Core Decomposition and Densest Subgraph in Multilayer Networks

Edoardo Galimberti
ISI Foundation & Univ. of Turin, Italy
edoardo.galimberti@isi.it

Francesco Bonchi
ISI Foundation, Italy
francesco.bonchi@isi.it

Francesco Gullo
UniCredit, R&D Dept., Italy
gullof@acm.org

ABSTRACT

Multilayer networks are a powerful paradigm to model complex systems, where various relations might occur among the same set of entities. Despite the keen interest in a variety of problems, algorithms, and analysis methods in this type of network, the problem of extracting dense subgraphs has remained largely unexplored.

As a first step in this direction, in this work we study the problem of *core decomposition of a multilayer network*. Unlike the single-layer counterpart in which cores are all nested into one another and can be computed in linear time, the multilayer context is much more challenging as no total order exists among multilayer cores; rather, they form a lattice whose size is exponential in the number of layers. In this setting we devise three algorithms which differ in the way they visit the core lattice and in their pruning techniques. We assess time and space efficiency of the three algorithms on a large variety of real-world multilayer networks.

We then move a step forward and showcase an application of the multilayer core-decomposition tool to the problem of *densest-subgraph extraction from multilayer networks*. We introduce a definition of multilayer densest subgraph that trades-off between high density and number of layers in which the high density holds, and show how multilayer core decomposition can be exploited to approximate this problem with quality guarantees.

1 INTRODUCTION

In social media and social networks, as well as in several other real-world contexts – such as biological and financial networks, transportation systems and critical infrastructures – there might be multiple types of relation among entities. Data in these domains is typically modeled as a *multilayer network* (also known as multidimensional network), i.e., a graph where multiple edges of different types may exist between any pair of vertices [17].

Extracting dense structures from large graphs has emerged as a key graph-mining primitive in a variety of scenarios [28]. Although the literature on multilayer graphs is growing fast, the problem of extracting dense subgraphs in this type of graph has been, surprisingly, largely unexplored. In standard graphs, among the many definitions of dense structures, *core decomposition* plays a central role as it can be computed in linear time [7], and can be used to speed-up/approximate dense-subgraph extraction according to various other definitions. For instance, core decomposition provides a heuristic for maximal-clique finding [18], as a k-clique is guaranteed to be contained into a (k–1)-core, which can be significantly smaller than the original graph. Also, core decomposition is at the basis of approximation algorithms for the densest-(at-least-k-)subgraph problem [2, 27] and betweenness centrality [23].

In this work we study the problem of *core decomposition in multilayer networks*, and show how it finds application to the problem of *densest-subgraph extraction from multilayer networks*.

1.1 Background and related work

Core decomposition. Let us first recall the classic notion of core decomposition in a simple, single-layer, graph $G = (V, E)$. For every vertex $u \in V$, let $deg(u)$ and $deg_S(u)$ denote the degree of u in G and in a subgraph S of G, respectively. Also, given a set of vertices $C \subseteq V$, let $E[C]$ denote the subset of edges induced by C.

Definition 1 (core decomposition). *The* k-core *(or core of order* k*) of* G *is a* maximal *subgraph* $G[C_k] = (C_k, E[C_k])$ *such that* $\forall u \in C_k : deg_{C_k}(v) \geq k$. *The set of all* k*-cores* $G = C_0 \supseteq C_1 \supseteq \cdots \supseteq C_{k^*}$ ($k^* = \arg\max_k C_k \neq \emptyset$) *is the* core decomposition *of* G.

Core decomposition can be computed in linear time by iteratively removing the smallest-degree vertex and setting its core number as its degree at the time of removal [7]. Core decomposition has been employed for analyzing/visualizing complex networks [1] in several domains, e.g., bioinformatics [40], and social networks [26]. It has been studied under various settings, such as distributed [29, 32], streaming [33], and disk-based [16], and for various types of graph, such as uncertain [13], directed [21], and weighted [20] graphs.

In this paper we adopt the definition of a multi-layer core by Azimi-Tafreshi *et al.* [4], which study the core-percolation problem from a physics standpoint, without providing any algorithm. They characterize cores on 2-layer Erdős-Rényi and 2-layer scale-free networks, then they analyze real-world (2-layer) air-transportation networks. To the best of our knowledge, *no prior work has studied how to efficiently compute the complete core decomposition of multilayer networks.*

Densest subgraph. Several notions of *density* exist in the literature, each of which leading to a different version of the problem of extracting a single dense subgraph. While most variants are **NP**-hard, or even inapproximable, extracting dense subgraphs according to the *average-degree density* (i.e., two times the number of edges divided by the number of vertices) is solvable in polynomial time [22]. As a result, such a density has attracted most of the research in the field, so that the subgraph maximizing the average-degree density is commonly referred to as the *densest subgraph.*

CIKM'17, November 6–10, 2017, Singapore.

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132993

Goldberg [22] provides an exact solution based on iteratively solving ad-hoc-defined minimum-cut problem instances. Although principled and elegant, Goldberg's algorithm cannot scale to large graphs. Asahiro *et al.* [3] and Charikar [15] provide a more efficient (linear-time) $\frac{1}{2}$-approximation algorithm. The algorithm greedily removes the smallest-degree vertex, until the graph has become empty. Among all subgraphs produced during this vertex-removal process, the densest one is returned as output. Note that this algorithm resembles the one used for core decomposition. In fact, it can be proved that the inner-most core of a graph is itself a $\frac{1}{2}$-approximation of the densest subgraph.

The densest-subgraph problem has been studied in a streaming setting [5, 9], and a top-k fashion [6, 19]. Also, a number of works depart from the classic average-degree density and focus on, e.g., quasi-clique-based density [36], or triangle density [37, 38].

Dense structures in multilayer networks. Surprisingly, little attention has been paid to the problem of extracting dense subgraphs from multilayer networks. To the best of our knowledge, the only attempt to generalize the densest-subgraph problem to the multilayer context has been carried out by Jethava *et al.* [24], who formulate the *densest common subgraph* problem, i.e., find a subgraph maximizing the minimum average degree over *all* layers. Jethava *et al.* devise a linear-programming formulation of the problem, as well as a heuristic greedy algorithm. However, no approximation algorithms with provable quality guarantees are provided. Some other works deal with the specific case of 2-layer networks [34, 39, 43]. Other marginally-related works focus on *community detection* [8, 14, 30, 31, 35, 42], *subspace clustering* [11], i.e., finding clusters of vertices that are densely connected by edges with similar labels for all possible label sets, *mining closed relational graphs* [41], i.e., finding all frequent subgraphs of a multilayer graph having large minimum cut, and *extracting frequent cross-graph quasi-cliques* [25], i.e., finding all subgraphs satisfying the quasi-clique condition in at least a fraction of layers equal to *min_sup*, and having size larger than *min_size*.

In this work we introduce a formulation of the densest-subgraph problem in multilayer networks that generalizes the densest common subgraph problem studied in [24] by trading off between high density and number of layers where the high density is observed. We apply our multilayer core-decomposition tool to solve both our general formulation of multilayer densest subgraph and the specific densest-common-subgraph variant by Jethava *et al.* [24] with provable approximation guarantees. Furthermore, we show that multilayer core decomposition can profitably be exploited to speed-up the problem of *finding frequent cross-graph quasi-cliques* [25].

1.2 Challenges and contributions

Let $G = (V, E, L)$ be a multilayer graph, where V is a set of vertices, L is a set of layers, and $E \subseteq V \times V \times L$ is a set of edges. Given an $|L|$-dimensional integer vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, the *multilayer* $\mathbf{k}$*-core* of G is a maximal subgraph whose vertices have at least degree k_ℓ in that subgraph, for all layers ℓ. Vector $\mathbf{k}$ is dubbed *coreness vector* of the core. The set of all *non-empty* and *distinct* multilayer cores constitutes the *multilayer core decomposition* of G. A major challenge of computing the complete core decomposition of multilayer networks is that *the number of multilayer cores can be exponential in the number of layers*, which makes the problem inherently hard as no polynomial-time algorithm may exist in the general case. In fact, unlike the single-layer case where cores are all nested into each other, no total order exists among multilayer cores. Rather, they form a *core lattice* defining a relation of partial containment. As a result, the multilayer core-decomposition problem cannot be solved in linear time like in single-layer graphs: algorithms in the multilayer setting must be crafted carefully to handle this exponential blowup, and avoid, to the maximum possible extent, the computation of unnecessary (i.e., empty or non-distinct) cores.

We devise three algorithms that exploit effective pruning rules during the visit of the lattice. The first two methods are based on a BFS and a DFS strategy, respectively: the BFS method exploits the rule that a core is contained into the intersection of all its fathers in the lattice, while the DFS method iteratively performs a single-layer core decomposition that computes cores along a path from a non-leaf lattice core to a leaf all at once. The third method adopts a HYBRID strategy embracing the main pros of BFS and DFS, and equipped with a *look-ahead* mechanism to skip non-distinct cores.

As a major application of multilayer core decomposition, we turn our attention to the problem of *extracting the densest subgraph from a multilayer network.* To the best of our knowledge, the only existing attempt to formulate this problem is the one by Jethava *et al.* [24], who aim at extracting a subgraph that maximizes the minimum average degree over *all* layers. A major limitation of this formulation is that, considering all layers, even the noisy/insignificant layers would contribute to selecting the output subgraph, which would be not really dense, thus preventing us from finding a subgraph being dense in a still large subset of layers. Another simplistic approach at the other end of the spectrum corresponds to flattening the input multilayer graph and resorting to single-layer densest-subgraph extraction. However, this would mean disregarding the different semantics of the layers, incurring in a severe information loss.

Within this view, in this work we introduce a problem statement that generalizes the formulation by Jethava *et al.* by accounting for a trade-off between high density and number of layers exhibiting the high density. Given a multilayer graph $G = (V, E, L)$, the average-degree density of a subset of vertices S in a layer ℓ is defined as the number of edges induced by S in ℓ divided by the size of S, i.e., $\frac{|E_\ell[S]|}{|S|}$. We define the *multilayer densest subgraph* as the subset of vertices S^* such that the function

$$\max_{\hat{L} \subseteq L} \min_{\ell \in \hat{L}} \frac{|E_\ell[S^*]|}{|S^*|} |\hat{L}|^\beta$$

is maximized. $\beta \in \mathbb{R}^+$ is a parameter controlling the importance of the two sides of the same coin of our problem, i.e., high density and number of layers exhibiting such a density. It can be observed that this problem statement naturally achieves the desired trade-off: the larger the subset $\hat{L}$ of selected layers, the smaller the minimum density $\min_{\ell \in \hat{L}} \frac{|E_\ell[S]|}{|S|}$ registered in those layers.

Similarly to the single-layer case in which the core decomposition can be used to obtain a $\frac{1}{2}$-approximation of the densest subgraph, in this work we show that computing the multilayer core decomposition of the input graph and selecting the core maximizing the proposed multilayer density function achieves a $\frac{1}{2|L|^\beta}$-approximation for the general multilayer-densest-subgraph problem formulation, and a $\frac{1}{2}$-approximation for the Jethava *et al.*'s formulation. The latter one is a per-se noteworthy advancement to the state of the art, as, to the best of our knowledge, no approximation algorithms with provable quality guarantees have been so far devised for the Jethava *et al.*'s problem.

Summarizing, this work has the following contributions:

- We define the problem of *core decomposition in multilayer networks*, characterizing its usefulness, its relation to other problems, and its intrinsic complexity. We then devise three algorithms that solve multilayer core decomposition efficiently based on different pruning techniques.
- We study the problem of *densest-subgraph extraction in multilayer networks*, formulating it as an optimization problem that trades-off between high density and number of layers exhibiting high density.
- We exploit multilayer core decomposition to efficiently approximate both our definition of multilayer densest subgraph and the one by Jethava *et al.* [24] with provable quality guarantees.

2 MULTILAYER CORE DECOMPOSITION

We are given an undirected multilayer graph $G = (V, E, L)$, where V is a set of vertices, L is a set of layers, and $E \subseteq V \times V \times L$ is a set of edges. Let E_ℓ denote the subset of edges in layer $\ell \in L$. For a vertex $u \in V$ we denote by $deg(u, \ell)$ and $deg(u)$ its degree in layer ℓ and over all layers, respectively, i.e., $deg(u, \ell) = |\{e = (u, v, \ell) : e \in E_\ell\}|$, $deg(u) = |\{e = (u, v, \ell) : e \in E\}| = \sum_{\ell \in L} deg(u, \ell)$.

For a subset of vertices $S \subseteq V$ we denote by $G[S]$ the subgraph of G induced by S, i.e., $G[S] = (S, E[S], L)$, where $E[S] = \{e = (u, v, \ell) \mid e \in E, u \in S, v \in S\}$. For a vertex $u \in V$ we denote by $deg_S(u, \ell)$ and $deg_S(u)$ its degree in subgraph S considering layer ℓ only and all layers, respectively, i.e., $deg_S(u, \ell) = |\{e = (u, v, \ell) : e \in E_\ell[S]\}|$, $deg_S(u) = |\{e = (u, v, \ell) : e \in E[S]\}| = \sum_{\ell \in L} deg_S(u, \ell)$. Finally, let $\mu(\ell)$ and $\mu(\hat{L})$ denote the minimum degree of a vertex in layer ℓ and in a subset $\hat{L} \subseteq L$ of layers, respectively. Let also $\mu(S, \ell)$ and $\mu(S, \hat{L})$ denote the corresponding counterparts of $\mu(\ell)$ and $\mu(\hat{L})$ for a subgraph (induced by a vertex set) S.

A core of a multilayer graph is characterized by an $|L|$-dimensional integer vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, dubbed *coreness vector*, whose components k_ℓ denote the minimum degree allowed in layer ℓ: [1]

Definition 2 (multilayer core and coreness vector). *Given a multilayer graph $G = (V, E, L)$ and an $|L|$-dimensional integer vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, the multilayer* $\mathbf{k}$-core *of G is a* maximal *subgraph $G[C] = (C \subseteq V, E[C], L)$ such that $\forall \ell \in L : \mu(C, \ell) \geq k_\ell$. Vector $\mathbf{k}$ is referred to as* coreness vector *of $G[C]$.*

Given a coreness vector $\mathbf{k}$, we denote by $C_\mathbf{k}$ the corresponding core. Also, as a $\mathbf{k}$-core is fully identified by the vertices belonging to it, we hereinafter refer to it by its vertex set $C_\mathbf{k}$ and the induced subgraph $G[C_\mathbf{k}]$ interchangeably.

A set of vertices $C \subseteq V$ may correspond to multiple cores. For instance, in the graph in Figure 1 the set {A,B,D,E} corresponds to both $(3, 0)$-core and $(3, 1)$-core. In other words, a multilayer core can be described by more than one coreness vector. However, as formally shown next, among such multiple coreness vectors there exists one and only one that is not dominated by any other. We call this vector the *maximal coreness vector* of C. In the example in Figure 1 the maximal coreness vector of {A,B,D,E} is $(3, 1)$.

Definition 3 (maximal coreness vector). *Let $G = (V, E, L)$ be a multilayer graph, $C \subseteq V$ be a core of G, and $\mathbf{k} = [k_\ell]_{\ell \in L}$ be a coreness vector of C. $\mathbf{k}$ is said* maximal *if there does not exist any coreness vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ of C such that $\forall \ell \in L : k'_\ell \geq k_\ell$ and $\exists \hat{\ell} \in L : k'_{\hat{\ell}} > k_{\hat{\ell}}$.*

Theorem 1. *Multilayer cores have a unique maximal coreness vector.*

Proof. We prove the theorem by contradiction. Assume two maximal coreness vectors $\mathbf{k} = [k_\ell]_{\ell \in L} \neq \mathbf{k}' = [k'_\ell]_{\ell \in L}$ exist for a multilayer core C. As $\mathbf{k} \neq \mathbf{k}'$ and they are both maximal, there exist two layers $\hat{\ell}$ and $\bar{\ell}$ such that $k_{\hat{\ell}} > k'_{\hat{\ell}}$ and $k'_{\bar{\ell}} > k_{\bar{\ell}}$. By definition of multilayer core (Definition 2), it holds that $\forall \ell \in L : \mu(C, \ell) \geq k_\ell, \mu(C, \ell) \geq k'_\ell$. This means that the vector $\mathbf{k}^* = [k^*_\ell]_{\ell \in L}$, with $k^*_\ell = \max\{k_\ell, k'_\ell\}, \forall \ell \in L$, is a further coreness vector of C. For this vector it holds that $\forall \ell \neq \hat{\ell}, \ell \neq \bar{\ell} : k^*_\ell \geq k'_\ell$, $k^*_{\hat{\ell}} > k'_{\hat{\ell}}$, and $k^*_{\bar{\ell}} > k_{\bar{\ell}}$. Thus, $\mathbf{k}^*$ dominates both $\mathbf{k}$ and $\mathbf{k}'$, which contradicts the hypothesis of maximality of $\mathbf{k}$ and $\mathbf{k}'$. The theorem follows. □

The (first) problem we tackle in this work is the following:

Problem 1 (Multilayer Core Decomposition). *Given a multilayer graph G, find the set of all* non-empty *and* distinct *cores of G, along with their corresponding maximal coreness vectors. Such a set forms the* multilayer core decomposition *of G.*

Characterization. Cores of a single-layer graph are all nested one into another. This makes it possible to define the notions of *innermost core*, i.e., the core of highest order, and *core index* (or *core number*) of a vertex u, i.e, the highest order of a core containing u. In the multilayer setting the picture is more complex, as multilayer cores are not necessarily all nested into each other. As a result, the core index of a vertex is not unambiguously defined, while there can exist multiple inner-most cores:

Definition 4 (inner-most multilayer cores). *The* inner-most cores *of a multilayer graph are all those cores with maximal coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ such that there does not exist any other core with coreness vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ where $\forall \ell \in L : k'_\ell \geq k_\ell$ and $\exists \hat{\ell} \in L : k'_{\hat{\ell}} > k_{\hat{\ell}}$.*

To this purpose, look at the example in Figure 1. It can be observed that: (*i*) cores are not nested into each other, (*ii*) $(3, 1)$-core, $(1, 3)$-core and $(2, 2)$-core are the inner-most cores, and (*iii*) vertices B and E belong to (inner-most) cores $(3, 1)$, $(1, 3)$, and $(2, 2)$, thus making their core index not unambiguously defined.

Another interesting insight into the notion of multilayer cores is about their relationship with (quasi-)cliques. In single-layer graphs it is well-known that cores can be exploited to speed-up the problem of finding cliques, as a clique of size k is guaranteed to be contained into the $(k - 1)$-core. Interestingly, a similar relationship holds in the multilayer context too. Given a multilayer graph $G = (V, E, L)$, a layer $\ell \in L$, and a real number $\gamma \in (0, 1]$, a subgraph $G[S] = (S \subseteq V, E[S], L)$ of G is said to be a γ*-quasi-clique* in layer ℓ if all its vertices have at least $\gamma(|S| - 1)$ neighbors in layer ℓ within S, i.e.,

[1] Definition 2 corresponds to the notion of $\mathbf{k}$-core used by Azimi-Tafreshi *et al.* [4] for the multilayer core-percolation problem. As discussed in Section 1.1, Azimi-Tafreshi *et al.* do not study (or devise any algorithm for) the problem of computing the entire core decomposition of a multilayer graph. Core percolation is studied by analyzing *a single* core of interest computed with the simple iterative-peeling algorithm (Algorithm 1).

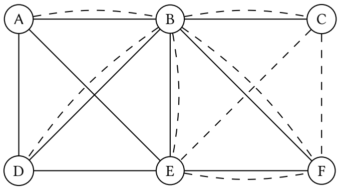

Figure 1: Example 2-layer graph (solid edges refer to the first layer, while dashed edges to the second layer) with the following k-cores: (0, 0) = (1, 0) = (0, 1) = (1, 1) = {**A,B,C,D,E,F**}, (2, 0) = (2, 1) = {**A,B,D,E,F**}, (3, 0) = (3, 1) = {**A,B,D,E**}, (0, 2) = (1, 2) = (0, 3) = (1, 3) = {**B,C,E,F**}, (2, 2) = {**B,E,F**} **.**

$\forall u \in S : deg_S(u, \ell) \geq \gamma(|S| - 1)$. Jiang *et al.* [25] study the problem of extracting *frequent cross-graph quasi-cliques*: given a multilayer graph $G = (V, E, L)$, a function $\Gamma : L \rightarrow (0, 1]$ assigning a real value to every layer in L, a real number $min\_sup \in (0, 1]$, and an integer $min\_size \geq 1$, find all maximal subgraphs $G[S]$ of G of size larger than min_size such that there exist at least $min\_sup \times |L|$ layers ℓ for which $G[S]$ is a $\Gamma(\ell)$-quasi-clique.

The following theorem shows that a frequent cross-graph quasi-clique of size K is necessarily contained into a $\mathbf{k}$-core described by a maximal coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ such that there exists a fraction of at least min_sup layers ℓ where $k_\ell = \lfloor \Gamma(\ell)(K-1) \rfloor$. Similarly to the single-layer setting, this finding can profitably be exploited, among others, to speed-up the multilayer (quasi-)clique extraction process.

Theorem 2. *Given a multilayer graph $G = (V, E, L)$, a real-valued function $\Gamma : L \rightarrow (0, 1]$, a real number* min_sup $\in (0, 1]$*, and an integer* min_size ≥ 1*, a frequent cross-graph quasi-clique of size K of G complying with parameters Γ,* min_sup*, and* min_size *is contained into a $\mathbf{k}$-core with maximal coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ such that* $|\{\ell \in L : k_\ell = \lfloor \Gamma(\ell)(K-1) \rfloor\}| \geq \text{min\_sup} \times |L|$.

Proof. Assume that a cross-graph quasi-clique S of size K is not contained into any $\mathbf{k}$-core with maximal coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ such that $|\{\ell \in L : k_\ell = \lfloor \Gamma(\ell)(K-1) \rfloor\}| \geq min\_sup \times |L|$. This means that S contains a vertex u such that $|\{\ell \in L : deg_S(u, \ell) \geq \Gamma(\ell)(K-1)\}| < min\_sup \times |L|$. This violates the definition of cross-graph quasi-clique. □

3 ALGORITHMS

A major challenge of the Multilayer Core Decomposition problem is that the number of multilayer cores to be output may be exponential in the number of layers. Specifically, denoting by K_ℓ the maximum order of a core for layer ℓ, the number of multilayer cores is $O(\prod_{\ell \in L} K_\ell)$. This makes Multilayer Core Decomposition intrinsically hard: *in the general case, no polynomial-time algorithm can exist.* The challenge in this context hence lies in handling this exponential blowup by early recognizing and skipping unnecessary portions of the core lattice, such as non-distinct and/or empty cores.

Given a multilayer graph $G = (V, E, L)$ and a coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, finding the corresponding core can easily be solved in $O(|E| + |V| \times |L|)$ time by iteratively removing a vertex u having $deg_{G'}(u, \ell) < k_\ell$ in some layer ℓ, where G' denotes the current graph resulting from all previous vertex removals (Algorithm 1, where the set S of vertices to be considered is set to $S = V$). Therefore, a naïve algorithm to compute the entire multilayer core decomposition consists of generating all possible coreness vectors, run the multilayer core-detection algorithm just described for each of such vectors, and retain only non-empty and distinct cores. This naïve method requires all vectors $[k_\ell]_{\ell \in L}$, where each k_ℓ component is varied within the interval $[0..K_\ell]$.[2] This corresponds to a $\Theta(\prod_{\ell \in L} K_\ell)$ number of vectors. As a result, the overall time complexity of the method is $O\Big((|E| + |V| \times |L|) \times \prod_{\ell \in L} K_\ell\Big)$.

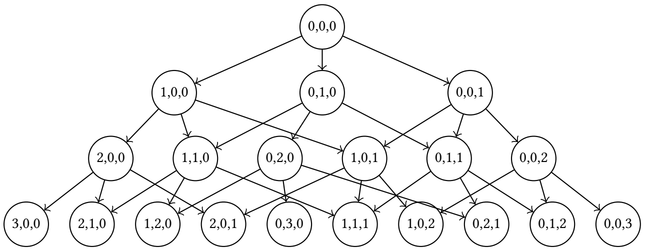

Figure 2: Core lattice of a 3-layer graph.

This approach has two major weaknesses: (*i*) each core is computed starting from the whole input graph, and (*ii*) by enumerating all possible coreness vectors beforehand a lot of non-distinct and/or empty (thus, unnecessary) cores may be computed. In the following we present three methods that solve Multilayer Core Decomposition much more efficiently.

3.1 Search space

Although multilayer cores are not all nested into each other, a notion of partial containment can still be defined. Indeed, it can easily be observed that a $\mathbf{k}$-core with coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ is contained into any $\mathbf{k}'$-core described by a coreness vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ whose components k'_ℓ are all no more than components k_ℓ, i.e., $k'_\ell \leq k_\ell, \forall \ell \in L$. This result is formalized next:

Fact 1. *Given a multilayer graph $G = (V, E, L)$ and two cores $C_\mathbf{k}$ and $C_{\mathbf{k}'}$ of G with coreness vectors $\mathbf{k} = [k_\ell]_{\ell \in L}$ and $\mathbf{k}' = [k'_\ell]_{\ell \in L}$, respectively, it holds that if $\forall \ell \in L : k'_\ell \leq k_\ell$, then $C_\mathbf{k} \subseteq C_{\mathbf{k}'}$.*

Proof. Combining the definition of multilayer core (Definition 2) and the hypothesis on vectors $\mathbf{k}$ and $\mathbf{k}'$, it holds that $\forall \ell \in L : \mu(C_\mathbf{k}, \ell) \geq k_\ell \geq k'_\ell$. This means that $C_\mathbf{k}$ satisfies the definition of $\mathbf{k}'$-core, thus implying that all vertices in $C_\mathbf{k}$ are part of $C_{\mathbf{k}'}$ too. The fact follows. □

Based on Fact 1, the search space of our problem can be represented as a lattice defining a partial order among all cores (Figure 2). Such a lattice, which we call the *core lattice*, corresponds to a dag where nodes represent cores,[3] and links represent relationships of containment between cores (a “father” node contains all its “child” nodes). We assume the core lattice keeping track of non-empty and not necessarily distinct cores: a core is present in the lattice as many times as the number of its coreness vectors. Each level i of the lattice represents the children of cores at lattice level $i - 1$. In particular, level i contains all those cores whose coreness vector results from increasing one and only one component of its fathers' coreness vector by one. Formally, a lattice level i contains all $\mathbf{k}$-cores with coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ such that there exists a core at lattice level $i-1$ with coreness vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ where: $\exists \ell \in L : k_\ell = k'_\ell + 1$, and $\forall \hat{\ell} \neq \ell : k_{\hat{\ell}} = k'_{\hat{\ell}}$. As a result, level 0 contains the root only, which corresponds to the whole input graph (i.e., the $[0]_{|L|}$-core), the leaves correspond to inner-most cores,

[2] K_ℓ values can be derived beforehand by computing a single-layer core decomposition in each layer ℓ. This process overall takes $O(|E|)$ time.

[3] Throughout the paper we use the term “node” to refer to elements of the core lattice, and “vertex” for the elements of the multilayer graph.

and any non-leaf node has at least one and at most $|L|$ children. Moreover, every level i contains all cores whose coreness-vector components sum to i.

Solving the Multilayer Core Decomposition problem is hence equivalent to building the core lattice of the input graph. The efficient methods we present next are all based on smart core-lattice building strategies that extract cores from smaller subgraphs, while also attempting to minimize the visit/computation of unnecessary (i.e., empty/non-distinct) cores.

3.2 Breadth-first algorithm

Two interesting corollaries can be derived from Fact 1. First, any non-empty $\mathbf{k}$-core is necessarily contained in the intersection of all its father nodes of the core lattice. Second, any non-empty $\mathbf{k}$-core has *exactly* as many fathers as the number of non-zero components of its coreness vector $\mathbf{k}$:

Corollary 1. *Given a multilayer graph G, let C be a core of G and $\mathcal{F}(C)$ be the set of fathers of C in the core lattice of G. It holds that $C \subseteq \bigcap_{\hat{C} \in \mathcal{F}(C)} \hat{C}$.*

Proof. By definition of core lattice, the coreness vector of all father cores $\mathcal{F}(C)$ of C is dominated by the coreness vector of C. Thus, according to Fact 1, it holds that $C \subseteq C'$, $\forall C' \in \mathcal{F}(C)$. Assume a vertex $u \notin \bigcap_{\hat{C} \in \mathcal{F}(C)} \hat{C}$, $u \in C$ exists. This implies that there exists a father core $C' \in \mathcal{F}(C)$ such that $C \nsubseteq C'$, thus leading to a contradiction. □

Corollary 2. *Given a multilayer graph G, let C be a core of G with coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, and $\mathcal{F}(C)$ be the set of fathers of C in the core lattice of G. It holds that $|\mathcal{F}(C)| = |\{k_\ell : \ell \in L, k_\ell > 0\}|$.*

Proof. By definition of core lattice, a core C at level i is assigned a coreness vector whose components sum to i, while the fathers $\mathcal{F}(C)$ of C have coreness vector whose components sum to $i - 1$. Then, the coreness vector of a father of C can be obtained by decreasing a non-zero component of the coreness vector of C by one (zero components would lead to negative coreness vector components, thus they do not count). This means that the number of fathers of C is upper-bounded by the non-zero components of its coreness vector. More precisely, the number of fathers of C is exactly equal to this number, as, according to Corollary 1, no father of C can be empty, otherwise C would be empty too and would not be part of the core lattice. □

The above corollaries pave the way to a breadth-first search building strategy of the core lattice, where cores are generated level-by-level by properly exploiting the rules in the two corollaries (Algorithm 2). Although the worst-case time complexity of this bfs-ml-cores method remains unchanged with respect to the naïve algorithm, the bfs method is expected to be much more efficient in practice, due to the following main features: (*i*) cores are not computed from the initial graph every time, but from a much smaller subgraph given by the intersection of all their fathers; (*ii*) in many cases, i.e., when the rule in Corollary 2 (which can be checked in constant time) arises, no overhead due to the intersection among father cores is required; (*iii*) the number of empty cores computed is limited, as no empty core may be generated from a core that has already been recognized as empty.

Algorithm 1 $\mathbf{k}$-core

```
Input: A multilayer graph $G = (V, E, L)$, a set $S \subseteq V$ of vertices, an
       $|L|$-dimensional integer vector $\mathbf{k} = [k_\ell]_{\ell \in L}$.
Output: The $\mathbf{k}$-core $C_\mathbf{k}$ of $G$.
while $\exists u \in S, \exists \ell \in L : deg_S(u, \ell) < k_\ell$ do
    $S \leftarrow S \setminus \{u\}$
$C_\mathbf{k} = S$
```

Algorithm 2 bfs-ml-cores

```
Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathbf{C}$ of all non-empty multilayer cores of $G$.
$\mathbf{C} \leftarrow \emptyset$, $\mathbf{Q} \leftarrow \{[0]_{|L|}\}$, $\mathcal{F}([0]_{|L|}) \leftarrow \emptyset$     {$\mathcal{F}$ keeps track of father nodes}
while $\mathbf{Q} \neq \emptyset$ do
    dequeue $\mathbf{k} = [k_\ell]_{\ell \in L}$ from $\mathbf{Q}$
    if $|\{k_\ell : k_\ell > 0\}| = |\mathcal{F}(\mathbf{k})|$ then                {Corollary 2}
        $F_\cap \leftarrow \bigcap_{F \in \mathcal{F}(\mathbf{k})} F$                      {Corollary 1}
        $C_\mathbf{k} \leftarrow$ $\mathbf{k}$-core($G$, $F_\cap$, $\mathbf{k}$)                 {Algorithm 1}
        if $C_\mathbf{k} \neq \emptyset$ then
            $\mathbf{C} \leftarrow \mathbf{C} \cup \{C_\mathbf{k}\}$
            for all $\ell \in L$ do                             {enqueue child nodes}
                $\mathbf{k}' \leftarrow [k_1, \ldots, k_\ell + 1, \ldots, k_{|L|}]$
                enqueue $\mathbf{k}'$ into $\mathbf{Q}$
                $\mathcal{F}(\mathbf{k}') \leftarrow \mathcal{F}(\mathbf{k}') \cup \{C_\mathbf{k}\}$
```

3.3 Depth-first algorithm

Although being much smarter than the naïve method, bfs-ml-cores still has some limitations. First, it visits every core as many times as the number of its fathers in the core lattice. Also, as a second limitation, consider a path $\mathcal{P}$ of the lattice connecting a non-leaf node to a leaf by varying the same ℓ-th component of the corresponding coreness vectors. It is easy to see that the computation of all cores within $\mathcal{P}$ with bfs-ml-cores takes $O(|\mathcal{P}| \times (|E| + |V| \times |L|))$ time, as the core-decomposition process is re-started at every level of the lattice. This process can in principle be performed more efficiently, i.e., so as to take $O(|\mathcal{P}| + |E| + |V| \times |L|)$ time, as it actually corresponds to (a simple variant of) a single-layer core decomposition.

To address the two above cons, we propose a method performing a depth-first search on the core lattice. The method, dubbed dfs-ml-cores (Algorithm 3), iteratively picks a non-leaf core $\mathbf{k} = [k_1, \ldots, k_\ell, \ldots, k_{|L|}]$ and a layer ℓ such that $k_\ell = 0$, and computes all cores $[k_1, \ldots, k_\ell + 1, \ldots, k_{|L|}], \ldots, [k_1, \ldots, K_\ell, \ldots, k_{|L|}]$ with a run of the CoreDecomposition$(G, C_\mathbf{k}, \mathbf{k}, \ell)$ subroutine.[4]

A side effect of this strategy is that the same core may be computed multiple times. As an example, in Figure 2 the $(1, 2, 0)$-core is computed by core decompositions initiated at both cores $(1, 0, 0)$ and $(0, 2, 0)$. To reduce (but not eliminate) these multiple core computations, the dfs-ml-cores method exploits the following result:

Theorem 3. *Given a multilayer graph $G = (V, E, L)$, let $[\ell_1, \ldots, \ell_{|L|}]$ be an order defined over set L. Let $\mathbf{Q}_0 = \{[0]_{|L|}\}$, and, $\forall i \in [1..|L|]$, let $\mathbf{Q}_i = \{\mathbf{k}' \in CoreDecomposition(G, C_\mathbf{k}, \mathbf{k}, \ell) \mid \mathbf{k} \in \mathbf{Q}_{i-1}, \ell \in (\ell_i..\ell_{|L|}], k_\ell = 0\}$ and $\mathbf{C}_i = \{\mathbf{k}' \in CoreDecomposition(G, C_\mathbf{k}, \mathbf{k}, \ell) \mid \mathbf{k} \in \mathbf{Q}_{i-1}, \ell \in [\ell_1..\ell_i], k_\ell = 0\}$. The set $\mathbf{C} = \{C_\mathbf{k} \mid \mathbf{k} \in \bigcup_{i=0}^{|L|} \mathbf{Q}_i \cup \bigcup_{i=1}^{|L|} \mathbf{C}_i\}$ is the multilayer core decomposition of G.*

[4] Specifically, the CoreDecomposition subroutine returns cores corresponding to all coreness vectors obtained by varying the ℓ-th component of $\mathbf{k}$ within $[0..K_\ell]$. In addition, it discards vertices violating the coreness condition specified by vector $\mathbf{k}$, i.e., vertices whose degree in some layer $\hat{\ell} \neq \ell$ is less than the $\hat{\ell}$-th component of $\mathbf{k}$.

Algorithm 3 DFS-ML-cores

```
Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathbf{C}$ of all non-empty multilayer cores of $G$.
$\mathbf{C} \leftarrow \{V\}$,  $R \leftarrow L$,  $\mathbf{Q} \leftarrow \{[0]_{|L|}\}$,  $\mathbf{Q}' \leftarrow \emptyset$
while $R \neq \emptyset$ do
    remove a layer from $R$
    for all $\mathbf{k} \in \mathbf{Q}$ do
        $\forall \ell \in R$ s.t. $k_\ell = 0 : \mathbf{Q}' \leftarrow \mathbf{Q}' \cup \mathsf{CoreDecomposition}(G, C_\mathbf{k}, \mathbf{k}, \ell)$
        $\forall \ell \in L \setminus R$ s.t. $k_\ell = 0 :$
        $\mathbf{C} \leftarrow \mathbf{C} \cup \{C_{\mathbf{k}'} \mid \mathbf{k}' \in \mathsf{CoreDecomposition}(G, C_\mathbf{k}, \mathbf{k}, \ell)\}$
    $\mathbf{C} \leftarrow \mathbf{C} \cup \{C_\mathbf{k} \mid \mathbf{k} \in \mathbf{Q}'\}$,  $\mathbf{Q} \leftarrow \mathbf{Q}'$,  $\mathbf{Q}' \leftarrow \emptyset$
```

Proof. The multilayer core decomposition of G is formed by the union of all non-empty and distinct cores of all paths $\mathcal{P}$ of the lattice connecting a non-leaf node to a leaf by varying the same ℓ-th component of the corresponding coreness vectors.

Since some of the paths overlap, all cores of the paths $\mathcal{P}_i$, whose coreness vectors $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ have i non-zero components, i.e., whose coreness vectors $\mathbf{k}'$ are in $\mathbf{Q}_i \cup \mathbf{C}_i = \{\mathbf{k}' : |\{k'_\ell : \ell \in L, k'_\ell > 0\}| = i\}$, are derived by executing single-layer core decompositions initiated at a subset of cores of the paths $\mathcal{P}_{i-1}$, whose coreness vectors $\mathbf{k} = [k_\ell]_{\ell \in L}$ have $i - 1$ non-zero components. Such a subset of cores is represented by the set of coreness vectors $\mathbf{Q}_{i-1} = \{\mathbf{k} : |\{k_\ell : \ell \in [\ell_2..\ell_{|L|}], k_\ell > 0\}| = i - 1\}$, i.e., the set of coreness vectors $\mathbf{k}$ whose number of non-zero components corresponding to layers within $[\ell_2..\ell_{|L|}]$ is equal to $i - 1$. In addition, single-layer core decompositions for the layers where $k_\ell \neq 0$ are avoided, since it is equivalent to visit cores in $\mathcal{P}_{i-1}$.

As a result, the set $\{C_\mathbf{k} \mid \mathbf{k} \in \bigcup_{i=0}^{|L|} \mathbf{Q}_i \cup \bigcup_{i=1}^{|L|} \mathbf{C}_i\}$ correctly contains all possible coreness vectors of the core lattice. □

Referring to the pseudocode in Algorithm 3, the result in Theorem 3 is implemented by keeping track of a subset of layers $R \subseteq L$. At the beginning $R = L$, and, at each iteration of the main cycle, a layer ℓ is removed from it. The algorithm is independent of the removal order. Set $\mathbf{Q}$ keeps track of (the coreness vector of) all lattice nodes where the current single-layer core-decomposition processes need to be run from. $\mathbf{Q}'$ stores the (coreness vector of) cores computed from each node in $\mathbf{Q}$ and for each layer within R, while also forming the basis of $\mathbf{Q}$ for the next iteration.

In summary, compared to BFS-ML-cores, the DFS method reduces both the time complexity of computing all cores in a path $\mathcal{P}$ from a non-leaf node to a leaf of the core lattice (from $O(|\mathcal{P}| \times (|E| + |V| \times |L|))$ to $O(|\mathcal{P}| + |E| + |V| \times |L|)$), and the number of times a core is *visited*, which may now be smaller than the number of its fathers. On the other hand, DFS-ML-cores comes with the aforementioned issue that some cores may be *computed* multiple times (while in BFS-ML-cores every core is computed only once). Furthermore, cores are computed starting from larger subgraphs, as intersection among multiple fathers can not exploited.

3.4 Hybrid algorithm

The ultimate output of both BFS-ML-cores and DFS-ML-cores correctly corresponds to all distinct cores of the input graph and the corresponding maximal coreness vectors.[5] Nevertheless, none of these methods is able to skip the computation of non-distinct cores. Indeed, both methods need to compute every core C as many times as the number of its coreness vectors in order to guarantee completeness. To address this limitation we devise a further method where the main peculiarities of both BFS-ML-cores and DFS-ML-cores are joined into a "hybrid" lattice-visit strategy. This HYBRID-ML-cores method exploits the following corollary of Theorem 1, stating that the maximal coreness vector of a core C is given by the vector containing the minimum degree of a vertex in C for each layer:

Algorithm 4 HYBRID-ML-cores

```
Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathbf{C}$ of all non-empty multilayer cores of $G$.
$\mathbf{Q} \leftarrow \{[0]_{|L|}\}$,  $\mathcal{F}([0]_{|L|}) \leftarrow \emptyset$           {$\mathcal{F}$ keeps track of father nodes}
$\mathbf{Q}' \leftarrow \bigcup_{\ell \in L} \mathsf{CoreDecomposition}(G, V, [0]_{|L|}, \ell)$   {looked-ahead cores}
$\mathbf{C} \leftarrow \{C_\mathbf{k} \mid \mathbf{k} \in \mathbf{Q}'\}$
while $\mathbf{Q} \neq \emptyset$ do
    dequeue $\mathbf{k} = [k_\ell]_{\ell \in L}$ from $\mathbf{Q}$
    if $|\{k_\ell : k_\ell > 0\}| = |\mathcal{F}(\mathbf{k})| \wedge \mathbf{k} \notin \mathbf{Q}'$ then                {Corollary 2}
        $F_\cap \leftarrow \bigcap_{F \in \mathcal{F}(\mathbf{k})} F$                                      {Corollary 1}
        $C_\mathbf{k} \leftarrow \mathbf{k}\text{-core}(G, F_\cap, \mathbf{k})$                                 {Algorithm 1}
        if $C_\mathbf{k} \neq \emptyset$ then
            $\mathbf{C} \leftarrow \mathbf{C} \cup \{C_\mathbf{k}\}$
            $\mathbf{d}_\mu(C_\mathbf{k}) \leftarrow [\mu(C_\mathbf{k}, \ell)]_{\ell \in L}$  {look-ahead mechanism (Cor. 3)}
            $\mathbf{Q}' \leftarrow \mathbf{Q}' \cup \{\mathbf{k}' \mid \mathbf{k} \leq \mathbf{k}' \leq \mathbf{d}_\mu(C_\mathbf{k})\}$
    if $\mathbf{k} \in \mathbf{Q}'$ then
        for all $\ell \in L$ do                                {enqueue child nodes}
            $\mathbf{k}' \leftarrow [k_1, \ldots, k_\ell + 1, \ldots, k_{|L|}]$
            enqueue $\mathbf{k}$' into $\mathbf{Q}$
            $\mathcal{F}(\mathbf{k}') \leftarrow \mathcal{F}(\mathbf{k}') \cup \{C_\mathbf{k}\}$
```

COROLLARY 3. *Given a multilayer graph $G = (V, E, L)$, the maximal coreness vector of a multilayer core C of G corresponds to the $|L|$-dimensional integer vector $\mathbf{d}_\mu(C) = [\mu(C, \ell)]_{\ell \in L}$.*

Proof. By Definition 2, vector $\mathbf{d}_\mu(C)$ is a coreness vector of C. Assume that $\mathbf{d}_\mu(C)$ is not maximal, meaning that another coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ dominating $\mathbf{d}_\mu(C)$ exists. This implies that $k_\ell \geq \mu(C, \ell)$, and $\exists \hat{\ell} \in L : k_{\hat{\ell}} > \mu(C, \hat{\ell})$. By definition of multilayer core, all vertices in C have degree larger than the minimum degree $\mu(C, \hat{\ell})$ in layer $\hat{\ell}$, which is a clear contradiction. □

Corollary 3 gives a rule to skip the computation of non-distinct cores: given a core C with coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, all cores with coreness vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ such that $\forall \ell \in L : k_\ell \leq k'_\ell \leq \mu(C, \ell)$ are guaranteed to be equal to C and do not need to be explicitly computed. For instance, in Figure 2, assume that the min-degree vector of the $(0, 0, 1)$-core is $(0, 1, 2)$. Then, cores $(0, 0, 2)$, $(0, 1, 1)$, and $(0, 1, 2)$ can immediately be set equal to the $(0, 0, 1)$-core. The HYBRID-ML-cores algorithm we present here (Algorithm 4) exploits this rule by performing a breadth-first search equipped with a "look-ahead" mechanism resembling a depth-first search. Moreover, HYBRID-ML-cores starts with a single-layer core decomposition for each layer so as to have more fathers early-on for intersections. Cores interested by the look-ahead rule are still *visited* and stored in $\mathbf{Q}'$, as they may be needed for future core computations. However, no further computational overhead is required for them.

3.5 Discussion

We already discussed (in the respective paragraphs) the strengths and weaknesses of BFS-ML-cores and DFS-ML-cores: the best among the two is determined by the peculiarities of the specific input graph. On the other hand, HYBRID-ML-cores profitably exploits the

[5] Pseudocodes in Algorithms 2 and 3 guarantee this as cores are added to a set $\mathbf{C}$ that does not allow duplicates. Any real implementation can easily take care of this by checking whether a core is already in C, and update it in case the corresponding coreness vector contains the previously-stored one.

main nice features of both bfs-ml-cores and dfs-ml-cores, thus is expected to outperform both methods in most cases. However, in those graphs where the number of non-distinct cores is limited, the overhead due to the look-ahead mechanism can make the performance of hybrid-ml-cores degrade.

In terms of space requirements, bfs-ml-cores needs to keep in memory all those cores having at least a child in the queue, i.e., at most two levels of the lattice (the space taken by a multilayer core is $O(|V|)$). The same applies to hybrid-ml-cores with the addition of the cores computed through single-layer core decomposition and look-ahead, until all their children have been processed. dfs-ml-cores instead requires to store all cores where the single-layer core-decomposition process should be started from, both in the current iteration and the next one. Thus, we expect dfs-ml-cores to take more space than bfs-ml-cores and hybrid-ml-cores, as in practice the number of cores to be stored should be more than the cores belonging to two lattice levels.

4 EXPERIMENTS

In this section we present experiments to (i) compare the proposed algorithms in terms of runtime, memory consumption, and search-space exploration; (ii) characterize the output core decompositions.

Datasets. We select publicly-available real-world multilayer networks, whose main characteristics are summarized in Table 1.

Homo[6] and SacchCere[6] are networks describing different types of genetic interactions between genes in Homo Sapiens and Saccharomyces Cerevisiae, respectively. ObamaInIsrael[6] represents different types of social interaction (e.g., *re-tweeting*, *mentioning*, and *replying*) among Twitter users, focusing on Barack Obama's visit to Israel in 2013. Similarly, Higgs[6] is built by tracking the spread of news about the discovery of the Higgs boson on Twitter, with the additional layer for the *following* relation. Friendfeed[7] contains public interactions among users of Friendfeed collected over two months (e.g., *commenting*, *liking*, and *following*). FriendfeedTwitter[7] is a multi-platform social network, where layers represent interactions within Friendfeed and Twitter between users registered to both platforms [17]. Amazon[8] is a co-purchasing *temporal network*, containing four snapshots between March and June 2003. Finally, DBLP[9] is derived following the methodology in [12]. For each co-authorship relation (edge), the bag of words resulting from the titles of all papers co-authored by the two authors is collected. Then *LDA* topic modeling [10] is applied to automatically identify a hundred topics. Among these, ten topics that are recognized as the most relevant to the data-mining area have been hand-picked. Every selected topic corresponds to a layer. An edge between two co-authors in a certain layer exists if the relation between those co-authors is labeled with the topic corresponding to that layer.

Implementation. All methods are implemented in Python (v. 2.7.12) and compiled by Cython. The experiments run on a machine equipped with Intel Xeon CPU at 2.7GHz and 128GB RAM.[10]

Table 1: Characteristics of the real-world datasets: number of vertices ($|V|$), number of edges ($|E|$), number of layers ($|L|$).

dataset	$\|V\|$	$\|E\|$	$\|L\|$	domain
Homo	18k	153k	7	genetic
SacchCere	6.5k	247k	7	genetic
DBLP	513k	1.0M	10	co-authorship
ObamaInIsrael	2.2M	3.8M	3	social
Amazon	410k	8.1M	4	co-purchasing
FriendfeedTwitter	155k	13M	2	social
Higgs	456k	13M	4	social
Friendfeed	510k	18M	3	social

Comparative evaluation. We compare the naïve baseline (for short n) and the three proposed methods bfs-ml-cores (for short bfs), dfs-ml-cores (dfs), hybrid-ml-cores (h) in terms of running time, memory usage, and number of computed cores (as a measure of the explored search-space portion). The results of this comparison are shown in Table 2. As expected, n is the least efficient method. Due to its excessive requirements, we could run it in reasonable time (i.e., one week) only on the smaller datasets, where it is anyway outperformed by our methods by 1–4 orders of magnitude. Among the proposed methods, h achieves the best performance in most datasets, as expected. In some cases, however, h is comparable to bfs, thus confirming the fact that in datasets where the number of non-distinct cores is not so large the performance of the two methods gets closer. A similar reasoning holds between bfs and dfs (at least with a small/moderate number of the layers, see next): bfs is faster in most cases, but, due to the respective pros and cons discussed in Section 2, it is not surprising that the two methods achieve comparable performance in a number of other cases.

To test the behavior with varying the number of layers, Figure 3 shows the running times of the proposed methods on different versions of the DBLP dataset, obtained by selecting a variable number of layers, from 2 to 10. While the performance of the three methods is comparable up to six layers, beyond this threshold the execution time of dfs grows much faster than bfs and h. This attests that the pruning rules of bfs and h are more effective as the layers increase. To summarize, dfs is expected to have runtime comparable to (or better than) bfs and h when the number of layers is small, while h is faster than bfs when the number of non-distinct cores is large.

The number of computed cores is always larger than the output cores as all methods might compute empty cores or, in the case of dfs, the same core multiple times. Table 2 shows that dfs computes more cores than bfs and h, which conforms to its design principles.

Finally, all methods turn out to be memory-efficient, taking no more than 1.5GB of memory.

Core-decomposition characterization. In Figure 4 (and Table 2) we show the number of all output cores and inner-most cores extracted from each selected dataset. The number of cores differs quite a lot from dataset to dataset, depending on dataset size, number of layers, and density. The fraction of inner-most cores exhibits a non-decreasing trend as the layers increase, ranging from 0.3% of the total number of output cores (FriendfeedTwitter) to 22% (DBLP).

Figure 5 reports the distribution of number of cores, core size, and average-degree density (i.e., number of edges divided by number of vertices) of the subgraph corresponding to a core. Distributions are shown by level of the lattice[11] for the SacchCere and Friendfeed

[6] http://deim.urv.cat/~manlio.dedomenico/data.php
[7] http://multilayer.it.uu.se/datasets.html
[8] https://snap.stanford.edu/data/
[9] http://dblp.uni-trier.de/xml/
[10] All code and datasets are available at https://goo.gl/8741Gs.

[11] Recall that the lattice level has been defined in Section 3.1: level i contains all cores whose coreness-vector components sum to i.

Table 2: Comparative evaluation: proposed methods and baseline.

dataset	#output cores	method	time (s)	memory (MB)	#computed cores
Homo	1 845	N	1 024	27	12 112
		BFS	10	26	3 043
		DFS	20	27	6 937
		H	9	25	2 364
SacchCere	74 426	N	19 282	55	278 402
		BFS	802	34	89 883
		DFS	2 117	57	223 643
		H	819	35	83 978
DBLP	3 346	N	104 361	608	34 572
		BFS	66	612	6 184
		DFS	219	627	38 887
		H	26	521	5 037
Obama InIsrael	2 573	N	28 936	1 286	3 882
		BFS	184	1 299	3 313
		DFS	121	1 384	3 596
		H	134	1 147	2 716
Amazon	1 164	BFS	2 349	534	1 354
		DFS	3 809	619	2 459
		H	2 464	536	1 334
Friendfeed Twitter	76 194	BFS	47 735	215	80 664
		DFS	1 523	267	80 745
		H	41 599	268	76 419
Higgs	8 077	BFS	1 775	465	12 773
		DFS	433	490	14 119
		H	1 544	493	9 389
Friendfeed	365 666	BFS	45 568	465	546 631
		DFS	12 211	591	568 107
		H	37 495	490	389 323

datasets. Although the two datasets have very different scales, the distributions exhibit similar trends. Being limited by the number of layers, the number of cores in the first levels of the lattice is very small, but then it exponentially grows until reaching its maximum within the first 25 − 30% visited levels. The average size of the cores is close to the number of vertices in the first lattice level, when cores' degree conditions are not very strict. Then it decreases as the number of cores gets larger, with a maximum reached when very small cores stop "propagating" in the lower lattice levels. Finally, the average (average-degree) density tends to increase for higher lattice level. However, there are a couple of exceptions: it decreases (*i*) in the first few levels of SacchCere's lattice, and (*ii*) in the last levels of both SacchCere and Friendfeed, where the core size starts getting smaller, thus implying small average-degree values.

5 MULTILAYER DENSEST SUBGRAPH

In this section we showcase the usefulness of the multilayer core-decomposition tool introduced above. Particularly, we show how to exploit it to devise an algorithm with approximation guarantees for multilayer densest subgraph, thus extending to the multilayer case the intuition at the basis of the well-known $\frac{1}{2}$-approximation algorithm [3, 15] for single-layer densest-subgraph extraction.

Problem definition. As discussed in Section 1.2, the densest subgraph of a multilayer graph should provide a good trade-off between large density and the number of layers where such a large density is exhibited. We achieve this with the following optimization problem:

PROBLEM 2 (MULTILAYER DENSEST SUBGRAPH). *Given a multilayer graph $G = (V, E, L)$, a positive real number β, and a real-valued function $\delta : 2^V \rightarrow \mathbb{R}^+$ defined as:*

$$\delta(S) = \max_{\hat{L} \subseteq L} \min_{\ell \in \hat{L}} \frac{|E_\ell[S]|}{|S|} |\hat{L}|^\beta, \tag{1}$$

find a subset $S^* \subseteq V$ of vertices that maximizes function δ, i.e.,

$$S^* = \arg\max_{S \subseteq V} \delta(S).$$

The role of parameter β in Problem 2 is to control the importance of the two ingredients of the objective function δ, i.e., density and number of layers exhibiting such a density: the smaller β the larger the importance to be given to the former aspect (density), and vice versa. Also, as a nice side effect, solving the MULTILAYER DENSEST SUBGRAPH problem allows for automatically finding a set of layers of interest for the densest subgraph S^*.

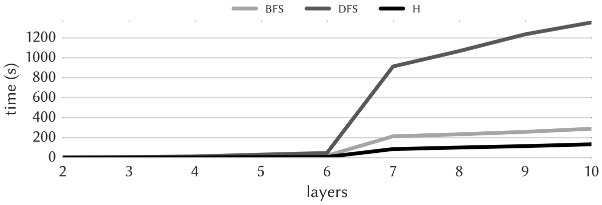

Figure 3: Runtime of the proposed methods with varying the number of layers (DBLP dataset).

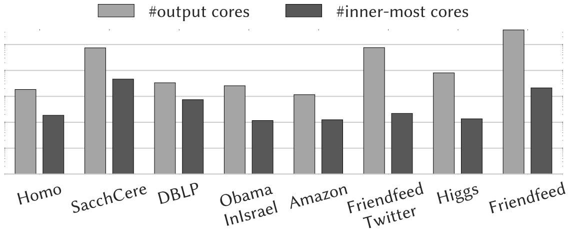

Figure 4: Number of output cores (total and inner-most).

Algorithms. It is easy to see that for $|L| = 1$ Problem 2 corresponds to the traditional densest-subgraph problem in single-layer graphs [22]. Such a problem is solvable in polynomial time, but the time complexity of exact algorithms is $\Omega(|V| \times |E|)$ [22], thus unaffordable for large graphs. Hence, we cannot hope efficient exact solutions to exist for MULTILAYER DENSEST SUBGRAPH either. We then follow the intuition coming from the single-layer setting, i.e., that core decomposition can provide good quality approximations.

Specifically, we devise an approximation algorithm based on the multilayer core decomposition discussed above. The algorithm is very simple: it computes the multilayer core decomposition of the input graph, and, among all cores, takes the one maximizing the objective function δ as the output densest subgraph (Algorithm 5). Despite its simplicity, the algorithm achieves provable approximation guarantees proportional to the number of layers of the input graph, precisely equal to $\frac{1}{2|L|^\beta}$. We next formally prove this result.

Algorithm 5 ML-densest

```
Input: A multilayer graph $G = (V, E, L)$ and a real number $\beta \in \mathbb{R}^+$.
Output: The densest subgraph $S^*$ of $G$.
$\mathbf{C} \leftarrow$ MultiLayerCoreDecomposition($G$)
$S^* \leftarrow \arg\max_{C \in \mathbf{C}} \delta(C)$        {Equation (1)}
```

Approximation guarantees. Let $\mathbf{C}$ be the core decomposition of the input multilayer graph $G = (V, E, L)$ and C^* denote the core in $\mathbf{C}$ maximizing the density function δ, i.e., $C^* = \arg\max_{C \in \mathbf{C}} \delta(C)$. Then, C^* corresponds to the subgraph output by the proposed ML-densest algorithm. Let also $C^{(\mu)}$ denote the subgraph maximizing the minimum degree in a single layer, i.e., $C^{(\mu)} = \arg\max_{S \subseteq V} f(S)$, where $f(S) = \max_{\ell \in L} \mu(S, \ell)$, while $\ell^{(\mu)} = \arg\max_{\ell \in L} \mu(C^{(\mu)}, \ell)$. It is easy to see that $C^{(\mu)} \in \mathbf{C}$. Finally, let S^*_{SL} be the densest subgraph among all single-layer densest subgraphs, i.e., $S^*_{\mathrm{SL}} = \arg\max_{S \subseteq V} g(S)$, where $g(S) = \max_{\ell \in L} \frac{|E_\ell[S]|}{|S|}$,

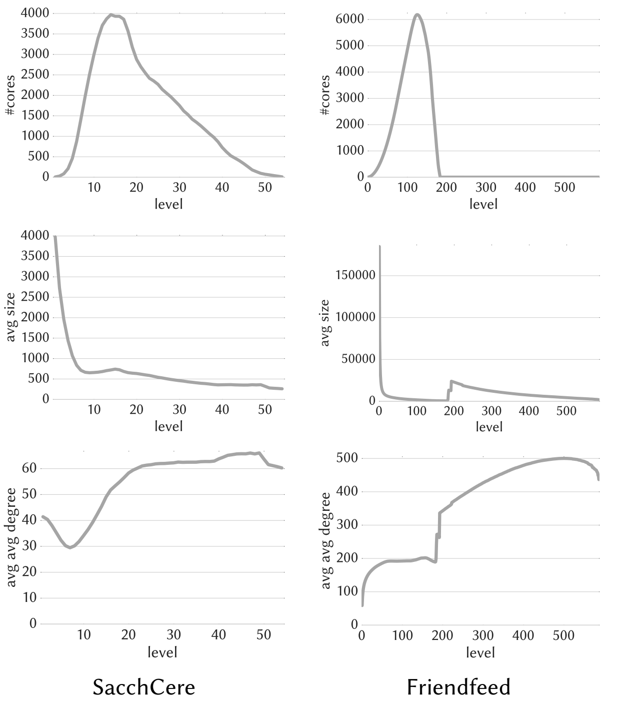

Figure 5: SacchCere and Friendfeed datasets: distribution of number of cores (top), average core size (middle), and average average-degree density of a core (bottom) to the core-lattice level.

and ℓ^* be the layer where S^*_{SL} exhibits its largest density, i.e., $\ell^* = \arg\max_{\ell \in L} \frac{|E_\ell[S^*_{\mathrm{SL}}]|}{|S^*_{\mathrm{SL}}|}$. We start by introducing the following two lemmas that can straightforwardly be derived from the definitions of C^*, $C^{(\mu)}$, S^*_{SL}, $\ell^{(\mu)}$, and ℓ^*:

Lemma 1. $\delta(C^*) \geq \delta(C^{(\mu)})$.

Proof. By definition, $C^{(\mu)}$ is a multilayer core described by (among others) the coreness vector $\mathbf{k} = [k_\ell]_{\ell \in L}$ with $k_{\ell^{(\mu)}} = \max_{\ell \in L} \mu(C^{(\mu)}, \ell)$, and $k_\ell = 0$, $\forall \ell \neq \ell^{(\mu)}$. Then $C^{(\mu)} \in \mathbf{C}$. As $C^* = \arg\max_{C \in \mathbf{C}} \delta(C)$, it holds that $\delta(C^*) \geq \delta(C^{(\mu)})$. □

Lemma 2. $\delta(S^*) \leq \frac{|E_{\ell^*}[S^*_{SL}]|}{|S^*_{SL}|}|L|^\beta$.

Proof.

$$\delta(S^*) = \max_{\hat{L} \subseteq L} \min_{\ell \in \hat{L}} \frac{|E_\ell[S^*]|}{|S^*|}|\hat{L}|^\beta \leq \max_{\ell \in L} \frac{|E_\ell[S^*]|}{|S^*|}|L|^\beta \leq \frac{|E_{\ell^*}[S^*_{\mathrm{SL}}]|}{|S^*_{\mathrm{SL}}|}|L|^\beta. \ \square$$

The following further lemma shows a lower bound on the minimum degree of a vertex in S^*_{SL}:

Lemma 3. $\mu(S^*_{SL}, \ell^*) \geq \frac{|E_{\ell^*}[S^*_{SL}]|}{|S^*_{SL}|}$.

Proof. As S^*_{SL} is the subgraph maximizing the density in layer ℓ^*, removing the minimum-degree node from S^*_{SL} cannot increase that density. Thus, it holds that:

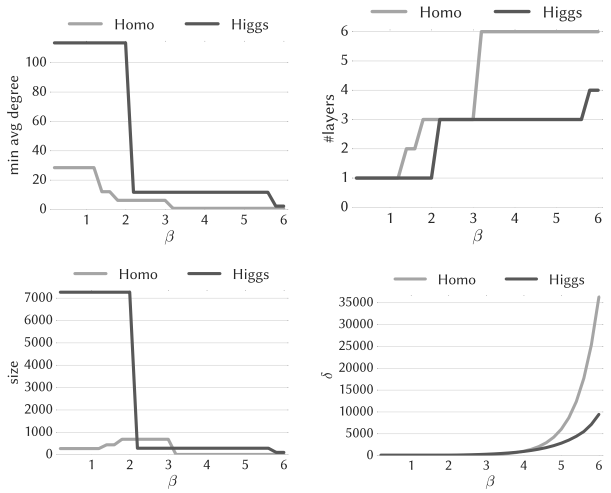

Figure 6: Multilayer densest-subgraph extraction (Homo and Higgs datasets): minimum average-degree density in a layer, number of selected layers, size, and objective-function value δ of the output densest subgraphs with varying β.

$$\frac{|E_{\ell^*}[S^*_{\mathrm{SL}}]|}{|S^*|} \geq \frac{|E_{\ell^*}[S^*_{\mathrm{SL}}]| - \mu(S^*_{\mathrm{SL}}, \ell^*)}{|S^*_{\mathrm{SL}}| - 1}$$

$$\Leftrightarrow \quad \mu(S^*_{\mathrm{SL}}, \ell^*) \geq |E_{\ell^*}[S^*_{\mathrm{SL}}]| \frac{|S^*_{\mathrm{SL}}| - 1}{|S^*_{\mathrm{SL}}|} - |E_{\ell^*}[S^*_{\mathrm{SL}}]|$$

$$\Leftrightarrow \quad \mu(S^*_{\mathrm{SL}}, \ell^*) \geq \frac{|E_{\ell^*}[S^*_{\mathrm{SL}}]|}{|S^*_{\mathrm{SL}}|}. \qquad \square$$

The approximation factor of the proposed ML-densest algorithm is ultimately stated in the next theorem:

Theorem 4. $\delta(C^*) \geq \frac{1}{2|L|^\beta}\delta(S^*)$.

Proof.

$$\begin{aligned}
\delta(C^*) &\geq \delta(C^{(\mu)}) && \{\text{Lemma 1}\} \\
&\geq \max_{\ell \in L} \frac{|E_\ell[C^{(\mu)}]|}{|C^{(\mu)}|}1^\beta = \max_{\ell \in L} \frac{|E_\ell[C^{(\mu)}]|}{|C^{(\mu)}|} && \{\text{Equation (1)}\} \\
&\geq \frac{1}{2} \max_{\ell \in L} \mu(C^{(\mu)}, \ell) && \{\text{as avg degree} \geq \text{min degree}\} \\
&= \frac{1}{2}\mu(C^{(\mu)}, \ell^{(\mu)}) && \{\text{by definition of } C^{(\mu)}\} \\
&\geq \frac{1}{2}\mu(S^*_{\mathrm{SL}}, \ell^*) && \{\text{optimality of } C^{(\mu)} \text{ w.r.t. min degree}\} \\
&\geq \frac{1}{2}\frac{|E_{\ell^*}[S^*_{\mathrm{SL}}]|}{|S^*_{\mathrm{SL}}|} && \{\text{Lemma 3}\} \\
&\geq \frac{1}{2|L|^\beta}\delta(S^*). && \{\text{Lemma 2}\}
\end{aligned}$$

□

When $\beta = 0$ and $\hat{L} = L$ (instead of being any subset of L) Problem 2 corresponds to the densest-common-subgraph problem formulated by Jethava *et al.* [24], which asks for a subgraph maximizing the minimum average-degree density over all layers. It is straightforward to see that Theorem 4 carries over to this case and thus the proposed ML-densest algorithm achieves a $\frac{1}{2}$-approximation guarantee for the Jethava *et al.*'s formulation.

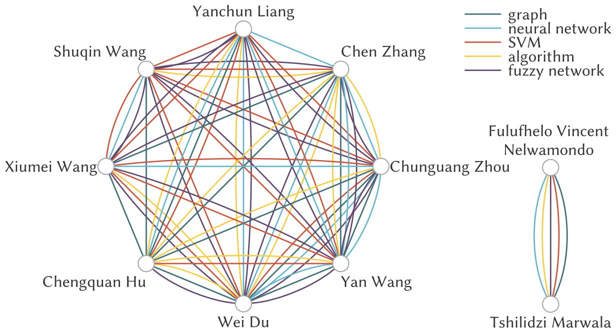

Figure 7: Multilayer densest subgraph extracted by Algorithm 5 from the DBLP dataset ($\beta = 2.2$).

5.1 Multilayer densest subgraph: experiments

We experimentally evaluate our ML-densest algorithm (Algorithm 5) on the datasets in Table 1. Figure 6 reports the results – minimum average-degree density in a layer, number of selected layers, size, objective-function value δ – on the Homo and Higgs datasets, with varying β. The remaining datasets, which we omit due to space constraints, exhibit similar trends on all measures.

The trends observed in the figure conform to what expected: the smaller β, the more the objective function privileges solutions with large average-degree density in a few layers (or even just one layer, for β close to zero). The situation is overturned with larger values of β, where the minimum average-degree density drops significantly, while the number of selected layers stands at 6 for Homo and 4 for Higgs. In-between β values lead to a balancing of the two terms of the objective function, thus giving more interesting solutions. Also, by definition, δ as a function of β draws exponential curves.

Finally, as anecdotal evidence of the output of Algorithm 5, in Figure 7 we report the densest subgraph extracted from DBLP. The subgraph contains 10 vertices and 5 layers automatically selected by the objective function δ. The minimum average-degree density is encountered on the layers corresponding to topics "*graph*" and "*algorithm*" (green and yellow layers in the figure), and is equal to 1.2. The objective-function value is $\delta = 41.39$. Note that the subgraph is composed of two connected components. In fact, like the single-layer case, multilayer cores are not necessarily connected.

6 CONCLUSIONS

The contribution of this work is twofold: (1) develop efficient algorithms for the intrinsically exponential problem of core decomposition of a multilayer graph; (2) study densest-subgraph extraction in multilayer graphs as a proper optimization problem trading off between high density and layers exhibiting high density, and show how core decomposition can be used to approximate this problem with quality guarantees.

REFERENCES

[1] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *NIPS*, 2005.
[2] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, 2009.
[3] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2), 2000.
[4] N. Azimi-Tafreshi, J. Gómez-Gardenes, and S. Dorogovtsev. k-core percolation on multiplex networks. *Physical Review E*, 90(3), 2014.
[5] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5), 2012.
[6] O. D. Balalau, F. Bonchi, T. H. Chan, F. Gullo, and M. Sozio. Finding subgraphs with maximum total density and limited overlap. In *WSDM*, 2015.
[7] V. Batagelj and M. Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *ADAC*, 5(2), 2011.
[8] M. Berlingerio, M. Coscia, and F. Giannotti. Finding redundant and complementary communities in multidimensional networks. In *CIKM*, 2011.
[9] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*, 2015.
[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3, 2003.
[11] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *KDD*, 2012.
[12] F. Bonchi, A. Gionis, F. Gullo, C. E. Tsourakakis, and A. Ukkonen. Chromatic correlation clustering. *TKDD*, 9(4), 2015.
[13] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *KDD*, 2014.
[14] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *PKDD*. 2005.
[15] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.
[16] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu. Efficient core decomposition in massive networks. In *ICDE*, 2011.
[17] M. E. Dickison, M. Magnani, and L. Rossi. *Multilayer Social Networks*. Cambridge University Press, 2016.
[18] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *ISAAC*, 2010.
[19] E. Galbrun, A. Gionis, and N. Tatti. Top-k overlapping densest subgraphs. *DAMI*, 30(5), 2016.
[20] A. Garas, F. Schweitzer, and S. Havlin. A k -shell decomposition method for weighted networks. *New Journal of Physics*, 14(8), 2012.
[21] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. D-cores: measuring collaboration of directed graphs based on degeneracy. *KAIS*, 35(2), 2013.
[22] A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, 1984.
[23] J. Healy, J. Janssen, E. E. Milios, and W. Aiello. Characterization of graphs using degree cores. In *WAW*, 2006.
[24] V. Jethava and N. Beerenwinkel. Finding dense subgraphs in relational graphs. In *ECML-PKDD*, pages 641–654, 2015.
[25] D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *TKDD*, 2(4):16, 2009.
[26] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse. Identifying influential spreaders in complex networks. *Nature Physics 6, 888*, 2010.
[27] G. Kortsarz and D. Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2), 1994.
[28] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. 2010.
[29] A. Montresor, F. D. Pellegrini, and D. Miorandi. Distributed k-core decomposition. *TPDS*, 24(2), 2013.
[30] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980), 2010.
[31] E. E. Papalexakis, L. Akoglu, and D. Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *FUSION*, 2013.
[32] K. Pechlivanidou, D. Katsaros, and L. Tassiulas. Mapreduce-based distributed k-shell decomposition for online social networks. In *SERVICES*, 2014.
[33] A. E. Sariyüce, B. Gedik, G. Jacques-Silva, K. Wu, and Ü. V. Çatalyürek. Streaming algorithms for k-core decomposition. *PVLDB*, 6(6), 2013.
[34] C.-Y. Shen, H.-H. Shuai, D.-N. Yang, Y.-F. Lan, W.-C. Lee, P. S. Yu, and M.-S. Chen. Forming online support groups for internet and behavior related addictions. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 163–172. ACM, 2015.
[35] L. Tang, X. Wang, and H. Liu. Community detection in multi-dimensional networks. Technical report, DTIC Document, 2010.
[36] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *KDD*, 2013.
[37] C. E. Tsourakakis. The k-clique densest subgraph problem. In *WWW*, 2015.
[38] N. Wang, J. Zhang, K.-L. Tan, and A. K. H. Tung. On triangulation-based dense neighborhood graph discovery. *PVLDB*, 4(2), 2010.
[39] Y. Wu, R. Jin, X. Zhu, and X. Zhang. Finding dense and connected subgraphs in dual networks. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 915–926. IEEE, 2015.
[40] S. Wuchty and E. Almaas. Peeling the yeast protein network. *Proteomics*, 5(2), 2005.
[41] X. Yan, X. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD*, pages 324–333, 2005.
[42] Z. M. Yin and S. S. Khaing. Multi-layered graph clustering in finding the community cores. *Int. J. Adv. Res. Comput. Eng. Technol*, 2, 2013.
[43] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin. When engagement meets similarity: efficient (k, r)-core computation on social networks. *Proceedings of the VLDB Endowment*, 10(10):998–1009, 2017.