Core Decomposition of Massive, Information-Rich Graphs

Francesco Bonchi[1], Francesco Gullo[2] and Andreas Kaltenbrunner[3]
[1]ISI Foundation, Turin, Italy
[2]UniCredit, R&D Dept, Rome, Italy
[3]Eurecat, Barcelona, Spain

Synonyms

Core graph decomposition; *k*-Core decomposition; *k*-Coreness; *k*-Cores; *k*-Shell decomposition

Glossary

Core decomposition	The set of all *k*-cores of a graph, for all *k*.
Core number (or core index)	For each vertex of a graph, the highest order of a core containing that vertex.
Degeneracy	The highest order of a core of a graph. It corresponds to the maximum core number over all vertices of the graph.
Distributed graph	Graph that is stored across multiple machines.
Graph (or network)	A set of objects (called *vertices* or *nodes*) connected to each other by *links* (also known as *edges* or *arcs*). Links can be represented as unordered or ordered pairs of vertices. In the former case, the graph is said to be *undirected*, otherwise it is *directed*. Links may be assigned weights. In this case, the graph is said *weighted*.
k-core (or core of order *k*)	Maximal subgraph where each vertex is connected to at least *k* other vertices within the subgraph.
k-shell	Subgraph induced by all vertices belonging to the *k*-core but not to the $(k + 1)$-core.
Multilayer graph	Graph composed of multiple layers, with each layer corresponding to a specific type of relation among vertices. It can be viewed as a superposition of several graphs, each one having its own edge set. It is also known as *multiplex graph* or *multidimensional* graph.
Temporal graph	Graph whose structure can change over time. It may be represented as a set of *snapshots*, one per temporal instant, where each snapshot

© Springer Science+Business Media LLC 2017
R. Alhajj, J. Rokne (eds.), *Encyclopedia of Social Network Analysis and Mining*,
DOI 10.1007/978-1-4614-7163-9_110176-1

	is a graph with its own fixed structure.
Uncertain (or probabilistic) graph	Graph whose edges and/or vertices are assigned a probability of existence.

Definition

Let $G = (V, E)$ be an undirected graph, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. For a subset of vertices $H \subseteq V$, let $G[H]$ denote the subgraph of G induced by H, i.e., $G[H] = (H, E[H])$, where $E[H] = \{(u, v) \in E | u \in H, v \in H\}$. Let also $deg_H(u)$ denote the degree of u in $G[H]$, and $\mu(H)$ the minimum degree of a vertex in $G[H]$, i.e., $\mu(H) = \min_{u \in H} \deg_H(u)$.

The *k-core* (or *core* of order k) of a graph $G = (V, E)$ is defined as a maximal subgraph in which every vertex is connected to at least k other vertices within that subgraph. A k-core is fully identified by the vertices belonging to it, which we denote by C_k. The overall k-core subgraph is given by the subgraph of G induced by C_k. All cores are nested into each other: $V = C_0 \supseteq C_1 \supseteq L \supseteq C_{|V|}$, and the “difference” between two consecutive cores, i.e., the set $S_k = C_k \setminus C_{k+1}$, is referred to as *k-shell*. The set of all k-shells therefore forms a partition of the vertex set V. Note that a k-core (or a k-shell) does not necessarily induce a connected subgraph, and that k-cores are not necessarily all distinct, i.e., it may happen that, for some k, $C_k = C_{k+1}$ (and the corresponding k-shell $S_k = \emptyset$).

Let k^* denote the highest order of a core in G, i.e., the *degeneracy* of $G : k^* = \max \{k \in [1..|V|] \mid \nexists\, k' > k s.t. C_{k'} \subset C_k\}$. The set $\mathbf{C} = \{C_k\}_{k=0}^{k^*}$ of all cores forms the *core decomposition* of G. The *core number* (or *core index*) of a vertex $u \in V$, denoted $c(u)$, is the highest order of a core that contains $u : c(u) = \max \{k \in [0 .. k^*] \mid u \in C_k\}$. It is easy to see that the order of a core is guaranteed to be less than or equal to the minimum degree of a vertex in that core, i.e., $k \leq \mu(C_k)$. In the case $C_k \supset C_{k+1}$ (i.e., the shell S_k is nonempty), the equality $k = \mu(C_k)$ holds.

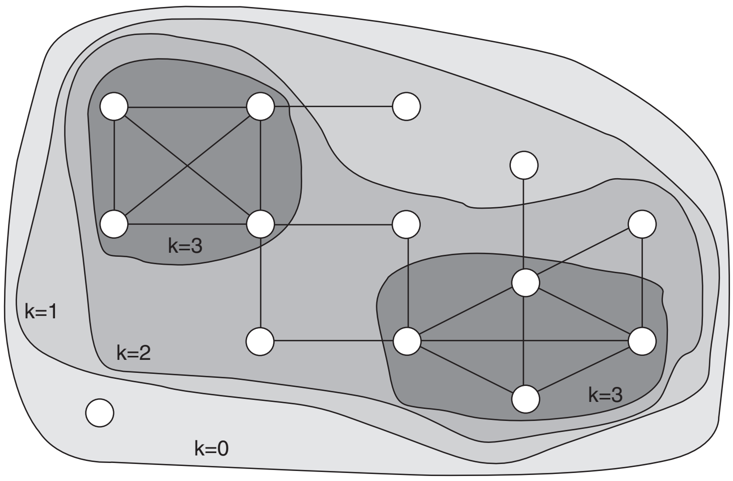
Figure 1 gives an illustration of core decomposition of a toy graph.

Introduction

Recent advances in social and information science have shown that linked data is pervasive in our society and the world around us. *Graphs* have become a ubiquitous model to represent real-world structured data. They are routinely used to describe a large variety of data such as the Web, social networks, knowledge bases, (heterogeneous) information networks, biological networks, and many more. As the ability to collect data has increased geometrically in the recent years, these graphs typically count millions, or even billions, of vertices and edges. Justifiably, there has been an increasing demand for methods that are able to cope with graphs at a large scale.

At the same time, the proliferation of heterogeneous data acquired from a variety of sources has given rise to more and more *complex* linked-data representations. As a result, today’s real-world graphs exhibit a wide set of additional information assigned to their vertices and/or edges: weights, labels, feature vectors, probabilities of existence, probability distributions over weights or labels, time series capturing the dynamic evolution of the network, and so on (Bonchi et al. 2015, 2014a; Khan et al. 2015, 2014; Parchas et al. 2015; Ruchansky et al. 2015). These enriched graphs constitute a unique opportunity, but also a serious challenge, for improving the quality of processing algorithms.

Finding dense substructures in a graph is a fundamental primitive in many graph-analysis tasks (Lee et al. 2010; Tsourakakis et al. 2013). Many different definitions of a dense subgraph have been proposed, e.g., *cliques, n-cliques, n-clans, k-plexes, f-groups, n-clubs, lambda sets*. Most of these definitions are computationally intensive: **NP**-hard or of quadratic time complexity. In this respect, the concept of *core decomposition* is particularly appealing because it can be computed in linear time and is related to many of the various definitions of a dense subgraph.



Core Decomposition of Massive, Information-Rich Graphs, Fig. 1 Core decomposition. The graph has four cores. The 0-core is the entire graph, while the 3-core has two connected components

Key Points

The research community has recently devoted increasing attention to applying the core-decomposition tool to graphs with the peculiarities of the ones arising in today's applications (Malliaros et al. 2016). Effort has been devoted to handle massive graphs, by devising external-memory algorithms (Cheng et al. 2011; Wen et al. 2016), methods suitable for distributed graphs (Montresor et al. 2013; Aksu et al. 2014), or algorithms that can exploit multicore parallel processing (Dasari et al. 2014). At the same time, challenging contexts requiring highly efficient solutions have been taken into consideration, such as maintenance of cores in a dynamic setting (Sariyüce et al. 2013, 2016; Li et al. 2014) or local estimation of cores (O'Brien and Sullivan 2014). Finally, research has also focused on special types of graph that, apart from the usual information about vertices and links, come with additional features that call for the development of ad hoc nontrivial solutions. This is the case of uncertain graphs (Bonchi et al. 2014b), temporal graphs (Wu et al. 2015), and multilayer graphs (Azimi-Tafreshi et al. 2014).

Historical Background

Core decomposition has been introduced by Seidman (1983) to quantify the global position of a vertex in a network. Since then, core decomposition has been widely used in tasks like analyzing the complex nature of a network (Bollobás 1984; Luczak 1991; Dorogovtsev et al. 2006; Alvarez-Hamelin et al. 2008; Carmi et al. 2007), network visualization (Batagelj et al. 1999; Gaertler and Patrignani 2004; Alvarez-Hamelin et al. 2005), and influence spreading (Bae and Kim 2014; Bonchi et al. 2014b; Pei and Makse 2013; Rossi et al. 2015). More recently, the attention has been shifted to massive graphs (Aksu et al. 2014; Cheng et al. 2011; Dasari et al. 2014; Montresor et al. 2013; Wen et al. 2016) and graphs coming with additional information such as attributes, probabilities, or labels (Azimi-Tafreshi et al. 2014; Bonchi et al. 2014b; Wu et al. 2015).

Algorithm 1 *k*-CORES

```
Input: A graph $G = (V, E)$.
Output: A $|V|$-dimensional vector
c containing the core number of each
$v \in V$.
1: c ← ∅, d ← ∅, D ← [∅, ..., ∅]
2: for all $v \in V$ do
3:   d[$v$] ← $deg(v)$
4:   D[$deg(v)$] ← D[$deg(v)$] ∪ {$v$}
5: end for
6: for all $k = 0, 1, \ldots, |V|$ do
7:   while D[$k$] ≠ ∅ do
8:     pick and remove a vertex $v$ from D
[$k$]
```

```
9:      c[$v$] ← $k$
10:     for all $u$ : ($u$, $v$) ∈ $E$, d[$u$] > $k$ do
11:       move $u$ from D[d[$u$]] to D[d[$u$] − 1]
12:       d[$u$] ← d[$u$] − 1
13:     end for
14:     remove $v$ from $G$
15:   end while
16: end for
```

Core Decomposition

Core decomposition of a simple, unweighted graph $G = (V, E)$ can be computed in time linear in the size of G. The algorithm designed by Batagelj and Zaveršnik (2011) (Algorithm 1) iteratively removes the smallest-degree vertex from the graph and sets the core number of the removed vertex equal to the number reached so far. Vertices are (kept) ordered based on their degree by bin sort, thanks to the fact that the degree of a vertex in G is a bounded integer quantity. Specifically, the idea is to employ a vector $\mathbf{D}$ of size $|V|$ whose single cells $\mathbf{D}[i]$ store the set of all vertices in the current graph that have degree equal to i. This way, ordering vertices at the beginning of the algorithm and keeping them ordered during the whole execution can be performed in $\mathcal{O}(|V|)$ time and $\mathcal{O}(1)$ time, respectively. As a result, given that each vertex and all its neighbors are visited only once, the overall time complexity of the algorithm is $\mathcal{O}(|V| + |E|)$. The one by Batagelj and Zaveršnik is an in-memory algorithm: it exploits random access of the graph vertices and edges, thus requiring the whole input graph to be loaded into main memory.

A number of variants and optimizations of the basic Batagelj and Zaveršnik's algorithm have been proposed. Khaouid et al. (2015) devise an implementation of the algorithm within the popular *Webgraph* graph-compression framework (Boldi and Vigna 2004). The idea is to efficiently implement the data structures needed for bin sort (by using flat arrays instead of hash-tables), while at the same time exploiting the Webgraph APIs to lazily access a compressed version of the input graph. Dasari et al. (2014) define a variant, dubbed ParK, which reduces the working-set size and the number of random accesses, while also being amenable to parallelization on multi-core architectures.

Generalized cores. Batagelj and Zaveršnik (2011) also define a method to compute *generalized cores* efficiently. Traditional cores, i.e., cores based on the degree of a vertex, are a special case of generalized cores, which are instead specific of any *vertex property function*. As long as the vertex property function under consideration meets the requirement of being *monotone*, Batagelj and Zaveršnik (2011) show that the corresponding (generalized) core decomposition can be computed in $\mathcal{O}(|E| \times \max\{\Delta, \log |V|\})$ time, where Δ is the maximum degree of a vertex in G.

Formally, for any vertex $v \in V$ and $C \subseteq V$, a vertex property function on G is a function $\phi(v, C) : V \times 2^V \rightarrow R$. A vertex property function $\phi(v, C)$ is said *monotone* if $\forall C_1, C_2 \subseteq V : C_1 \subseteq C_2$ implies that $\forall v \in V : \phi(v, C_1) \leq \phi(v, C_2)$. Examples of monotone vertex property functions are degree, in-degree, out-degree, weighted (in/out-)degree, maximum weight of an edge incident to a vertex, and number of cycles of fixed length k passing through a vertex. Finding cores based on a monotone vertex property function can be carried out by a slight modification of Algorithm 1. The main idea is still to iteratively remove a vertex exhibiting the smallest value of the vertex property function under consideration. To do so, though, depending on the specific vertex property function (i.e., if it is not a bounded integer quantity), it may happen that bin sort cannot anymore be employed to order vertices. In general, a priority queue with logarithmic-time access/update operations should instead be used. That is why, in the general case, the overall time complexity becomes the one reported above, i.e., $\mathcal{O}(|E| \times \max\{\Delta, \log |V|\})$.

Directed graphs. Giatsidis et al. (2011) extend core decomposition to directed graphs. They introduce the notion of (k, l)-*D-core*, which is defined as a maximal subgraph of the input directed graph where each node has in-degree at least k and out-degree at least l within that subgraph. The intuition behind (k, l)-D-cores is to look for subgraphs whose nodes share enough out-links and in-links with the other nodes of the subgraph.

Given two positive integers k and l, the (k, l)-D-core of a directed graph can be computed by a simple algorithm that iteratively removes nodes having in-degree less than k or out-degree less than l, until none of such nodes remains in the graph. Core decomposition in a directed graph is therefore defined as the set of all (k, l)-D-cores, for all k and l. It can be computed by running the procedure for computing a single (k, l)-D-core for all possible values of k and l. Its worst-case time complexity is thus $\mathcal{O}(\Delta_{\text{in}} \times \Delta_{\text{out}} \times |E|)$, where Δ_{in} and Δ_{out} are the maximum in-degree and out-degree of a node in the input graph, respectively.

External-Memory Core Decomposition

When the input graph is too large to fit in main memory a valuable option is to resort to external-memory algorithms.

Cheng et al. (2011) propose an external-memory algorithm for core decomposition which follows a top-down approach. That method, dubbed `EMcore`, recursively computes k-cores from larger values of k to smaller ones, and progressively reduces search space and disk I/O cost by removing the vertices in each computed k-core. `EMcore` is based on three main components: a method for properly partitioning the input graph, a way to estimate the upper bound on the core number of a vertex, and a recursive top-down core-decomposition procedure. The algorithm processes the partitioned graph in multiple iterations, starting from vertices having the largest core number $k_{\max}$, and proceeding down to the vertices of core number 2. In each iteration, it loads into memory a subgraph containing all vertices with estimated core number belonging to an interval $[k_i, k_j]$, and the core decomposition of this subgraph is computed by following the traditional bottom-up in-memory method by Batagelj and Zaveršnik (2011). The `EMcore` algorithm takes $\mathcal{O}(k_{\max} \times (|V| + |E|))$ CPU time, $\mathcal{O}\left(\frac{k_{\max} \times (|V|+|E|)}{B}\right)$ I/O time, and $\mathcal{O}\left(\frac{|V|+|E|}{B}\right)$ disk-block space, where B is the size of a disk block.

Wen et al. (2016) devise `SemiCore`*, an I/O-efficient core-decomposition method that follows a semiexternal model. The employed model assumes that only vertices can be loaded into main memory, while edges are kept stored on disk. Specifically, `SemiCore`* keeps the core number of each vertex in memory and updates core numbers iteratively until convergence. In each iteration, only sequential scans of edges on disk are required. `SemiCore`* comes with guaranteed bounded memory requirement of $\mathcal{O}(|V|)$, and its CPU time complexity and I/O complexity are $\mathcal{O}(l \times (|V| + |E|))$ and $\mathcal{O}\left(\frac{l \times (|V|+|E|)}{B}\right)$, respectively, where $l \in [1, |V|]$ is the number of iterations needed for convergence (with $l = |V|$ in practice).

Distributed Core Decomposition

Due to the proliferation of data produced and stored by today's applications, it is not uncommon to come across graphs that are distributed through multiple machines. For such graphs, traditional core-decomposition methods do not work. There is hence need for a careful design of ad hoc methods that can profitably take into account the peculiarities of the distributed setting.

Montresor et al. (2013) devise an algorithm for distributed core decomposition which works under two different computational models: both *one-to-one* model, where one computational unit is associated with one vertex in the graph, and communication occurs only through direct messages between vertices connected by an edge, and *one-to-many* model in which one host stores multiple vertices together with their local and remote incident edges and communication occurs through messages exchanged between hosts. The main idea of Montresor et al's algorithm is to maintain for each vertex an upper bound on its core number. At the beginning, this upper bound corresponds to the degree, and is tightened at each iteration, until it converges to the true core number. The algorithm requires $\mathcal{O}(|V|)$ iterations to converge to the ultimate solution, while the overall size of the messages exchanged during its execution is bounded by $\mathcal{O}(\Delta \times |E|)$, where Δ is the maximum degree of a vertex in the input graph.

Khaouid et al. (2015) provide an implementation of Montresor et al's algorithm on the *GraphChi* graph engine (Kyrola et al. 2012) and the *Webgraph* compression framework (Boldi and Vigna 2004), and show that such implementations scale to really large datasets using only a single PC.

Aksu et al. (2014) propose a distributed core-decomposition algorithm that works under a computational model similar to the one-to-many model considered by Montresor et al. (2013), where the input graph is partitioned across multiple hosts. Nevertheless, unlike Montresor et al, Aksu et al assume that the subgraph stored in a single host may still be too large to fit in main memory. The algorithm by Aksu et al runs on the partitioned graph data in parallel and takes advantage of the k-core properties to prune unnecessary computation. Two main pruning rules are used: each vertex of a k-core is guaranteed to have (i) degree at least k, and (ii) at least k neighbors with degree at least k.

Efficient Maintenance of Cores

In a streaming setting the structure of the underlying graph data is assumed to be dynamic, in the sense that, at each time instant, there could be insertions/deletions affecting vertices/edges of the graph. This context calls for algorithms that are able to incrementally update a previously computed core-decomposition structure with little processing effort, instead of rebuilding it from scratch.

A major goal in defining an incremental core-decomposition algorithm is, given a set of insertion/deletion operations, to identify a vertex set that (i) contains *all* vertices whose core number needs to be updated, so as to still guarantee correctness of the updated core decomposition, and (ii) has size as small as possible, so that the number of unnecessary updates is minimized. Saríyüce et al. (2013) introduce a set of theoretical findings to efficiently build a vertex set exhibiting these two properties when a single edge is added to or removed from the graph (extensions to multiple edge insertions/deletions and vertex insertions/deletions are trivial). Based on these findings, Saríyüce et al define three algorithms that trade off between pruning power and time spent in executing pruning rules. Pruning power is basically determined by the capability of retrieving a smaller subset of vertices to be processed for an update operation. This is directly proportional to the number of pruning rules implemented: more pruning rules lead to smaller sets. Enhanced versions of these algorithms have been defined in (Saríyüce et al. 2016).

Other theoretical results to identify small sets of vertices whose core number is affected by an edge insertion/deletion have been introduced by Li et al. (2014). Specifically, Li et al devise a three-step algorithm. First, it is shown that the only vertices that may need their core number to be updated are the vertices that are reachable from the end vertices of the inserted/deleted edge. Based on this finding, a coloring algorithm is employed to find the actual vertices satisfying such a requirement. Second, from the vertices found by the coloring algorithm, a further coloring algorithm is run to identify all vertices that ultimately need their core number updated. In the third and last step, the core number of such vertices is updated by a fast linear-time procedure.

Further effort on efficient maintenance of core decomposition has been devoted in the context of external-memory methods (Wen et al. 2016) and distributed graphs (Aksu et al. 2014).

Local Estimation of Cores

An alternative realistic scenario is when the input graph cannot be observed as a whole, due to, e.g., scale, privacy, or business reasons, and the core number of a vertex should be estimated by looking only at its close neighborhood. This scenario is considered by O'Brien and Sullivan (2014), who propose a method to estimate the core number of a given vertex by only taking into account properties local to that vertex. Specifically, O'Brien and Sullivan define a local estimator that accesses properties of the graph within a region of fixed radius δ around the given vertex. The estimator is based on the main observation that the core number of a vertex can be precisely determined from the (exact) core number of its neighbors. Clearly, the latter information is not available in a real scenario. However, O'Brien and Sullivan elaborate on this finding and base their estimator on the idea of iteratively incorporating upper bounds on the core numbers of the neighbors of the input vertex. This method returns an upper bound on the actual core number of the input vertex, which becomes tighter as the number of considered neighbors increases, i.e., the radius δ gets larger. This way, δ acts as a knob to trade off between accuracy and efficiency.

Core Decomposition of Uncertain Graphs

An *uncertain* (or *probabilistic*) graph is a triple $\mathcal{G} = (V, E, p)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and p: $E \to (0, 1]$ is a function that assigns a probability of existence to each edge.

Bonchi et al. (2014b) define core decomposition in uncertain graphs based on the notion of (k, η)-*core*: given an uncertain graph $\mathcal{G} = (V, E, p)$, and a threshold $\eta \in [0, 1]$, the (k, η)-core of $\mathcal{G}$ is a maximal subgraph $\mathcal{H} = (C, E| C, p)$ such that the probability that each vertex $v \in C$ has degree at least k in $\mathcal{H}$ is greater than or equal to η, i.e., $\forall v \in C : \Pr[\deg_{\mathcal{H}}(v) \geq k] \geq \eta$. The corresponding (k, η)-*core decomposition* is defined as the set of all (k,η)-cores of $\mathcal{G}$, for a given η and all k.

To compute a (k, η)-core decomposition of an uncertain graph, Bonchi et al design an algorithm that resembles the traditional algorithm by Batagelj and Zaveršnik for the deterministic case. Let the η-*degree* of a vertex v be the maximum degree such that the probability for v to have that degree is no less than η. Bonchi et al's algorithm iteratively removes the vertex having the smallest η-degree until all vertices have been processed. Despite the similarity to Batagelj and Zaveršnik's algorithm, this method comes with a major critical point given by the computation (and update) of η-degrees. While its counterpart in the deterministic case (i.e., computing/updating the degree of a vertex) is a straightforward operation, such a step in uncertain graphs needs a great deal of attention, as approaching it naïvely may even lead to intractable (exponential) time complexity. Ultimately, Bonchi et al show how to overcome the exponential-time complexity by devising an efficient dynamic-programming method. As a result, it is demonstrated that computing the (k, η)-core decomposition of an uncertain graph can be carried out in $\mathcal{O}(\Delta \times |E|)$ time, where Δ is the maximum η-degree of a vertex in the input uncertain graph.

Core Decomposition of Temporal Graphs

A (discrete) *temporal* (or *time-evolving*) graph is a triple $G = (V, T, f)$, where V is a set of vertices, T is a set of discrete time instants, and f: $T \to 2^{V \times V}$ is a function assigning an edge set $E_t \subseteq V \times V$ to every time instant $t \in T$.

Wu et al. (2015) study the problem of core decomposition in temporal graphs. They introduce the notion of (k, h)-*core*, which is defined as follows. For any two vertices u, v, let $\pi(u, v)$ denote the total number of edges connecting u and v over all time instants, i.e., $\pi(u, v) = |\{t \mid t \in T, (u, v) \in f(t)\}|$. The (k, h)-core of a temporal graph G is a maximal subgraph H where every vertex of H is connected to at least k other vertices in H with which it shares at least h temporal edges, i.e., $\forall u \in H$: $|\{v \mid v \in H, \pi(u, v) \geq h\}| \geq k$. Wu et al. (2015) propose two distributed algorithms to compute (k, h)-cores in temporal graphs, one working under Pregel's vertex-centric computing model (Malewicz et al. 2010), and a second one following Blogel's block-centric model (Yan et al. 2014).

Core Decomposition of Multilayer Graphs

A *multilayer* (or *multiplex*, or *multidimensional*) graph is a triple $G = (V, E, L)$, where V is a set of vertices, L is a set of layers, and E is a set of per-layer edges, i.e., a set of triples (u, v, l), where $u, v \in V$ and $l \in L$.

Azimi-Tafreshi et al. (2014) define the notion of **k**-core: given a multilayer graph $G = (V, E, L)$ and an $|L|$-dimensional integer vector $\mathbf{k} = \{k_l\}_{l \in L}$, the **k**-core of G is defined as the largest subgraph in which each vertex has at least k_l edges in that subgraph, for each layer $l \in L$. Azimi-Tafreshi et al. (2014) first show that, for any given vector $\mathbf{k} = \{k_l\}_{l \in L}$, the **k**-core of a multilayer graph can be computed by iteratively removing vertices whose degree in a layer $l \in L$ is less than k_l, until no such vertices appear in the graph. Based on this, Azimi-Tafreshi et al. (2014) generalize the theory of *k*-core percolation in simple graphs (Goltsev et al. 2006) to **k**-core percolation in multilayer graphs. They derive self-consistency equations to obtain the birth points of the **k**-cores and their relative sizes for uncorrelated multilayer graphs with an arbitrary degree distribution. Such results are studied in detail for the case of graphs composed of two layers, where equations are solved for Erdös-Renyi and scale-free multilayer graphs.

Key Applications

Core decomposition has been traditionally used to quantify the global position of a vertex in a network, analyze the complex nature of a network, and discover dense substructures (Seidman 1983). It has been employed to describe the evolution of random graphs (Bollobás 1984; Luczak 1991; Dorogovtsev et al. 2006); for analyzing complex networks (e.g., the Web), in particular their hierarchies, self-similarity, centrality, and connectivity (Alvarez-Hamelin et al. 2008); for discovering cooperative processes within a network (Carmi et al. 2007); for designing effective network-visualization tools (Batagelj et al. 1999; Gaertler and Patrignani 2004; Alvarez-Hamelin et al. 2005).

As another appealing feature, core decomposition provides principled ways of speeding-up the computation of more complex definitions of a dense subgraph. As an example, it can serve as an easy method to improve the efficiency of finding maximal cliques (Eppstein et al. 2010), as a clique of size k is guaranteed to be contained into a $(k-1)$-core, which can be significantly smaller than the original input graph. Moreover, core decomposition is at the basis of linear-time approximation algorithms for the *densest-subgraph* problem (Kortsarz and Peleg 1994) and the *densest at-least-k-subgraph* problem (Andersen and Chellapilla 2009). It is also used to approximate *betweenness centrality* (Healy et al. 2006), for designing proper indexing structures (Barbieri et al. 2015), and for clustering (Giatsidis et al. 2014).

As far as social networks, core decomposition has been largely employed for community detection (Papadopoulos et al. 2012). Furthermore, Kitsak et al. (2010) show that the core index is an indicator of influence of a user with respect to information spreading in a network, while Bonchi et al. (2014b) employ core decomposition of uncertain graphs to speed-up the detection of influential spreaders. Further work on using core decomposition for influential-spreader discovery has been conducted by Pei and Makse (2013), Bae and Kim (2014), and Rossi et al. (2015). Also, core decomposition has been employed by Garcia et al. (2013) to do an “autopsy” of the dead Friendster social network and, in the context of uncertain graphs, for the problem of *task-driven team formation* (Bonchi et al. 2014b).

Finally, core decomposition has been applied in several orthogonal domains, from bioinformatics – e.g., for protein-interaction discovery (Altaf-Ul-Amin et al. 2003; Bader and Hogue 2003; Wuchty and Almaas 2005), or gene-network analysis (Cheng et al. 2013), to the inspection of large-scale software systems (Zhang et al. 2010) and an analysis of the diffusion network of political parties on Twitter (Aragón et al. 2013).

Future Directions

Despite the great deal of attention received since its introduction, several aspects of core decomposition still remain unexplored. As an example, several types of information-rich network that are commonly encountered in today’s applications, such as heterogeneous information networks, signed networks, or (edge-)labeled graphs, still miss a principled definition of the notion of *k*-core and the corresponding problem of core decomposition. Also, the streaming context abounds of interesting unstudied problems, such as approximated core-decomposition maintenance, or efficient incremental update for multiple insertion/deletion operations performed at a time. Other interesting directions include core decomposition in uncertain graphs where the assumption of independence among edge probabilities does not hold, or in temporal graphs where a time window of interest is provided as input by the user.

Cross-References

- ▶ A Method for Characterizing Communities in Dynamic Attributed Complex Networks
- ▶ A Repository for Multirelational Dynamic Networks
- ▶ Analysis and Visualization of Dynamic Networks
- ▶ Core Decomposition of Massive, Information-Rich Graphs
- ▶ Distributed Processing of Networked Data
- ▶ Mining Time-Evolving Graphs
- ▶ Multiplex Networks
- ▶ Temporal Networks

References

Aksu H, Canim M, Chang YC, Korpeoglu I, Ulusoy Ö (2014) Distributed *k*-core view materialization and maintenance for large dynamic graphs. IEEE Trans Knowledge Data Eng (TKDE) 26(10):2439–2452

Altaf-Ul-Amin M, Nishikata K, Koma T, Miyasato T, Shinbo Y, Md WC, Maeda M, Oshima T, Mori H, Kanaya S (2003) Prediction of protein functions based on *k*-cores of protein-protein interaction networks and amino acid sequences. Genome Inform 14:498–499

Alvarez-Hamelin JI, Dall'Asta L, Barrat A, Vespignani A (2005) Large scale networks fingerprinting and visualization using the *k*-core decomposition. In: Proceedings of the international conference on neural information processing systems (NIPS), pp 41–50

Alvarez-Hamelin JI, Dall'Asta L, Barrat A, Vespignani A (2008) K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. Networks Heterogeneous Media 3(2):395–411

Andersen R, Chellapilla K (2009) Finding dense subgraphs with size bounds. In: Proceedings of the international work on algorithms and models for the web-graph (WAW), pp 25–37

Aragón P, Kappler KE, Kaltenbrunner A, Laniado D, Volkovich Y (2013) Communication dynamics in Twitter during political campaigns: the case of the 2011 Spanish national election. Policy & Internet 5(2):183–206

Azimi-Tafreshi N, Gómez-Gardeñes J, Dorogovtsev SN (2014) *k*-Core percolation on multiplex networks. Phys Rev E 90(3):032816

Bader GD, Hogue CWV (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4:2

Bae J, Kim S (2014) Identifying and ranking influential spreaders in complex networks by neighborhood coreness. Phys A: Stat Mech Appl 395:549–559

Barbieri N, Bonchi F, Galimberti E, Gullo F (2015) Efficient and effective community search. Data Mining Knowledge Discov 29(5):1406–1433

Batagelj V, Zaveršnik M (2011) Fast algorithms for determining (generalized) core groups in social networks. ADAC 5(2):129–145

Batagelj V, Mrvar A, Zaversnik M (1999) Partitioning approach to visualization of large graphs. In: Proceedings of the international symposium on graph drawing, pp 90–97

Boldi P, Vigna S (2004) The webgraph framework I: compression techniques. In: Proceedings of the international conference on World Wide Web (WWW), pp 595–602

Bollobás B (1984) The evolution of random graphs. Trans Am Math Soc 286:257–274

Bonchi F, Gionis A, Gullo F, Ukkonen A (2014a) Distance oracles in edge-labeled graphs. In: Proceedings of the international conference on extending database technology (EDBT), pp 547–558

Bonchi F, Gullo F, Kaltenbrunner A, Volkovich Y (2014b) Core decomposition of uncertain graphs. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 1316–1325

Bonchi F, Gionis A, Gullo F, Tsourakakis CE, Ukkonen A (2015) Chromatic correlation clustering. ACM Trans Knowledge Discov Data 9(4):34

Carmi S, Havlin S, Kirkpatrick S, Shavitt Y, Shir E (2007) MEDUSA - new model of internet topology using *k*-shell decomposition. Proc Nat Acad Sci 104:11,150–11,154

Cheng J, Ke Y, Chu S, Ozsu MT (2011) Efficient core decomposition in massive networks. In: Proceedings of the IEEE international conference on data engineering (ICDE), pp 51–62

Cheng Y, Lu C, Wang N (2013) Local *k*-core clustering for gene networks. In: Proceedings of the IEEE international conference on bioinformatics and biomedicine, pp 9–15

Dasari NS, Ranjan D, Zubair M (2014) ParK: an efficient algorithm for *k*-core decomposition on multicore processors. In: Proceedings of the IEEE international conference on big data, pp 9–16

Dorogovtsev SN, Goltsev AV, Mendes JFF (2006) *k*-Core organization of complex networks. Phys Rev Lett 96(4):040,601

Eppstein D, Löffler M, Strash D (2010) Listing all maximal cliques in sparse graphs in near-optimal time. In: Proceedings of the international symposium on algorithms and computation (ISAAC), pp 403–414

Gaertler M, Patrignani M (2004) Dynamic analysis of the autonomous system graph. In: Proceedings of the international work on inter-domain performance and simulation, pp 13–24

Garcia D, Mavrodiev P, Schweitzer F (2013) Social resilience in online communities: the autopsy of friendster. In: Proceedings of ACM conference on online social networks (COSN), pp 39–50

Giatsidis C, Thilikos DM, Vazirgiannis M (2011) D-cores: measuring collaboration of directed graphs based on degeneracy. In: Proceedings of IEEE international conference on data mining (ICDM), pp 201–210

Giatsidis C, Malliaros FD, Thilikos DM, Vazirgiannis M (2014) CoreCluster: a degeneracy based graph clustering framework. In: Proceedings of AAAI conference on artificial intelligence, pp 44–50

Goltsev AV, Dorogovtsev SN, Mendes JFF (2006) *k*-core (bootstrap) percolation on complex networks: critical phenomena and nonlocal effects. Physical Review E 73:056101

Healy J, Janssen J, Milios EE, Aiello W (2006) Characterization of graphs using degree cores. In: Proceedings of international work on algorithms and models for the Web-Graph (WAW), pp 137–148

Khan A, Bonchi F, Gionis A, Gullo F (2014) Fast reliability search in uncertain graphs. In: Proceedings of international conference on extending database technology (EDBT), pp 535–546

Khan A, Gullo F, Wohler T, Bonchi F (2015) Top-k reliable edge colors in uncertain graphs. In: Proceedings of the ACM international conference on information and knowledge management (CIKM), pp 1851–1854

Khaouid W, Barsky M, Srinivasan V, Thomo A (2015) K-core decomposition of large networks on a single PC. Proc VLDB Endowment 9(1):13–23

Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, Makse HA (2010) Identifying influential spreaders in complex networks. Nat Phys 6(11):888–893

Kortsarz G, Peleg D (1994) Generating sparse 2-spanners. J Algorithms 17(2):222–236

Kyrola A, Blelloch G, Guestrin C (2012) GraphChi: Large-scale graph computation on just a PC. In: Proceedings of the USENIX conference on operating systems design and implementation (OSDI), pp 31–46

Lee VE, Ruan N, Jin R, Aggarwal C (2010) A survey of algorithms for dense subgraph discovery. In: Managing and mining graph data. Springer, New York, pp 303–336

Li R, Yu JX, Mao R (2014) Efficient core maintenance in large dynamic graphs. IEEE Trans Knowledge Data Eng 26(10):2453–2465

Luczak T (1991) Size and connectivity of the *k*-core of a random graph. Discret Math 91(1):61–68

Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 135–146

Malliaros FD, Papadopoulos AN, Vazirgiannis M (2016) Core decomposition in graphs: concepts, algorithms and applications. In: Proceedings of international conference on extending database technology (EDBT), pp 720–721

Montresor A, Pellegrini FD, Miorandi D (2013) Distributed *k*-core decomposition. IEEE Trans Parallel Distribut Syst 24(2):288–300

O'Brien MP, Sullivan BD (2014) Locally estimating core numbers. In: Proceedings of IEEE international conference on data mining (ICDM), pp 460–469

Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P (2012) Community detection in social media. Data Mining Knowledge Discov 24(3):515–554

Parchas P, Gullo F, Papadias D, Bonchi F (2015) Uncertain graph processing through representative instances. ACM Trans Database Syst 40(3):20

Pei S, Makse HA (2013) Spreading dynamics in complex networks. J Stat Mech Theory Exper 12

Rossi MG, Malliaros FD, Vazirgiannis M (2015) Spread it good, spread it fast: Identification of influential nodes in social networks. In: Proceedings of the internatinal conference on World Wide Web - Companion Volume (WWW Companion), pp 101–102

Ruchansky N, Bonchi F, García-Soriano D, Gullo F, Kourtellis N (2015) The minimum wiener connector problem. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 1587–1602

Saríyüce AE, Gedik B, Jacques-Silva G, Wu KL, Çatalyürek ÜV (2013) Streaming algorithms for *k*-core decomposition. Proc VLDB Endowment 6(6):433–444

Saríyüce AE, Gedik B, Jacques-Silva G, Wu KL, Çatalyürek ÜV (2016) Incremental *k*-core decomposition: algorithms and evaluation. VLDB J 25(3):425–447

Seidman SB (1983) Network structure and minimum degree. Soc Networks 5(3):269–287

Tsourakakis C, Bonchi F, Gionis A, Gullo F, Tsiarli M (2013) Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 104–112

Wen D, Qin L, Zhang Y, Lin X, Yu JX (2016) I/O efficient core graph decomposition at web scale. In: Proceedings of IEEE international conference on data engineering (ICDE)

Wu H, Cheng J, Lu Y, Ke Y, Huang Y, Yan D, Wu H (2015) Core decomposition in large temporal graphs. In: Proceedings of IEEE internation conference on big data, pp 649–658

Wuchty S, Almaas E (2005) Peeling the yeast protein network. Proteomics 5(2):444–449

Yan D, Cheng J, Lu Y, Ng W (2014) Blogel: a block-centric framework for distributed computation on real-world graphs. Proc VLDB Endowment 7(14):1981–1992

Zhang H, Zhao H, Cai W, Liu J, Zhou W (2010) Using the *k*-core decomposition to analyze the static structure of large-scale software systems. J Supercomput 53(2):352–369