Available Online: 2 March 2026 DOI: 10.54254/2977-3903/2026.31968

A gentle introduction to approximation for k-core/coreness: a mathematically-oriented survey of algorithms and error bounds

Jinyu Liu

Sichuan University, Chengdu, China

liujinyu1@stu.scu.edu.cn

Abstract. The survey presents an overview of several algorithms associated with the approximation of k-core decomposition, and this survey aims to serve as a gentle introduction for students majoring in mathematics to gain an understanding of hypergraph. Due to their unique mathematical structure, hypergraphs have become good modeling tools for capturing multi-way relationships in sophisticated systems. Mathematical techniques play an important role in the k-core approximation by offering theoretical foundations and guaranty. Many of them are utilized—such as probabilistic bounds, structural constraints, inequations, and invariants—in order to design scalable and stable approximation algorithms. This survey reviews three recent algorithms, each of them representing various mathematical thought and different methodological paradigms, including parallel, stream computation, and dynamic approximation. In addition, the survey highlights that algorithms can still be improved and it is meaningful to find a deeper understanding of the stability and limiting behaviors of k-core structures.

Keywords: k-core decomposition, approximation, random graphs with prescribed k-core sequences, streaming algorithms, parallel batch-dynamic algorithms

1. Introduction

1.1. Background and motivation

Hypergraphs have emerged as a frontier research direction in contemporary computer science. Thanks to their ability to model higher-order interactions involving multiple entities, hypergraphs exhibit broad application potential in modern data science. They can accurately characterize multi-protein complex reactions and intricate metabolic pathways in bioinformatics, and they can also reconstruct genuine group-level collaborations in social networks—relationships that go beyond simple pairwise interactions—thereby providing powerful modeling tools for detecting organized fraud in financial risk control. In addition, from a personal perspective, system stability analysis and the identification of structural "backbones" in engineering systems may become among the most important application domains for hypergraphs, and especially for k-core analysis, in the future. In critical infrastructures such as cybersecurity systems and power grids, cascading

failures pose persistent risks; the ability to identify, in real time, the "backbone" that maintains global connectivity can effectively prevent systemic collapse.

However, hypergraphs encountered in modern industrial applications often reach scales of tens of millions or even hundreds of billions of vertices or hyperedges. For instance, in Pinterest's recommendation system, user–Pin–Board relationships naturally form a large-scale hypergraph. According to a study presented at KDD 2021, the Pinterest PIN-board hypergraph contains hundreds of millions of vertices (users and images) and hundreds of millions of hyperedges (Boards), where each hyperedge may connect dozens or even hundreds of vertices. What's more, Hu et al. reported that hypergraphs built from chemical–gene–disease relations extracted from PubMed typically contain millions to tens of millions of hyperedges. Each hyperedge corresponds to a publication or a complex biological event and may connect anywhere from a few to dozens of vertices, further increasing the computational burden associated with peeling operations in exact k-core decomposition.

Under such circumstances, computing exact k-core values becomes an unrealistic goal: repeatedly peeling vertices from massive hypergraphs and performing iterative updates inevitably incurs prohibitive computational and time costs. Any exact peeling-based k-core decomposition requires extensive repeated accesses and updates. As a result, various approximation approaches for k-core decomposition have gained significant attention. Due to the intrinsic structural complexity of hypergraphs, which makes them difficult to interpret using low-dimensional spaces or vector representations, the objectives of the k-core approximation have consistently emphasized low computational cost, high-probability guaranties on the range in which the true k-core value lies, and strong stability.

Recently, the k-core approximation is a particularly active research topic. That's because the k-core structure captures the relationships among objects under study (which may be viewed as elements of a set) and reflects the relative importance of certain objects within the global structure.

1.2. Approximation and mathematical logic

As a mathematics student, the author has long observed that most of my mathematical training has focused primarily on theoretical developments, with relatively limited exposure to engineering-oriented applications. This is, in my view, an unfortunate gap. One motivation for writing this survey is therefore to introduce k-core approximation methods for hypergraphs, while emphasizing the crucial role played by mathematical principles in their design and analysis.

From a mathematical standpoint, it is also natural to draw an analogy between numerical analysis and k-core computation on hypergraphs. In numerical analysis, it is impossible to reconstruct an unknown function exactly from a finite set of sampled data points. The practical objective is instead to construct an approximation whose error lies within an acceptable range and whose behavior is sufficiently stable. Even so, when spline interpolation is employed, there remains a substantial probability that the interpolating function fails to converge well to the true function in the neighborhood of certain nodes; moreover, smaller step sizes often correspond to dramatically increased computational costs. A crucial insight in numerical analysis is that the choice of approximation method itself has a profound impact on the quality of the result—for instance, polynomial approximations are generally ill-suited for functions with rational structures.

In numerical analysis, we often attempt to approximate an unknown function using limited data points, fully aware that an exact expression is unattainable for real-world functions. Accordingly, the core objective is never to "find the true function," but to construct a stable and convergent approximation within acceptable error bounds. Similar principles apply to k-core approximation in hypergraphs.

I also view the k-core value as analogous to concepts such as Principal Component Analysis (PCA) and variance in regression analysis. In regression theory, the fitting process is often endowed with a geometric interpretation: we seek the projection of the dependent variable onto the linear space spanned by the relevant independent variables, and different projection directions correspond to different explanatory powers. The direction along which the variance of the dependent variable is maximized typically provides the strongest explanation of the underlying phenomenon. Of course, this analogy reflects only my personal intuition. As Box famously stated, "All models are wrong, but some are useful." Ultimately, the goal of k-core approximation research lies in its practical relevance.

1.3. Role of approximation

When processing such massive and dynamically evolving data, k-core approximation plays the role of an "efficiency engine": it can strip away noise from hypergraphs with hundreds of millions of elements within seconds, rapidly isolating the most cohesive core substructures. These approximation techniques are widely used in robustness evaluation of large-scale communication networks and in structural simplification (pruning) prior to training hypergraph neural networks. In summary, hypergraphs provide the depth needed to describe complex systems, while k-core approximation algorithms supply the speed required for industrial-scale data processing. Their combination makes it efficient and feasible to extract the hard core structures of complex systems.

Thus, approximation is not merely about controlling error magnitude, but also about selecting appropriate methods for different structural forms.

1.4. Contributions and organization

In summary, this survey focuses on several relatively recent k-core approximation algorithms that, in my personal view, are closely connected to mathematical theory—particularly probability theory and sampling methods. The primary aim is to analyze the mathematical logic underlying these algorithms and to provide mathematics students with an illustrative perspective on how abstract theoretical tools can influence cutting-edge research in applied domains. It should also be acknowledged that the author is not an expert in computer science, and the contributions and insights in this paper are inevitably constrained. The author is grateful for the reader's understanding and forbearance.

2. Related work

The study of k-core decomposition has evolved significantly from sequential to parallel settings. Understanding the advancements in general graphs is crucial for comprehending the complexities involved in hypergraph extensions and k-core decomposition.

2.1. K-core decomposition in general graphs

The parallelization of the peeling process has been the primary focus of recent research. Early parallel algorithms, such as Julienne [1] , adopted an offline strategy. Let $F = v \in V_{active} | d(v) \leq k$, Julienne utilizes a batch-synchronous peeling strategy. In each iteration, it constructs a multiset L by aggregating the adjacency lists of all vertices currently in the frontier F . Each occurrence of a vertex $u \in L$ means a unit reduction in its degree. To process these updates efficiently, a parallel semisort-based histogramming algorithm is applied to compute the total degree decrement for each affected vertex. Subsequently, a parallel pack operation identifies the subsequent frontier by filtering vertices whose updated degrees fall below the threshold k .

Although this algorithm displays great theoretical elegance and accuracy, the high implementation complexity of Julienne hinders its practical adoption in real-world systems.

To address these practical limitations, several *online strategies* were proposed. Algorithms such as ParK [2] and PKC [3] focus on asynchronous or more frequent updates to the core numbers, which significantly accelerates the decomposition process in real-world large-scale graphs by reducing the waiting time between peeling levels.

A recent significant advancement was made by Liu and Dong et al. [4]. Their work is a breakthrough for bridging the gap between theoretical efficiency and practical performance. By introducing a novel Hierarchical Bucketing Structure (HBS), the algorithm manages vertex degrees and bucket updates with minimal contention. Furthermore, through rigorous Span Analysis, they provided strong theoretical bounds on the algorithm's parallel complexity, making it a state-of-the-art framework for super-scale graph analysis.

2.2. K-core decomposition in hypergraphs

Building upon the foundations of general graph algorithms, hypergraph k -core decomposition introduces additional challenges. Since a single hyperedge connects multiple vertices, its removal during the peeling process triggers a more complex chain of degree updates.

3. Preliminary

3.1. Hypergraph definition

In mathematics, the definition of a hypergraph strictly generalizes that of an ordinary graph, in which each edge connects exactly two vertices. The fundamental difference between hypergraphs and ordinary graphs lies in the fact that a hyperedge may contain an arbitrary number of vertices. As a result, hypergraphs are frequently employed as mathematical models for studying multi-way relationships and for evaluating system stability.

3.2. Mathematical foundations

3.2.1. Chernoff bounds

Let x1,…,xn be independent bernoulli random variables. the upper tail bound satisfies:

$$P(X \geq (1+\delta)m) \leq e^{-\frac{\delta^2 m}{3}}, \quad for\ 0 < \delta < 1 \tag{1}$$

Although Bernoulli random variables only take values in {0,1}, they can in fact represent a wide range of relationships. For example, the value 0 may indicate deletion while 1 indicates retention; a variable may encode whether a vertex is adjacent to a given vertex, or whether a particular hyperedge is selected during sampling. Chernoff bounds enable us to determine an upper bound on the k -core value of a vertex with high probability. One important observation is that when $\delta < 1$, a larger expectation which leads to a smaller probability of prediction error becomes available. In other words, increasing the number of samples reduces the probability of error.

3.2.2 Markov chains

In approximation algorithms, vertices whose degrees fall below a prescribed threshold will not be removed all at once, as doing so would be computationally expensive. Instead, vertices are selected for deletion according to certain probabilities. Once the remaining vertex set and hyperedge set at the current step are known, all

information required to compute the current degree of each vertex is available. Moreover, the algorithm's decision regarding which vertex to delete next depends solely on the current degrees.

The historical evolution of a vertex—such as its degree five minutes ago or the path through which it was retained—has no influence on the next deletion decision.

Therefore, the transition probability fully describes the algorithm's behavior on the remaining hypergraph, and the process can naturally be viewed as a Markov process. So, the formulation can hold:

$$P[S_{t+1} = B|S_t = A] = \prod_{v \in A \backslash B} P(v) \prod_{v \in B} 1 - P(v) \tag{2}$$

Let S_t denote the set of vertices remaining in the hypergraph at step t. The above formulation describes the state evolution of the remaining vertex set under successive peeling operations. Ultimately, S_t converges to the true k -core subgraph. This convergence can proven by using martingale theory and the Azuma–Hoeffding inequality.

3.2.3. Martingales and Azuma–Hoeffding inequalities

A martingale is a special type of stochastic process satisfying

$$E[X_{n+1}|X_1, X_2, \ldots, X_n] = X_n. \tag{3}$$

In this survey, let X_t denote the proportion of vertices in the remaining graph at step t whose degrees exceed a given threshold k .

The Azuma–Hoeffding inequality applies to stochastic process with bounded differences. Suppose $X_1, X_2, ..., X_n$ is such a process, and assume that $|X_t - X_{t-1}| \leq c_t$.

Then the probability of deviation from the initial value admits the bound:

$$P(|X_n - X_0| \geq t) \leq 2 * exp(-t^2 / \sum_{i=0}^{n} c_i^2) \tag{4}$$

The Azuma–Hoeffding inequality shows that, as long as each step induces only a bounded change, the probability of deviating significantly from the initial value decreases exponentially.

3.2.4. Convergence of S_T via Martingale theory

Let $f(X_t)$ denote the proportion of vertices remaining in the hypergraph at step t and $X_t = E[f(S_n)|S_t]$ This process satisfies the defining properties of a martingale. By applying the Azuma–Hoeffding inequality, we can prove that the difference between X_t and $E[f(S_n)]$ tends to 0 with high probability.

3.2.5. Reservoir sampling

In practical settings, hypergraph streams are often dynamic: old connections may disappear while new ones continuously arrive. For the purpose of system stability analysis, it is therefore crucial that the k -core computed on the sampled hypergraph be able to track the evolution of the original hypergraph in real time. If sampling only retains early-arriving hyperedges, significant bias may arise in the computed k -core.

Reservoir sampling addresses this issue by maintaining an unbiased sample of fixed size M . Let the stream consist of a sequence of hyperedges. For the i -th incoming hyperedge:

• 1. If $i \leq M$, the hyperedge is inserted directly into the reservoir.

• 2. If $i > M$, the hyperedge is selected with probability $P = M/i$; if selected, it replaces a randomly chosen hyperedge currently in the reservoir.

This procedure guarantees that, at any time, each hyperedge in the stream has exactly probability M/i of being retained in the reservoir. As a result, the k -core computed on the sampled hypergraph becomes an unbiased estimator of the k -core of the original hypergraph.

4. Representative k-core approximation methods

4.1. Generating random graphs with prescribed core-sequences

Although the work of Van Koevering et al. does not directly address k -core approximation algorithms, its contribution goes beyond algorithmic approximation and provides a form of model-based approximation. The paper focuses on the construction of a class of ordinary graphs that realize a given k -core profile—represented in the original paper as a core-value sequence—while preserving this core sequence exactly.

The primary objective of this study is not to design an efficient or simplified k -core decomposition algorithm, but rather to establish, from a theoretical perspective, the intrinsic robustness of coreness itself. The authors show that as long as structural changes to a graph are properly constrained, the core-value sequence remains invariant, and the coreness of relative vertices deviates only with very small probability. Besides, the paper also proves that it is still possible to construct ordinary graphs that realize the same core-value sequence even when precise knowledge of the underlying edge structure is not available but only coreness information is obtained. Such constructions serve as statistically meaningful approximation models or reference baselines for real networks and even more complex hypergraph structures. (It should be noted that the original paper considers simple undirected graphs.)

Van Koevering et al. regard the set of all graphs that realize a given core-value sequence as a state space, and they define a class of local graph transformations to construct a Markov random process over this space. To implement this Markov process, it is necessary to establish the existence of a valid initial state, which in turn requires the given core-value sequence to be realizable. The realization of the initial state is guaranteed by a sequence of structural lemmas.

Let $C = C_1 \geq C_2 \geq ... \geq C_n$ denote the known coreness values of the vertices, and define a d-regular graph as a graph in which every vertex has degree exactly d. The key lemmas are as follows:

• If d is even, then for any number of vertices satisfying $n \geq d + 1$, there exists at least one d-regular graph. (Not refer to any hypergraphs but just the number of vertices.) If d is odd, then for any even $n \geq d + 1$, there exists at least one d-regular graph on n vertices.

• For any integers d and n, if $n \geq d + 1$, there exists a graph whose minimum vertex degree is at least d.

• For a core-value sequence satisfying $C_i = C$ for all i, where $C = C_1 \geq C_2 \geq ... \geq C_n$, such a sequence is realizable by at least one graph if and only if $n \geq d + 1$ [5].

These lemmas provide the theoretical foundation for generating graphs that preserve the given core-value sequence. Starting from a valid initial graph, the authors define three types of graph transformations that can be applied to generate a sequence of graphs, each maintaining the given core-value sequence. Under appropriate legality conditions, this construction can be viewed as a Markov process. Importantly, the emphasis in the original paper is on existence: that is, the existence of such operations that preserve the core-value sequence.

The three transformations are defined as follows:

1. Add and Delete: For any vertices i and j, if there is no edge between them in graph G, an edge (i, j) may be added without affecting the core-value sequence. Similarly, an existing edge (i, j) may be deleted without changing the core-value sequence.

2. Move Endpoint: Let vertices h, i, j satisfy the conditions that the coreness of j is strictly smaller than that of h and i, that (h, j) is an edge in G, while (i, j) is not. Then the edge (h, j) may be deleted and replaced by the edge (i, j) .

3. Core Collapse and Core Expand: Let vertices h, i, j satisfy $C_h \geq C_i = C_j$. If both (h, i) and (h, j) are edges in G while (i, j) is not, then edges (h, i) and (h, j) may be deleted and replaced by (i, j).

Conversely, if (i, j) is an edge and neither (h, i) nor (h, j) is not, then (i, j) may be deleted and replaced by edges (h, i) and (h, j) [5].

Under the appropriate legality conditions, none of these transformations changes the coreness of any vertex. By randomly selecting among these valid transformations, a Markov chain whose state space consists of all graphs realizing the given core-value sequence is constructed. This Markov process provides a theoretical basis for randomly sampling possible network structures when all kinds of information are not available except coreness information.

Furthermore, the authors prove that the state space consisting of all graphs that realize a given core-value sequence is connected. The key idea is that, through an elaborately designed set of local transformations, it is possible to build a transformation path between any two graphs sharing the same core-value sequence with finite-length.

Specifically, the authors fix a common core-deletion order and show that any graph realizing the given core-value sequence can be gradually transformed—without altering vertex coreness—into a canonical graph structure. In this process, the Add and Delete operation is used to adjust local edge configurations without affecting coreness; Move Endpoint redistributes edges within higher-core layers while preserving the connectivity of lower-core vertices; and Core Collapse and Core Expand reorganize edge distributions across different core levels.

Through these steps, the authors demonstrate that any two valid graphs can be transformed into the same canonical structure and subsequently into one another, thereby establishing the connectivity of the entire state space. This result guarantees that the previously defined Markov process is irreducible, thus ensuring that random sampling and statistical analysis of graph structures are theoretically feasible when only core-value information is given.

4.2. Sketching k-cores in parallel and streaming

The algorithm studied by Esfandiari et al. aims to achieve a $1-\epsilon$ -approximation of k -cores by uniformly sampling hyperedges (referred to as a stream in the original paper), under a significantly reduced memory footprint. In this context, a $1-\epsilon$ -approximation means that the resulting subhypergraph guarantees that every vertex has degree at least $(1-\epsilon) * k$. Although such an approximation allows a controlled amount of error, it ensures that the degree condition holds for all vertices with high probability. Significantly, the approximation target considered by Esfandiari et al. is not the numerical error in the coreness value of individual vertices, but rather the structural preservation of the k -core subhypergraph.

Let $G=(V, E)$ denote a hypergraph, where V is the vertex set with $|V|=n$, and E is the hyperedge set with $|E|=m$. Let $d(v)$ denote the degree of vertex v, and let $C(i)$ denote the coreness of vertex i.

To approximate the coreness of vertices in the hypergraph G, Esfandiari et al. introduce the concept of core labeling. Core labeling assigns each vertex to a discrete coreness level, rather than estimating its exact coreness value. These levels are defined using a logarithmic partition (with base 2), which allows for constant-factor approximation error while significantly simplifying the structural complexity. Since coreness values may have a wide range, directly approximating them would result in a large value space. The logarithmic coreness representation compresses this space into $O(log(n))$ levels, thereby making the sketching process probabilistically tractable.

At a high level, the proposed method can be interpreted as constructing a sparse sketch of the hypergraph via independent and identically distributed Bernoulli sampling. The procedure consists of the following steps [6]:

1. Hyperedge Sampling and Labeling: Each hyperedge is sampled independently with probability p, where

$$p \in O(log(\frac{n}{\epsilon*(n^2)})) \tag{5}$$

with the logarithmic factor ensuring that all vertices are simultaneously correct with high probability. This sampling rate is chosen so that Chernoff bounds can be applied to guarantee that the relative error of the approximated degrees for all vertices does not exceed ϵ with high probability.

The sampled hyperedges form a sparse hypergraph (denote as H_0), on which an algorithm referred to as ExclusiveCorenessLabeling (a previously known procedure in that paper) is applied to obtain a unique label $l_0(i)$ for each vertex i. If $l_0(i) \geq C * log(n)$, then the coreness of vertex i in the original hypergraph G can be accurately estimated. Vertices whose coreness can be reliably estimated are subsequently removed (peeled) from G.

(Intuitively, ExclusiveCorenessLabeling simulates the k -core peeling process on the sparse sketch and uses probabilistic bounds to guarantee that if a vertex is assigned to a certain coreness level in the sketch, then its true coreness in the original hypergraph lies within the corresponding interval with high probability.)

2. Iterative Refinement: For the remaining subhypergraph of G, hyperedges are again sampled with double probability $2p$, and Step 1 is repeated.

3. Repetition: The operations described in Steps 1 and 2 will be performed approximately $O(log(n))$ times. Since each intermediate hypergraph (denote as H_i) is sparse, the total memory usage remains small throughout the process.

The performance guarantees of the algorithm are established through a sequence of lemmas. Let G denote the original hypergraph and H denote the sampled hypergraph obtained by Bernoulli sampling.

Let $\delta, \epsilon \in (0,1)$, and let f be a function of n satisfying $f(n) \geq 48 * log(\frac{n}{\delta})/(\epsilon^2))$. Assume that each hyperedge is sampled independently with probability p and $p \geq 48 * log(\frac{n}{\delta})/((\epsilon^2) * f(n))$. Then, for any vertex $v \in G$, the following statements hold with high probability $1 - \delta/(3 * n^2)$: (use $d(v)$ to denote the degree of vertex v).

If the degree $d(v)$ in G satisfies $d(v) \geq f(n)$, then the degree $h(v)$ of v in H satisfies $|h(v) - p * d(v)| \leq \epsilon * h(v)$ and $d(v) \geq 2 * (1 - \epsilon) * f(n)$.

Conversely, if $d(v) < f(n)$, then $h(v) < 2p * f(n)$.

Under the same conditions on $f(n)$ and p as above, the following statements hold with probability at least $1 - \delta/(3 * n^2)$:

If $h(v) \geq 2p * f(n)$, as a result, $|h(v) - p * d(v)| \leq \epsilon * h(v)$ and $d(v) \geq 2 * (1 - \epsilon) * f(n)$.

When $h(v) < 2p * f(n)$, we have $d(v) \leq 2 * (1 + \epsilon) * f(n)$. This lemma also holds in directed hypergraphs.

Using the previously defined labeling scheme, let λ denote the set of vertices randomly sampled from G, and let H_0 denote a random hypergraph induced on λ while $H' = H \cup H_0$. Let $l(v)$ denote the label assigned to vertex $v \in V \backslash \lambda$ by ExclusiveCorenessLabeling and $C'_H(v)$ denote the coreness value of vertex $v \in H'$. Then, for any vertex $v \in V \backslash \lambda$, the following properties hold:

(1) $l(v) \geq C'_H(v)$

(2) Equality in the bound occurs only when $v \in H'$.

For each sampling round j(the j-th sampling), and for any vertex v added to λ , the coreness of v in G ,with probability $1 - 1/3n$ satisfies: $(\frac{2*(1-\epsilon)*n}{2^{j-1}})$.

Based on these results, the authors further embed the sketching technique into the MapReduce and streaming computation models, obtaining scalable approximate k -core algorithms. This part of the work primarily focuses on algorithm engineering and adaptation to computational models, rather than introducing new theoretical tools [6].

4.3. Parallel batch-dynamic algorithms

In the work of Shun et al., the core mathematical ideas are primarily embodied in their specialized level update strategy and the strict control of approximation error bounds. The paper studies the problem of approximate k -core maintenance in dynamic hypergraphs, which can be interpreted mathematically as follows: when the graph structure is undergoing continuous updates, do local perturbations induce large global deviations in coreness estimation? This perspective closely parallels a central concern in numerical analysis—namely, whether approximations or numerical computations remain stable under variations of the input variables (for instance, without incurring large relative errors). The method proposed by Julian Shun et al. addresses this challenge by introducing a carefully designed hierarchical mechanism that confines the error introduced by dynamic updates within a controllable range.

Similar to the approach of Esfandiari et al., this paper adopts a bucketing strategy and introduces the concepts of levels and groups. A level corresponds to an exponential discretization of possible coreness values, compressing continuously varying coreness into $O(logn)$ stable layers. Each level corresponds to an interval of coreness values $(1+\delta)^{l-1} \leq coreness(v) \leq (1+\delta)^{l}$. However, a level is neither determined nor statically computed; instead, it is dynamically maintained via a set of degree-based inequalities, its level is locally increased or decreased to satisfy these inequalities. (Detailed descriptions of these updates rules are provided in Section 5.1 and 5.3 of the original paper.)

The vertices of the hypergraph G are partitioned into $K = O(logn)$ levels (denoted by l), which are further grouped into $\lceil log_{1+\delta}(n) \rceil + 1$ groups (denoted by g). Each group contains $\lceil log_{1+\delta}(n) \rceil$ levels. The paper also proposes a specialized data structure known as the Parallel Level Data Structure (PLDS). Within this framework, the level of a vertex serves as the primary variable encoding an approxiamtion of its coreness, while its validity is enforced through a series of degree inequalities. The PLDS acts as the underlying mechanism that efficiently maintains vertex movements across levels during dynamic updates. This design is significant for reducing both the work and the depth of the algorithm in parallel settings.

According to the authors, in hypergraphs, updating the level of a single vertex might trigger updates to many other vertices, potentially leading to extremely high computational costs. Consequently, after each edge is updated, whether and how a vertex's level is adjusted depends on whether two key inequalities are satisfied. Let V_l denote the set of vertices assigned to level l , and let Z_l denote the set of vertices whose level is at least l . The PLDS is designed to enforce the following invariants (as stated in Section 5.3 of the paper):

Here, λ and δ are fixed positive constants.

• Degree Upper Bound: If a vertex $v \in V_l$ with $l < K$ satisfies the prescribed conditions, then v has at most $(2+\frac{3}{\lambda}) * (1+\delta)^{i}$ neighbors in Z_l .

• Degree Lower Bound: If a vertex $v \in V_l$ with $l < K$ and $l-1 \in g_i$, then v has at least $(1+\delta)^{i}$ neighbors in Z_{l-1} [7].

Through these upper and lower degree bounds, Section 5.3 ensures that each vertex's degree within the induced subgraph corresponding to its level remains confined to a stable interval, preventing uncontrolled drift of level assignments under dynamic changes. However, a level only provides a coarse-grained approximation of a vertex's true coreness and does not directly correspond to the exact k -core decomposition. To address this, Section 5.6 further analyzes the group structure induced by levels, demonstrating how coreness estimates can be constructed at the group level and proving that these estimates satisfy an overall approximation ratio of $2-\epsilon$. This analysis shows that, although the algorithm relies solely on local degree inequalities to maintain vertex levels, a theoretically guaranteed global approximation of coreness can still be achieved through an appropriate grouping of these levels.

In Section 5.6, the authors present a concrete estimation procedure, which can be regarded as equivalent to the approximation operation in this context. The estimation, referred to as CorenessEstimation (Definition 5.11 in the original paper), is defined as follows. Let $k(v)$ denote the true coreness value of vertex v, and let $k'(v)$ denote its estimated coreness [7].

1. CorenessEstimation: $k'(v) = (1+\delta)^{max(0,\lfloor (l(v)+1)/(4\lceil log_{1+\delta}(n)\rceil)\rfloor - 1)}$.

2. W.L.O.G, let $\delta = \epsilon/3$ and $\lambda = 9/\epsilon + 3$, so the CorenessEstimation satisfies:

a. If $k(v) > (2+3/\lambda) * (1+\delta)^{g'}$, then $k'(v) \geq (1+\delta)^{g'}$; conversely, if $k(v) < (1+\delta)^{g'-1}/(2+3/\lambda)$ then: $k'(v) \leq (1+\delta)^{g'}$.

b. As a direct consequence, for any constant $\epsilon > 0$, $k(v)/(2+\epsilon) \leq k'(v) \leq k(v) * (2+\epsilon)$.

Overall, the algorithm achieves stable k-core approximation in dynamic environments by maintaining a hierarchy of levels governed by inequality-based constraints. In contrast to approximation methods that rely on random sampling, the error control in this approach is derived primarily from structural constraints rather than probabilistic bounds, thereby providing a deterministic approximation guarantee of $2-\epsilon$. Moreover, the algorithm establishes provable upper bounds on update costs while preserving approximation accuracy, granting it strong stability and scalability in dynamic graph settings. These characteristics render the method not only an effective approximation algorithm but also a valuable framework for understanding the evolution of k-core structures in dynamically changing networks.

5. Conclusion and outlook

Although this survey reviews several algorithms for k-core decomposition, covering parallel computing, streaming and dynamic approximation methods , the mathematical theories mentioned by this survey should still be considered as introductory. Nevertheless, the three algorithms examined in this survey and the mathematical thought behind them exhibit clear differences.

From a higher-level perspective, the common goal of these algorithms is not to compute the exact coreness value but trying to figure out the "stable core" of the structure of the hypergraph in a dynamic changing environment with limited computing resources. Probabilistic approximation mainly aims at achieving high scalability on super-large scale hypergraph while structural methods make efforts to maintain local constraints in order to guarantee the control of global approximate error. This kind of difference reveals a significant point of general discussion: how to strike a balance between the randomness and the structural determinacy. However,limited by the research background and length, the survey only focuses on the simple rudiments of the role of mathematical methods in the design of new k-core decomposition algorithms.In spite of the limitation, the three algorithms and mathematical thoughts behind them have clearly embodied methodological paradigms.

Furthermore, the survey identifies several questions which may worth studying in the future. Initially, one recurring issue arises from the use of exponential bucketing schemes. For values of the form $(1+\delta)^n$, the determination of δ often depends on empirical experience or loose the theoretical upper bound. Therefore, developing more adaptive algorithms to automatically determine suitable δ values may become a promising research direction.

Moreover, if dynamic changes on hypergraphs exist a quantifiable upper bound on their variation magnitude, it is natural to ask what will happen when the bound is exceeded. Will the efficiency or accuracy of some algorithms sharply decrease? Addressing this problem may require introducing new mechanisms–such as novel bucketing strategies.

Besides, unified theoretical frameworks for balancing tighter approximation ratios against lower update costs remain absent. Nowadays, most of the existing algorithms focus primarily on coreness itself, while its semantic explanation during network evolution is still not well understood. Finally, from a mathematics perspective, a deeper understanding of the stability and limiting behaviors of k -core structures and their approximation methods under different graph models might offer stronger theoretical foundations for the design of relevant algorithms.

All in all, the approximation algorithms for k-core decomposition is not only a calculating tool for super-scale graph calculation, but also provides a practical breakthrough point with theoretical depth for the research of dynamic network structure.

References

[1] Dhulipala, L., Blelloch, G. E., & Shun, J. (2017). Julienne: Isotropic high-level abstractions for parallel graph algorithms. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '17)* (pp. 253–264). Association for Computing Machinery. https: //doi.org/10.1145/3087556.3087579

[2] Dasari, N. S., Desh, R., & Zubair, M. (2014). ParK: An efficient algorithm for k-core decomposition on multicore processors. In *2014 IEEE International Conference on Big Data (Big Data)* (pp. 9–16). IEEE. https: //doi.org/10.1109/BigData.2014.7004394

[3] Vankadara, L. V. (2016). *PKC: A fast shared-memory algorithm for k-core decomposition*. arXiv preprint. https: //arxiv.org/abs/1602.05372

[4] Liu, Y., Dong, X., Blelloch, G. E., & Shun, J. (2022). Parallel k-core decomposition: Theory and practice. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '22)* (pp. 317–331). Association for Computing Machinery. https: //doi.org/10.1145/3503221.3508428

[5] Van Koevering, K., Benson, A. R., & Kleinberg, J. (2021). Random graphs with prescribed k-core sequences: A new null model for network analysis. In *Proceedings of the Web Conference 2021 (WWW '21)* (pp. 320–331). Association for Computing Machinery. https: //doi.org/10.1145/3442381.3450015

[6] Esfandiari, H., Lattanzi, S., & Mirrokni, V. (2018). Parallel and streaming algorithms for k-core decomposition. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)* (pp. 1424–1433). PMLR. http: //proceedings.mlr.press/v80/esfandiari18a.html

[7] Shun, J., Liu, Q. C., & Blelloch, G. E. (2021). Parallel batch-dynamic algorithms for k-core decomposition and related graph problems. *ACM Transactions on Parallel Computing, 8*(4), Article 21, 1–28. https: //doi.org/10.1145/3470638