Locally Densest Subgraph Discovery

Lu Qin[†], Rong-Hua Li[‡], Lijun Chang[§], Chengqi Zhang[†]

[†]Centre for Quantum Computation & Intelligent Systems, University of Technology, Sydney, Australia
[‡]Shenzhen University, China [§]The University of New South Wales, Australia

[†]{lu.qin,chengqi.zhang}@uts.edu.au [‡]rhli@szu.edu.cn [§]ljchang@cse.unsw.edu.au

ABSTRACT

Mining dense subgraphs from a large graph is a fundamental graph mining task and can be widely applied in a variety of application domains such as network science, biology, graph database, web mining, graph compression, and micro-blogging systems. Here a dense subgraph is defined as a subgraph with high density (#.edge / #.node). Existing studies of this problem either focus on finding the densest subgraph or identifying an optimal clique-like dense subgraph, and they adopt a simple greedy approach to find the top-k dense subgraphs. However, their identified subgraphs cannot be used to represent the dense regions of the graph. Intuitively, to represent a dense region, the subgraph identified should be the subgraph with highest density in its local region in the graph. However, it is non-trivial to formally model a locally densest subgraph. In this paper, we aim to discover top-k such representative locally densest subgraphs of a graph. We provide an elegant parameter-free definition of a locally densest subgraph. The definition not only fits well with the intuition, but is also associated with several nice structural properties. We show that the set of locally densest subgraphs in a graph can be computed in polynomial time. We further propose three novel pruning strategies to largely reduce the search space of the algorithm. In our experiments, we use several real datasets with various graph properties to evaluate the effectiveness of our model using four quality measures and a case study. We also test our algorithms on several real web-scale graphs, one of which contains 118.14 million nodes and 1.02 billion edges, to demonstrate the high efficiency of the proposed algorithms.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms; H.2.8 [**Database Applications**]: Data Mining

Keywords

Graph; Dense Subgraph; Big Data

1. INTRODUCTION

Mining dense subgraphs from a large graph is a fundamental graph mining task which has been widely used in a variety of application domains [28]. For example, in the network science domain, dense subgraphs represent cohesive groups or communities in a network. There are several community detection algorithms that are based on dense subgraphs [20, 14]. In the biology domain, the dense subgraph mining problem has been leveraged to identify regulatory motifs in genomic DNA [21] and to find complex patterns in a gene annotation graph [31]. In the graph database domain, algorithms for dense subgraphs discovery play an important role for creating elegant index structures to process reachability and distance queries efficiently [17, 26]. In the web mining domain, dense subgraph mining techniques are applied to link spam detection based on an interesting observation that dense subgraphs typically correspond to link spam farms [23]. In addition, dense subgraph mining has also been used for graph compression [11] and identifying stories in micro-blogging systems [4].

KDD '15, August 10-13, 2015, Sydney, NSW, Australia.
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08…$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783299

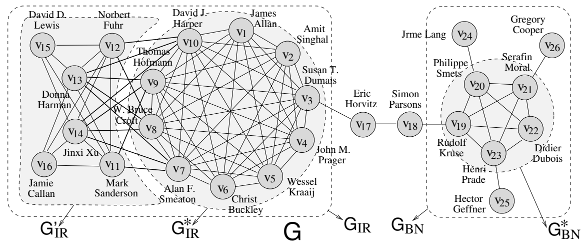


Figure 1: Part of the Coauthor Network

Motivation. The dense subgraph mining problem aims at identifying the subgraphs with high density (i.e., #.edge / #.node) [25, 6, 13, 7] from a large graph. Existing studies of this problem either focus on finding the densest subgraph (the subgraph with the highest density) [25, 6] or identifying an optimal clique-like dense subgraph (e.g., optimal quasi-clique proposed in [37]). To find top-k dense subgraphs, a simple greedy procedure, as suggested in [37], is used, which iteratively invokes the same algorithm k times in the residual graph after deleting the identified dense subgraphs in the previous iterations. The major drawback of these methods is that their results cannot be used to represent the dense regions of the graph. If the graph contains a large dense region, the top-k dense subgraphs identified by the above approaches may all belong to the same dense region, and other local dense regions may be neglected. For instance, Fig. 1 shows part of the collaboration network in the Coauthor dataset (http://arnetminer.org/), which includes two subgraphs G_{IR} and G_{BN} in two research areas Information Retrieval (IR) and Bayesian Networks (BN) respectively. If we use the greedy procedure to find the top-2 dense subgraphs based on either the densest subgraph model [25] or the optimal quasi-clique model [37], the result will be G^*_{IR} and G'_{IR}. Intuitively, the two dense subgraphs cannot fully reflect the top-2 representative dense regions of the graph, because G'_{IR} is located in the same dense region as G^*_{IR}. Instead, the more intuitive top-2 representative dense subgraphs should be G^*_{IR} and G^*_{BN}, which represent the densest communities in the two research areas IR and BN respectively. Motivated by this, in this paper, we propose a novel dense subgraph model, called the locally densest subgraph (LDS) model, which identifies the representative dense subgraphs each of which represents a dense region of the graph.

Our Model. The intuition of our LDS model is that a representative dense subgraph should be a subgraph with the highest density in its local region in the graph. Nevertheless, to formally define an LDS is non-trivial. A naive definition is to ensure that an LDS is not contained in a denser subgraph. Consider the graph in Fig. 1. Based on the naive definition, subgraph G^*_{BN} is not an LDS since it is contained in a denser subgraph G. The reason to exclude G^*_{BN} by the naive definition is that although the graph G is denser than G^*_{BN}, it is not compact, since the two subgraphs G_{IR} and G_{BN} of G are very loosely connected. To address this problem, we introduce the concept of *ρ-compact graph, which is a connected graph such that removing any subset S of nodes from the graph will result in removing at least $\rho \times |S|$ edges in the graph.* Clearly, a ρ-compact graph has a density of at least ρ and each node in it has a degree of at least $\lceil \rho \rceil$. Based on the concept of a ρ-compact graph, *we define LDS as a maximal ρ-compact subgraph with density equaling ρ.* Here, by maximal, we mean that it is not contained in a larger ρ-compact subgraph. Based on such a definition, in Fig. 1, subgraph G^*_{BN} with density 2 is an LDS since it is a maximal 2-compact subgraph. Subgraph G_{IR} with density $35/8$ is not an LDS because it is not $35/8$-compact. Subgraph G'_{IR} with density $13/6$ is not an LDS because it is contained in a larger $13/6$-compact subgraph G_{IR}. Note that the LDS model does not rely on the parameter ρ, since it is determined by the density of the subgraph itself.

Advantages of our Model. To the best of our knowledge, there is no similar definition of LDS in the literature. The main advantages of the LDS model are four-fold: (1) The LDS model is parameter-free. (2) An LDS with density ρ does not contain a subgraph with density larger than ρ, and it is not contained in any ρ'-compact subgraph with $\rho' \geq \rho$. Therefore, it fits well with the locally densest property. (3) All the LDSes in a graph are pairwise disjoint. As a result, one can use our LDS model to identify all the non-overlaping dense regions of a graph. (4) All the LDSes in a graph can be exactly computed in polynomial time; and they have a lot of good properties, based on which very effective pruning techniques can be designed to enable the algorithm to handle web-scale graphs.

Contributions. The main contributions of this paper are as follows:

(1) An elegant representative dense subgraph model. We introduce a new model, LDS, to identify the representative dense subgraphs of a graph. LDS has several nice structural properties, including locally densest, compact/cohesive, and pairwise disjoint.

(2) A polynomial algorithm with highly effective pruning strategies. We show that the exact top-k LDSes of a graph can be computed in polynomial time. We also propose three non-trivial pruning techniques to largely speedup the algorithm. With these techniques, the algorithm is able to handle web-scale graphs.

(3) Extensive experiments. In our experiments, we evaluate the effectiveness of our LDS model using four quality measures on real graphs with different graph properties. We perform a case study using the Coauthor network to demonstrate that our model can indeed find the representative dense subgraphs of a graph. We also test our algorithms on real web-scale graphs, one of which contains 118.14 million nodes and 1.02 billion edges, to demonstrate the high efficiency of the proposed algorithms.

Algorithm 1 Densest(graph G)

```
1: $low \leftarrow 0$; $high \leftarrow |E(G)|$; $g \leftarrow \emptyset$;
2: while $high - low \geq 1/|V(G)|^2$ do
3:     $mid \leftarrow (high + low)/2$;
4:     $g' \leftarrow \mathsf{TryDensity}(G, mid)$;
5:     if $g' \neq \emptyset$ then { $g \leftarrow g'$; $low \leftarrow mid$;} else $high \leftarrow mid$;
6: return $g$;
7: Procedure TryDensity(graph $G$, density $\rho$)
8: $G' \leftarrow G$;
9: Assign a weight 1 for every edge in $G'$;
10: Add a source node $s$ and a sink node $t$ in $G'$;
11: Add edge $(s, v)$ in $G'$ with weight $|E(G)|$ for every $v \in V(G') \setminus \{s, t\}$ ;
12: Add edge $(v, t)$ in $G'$ with weight $|E(G)| + 2 \times \rho - d(v, G)$ for every
    $v \in V(G') \setminus \{s, t\}$ ;
13: Compute the minimum $s$-$t$ cut, denoted by $S, T$, in $G'$;
14: return $G[S \setminus \{s\}]$;
```

Outline. In the rest of the paper, Section 2 presents the preliminaries. Section 3 introduces our LDS model. Section 4 shows the algorithm to identify the LDSes and explores three novel pruning techniques to speedup the algorithm. Section 5 evaluates our model and algorithms using extensive experiments. Section 6 reviews the related work and Section 7 concludes the paper.

2. PRELIMINARIES

Let $G = (V(G), E(G))$ be an undirected graph with $n = |V(G)|$ nodes and $m = |E(G)|$ edges. For each node $u \in V(G)$, we denote the neighbor set of u in G by $N(u, G)$, i.e., $N(u, G) = \{v|(u, v) \in E(G)\}$. Denote by $d(u, G)$ the degree of node u in G. We refer to $g = (V(g), E(g))$ as an induced subgraph of G if and only if $V(g) \subseteq V(G)$ and $E(g)$ is the induced edge set, i.e., $E(g) = \{(u, v)|u, v \in V(g), (u, v) \in E(G)\}$. Conversely, we refer to G as a supergraph of g. We use $G[S]$ to denote the induced subgraph of G induced by node set $S \subseteq V(G)$. When the context is obvious, we shall call an induced subgraph a subgraph for brevity. We also use the lowercase letter g to denote a subgraph and use the capital letter G to denote a general graph or a supergraph. A connected component of G is a maximal connected subgraph of G. Following the classic graph density definition [25, 6, 13, 7], the density of a graph G, denoted by $\mathsf{density}(G)$, is defined as:

$$\mathsf{density}(G) = \frac{|E(G)|}{|V(G)|} \tag{1}$$

Densest Subgraph. Based on the definition of density, the densest subgraph problem is to find the subgraph g of G such that $\mathsf{density}(g)$ is maximized. Note that the densest subgraph of a graph may not be unique, because there may be several densest subgraphs with the same density. We refer to the *maximal densest subgraph* as the largest subgraph with maximum density. The *densest subgraph component* is a connected component of the maximal densest subgraph. It is well known that Goldberg's parametric flow algorithm [25] can find the maximal densest subgraph in polynomial time by invoking $O(\log n)$ max-flow computations. For completeness, we depict Goldberg's algorithm in Algorithm 1. Later, we will invoke a subroutine of Goldberg's algorithm in our algorithm.

The general idea of Goldberg's algorithm is as follows: the algorithm uses a binary search procedure to find the optimal density (lines 2-5). In each step of the procedure, the algorithm guesses a density ρ in a binary search manner, and tries to find a subgraph g with density larger than ρ (lines 7-14). Such a subgraph can be identified by computing the minimum s - t cut in a carefully constructed graph G' (lines 8-12). The binary search procedure can terminate in $O(\log n)$ iterations [25]. The detailed description and correctness analysis of Goldberg's algorithm can be found in [25]. Based on Goldberg's results in [25], the following two lemmas can be immediately obtained:

Lemma 2.1: *For any two subgraphs g and g′ of G, if* $\text{density}(g) \neq \text{density}(g')$*, then* $|\text{density}(g) - \text{density}(g')| > 1/|V(G)|^2$. □

Lemma 2.2: *The procedure* TryDensity *in Algorithm 1 with parameters G and* $\rho - 1/|V(G)|^2$*, i.e.,* $\text{TryDensity}(G, \rho - 1/|V(G)|^2)$*, maximizes* $|E(g)| - \rho|V(g)|$ *over all subgraphs g of G, and returns the largest subgraph g in G with maximum* $|E(g)| - \rho|V(g)|$. □

Core Number. Next, we introduce r-core and core number, which are used in our problem analysis as well as our pruning techniques.

Definition 2.1: **(r-core and core number [34])** An *r-core subgraph g* of graph G is a subgraph of G such that for any $v \in V(g)$, $d(v, g) \geq r$. The *r-core* of G is the maximal r-core subgraph of G. For any $v \in V(G)$, the *core number* of v, denoted by $\text{core}(v, G)$, is the largest r such that v is contained in the r-core of G. □

The r-core model is a well known cohesive subgraph model which is widely used in social network analysis and graph mining tasks [9, 34, 15, 29]. Note that the r-core of a graph G with largest r may not be the subgraph with the highest density.

3. LOCALLY DENSEST SUBGRAPH

Densest subgraph computation is widely used in many graph mining tasks (e.g., [21, 32, 17, 4, 7, 6]). However, in many applications such as community detection [20, 14], finding one dense subgraph is usually not sufficient. Instead, top-k subgraphs to represent different dense regions of the graph are required.

Greed is Not Good. To find top-k dense subgraphs, we can adopt a straightforward greedy approach suggested in [37], which finds the densest subgraph [25] (or the optimal quasi-clique [37]) at a time, removes it from the graph, and repeats this procedure for k times. Nevertheless, such a greedy approach has several drawbacks: (1) The top-k results may not fully reflect the top-k densest regions of a graph. If the graph contains a very large dense region, subgraphs in other dense regions may have low chance of appearing in the top-k results. (2) A subgraph returned by the greedy approach can be partial and subsumed by a better subgraph. This makes it hard to characterize each result. (3) Such a greedy approach does not provide a formal definition of a result. A formal definition is important for graph mining tasks, because without a formal definition, it is not clear how to analyze each result. For the subgraphs identified by the greedy approach, except the density information, it is very hard to find other structural properties of each subgraph.

Dense or Compact? In order to compute representative dense subgraphs and avoid an identified subgraph being subsumed by a better subgraph, each identified subgraph should be densest in its local region. A straightforward way to define such a locally densest subgraph is to ensure that each identified subgraph is not contained in a denser subgraph. However, such a definition is not good because the denser subgraph may not be compact. To demonstrate this, let us consider the graph in Fig. 1. Intuitively, subgraph G^*_{BN} with density 2 should be a locally densest subgraph. However, based on the above definition, it is excluded from the results because it is contained in a denser subgraph G with density $43/13$. The reason to exclude G^*_{BN} is that the denser subgraph G is not compact since the two subgraphs G_{IR} and G_{BN} of G are very loosely connected. To address this problem, below, we first define a ρ-compact subgraph, which is used to model a locally densest subgraph.

Definition 3.1: **(ρ-compact)** A graph G is ρ-compact if and only if G is connected, and removing any subset of nodes $S \subseteq V(G)$ will result in the removal of at least $\rho \times |S|$ edges in G, where ρ is a nonnegative real number. □

The intuition of Definiton 3.1 is that a graph is compact if any subset of nodes is highly connected to others in the graph. By Definiton 3.1, we can obtain that if a graph G is ρ-compact, then every node in G has degree at least $\lceil \rho \rceil$ and thus it is a $\lceil \rho \rceil$-core subgraph. Furthermore, a ρ-compact graph G has density at least ρ. For any $\rho' > \rho$, a ρ'-compact graph is also a ρ-compact graph.

Definition 3.2: **(Maximal ρ-compact Subgraph)**. A ρ-compact subgraph g of G is a maximal ρ-compact subgraph of G if and only if there does not exist a supergraph g' of g ($g' \neq g$) in G such that g' is ρ-compact. □

Definition of LDS. Based on Definiton 3.2, we can formally define a locally densest subgraph (LDS) as follows:

Definition 3.3: **(Locally Densest Subgraph)** A subgraph g of G is a locally densest subgraph (LDS) of G if and only if g is a maximal $\text{density}(g)$-compact subgraph in G. □

Definiton 3.3 is parameter-free. By Definiton 3.3, we can obtain that (1) an LDS itself is compact, and (2) an LDS is not contained in a better subgraph that is more compact than itself. Note that no graph g can be ρ-compact with $\rho > \text{density}(g)$. The following example illustrates the definition of LDS.

Example 3.1: Consider the graph shown in Fig. 1. Subgraph G_{IR} with density $35/8$ is 4-compact, and it is also a maximal 4-compact subgraph. However, G_{IR} is not an LDS because it is not a maximal $35/8$-compact subgraph. Subgraph G'_{IR} with density $13/6$ is a $13/6$-compact subgraph. However, G'_{IR} is not an LDS because it is contained in a better subgraph G_{IR} which is 4-compact (and thus is $13/6$-compact). G^*_{IR} with density 4.5 is an LDS because it is a maximal 4.5-compact subgraph, and G^*_{BN} with density 2 is also an LDS because it it is a maximal 2-compact subgraph. □

Structural Properties of LDS. LDS has many useful properties. Below, we show some structural properties of LDS. In the next section, we will introduce more properties of LDS based on which efficient algorithm and pruning techniques can be developed.

The following lemma shows that any subgraph of an LDS cannot be denser than the LDS itself, and any supergraph of an LDS cannot be more compact than the LDS itself. This result indicates that an LDS is indeed a *locally* densest subgraph.

Lemma 3.1: **(Locally Densest Property)** *For any subgraph g′ of an LDS g in G,* $\text{density}(g') \leq \text{density}(g)$*; For any supergraph g′ of an LDS g in G, g′ is not ρ-compact for any* $\rho \geq \text{density}(g)$. □

Proof Sketch: The latter can be directly obtained from Definiton 3.3. Now we prove the former. Suppose to the contrary that there exits a subgraph g' of g with $\text{density}(g') > \text{density}(g)$. If we remove node set $S = V(g) \setminus V(g')$ from g, the number of edges removed is $|E(g)| - |E(g')| = \text{density}(g) \times |V(g)| - \text{density}(g') \times |V(g')| < \text{density}(g) \times (|V(g)| - |V(g')|) = \text{density}(g) \times |S|$. This contradicts the condition that g is $\text{density}(g)$-compact. □

Actually, we can prove that any connected subgraph of G that satisfies the locally densest property in Lemma 3.1 is an LDS. Therefore, Lemma 3.1 can provide an alternative definition of an LDS. Next, we show that an LDS g is also a cohesive subgraph, because it is more cohesive than an r-core for any $r < \text{density}(g)$.

Lemma 3.2: **(Cohesive Property)** *An LDS g in graph G is a* $\lceil \text{density}(g) \rceil$*-core subgraph of G.* □

Proof Sketch: Since g is $\text{density}(g)$-compact, the minimum degree of the nodes in g is no less than $\lceil \text{density}(g) \rceil$. As a result, g is a $\lceil \text{density}(g) \rceil$-core subgraph. □

Another useful property of the LDS model is that all the LDSes in a graph G are pairwise disjoint. As a result, the number of LDSes of a graph G is bounded by and usually much smaller than $|V(G)|$.

Based on such a property, the LDS model can be used to identify all the non-overlapping dense regions of a graph.

Lemma 3.3: **(Disjoint Property)** *Suppose that g and g' are two LDSes in G, then we have $V(g) \bigcap V(g') = \emptyset$.* $\square$

Proof Sketch: Without loss of generality, we assume that $\mathsf{density}(g) \geq \mathsf{density}(g')$. According to Definiton 3.3 and Lemma 3.1, we have $g \not\subseteq g'$. Suppose to the contrary that $V(g) \bigcap V(g') \neq \emptyset$. Since g' is an LDS, g' is a maximal $\mathsf{density}(g')$-compact subgraph. Let $\bar{g}$ be a subgraph induced by $V(g) \bigcup V(g')$. It is easy to show that $\bar{g}$ is $\mathsf{density}(g')$-compact, which contradicts the condition that g' is the maximal $\mathsf{density}(g')$-compact subgraph in G. $\square$

Based on the above properties of LDS, our model can be used to identify all the dense regions of a graph. Nevertheless, many real-world graph mining applications (e.g., community detection) typically require to find the top-k dense regions of a graph. In this paper, we mainly focus on finding the top-k LDSes with largest density from a graph, albeit our proposed techniques can also be extended to find all LDSes. We formulate our problem as follows.

Problem Statement. Given a graph G and an integer k, the LDS discovery problem is to compute the top-k LDSes with largest density in graph G.

4. TOP-K LDS DISCOVERY

In this section, we first present a basic algorithm that can solve the LDS discovery problem in polynomial time, and then we improve the algorithm by proposing several nontrivial pruning techniques to identify the top-k LDSes efficiently.

4.1 A Polynomial Algorithm

To compute all LDSes, we first show the following two lemmas.

Lemma 4.1: *Any densest subgraph component of graph G is an LDS in G.* $\square$

Proof Sketch: Let g be a densest subgraph component of G. We show that g is an LDS in G. First, we claim that g is $\mathsf{density}(g)$-compact. This can be proved by contradiction. Consider the case of deleting any subset S of nodes from g. Assume that the deletion of S results in removing less than $\mathsf{density}(g) \times |S|$ edges, then it can be easily derived that after deleting S from $V(g)$, the density of the residual graph is larger than $\mathsf{density}(g)$. This contradicts the condition that g is a densest subgraph component of G. Second, since g is a densest subgraph component of G, no supergraph of g in G has density $\geq \mathsf{density}(g)$. Hence, there does not exist a supergraph of g in G that is $\mathsf{density}(g)$-compact. $\square$

By Lemma 4.1, we can conclude that any densest subgraph component of the graph G, denoted by g, is the top-1 LDS in G. Moreover, we can prove that after deleting $V(g)$ from $V(G)$, all the other LDSes in the original G are still LDSes in the residual graph. More formally, we have the following lemma.

Lemma 4.2: *Let g be an LDS of G , then any LDS g' ($g' \neq g$) in G is still an LDS in G', where G' is the residual graph of G after removing g.* $\square$

Proof Sketch: Since all the LDSes in G are disjoint (Lemma 3.3), the LDS g' of G is a subgraph of G'. Therefore, to prove the lemma, it is sufficient to show that g' is a maximal $\mathsf{density}(g')$-compact subgraph in G'. This is true because g' is a maximal $\mathsf{density}(g')$-compact subgraph in G, and G' is a subgraph of G. Thus g' is also a maximal $\mathsf{density}(g')$-compact subgraph in G'. $\square$

Revisiting the Greedy Approach. By Lemma 4.1, we can determine the top-1 LDS in G by computing a densest subgraph component using the Goldberg's algorithm (Algorithm 1). By Lemma 4.2, we know that after removing an LDS, all the other LDSes in the original graph G are still LDSes in the residual graph G'. Therefore, in order to compute all LDSes of graph G, we need to revisit the greedy approach introduced in Section 3, which computes a densest subgraph component of the graph, removes it, and then computes all LDSes of the residual graph using the same procedure recursively until all nodes of the graph are removed. Lemma 4.1 and Lemma 4.2 ensure that such a greedy approach will not miss any LDS. However, as analyzed in Section 3, some results produced by the greedy approach may not be desirable. This indicates that the above greedy approach may introduce *false positive* results.

The reason for the greedy approach to introduce such false positive results is that, although any LDS g' in the original graph G is still an LDS in the residual graph G' after removing an LDS from G (Lemma 4.8), the converse may not be true, i.e., an LDS g' in the residual graph G' may not be an LDS in the original graph G. Let us consider an example. For the graph G shown in Fig. 1, by applying the greedy approach that iteratively invokes the Goldberg's algorithm, we first compute the subgraph G^*_{IR}, which is the top-1 LDS in G by Lemma 4.1. Next, if we remove G^*_{IR} from G and compute a densest subgraph component on the residual graph. We can get the second subgraph G'_{IR}. Obviously, G'_{IR} is an LDS in the residual graph but not an LDS in the original graph G. We continue the same procedure: removing G'_{IR} and computing a densest subgraph component on the residual graph, we can find the third subgraph G^*_{BN}, which is the second LDS in the original graph G. This example indicates that, in order to compute the correct top-k LDSes, an additional verification procedure to determine whether a result is false positive needs to be applied after each subgraph is generated in the greedy approach.

Detecting False Positives. In order to verify whether a subgraph g is an LDS in G, we need to follow Definiton 3.3 to check whether g is a maximal $\mathsf{density}(g)$-compact subgraph in G, which is nontrivial. Fortunately, after exploring the properties of the result returned by the $\mathsf{TryDensity}$ procedure in Algorithm 1, we can derive the following lemma, which can be used for LDS verification.

Lemma 4.3: *If G contains maximal ρ-compact subgraphs, then the result returned by the procedure* $\mathsf{TryDensity}(G, \rho - 1/|V(G)|^2)$ *is the set of all maximal ρ-compact subgraphs in G.* $\square$

Proof Sketch: It is easy to prove that any two maximal ρ-compact subgraphs are disjoint, since if not we can combine the two subgraphs to form a larger ρ-compact subgraph. Let G_1 the union of all maximal ρ-compact subgraphs in G, and let G_2 be the subgraph returned by $\mathsf{TryDensity}(G, \rho - 1/|V(G)|^2)$. By Lemma 2.2, G_2 maximizes $|E(G_2)| - \rho \times |V(G_2)|$, and it is the largest subgraph with maximum $|E(G_2)| - \rho \times |V(G_2)|$. Now we prove that G_1 and G_2 are the same.

(1) We prove that G_2 is a subgraph of G_1. To do this, it is sufficient to prove that each connected component g_2 of the graph G_2 is ρ-compact. Suppose to the contrary that a connected component g_2 of G_2 is not ρ-compact, then there exists a subset $S \subseteq V(g_2)$ such that removing S will result in removing $< \rho \times |S|$ edges from g_2. Let g'_2 be the graph after removing nodes S from g_2; we have $|E(g_2)| - |E(g'_2)| < \rho \times |S| = \rho \times (|V(g_2)| - |V(g'_2)|)$. Therefore, $|E(g_2)| - \rho \times |V(g_2)| < |E(g'_2)| - \rho \times |V(g'_2)|$. As a result, replacing g_2 by its subgraph g'_2 in G_2 will enlarge the value of $|E(G_2)| - \rho \times |V(G_2)|$. This contradicts the condition that G_2 has the maximum $|E(G_2)| - \rho \times |V(G_2)|$.

(2) We prove that G_1 is a subgraph of G_2. Suppose to the contrary that G_1 is not a subgraph of G_2, then according to (1), we can derive that $V(G_2) \subset V(G_1)$. Let $S = V(G_1) \setminus V(G_2)$,

Algorithm 2 LDS(graph G, integer k)

```
1: $G' \leftarrow G$;
2: for $i = 1$ to $k$ do
3:     find $\leftarrow$ false;
4:     while not find and $G' \neq \emptyset$ do
5:         $g \leftarrow$ any connected component of Densest($G'$);
6:         $G' \leftarrow$ the residual graph of $G'$ after deleting $g$;
7:         if Verify($g, G$) then { find $\leftarrow$ true; output $g$; }
8: Procedure Verify(subgraph $g$, graph $G$)
9: $g' \leftarrow$ TryDensity($G$, density($g$) $- 1/|V(G)|^2$);
10: return $g$ is a connected component in $g'$;
```

then $S \neq \emptyset$. Since G_1 is the union of all maximal ρ-compact subgraphs, removing S from G_1 will result in removing $\geq \rho \times |S|$ edges from G_1. In other words, $|E(G_1)| - |E(G_2|) \geq \rho \times |S| = \rho \times (|V(G_1)| - |V(G_2)|)$. Therefore, $|E(G_1)| - \rho \times |V(G_1)| \geq |E(G_2)| - \rho \times |V(G_2)|$. As a result, enlarging G_2 to G_1 will not decrease $|E(G_2)| - \rho \times |V(G_2)|$. This contradicts the condition that G_2 is the largest subgraph with maximum $|E(G_2)| - \rho \times |V(G_2)|$.

According to (1) and (2), the lemma is proved. □

The Algorithm. Armed with Lemma 4.3, after we compute the densest subgraph g in each iteration of the above greedy algorithm, we can invoke $\mathsf{TryDensity}(G, \mathsf{density}(g) - 1/|V(G)|^2)$ to verify whether g is a maximal $\mathsf{density}(g)$-compact subgraph in G by checking whether g is a connected component of the subgraph returned by $\mathsf{TryDensity}(G, \mathsf{density}(g) - 1/|V(G)|^2)$. If so, g must be an LDS in G (by definition); otherwise it is a false positive result. It is worth mentioning that the verification procedure is executed on the original graph G rather than the residual graph. The detailed description of our algorithm is outlined in Algorithm 2. In the worst case, our algorithm invokes the Goldberg's algorithm (line 5) and the verification procedure (line 7) at most $O(n)$ times, thus the time complexity of our algorithm is $O(m \cdot n \cdot (m+n) \cdot \log^2 n)$ by using the Sleator and Tarjan's maximal flow algorithm [19]. Note that in line 5, instead of using the Goldberg's algorithm, we can adopt any other algorithm such as [22] for densest subgraph computation. Clearly, this basic algorithm is costly, and thus cannot be used for large graphs. Below, we propose a practical optimized algorithm with several effective pruning techniques to handle large graphs.

4.2 The Optimized Algorithm

Our optimized algorithm involves three nontrivial optimization techniques, namely, pruning invalid nodes, optimizing $\mathsf{Densest}(G)$ computation, and optimizing LDS verification. Below, we first detail all these techniques, and then present our optimized algorithm.

4.2.1 *Pruning Invalid Nodes*

We first define the invalid node as follows.

Definition 4.1: (Invalid Node) A node v in a graph G is invalid if and only if there does not exist an LDS that contains v. □

Then, we prove that after deleting the invalid nodes, the LDSes in the original graph are still LDSes in the residual graph.

Lemma 4.4: *Let v be an invalid node in G, then after removing v from G, any LDS in G is still an LDS in the residual graph.* □

Proof Sketch: The proof is similar to the proof of Lemma 4.2. □

By Lemma 4.4, we can prune invalid nodes to reduce the size of the graph, since we can operate on the residual graph without missing any LDS. Moreover, after pruning invalid nodes, the graph can be divided into several smaller connected graphs each of which can be handled separately. Here, similar to Algorithm 2, we still need a verification procedure after computing an LDS from the residual graph by removing the invalid nodes. The remaining issue is how to determine the invalid nodes efficiently. Below, we propose two effective pruning rules to identify the invalid nodes.

Algorithm 3 Prune(subgraph G', graph G)

```
1: Compute core($v, G'$) for all $v \in V(G')$; $S_{\mathsf{invalid}} \leftarrow \emptyset$;
2: for all $v \in V(G')$ do
3:     $\underline{\rho}(v) \leftarrow$ core($v, G$)/2;
4:     $\overline{\rho}(v) \leftarrow \min\{\overline{\rho}(v)$, core($v, G'$)};
5: for all $v \in V(G')$ do
6:     if $\overline{\rho}(v) < \underline{\rho}(v)$ then $S_{\mathsf{invalid}} \leftarrow S_{\mathsf{invalid}} \bigcup \{v\}$;
7:     if $\exists u \in N(v, G)$ s.t. $\overline{\rho}(v) < \underline{\rho}(u)$ then
8:         $S_{\mathsf{invalid}} \leftarrow S_{\mathsf{invalid}} \bigcup \{v\}$;
9: $\tilde{G} \leftarrow$ the residual subgraph of $G'$ after removing $S_{\mathsf{invalid}}$.
10: return $\tilde{G}$;
```

Pruning Rules. We define $\underline{\rho}(v)$ as any ρ such that there exists a ρ-compact subgraph g of G that contains v. We also define $\overline{\rho}(v)$ as follows: if there is an LDS g in G that contains v, then $\overline{\rho}(v)$ is an upper bound of $\mathsf{density}(g)$; otherwise, $\overline{\rho}(v)$ can be any nonnegative real value. By these definitions, when v is contained in an LDS g, $\underline{\rho}(v)$ and $\overline{\rho}(v)$ can be deemed as the lower and upper bound of the density of g respectively. The pruning rules are detailed below.

Lemma 4.5: (Pruning Rules) *For any node $v \in V(G)$, v is invalid if either of the following two conditions is satisfied:*
(Rule-1)*: $\overline{\rho}(v) < \underline{\rho}(v)$;*
(Rule-2)*: there is a $u \in N(v, G)$ with $\overline{\rho}(v) < \underline{\rho}(u)$.* □

Proof Sketch: First, if a node v meets rule-1, then it clearly is an invalid node, as its upper bound is smaller than its lower bound. Second, for rule-2, we need to show that if $\overline{\rho}(v) < \underline{\rho}(u)$, then v cannot be contained in any LDS in G. Suppose, to the contrary, that there is an LDS g containing v, then we have $\mathsf{density}(g) \leq \overline{\rho}(v) < \underline{\rho}(u)$. Since there exists a $\underline{\rho}(u)$-compact subgraph g' that contains u, and u and v are adjacent in G, then by computing the subgraph induced by $V(g) \bigcup V(g')$, we will obtain a larger $\mathsf{density}(g)$-compact subgraph that contains g. Since g is an LDS, g must be the maximal $\mathsf{density}(g)$-compact subgraph, which is a contradiction. □

Computing $\overline{\rho}(v)$ and $\underline{\rho}(v)$. According to Lemma 4.5, we can determine whether a node v is invalid based on $\overline{\rho}(v)$ and $\underline{\rho}(v)$. The remaining problem is to efficiently compute $\overline{\rho}(v)$ and $\underline{\rho}(v)$ for any node v. The following two lemmas can be applied to achieve this goal. One is to compute $\underline{\rho}(v)$ and the other is to compute $\overline{\rho}(v)$.

Lemma 4.6: *An r-core subgraph is $\frac{r}{2}$-compact.* □

Proof Sketch: Let g be an r-core subgraph. Clearly, each node in g has degree at least r. If we delete any subset of nodes $S \subseteq V(g)$, then we will delete at least $\frac{r}{2} \times |S|$ edges, because each edge is deleted at most twice. Thus, by definition, g is $\frac{r}{2}$-compact. □

Lemma 4.6 indicates that for any $v \in V(G)$, $\underline{\rho}(v)$ can be assigned as $\mathsf{core}(v, G)/2$, because there is a $\mathsf{core}(v, G)/2$-compact subgraph of G that contains v.

Lemma 4.7: *If g is an LDS of G, then $\mathsf{core}(v, G) \geq \mathsf{density}(g)$ for all $v \in V(g)$.* □

Proof Sketch: The lemma can be immediately obtained from the result of Lemma 3.2. □

Based on Lemma 4.7, for any node $v \in V(G)$, we can set $\overline{\rho}(v)$ to be $\mathsf{core}(v, G)$. The reason is that if there is an LDS g containing v, then $\mathsf{core}(v, G)$ is an upper bound of $\mathsf{density}(g)$. If v is an invalid node, we can still set $\overline{\rho}(v)$ to be $\mathsf{core}(v, G)$, because by our definition, $\overline{\rho}(v)$ can be any nonnegative real value in such a case. Note that Lemma 4.5, Lemma 4.6, and Lemma 4.7 still hold if we want to compute the LDSes of a subgraph G' of G.

The Pruning Algorithm. Equipped with Lemma 4.5, Lemma 4.6, and Lemma 4.7, we design an algorithm to prune all the invalid nodes in a subgraph G' of G, which is detailed in Algorithm 3. The algorithm first computes $\underline{\rho}(v)$ and $\overline{\rho}(v)$ for each $v \in V(G')$

Algorithm 4 $\mathsf{Densest}^*$(graph G)

```
1: Compute $\underline{\rho_{\max}}$ by using the 1/2-approximation greedy algorithm [6];
2: Compute the $\lceil \underline{\rho_{\max}} \rceil$-core of $G$, denoted by $G'$;
3: $g^* \leftarrow \emptyset$;
4: for all connected component $g$ of $G'$ do
5:     $g' \leftarrow \mathsf{Densest}(g)$;
6:     if $g^* = \emptyset$ or $\mathsf{density}(g') > \mathsf{density}(g^*)$ then $g^* \leftarrow g'$;
7:     else if $g^* \neq \emptyset$ and $\mathsf{density}(g') = \mathsf{density}(g^*)$ then $g^* \leftarrow g^* \bigcup g'$;
8: $\rho^* \leftarrow \mathsf{density}(g^*)$;
9: $\underline{\rho}(v) \leftarrow \max\{\underline{\rho}(v), \rho^*\}$ for all $v \in V(g^*)$;
10: $\overline{\rho}(v) \leftarrow \min\{\overline{\rho}(v), \rho^* - \frac{1}{|V(G)|^2}\}$ for all $v \in V(G) \setminus V(g^*)$;
11: return $g^*$;
```

based on Lemma 4.6 and Lemma 4.7 respectively (lines 2-4), and then applies the pruning rules in Lemma 4.5 to prune invalid nodes in G' (lines 5-9). Note that $\underline{\rho}(v)$ only needs to be initialized once because it is independent of G'. We prune invalid nodes in a subgraph G' rather than G because Algorithm 3 can be applied for multiple times, i.e., each time after updating G to its residual graph G' by removing the identified LDSes and invalid nodes, we can invoke $\mathsf{Prune}(G', G)$ to compute and prune new invalid nodes.

4.2.2 *Optimizing* $\mathsf{Densest}(G)$ *Computation*

In order to show how to reduce the computational cost in the $\mathsf{Densest}(G)$ procedure in Algorithm 2, we first give an upper bound for the density of the densest subgraph.

Lemma 4.8: *If g is the maximal densest subgraph of G, then* $\mathsf{core}(v, G) \geq \mathsf{density}(g)$ *for all $v \in V(g)$.* □

Proof Sketch: Since each connected component of g is an LDS in G, the lemma immediately holds by Lemma 4.7. □

Lemma 4.8 indicates that if we can compute a lower bound $\underline{\rho_{\max}}$ for the density of the densest subgraph of G, then all nodes $v \in V(G)$ with $\mathsf{core}(v, G) < \underline{\rho_{\max}}$ cannot be contained in the maximal densest subgraph g of G. The rationality is that if a node $v \in V(G)$ with $\mathsf{core}(v, G) < \underline{\rho_{\max}}$ is contained in the maximal densest subgraph g, then we have $\mathsf{core}(v, G) < \underline{\rho_{\max}} \leq \mathsf{density}(g)$, which contradicts Lemma 4.8. To derive the lower bound $\underline{\rho_{\max}}$, we can make use of the well-known linear-time greedy algorithm proposed by Asahiro et al. [6] which produces a 1/2-approximation densest subgraph [13]. Below, we show that if a node is not contained in the maximal densest subgraph, then we can delete it from G when computing the maximal densest subgraph of G.

Lemma 4.9: *Let g be the maximal densest subgraph of G, for any node $v \in V(G)$, if $v \notin V(g)$, then g is still the maximal densest subgraph in the residual graph G' after removing v from G.* □

Proof Sketch: Since each connected component of g is an LDS in G, the lemma immediately holds by Lemma 4.4. □

The $\mathsf{Densest}^*$ **Algorithm**. Based on Lemma 4.8 and Lemma 4.9, we can remove the nodes whose core numbers are smaller than the lower bound $\underline{\rho_{\max}}$, and then compute the maximal densest subgraph in the residual graph. In other words, we can compute the maximal densest subgraph in the $\lceil \underline{\rho_{\max}} \rceil$-core of G rather than in the original graph G. Note that the $\lceil \underline{\rho_{\max}} \rceil$-core of G is typically much smaller than the original graph G, thus our optimized algorithm can significantly reduce the computational cost. The detailed algorithm is depicted in Algorithm 4. We first compute the lower bound $\underline{\rho_{\max}}$ by using the linear-time 1/2-approximation greedy algorithm (line 1), and then compute the $\lceil \underline{\rho_{\max}} \rceil$-core G' (line 2). Subsequently, we compute the maximal densest subgraph g^* in G' by invoking Goldberg's algorithm (lines 4-7). Here, since some nodes are removed from G, G can be divided into several smaller connected components, each of which can be handled individually. Moreover, based on $\mathsf{density}(g^*)$, we can update the lower bound $\underline{\rho}(v)$ for each node

Algorithm 5 Verify^*(ρ-compact subgraph g with density ρ, graph G)

```
1: $G' \leftarrow G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$;
2: if there does not exist an already computed LDS $g'$ with $V(g') \subseteq V(G')$ and $\mathsf{density}(g') > \rho$ then
3:     return true;
4: return $\mathsf{Verify}(g, G')$;
```

$v \in V(g^*)$ (line 9), and refine the upper bound $\overline{\rho}(v)$ for each node $v \in V(G) \setminus V(g^*)$ (line 10). Finally, the algorithm outputs the maximal densest subgraph g^* (line 11). The correctness of Algorithm 4 can be guaranteed by Lemma 4.8 and Lemma 4.9.

4.2.3 *Optimizing LDS Verification*

We explore possible ways to optimize the procedure of LDS verification. First, we show that any ρ-compact subgraph is contained in a connected component of $\lceil \rho \rceil$-core.

Lemma 4.10: *If g is a ρ-compact subgraph of G, then g is contained in a connected component of the $\lceil \rho \rceil$-core of G.* □

Proof Sketch: The proof can be easily obtained by definition. □

For a ρ-compact subgraph g, let $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$ be the connected component of the $\lceil \rho \rceil$-core of G that includes all the nodes in $V(g)$. $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$ exists by Lemma 4.10. We prove that any LDS g in G is an LDS in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, and vice versa.

Lemma 4.11: *For a ρ-compact subgraph g of G, g is an LDS in G if and only if g is an LDS in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$.* □

Proof Sketch: First, if g is an LDS in G, by Lemma 4.10, g is contained in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$. It is easy to prove that g is an LDS in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$. Second, if g is an LDS in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, then g is a maximal $\mathsf{density}(g)$-compact subgraph in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$. Since $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$ is a connected component of the $\lceil \rho \rceil$-core of G and $\rho \leq \mathsf{density}(g)$, there is no supergraph of g larger than g in G that is $\mathsf{density}(g)$-compact. Hence, g is a maximal $\mathsf{density}(g)$-compact subgraph in G, and thereby g is an LDS in G. □

Based on Lemma 4.11, if subgraph g is ρ-compact, we can verify whether g is an LDS in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, rather than in the original graph G. The subgraph $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$ is usually much smaller than the original graph G, thus the new verification algorithm will be much more efficient than the basic verification algorithm used in Algorithm 2. Moreover, based on Lemma 4.1 and Lemma 4.11, we can derive the following two lemmas, which can be applied to further optimize the verification algorithm.

Lemma 4.12: *If a ρ-compact subgraph g of G is a densest subgraph component of $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, then g is an LDS in G.* □

Proof Sketch: The proof can be immediately obtained by Lemma 4.1 and Lemma 4.11. □

Lemma 4.13: *For any ρ-compact subgraph g with density ρ, if there does not exist an LDS g' in G with* $\mathsf{density}(g') > \rho$ *that is contained in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, then g is a densest subgraph component of $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$.* □

Proof Sketch: Suppose to the contrary that, g is not a densest subgraph component of $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, then by Lemma 4.1 and Lemma 4.11, any densest subgraph component in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$, denoted by g^*, which is denser than g, must be an LDS in graph $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$. This clearly contradicts our condition. □

The Verify^* **Algorithm**. The optimized LDS verification algorithm Verify^* improves the verification process in two ways. First, by Lemma 4.11, the verification process for a ρ-compact subgraph g can be confined on the subgraph $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$ rather than on the original graph G. Second, by Lemma 4.12 and Lemma 4.13, we can further optimize the verification algorithm by checking whether there is an already computed LDS g' contained in $G_{\mathsf{core}=\lceil \rho \rceil}(V(g))$

Algorithm 6 LDS^*(graph G, integer k)

```
1: $\underline{\rho}(v) \leftarrow 0, \overline{\rho}(v) \leftarrow +\infty$ for all $v \in V(G)$; $\mathcal{H} \leftarrow \emptyset$;
2: $G' \leftarrow \mathsf{Prune}(G, G)$;
3: for all connected component $g$ of $G'$ do
4:     $\rho \leftarrow \max_{v \in V(g)}\{\overline{\rho}(v)\}$; $\mathcal{H}.\mathsf{Push}(g, \rho, \text{false})$;
5: for $i = 1$ to $k$ do
6:     find $\leftarrow$ false;
7:     while not find and $\mathcal{H} \neq \emptyset$ do
8:         $(g, \rho, \mathsf{exact}) \leftarrow \mathcal{H}.\mathsf{Pop}()$;
9:         if exact then
10:            if $\mathsf{Verify}^*(g, G)$ then { find $\leftarrow$ true; output $g$; }
11:            continue;
12:        $g^* \leftarrow$ any connected component of $\mathsf{Densest}^*(g)$;
13:        $\mathcal{H}.\mathsf{Push}(g^*, \mathsf{density}(g^*), \text{true})$;
14:        $G' \leftarrow$ the residual graph of $g$ after deleting $g^*$;
15:        $G' \leftarrow \mathsf{Prune}(G', G)$;
16:        for all connected component $g$ of $G'$ do
17:            $\rho \leftarrow \max_{v \in V(g)}\{\overline{\rho}(v)\}$; $\mathcal{H}.\mathsf{Push}(g, \rho, \text{false})$;
```

that is denser than g. Recall that Algorithm 2 outputs the LDSes according to the non-decreasing order of density. Thus, when we verify a ρ-compact subgraph g with density ρ, if no denser LDS contained in $G_{\mathsf{core}=\lceil\rho\rceil}(V(g))$ is already computed, then g is a densest subgraph component of $G_{\mathsf{core}=\lceil\rho\rceil}(V(g))$ (by Lemma 4.13), and thus g must be an LDS in G (by Lemma 4.12). The detailed description of the Verify^* algorithm is outlined in Algorithm 5. To verify a ρ-compact subgraph g with density ρ, the algorithm first computes $G_{\mathsf{core}=\lceil\rho\rceil}(V(g))$ (line 1). Then, the algorithm checks whether there is an already computed LDS with density larger than ρ in $G_{\mathsf{core}=\lceil\rho\rceil}(V(g))$ (line 2). If there is no such an LDS, then the algorithm can directly return true, as g must be an LDS in G by Lemma 4.12 and Lemma 4.13 (line 3). Otherwise, the algorithm needs to verify g in $G_{\mathsf{core}=\lceil\rho\rceil}(V(g))$ by Lemma 4.11 (line 4).

4.2.4 The LDS^* Algorithm

By combining all the pruning techniques, we can derive the optimized LDS discovery algorithm, called the LDS^* algorithm. We outline the LDS^* algorithm in Algorithm 6. Specifically, the algorithm maintains a priority queue $\mathcal{H}$ to compute the top-k LDSes. Each entry in $\mathcal{H}$ is a triplet that consists of three different elements: g, ρ, and a boolean variable. Here g denotes a subgraph, ρ denotes the priority of g, which is an upper bound for the density of any LDS contained in g, and the boolean variable is used to determine whether g is a ρ-compact subgraph with density ρ. For each subgraph g, ρ can be initialized to be the maximal value of $\overline{\rho}(v)$ over all $v \in V(g)$, because such a maximal value must be an upper bound for the density of any LDS contained in g.

Initially, Algorithm 6 sets $\underline{\rho}(v)$ to be 0 and $\overline{\rho}(v)$ to be $+\infty$ for every node v in G (line 1). Then, the algorithm invokes Algorithm 3 to prune the invalid nodes, resulting in a residual graph G' (line 2). Subsequently, the algorithm pushes all the connected components of G' into $\mathcal{H}$ (lines 3-4). After that, the algorithm finds the top-k LDSes in k iterations (lines 5-17). In each iteration, it processes the popped entry from $\mathcal{H}$, denoted by $(g, \rho, \mathsf{exact})$ (line 8). If g is a ρ-compact subgraph with density ρ (exact is true), then it invokes Algorithm 5 to verify whether it is an LDS, and if so, the algorithm outputs it and continues to the next iteration (lines 9-11). If exact is false, then the algorithm computes the maximal densest subgraph of g using Algorithm 4 (line 12). Since each densest subgraph component g^* of g must be $\mathsf{density}(g^*)$-compact, the algorithm randomly selects one densest subgraph component g^* of g and pushes $(g^*, \mathsf{density}(g^*), \mathsf{true})$ into the priority queue $\mathcal{H}$ (line 13). Note that here the priority of g^* is set to be $\mathsf{density}(g^*)$ and the boolean variable is set to be true, because g^* is a $\mathsf{density}(g^*)$-compact subgraph with density $\mathsf{density}(g^*)$. After that, the algorithm obtains the residual graph G' by deleting subgraph g^* (line 14). Then, it invokes Algorithm 3 to prune the invalid nodes in G' (line 15), and also pushes each connected component of G' into the priority queue $\mathcal{H}$ (lines 16-17). The correctness of the algorithm can be guaranteed by the results shown in Section 4.2.1, 4.2.2, and 4.2.3.

Dataset	n	m	$d_{\max}$	density
Indochina	7,414,866	194,109,311	256,425	26.18
UK	39,459,925	936,364,282	1,776,858	23.73
Livejournal	5,363,260	79,023,142	19,432	14.73
Patent	3,774,768	16,518,947	793	4.38
Arabic	22,744,080	639,999,458	575,628	28.14
WebBase	118,142,155	1,019,903,190	816,127	8.63
Coauthor	5,411	17,477	96	3.23

Table 1: Datasets

5. EXPERIMENTS

In this section, we show our experimental results. All of our experiments were conducted on a machine with an Intel Xeon 3.4GHz CPU and 32GB main memory running 64-bit Red Had Linux.

Datasets. We use 7 real-world graphs with different graph properties for testing. The detailed statistics of all the 7 datasets are shown in Table 1, where d_{max} is the maximum degree of nodes in the graph. Among all the graphs, Livejournal is a social network of a virtual-community social site, Patent is a citation network among US patents, Coauthor is a co-author network, and others are web graphs. Indochina, UK, Livejournal, Arabic, and WebBase are downloaded from LAW (http://law.di.unimi.it/), Patent is downloaded from Konect (http://konect.uni-koblenz.de/), and Coauthor is obtained from ArnetMiner (http://arnetminer.org). Due to lack of space, for effectiveness testing, we use four graphs with different statistics: Arabic (dense web graph), WebBase (sparse web graph), Livejournal (social network), and Patent (citation network). For efficiency testing, we use the four largest graphs among all the datasets listed in Table 1. Coauthor is used for a case study.

Algorithms. For effectiveness testing, we compare the dense subgraphs identified by our LDS model with the dense subgraphs computed by [37], which adopts the greedy approach to identify top-k dense subgraphs, and is the state-of-the-art algorithm for dense subgraph identification. [37] aims to find dense subgraphs called optimal quasi-cliques which are a set of clique-like dense subgraphs. We adopt the local search optimization strategy introduced in [37] to find each optimal quasi-clique. We denote our algorithm as LDS and the greedy approach introduced in [37] as QC. For QC, all parameters are set to their default values suggested in [37].

For efficiency testing, we compare two algorithms, denoted as LDS and LDS^*. LDS follows the framework in Algorithm 2 by replacing the Densest procedure at line 5 with $\mathsf{Densest}^*$ (Algorithm 4), and replacing the Verify procedure at line 7 with Verify^* (Algorithm 5). The reason that we use $\mathsf{Densest}^*$ and Verify^* is that, without adopting the two optimized algorithms, we cannot output a result in a reasonable time for any of the datasets due to the high time complexity of the algorithm. LDS^* follows Algorithm 6 with all optimization strategies involved. All algorithms are in-memory algorithms and are implemented in C++.

Quality Measures. We adopt the following four quality measures for effectiveness testing:

- *Density* ρ. The typical density definition shown in Eq. 1.
- *Relative Density* ρ_r. Relative density is a popular graph cluster-fitness measure that takes both the inter and intra edges of a subgraph into consideration [33]. Intuitively, a subgraph with high relative density indicates that the density of the subgraph is high and the density of its nearby region is relatively low. The relative density is formally defined as follows:

$$\rho_r(g, G) = \frac{|E(g)|}{|E(g)| + |E'(g, G)|} \quad (2)$$

where $E'(g, G) = \{(u, v) | (u, v) \in E(G), u \in V(g), v \notin V(g)\}$ is the set of inter-edges for subgraph g in G.

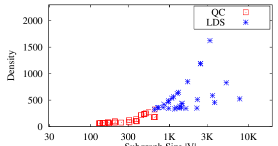
- *Edge Density ρ_e.* Edge density is the ratio of the number of edges in a graph to the number of edges in a complete graph with the same set of nodes, which is defined as:

$$\rho_e(g) = \frac{2 \times |E(g)|}{|E(g)| \times (|E(g)| - 1)} \tag{3}$$

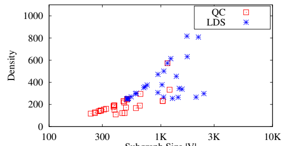
  Note that the edge density can reflect the density of a subgraph; however, edge density alone cannot be a good density measure. As an example, a subgraph of a single edge has edge density 1, but it is not a dense subgraph.

- *Diameter.* The diameter of a graph is the longest distance of all pairs of nodes in the graph, where the distance of two nodes is the minimum number of hops to reach from one node to another.

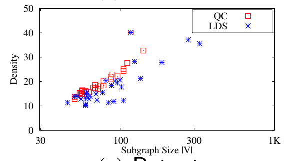
In our effectiveness testing, for each dataset, we compute the top-30 dense subgraphs by both LDS and QC, and draw the statistics for each of the above quality measures with respect to the size (number of nodes) of each subgraph. In the following, we show the effectiveness testings of the four quality measures in Exp-1 to Exp-4 respectively, report the result of efficiency testing in Exp-5, and perform a case study in Exp-6.



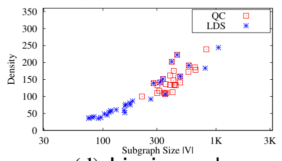
(a) Arabic



(b) WebBase



(c) Patent

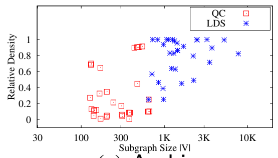


(d) Livejournal

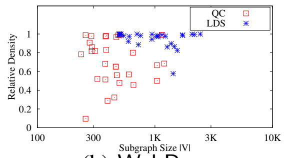
Figure 2: Density Testing

Exp-1: Density Testing. The results for density testing for the four datasets are shown in Fig. 2 (a)-(d) respectively. In Arabic (Fig. 2 (a)), we observe that LDS tends to find large dense subgraphs and QC tends to find small subgraphs with relatively lower density. For example, the top-1 subgraph reported by LDS has 3,250 nodes with density 1,624.5, whereas the top-1 subgraph reported by QC has 648 nodes with density 323.1. The results in WebBase (Fig. 2 (b)) are similar to those in Arabic. In Patent (Fig. 2 (c)), the subgraphs reported by LDS and QC have similar density range. For the same density, LDS tends to report larger subgraphs than QC to represent the corresponding local dense regions. In Livejournal (Fig. 2 (d)), the subgraphs reported by LDS have a wider size range (from 75 to 1051) and a wider density range (from 35.6 to 244.2), whereas the subgraphs reported by QC have a relatively narrower size range (from 221 to 820) and a relatively narrower density range (from 99.8 to 238.9). This is because a social network usually has a small number of large dense regions. In such a case, LDS tends to identify both large and small subgraphs to represent different local dense regions, whereas QC tends to find dense subgraphs within a certain large dense region in the graph.

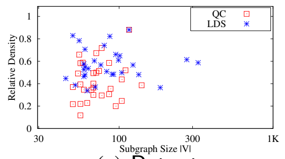
Exp-2: Relative Density Testing. In this experiment, we test the relative density and show our testing results in Fig. 3. In Arabic (Fig. 3 (a)), we observe that LDS tends to identify large subgraphs with higher relative density, whereas QC tends to find small subgraphs with lower relative density. In WebBase (Fig. 3 (b)), most of the subgraphs reported by LDS have a relative density higher



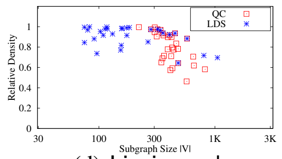
(a) Arabic



(b) WebBase



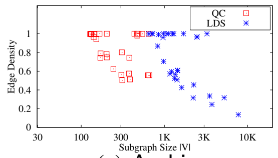
(c) Patent



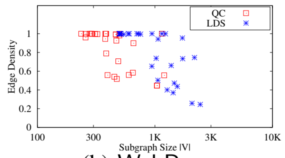
(d) Livejournal

Figure 3: Relative Density Testing

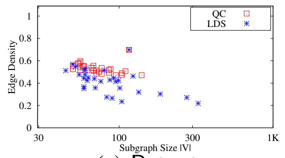
than 0.9, while almost half of the subgraphs reported by QC have a relative density lower than 0.5. This indicates that LDS does not only consider the density of each identified subgraph, but also tends to distinguish each subgraph in its local region. The results for Patent are shown in Fig. 3 (c). For the same subgraph size, the subgraphs reported by LDS usually have higher relative density than the subgraphs reported by QC. In Livejournal (Fig. 3 (d)), although LDS identifies a lot of smaller subgraphs than QC, the subgraphs reported by LDS have higher relative density than the subgraphs returned by QC on average. Such a result further confirms our observation in Exp-1, that is, LDS tends to identify both large and small subgraphs to represent different local dense regions, whereas QC tends to find dense subgraphs within a certain large dense region in the graph. Here, it is worth noting that in Arabic, WebBase, and Patent, the subgraphs identified by QC have a small size which may result in the low relative density. In Livejournal, the subgraphs identified by LDS also have a small size. However, the relative density of those subgraphs is high. This shows that LDS can indeed find the representative dense subgraphs of the graph.



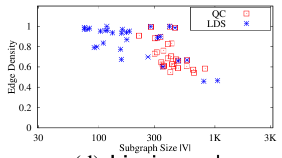
(a) Arabic



(b) WebBase



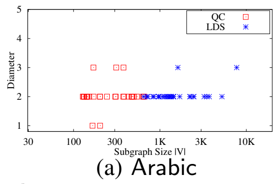
(c) Patent



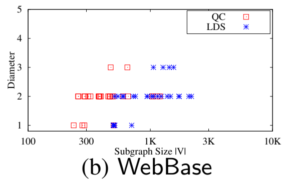
(d) Livejournal

Figure 4: Edge Density Testing

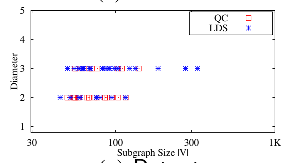
Exp-3: Edge Density Testing. The results for edge density testing are shown in Fig. 4. In the Arabic dataset (Fig. 4 (a)), we observe that both LDS and QC can identify a large number of subgraphs with high edge density. For the same edge density, the size of the subgraphs reported by LDS is usually larger than the size of the subgraphs reported by QC. In addition, LDS can also report some large subgraphs with relatively lower edge density. This indicates that the corresponding local regions do not contain large clique-like subgraphs. Therefore, although the edge density of the subgraphs is not high, the reported subgraphs are also representative in their corresponding local regions. The testing results for WebBase (Fig. 4 (b)) are similar to those for Arabic. In Fig. 4 (c), we can see that, in the Patent dataset, the edge density for the subgraphs identified by both LDS and QC is usually no larger than 0.6. The subgraphs reported by LDS have a wider range of size and edge density to represent different local dense regions of the graph. In the Livejournal dataset (Fig. 4 (d)), LDS can find small subgraphs with high edge density and large subgraphs with relatively lower edge density to represent different local regions of the graph, while QC usually outputs subgraphs with medium size and edge density.



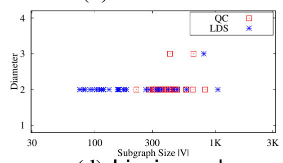
(a) Arabic



(b) WebBase



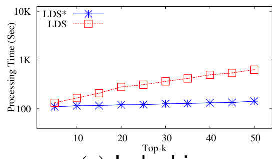
(c) Patent



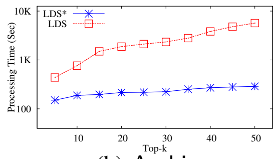
(d) Livejournal

Figure 5: Diameter Testing

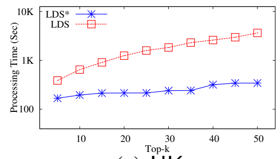
Exp-4: Diameter Testing. In this experiment, we test the diameter of the subgraphs returned by LDS and QC. The experimental results are shown in Fig. 5. For the Arabic dataset (Fig. 5 (a)), the diameters of the subgraphs returned by both LDS and QC are no larger than 3. Although the size of the subgraphs returned by LDS is larger than the size of those returned by QC, the diameters of the subgraphs returned by the two algorithms are similar. We have similar observation on the other three datasets WebBase (Fig. 5 (b)), Patent (Fig. 5 (c)), and Livejournal (Fig. 5 (d)). Comparing the results of diameter testing with the results of edge density testing shown in Fig. 4, it is worth noting that although LDS can report subgraphs with relatively lower edge density than QC, the diameters of the subgraphs identified by LDS are as small as the diameters of those identified by QC. This indicates that the nodes in the subgraphs returned by LDS are tightly connected.



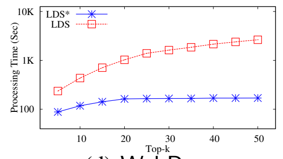
(a) Indochina



(b) Arabic



(c) UK



(d) WebBase

Figure 6: Efficiency Testing

Exp-5: Efficiency Testing. In this experiment, we test the efficiency of the two algorithms LDS and LDS*. We vary k from 5 to 50 and report the time to compute the top-k results for LDS and LDS*. The testing results for the four datasets are shown in Fig. 6. For all the four datasets, when k increases, the processing time for both LDS and LDS* increases. Both algorithms consume a certain amount of time to report the first result in order to compute the core numbers of all nodes and initialize some variables. LDS* outperforms LDS in all testing cases. The gap of the processing time between LDS and LDS* increases when k increases. The reason is that LDS does not compute and prune the invalid nodes. Therefore, to output each result, it needs to search globally from the whole graph by excluding the reported subgraphs. LDS* prunes the invalid nodes using two pruning rules and after each result is reported, it can be used to prune more invalid nodes. Furthermore, after removing invalid nodes, the whole graph can be divided into several smaller connected graphs each of which can be handled individually. Therefore, the search space for LDS* is largely reduced compared to LDS, especially when k is large. When k reaches 50, LDS* is 4.4, 19.7, 10.6, and 15.6 times faster than LDS for the Indochina (Fig. 6 (a)), Arabic (Fig. 6 (b)), UK (Fig. 6 (c)), and WebBase (Fig. 6 (d)) datasets, respectively.

k	LDS			Greedy			QC		
	n	ρ	area	n	ρ	area	n	ρ	area
1	45	14.6	DS	45	14.6	DS	25	11.6	IR
2	25	11.6	IR	25	11.6	IR	45	14.6	DS
3	28	10.4	BN	28	10.4	BN	28	10.4	BN
4	70	7.4	SW	70	7.4	SW	18	7.2	DS
5	28	5.1	DM	18	7.2	DS	23	6.7	SW
6	74	4.6	ML	247	6.0	DS	9	4.0	SW

Table 2: Case Study on Coauthor Dataset

Exp-6: Case Study. In this experiment, we perform a case study on the Coauthor dataset. The aim of this case study is to show that the subgraphs identified by LDS can indeed represent different dense regions of the whole graph. Coauthor is a dataset that records the coauthor relationship of researchers in different research areas including Database Systems (DS), Information Retrieval (IR), Machine Learning (ML), Data Mining (DM), Bayesian Networks (BN), and Semantic Web (SW). Each author in the network is labeled with the main research areas of the author. A subgraph belongs to a certain research area if all the authors (nodes) in the subgraph are labelled with the corresponding research area.

In addition to our LDS algorithm and the QC algorithm, we also report the subgraphs returned by the Greedy algorithm, which is to recursively identify the densest subgraph in the residual graph by deleting the identified dense subgraphs in the previous iterations. For each algorithm, we report the top-6 subgraphs, and show the number of nodes n, density ρ, and the research area for each returned subgraph. The experimental results are depicted in Table 2.

From the experimental results, we can see that the top-6 subgraphs returned by LDS belong to 6 different research areas, i.e., each returned subgraph can be used to represent the densest research community in its corresponding research area. In comparison, Greedy and QC only report the subgraphs in 4 research areas DS, IR, BN, and SW, with DM and ML missing. For Greedy, it finds two additional subgraphs in the DS area, one is small with 18 nodes and density 7.2, and the other is large with 247 nodes and density 6.0. Both subgraphs are subsumed by a denser subgraph that is in the same dense region with the densest subgraph in the DS area (the top-1 subgraph reported by LDS). Therefore, they cannot be used to represent the local dense regions. For QC, it finds an additional subgraph in the DS area and an addition subgraph in the SW area. For the two subgraphs in the SW area reported by QC, they have densities of 6.7 and 4.0 respectively. However, a denser subgraph with density 7.4 (the fourth subgraph reported by LDS) is not identified. Therefore, the subgraphs returned by LDS are the best to represent local dense regions of the graph.

6. RELATED WORK

Densest Subgraph Discovery. The densest subgraph discovery problem aims at finding the subgraph with maximal average degree from a graph [25]. It is well known that this problem can be solved in polynomial time by using the parametric flow technique [25, 22]. Moreover, as shown by Charikar [13], a linear-time greedy algorithm proposed by Asashiro et al. [6] can obtain a 1/2-approximation densest subgraph. Such a greedy algorithm is recently extended to MapReduce and streaming model by Bahmani et al. [7]. However, when we restrict the size of the densest subgraph, the problem becomes NP-hard [5]. Several interesting variants of the size-constrained densest subgraph as well as efficient approximation algorithms are discussed in [3, 27]. More recently, Tsourakakis et al. [37] propose another interesting variant of the densest subgraph model, termed optimal quasi-clique, based on a new definition of the density function, which can produce a subgraph with higher quality than the densest subgraph. Similar to the size-constrained densest subgraph, the optimal quasi-clique discovery problem is also NP-hard [36]. Unlike all these studies, in this work, we propose a new densest subgraph model, called locally densest subgraph (LDS). Based on our model, we are able to identify all the locally densest regions of a graph in polynomial time, which cannot be found by the previous densest subgraph models. We also propose a practical algorithm for finding the top-k locally densest subgraphs from a large graph.

Cohesive Subgraph Mining. Cohesive subgraph is an important concept in social network analysis and graph mining [39]. The cohesive subgraph mining focuses on enumerating all the cohesive subgraphs from a graph. Unlike the densest subgraph, which is defined based on the density, the cohesive subgraph is typically defined by a relaxation of clique model. In the literature, there are a number of cohesive subgraph models [34, 16, 18, 39]. Notable examples include maximal clique [16], k-plex [39], k-core [34, 9, 29], k-truss [18, 38], and maximal k-edge connected subgraph [12, 2]. For all these cohesive subgraph models, many practical algorithms are proposed. For example, James et al. propose several I/O efficient algorithms for finding maximal clique [16], k-core [15], and k-truss [38] from a disk-resident graph. Chang et al. [12] propose a linear-time algorithm for identifying maximal k-edge connected subgraph from a large graph. Akiba et al. [2] propose a different linear-time algorithm for finding the maximal k-edge connected subgraph based on a random edge contraction technique.

Other Models. In addition to the above two categories, there are related work on other models based on various graph properties (e.g., [1, 24, 30]) and graph applications (e.g., [35, 8, 10]). However, in this paper, we only focus on the density-based model, which is a key primitive in a variety of applications as shown Section 1.

7. CONCLUSION

In this paper, we study the problem of discovering the top-k locally densest subgraphs (LDSes) in a graph, which can be used to identify the local dense regions of a graph, and can be applied in a variety of application domains. We provide a parameter-free definition of an LDS with several useful properties. We show that the LDSes of a graph can be computed in polynomial time, and propose three novel optimization strategies to improve the algorithm. We conduct extensive experiments using seven real datasets to demonstrate the effectiveness and efficiency of our approach.

Acknowledgements. Lu Qin is supported by ARC DE140100999. Rong-Hua Li is supported by NSFC Grants 61402292 and Natural Science Foundation of SZU (grant no. 201438). Lijun Chang is supported by ARC DE150100563. Chengqi Zhang is supported by ARC DP140100545 and ARC DP140102206.

8. REFERENCES

[1] B. D. Abrahao, S. Soundarajan, J. E. Hopcroft, and R. Kleinberg. On the separability of structural classes of communities. In *Proc. of KDD'12*, 2012.
[2] T. Akiba, Y. Iwata, and Y. Yoshida. Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction. In *Proc. of CIKM'13*, 2013.
[3] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, volume 5427 of *Lecture Notes in Computer Science*. Springer, 2009.
[4] A. Angel, N. Koudas, N. Sarkas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *PVLDB*, 5(6), 2012.
[5] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1-3), 2002.
[6] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2), 2000.
[7] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5), 2012.
[8] P. Basaras, D. Katsaros, and L. Tassiulas. Detecting influential spreaders in complex, dynamic networks. *IEEE Computer*, 46(4), 2013.
[9] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.
[10] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proc. of WWW'13*, 2013.
[11] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM'08*, 2008.
[12] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang. Efficiently computing k-edge connected components via graph decomposition. In *Proc. of SIGMOD'13*, 2013.
[13] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proc. of APPROX'00*, 2000.
[14] J. Chen and Y. Saad. Dense subgraph extraction with application to community detection. *TKDE*, 24(7), 2012.
[15] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu. Efficient core decomposition in massive networks. In *Proc. of ICDE'11*, 2011.
[16] J. Cheng, Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu. Finding maximal cliques in massive networks. *Trans. Database Syst.*, 36(4), 2011.
[17] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. In *Proc. of SODA'02*, 2002.
[18] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. *Technique report, National Security Agency*, 2008.
[19] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill, 2001.
[20] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *WWW'07*, 2007.
[21] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. Motifcut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14), 2006.
[22] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *J. Comput.*, 18(1), 1989.
[23] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *Proc. of VLDB'05*, 2005.
[24] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proc. of KDD'12*, 2012.
[25] A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, 1984.
[26] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 3-hop: a high-compression indexing scheme for reachability query. In *SIGMOD'09*, 2009.
[27] S. Khuller and B. Saha. On finding dense subgraphs. In *Proc. of ICALP'09*, 2009.
[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*. Springer, 2010.
[29] R. Li, J. X. Yu, and R. Mao. Efficient core maintenance in large dynamic graphs. *TKDE*, 26(10), 2014.
[30] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Proc. of PAKDD'10*, 2010.
[31] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Proc. of RECOMB'10*, 2010.
[32] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Proc. of RECOMB'10*, 2010.
[33] S. E. Schaeffer. Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1), 2007.
[34] S. B. Seidman. Network structure and minimum degree. *Social networks*, 5(3), 1983.
[35] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In *Proc. of ICDM'14*, 2014.
[36] C. E. Tsourakakis. *Mathematical and Algorithmic Analysis of Network and Biological Data*. PhD thesis, Carnegie Mellon University, 2013.
[37] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proc. of KDD'13*, 2013.
[38] J. Wang and J. Cheng. Truss decomposition in massive networks. *PVLDB*, 5(9), 2012.
[39] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.