms:

- t_c-*Hypercoreness*: k_c nodes with highest t_c-hypercoreness in H are chosen (tie broken by node-degrees);
- *Degree* k_c nodes with highest degree in H are chosen;
- *Greedy* it first chooses the node with the highest degree and greedily chooses a node that increases the number of t_c-covered hyperedges most until k_c nodes are chosen.

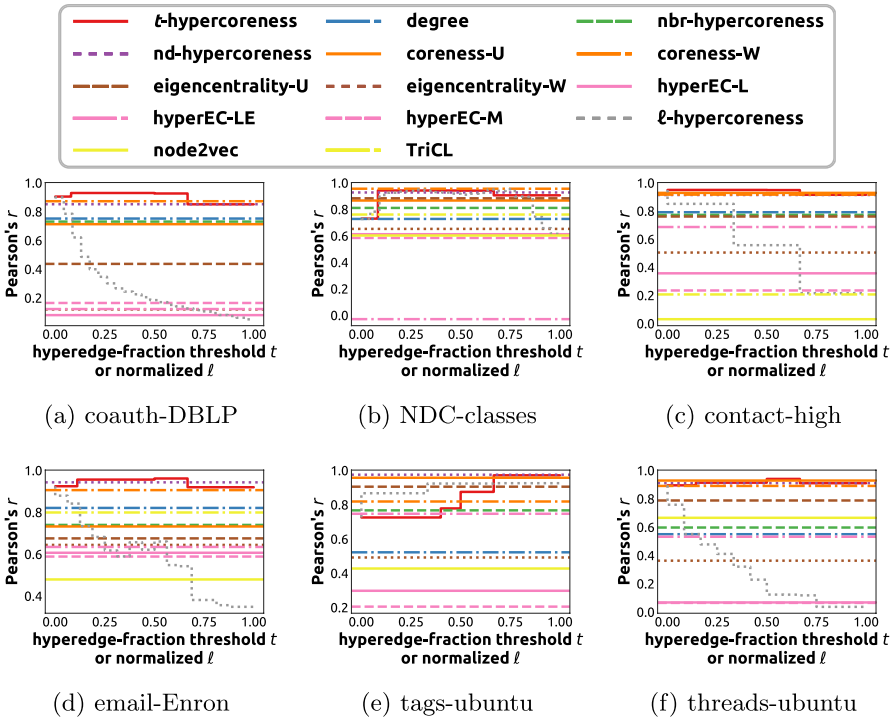
Table 4 Results of t-hypercoreness by choosing the t values based on sampled nodes. Sampled nodes: the results where the t values are chosen based on sampled nodes. Ground-truth best: the results where for each dataset, the best t value is chosen among the candidate values. Best t: the most indicative t value (in each of the five trials, or among the candidate values). Perm.: the average performance (the Pearson correlation coefficient; the higher the better) over the five trials. Rank: the rank (the lower the better) among all the baseline methods and each considered one (i.e., the result based on sampled nodes or using the ground-truth best t value)

	Sampled nodes			Ground-truth best		
Dataset	Best t	Perm	Rank	Best t	Perm	Rank
Coauth-DBLP	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	0.927 ± 0.000	1	$\frac{1}{2}$	0.927	1
Coauth-Geology	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	0.930 ± 0.000	1	$\frac{1}{2}$	0.930	1
NDC-classes	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{2}{3}, \frac{2}{3})$	0.939 ± 0.001	2	$\frac{1}{2}$	0.940	2
NDC-substances	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$	0.959 ± 0.000	1	$\frac{2}{3}$	0.959	1
Contact-high	(0, 0, 0, 0, 0)	0.947 ± 0.000	1	0	0.947	1
Contact-primary	$(0, 0, \frac{2}{3}, \frac{2}{3}, 1)$	0.970 ± 0.007	1	$\frac{2}{3}$	0.975	1
Email-Enron	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$	0.960 ± 0.000	1	$\frac{2}{3}$	0.960	1
Email-Eu	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{2}{3}, \frac{2}{3})$	0.975 ± 0.003	1	$\frac{2}{3}$	0.977	1
Tags-ubuntu	(1, 1, 1, 1, 1)	0.970 ± 0.000	2	1	0.970	2
Tags-math	(1, 1, 1, 1, 1)	0.990 ± 0.000	1	1	0.990	1
Tags-SO	(1, 1, 1, 1, 1)	0.844 ± 0.000	2	1	0.844	2
Threads-ubuntu	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$	0.938 ± 0.000	1	$\frac{2}{3}$	0.938	1
Threads-math	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$	0.971 ± 0.000	1	$\frac{2}{3}$	0.971	1
Threads-SO	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$	0.962 ± 0.000	1	$\frac{2}{3}$	0.962	1

In each dataset, we track the count of t_c-covered hyperedges by the k_c nodes chosen by each algorithm while varying k_c from 10 to 100. Then, we divide each count by the count obtained by the *degree* algorithm in the same setting. The relative counts are averaged over all datasets for $t_c \in \{0.6, 0.7, 0.8\}$ and reported in Fig. 12. We choose those t_c values because they require a majority of, but not all of, the constituent nodes to cover a hyperedge. On average, the algorithm t_c-*hypercoreness* outperforms the other two algorithms, with clear superiority when $t_c \in \{0.6, 0.7\}$.

6.3 Hypergraph vulnerability detection

Through the observations and applications, we have shown the significance of the proposed concepts and the importance of nodes in the (k, t)-hypercore with large k values. Thus, in order to reinforce the engagement of nodes in a hypergraph (e.g.,



(a) coauth-DBLP (b) NDC-classes (c) contact-high

(d) email-Enron (e) tags-ubuntu (f) threads-ubuntu

Fig. 8 t-Hypercoreness is consistently indicative of influence in all datasets. We show the Pearson correlation coefficients between the average number of ever-infected nodes and each of the considered quantity of the seed node. See Fig. 17 in Appendix C for the results on other datasets

user engagement in online social networks), intuitively, the (k, t)-hypercores should be paid close attention to. From another perspective, we should protect the nodes whose deletions will cause a large number of nodes to leave the (k, t)-hypercores. For example, online social network providers should try to make such nodes stay. Based on such ideas, in pairwise graphs, the *collapsed k-core problem* (Zhang et al. 2017a) and its variants (Zhu et al. 2018b, 2019) have been considered to find the critical users whose deletions reduce the size of k-core most, i.e., the most *vulnerable* nodes in the k-core. We generalize the problem to hypergraphs based on our proposed concepts.

Problem 2 (Collapsed (k, t)-hypercore problem) Given a hypergraph $H = (V, E)$, $k \in \mathbb{N}$, $t \in [0, 1]$, and $b \in \mathbb{N}$. The **collapsed** (k, t)**-hypercore problem** aims to find $B \in \binom{V}{b}$ so that the size (i.e., the number of nodes) of (k, t)-hypercore is minimized when all nodes in B are removed from H.

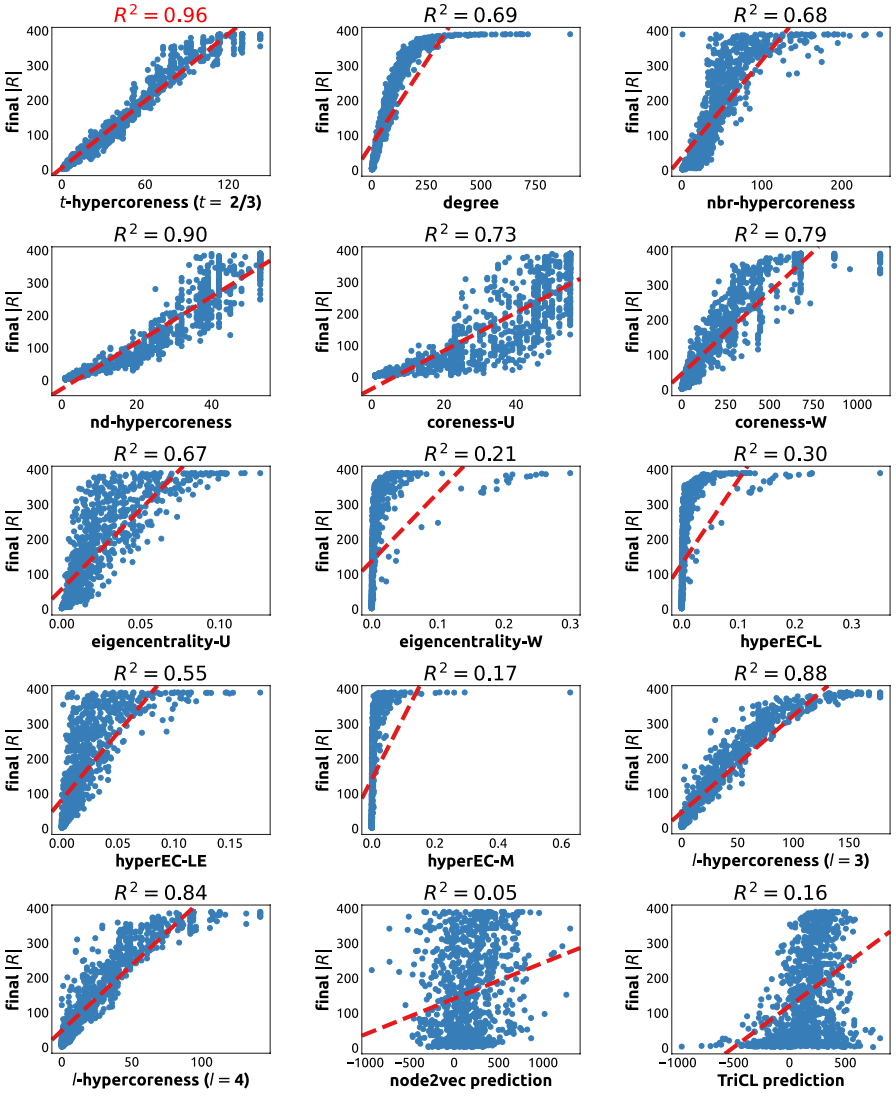


Fig. 9 t-Hypercoreness with a proper t value is the best indicator of influence among all considered centrality measures (Dataset: email-Eu). The red dashed line represents the best-fitted line, and the R^2 value is shown above each subfigure. The full results on all the datasets are in the supplementary document (Bu et al. 2023)

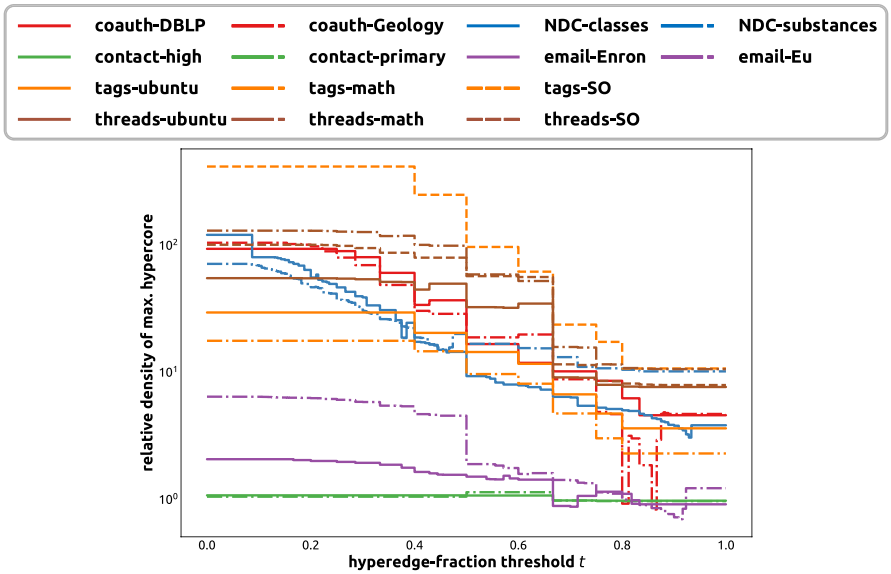
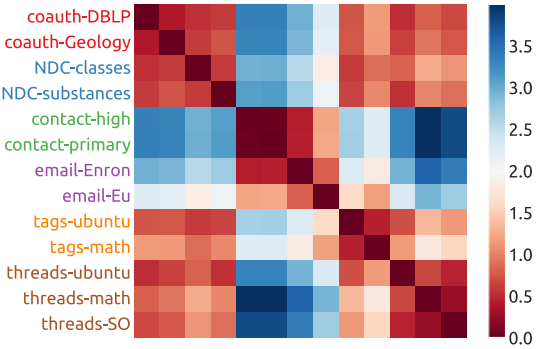


Fig. 10 Overall, hypercores are much denser than the whole hypergraph, and the density decreases as t increases. For each dataset, we report the relative density of the (c_t^*, t)-hypercore (i.e., the (k, t)-hypercore with maximal k) w.r.t t

Fig. 11 The RDMD distance is small between datasets in the same domain (0.456 in average) while the overall average is 1.741; the two means are significantly different with $p = 0.0035$ in the t-test. We report the RDMD distance between each pair of datasets except for *tag-SO*



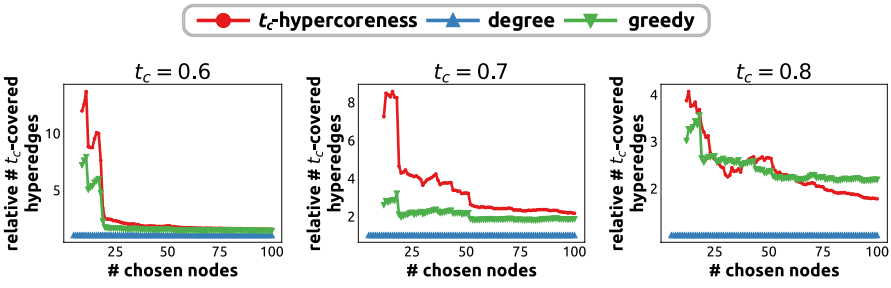


Fig. 12 Nodes chosen by t_c-hypercoreness cover most hyperedges. The performance of t_c-*hypercoreness*, *degree*, and *greedy* in solving the max (k_c, t_c)-vertex cover problem

Algorithm 5 HyCoM / HyCoM+

Input: $H = (V, E)$, k, t, budget b, and max. # candidates to check n_c
Output: set of the chosen collapsers $\mathcal{C}$

$\mathcal{D}(i) \leftarrow |e_i|$; $\tilde{\mathcal{T}}(i) \leftarrow \max(\lceil t|e_i| \rceil, 2), \forall i \in I_E$
$H' = (V', E') \leftarrow$ Alg. 1 with H, k, t, and $\mathcal{D}$
$\mathcal{C} \leftarrow \varnothing$
for $i \in \{1, 2, \ldots, b\}$ **do**
  $v^* \leftarrow$ bestCollapser()
  $H' = (V', E') \leftarrow$ Alg. 1 with $H' \setminus \{v^*\}$, k, t, and $\mathcal{D}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{v^*\}$
end for
return $\mathcal{C}$

bestCollapser: find the best collapser

procedure bestCollapser ▷ The blue parts are for HyCoM+
  if $i > 1$ **then goto** Line 13 ▷ Construct $\tilde{E}$ once
  $\tilde{E}(u, v) \leftarrow \{e'_i \in E' : \{u, v\} \subseteq e'_i, |e'_i| = \tilde{\mathcal{T}}(i)\}, \forall u, v$
  $T \leftarrow \{v \in V' : \exists u \ s.t. \ |\tilde{E}(u, v)| > d(u;\ H') - k\}$
  $\mathcal{F}(v) \leftarrow \{u \in V' : |\tilde{E}(u, v)| > d(u;\ H') - k\}, \forall v \in T$
  sort T in the descending order by $|\mathcal{F}(\cdot)|$ and degree
  $n_{col}, v_{col}, n_t \leftarrow |V|, -1, 0$
  while $T \neq \varnothing$ **and** $n_t \neq n_c$ **do**
    $v_0 \leftarrow$ the first element in T
    $H^* \leftarrow$ Alg. 1 with $H' \setminus \{v_0\}$, k, t, and $\mathcal{D}$
    if $|V(H^*)| < n_{col}$ **then** $n_{col}, v_{col} \leftarrow |V(H^*)|, v_0$
    $T \leftarrow T \setminus (V \setminus V(H^*))$
    $n_t \leftarrow n_t + 1$
  end while
  update $\tilde{E}$ by the difference between H' and H_{col}
  return v_{col}
end procedure

Algorithm 5 (with $n_c = -1$) shows the generalization of CKC (Zhang et al. 2017a), which was originally designed for the collapsed k-core problem, to the collapsed (k, t)-hypercore problem. Following CKC, in each round, we find a best collapser (i.e., a node that reduces the size of (k, t)-hypercore most) in the candidate set T, and update the (k, t)-hypercore after removing the chosen collapser, until all b collpasers are chosen. However, the naive generalization encounters the following problems:

- CKC only considers simple pairwise graphs. In simple pairwise graphs, where at most one edge exists between each node pair, the candidate set T, i.e., the set of nodes whose deletion will result in the removal of some other node, simply consists of the neighbors of the nodes with degree k. In hypergraphs, two nodes may co-exist in *multiple* hyperedges. Therefore, we need to additionally check the number of *endangered hyperedges* in the set $\tilde{E}$, where endangered hyperedges are the ones with exactly the minimum size satisfying

the threshold determined by t. Furthermore, we need to count each node pair in each endangered hyperedge, which is time-consuming with time complexity $O(\sum_{e \in E'} |e|^2)$ (Line 12). To make the situation worse, this process is repeated in each round (b times in total).
- CKC computes the k-core after removing each candidate to evaluate the candidates. Similarly, (k, t)-hypercore computation is required for *each* candidate (Line 19), where the number of all candidates can be large. Compared to core computation with time complexity linear in the number of edges (Batagelj and Zaversnik 2003), as shown in Thm. 1, (k, t)-hypercore computation has considerably higher time complexity.

We propose HyCoM (**Hy**per-**Co**re **M**inimization) and the further-optimized HyCoM+, which are described in Algorithm 5, to address the above problems with the following improvements:

- *Only checking the most promising candidates* Although we may have a large number of candidates, not every candidate is likely to be the best collapser. Intuitively, we may set a maximum number of candidates to check in each round ($n_c \ll T$) and only check the most promising ones. The technique reduces the time of hypercore decompositions in each round from $O(T)$ to $O(n_c)$, which gives HyCoM. We further improve the algorithm by the following two techniques and have HyCoM+.
- *Sorting candidates by the number of direct followers* In HyCoM, the degrees are used to sort the candidates and to find the promising ones. However, the degree does not always imply a node's ability in (hyper)core minimization. The *direct followers* ($\mathcal{F}(\cdot)$ in Line 14, i.e., the nodes that will leave the hypercore immediately due to the deletion) of each candidate are accessible without additional cost during the process of finding candidates, and the number of direct followers provides a lower bound for the total number of followers. Thus, we use the number of direct followers to find the most promising candidates (Line 15).
- *Incremental update of the endangered hyperedges* It is necessary to find the endangered hyperedges, and we incrementally update the set whenever the hypercore is updated (Line 24) instead of computing it from scratch in each round. By doing so, during the whole process, $\tilde{E}$ is constructed from scratch only once. The number of hyperedges needed to be checked in each round is the size of the *symmetric difference* between the current set of endangered hyperedges and that in the previous round, which is empirically much less than the total number of hyperedges in the hypercore.

Algorithms. The algorithms to compare are as follows:

- HyperCKC: the naive generalization of CKC, which is equivalent to HyCoM with $n_c = -1$;
- HyCoM-n_c: HyCoM with $n_c \in \{1, 10, 100\}$;
- HyCoM+: HyCoM+ with $n_c = 1$, i.e., the fastest version.

Settings. We conducted all the experiments on a machine with i9-10900K CPU and 64GB RAM. All algorithms are implemented in C++, and complied by G++ with O3 optimization.

Results. We show the results on five relatively large datasets: coauth-DBLP, coauth-Geology, tags-SO, threads-math, and threads-SO, and use $k = 10$ and $t = 0.6$. Full results, where we use different datasets and different (k, t) values, are in Table 7 in Appendix C. In Fig. 13, we report the running time and the reduction in the size of (k, t)-hypercore size when using different algorithms with $b = 100$, where HyCoM+ shows outstanding efficiency and competent effectiveness. We do not count the time used on the initial (k, t)-hypercore computation since it is common in all algorithms. In particular, in the tag-SO, thread-math, and threads-SO datasets, the performance of HyCoM+ is comparable or even better than that of hyperCKC while HyCoM+ is 16.5-27.9× faster than hyperCKC. Besides, on the largest dataset threads-SO whose input hypercore has 301K nodes and 5.7M hyperedges, HyCoM+ takes only 38.2 seconds. In Fig. 14, we show the linear scalability w.r.t the budget and the hypercore size of HyCoM and HyCoM+, where we generate synthetic hypergraphs by upscaling the original ones. In particular, we duplicate each hyperedge up to 64×, which is simple and generates realistic hypergraphs.

7 Related work

***k*-*Hypercores*.** The concept of k-cores in pairwise graphs was first proposed in (Seidman 1983) and has been used for various applications (Shin et al. 2018a; Alvarez-Hamelin et al. 2008, 2006; Peng et al. 2014; Corominas-Murtra et al. 2014; Luo et al. 2009; Wood and Hicks 2015; Malliaros et al. 2020). Most previous works (Hua et al. 2023; Luo et al. 2021, 2022; Gabert et al. 2021a, b; Sun et al. 2020) are based on the straightforward generalization of k-cores to hypergraphs assuming fragile hyperedges (i.e., a hyperedge is removed when *any* node leaves it), which is included in the proposed (k, t)-hypercore with $t = 1$. Limnios et al. (2021) and Vogiatzis (2013) considered a variant where each hyperedge is kept until only one node remains in it, which is equivalent to the (α, β)-core in bipartite graphs (Liu et al. 2020; Sarıyüce and Pinar 2018) with $(\alpha, \beta) = (k, 2)$, and the proposed (k, t)-hypercore with $t = 0$. No existing work has investigated the spectrum between the two extreme cases above, which is covered by our proposed concepts.

***Generalized k*-*cores*.** Zhang and Parthasarathy (2012) generalized k-cores to triangle k-cores, which are also known as k-trusses, to extract the information in pairwise graph. Peng et al. (2018) generalized k-cores on uncertain graphs, where each edge exists in a probabilistic way. Specifically, they considered the problem of k-core decomposition on uncertain graphs, and propose the concept of (k, θ)-cores. Wang et al. (2018) generalized k-cores on geo-social networks. Specifically, they proposed the radius-bounded k-core by taking the spatial constraints into consideration. Zhang et al. (2020) generalized k-cores to (k, p)-cores. Specifically, given k and p, they further required each node in the (k, p)-core to have at least p fraction of its neighbors in the (k, p)-core. Lu et al. (2022) further investigated (k, p)-cores on dynamic

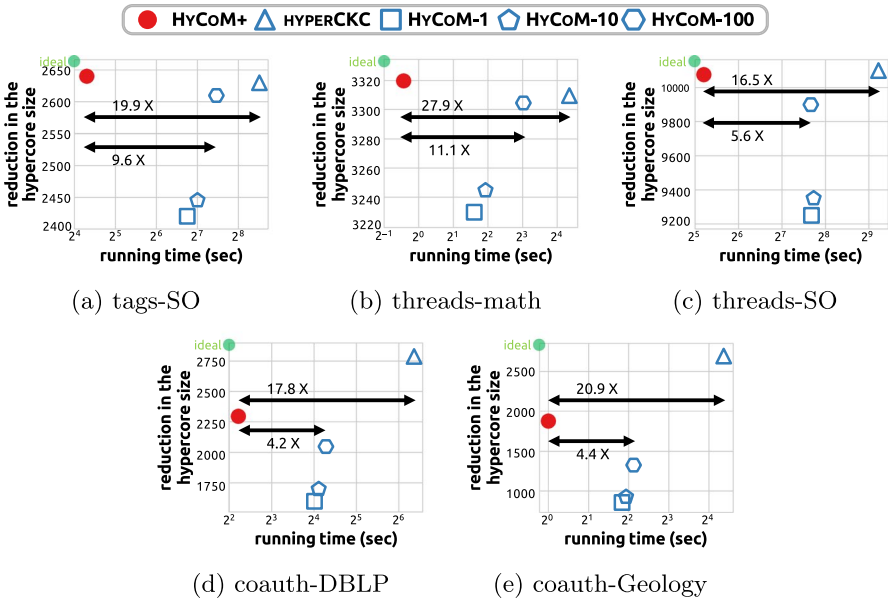


Fig. 13 HyCoM+ shows outstanding efficiency and comparable effectiveness. Given budget $b = 100$, we show the average running time over ten trials and the amount of reduction in the size of the (10, 0.6)-hypercore by different algorithms

graphs. Bonchi et al. (2019) generalized k-cores to (k, h)-cores. Specifically, they relaxed the node-degree condition by requiring each node in the (k, h)-cores to have at least k other nodes at a distance at most h, i.e., to have at least k h-hop neighbors. Dai et al. (2021) further investigated (k, h)-cores. Zhang et al. (2017b) generalized k-cores to (k, r)-cores, where they took the similarity between each pair of nodes w.r.t the attributes also into consideration. Victor et al. (2021) generalized k-cores by combining multiple node properties and introducing the notion of data depth. Chen et al. (2021) generalized triangle k-cores, i.e., k-trusses to (k, τ)-trusses, by taking the h-hop neighbors of each node into consideration, which is based on a similar idea of the (k, h)-cores (Bonchi et al. 2019). Sarıyüce and Pinar (2018) and Shin et al. (2018b) proposed to find dense substructures in bipartite graphs and tensors, respectively, by adapting the standard 'peeling' algorithm for obtaining the k-core. Gabert et al. (2021b) used k-nuclei, a generalization of k-cores and k-trusses, to detect dense substructures. Preti et al. (2021) generalized k-trusses to simplicial complexes.

Patterns in real-world hypergraphs. Do et al. (2020) proposed to convert hypergraphs into pairwise graphs where each k-subset of the node set is regarded as a node in the converted graph and found some pervasive structural patterns. Lee et al. (2020) defined hypergraph motifs that depict the connectivity patterns among each of three connected hyperedges. They revealed that the frequencies of hypergraph motifs are similar in hypergraphs in the same domain. Lotito et al. (2022) also studied hypergraph motifs using different definitions, and Kim et al. (2023) recently

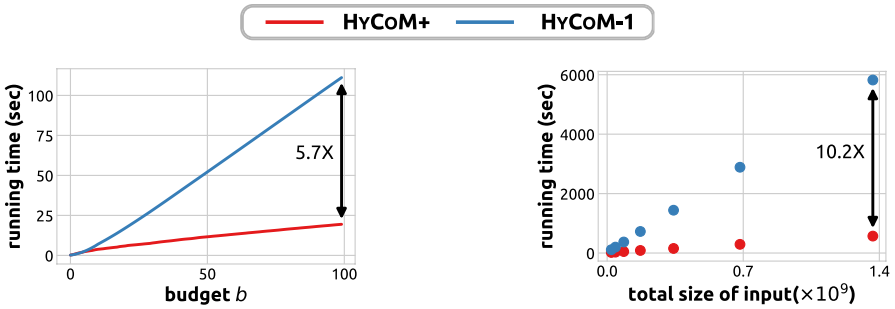


Fig. 14 HyCoM + has linear scalability w.r.t the budget and the hypercore size. On the left, we show the running time of HyCoM-1 and HyCoM+ with b increasing. On the right, we show the running time of HyCoM-1 and HyCoM+ while upscaling the tags-SO dataset ($b = 100$). HyCoM+ takes less than 10 minutes (574 s) when the total size of the input hypergraph is 1.37B (64× upscaled)

studied motifs in simplicial complexes.[8] Lee et al. (2021) defined the degree of overlaps of hyperedges and found some patterns related to the overlap. Moreover, temporal patterns have also been explored (Ko et al. 2022; Benson et al. 2018a, b). Structural properties (e.g., node centrality measures, the number of graph motifs involving each node) have been used as features of nodes in pairwise graphs (Cui et al. 2022; He et al. 2021). We believe that structural properties on hypergraphs can also be useful for feature representation (Arya et al. 2020), especially as inputs of hypergraph neural networks (Feng et al. 2019; Jiang et al. 2019; Liao et al. 2021; Bai et al. 2021; Huang and Yang 2021; Chien et al. 2021; Gao et al. 2022; Kim et al. 2022; Xia et al. 2022; Lee and Shin 2023; Wu and Ling 2023; Han et al. 2023).

Influential-node identification in hypergraphs. Besides the trivial degree centrality, a variety of node centrality measures (e.g., eigenvector centrality (Bonacich and Lloyd 2001) and coreness (Kitsak et al. 2010)) have been used to find influential nodes in pairwise graphs (Rossi et al. 2015), and some have been generalized to hypergraphs (Benson 2019). Overall, influential-node identification in hypergraphs is still underexplored, although some analyses have been made (Zhu et al. 2018a; Antelmi et al. 2021; Xie et al. 2023; Li et al. 2023). We provide an efficient and effective metric for practical use.

8 Conclusion

In this paper, we proposed the notion of (k, t)-hypercores and some related concepts (Definitions 5-7) for which we presented the theoretical properties (Propositions 1-2) and computation algorithms (Algorithms 1-3) with analyses (Theorems 1-3). Through extensive experiments on real-world hypergraphs, we presented interesting findings from various perspectives (Observations 1-5), including striking similarities of the hypercore structure within each domain. We also demonstrated

[8] Simplicial complexes can be seen as a special class of hypergraphs.

the usefulness of the proposed concepts in identifying influential nodes (Figs. 8-9), detecting dense substructures (Fig. 12), and revealing vulnerabilities (Figs. 13-14). For reproducibility, we made the code and datasets publicly available online (Bu et al. 2023).

A. Proofs

A.1 Proof of proposition 1

Proof Since H is finite, the number of subhypergraphs of H is also finite. Therefore, there exists *one* subhypergraph with maximal total size (which is possibly an empty hypergraph) where each node has degree at least k and at least t proportion of the constituent nodes remain in each hyperedge, completing the proof of existence. To show the uniqueness, suppose the opposite, and let $C^1 = (V^1, E^1)$ and $C^2 = (V^2, E^2)$ be two distinct (k, t)-hypercores of H. Then we consider the hypergraph $C' = (V', E')$ with $E' = \{e_i^1 \cup e_i^2 : i \in I_{E^1} \cup I_{E^2}\}$. Clearly, C' is a subhypergraph of H with a larger total size that satisfies the node-degree and hyperedge-fraction conditions, which contradicts the maximality and completes the proof. □

A.2 Proof of proposition 2

Proof Suppose that $C_{k,t_2}(H)$ is not a subhypergraph of $C_{k,t_1}(H)$. Then we take the union $C_{k,t_2}(H) \cup C_{k,t_1}(H)$ and we obtain a hypergraph that is strictly larger than $C_{k,t_1}(H)$ and satisfies the conditions of (k, t_1)-hypercore, which contradicts with the maximality, completing the proof. The second statement can be proved similarly. □

A.3 Proof of lemma 1

Proof This equivalence is immediate by two facts. First, for each node $v \in V$, the degree of v in H is equal to the degree of v in $G_{bp}(H)$. Second, for each hyperedge $e \in E$, the number of nodes in e is equal to the degree of e in $G_{bp}(H)$. With the above two facts, this equivalence immediately follows. □

A.4 Proof of lemma 2

Proof By Definition 5, when $t = 0$, the definition of $C_{k;t=0}(H)$ is the maximal subhypergraph of H where (1) every node in $C_{k,t=0}(H)$ has degree at least k and (2) at least two nodes remain in every hyperedge of $C_{k,t=0}(H)$. Such a definition exactly coincides with $\tilde{C}_{k;\ell=2}(H)$, completing the proof. □

A.5 Proof of lemma 3

We can understand the differences between $(k;\ell)$-hypercores and (k, t)-hypercores by two intuitions. When we obtain the $(k;\ell)$-hypercore of a given H with $\ell > 2$, all hyperedges of cardinality 2 are removed in the first place. Therefore, if we want to find an ℓ such that $\tilde{C}_{k;\ell} = C_{k,t}$ where $C_{k,t}$ contains any hyperedge of cardinality 2, the only possible ℓ value is $\ell = 2$. Since the threshold in the (k, t)-hypercore is proportional, it imposes different absolute cardinality thresholds for hyperedges of different sizes. On the contrary, the $(k;\ell)$-hypercore imposes the same absolute cardinality threshold for each hyperedge. We shall show two counterexamples from the two intuitions above.

Proof Consider $H = (V, E)$ with $E = \{\{1, 2\}, \{1, 3\}\{1, 2, 3, 4\}, \{1, 3, 4, 5, 6\}\}$. The $(k = 2, t = 3/4)$-hypercore of H consists of the hyperedges $\{\{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$, where the hyperedge $\{1, 3, 4, 5, 6\}$ is totally removed since only $3/5 < t = 3/4$ of the constituent nodes remain by the node-degree threshold $k = 2$. For the $(k;\ell)$-hypercore, the $(k = 2;\ell = 2)$-hypercore of H consists of the hyperedges $\{\{1, 2\}, \{1, 3\}, \{1, 2, 3, 4\}, \{1, 3, 4\}\}$; the $(k = 2;\ell = 3)$-hypercore of H consists of the hyperedges $\{\{1, 3, 4\}, \{1, 3, 4\}\}$; when $\ell \geq 4$, the $(k = 2;\ell)$-hypercore of H is empty, completing the proof.

We show another counterexample. Consider $H = (V, E)$ with $E = \{\{1, 2, 3, 4\}, \{1, 2, 5, 6\}, \{5, 6, 7, 8\}, \{3, 4, 9, 10, 11\}, \{1, 2, 3, 4, 5, 6, 7, 8\}\}$. The $(k = 3, t = 1/2)$ -hypercore of H consists of the hyperedges $\{\{1, 2\}, \{1, 2, 5, 6\}$, For the $(k;\ell)$ -hypercore, when $\ell = 2$, the $(k = 3;\ell = 2)$-hypercore of H consists of the hyperedges $\{\{1, 2, 3, 4\}, \{1, 2, 5, 6\}, \{5, 6\}, \{3, 4\}, \{1, 2, 3, 4, 5, 6\}\}$; when $\ell \geq 3$, the $(k = 3;\ell)$-hypercore of H is empty, completing the proof. □

Remark 1 Our proposed (k, t)-hypercore allows arbitrarily fine-grained adjustment since the value of t is continuous in [0, 1], while ℓ must be an integer. In real-world hypergraphs, many hyperedges are of cardinality 2. Therefore, the $(k;\ell)$-hypercore with $\ell > 2$ is significantly less meaningful than the (k, t)-hypercore since many hyperedges are not taken into consideration at all. See Tbl. 5 for the detailed number of hyperedges of different cardinality in each dataset we have used. See Figs. 8 and 9 for the performance of hypercoreness w.r.t $(k;\ell)$-hypercore to indicate the influence of nodes. Note again that the $(k;\ell = 2)$-hypercore is included in our proposed concept as the $(k, t = 0)$-hypercore. We observe that in most datasets, the $(k;\ell)$ -hypercores become less meaningful and fail to indicate the influence of nodes when ℓ becomes large, as expected.

A.6 Proof of theorem 1

Proof *Correctness.* The size of a hyperedge changes only when some node in $\mathcal{R}$ is removed from it, and the degree of a node changes only when some incident hyperedge is removed. Therefore, when Algorithm 1 ends, each node has degree at least k, otherwise it must have been included in $\mathcal{R}$ and removed, and each

hyperedge satisfies the hyperedge-fraction condition, otherwise it must have been removed. This implies that the output of Algorithm 1 satisfies both the node-degree and hyperedge-fraction conditions w.r.t H, k, and t. We now show the maximality. Suppose not, and let (v, e_i) be the first node-hyperedge pair that appears during the process of Algorithm 1 with $v \in e_i \in E(C_{k,t})$ but $v \notin e'_i \in E'$, where $C' = (V', E')$ is the returned hypergraph. This implies that v is removed from e, and thus v is included in $\mathcal{R}$ because its degree has been below k. However, by the definition of the (k, t)-hypercore and the assumption that (v, e_i) is the first pair, before the deletion, the degree of v is at least k, which completes the proof by contradiction.

Time complexity. We assume the input hypergraph has been loaded in the memory and thus do not count the complexity of loading the hypergraph. Checking the initial degrees (Line 1) takes $O(|V|)$. In the while loop, each node is added to the set of nodes to be removed at most once since each node is added exactly when its degree decreases from k to $k-1$. Therefore, this process takes $O(|V|)$. By checking the incident edges of each node in $\mathcal{R}$, we find all e'_is intersecting with $\mathcal{R}$, which takes $O(|\mathcal{R}|) = O(|V|)$. Hash tables are used to implement the sets. Before a hyperedge $e \in E$ is totally removed, at least $\max(\lceil t|e| \rceil, 2)$ nodes remain in it (otherwise it has been removed earlier), and thus it can be visited at most $|e| - \max(\lceil t|e| \rceil, 2) + 1$ times (because one node is removed at each time). This process takes $O(\sum_{e \in E}(|e| - \max(\lceil t|e| \rceil, 2) + 1)) = O(|E| + (1-t)\sum_{e \in E}|e|)$. Therefore, the total time complexity is $O(|V|) + O(|E| + (1-t)\sum_{e \in E}|e|) = O(|E| + (1-t)\sum_{e \in E}|e|)$. □

A.7 Proof of theorem 2

Proof *Correctness.* For each node v, the assignment of $c_t(v)$ happens only once when $v \in \mathcal{R}$, i.e., before its deletion. By Theorem 1, $c_t(v) = k-1$ implies that v is not in the (k, t)-hypercore but in the previous (k', t)-hypercore where each node has degree at least $k-1$, i.e., v is in the $(k-1, t)$-hypercore.

Time complexity. The values of k increases $O(c_t^*)$ times, thus the process in Lines 4 and 5 is repeated for $O(c_t^*)$ times and takes $O(c_t^*|V|)$. The assignment of t-hypercoreness of each node (Line 7) takes $O(|V|)$. As shown in the proof of Theorem 1, each hyperedge is visited at most $|e| - \max(\lceil t|e| \rceil, 2) + 1$ times before being deleted and each node is added to the set of nodes to be removed only once. Therefore, the remaining process takes $O(|V| + |E| + (1-t)\sum_{e \in E}|e|)$. □

A.8 Proof of theorem 3

Proof *Correctness.* For each node v, the assignment of $f_k(v)$ happens only once when $v \in \mathcal{R}$, i.e., before its deletion. By Theorem 1, $f_k(v) = t$ implies that v is in the (k, t)-hypercore with degree k and is in at least one hyperedge that is in the (k, t)-hypercore but not in any (k, t')-hypercore with $t' > t$. Therefore, v is not in any (k, t') -hypercore with $t' > t$.

Table 5 The number of hyperedges of different cardinality in each dataset. For each dataset, we list the number of hyperedges of each specific size. Specifically, in most datasets, a large number of hyperedges are of cardinality 2. We use E_s to denote the set of hyperedges of cardinality s, for each s

Dataset	$\|E\|$	$\|E_2\|$(%)	$\|E_3\|$(%)	$\|E_4\|$(%)	$\|E_5\|$(%)	$\|\bigcup_{s>5} E_s\|$(%)
Coauth-DBLP	2,169,663	693,364 (31.96)	667,302 (30.76)	419,431 (19.33)	205,965 (09.49)	183,601 (08.46)
Coauth-Geology	908,516	275,736 (30.35	227,950 (25.09)	159,509 (17.56	99,140 (10.91)	146,181 (16,09
NDC-classes	1047	297 (28.37)	121 (11.56%)	125 (11.94)	94 (08.98)	410 (39.16)
NDC-substances	6264	1130 (18.04)	745 (11.89)	535 (08.54%)	500 (07.98%)	3354 (53.54)
Contact-high	7818	5498 (70.32)	2091 (26.75)	222 (02.84)	7 (00.09)	0 (00.00)
Contact-primary	12,704	7748 (60.99)	4600 (36.21)	347 (02.73)	7 (00.09)	0 (00.00)
Email-Enron	1457	809 (55.53)	317 (21.76)	138 (09.47)	63 (04.32)	130 (08.92)
Email-Eu	24,399	12,753 (52.27)	4938 (20.24)	2294 (09.40)	1359 (05.57)	3055 (12.52)
Tags-ubuntu	145,053	28,138 (19.40)	52,282 (36.04)	39,158 (27.00)	25,475 (17.56)	0 (00.00)
Tags-math	169,259	25,253 (14.92)	63,870 (37.74)	50,892 (30.07)	29,244 (17.28)	0 (00.00)
Tags-SO	5,517,054	399,051 (07.23)	1,537,702 (27.87)	1,947,542 (35.30%)	1,632,759 (29.59)	0 (00.00)
Threads-ubuntu	115,987	88,301 (76.13)	21,621 (18.64)	4,560 (03.93)	1,117 (00.96)	388 (00.33)
Threads-math	535,323	319,601 (59.70)	142,065 (26.54)	49,198 (09.19)	16,402 (03.06)	8057 (01.51)
Threads-SO	8,589,420	5,210,916 (60.67)	2,102,208 (24.47)	787,701 (09.17)	299,172 (03.48)	189,423 (02.21)

Table 6 For each dataset, we report the information gain over degree, for each of the considered quantities: ℓ-hypercoreness with $\ell \in \{3, 4, 5\}$, neighbor-hypercoreness, and neighbor-degree-hypercoreness

Dataset	$\ell = 3$	$\ell = 4$	$\ell = 5$	Neighbor	Neighbor-degree
Coauth-DBLP	1.386	1.470	1.273	3.053	0.701
Coauth-Geology	1.239	1.394	1.313	3.262	0.447
NDC-classes	1.186	1.190	1.159	2.515	0.592
NDC-substances	1.221	1.459	1.607	3.974	0.975
Contact-high	1.456	0.893	0.173	1.685	1.231
Contact-primary	0.725	0.534	0.204	0.886	0.545
Email-Enron	1.228	1.176	1.114	1.340	1.020
Email-Eu	1.706	1.654	1.627	1.929	1.305
Tags-ubuntu	2.104	2.176	1.919	2.734	1.467
Tags-math	1.489	1.683	1.531	1.876	1.203
Tags-SO	2.530	3.213	3.007	4.046	2.219
Threads-ubuntu	0.967	0.451	0.207	1.253	0.409
Threads-math	1.510	1.061	0.616	2.146	0.626
Threads-SO	1.738	1.261	0.801	2.394	0.930

Time complexity. Recording the hyperedge sizes (Line 1) takes $O(|E|)$. By Theorem 1, computing $C_{k,0}$ (Line 2) takes $O(\sum_{e \in E} |e|)$. As shown in the previous proofs, the while loop (Lines 4 to 12) takes $O(|V| + |E| + \sum_{e \in E} |e|) = O(\sum_{e \in E} |e|)$. □

Details of datasets

In this section, we provide more details of the datasets used in our experiments.

- *Coauth-DBLP/Geology.* In these two *coauthorship* hypergraphs, each hyperedge represents a publication, and the constituent nodes of a hyperedge represent the authors of the corresponding publication.
- *NDC-classes/substances.* In these two hypergraphs from the *National Drug Code (NDC) Directory*, each hyperedge represents a drug (with its unique NDC code), and the constituent nodes of a hyperedge represent the class labels (for NDC-classes) or the ingredients (for NDC-substances) of the drug.
- *Contact-high/primary.* In these two *contact* hypergraphs, each hyperedge represents a group of interacting individuals (the constituent nodes) within a predetermined time period.

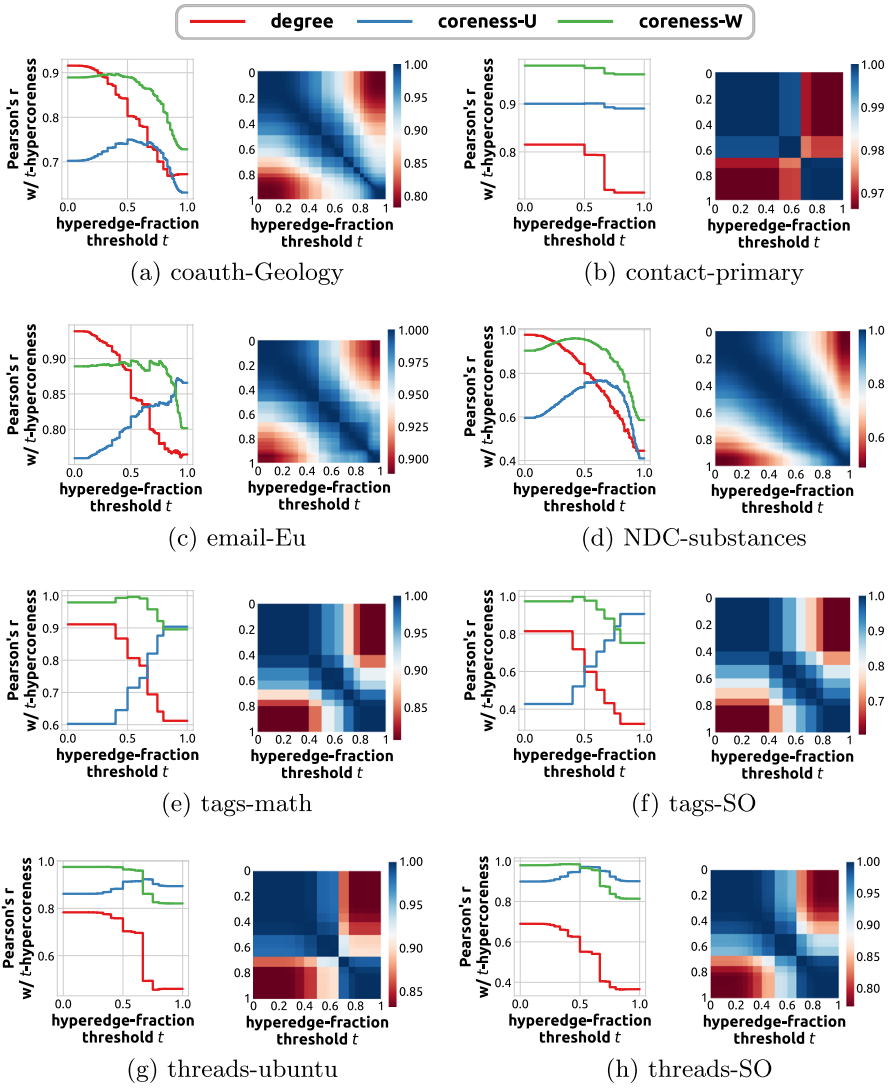


Fig. 15 Supplementary results for Fig. 6

- *Email-Enron/Eu.* In these two *email* hypergraphs, each hyperedge represents an email (possibly sent to multiple people individually at the same time), which contains the sender and all the receivers as its constituent nodes.
- *Tags-ubuntu/math/SO.* In these three *tags* hypergraphs from https://stackoverflow.com/, each node represents a tag, and each hyperedge represents a question, where each constituent node represents a tag applied to the question.
- *Threads-ubuntu/math/SO.* In these three *threads* hypergraphs also from https://stackoverflow.com/, each hyperedge represents a thread, where each constituent node represents a person that participates in it.

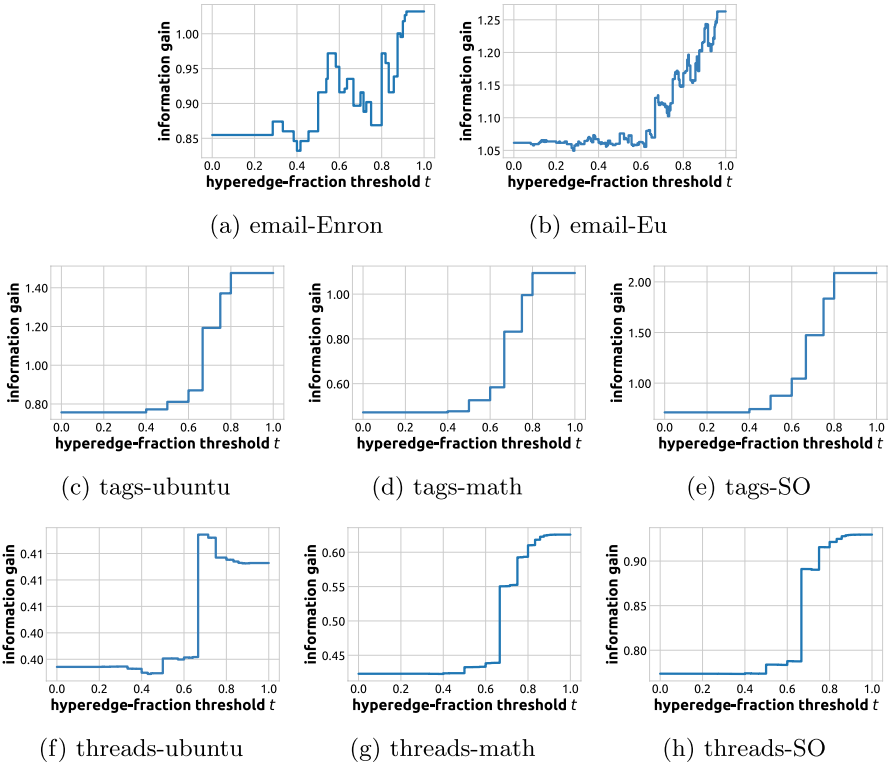


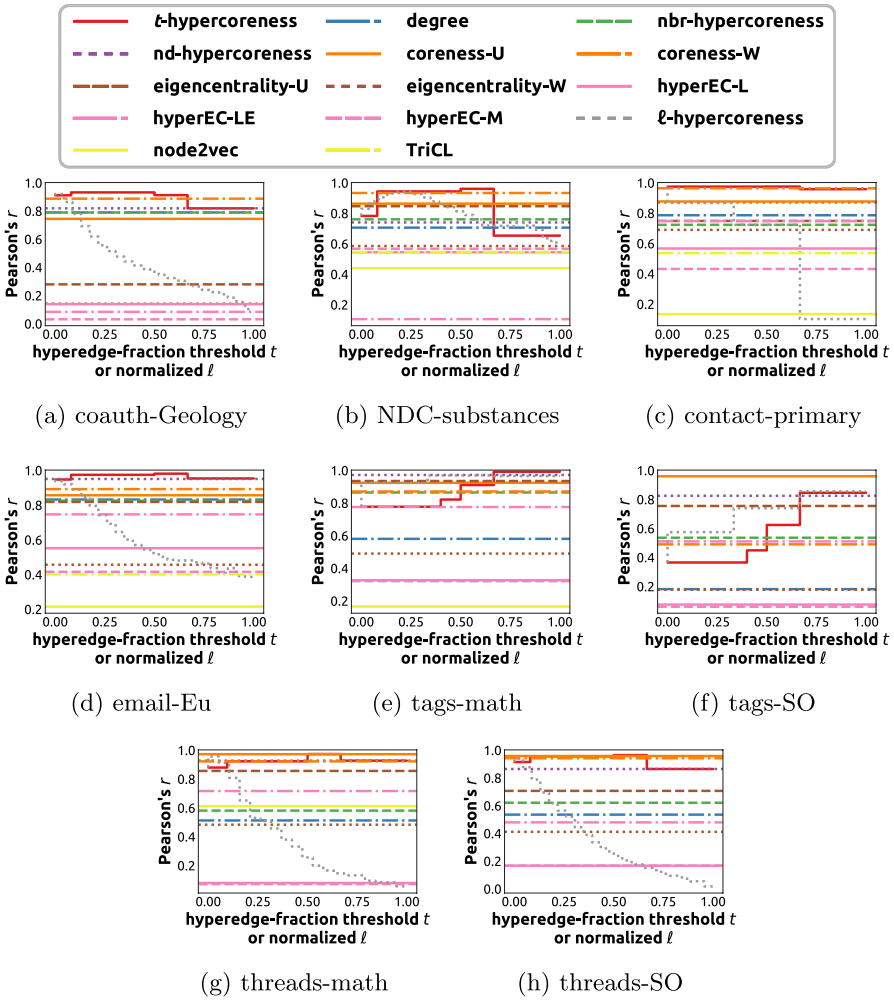
Fig. 16 Supplementary results for Fig. 7

We have used the preprocessed version of the datasets where each hyperedge consists of at most 25 nodes. In Table 5, we report the number of hyperedges of different cardinality in each dataset. Notably, on https://www.cs.cornell.edu/~arb/data/, the *full* version of the datasets, in which the cardinality of the hyperedges is not limited, is also available.

Additional experimental results

In this section, we provide additional experimental results supplementing the main text. In Fig. 15, we report the results regarding the statistical difference between t-hypercoreness and other centrality measures, as well as among t-hypercoreness with different t, on the datasets not covered in the main text. In Fig. 16, we report the results regarding the information gain over degree, on the datasets not covered in the main text. In Table 6, for each dataset, we report the information gain over degree for the following quantities: ℓ-hypercoreness with $\ell \in \{3, 4, 5\}$,[9]

[9] The case $\ell = 2$ is included in the proposed concept of t-hypercoreness with $t = 0$.



(a) coauth-Geology (b) NDC-substances (c) contact-primary

(d) email-Eu (e) tags-math (f) tags-SO

(g) threads-math (h) threads-SO

Fig. 17 Supplementary results for Fig. 8

neighbor-hypercoreness, and neighbor-degree-hypercoreness (Definitions. 9, 12, and 14). Notably, we do not claim that higher information gain is always better, since degree is still a reason measure by cohesiveness, and being too different from degrees can be negative as a cohesiveness measure. In Fig. 17, we report the results on influential-node identification, on the datasets not covered in the main text. In Table 7, we report the full results of the collapsed (k, t)-hypercore problem.

Table 7 Full results of the collapsed (k, t)-hypercore problem ($b = 100$). Time: running time (in seconds). Red.: reduction in the hypercore size

			HyperCKC		HyCoM-1		HyCoM-10		HyCoM-100		HyCoM+	
Dataset	k	t	Time	Red	Time	Red	Time	Red	Time	Red	tTme	Red
Coauth-DBLP	5	0	63.01	1176	15.80	680	16.44	729	17.50	812	5.37	1088
	5	0.2	63.38	1185	15.87	691	16.47	738	17.54	819	5.41	1091
	5	0.4	70.43	1378	18.21	831	18.87	897	19.91	986	6.28	1250
	5	0.6	113.38	2536	34.17	1675	35.00	1795	36.18	1969	11.83	2259
	5	0.8	130.19	6681	43.21	4487	43.90	5075	45.86	5785	13.88	5943
	5	1	87.75	7523	25.40	5914	26.10	6108	27.93	6752	8.70	6731
	10	0	50.42	1181	9.74	661	10.08	699	11.37	832	2.51	1008
	10	0.2	50.82	1195	9.74	676	10.21	716	11.46	852	2.55	1019
	10	0.4	56.94	1462	11.38	844	11.78	877	13.13	1001	3.08	1206
	10	0.6	75.03	2798	16.05	1621	16.45	1711	18.14	2050	4.30	2328
	10	0.8	0.24	2699	0.034	2742	0.051	2699	0.13	2699	0.13	2740
	20	0	33.32	1251	5.21	590	5.45	632	6.72	788	1.02	945
	20	0.2	33.56	1286	5.24	588	5.50	617	6.73	802	1.03	1000
	20	0.4	34.72	1558	5.14	749	5.41	784	6.71	1003	1.11	1206
	20	0.6	13.67	2817	2.06	1813	2.19	1975	3.17	2288	0.51	2218

Table 7 (continued)

Dataset	k	t	HyperCKC		HyCoM-1		HyCoM-10		HyCoM-100		HyCoM+	
			Time	Red	Time	Red	Time	Red	Time	Red	tTme	Red
Coauth-Geology	5	0	16.65	859	3.83	343	4.03	352	4.44	423	1.35	708
	5	0.2	16.79	877	3.84	344	4.05	352	4.46	429	1.36	720
	5	0.4	20.55	1120	4.60	386	4.86	403	5.26	518	1.77	865
	5	0.6	36.58	2004	9.22	660	9.58	686	10.07	914	3.29	1578
	5	0.8	32.32	5334	7.58	2992	7.92	3164	8.45	3986	2.53	3891
	5	1	4.67	7294	0.94	9498	0.94	9576	1.24	9536	0.47	6891
	10	0	12.81	856	2.65	316	2.77	328	3.14	423	0.64	720
	10	0.2	12.97	857	2.68	331	2.80	349	3.17	441	0.65	715
	10	0.4	16.81	1132	3.34	374	3.49	386	3.89	524	0.85	926
	10	0.6	21.51	2652	3.92	914	4.03	966	4.61	1311	1.03	1867
	10	0.8	0.0027	299	0.0023	388	0.0027	299	0.0027	299	0.03	388
	20	0	9.08	940	1.73	319	1.81	330	2.14	462	0.29	689
	20	0.2	9.18	977	1.74	334	1.81	343	2.22	476	0.30	713
	20	0.4	10.13	1470	1.78	528	1.85	550	2.23	742	0.32	1105
	20	0.6	0.031	459	0.0093	529	0.014	459	0.030	459	0.033	527

Table 7 (continued)

Dataset	k	t	HyperCKC		HyCoM-1		HyCoM-10		HyCoM-100		HyCoM+	
			Time	Red	Time	Red	Time	Red	Time	Red	tTme	Red
tags-SO	5	0	191.84	1042	42.03	930	55.66	962	108.93	1012	7.87	1042
	5	0.6	361.12	2103	111.72	1924	125.07	1968	184.97	2056	18.83	2102
	5	0.8	605.38	9669	185.49	9232	208.27	9306	296.96	9550	25.39	9663
	5	1	488.02	15,560	152.06	15,057	177.59	15,150	252.22	15,517	20.05	15,552
	10	0	202.64	1258	42.03	1126	55.65	1160	107.56	1203	8.26	1256
	10	0.6	380.46	2638	110.93	2423	124.09	2450	185.16	2609	18.99	2640
	10	0.8	609.48	11,253	179.73	10,919	203.65	11,035	291.75	11,227	24.31	11,244
	10	1	476.07	16,950	145.35	16,687	169.36	16,750	240.98	16,947	19.44	16,951
	20	0.4	213.17	1328	41.38	1198	54.94	1224	107.45	1313	8.49	1327
	20	0.6	394.77	2905	108.75	2743	122.387	2796	183.53	2885	18.99	2900
	20	0.8	586.81	11,431	166.97	11,147	190.06	11,254	275.55	11,439	22.98	11,438
	20	1	445.40	16,427	131.03	16,302	153.92	16,355	219.95	16,427	18.12	16,437

Table 7 (continued)

Dataset	k	t	HyperCKC		HyCoM-1		HyCoM-10		HyCoM-100		HyCoM+	
			Time	Red	Time	Red	Time	Red	Time	Red	tTme	Red
Threads-math	5	0	18.65	4549	2.89	4319	3.61	4395	7.04	4543	0.70	4552
	5	0.4	18.71	4564	2.92	4345	3.62	4419	7.12	4557	0.70	4567
	5	0.6	21.23	5339	3.93	5209	4.62	5259	8.35	5339	0.91	5336
	5	0.8	32.89	9366	8.31	9262	9.52	9296	14.53	9366	1.72	9359
	5	1	32.61	9741	8.59	9619	9.74	9670	14.54	9741	1.72	9731
	10	0	15.93	2708	2.40	2545	3.02	2595	6.33	2697	0.50	2708
	10	0.4	16.08	2722	2.45	2565	3.07	2615	6.39	2713	0.51	2708
	10	0.6	18.93	3318	3.26	3228	3.96	3241	7.65	3.316	0.67	3320
	10	0.8	22.86	5618	4.78	5522	5.71	5582	9.51	5618	0.94	5608
	10	1	21.49	5773	4.53	5644	5.32	5707	8.96	5768	0.87	5760
	20	0	13.06	1658	1.95	1573	2.49	1598	5.59	1661	0.37	1673
	20	0.4	13.25	1691	1.99	1594	2.55	1623	5.64	1690	0.38	1683
	20	0.6	16.12	2157	2.62	2099	3.29	2143	6.69	2160	0.49	2159
	20	0.8	12.80	3693	2.09	3627	2.63	3667	5.03	3692	0.43	3701
	20	1	10.83	3625	1.68	3589	2.12	3616	4.25	3635	0.36	3622

Table 7 (continued)

Dataset	k	t	HyperCKC		HyCoM-1		HyCoM-10		HyCoM-100		HyCoM+	
			Time	Red	Time	Red	Time	Red	Time	Red	tTme	Red
Threads-SO	5	0	634.38	13,739	183.04	11,927	186.79	12,209	205.62	13,083	41.64	13,736
	5	0.4	642.64	13,748	182.99	11,935	187.02	12,216	205.41	13,093	41.60	13,745
	5	0.6	653.52	15,050	224.97	13,462	227.72	13,659	240.00	14,536	49.05	15,046
	5	0.8	1409.66	31,383	522.59	29,317	529.38	30,198	565.72	31,133	112.59	31,336
	5	1	1592.06	36,082	572.42	34,044	580.25	34,663	638.44	35,913	133.73	36,013
	10	0	562.91	8425	148.98	7432	154.16	7532	170.28	8100	29.87	8417
	10	0.4	564.14	8449	149.99	7456	153.27	7558	170.84	8124	30.08	8433
	10	0.6	641.40	10,100	204.75	9210	205.99	9331	219.53	9879	38.35	10,065
	10	0.8	1105.59	21,202	340.25	19,490	346.77	19,819	375.53	21,091	67.12	21,099
	10	1	1131.47	23,234	342.87	22,056	349.27	22,311	382.53	23,058	66.88	23,115
	20	0	477.38	5155	120.34	4502	123.52	4621	140.99	4943	19.31	5124
	20	0.4	482.70	5177	123.59	4539	128.87	4645	145.46	4967	19.66	5154
	20	0.6	610.01	7060	165.86	6457	170.97	6591	187.87	6889	28.21	7019
	20	0.8	690.80	14,129	158.07	13,622	161.77	13,725	185.95	14,029	27.34	14,009
	20	1	533.83	14,734	109.77	14,259	113.39	14,358	134.04	14,614	20.65	14,461

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10618-023-00956-2.

Funding This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1C1C1008296) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Adamic LA, Lukose RM, Puniyani AR et al. (2001) Search in power-law networks. Phys Rev E 64(4):046–135

Albert R, Barabási AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47

Alvarez-Hamelin JI, Dall'Asta L, Barrat A, et al. (2006) Large scale networks fingerprinting and visualization using the k-core decomposition. In: NeurIPS

Alvarez-Hamelin JI, Dall'Asta L, Barrat A et al. (2008) K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. Netw Heterog Media 3(2):371

Antelmi A, Cordasco G, Spagnuolo C et al. (2021) Social influence maximization in hypergraphs. Entropy 23(7):796

Arafat NA, Khan A, Rai AK, et al. (2023) Neighborhood-based hypergraph core decomposition. PVLDB 16

Arya D, Gupta DK, Rudinac S, et al. (2020) Hypersage: Generalizing inductive representation learning on hypergraphs. arXiv preprint arXiv:2010.04558

Bai S, Zhang F, Torr PH (2021) Hypergraph convolution and hypergraph attention. Pattern Recogn 110(107):637

Batagelj V, Zaversnik M (2003) An $o(m)$ algorithm for cores decomposition of networks. In: arXiv

Benson AR (2019) Three hypergraph eigenvector centralities. SIAM J Math Data Sci 1(2):293–312

Benson AR, Abebe R, Schaub MT et al. (2018) Simplicial closure and higher-order link prediction. PNAS 115(48):E11221–E11230

Benson AR, Kumar R, Tomkins A (2018b) Sequences of sets. In: KDD

Blanco R, Lioma C (2012) Graph-based term weighting for information retrieval. Inf Retr 15(1):54–92

Bodó Á, Katona GY, Simon PL (2016) Sis epidemic propagation on hypergraphs. Bull Math Biol 78(4):713–735

Bonacich P, Lloyd P (2001) Eigenvector-like measures of centrality for asymmetric relations. Social Netw 23(3):191–201

Bonchi F, Khan A, Severini L (2019) Distance-generalized core decomposition. In: SIGMOD

Bu F, Lee G, Shin K (2023) Code, datasets, and supplementary materials. https://github.com/bokveizen/non-fragile-hypercore

Chein M, Mugnier ML (2008) Graph-based knowledge representation: computational foundations of conceptual graphs. Springer

Chen Z, Yuan L, Han L, et al. (2021) Higher-order truss decomposition in graphs. In: TKDE

Chien E, Pan C, Peng J, et al. (2021) You are allset: a multiset function framework for hypergraph neural networks. arXiv preprint arXiv:2106.13264

Corominas-Murtra B, Fuchs B, Thurner S (2014) Detection of the elite structure in a virtual multiplex social system by means of a generalised k-core. PLoS ONE 9(12):e11,2606

Cui H, Lu Z, Li P, et al. (2022) On positional and structural node features for graph neural networks on non-attributed graphs. In: CIKM

Dai Q, Li RH, Qin L, et al. (2021) Scaling up distance-generalized core decomposition. In: CIKM

Debnath S, Ganguly N, Mitra P (2008) Feature weighting in content based recommendation system using social network analysis. In: WWW

Do MT, Yoon Se, Hooi B, et al. (2020) Structural patterns and generative models of real-world hypergraphs. In: KDD
Feng Y, You H, Zhang Z, et al. (2019) Hypergraph neural networks. In: AAAI
Gabert K, Pinar A, Çatalyürek ÜV (2021a) Shared-memory scalable k-core maintenance on dynamic graphs and hypergraphs. In: IPDPSW
Gabert K, Pinar A, Çatalyürek ÜV (2021b) A unifying framework to identify dense subgraphs on streams: Graph nuclei to hypergraph cores. In: WSDM
Gao Y, Feng Y, Ji S, et al. (2022) Hgnn$^+$: General hypergraph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence
Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864
Han Z, Zheng X, Chen C, et al. (2023) Intra and inter domain hypergraph convolutional network for cross-domain recommendation. In: WWW
He T, Ong YS, Bai L (2021) Learning conjoint attentions for graph neural nets. In: NeurIPS
Hua QS, Zhang X, Jin H et al. (2023) Revisiting core maintenance for dynamic hypergraphs. IEEE Trans Parallel Distrib Syst 34:981–994
Huang J, Yang J (2021) Unignn: a unified framework for graph and hypergraph neural networks. arXiv preprint arXiv:2105.00956
Jiang J, Wei Y, Feng Y, et al. (2019) Dynamic hypergraph neural networks. In: IJCAI
Kim H, Ko J, Bu F, et al. (2023) Characterization of simplicial complexes by counting simplets beyond four nodes. In: WWW
Kim J, Oh S, Cho S, et al. (2022) Equivariant hypergraph neural networks. In: ECCV
Kitsak M, Gallos LK, Havlin S et al. (2010) Identification of influential spreaders in complex networks. Nat Phys 6(11):888–893
Ko J, Kook Y, Shin K (2022) Growth patterns and models of real-world hypergraphs. KAIS 64(11):2883–2920
Konstantinova EV, Skorobogatov VA (2001) Application of hypergraph theory in chemistry. Discret Math 235(1–3):365–383
Lee D, Shin K (2023) I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In: AAAI
Lee G, Shin K (2021) Thyme+: Temporal hypergraph motifs and fast algorithms for exact counting. In: ICDM
Lee G, Ko J, Shin K (2020) Hypergraph motifs: concepts, algorithms, and discoveries. PVLDB 13(11):2256–2269
Lee G, Choe M, Shin K (2021) How do hyperedges overlap in real-world hypergraphs? - patterns, measures, and generators. In: WWW
Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: Densification and shrinking diameters. TKDD 1(1):2–es
Li P, Wang H, Li K, et al. (2023) Influence without authority: Maximizing information coverage in hypergraphs. In: SDM
Liao X, Xu Y, Ling H (2021) Hypergraph neural networks for hypergraph matching. In: ICCV
Limnios S, Dasoulas G, Thilikos DM, et al. (2021) Hcore-init: Neural network initialization based on graph degeneracy. In: ICPR
Liu B, Yuan L, Lin X et al. (2020) Efficient (α, β)-core computation in bipartite graphs. VLDB J 29(5):1075–1099
Lotito QF, Musciotto F, Montresor A et al. (2022) Higher-order motif analysis in hypergraphs. Commun Phys 5(1):79
Lu Z, Zhu Y, Zhong M, et al. (2022) On time-optimal (k, p)-core community search in dynamic graphs. In: ICDE
Luo F, Li B, Wan XF, et al. (2009) Core and periphery structures in protein interaction networks. In: BMC bioinformatics
Luo Q, Yu D, Cai Z, et al. (2021) Hypercore maintenance in dynamic hypergraphs. In: ICDE
Luo Q, Yu D, Cai Z et al. (2022) Toward maintenance of hypercores in large-scale dynamic hypergraphs. VLDB J 32:1–18
Malliaros FD, Giatsidis C, Papadopoulos AN et al. (2020) The core decomposition of networks: theory, algorithms and applications. VLDB J 29:61–92

Mastrandrea R, Fournet J, Barrat A (2015) Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. PloS one 10(9):e0136497
McGlohon M, Akoglu L, Faloutsos C (2008) Weighted graphs and disconnected components: patterns and a generator. In: KDD
Mihalcea R, Radev D (2011) Graph-based Natural Language Processing and Information Retrieval. Cambridge University Press
Peng C, Kolda TG, Pinar A (2014) Accelerating community detection by using k-core subgraphs. In: arXiv
Peng Y, Zhang Y, Zhang W, et al. (2018) Efficient probabilistic k-core computation on uncertain graphs. In: ICDE
Preti G, De Francisci Morales G, Bonchi F (2021) Strud: Truss decomposition of simplicial complexes. In: WWW
Qu C, Tao M, Yuan R (2018) A hypergraph-based blockchain model and application in internet of things-enabled smart homes. Sensors 18(9):2784
Quinlan JR (1986) Induction of decision trees. Mach Learn 1:81–106
Rossi MEG, Malliaros FD, Vazirgiannis M (2015) Spread it good, spread it fast: Identification of influential nodes in social networks. In: WWW
Sarıyüce AE, Pinar A (2018) Peeling bipartite networks for dense subgraph discovery. In: WSDM
Seidman SB (1983) Network structure and minimum degree. Social Netw 5(3):269–287
Shin K, Eliassi-Rad T, Faloutsos C (2018) Patterns and anomalies in k-cores of real-world graphs with applications. KAIS 54(3):677–710
Shin K, Hooi B, Faloutsos C (2018) Fast, accurate, and flexible algorithms for dense subtensor mining. TKDD 12(3):1–30
Silva NB, Tsang R, Cavalcanti GD, et al. (2010) A graph-based friend recommendation system using genetic algorithm. In: CEC
Sinha A, Shen Z, Song Y, et al. (2015) An overview of microsoft academic service (mas) and applications. In: WWW
Sun B, Chan THH, Sozio M (2020) Fully dynamic approximate k-core decomposition in hypergraphs. TKDD 14(4):1–21
Torres L, Blevins AS, Bassett DS et al. (2021) The why, how, and when of representations for complex systems. SIAM Rev 63:435–485
Tudisco F, Higham DJ (2021) Node and edge nonlinear eigenvector centrality for hypergraphs. Commun Phys 4(1):1–10
Victor F, Akcora CG, Gel YR, et al. (2021) Alphacore: data depth based core decomposition. In: KDD
Vogiatzis D (2013) Influence study on hyper-graphs. In: AAAI Symposia
Wang K, Cao X, Lin X, et al. (2018) Efficient computing of radius-bounded k-cores. In: ICDE
Watts DJ, Strogatz SH (1998) Collective dynamics of small-world networks. Nature 393(6684):440–442
Wood CI, Hicks IV (2015) The minimal k-core problem for modeling k-assemblies. J Math Neurosci 5(1):1–19
Wu T, Ling Q (2023) Self-supervised heterogeneous hypergraph network for knowledge tracing. Inf Sci 624:200–216
Xia L, Huang C, Xu Y, et al. (2022) Hypergraph contrastive collaborative filtering. In: SIGIR
Xie M, Zhan X, Liu C, et al. (2023) An efficient adaptive degree-based heuristic algorithm for influence maximization in hypergraphs. Inf Process Manage 60(2):103161
Yang C, Wang R, Yao S, et al. (2022) Semi-supervised hypergraph node classification on hypergraph line expansion. In: CIKM
Yin H, Benson AR, Leskovec J, et al. (2017) Local higher-order graph clustering. In: KDD
Zhang C, Zhang F, Zhang W, et al. (2020) Exploring finer granularity within the cores: efficient (k, p)-core computation. In: ICDE
Zhang F, Zhang Y, Qin L, et al. (2017a) Finding critical users for social network engagement: the collapsed k-core problem. In: AAAI
Zhang F, Zhang Y, Qin L, et al. (2017b) When engagement meets similarity: efficient (k, r)-core computation on social networks. In: PVLDB
Zhang Y, Parthasarathy S (2012) Extracting analyzing and visualizing triangle k-core motifs within networks. In: ICDE
Zhu J, Zhu J, Ghosh S et al. (2018) Social influence maximization in hypergraph in social networks. TNSE 6(4):801–811
Zhu W, Chen C, Wang X, et al. (2018b) K-core minimization: an edge manipulation approach. In: CIKM

Zhu W, Zhang M, Chen C, et al. (2019) Pivotal relationship identification: the k-truss minimization problem. In: IJCAI

Zien JY, Schlag MD, Chan PK (1999) Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. IEEE Trans Comput Aided Des Integr Circuits Syst 18(9):1389–1399