# Finer-Grained Engagement in Hypergraphs

Qi Luo*, Dongxiao Yu*, Yu Liu*, Yanwei Zheng*, Xiuzhen Cheng*, Xuemin Lin†
*School of Computer Sciecnce and Technology, Shandong University, Qingdao, P.R. China
†Antai College of Economics and Management, Shanghai Jiao Tong Univeristy, Shanghai, P.R. China
{luoqi2018, yuliu}@mail.sdu.edu.cn, {dxyu, zhengyw, xzcheng}@sdu.edu.cn, xuemin.lin@sjtu.edu.cn

2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00039

***Abstract*—Vertex engagement has extraordinary significance for social resilience and network stability. There have been lots of existing work studying this fundamental problem in pairwise graphs, but in the more generalized hypergraphs, it has not been well explored, due to the great challenges of sparsity, complex connectivity and dynamicity of hypergraphs. In this work, we initialize the study of the vertex engagement problem in hypergraphs. Based on the observation that the engagement of vertices in hypergraphs needs to consider two critical parameters, group engagement and neighbor engagement, we propose a vertex engagement model integrating the merits of these two measures, called constrained core, to address the ineffectiveness and incomprehensiveness caused by just using a single engagement factor. By giving an algorithm for the constrained core decomposition, we show that the constrained core number of vertices can be computed in linear time. Furthermore, by showing a localized property of contained core, efficient maintenance algorithms for updating the constrained core number of vertices in dynamic hypergraphs are proposed, to avoid the large amount of redundant computations caused by the decomposition from scratch. Extensive experiments conducted on real-world hypergraphs well exhibit the effectiveness of our model and the efficiency of the proposed algorithms.**

***Index Terms*—Structural stability, vertex engagement, hypergraph**

## I. Introduction

The structural stability [1] of a network mirrors the ability of the network to maintain sustainable services. The removal of critical vertices may greatly destabilize the network [2]. For example, in social networks, the decision by a user to stay or leave in the network has a direct effect on his neighbors, as it leads to the departure or retention of his neighbors, creating a cascade reaction. Friendster was once a popular social application with 115 million users, but it has collapsed due to departure of critical users [3]. Therefore, how to measure and discover the critical vertices is crucial to maintain the stability of networks. Recent trends [4]–[6] have shown that vertex engagement is a significant indicator measuring the importance of vertices. It shows the vertex engagement can be captured by the networked coordination games [7], where engagement can be viewed as a network model based on direct-benefit effects. The more neighbors a vertex has, the more benefits the vertex will gain in the network [8], and the engagement of a vertex can be expressed as it reaches a Nash Equilibrium [4] by engaging with its neighbors.

Current studies on vertex engagement are all based on pairwise graphs, where the edges in pairwise graphs represent the relationship between two vertices. However, instead of pairwise relationship, the polyadic relationships are ubiquitous and widespread used in real-world scenarios. Many real-world relationships occur simultaneously between more than two entities [9]. For example, the publication of a paper is a collaborative relationship among multiple authors, and pairwise graphs cannot capture such higher-order collaboration relationships completely, as pairwise graphs cannot distinguish which paper is contributed from which authors in collaborating when no other attributed information is available. Hypergraphs [10] are a natural extension of graphs by allowing various sizes of edges. Formally, a hypergraph consists of a set of vertices and a set of hyperedges, where each hyperedge is a non-empty subset containing an arbitrary number of vertices. Polyadic relationships modeling by hypergraphs have been proven to be fruitful and indeed inevitable for challenging tasks [11].

Studying the vertex engagement can resolve other fundamental problems in hypergraphs, such as mining dense sub-hypergraphs [12] and constructing hierarchies of hypergraphs [13]. The dense subgraph metrics in pairwise graphs are significantly different in hypergraphs. For $k$-truss involves exponential numbers of hyper-triangles [14], and $k$-core does not reflect cohesiveness in sparse hypergraphs [15]. Depicting the vertex engagement in hypergraphs facilitates us finding suitable dense sub-hypergraph metrics. Further, constructing hierarchies among dense sub-hypergraphs helps us understand hypergraph structure and can be used for hypergraph visualization.

Although the hypergraph precisely models the ubiquitous polyadic relationship in real-world networks, the fundamental vertex engagement has not been well explored in this generalized graph. The modeling approaches based on Nash Equilibrium between vertices and its neighbors lose the generality due to the sparsity problems of hypergraphs. If the same groups of vertices are represented by a hypergraph and a pairwise graph, respectively, the number of hyperedges in the hypergraph is usually much smaller than the number of edges in the pairwise graph due to the polyadic relationship representation by the hyperedges. If the engagement modeling methods used in pairwise graphs are directly transferred to hypergraphs, numerous vertices will express consistent engagements. This results in the inability of modeling refined engagement, and

Yanwei Zheng is the corresponding author.

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1005900, in part by National Natural Science Foundation of China (NSFC) under Grant 62122042, and in part by Shandong University multidisciplinary research and innovation team of young scholars under Grant 2020QNQT017.

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00039

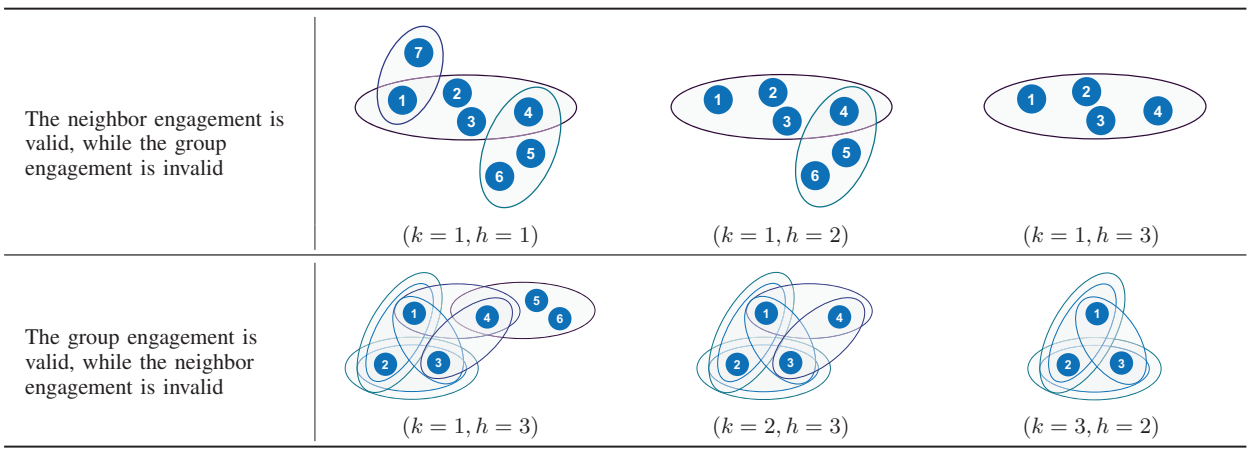



Fig. 1. Examples of constrained core, each hypergraph is labeled with a pair of values $(k,h)$ to represent the degree constraint $k$ and neighbor constraint $h$, respectively. In the first case, the degree constraint cannot distinguish the importance of vertices, while the neighbor constraint can distinguish the set of vertices with different neighbor constraint well. In the second case, the neighbor constraint cannot distinguish the importance of vertices, while the degree constraint can distinguish the set of vertices with different degree constraint well. Single usage of the number of neighbors or the degree may cause group engagement or neighbor engagement to be invalid, the constrained core well combines both engagement measurements.

the solutions in pairwise graphs cannot be applied effectively in hypergraphs anymore. Out of the above challenge, the study of the vertex engagement in hypergraphs faces the following two challenges.

- **Challenge 1.** The connections of vertices in hypergraphs is much more complex than that in pairwise graphs. In pairwise graphs, the number of edges connecting to neighbors and the number of neighbors are the same, while in hypergraphs they are two independent indicators. Moreover, the hypergraph can express higher-order information of vertex connection that is impossible expressed by pairwise graphs. For example, the group characteristic featured by vertices in the same hyperedge cannot be accurately demonstrated in pairwise graphs. Hence, it is hard to give a precise measure of the vertex engagement in hypergraphs just using a single metric as in pairwise graphs.
- **Challenge 2.** The dynamicity of hypergraphs are much more complex than pairwise graphs. In pairwise graphs, the insertion/deletion of one edge will directly increase/decrease one neighbor of the two vertices incident to the edge [16]–[18]. However, the insertion/deletion of a hyperedge may change the neighbors of more than two vertices by an uncertain value. For example, when two vertices are neighbors in multiple hyperedges, these two vertices may remain neighbors even after several hyperedges are deleted. In contrast, two vertices in a pairwise graph will no longer remain neighbors after the edge between them is deleted. Therefore, the update of the vertex engagement level in dynamic hypergraphs is significantly more complicated.

To address the above challenges, we delve into the local engaging properties of the vertices in hypergraphs. We observe that there is a difference between pairwise graphs and hypergraphs regarding the definition of degree. The degree of a vertex is equal to the number of neighbors of the vertex in a pairwise graph, while in a hypergraph, this is not always the case because hyperedges represent relationships of multiple vertices, and the degree of a vertex does not equal the number of neighbors. We also observe that the distribution of the number of neighbors of vertices in hypergraphs and the distribution of the degrees of the vertices have the same scale-free nature, as shown in Fig. 3 in our experiment. Based on the above observations, the engagement of vertices in hypergraphs needs to be performed from a more comprehensive perspective. Both the number of neighbors of the vertex and the number of hyperedges where the vertex is contained need to be considered simultaneously for modeling engagement in hypergraphs.

In particular, we consider two intuitive and important criteria in real-world hypergraphs: *group engagement* and *neighbor engagement*. The group engagement of a vertex is related with the degree of the vertex, which is hyperedge-oriented and exhibits potential on diversity of hyperedges, while the neighbor engagement is related with the associated vertices, which is vertex-oriented and neighbor-dependent. Because the number of neighbors and the degree of a vertex illustrates different characteristics in hypergraphs, they can model the engagement in different perspectives. When a vertex is contained in a few hyperedges but have many neighbors, the neighbor engagement can highlight its connection to the associated vertices, and when a vertex is contained in many hyperedges but only have a few neighbors, the group engagement can highlight its connection to the associated hyperedges.

To combine the superiority of group engagement and neighbor engagement, we propose a novel model, *constrained core*, which is defined as a maximal connected sub-hypergraph in which each vertex is contained in at least $k$ hyperedges (group engagement) and has at least $h$ neighbors (neighbor engagement). Each vertex is labeled with a *constrained core*

*number* $(k, h)$, indicating the maximal constrained core that the vertex can be in. The constrained core number can be a prefect combination of group engagement and neighbor engagement. Fig. 1 shows examples of constrained core, in cases where either group engagement or neighbor engagement alone would be ineffective, the constrained core always makes use of the merits of both metrics. The difference between $k$-core and constrained core is that $k$-core only considers group engagement. The group engagement and neighbor engagement in pairwise graphs are equivalent, thus constrained core has only one valid parameter in pairwise graphs and it is equivalent to $k$-core, while $k$-core in hypergraphs where only group engagement works is inferior to the constrained core. In event detection of social networks (vertices represent users and hyperedges represent interaction events), a hot event can be either a hyperedge containing many vertices or a hyperedge containing vertices of high degree. The $k$-core can only discover frequently interacting users, while the constrained core can also reveal hot events followed by active groups through the hyperedges. Searching for active groups and hot events that satisfy the demand can be realized by scaling the values of the degree constraint and the neighbor constraint.

As the neighbors of a vertex are determined by the hyperedges in which the vertex is contained, we set the prioritization order relationship between group engagement and neighbor engagement, and calculate the constrained core numbers of vertices with group engagement as the high priority. Based on the hierarchical structure of the constrained core, we propose a peel-like algorithm for constrained core decomposition. The algorithm computes the constrained core numbers of vertices by peeling off the hyperedges layer by layer and updating the degrees and neighbors of the vertices. Then, in dynamic hypergraphs, we show that the constrained core satisfies a localized property, which helps to bound the range of possible changes of constrained core number when the hypergraph changes. Therefore, instead of recalculating the constrained core number for all vertices, only a small amount of updates is needed for maintaining the constrained core number of vertices. We summarize our contributions as follows:

- We are the first to study the problem of vertex engagement in hypergraphs. A novel metric called constrained core number is proposed, which combines the superiority of group engagement and neighbor engagement to characterize the importance of vertices in hypergraphs.
- Based on the partial order dependence of group engagement and neighbor engagement, we propose a decomposition algorithm which is able to compute the constrained core number for each vertex in linear time.
- Based on the localized nature of constrained core, we give maintenance algorithms to update the constrained core number of vertices in dynamic hypergraphs. The maintenance algorithms can quickly identify the vertices that need to update their constrained core numbers, thus avoiding the large amount of redundant computations caused by the decomposition from scratch.
- Comprehensive experiments on 10 real-world hypergraphs are conducted to evaluate our models. We analyze the advantages and generalizability of our proposed constrained core, which is not only effective for a single engagement metric but also integrates the superiority of both types of engagements. Our constrained core maintenance algorithms are much more efficient than the recomputation approach in dynamic hypergraphs, and its computation time scales stably with the cardinality of the dynamic hyperedge.

**Roadmap.** The remaining sections of this paper are outlined as follows: Section II will provide a brief survey of research with respect to engagement. Section III will present the notations and descriptions in this paper. In Section IV, the decomposition algorithm of constrained core will be proposed. The theoretical basis and maintenance algorithms in dynamic hypergraphs will be given in Section V. Experiments and analysis of results on real-world hypergraphs will be reported in Section VI. Section VII will conclude the whole paper.

## II. Related Work

Recent studies of engagement argue that the Nash Equilibrium of a vertex depends on the measure of core number [4], [5], [8], which is the maximum $k$ of $k$-core a vertex belongs to, and $k$-core is a maximal-connected subgraph in which the degree of each vertex is no less than $k$. $k$-core has been shown to be a pivotal metric of engagement [2], [4]–[6], [19]–[22], and $k$-core was first introduced by [23] and [24].

Malliaros et al. [25] used the degeneracy property of the $k$-core to quantify the dynamics of participation in real social networks. Bhawalkar et al. [5] considered vertices that are not in the $k$-core in the problem of anchoring $b$ vertices to increase the size of the $k$-core. The problem of collapsed coreness [26] aims to protect critical nodes from attacks, and anchored coreness [27] aims to enhance key nodes for network stability. Zhang et al. [22], [28] investigated the collapsed $k$-core problem to find the key users of social network participation.

Recently, $k$-core has also been involved into hypergraphs, which was firstly proposed in [29]. The core decomposition in hypergraphs attracted a lot of research, including sequential algorithms [30], parallel algorithms [31], [32], and distributed algorithm [33]. There are also a few works on maintaining $k$-core in dynamic hypergraphs, including both exact algorithms [15] and approximate algorithms [34].

We can apply $k$-core to model the engagement of vertices in hypergraphs, but $k$-core in hypergraphs suffers from the problem of sparsity. As shown in our experiments (Fig. 7), the maximum core number in many large-scale hypergraphs is very small. Therefore, it is not appropriate to simply use the $k$-core directly ported from paired graphs as a measure of engagement, which would make it lose its functionality.

## III. Preliminaries

Considering an unweighted and undirected hypergraph $G = (V, E)$ on a finite set of vertices $V$, where $E \subset 2^V$ is a set of hyperedges. Each hyperedge $e \in E$ represents a set of

$|e|$ vertices that take interaction. We denote $n = |V|$ as the number of vertices, and $m = |E|$ as the number of hyperedges. We use $D_G(v)$ to denote the set of hyperedges containing $v \in V$ and $|D_G(v)|$ to denote the degree of $v$, i.e., the number of hyperedges containing $v$. The neighbors of a vertex $v \in V$, denoted as $N_G(v)$, is the set of vertices that share the same hyperedge with $v$. The cardinality of a hyperedge $e$, denoted as $|e|$, is the number of vertices contained in this hyperedge. Let $c_G$ denote the maximum cardinality of hyperedges in $G$, i.e., $c_G = \max\{|e||e \in E\}$, and $d_G$ the maximum degree of vertices is $G$, i.e., $d_G = \max\{|D_G(v)||v \in V\}$.

Table I summarizes the notations and descriptions. The subscript is omitted when the context is clear.

TABLE I
NOTATIONS

| Notion | Description |
|---|---|
| $G$=$(V, E)$ | an unweighted and undirected hypergraph |
| $V(H)$ | the vertices in sub-hypergraph $H$ |
| $E(H)$ | the hyperedges in sub-hypergraph $H$ |
| $H_{(k,h)}$ | a constrained core restricted by $(k, h)$ |
| $D_G(v)$ | the set of hyperedges containing $v$ in $G$ |
| $N_G(v)$ | the neighbors of $v$ in $G$ |
| $c_G$ | the maximum cardinality of hyperedges in $G$ |
| $d_G$ | the maximum degree of vertices in $G$ |
| $r_G$ | the maximum number of neighbors vertices in $G$ |
| $\mathit{CoCore}_G(v)$ | the constrained core number of $v$ in $G$ |
| $\mathit{CoCore}^k_G(v)$ | the degree constraint of $v$ in $G$ |
| $\mathit{CoCore}^h_G(v)$ | the neighbor constraint of $v$ in $G$ |

Based on the above definitions, we present the definition of constrained core.

*Definition 1 (**Constrained Core**):* Given a hypergraph $G = (V, E)$, a sub-hypergraph of $G$ denoted by $H_{(k,h)}$, is a *constrained core* with group engagement parameter $k$ and neighbor engagement parameter $h$, if it satisfies the following conditions:

(1) for each vertex $v \in H_{(k,h)}$, $D_{H_{(k,h)}}(v) \geq k$;

(2) for each $v \in H_{(k,h)}$, $|N_{H_{(k,h)}}(v)| \geq h$;

(3) $H_{(k,h)}$ is a maximal connected sub-hypergraph.

As there are two parameters in the definition of constrained core, there can be three different ways to define the constrained core number of vertices, i.e., $(i)$ first consider the group engagement of a vertex and then determine the neighbor engagement, $(ii)$ first consider the neighbor engagement and then determine the group engagement, and $(iii)$ consider the group engagement and the neighbor engagement simultaneously. We here take the first approach. This is because the interactions of vertices in hypergraphs are initiated by groups. It is the connection of vertices via hyperedges that allows vertices to be connected to their neighbors. In addition, it is feasible to determine the neighbors of a vertex by its groups, but not vice versa. The other approaches are left for future work.

*Definition 2 (**Constrained Core Number**):* Given a hypergraph $G = (V, E)$, the constrained core number $(k_v, h_v)$ of a vertex $v$, denoted as $\mathit{CoCore}(v)$, is defined as follows.

(1) $k_v$ is the maximum value of $k$ such that $v$ is contained in a constrained core, denoted as $\mathit{CoCore}^k(v)$;

(2) $h_v$ is the maximum value of $h$ such that $v$ is in a constrained core with group engagement parameter $k_v$, denoted as $\mathit{CoCore}^h(v)$.

The constrained core number of a hyperedge $e$, denoted by $\mathit{CoCore}(e) = (k, h)$, is the minimum constrained core number of the vertices contained in this hyperedge. We call $k$ as the degree constraint and $h$ as the neighbor constraint. The constrained core number of a hyperedge can be computed from the constrained core numbers of the vertices it contains. Suppose $\mathit{CoCore}(e) = (k_e, h_e)$, then

$$\begin{aligned} k_e &= \min\{\mathit{CoCore}^k(v)|v \in e\}, \\ h_e &= \min\{\mathit{CoCore}^h(v)|v \in e, \mathit{CoCore}(v)^k = k_e\}. \end{aligned} \tag{1}$$

**Problem Statement.** In this paper, we study the computation of constrained core numbers of vertices in a given hypergraph $G$. Specifically, we will focus on the following three problems.

(1) **Constrained Core Computation**: Computing the constrained cores for given $k$ and $h$ values;

(2) **Constrained Core Decomposition**: Computing constrained core numbers for vertices in $G$;

(3) **Constrained Core Maintenance**: Updating the constrained core numbers of vertices in dynamic hypergraphs avoiding fully recomputation.

## IV. Constrained Core Computation and Decomposition

In this section, we first propose the algorithm to compute the constrained cores for given thresholds. Then we implement an auxiliary index which records the number of neighbor relationships of vertices and is able to update the neighbors in a timely manner after the hypergraph changes. Based on this index, we propose a constrained core decomposition algorithm by peeling hyperedges.

### A. Computation of Constrained Core

Algorithm 1 shows the process of computing constrained cores. In the algorithm, the constrained cores are computed by iteratively deleting the vertices whose degrees are less than $k$ or number of neighbors are less than $h$. The final remaining sub-hypergraphs are constrained cores restricted by $(k, h)$. To update the set of neighbors of vertices in time, we add an auxiliary data structure, called neiCount, to store the number of occurrences of neighbors of each vertex.

**Analysis.** Each vertex in the sub-hypergraph $H$ returned by Algorithm 1 satisfies the degree constraint $k$ and the neighbor constraint $h$; otherwise, Algorithm 1 will proceed to delete the vertices that violate the two constraints. Suppose $H$ is not a maximal constrained core, then there is a sub-hypergraph $H'$ of $H$ which is a larger constrained core. But it will meet the condition in Line 3 to continue the deletion operation.

In Algorithm 1, the deletion of vertices and hyperedges require $O(n)$ and $O(m)$ time, respectively. The update of degree takes $O(d_G c_G)$ time, and the neighbor update takes

**Algorithm 1** Constrained core computation

**Input:** $G$=$(V, E)$, the degree constraint $k$, the neighbor constraint $h$
**Output:** constrained cores restricted by $(k, h)$

1: $H \leftarrow G$;
2: compute $D_H(v)$,$neiCount_v$ for $v \in V$;
3: **while** $\exists v \in V(H)$ such that $|neiCount_v| < h$ **or** $|D_H(v)| < k$ **do**
4: $V(H) \leftarrow V(H) \setminus \{v\}$;
5: $E(H) \leftarrow E(H) \setminus D_H(v)$;
6: update $D_H(u)$,$neiCount_u$ for $u \in N_H(v)$ ;
7: **return** $H = (V(H), E(H))$;

$O(d_G c_G^2)$ time where $d_G$ is the maximum degree and $c_G$ is the maximum cardinality. Thus, the algorithm can be correctly executed in $O(nd_G c_G + md_G c_G^2)$.

### B. Constrained Core Decomposition

Algorithm 2 shows the decomposition process of constrained core numbers. Because the degree constraint has the high priority, the constrained core decomposition can be computed on each level of degree constraint (Line 5). Firstly, we select the vertices with the degree constraint $k$ which can be computed by the $k$-core decomposition [15]. Then the minimum number of neighbors $h_{min}$ among all the vertices in current $k$-core is computed (Line 7). We remove all vertices whose neighbors are $h_{min}$ from $H$ (Lines 8-12). The above iteration is executed until $H$ is empty (Line 6). Finally, we return the constrained core numbers of all vertices and hyperedges (Line 14).

**Algorithm 2** Constraint core decomposition

**Input:** $G$=$(V, E)$
**Output:** *CoCore*

1: *CoCore* $\leftarrow \emptyset$;
2: initial $D_v$, $neiCount_v$ for $v \in V$;
3: **while** $G$ is not empty **do**
4: $k \leftarrow 1$;
5: $H \leftarrow k$-core of $G$; ▷ *k-core decomposition [15]*
6: **while** $H$ is not empty **do**
7: $h_{min} \leftarrow \min\{|neiCount_v| : v \in H\}$;
8: **while** $\exists v \in H$ such that $|D_H(v)| < k$ or $|neiCount_v| \leq h_{min}$ **do**
9: *CoCore*$(v) = (k, h_{min})$;
10: **for** $e \in D_H(v)$ **do**
11: *CoCore*$(e) = (k, h_{min})$;
12: delete $v$ and $D_H(v)$ from $H$;
13: $k \leftarrow k + 1$;
14: **return** $CoCore$;

**Analysis.** The algorithm computes the constrained core numbers by first computing the degree constraint, and in each iteration the degree constraint is computed from the $k$-core. For a given $k$-core, the first neighbor constraint $h_1$ in Line 7 is the minimum number of neighbors of all vertices in the $k$-core. After deleting the vertices with number of neighbors not greater than $h_1$, the second neighbor constraint $h_2$ is computed. Then the vertices just deleted are at most in the constrained core $H_{(k,h_1)}$. Recursively, we can get the constrained core number for all vertices. For each $k$-core, we perform an operation similar to Algorithm 1, then Algorithm 2 can be correctly executed in $O(k_{max} d_G c_G \cdot (n + mc_G))$, where $k_{max}$ is the maximum degree constraint.

## V. Constrained Core Maintenance

In this section, we propose the constrained core maintenance algorithms in dynamic hypergraphs where hyperedges are inserted or deleted.

Different from the pairwise graphs where the insertion or deletion of an edge can only make both the degree and the neighbor constraints change with the same value at most one, in hypergraphs, the insertion or deletion of a hyperedge can cause the degree of each vertex in this hyperedge to change by at most one, while the number of neighbors of each vertex can be possible to change by more than one or even not to change. Hence, it is much harder to identify the vertices whose constrained core numbers need to update after the hypergraph changes.

Subsequently, we first present some theoretical results that can provide necessary conditions for the neighbor constraint with which the vertices may update the constrained core number after the hypergraph changes, such that the searching range can be greatly reduced, to avoid redundant computations caused by decomposition from scratch. The maintenance algorithms for both the scenarios of hyperedge insertion and deletion are then given based on these necessary conditions.

### A. Theoretical Basis

At first, we introduce a localized property of constrained core, which gives a localized manner for computing the constrained core number, based on which we then show how to search the vertices which may update the constrained core numbers after inserting/deleting a hyperedge.

*1) Localized Property:* When updating the constrained core number in dynamic hypergraphs, too many search operations and global re-decomposition can both result in a lot of redundant computation. Therefore, we need to find an efficient way to calculate the constraint core number based only on the local information of the vertices to reduce unnecessary calculations.

We consider a vertex $v$. Assuming *CoCore*$(v) = (k, h)$, and $v$ is contained in a constrained core $H_{(k,h)}$. The constrained core number of each hyperedge and each vertex in $H_{(k,h)}$ is at least $(k, h)$ by the definition of constrained core. We observe that at least $k$ hyperedges contain $v$ and the degree constraint of each of those hyperedges is not less than $k$; $v$ has at least $h$ neighbors and the neighbor constraint of each of those neighbors is not less than $h$. We refer to the above properties as the localized nature of the constrained core. Formally,

$$\begin{aligned} CoCore^k(v) = \max\{k || \{e | e \in D(v), \\ CoCore^k(e) = k\}| \geq k\}. \end{aligned} \tag{2}$$

$$\begin{aligned} CoCore^h(v) = \max\{h | \\ |\{u | CoCore^h(u) \geq CoCore^h(v)\}| \geq h, \\ u \in e \setminus \{v\}, e \in D(v), \\ CoCore(e) = CoCore(v)\}. \end{aligned} \tag{3}$$

This localized property provides us an approach to update the constrained core numbers in dynamic hypergraphs, by estimating the change of the constrained core number of a vertex based only on the change of the constrained core numbers of its neighbors.

*2) Insertion of A Hyperedge:* We first consider the case that a hyperedge $e_0$ is inserted to a hypergraph $G$. At first, we introduce a concept of *Pre-CoCore Number* on the newly inserted hyperedge $e_0$, which can be seen as the initial value of the constrained core number of $e_0$. More importantly, with the involving of the pre-CoCore number, as shown later, the constrained core number update principle for the inserted hyperedge is the same as that for other hyperedges existing in the hypergraph.

*Definition 3 (**Pre-CoCore Number**):* After inserting a hyperedge $e_0$ into a hypergraph $G$, the pre-CoCore number of $e_0$, denoted by $\overline{CoCore}(e_0) = (k_0, h_0)$, is defined as

$$k_0 = \min\{CoCore^k(v) | v \in e_0\} \quad (4)$$

$$h_0 = \min\{CoCore^h(v) | v \in e_0, CoCore^k(v) = k_0\} \quad (5)$$

We observe that inserting $e_0$ into a hypergraph, the degree increment of the vertices in $e_0$ is one. It is easy to conclude that after $e_0$ is inserted, the degree constraint value of each vertex can change by at most one, and it is able to update this indicator using a similar approach for maintaining the core number in [15]. However, the insertion of $e_0$ can make the increment on the number of neighbors of the vertices in $e_0$ be any value between $[0, |e_0| - 1]$. This makes it very hard to identify the vertices which will change the neighbor constraint value and how many the change will be. Subsequently, we give necessary conditions for identifying the neighbor constraint values with which the vertices may change their constrained core numbers. These conditions can help greatly reduce the search space for finding potential vertices.

Notice that by the localized property of constrained core, computing the neighbor constraint of a vertex need to check all neighbors with the same degree constraint. Therefore, the update of the neighbor constraint and the update of the degree constraint are not completely independent. The update of the neighbor constraint needs to consider the change of related vertices' degree constraint. So in the subsequent discussion, we consider two cases depending on whether in the new hypergraph obtained by inserting $e_0$, the degree constraint value of $e_0$ is increased based on its pre-CoCore number.

Assume that $\overline{CoCore}(e_0) = (k_0, h_0)$. The following concept of *Head Vertex Set* will be used in the analysis, which defines the key vertices that affect the constrained core number of a hyperedge. Fig. 2 shows an example of the head vertex set.

*Definition 4 (**Head Vertex Set**):* For a hyperedge $e$, the Head Vertex Set of $e$, denoted by $HVS(e)$, is the set of vertices in $e$ such that each vertex $v \in e$ satisfies $CoCore(v) = CoCore(e)$. Each vertex in $HVS(e)$ is called a head vertex of $e$.

The update of degree constraint can be independent of the update of neighbor constraint, but not vice versa. Therefore, in the subsequent, we discuss the update of neighbor constraint in different cases according to the update of degree constraint.

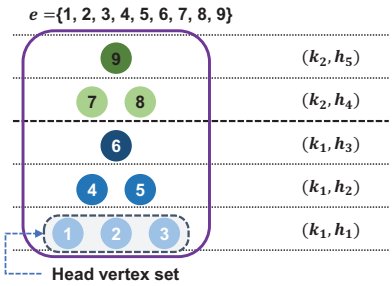



Fig. 2. Example of Head Vertex Set. The hyperedge $e$ contains 9 vertices, and the constrained core numbers corresponding to the vertices at each layer are labeled on the right, where $k_1 < k_2$, $h_1 < h_2 < h_3$, $h_4 < h_5$.

**Insertion Case 1: The degree constraint value of $e_0$ does not change.**

In this case, the insertion of $e_0$ does not change the degree constraint of any vertex, while the neighbor constraint values of vertices may be changed. We further divide the discussion into tow sub-cases depending on whether the insertion of $e_0$ makes the set of neighbors of at least one vertex in $HVS(e_0)$ keep unchanged.

**Insertion Case 1.1: The neighbor set of at least one vertex in $HVS(e_0)$ does not change.** As shown below, the constrained core numbers of vertices do not change after inserting $e_0$ in this sub-case.

*Theorem 1:* After a hyperedge $e_0$ is inserted into a hypergraph $G$ where $\overline{CoCore}(e_0) = (k_0, h_0)$, let $G_+ = G \cup \{e_0\}$. If $CoCore^k_{G_+}(e_0) = k_0$ and there exists at least one vertex $v$ in $HVS(e_0)$ such that $N_{G_+}(v) = N_G(v)$, then the constrained core number of all vertices keeps unchanged, i.e., for any vertex $u \in G$, $CoCore_{G_+}(u) = CoCore_G(u)$.

*Proof:* We consider vertices $v_1, v_2 \in HVS(e_0)$ and an arbitrary vertex $v_3 \in e_0 \setminus HVS(e_0)$. Assume that the neighbors of $v_1$ do not change and the number of $v_2$'s neighbors increase after the insertion of $e_0$, i.e., $N_{G_+}(v_1) = N_G(v_1)$ and $N_{G_+}(v_2) \neq N_G(v_2)$. Assume $CoCore_G(v_2) = (k_0, h_0)$ and $CoCore_{G_+}(v_2) = (k_0, h_0 + x)$ where $x > 0$. But by Eqt. 3, $CoCore_{G+}(v_1) = CoCore_G(v_1)$, which concludes that $CoCore_{G_+}(e_0) = (k_0, h_0)$ by Eqt. 1. This means the constrained core number of $v_2$ cannot change by Eqt. 1, which is a contradiction. ■

**Insertion Case 1.2: The neighbor sets of all vertices in $HVS(e_0)$ change.** In this case, the neighbor constraint values of vertices may change due to two reasons: $(i)$ the number of neighbors of this vertex is increased; $(ii)$ the neighbor constraint values of this vertex's neighbors are increased.

*Theorem 2:* After a hyperedge $e_0$ is inserted into a hypergraph $G$ where $\overline{CoCore}(e_0) = (k_0, h_0)$, let $G_+ = G \cup \{e_0\}$. If $CoCore^k_{G_+}(e_0) = k_0$ and the neighbor sets of all vertices in $HSV(e_0)$ are changed, then a vertex $w$ may change its constrained core number only if $CoCore^k_G(w) = k_0$ and

$\mathit{CoCore}^h_G(w) \in [h_0, h_{UB}]$, where

$$\begin{aligned} h_{UB} = &\min\{\hat{h}_{v_i} | v_i \in \mathit{HVS}(e_0), \\ &\hat{h}_{v_i} = \arg\max_h |\{u | N_{G_+}(u) \geq N_{G_+}(v_i)\}| \geq h, \\ &u \in e \setminus \{v\}, e \in D_{G_+}(v), \\ &\mathit{CoCore}^k(e) = k_0\}. \end{aligned} \tag{6}$$

*Proof:* When the degree constraint of $e_0$ does not increase, the degree constraint of all vertices in $G_+$ will keep the same as proved in [15]. We next analyze the neighbor constraint change in the following three cases.

Case 1: $\mathit{CoCore}^k_G(w) = k_w \neq k_0$. For vertices with degree constraint $k_w < k_0$, the number of neighbors with neighbor constraint greater than their own neighbor constraint does not increase; for vertices with degree constraint $k_w > k_0$, the constrained core $H_{(k_w,\cdot)}$ does not change, and the neighbor constraint of all vertices with degree constraint $k_w$ does not change consequently.

Case 2: $\mathit{CoCore}_G(w) = (k_0, h_1)$, where $h_1 < h_0$. Assume we delete $e_0$ from $G_+$, since $\mathit{CoCore}^h_{G_+}(e_0) \geq h_0 > h_1$ and the deletion of $e_0$ does not change the constrained core numbers of vertices in the constrained core $H_{(k_0,h_1)}$, so there must exist another $H_{(k_0,h'_1)}$ where $h'_1 \geq h_1$ containing $w$, which contradicts the definition that the constrained core is maximal.

Case 3: $\mathit{CoCore}_G(w) = (k_0, h_2)$, where $h_2 > h_{UB}$. Assume the neighbor constraint of $w$ is increased to $h'_2 > h_2$, then there is a constrained core $H_{(k_0,h'_2)}$ containing $w$ in $G_+$, and each neighbor of $w$ has neighbor constraint at least $h'_2$. By Eqt. 6, there are at most $h_{UB}$ neighbors of $w$ each of which has neighbor constraint at least $h_{UB}$. However, $h_{UB} < h_2 < h'_2$, which is a contradiction. ■

**Insertion Case 2: The degree constraint value of $e_0$ is increased.**

*Theorem 3:* After a hyperedge $e_0$ is inserted into a hypergraph $G$ where $\overline{\mathit{CoCore}}(e_0) = (k_0, h_0)$, let $G_+ = G \cup \{e_0\}$. If $\mathit{CoCore}^k_{G_+}(e_0) = k_0 + 1$, a vertex $w$ may change its constrained core number only if $\mathit{CoCore}^k_G(w) = k_0 + 1$ and $\mathit{CoCore}^h_G(w) \leq h_{UB}$, where

$$\begin{aligned} h_{UB} = &\min\{\hat{h}_{v_i} | v_i \in e_0, \\ &\hat{h}_{v_i} = \arg\max_h |\{u | N_{G_+}(u) \geq N_{G_+}(v_i)\}| \geq h, \\ &\{u | u \in e \setminus \{v\}, e \in D_{G_+}(v), \\ &\mathit{CoCore}^k_G(e) = k_0 + 1\}\}. \end{aligned} \tag{7}$$

*Proof:* We prove the theorem in the following three cases:

Case 1: $\mathit{CoCore}^k_G(w) = k_w < k_0 + 1$. The neighbors of $w$ are still neighbors when the degree constraint of these neighbors increase. Thus, the neighbor constraint of $w$ will not change.

Case 2: $\mathit{CoCore}^k_G(w) = k_w > k_0 + 1$. The degree constraint of $e_0$ is $k_0 + 1$, since $e_0$ is not in the constrained core $H_{(k_w,\cdot)}$. Thus, the constrained core numbers of vertices in $H_{(k_w,\cdot)}$ remain the same as $G$.

Case 3: $\mathit{CoCore}_G(w) = (k_0 + 1, h_w)$, if $h_w > h_{UB}$, assume the neighbor constraint of $w$ has decreased as $h'_w < h_w$, then there is a constrained core $H_{(k_0+1,h'_w)}$ containing $w$ in $G_+$, and each neighbor of $w$ with neighbor constraint at least $h'_w$. By Eqt. 6, there are at most $h_{UB}$ neighbors of $w$ and each of which has at least $h_{UB}$ neighbors. However, $h_{UB} < h_2 < h'_2$, which is a contradiction. ■

*3) Deletion of A Hyperedge:* We also denote the hyperedge deleted as $e_0$. In the following, we assume that the degree constraint update has been accomplished. The discussion is divided into two cases similar to the insertion scenario.

**Deletion Case 1: The degree constraint value of $e_0$ does not change.**

Using a similar approach as in the proof of Theorem 2, we can get the following result.

*Theorem 4:* After a hyperedge $e_0$ is deleted from a hypergraph $G$ where $\mathit{CoCore}(e_0) = (k_0, h_0)$, let $G_- = G \setminus \{e_0\}$. If $\mathit{CoCore}^k_{G_-}(e_0) = k_0$, then a vertex $w$ may change its constrained core number only if $\mathit{CoCore}^k_G(w) = k_0$ and $\mathit{CoCore}^h_G(w) \in [h_{LB}, h_0]$, where

$$\begin{aligned} h_{LB} = &\min\{\bar{h}_{v_i} | v_i \in \mathit{HVS}(e_0), \\ &\bar{h}_{v_i} = \arg\max_h |\{u | \mathit{CoCore}^h(u) \geq h_0\}| \geq h, \\ &u \in e \setminus \{v\}, e \in D_{G_-}(v), \\ &\mathit{CoCore}(e) = \mathit{CoCore}(e_0)\}. \end{aligned} \tag{8}$$

**Deletion Case 2: The degree constrained value of $e_0$ is increased.**

Different from Deletion Case 1, the degree constraint values of vertices may be decreased to from $k_0$ to $k_0 - 1$ in this case, as shown in [15]. Hence, it needs to consider the neighbor constraint values of vertices whose degree constraint value is $k_0$ and $k_0 - 1$. Using a similar approach for proving Theorem 3, we can get the following result.

*Theorem 5:* After a hyperedge $e_0$ is deleted from a hypergraph $G$ where $\mathit{CoCore}(e_0) = (k_0, h_0)$, let $G_- = G \setminus \{e_0\}$. If $\mathit{CoCore}^k_{G_-}(e_0) = k_0 - 1$, then a vertex $w$ may change its constraint core number only if $\mathit{CoCore}^k_G(w) \in [k_0 - 1, k_0]$ and $\mathit{CoCore}^h_G(w) \in [h_{LB}, h_0]$.

### *B. Maintenance Algorithms*

In subsequence, we present the incremental and decremental algorithms with respect to the insertion and deletion of a hyperedge to maintain the constrained core number of vertices, respectively. The insertion/deletion of a vertex can be decomposed into the insertion/deletion of the set of hyperedges containing that vertex, which is an opportunity to be discussed in future work.

The two main challenges of designing maintenance algorithms in hierarchical graphs are (1) bounding the change hierarchies and (2) identifying the change vertices [16]–[18]. The previous maintenance algorithms [15], [35] only need to identify one specific hierarchy of change and mainly focus on searching the set of vertices to be updated in the specific hierarchy. In contrast, our model is a finer-grained model, and the change may occur on multiple hierarchies as shown before. Consequently, it requires a large amount of computations on bounding updated hierarchies, and hence a careful trade-off

**Algorithm 3** Incremental Algorithm

**Input:** $G$=$(V, E)$, $\mathit{CoCore}_G(v)$, $e_0$
**Output:** *CoCore* for $G \cup \{e_0\}$
1: $G_+ \leftarrow G \cup \{e_0\}$;
2: $(k_0, h_0) \leftarrow \overline{\mathit{CoCore}}(e_0)$; ▷ *Eqt. 3*
3: $\mathit{CoCore}_G(e_0) \leftarrow (k_0, h_0)$;
4: $\mathit{CoCore}_{G_+} \leftarrow \mathit{CoCore}_G$;
5: update degree constraint in $\mathit{CoCore}_{G_+}$;
6: **if** $\mathit{CoCore}^k_{G+}(e_0) = k_0$ **then**
7: **if** $\exists v \in \mathit{HVS}(e_0)$ such that $N_{G_+}(v) = N_G(v)$ **then**
8: **return** $\mathit{CoCore}_{G_+}$; ▷ *Theorem 1*
9: compute $h_{UB}$ by Eqt. 6;
10: update $\mathit{CoCore}_{G_+}$ from $(k_0, h_0)$ to $(k_0, h_{UB})$;
11: **else**
12: compute $h_{UB}$ by Eqt. 7;
13: update $\mathit{CoCore}_{G_+}$ from $(k_0 + 1, h_0)$ to $(k_0 + 1, h_{UB})$;
14: **return** $\mathit{CoCore}_{G_+}$;

**Algorithm 4** Decremental Algorithm

**Input:** $G$=$(V, E)$, $\mathit{CoCore}_G(v)$, $e_0$
**Output:** *CoCore* for $G \setminus \{e_0\}$
1: $(k_0, h_0) \leftarrow \mathit{CoCore}_G(e_0)$;
2: $G_- \leftarrow G \cup \{e_0\}$;
3: $\mathit{CoCore}_{G_-} \leftarrow \mathit{CoCore}_G$;
4: update degree constraint in $\mathit{CoCore}_{G_-}$;
5: **if** $\mathit{CoCore}^k_{G-}(e_0) = k_0$ **then**
6: **if** $\forall v \in \mathit{HVS}(e_0)$ such that $N_{G_-}(v) = N_G(v)$ **then**
7: **return** $\mathit{CoCore}_{G_-}$;
8: compute $h_{LB}$ by Eqt. 8;
9: update $\mathit{CoCore}_{G_-}$ from $(k_0, h_{LB})$ to $(k_0, h_0)$;
10: **else**
11: compute $h_{LB}$ by Eqt. 8;
12: update $\mathit{CoCore}_{G_-}$ from $(k_0 - 1, h_{LB})$ to $(k_0 - 1, h_0)$ and from $(k_0, h_{LB})$ to $(k_0, h_0)$;
13: **return** $\mathit{CoCore}_{G_-}$;

between re-decomposing update hierarchies and searching for update vertices on the updated hierarchies is necessary. As shown above, the computation of bounding the range of the constrained core numbers with which the vertices may update is very complicated. If we further distinguish the possible change of each vertex's constrained core number specifically, it needs too many computations. So in our algorithm, we utilize a more efficient approach which performs the re-decomposition process directly to compute the constrained core numbers of vertices whose constrained core numbers are in the computed range. The experimental results in the next section illustrate that our approach is effective on real-world hypergraphs.

Algorithm 3 shows the process of incremental constrained core maintenance. We consider the scenario that a hyperedge $e_0$ is inserted into a hypergraph $G$, where the constrained core numbers have been given by Algorithm 2. At the beginning of the algorithm, the *pre*-CoCore$(e_0)$ is computed (Line 2). Then the degree constraints are updated by the incremental degree-based core maintenance algorithm in [15] (Line 5). If the degree constraint of $e_0$ has not increased, then we determine if there is an update of the vertex's neighbors. If there is no update of the neighbors, then all the neighbor constraints will not be updated (Line 7-8). If there is an update of the neighbors, we compute the upper and lower bounds of possible changes of neighbor constraint by Theorem 2, and reformulate the neighbor constraint of the vertices using the constrained core decomposition algorithm starting with $(k_0, h_{LB})$ and ending with $(k_0, h_{UB})$ (Lines 9-10). If the degree constraint of $e_0$ has increased, then we preform this recalculation process using the bounds in Theorem 3 (Lines 12-13).

Algorithm 4 shows the process of decremental constrained core maintenance which is similar to Algorithm 3, thus the description is omitted.

**Running Example.** If a hyperedge $e$ in Fig. 2 is inserted, we first calculate that $k_1$ is increased or not. Then we compute $h_{UB}$ by vertices 1, 2, and 3, and execute local decomposition to update neighbor constraint in range [$h_1$,$h_{UB}$]. If $e$ in Fig. 2 is deleted, we first calculate $k_1$ is decreased or not. Then we compute $h_{LB}$ by vertices 1, 2, and3, and execute local decomposition to update neighbor constraint in range [$h_{LB}$,$h_1$].

## VI. Experiments

In this section, we evaluate the effectiveness of the proposed constrained core, the efficiency of our decomposition algorithm, and the stability of the presented maintenance algorithms.

### A. Datasets and Setting

**Datasets.** We use 10 real-world hypergraphs[1] and Table II shows the statistics of these hypergraphs. CongressBill (CoBi) is extracted from co-sponsors of legislative bills put forth in both the House of Representatives and the Senate, in which vertices are US Congresspersons and hyperedges are comprised of the sponsor. Drug Abuse Warning (DrAb) is the categories of patients' drug combination. The hyperedges of Mathoverflow-Answers (MaAn) are sets of questions answered by users on Math Overflow. Microsoft Academic Graph (MiAc) and Coauth-MAG-History (CoMH) are subsets of the Microsoft Academic Graph where vertices are authors, hyperedges correspond to publications from those authors. Walmart-Trips (WaTr) is a hypergraph where the hyperedges are sets of co-purchased products at Walmart. Threads-Ask-Ubuntu (ThAU) is derived from threads on Ask Ubuntu posts, Threads-Math-Sx (ThMS) is derived from threads on math Stack Exchange posts. Trivago-Clicks (TrCl) is derived from the ACM RecSys Challenge 2019[2], where vertices are accommodations and hyperedges are sets of accommodations. Coauth-DBLP (CoDB) is derived from the computer science online bibliography DBLP.

**Algorithms**. We compare the constrained core number with the degree-based core number and the neighbor-based core number, which use only a single metric, degree or neighbor

[1]All the datasets can be downloaded in ARB [36].
[2]https://recsys.acm.org/recsys19/challenge/

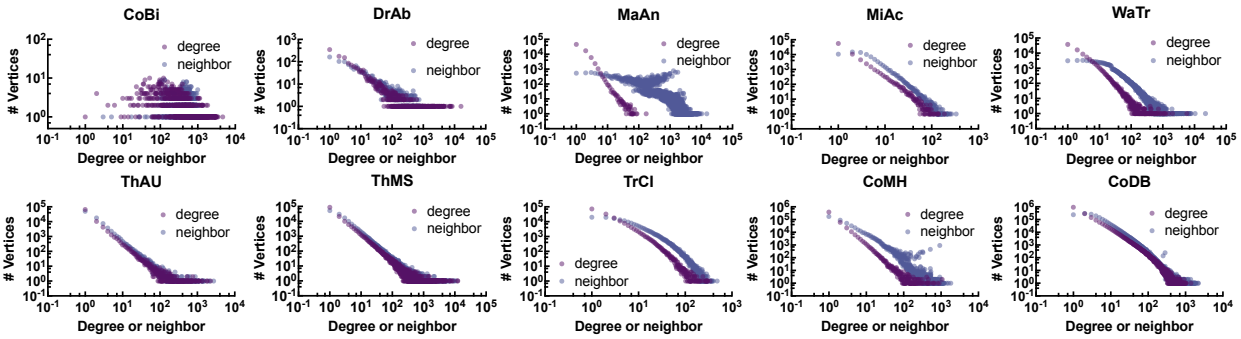
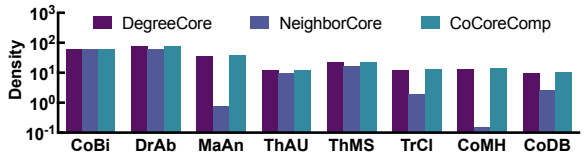
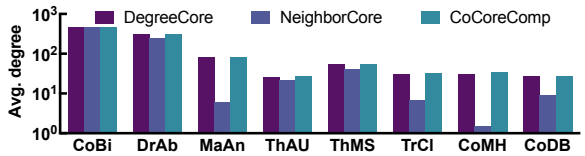



Fig. 3. Distribution of the degree and the number of neighbors of vertices.




(a) Comparision of density




(b) Comparision of average degree

Fig. 4. Comparison of the three types of cores.

TABLE II
THE STATISTICS OF HYPERGRAPHS.

| Name | $\|V\|$ | $\|E\|$ | $deg_{max}$ | $c_{max}$ | $\bar{deg}$ | $\bar{c}$ |
|---|---|---|---|---|---|---|
| CoBi | 1718 | 105929 | 4620 | 25 | 465.27 | 7.55 |
| DrAb | 2109 | 87034 | 16090 | 22 | 162.53 | 3.94 |
| MaAn | 73851 | 5446 | 173 | 1784 | 1.78 | 24.19 |
| MiAc | 80198 | 51889 | 187 | 25 | 2.25 | 3.48 |
| WaTr | 88860 | 69906 | 5733 | 25 | 5.18 | 6.59 |
| ThAU | 90054 | 117764 | 2247 | 14 | 3.01 | 2.30 |
| ThMS | 153806 | 563710 | 12403 | 21 | 9.46 | 2.58 |
| TrCl | 172738 | 233202 | 339 | 85 | 4.21 | 3.12 |
| CoMH | 503868 | 308934 | 1077 | 925 | 1.78 | 2.90 |
| CoDB | 1836596 | 2955129 | 1399 | 280 | 5.22 | 3.24 |

number, to model engagement in hypergraphs. For the self-completeness of this paper, we present the concept of *degree-based core* to depict the group engagement, and the concept of *neighbor-based core* to depict the neighbor engagement.

*Definition 5 (**Degree-based Core**):* A degree-based core constrained by $k$ is a maximal connected sub-hypergraph $H = (V', E')$ of $G = (V, E)$, such that $\forall v \in V'$, $|D_H(v)| \geq k$.

*Definition 6 (**Neighbor-based Core**):* A neighbor-based core constrained by $h$ is a maximal connected sub-hypergraph $H = (V', E')$ of $G = (V, E)$, such that $\forall v \in V'$, $|N_H(v)| \geq h$.

The degree-based core number of $v$ in $G$, denoted as $DCore(v) = k$, is the maximum value of $k$ such that there exists a degree-based core constrained by $k$ containing $v$ and there is not any degree-based core constrained by $(k+1)$ containing $v$. The neighbor-based core number of $v$ in G, denoted as $NCore(v) = h$, is the maximum value of $h$ such that there exists a neighbor-based core constrained by $h$ containing $v$ and there is not any neighbor-based core constrained by $(h+1)$ containing $v$.

The algorithms used in the experiments are as follows:

- DegreeCore: the degree-based core decomposition algorithm in [29].
- NeighborCore: the neighbor-based core decomposition algorithm.
- CoCoreComp: the constrained core computation algorithm (Algorithm 1).
- CoCoreDecomp: the constrained core decomposition algorithm (Algorithm 2).
- CoCoreIncrement: the incremental constrained core algorithm for hyperedge insertion (Algorithm 3).
- CoCoreDecrement: the decremental constrained core algorithm for hyperedge deletion (Algorithm 4).

**Settings.** All algorithms are implemented in C++ and compiled by GNU GCC 9.3.0. All experiments are conducted on a machine with an Intel Xeon Platinum 8276M 2.2GHz CPU and 120GB memory in Ubuntu 20.04.2 LTS.

### B. Results and Analysis

We first show the engagement metric distributions of the datasets, and then evaluate the distribution of constrained core number, the executed time of decomposition and maintenance algorithms. Finally, we compare the constrained core with degree-based core and neighbor-based core.

**Experiment 1.** *Distribution of the degree and the number of neighbors of vertices*. Fig. 3 illustrates the distribution results of each hypergraph. The distributions of these two indicators exhibit very similar properties in most of the hypergraphs, and

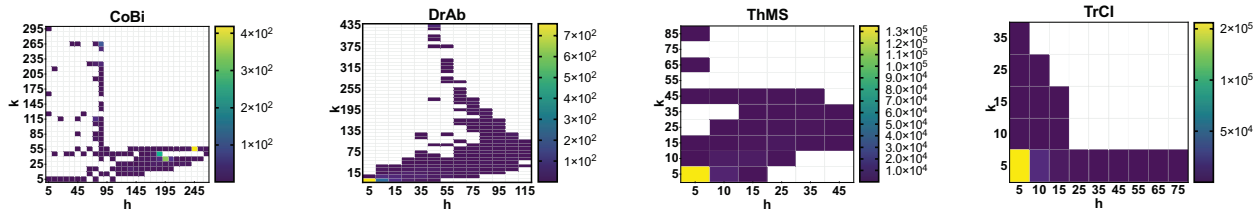



Fig. 5. Distribution of the constrained core numbers of vertices.

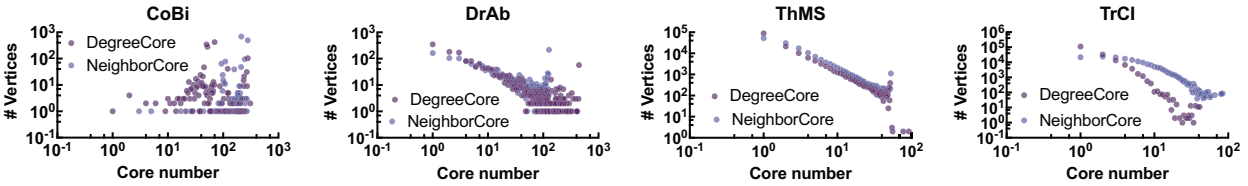



Fig. 6. Distribution of the degree-based core number and the neighbor-based core number of vertices.

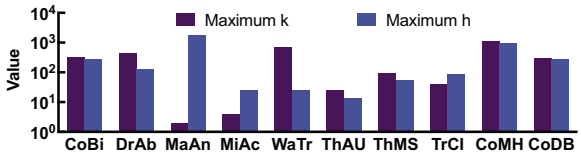



Fig. 7. Comparison of the maximum degree-based core number and the maximum neighbor-based core number.

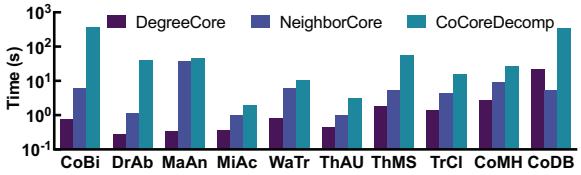



Fig. 8. Comparison of the decomposition time of the three cores.

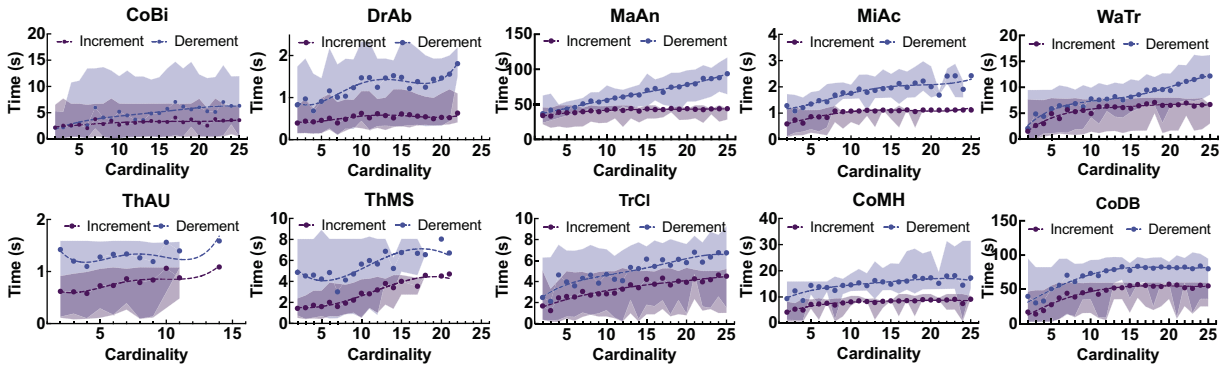
in several hypergraphs, it shows a power-law distribution for both indicators. The distribution of the number of neighbors of vertices in the same hypergraph is finer and tighter than that of the degree of vertices.

**Experiment 2.** *Comparison of three types of cores*. To measure the cohesiveness of the three types of cores, we compare the three types of cohesive subgraphs, the degree-based core constrained by $k = 10$, neighbor-based core constrained by $h = 10$, and constrained core $(10, h_{max})$ (where $h_{max}$ is the maximum neighbor constraint when $k = 10$). Fig. 4 reports the density (the ratio of the number of hyperedges and the number of vertices) and the average degree of the three types of cores. The density and the average degree of the constrained core are higher than the other two cores in all datasets, especially the neighbor-based core. Table III presents the specific cases in TrCl, in NeighborCore, a vertex has a large degree and many neighbors, while the opposite is true for DegreeCore, where both metrics are small for the constrained core. Therefore, the constrained core exhibit the strongest cohesive nature.

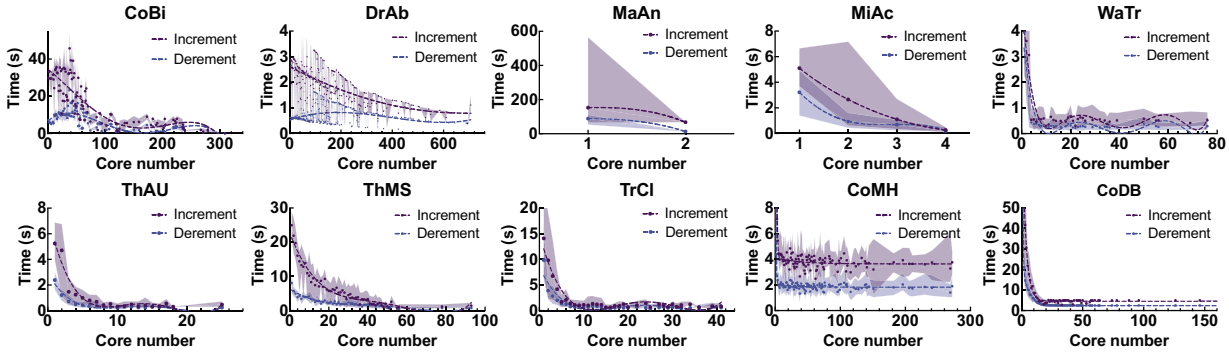
TABLE III
CASE STUDY OF TRCL. $m$ AND $n$ ARE THE NUMBER OF HYPEREDGES AND VERTICES OF CORES. *deg* AND *nei* ARE THE DEGREE AND NUMBER OF NEIGHBORS OF THE VERTEX.

| k=10 h=10 or max. | NeighborCore m=76315, n=38530 | | DegreeCore m=8182, n=662 | | CoCore m=7562, n=585 | |
|---|---|---|---|---|---|---|
| Vertex ID | deg | nei | deg | nei | deg | nei |
| 89434 | 89 | 91 | 44 | 27 | 38 | 26 |
| 100856 | 73 | 81 | 21 | 10 | 20 | 8 |
| 137545 | 22 | 22 | 19 | 17 | 18 | 16 |
| 153278 | 98 | 49 | 85 | 26 | 84 | 25 |

**Experiment 3.** *Distribution of core numbers of vertices*. Fig. 5 shows the distribution results of 4 representative hypergraphs. It can be seen that the range of values of the degree constraint $k$ of vertices is wider than that of the neighbor constraint $h$ in all hypergraphs. When the degree constraint of vertices is small, the neighbor constraint can make a more refined hierarchy of hypergraph. But when the degree constraint of vertices is large, the effect of neighbor constraint on the hierarchy of hypergraphs will be greatly weakened. Fig. 6 shows the distributions of the degree-based core number and the neighbor-based core number of vertices for the same hypergraphs in Fig. 5. It can be seen that the distributions of core numbers are highly correlated with the distributions of the degree and the number of neighbors of vertices (Fig. 3). Fig. 7 shows the maximum degree-based core number and the maximum neighbor-based core number for each hypergraph.




(a) Uptate time under different cardinalities of dynamic hyperedges




(b) Uptate time under different core numbers of dynamic hyperedges

Fig. 9. Comparison of the update time of incremental and decremental constrained core maintenance.

In most hypergraphs, the maximum $k$ and the maximum $h$ are similar. But in sparse hypergraphs with large cardinality such as MaAn and MiAc, the maximum $k$ is small, while the maximum $h$ still remains large. In such hypergraphs, the neighbor-based core number and the constrained core number remain valid, but the degree-based core number is invalid.

**Experiment 4.** *Decomposition time of core numbers*. Fig. 8 reports the time of decomposing the three types of cores in all hypergraphs. It shows that NeighborCore is slower than DegreeCore because the update of the degree is easier than updating the number of neighbors of vertices. The deletion of a hyperedge only makes the degree of each vertex in this hyperedges decrease by one, but may incur an uncertain change on the neighbors of a vertex, due to the fact that these vertices may still be neighbors. In hypergraphs with small cardinalities, the runtime of DegreeCore and NeighborCore are quite close, because the hypergraphs with small cardinalities are similar to the pairwise graphs where the degree of a vertex approximates to the number of neighbors of the vertex. CoCoreDecomp is the slowest one because it is a hybrid algorithm of the two previous decomposition algorithms. However, for most hypergraphs, the runtime for CoCoreDecomp is much smaller than the sum of the running time for NeihgborCore and DegreeCore.

**Experiment 5.** *Maintenance time of constrained core numbers*. We compare the running time of CoCoreIncrement and CoCoreDecrement on all hypergraphs. We randomly select 20 hyperedges for each hypergraph at each value of cardinality from 2 to 25, and record the time to update the constrained core numbers for the deletion and insertion of these hyperedges. Fig. 9(a) reports the update time under different cardinalities of hyperedges for each hypergraph in the incremental and decremental cases, where the dashed lines are the fitted curve of the average update time, and the colored zones are the bounds of the update time. It shows that the smaller the cardinality of the dynamic hyperedge, the shorter the time needed to update the constrained core numbers of vertices, for both the insertion or the deletion cases. When the cardinality of the dynamic hypergraphs increase to a large number, the runtime of CoCoreIncrement and CoCoreDecrement increases very slowly, which demonstrates that our algorithm achieves good stability in large-scale datasets. For the same hypergraph, the runtime of CoCoreDecrement is longer than that of CoCoreIncrement. This is because CoCoreDecrement needs to update the neighbor constraints of vertices with two different degree constraints. Fig. 9(b)

reports the update time under different values of dynamic hyperedges $k$ and the setting is the same as Fig. 9(a). As the core number of the dynamic hyperedge increases, the update time becomes smaller. From these two experiments, we can speculate that the update time of the maintenance algorithms is positively related to the scale of the constrained core where the dynamic hyperedge belongs.

**Summarization**. From the above experiments, it can be concluded that both degree-based core number and neighbor-based core number can reflect the importance of the vertices to some extent in hypergraphs, but they may lose their usefulness on specific categorizes of hypergraphs. The constrained core number is a satisfactory balance between these two metrics, such that it can perfectly reflect the refined engagement of the hypergraph. In addition, the constrained core decomposition algorithm exhibits good efficiency and the constrained core maintenance algorithms exhibit good stability on real datasets.

## VII. Conclusion

We initialized the study of the fundamental vertex engagement problem in the general hypergraphs. The constrained core was presented to model the vertex engagement by integrating the superiority of both group engagement and neighbor engagement, and it was shown that the constrained core decomposition can be completed in linear time. To avoid redundant computations caused by the decomposition algorithm when only a few vertices need to update their constrained core numbers in dynamic hypergraphs, efficient maintenance algorithms for both hyperedge insertion and deletion were also given. Extensive experiments demonstrate that the proposed model is effective, and our algorithms are efficient and stable.

Our work has shed some light on the modeling of vertex engagement in the complex hypergraphs. In the future, it deserves more efforts to investigating practical structural properties of hypergraphs such that more precise metrics for vertex engagement can be derived.

## References


[1] K. Seki and M. Nakamura, "The mechanism of collapse of the friendster network: What can we learn from the core structure of friendster?" *Soc. Netw. Anal. Min.*, vol. 7, no. 1, pp. 10:1–10:21, 2017.

[2] Q. Linghu, F. Zhang, X. Lin, W. Zhang, and Y. Zhang, "Global reinforcement of social networks: The anchored coreness problem," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM, 2020, pp. 2211–2226.

[3] K. Seki and M. Nakamura, "The mechanism of collapse of the friendster network: What can we learn from the core structure of friendster?" *Soc. Netw. Anal. Min.*, vol. 7, no. 1, pp. 10:1–10:21, 2017.

[4] F. D. Malliaros and M. Vazirgiannis, "To stay or not to stay: modeling engagement dynamics in social graphs," in *22nd ACM International Conference on Information and Knowledge Management*, Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, Eds. ACM, 2013, pp. 469–478.

[5] K. Bhawalkar, J. M. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma, "Preventing unraveling in social networks: The anchored k-core problem," *SIAM J. Discret. Math.*, vol. 29, no. 3, pp. 1452–1475, 2015.

[6] Q. Linghu, F. Zhang, X. Lin, W. Zhang, and Y. Zhang, "Towards user engagement dynamics in social networks," *CoRR*, vol. abs/2110.12193, 2021.

[7] M. Aalabaf-Sabaghi, "Networks, crowds and markets: Reasoning about a highly connected world," *Journal of the Royal Statistical Society Series A*, vol. 175, no. 4, pp. 1073–1073, 2012.

[8] A. Harkins, "Network games with perfect complements," 2013.

[9] P. S. Chodrow and A. Mellor, "Annotated hypergraphs: Models and applications," *Applied Network Science*, vol. 5, p. 9, 2020.

[10] C. Berge, "Graphs and hypergraphs," 1973.

[11] X. Ouvrard, "Hypergraphs: an introduction and review," *ArXiv*, vol. abs/2002.05014, 2020.

[12] S. Hu, X. Wu, and T. H. Chan, "Maintaining densest subsets efficiently in evolving hypergraphs," in *Proceedings of Conference on Information and Knowledge Management, CIKM*. ACM, 2017, pp. 929–938.

[13] A. E. Sariyüce and A. Pinar, "Fast hierarchy construction for dense subgraphs," *Proc. VLDB Endow.*, vol. 10, no. 3, pp. 97–108, 2016.

[14] G. Preti, G. D. F. Morales, and F. Bonchi, "Strud: Truss decomposition of simplicial complexes," in *WWW '21: The Web Conference*, 2021, pp. 3408–3418.

[15] Q. Luo, D. Yu, Z. Cai, X. Lin, and X. Cheng, "Hypercore maintenance in dynamic hypergraphs," in *International Conference on Data Engineering*, 2021, pp. 2051–2056.

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

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

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

[19] S. Wu, A. D. Sarma, A. Fabrikant, S. Lattanzi, and A. Tomkins, "Arrival and departure dynamics in social networks," in *Sixth ACM International Conference on Web Search and Data Mining, WSDM*. ACM, 2013, pp. 233–242.

[20] P. Lee, L. V. S. Lakshmanan, and E. E. Milios, "CAST: A context-aware story-teller for streaming social content," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM*. ACM, 2014, pp. 789–798.

[21] F. Zhang, W. Zhang, Y. Zhang, L. Qin, and X. Lin, "OLAK: an efficient algorithm to prevent unraveling in social networks," *Proc. VLDB Endow.*, vol. 10, no. 6, pp. 649–660, 2017.

[22] F. Zhang, J. Xie, K. Wang, S. Yang, and Y. Jiang, "Discovering key users for defending network structural stability," *World Wide Web*, 2021.

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

[24] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *J. ACM*, vol. 30, no. 3, pp. 417–427, 1983.

[25] F. D. Malliaros and M. Vazirgiannis, "To stay or not to stay: modeling engagement dynamics in social graphs," in *International Conference on Information and Knowledge Management*. ACM, 2013, pp. 469–478.

[26] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Finding critical users for social network engagement: The collapsed k-core problem," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 245–251.

[27] Q. Linghu, F. Zhang, X. Lin, W. Zhang, and Y. Zhang, "Global reinforcement of social networks: The anchored coreness problem," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM, 2020, pp. 2211–2226.

[28] F. Zhang, X. Lin, Y. Zhang, L. Qin, and W. Zhang, "Efficient community discovery with user engagement and similarity," *VLDB J.*, vol. 28, no. 6, pp. 987–1012, 2019.

[29] M. Leng, L. Sun, J. Bian, and Y. Ma, "An $o(m)$ algorithm for cores decomposition of undirected hypergraph," *Journal of Chinese Computer Systems*, vol. 34, no. 11, pp. 2568–2573, 2013.

[30] M. Leng and L. Sun, "Comparative experiment of the core property of weighted hyper-graph based on the ispd98 benchmark," *Journal of Information and Computational ence*, vol. 10, no. 8, pp. 2279–2290, 2013.

[31] J. Jiang, M. Mitzenmacher, and J. Thaler, "Parallel peeling algorithms," *ACM Trans. Parallel Comput.*, vol. 3, no. 1, pp. 7:1–7:27, 2016.

[32] J. Shun, "Practical parallel hypergraph algorithms," in *PPoPP '20: 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, 2020, pp. 232–249.
[33] K. Gabert, A. Pinar, and Ü. V. Çatalyürek, "Shared-memory scalable k-core maintenance on dynamic graphs and hypergraphs," in *IEEE International Parallel and Distributed Processing Symposium Workshops*, 2021, pp. 998–1007.
[34] B. Sun, T. H. Chan, and M. Sozio, "Fully dynamic approximate k-core decomposition in hypergraphs," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 4, pp. 39:1–39:21, 2020.
[35] A. E. Sariyüce, B. Gedik, G. Jacques-Silva, K. Wu, and Ü. V. Çatalyürek, "Incremental k-core decomposition: algorithms and evaluation," *VLDB J.*, vol. 25, no. 3, pp. 425–447, 2016.
[36] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. M. Kleinberg, "Simplicial closure and higher-order link prediction," *Proc. Natl. Acad. Sci. USA*, vol. 115, no. 48, pp. E11 221–E11 230, 2018.