International Journal of Modern Physics C
© World Scientific Publishing Company

Mining Connected Global and Local Dense SubGraphs for BigData

Bo Wu and Haiying Shen

Department of Electrical and Computer Engineering, Clemson University
Clemson, South Carolina 29634, USA
bwu2(shenh)@clemson.edu

The problem of discovering *connected* dense subgraphs of natural graphs is important in data analysis. Discovering dense subgraphs that do not contain denser subgraphs or are not contained in denser subgraphs (called *significant dense* subgraphs) is also critical for wide-ranging applications. In spite of many works on discovering dense subgraphs, there are no algorithms that can guarantee the connectivity of the returned subgraphs or discover significant dense subgraphs. Hence, in this paper, we define two subgraph discovery problems to discover connected and significant dense subgraphs, propose polynomial-time algorithms and theoretically prove their validity. We also propose an algorithm to further improve the time and space efficiency of our basic algorithm for discovering significant dense subgraphs in big data by taking advantage of the unique features of large natural graphs. In the experiments, we use massive natural graphs to evaluate our algorithms in comparison with previous algorithms. The experimental results show the effectiveness of our algorithms for the two problems and their efficiency. This work is also the first that reveals the physical significance of significant dense subgraphs in natural graphs from different domains.

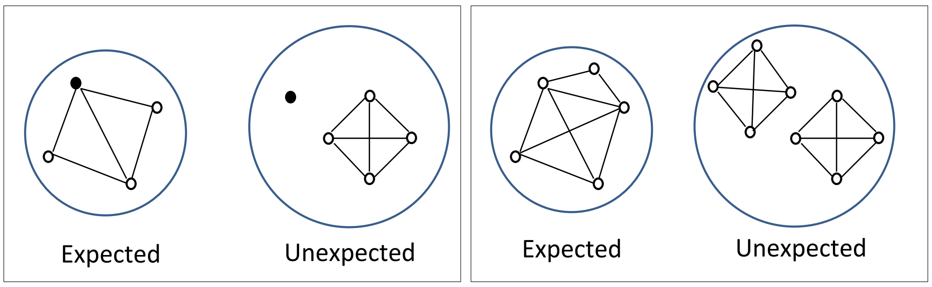
Keywords: Densest subgraph; Complex network; Natural graph.

PACS Nos.: 11.25.Hf, 123.1K

1. INTRODUCTION

The natural graph is one of the most important data structures in this big data era. The emergence of many networks, such as the World Wide Web, the social networks, big molecules, and gene network, has raised the importance of analyzing natural graphs. Natural graphs have many interesting and important features. A challenging problem in natural graph analysis is defining and identifying communities, which plays a significant role in many applications such as social network community mining, computational biology, link spam detection and network design [1, 7, 13, 16, 20, 25]. In spite of many definitions of community [14, 24, 27], the definition of the dense subgraph performs well in various applications [7, 13, 16, 20, 25] and it is much more easier to implement when the datasets become large.

The dense subgraph problem has been widely studied in theoretical works [2, 3, 6, 11, 12, 14, 19, 25] and application works [7, 13, 20, 25]. These works are focused on different subproblems such as discovering the densest subgraph [3, 6, 14, 19], the k-densest subgraph [11], and the densest subgraph with the restriction of containing



(a) With the restriction (b) Without restrictions
Fig. 1. Connectivity problem in previous algorithms

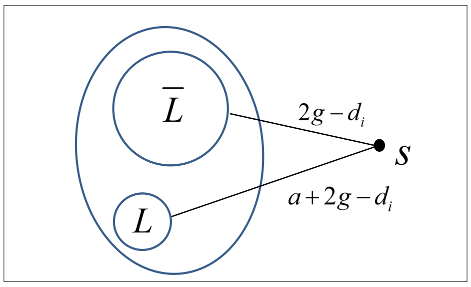
a specific subset and of distance [25].

Among the densest subgraph discovery methods [3,6,14,25], the precise method in [25] finds the densest subgraph with the restriction of containing a specific vertex subset, and the precise method in [14] and approximate method in [3,6] find the densest subgraph with no restrictions. However, the precise algorithm in [25] and the approximate algorithms in [3,6] cannot guarantee the connectivity of the returned subgraphs, which however is always required in community detection in many applications (e.g., community discovery in social network, function module discovery in gene structure). Figure 8 shows possible returned subgraphs when discovering the densest subgraph containing a specific vertex subset using the algorithm in [25]. Figure 7 shows possible identified densest subgraph without restrictions using the algorithms in [3,6]. We see that the returned subgraphs may contain multiple isolated connected subgraphs, since they try to find a denser subgraph. Though the precise method [14] precisely solves the densest subgraph problem in polynomial time, which can return connected subgraphs, it lacks the capability of handling large datasets due to very high memory and time consumption. Therefore, a computation efficient method for identifying connected densest subgraph is highly desired.

When it comes to find multiple dense subgraphs in a graph, previous algorithms [13,25] just find all subgraphs with density higher than a threshold. These algorithms neglect the connectivity problem of the detected subgraphs. Further, considering the graph structure and the real applications such as community detecting [7] and gene pattern annotation [25], a denser subgraph may be not a good choice since it contains or is contained in a denser subgraph, while a sparser subgraph may be a good choice since it does not contains or is not contained in a denser subgraph. However, there are no previous works that can identify dense subgraphs which do not contain denser subgraphs or are contained in denser subgraphs (called *significant dense subgraphs*).

In this paper, we aim to handle the aforementioned problems existing in previous dense subgraph identification algorithms. Specifically, we define two new subproblems and summarize our contribution below.

- We define the *Discovering local connected densest subgraph problem* (LCDS)



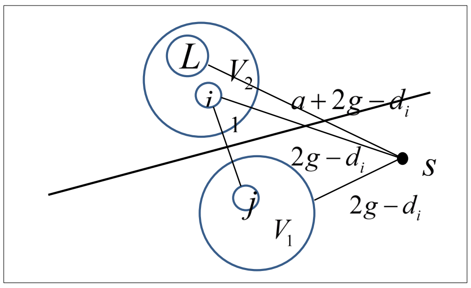


Fig. 2. Graph construction

Fig. 3. TCPM graph partition

which is to discover the connected densest subgraph with the restriction of containing a specific vertex subset. Then we propose a precise algorithm to solve it in polynomial time and prove the algorithm's effectiveness and design rationality.

- We define the *Discovering global significant dense subgraphs problem* (GS-DS), which is to discover all significant dense subgraphs and also are connected. Then we propose an algorithm to solve it in polynomial time and prove the algorithm's effectiveness and design rationality.
- We first invent the *Two connected partitions min-cut* (TCPM) technique which can discover the LCDSs and GSDSs. TCPM technique is different from min-cut max-flow technique used in discovering the original densest subgraph [14].
- Based on the feature of natural graphs, we provide an improved GSDS algorithm to reduce the time and space complexity of our basic GSDS algorithm, which can easily handle data with GB-level size in one PC.
- We conduct extensive experiments on massive natural graphs. The results show the effectiveness and efficiency of our algorithms in comparison with previous algorithms, and the enhanced efficiency of our improved GSDS algorithm.

The rest of this paper is organized as follows. Section 7 presents the related work. Section 2 introduces the basic concepts and techniques. Section 3 defines the LCDS problem and proposes a polynomial-time solution. Section 4 defines the GSDS problem and proposes a basic polynomial-time algorithm and an improved algorithm. Section 5 presents experimental results and analyzes the structure of the natural graphs. Section 8 summarizes the paper with remarks on our future works.

2. Preliminaries

We first introduce the concepts used in this paper. Let $G = (V, E)$ be an undirected graph. For a subset $S \subseteq V$, the *induced edge set* is defined as $E(S) = E \cap S^2$ and the *induced degree* of a node $i \in S$ is defined as $\deg_S(i) = |\{j|(i,j) \in E(S)\}|$. In the remaining parts of this paper, we use vertex subset S to denote graph $G_S = (S, E(S))$ for short sometimes.

Definition 1. Density of an undirected graph [3]: Let $G = (V, E)$ be an undirected graph. Given $S \subseteq V$, its density $\rho(S)$ is defined as $\rho(S) = \frac{|E(S)|}{|S|}$. The maximum density $\rho^*(S)$ of the graph is $\rho^*(S) = \max\limits_{S \subseteq V}\{\rho(S)\}$.

The densest subgraph is the subgraph that has the maximum density.

Definition 2. Capacity of an edge [26]: Let $N = (V, E)$ be a network (directed graph) with s and t being the source and the sink of N respectively. The capacity of an edge (u, v) is denoted by $c(u, v)$. It represents the maximum amount of flow that can pass through an edge.

Definition 3. Capacity of a cut [26]: A cut $C = (V_1, V_2)$ is a partition of V, where $V_1 \cup V_2 = V$, and $V_1 \cap V_2 = \emptyset$. The cut-set of C is the set $\{(u, v) \in E | u \in V_1, v \in V_2\}$. The capacity of a cut $C = (V_1, V_2)$ is defined by $c(V_1, V_2) = \sum\limits_{(u,v) \in V_1 \times V_2} c(u, v)$. It represents the sum of the capacities of the edges connecting two partitions V_1 and V_2 (i.e., cut-set).

Definition 4. Min-cut max-flow (min-cut in short) problem [26]: This problem is to minimize $c(V_1, V_2)$, that is, to determine V_1 and V_2 such that the capacity of cut $c(V_1, V_2)$ is minimal.

When all the capacities of the edges in the graph are nonnegative, the min-cut problem can be solved in polynomial time [26]. However, if there are negative edges, the min-cut problem is an NP-hard problem [23]. The min-cut problem with nonnegative capacities was applied to solve the original densest subgraph problem [14]. In order to solve the LCDS problem in this paper, we first propose a two connected partitions min-cut problem as follows:

Definition 5. Two connected partitions min-cut problem (TCPM): This problem is to minimize $c(V_1, V_2)$, that is, to determine V_1 and V_2 such that the capacity of the cut is minimal, and meanwhile vertex subsets V_1 and V_2 are connected, respectively.

This problem can be solved in polynomial time using a simple min-cut algorithm [26], since a simple min-cut algorithm just considers the min-cut of the two connected partitions and negative capacity does not influence the validity of the algorithm.

3. Discovering the LCDS

In this section, we define the LCDS problem and propose a polynomial-time solution.

Definition 6. Local connected densest subgraph of a vertex subset *L*: Given a connected graph $G = (V, E)$, and a certain vertex subset L, where $L \subseteq V$. We call $G_L = (L^+, E(L^+))$ a local connected densest subgraph (LCDS) of L, if and only if that subgraph G_L is connected, $L \subseteq L^+$, and there are no such connected subgraph $G_S = (S, E(S))$ that satisfies $\rho(S) > \rho(L^+)$, and $L \subseteq S$.

3.1. *A Polynomial-time Algorithm*

According to the interesting relationship between LCDS and TCPM problems shown later, the LCDS problem can be solved by trial and error in polynomial time. Given an estimated density of the LCDS of L (denoted by g), we can construct a specific graph (denoted by N). The LCDS can be reduced from one partition of TCPM of N if the estimation is right. Otherwise, we can judge whether g is too big or small based on the result whether the capacity of TCPM (denoted by c_{min}) is bigger than a certain value. Therefore, we can adjust the lower and upper bounds of the density of the LCDS of L (denoted by l_{min} and u_{min}, respectively) and give g a new value based on l_{min} and u_{min}. In this way, we apply a binary searching scheme to reach the "just right" g.

Algorithm 1: Algorithm for discovering LCDS of L

```
1: Given: $G = (V, E)$;
2: $l_{min} \leftarrow 0, u_{min} \leftarrow n$;
3: while $(l_{min} - u_{min}) \geq \frac{1}{n(n-1)}$ do
       $g \leftarrow \frac{l_{min}+u_{min}}{2}$;
       Construct $N = (V_N, E(V_N))$;
       Find TCPM $(V_1 \cup s, V_2)$;
       Calculate $c_{min}$;
       if $c_{min} \geq a|L|$ then
           $u_{min} \leftarrow g$
       end
       if $c_{min} < a|L|$ then
           $l_{min} \leftarrow g$
       end
   end
4: return subgraph of $G$ induced by $V_1$;
```

We first set $l_{min} = 0$ and $u_{min} = n$ where $n = |V|$. In the step of graph construction, given a g, we convert the graph $G = (V, E)$ to graph $N = (V_N, E_N)$ as shown in Figure 2. We add a vertex s to the set of vertices of V, allocate each edge of E by a capacity of 1, connect every vertex i of $V \backslash L$ to vertex s by an edge of capacity $(2g - d_i)$, and connect every vertex i of L to vertex s by an edge of capacity $(a + 2g - d_i)$, where a is a negative constant smaller than the twice of the sum of other negative edge capacities in the graph N and d_i is the degree of vertex i of G. More formally,

$$\begin{aligned}
V_N &= V \cup \{s\} \\
E_N &= \{(i,j)|\{i,j\} \in E(V)\} \cup \{(i,s)|i \in L\} \\
&\quad \cup\{(i,s)|i \in V\backslash L\} \\
c_{ij} &= 1 && \{i,j\} \in E(V) \\
c_{si} &= 2g - d_i && i \in V\backslash L \\
c_{it} &= a + 2g - d_i && i \in V
\end{aligned}$$

Then, we find a TCPM $(V_1 \cup s, V_2)$ of N. In Theorem 1, we find the relationship between c_{min} and the density bounds of the LCDS. Therefore, we can adjust g based on c_{min}. If $c_{min} \geq a|L|$, we update u_{min} with the current value of g; if $c_{min} < a|L|$, we update $l_{min}$with the current value of g. When the stop condition ($(l_{min} - u_{min}) < \frac{1}{n(n-1)}$ which is proved in Theorem 2) is satisfied, we get the final LCDS of L from V_2. Otherwise, we update g by $g = (l_{min} + u_{min})/2$ using the binary searching scheme, re-construct N based on the updated g, and repeat the above process until the stop condition is satisfied.

In the following, we analyze the validity of this algorithm including the determination of lower and upper bounds of the density of the LCDS of L, the determination of the stop condition, the connectivity of the discovered LCDS of L and its time complexity.

3.2. *Proving the Validity of the Algorithm*

3.2.1. *Relationship between Capacity and Density Bounds of the LCDS*

In order to continually narrow the scope of the density of the LCDS, we need a method to determine the lower and upper bounds based on the constructed TCPM in each loop in Algorithm 1. Therefore, we prove the relationship between the possible capacity of the constructed TCPM and the bounds of the LCDS below.

Lemma 1. *Suppose $(V_1 \cup \{s\}, V_2)$ is a TCPM of the above constructed graph $N = (V_N, E_N)$, then vertex subset $L \subset V_2$.*

Proof. We know that any cut $c(V_1, V_2)$ where $L \subset V_2$ has capacity $c_L \leq (a|L| - \frac{a}{2})$, since a is a negative constant smaller than the twice of the sum of other negative edge capacities in the graph N. Suppose there is a cut with capacity c_{L^-} which a vertex subset L^- where $L^- \subset L$ and $L^- \subset V_1$, then $c_{L^-} > a|L\backslash L^-| + \frac{a}{2}$. Also, we know $(a|L\backslash L^-| + \frac{a}{2}) \geq (a|L| - \frac{a}{2})$, since $L^- \neq \emptyset$. Therefore, $c_{L^-} > c_L$. Hence, vertex subset $L \subset V_2$ for TCPM $(V_1 \cup \{s\}, V_2)$. □

Theorem 1. *Suppose $c(V_1 \cup s, V_2)$ is a TCPM of the graph $N = (V_N, E_N)$, which has capacity c_{min} and L^+ is the LCDS of L, then the g parameter in Algorithm 1 satisfies $g \geq \rho(L^+)$ if and only if $c_{min} \geq a|L|$, and it satisfies $g \leq \rho(L^+)$ if and only if $c_{min} \leq a|L|$.*

Proof. As we can see from Figure 3, the capacity of the TCPM equals:

$$\begin{aligned}c_{min} &= \sum_{i\in V_1, j\in V_2} c_{ij} \\ &= \sum_{j\in V_2} c_{sj} + \sum_{i\in V_1, j\in V_2} c_{ij}\end{aligned}$$

We know that vertex s is only connected to vertices in vertex subset L from the definition of TCPM. Also based on Lemma 1, we know that $L \subset V_2$. Hence, $\sum_{j\in V_1} c_{sj} = 0$. Therefore, we have:

$$\begin{aligned}c_{min} &= \sum_{i\in L}(a + 2g - d_i) + \sum_{i\in V_2\setminus L}(2g - d_i) + \sum_{i\in V_1, j\in V_2} 1 \\ &= a|L| + 2|V_2|(g - \frac{\sum_{i\in V_2} d_i - \sum_{i\in V_1, j\in V_2} 1}{2|V_2|}) \\ &= a|L| + 2|V_2|(g - \rho(V_2))\end{aligned}$$

Since $|V_2| \geq |L|$ and $|L| > 0$, $2|V_2|(g - \rho(V_2)) = 0$ if and only if $g = \rho(V_2)$. Suppose $g \leq \rho(L^+)$, then we can find a cut $c(V_1, V_2)$ where $\rho(V_2) \geq g$. Such a cut can lead to $2|V_2|(g - \rho(V_2)) \leq 0$. Hence, capacity $c_{min} \leq a|L|$. Conversely, suppose $c_{min} \leq a|L|$, then, $2|V_2|(g - \rho(V_2)) \leq 0$. Then $g \leq \rho(V_2)$. We know $\rho(V_2) \leq \rho(L^+)$ from the definition. Therefore, $g \leq \rho(L^+)$.

Suppose $g \geq \rho(L^+)$, then we know $\rho(V_1) \leq \rho(L^+)$ from the definition. Hence, $2|V_2|(g - \rho(V_2)) \geq 0$. Therefore the TCPM $c_{min} \geq a|L|$. Conversely, suppose the TCPM $c_{min} \geq a|L|$. Then, $2|V_2|(g - \rho(V_2)) \geq 0$. Then, $g \neq \rho(V_2)$. Also $\rho(V_2) \leq \rho(L^+)$. Therefore, $g \leq \rho(L^+)$. □

Based on the above property, we design the binary searching scheme of the algorithm.

3.2.2. *Determining the Stop Condition of the Algorithm*

Lemma 2. *Suppose there is a connected graph $N = (V_N, E_N)$, two vertex subsets S_1 and S_2 where $S_1, S_2 \subset V_N$. Then, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$, where $n = |V_N|$.*

Proof. Suppose the number of the edges in subgraph $G_1 = (S_1, E(S_1))$ and $G_2 = (S_2, E(S_2))$ is m_1 and m_2, respectively. Then, $\rho(S_1) = \frac{m_1}{|S_1|}$, and $\rho(S_2) = \frac{m_2}{|S_2|}$. We have:

$$\begin{aligned}|\rho(S_1) - \rho(S_2)| &= |\frac{m_1}{|S_1|} - \frac{m_2}{|S_2|}| \\ &= \frac{|m_1|S_2| - m_2|S_1||}{|S_1||S_2|}\end{aligned}$$

Since $\rho(S_1) \neq \rho(S_2)$, we can divide the above equation to three conditions: i) $|S_1| > |S_2|, m_1 \leq m_2$; ii) $|S_1| > |S_2|, m_1 \geq m_2$; and iii) $|S_1| = |S_2|, m_1 \leq m_2$.

In case i), $|S_1| \cdot |S_2| \leq \frac{1}{n(n-1)}$ and $m_1|S_2| - m_2|S_1| \geq m_1$. Then, we have $|\rho(S_1) - \rho(S_2)| = \frac{m_1}{n(n-1)}$. Hence, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$. Similarly, in case ii), $|\rho(S_1) -$

$\rho(S_2)| \geq \frac{1}{n(n-1)}$. In case iii), $|S_1| < n$ since $\rho(S_1) \neq \rho(S_2)$. Therefore, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{(n-1)^2}$. In all the three conditions, $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{n(n-1)}$. □

Theorem 2. *Suppose $|u_{min} - l_{min}| < \frac{1}{n(n-1)}$, then we can ensure that the graph with density bigger than l_{min} is the LCDS solution.*

Proof. Based on Lemma 2, we know we can guarantee that there is only one subgraph with a density bigger than l_{min} if $|u_{min} - l_{min}| < \frac{1}{n(n-1)}$. Then, proof completes. □

3.2.3. *Connectivity Guarantee of the LCDS of L*

Theorem 3. *Suppose $(V_1 \cup s, V_2)$ is a TCPM of the graph $N = (V_N, E_N)$, then vertex subset V_2 is connected.*

Proof. V_2 is one partition of the TCPM. Therefore, we know that V_2 is connected from the definition of the TCPM problem. □

Theorem 3 indicates that the LCDS of L discovered by Algorithm 1 is connected.

3.2.4. *Time Complexity*

Previous studies have proved that TCPM problem can be solved in polynomial time [26]. In the experiment, we apply a simple min-cut algorithm [18] with time complexity $O(|V||E|+|V|^2 log|V|)$. Also, the stop condition can be met in $O(log|V_N|)$ times of estimations (the time complexity of binary search) and each estimation requires one TCPM computation. Then, the total time complexity of the LCDS discovery algorithm is $O(|V||E|log|V| + |V|^2 log^2|V|)$.

4. Discovering the GSDSs

In this section, we define the GSDS problem, and propose a polynomial-time solution.

Definition 7. Global significant dense subgraph: Given a connected graph $G = (V, E)$, and a connected subgraph $G_S = (S, E(S))$ where $S \subset V$. G_S is a GSDS if and only if there are no such connected subgraph $G_{S^+} = (S^+, E(S^+))$ that satisfies $\rho(S^+) > \rho(S)$ and $S \subset S^+$, and also there are no such connected subgraph $G_{S^-} = (S^-, E(S^-))$ that satisfies $\rho(S^-) > \rho(S)$ and $S^- \subset S$.

4.1. ***A Basic Polynomial-time Algorithm***

Based on the definition, the idea of this basic algorithm is to find candidates, which do not contain denser subgraphs, and then check whether they are contained in denser subgraphs to get the final results. This algorithm checks GSDSs based on the following three rules. First, the densest subgraph is a GSDS. Second, GSDSs are disjointed from each other. Third, if a candidate subgraph is the LCDS of itself and does not contain denser subgraphs, then it is a GSDS. Accordingly, we reduce the GSDS problem of G to a series of computations of discovering the densest subgraph and the LCDS problems, which can be solved by the algorithm in [14] and Algorithm 1, respectively.

Algorithm 2 presents the pseudocode of this basic polynomial-time algorithm. In step 1 (block 3 - 4), we find the densest subgraph $G_1 = (V_1, E(V_1))$ of G, and add G_1 to GSDS list D_{list}. In step 2 (block 6), we find the densest subgraph $G_2 = (V_2, E(V_2))$ from the remaining graph $G \backslash G_1$. For subgraph G_2, there are no denser subgraphs inside G_2 based on the definition of the densest subgraph. However, to check whether there are denser subgraphs containing G_2, we find LCDS $G_{2L} = (V_{2L}, E(V_{2L}))$ of V_2. If $\rho(V_2) \geq \rho(V_{2L})$, then there are no denser subgraphs contain G_2, which we will prove in Theorem 6, and add G_2 to D_{list}. We continue this process in the remaining graph of G until the remaining of graph G becomes empty. It is obvious that the maximum times of the process is $|V|$.

Algorithm 2: Basic algorithm for discovering GSDSs

```
1: Given: $G = (V, E)$;
2: $G_0 = (V_0, E_0) \leftarrow G$;
3: Find densest subgraph $G_1 = (V_1, E(V_1))$ of $G$;
4: Add $G_1$ to $D_{list}$;
5: $G \leftarrow G \backslash G_1$, $i \leftarrow 2$;
6: while $G \neq \emptyset$ do
       Find $G_i = (V_i, E(V_i))$ of $G$;
       // $G_i = (V_i, E(V_i))$ is the densest subgraph of $G$
       Find $G_{iL} = (V_{iL}, E(V_{iL}))$;
       // $G_{iL} = (V_{iL}, E(V_{iL}))$ is the LCDS of $V_i$ of $G_0$
       $G \leftarrow G \backslash G_i$, $i++$;
       if $\rho(V_i) \geq \rho(V_{iL})$ then
           Add $G_i$ to $D_{list}$;
       end
   end
7: return $D_{list}$;
```

4.2. *Proving the Validity and Properties of the Algorithm*

In this section, we prove the three rules followed in this algorithm by studying the properties of GSDSs and analyze the time complexity.

4.2.1. *Properties of GSDSs*

Theorem 4. *Suppose $G_1 = (V_1, E(V_1))$ is the densest subgraph of graph $G = (V, E)$, then G_1 is a GSDS.*

Proof. From the definition of the densest subgraph, we know there are no subgraphs of G that are denser than G_1. Then, there are no subgraphs that contain G_1 are denser than G_1. Also, there are no subgraphs contained in G_1 that are more denser than G_1. Therefore, by the definition of the GSDS, G_1 is a GSDS. $\square$

Theorem 4 supports our first rule.

Theorem 5. *Suppose $G_1 = (V_1, E(V_1))$ and $G_2 = (V_2, E(V_2))$ are two GSDSs of graph $G = (V, E)$, then we have $V_1 \cap V_2 = \emptyset$.*

Proof. We prove it by contradiction. Suppose $V_1 \cap V_2 \neq \emptyset$, then we have a connected subgraph $G_{12} = ((V_1 \cup V_2), E(V_1 \cup V_2))$. Then we calculate the density $\rho(V_1 \cup V_2)$ as follows:

$$\rho(V_1 \cup V_2) = \frac{E(V_1 \cup V_2)}{|V_1 \cup V_2|}$$

Let $V_b = V_1 \cap V_2$. Then we have:

$$\rho(V_1 \cup V_2) = \frac{\rho(V_1)|V_1| + \rho(V_2)|V_2| - \rho(V_b)|V_b|}{|V_1| + |V_2| - |V_b|}$$

Since G_1 and G_2 are GSDSs, we have $\rho(V_1) > \rho(V_b)$ and $\rho(V_2) > \rho(V_b)$. Suppose $\rho(V_1) \geq \rho(V_2)$, then we have:

$$\begin{aligned}\rho(V_1 \cup V_2) &> \frac{\rho(V_2)|V_1| + \rho(V_2)|V_2| - \rho(V_2)|V_b|}{|V_1| + |V_2| - |V_b|} \\ &= \rho(V_2)\end{aligned}$$

Therefore, G_2 is not a GSDS from the definition. It contradicts with the assumption. Therefore, we must have $V_1 \cap V_2 = \emptyset$. $\square$

Theorem 5 supports our second rule, which guarantees that we can find all the GSDSs.

Theorem 6. *Suppose ${G_1}^* = ({V_1}^*, E({V_1}^*))$ is the densest subgraph of part of the graph G, then ${G_1}^*$ is the GSDS of G, if and only if ${G_1}^*$ is the LCDS of ${V_1}^*$ of G.*

Proof. Firstly, we prove that ${G_1}^*$ is the GSDS if ${G_1}^*$ is the LCDS of ${V_1}^*$ of G. From the definition of the LCDS, we know there are no subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_1}^* \subset {V_0}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Also, from the densest subgraph definition, we know there are no subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_0}^* \subset {V_1}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Therefore, ${G_1}^*$ is the LCDS of ${V_1}^*$ of G. Secondly, we prove that ${G_1}^*$ is the GSDS only if ${G_1}^*$ is the LCDS of ${V_1}^*$ in G by contradiction. Suppose ${G_1}^*$ is not LCDS of ${V_1}^*$ in G, then there is a subgraph ${G_0}^* = ({V_0}^*, E({V_0}^*))$ where ${V_1}^* \subset {V_0}^*$ and $\rho({V_0}^*) > \rho({V_1}^*)$. Then ${G_1}^*$ is not a GSDS by the definition. Therefore, the condition that ${G_1}^*$ is the LCDS of ${V_1}^*$ of G is the necessary condition for that ${G_1}^*$ is a GSDS. $\square$

Theorem 6 supports our third rule.

4.2.2. *Time Complexity*

In order to discover the densest subgraph, we choose the push-relabel algorithm with dynamic trees [9]. The time complexity of this algorithm is $O(|V||E|log(|V|^2/|E|))$. For the TCPM problem, we choose a simple min-cut algorithm [26]. The time complexity of this algorithm is $O(|V||E| + |V|^2 log|V|)$. Therefore, the total time complexity for computing one LCDS is $O(|V||E|log(|V|^2/|E|)+|V||E|+|V|^2 log|V|)$. The total time complexity for computing one GSDS is $O(|V|^2|E|log(|V|^2/|E|)log|V| + |V|^2|E| + |V|^3 log|V|)$. For natural graphs, the time complexity is approximately $O(V^2 E log^2 V)$, since natural graphs are usually sparse graphs, which makes $|V| \approx |E|$.

Even though the basic GSDS algorithm can solve the problem in polynomial time, a time complexity of $O(V^2 E log^2 V)$ is still too high for large datasets especially in this big data era. To reduce its time complexity, we propose an improved GSDS algorithm below.

4.3. *An Improved Algorithm for Large Datasets*

It is well-known that the natural graphs usually follow a power-law degree distribution [4]. For graphs with such a feature, most of the vertices have low probabilities to be in the GSDSs, since they have very low degrees. Therefore, the basic idea of this improved algorithm is trying to reduce the initial size of the dataset by deleting the vertices with very low degrees. The detailed process is presented in Algorithm 3. For a given $G = (V, E)$, we first delete all the vertices, which have degrees equal or smaller than the maximum density of remaining graph during the deleting process (blocks 1-3) streamingly. Then, in addition to the same process as the basic algorithm, we delete the neighbors of the vertices of each densest subgraph found in the remaining graph (blocks 5-10). Block 11 returns the results. In the following, we prove the correctness of this algorithm.

4.3.1. *Properties Used for The Improvement*

Lemma 3.

Suppose $G_S = (V_S, E(V_S))$ is the densest subgraph of graph G, then we have $deg_{V_S}(i) \geq \rho(V_S)$ for any vertex i, where $i \in V_S$.

Proof. We prove it by contradiction. Suppose there is at least one vertex i, where $i \in V_S$, and $deg_{V_S}(i) < \rho(V_S)$, then we delete vertex i from G_S, and get graph $G_{S^-} = (V_S \backslash \{i\}, E(V_S \backslash i))$. The density of graph G_{S^-} is:

$$\begin{aligned} \rho(V_S \backslash \{i\}) &= \frac{|E(V_S \backslash i)|}{|V_S \backslash \{i\}|} \\ &= \frac{\rho(V_S)|V_S| - deg_{V_S}(i)}{|V_S| - 1} \end{aligned}$$

Since $deg_{V_S}(i) < \rho(V_S)$, we have:

Algorithm 3: Improved algorithm for discovering GSDSs

```
 1: Given: $G = (V, E)$;
 2: $S \leftarrow V$, $S_p \leftarrow \emptyset$, $\rho_{max} \leftarrow \rho(S)$;
 3: while $S \neq S_p$ do
        $S_p \leftarrow S$;
        $S_c \leftarrow \{i \in S | deg_S(i) \leq \rho_{max}\}$;
        $S \leftarrow S \backslash S_c$;
        if $\rho(S) > \rho_{max}$ then
            $\rho_{max} \leftarrow \rho(S)$;
        end
    end
 4: $G_0 = (S_0, E(S_0)) \leftarrow G_S = (S, E(S))$;
 5: Find densest subgraph $G_1 = (V_1, E(V_1))$ of $G_S$;
 6: Add $G_1$ to $D_{list}$;
 7: ${V_1}^* \leftarrow V_1 \cup \{j | (i, j) \in E(S), i \in V_1\}$;
 8: ${G_1}^* = ({V_1}^*, E({V_1}^*))$;
 9: $G_S \leftarrow G_S \backslash {G_1}^*$, $i \leftarrow 2$;
10: while $G_S \neq \emptyset$ do
        Find $G_i = (V_i, E(V_i))$;
        // it is the densest subgraph of $G_S$
        Find $G_{iL} = (V_{iL}, E(V_{iL}))$;
        // it is the LCDS of $V_i$ of $G_0$
        ${V_i}^* \leftarrow V_i \cup \{j | (i, j) \in E(S), i \in V_i\}$;
        ${G_i}^* = ({V_i}^*, E({V_i}^*))$;
        $G_S \leftarrow G_S \backslash {G_i}^*$, $i{+}{+}$;
        if $\rho(V_i) \geq \rho(V_{iL})$ then
            Add $G_i$ to $D_{list}$;
        end
    end
11: return $D_{list}$;
```

$$\begin{aligned}\rho(V_S \backslash \{i\}) &> \frac{\rho(V_S)(|V_S| - 1)}{|V_S| - 1} \\ &> \rho(V_S)\end{aligned}$$

This contradicts with the precondition. Therefore, we have $deg_{V_S}(i) \geq \rho(V_S)$ for any vertex i, where $i \in V_S$. □

Theorem 7. *After we delete all the vertices with degrees less or equal than the maximum density of the remaining graph of G in block 3 of Algorithm 3 to obtain G_S, all the GSDSs of G are in G_S.*

Proof. We prove it by contradiction. Suppose there is one GSDS $G_x = (V_x, E(V_x))$, where $G_x \not\subset G_S$, then there is a vertex subset I where $I \subset V_x$ and $I \not\subset V_s$. There is a time in the deleting process that the first vertex i in I is deleted from the current V_s (denoted by ${V_s}^+$). Therefore, we have:

$$\begin{aligned}\rho(V_x) &\leq deg_{V_x}(i) \\ &\leq deg_{{V_s}^+}(i) \\ &< \rho({V_s}^+)\end{aligned}$$

This implies that $\rho(V_x) < \rho({V_s}^+)$, and at this moment, we have $V_x \subset {V_s}^+$. Hence, from the definition of GSDS, we know that G_x is not a GSDS of G. This contradicts with the assumption. Therefore, for any GSDS G_x of G, we have $V_x \subset V_S$. □

Based on Theorem 7, we design block 3 in Algorithm 3.

Theorem 8. *Suppose $G_1 = (V_1, E(V_1))$ and $G_2 = (V_2, E(V_2))$ are two GSDSs of graph $G = (V, E)$, then there is no such an edge (v_1, v_2) in graph G, where $v_1 \in V_1$ and $v_2 \in V_2$.*

Proof. We prove it by contradiction. Suppose there is at least one edge (v_1, v_2) where $v_1 \in V_1$ and $v_2 \in V_2$, then we calculate the density of $G_0 = (V_1 \cup V_2, E(V_1 \cup V_2))$ as follows:

$$\begin{aligned}\rho(V_1 \cup V_2) &= \frac{|E(V_1 \cup V_2)|}{|V_1 \cup V_2|} \\ &\geq \frac{\rho(V_1)|V_1| + \rho(V_2)|V_2| + 1}{|V_1| + |V_2|}\end{aligned}$$

Suppose $\rho(V_1) \geq \rho(V_2)$, then we have:

$$\begin{aligned}\rho(V_1 \cup V_2) &> \frac{\rho(V_2)|V_1| + \rho(V_2)|V_2|}{|V_1| + |V_2|} \\ &= \frac{\rho(V_2)(|V_1| + |V_2|)}{|V_1| + |V_2|} \\ &= \rho(V_2)\end{aligned}$$

Also, we know that $V_2 \subset V_1 \cup V_2$. Therefore, G_2 is not a GSDS. This contradicts with the precondition. Therefore, there is no such an edge (v_1, v_2) in graph G, where $v_1 \in V_1$ and $v_2 \in V_2$. □

Based on Theorem 8, we design block 10 of Algorithm 3.

Theorem 9. *Suppose $G^* = (V^*, E(V^*))$ is a subgraph of $G = (V, E(V))$ and ${G_1}^* = ({V_1}^*, E({V_1}^*))$ is the densest subgraph of G^*, then there are no GSDSs of G in G^*, if $\rho({V_1}^*) < \rho(V)$.*

Proof. Since $V^* \subseteq V$ and ${V_1}^* \subseteq V^*$, we have ${V_1}^* \subseteq V$. Also, we know $\rho({V_1}^*) < \rho(V)$. Also, by the definition of the GSDS, ${G_1}^*$ is not a GSDS of G. Based on the definition of the densest subgraph, we know that, for any subgraph ${G_i}^* = ({V_i}^*, E({V_i}^*))$ in G^*, $\rho({V_i}^*) < \rho({V_1}^*) < \rho(V)$. Also, ${V_i}^* \subseteq V$. Hence, ${G_i}^*$ is not a GSDS of G. Therefore, there are no GSDSs of G in G^*. □

Based on Theorem 9, we design the stop condition of the loop in block 10 of Algorithm 3.

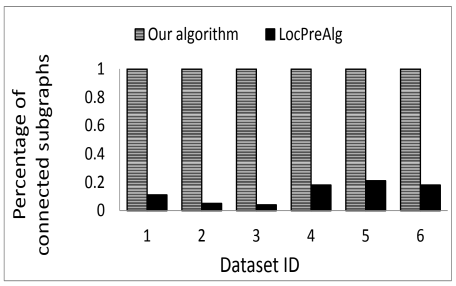
ID	Description	Domain
Dataset 1 [22]	Collaboration network	Social network
Dataset 2 [21]	Facebook	Social network
Dataset 3 [28]	Power Grid	Technology
Dataset 4 [17]	Protein interaction	Chemistry
Dataset 5 [15]	E-mail interchanges	Information
Dataset 6 [10]	Metabolic network	Biology

ID	Description	Domain
Dataset 7 [29]	LiveJournal	Social network
Dataset 8 [29]	Orkut	Social network

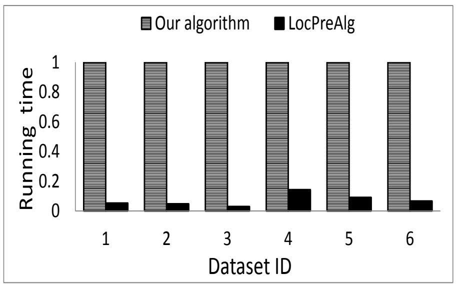
Datasets	# of vertices		# of edges		size (KB)	
	Before	After	Before	After	Before	After
Dataset 1	5,242	244	28,980	7,238	344	50
Dataset 2	9,877	599	25,998	8,264	644	59
Dataset 3	4,941	3,353	13,188	5,006	107	80
Dataset 4	1,846	624	4,406	1,121	34	15
Dataset 5	1,133	349	10,902	2,681	79	37
Dataset 6	453	66	2,066	301	14	4
Dataset 7	3,997,962	4,136	34,681,189	650,724	489,799	8,818
Dataset 8	3,072,441	20,723	117,185,083	2,087,932	1,728,293	100,352

5. Performance Evaluation and Analysis of Massive Natural Graphs

Recall that the precise algorithm [25] (denoted by *LocPreAlg*) that finds the densest subgraph containing a specific vertex subset and the approximate algorithms [3, 6] (denoted by *GloAppxAlg1* and *GloAppxAlg2*) that find the densest subgraph neglect the connectivity of the returned subgraphs. Although the previous precise algorithm [14] (denoted by *GloPreAlg*) can guarantee the connectivity of the returned subgraph, it is at the cost of high time and memory complexity. Also, no previous work can find GSDSs. In this section, we conduct experiments to show the effectiveness and efficiency of our proposed LSDS (Algorithm 1) and GSDS (Algorithm 3) algorithms in solving these problems in comparison with these previous algorithms. We also show the enhanced efficiency of our improved GSDS algorithm (Algorithm 3). We use two groups of massive datasets in Table 1 and Table 2 from different domains in our experiments. We use big datasets in Table 2 to test the capacity of manipulating big data of Algorithm 3. The algorithms are implemented by C language. The testing platform is one PC with 2.1GHz Intel core i3 processor with 2 cores, and a 4GB memory. The Operating System is Ubuntu 10.0. In all the following figures, we use the results of our algorithms as a baseline and shows the ratios of the results of other algorithms to our algorithms. Further, we use Algorithm 3 to extract the GSDSs of the massive natural graphs from social networks, technology, chemistry, and biology and conduct simple analysis to reveal the physical significance of GSDSs in natural graphs from different domains.



(a) Discovering connected subgraphs



(b) The running time

Fig. 4. The performances of the LCDS algorithm compared to previous algorithms

5.1. *Performance Evaluation of the LSDS Algorithm*

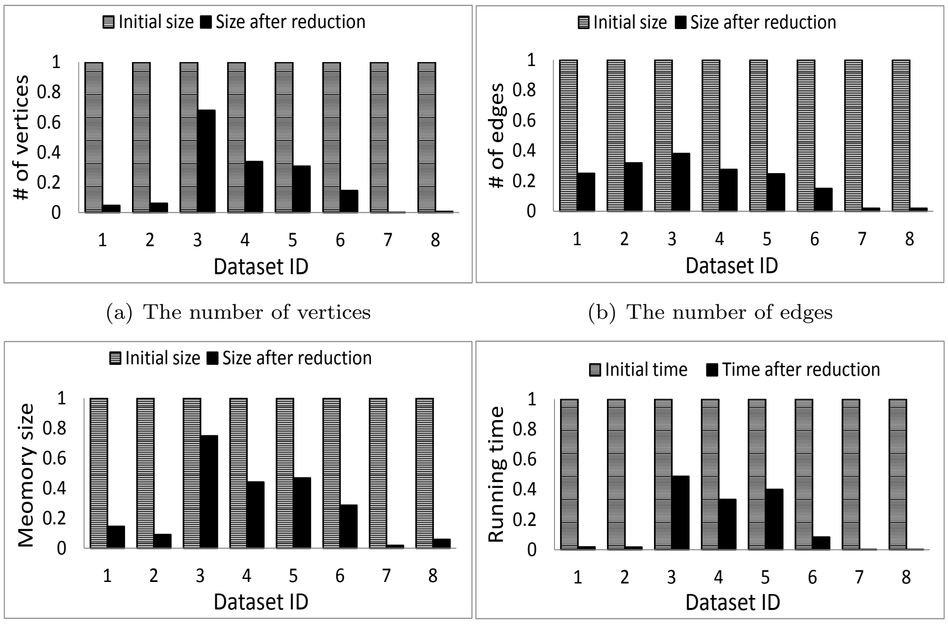
We randomly select a specific vertex subset from each dataset, and then use Algorithm 1 and *LocPreAlg* to find the LCDS of the selected vertex subset. Since both algorithms are not suitable for large datasets, we conduct experiments on datasets in Table 1. We repeated the experiment on each dataset for 100 times and calculated the percentage of connected subgraphs in the 100 returned subgraphs.

Figure 4(a) shows ratio of the percentage of connected subgraphs in *LocPreAlg* compared to Algorithm 1. For the previous algorithm, only 13% (for average) of the outcomes are connected, while for Algorithm 1, 100% of the outcomes are connected. We further analyze the 13% of the connected outcomes of the previous algorithm. We find that it is connected just because the specific vertex subset only contains one vertex, which is contained in the densest subgraph of the initial graph. For other outcomes of *LocPreAlg*, they are disconnected in order to increase the density. This results show that Algorithm 1 can guarantee the connectivity of the returned graphs for the LCDS problem.

Since Algorithm 1 and *LocPreAlg* both require to store all the data into the memory, the memory usages for both algorithms are same. Figure 4(b) shows the ratio of the time used by *LocPreAlg* compared to Algorithm 1. We see that *LocPreAlg* is far more efficient than Algorithm 1. The reason is because that we apply a simple min-cut algorithm [26] for solving the TCPM problem with time complexity $O(|V||E| + |V|^2 log|V|)$, while *LocPreAlg* applied a push relabeled algorithm [9] for solving the min-cut max-flow problem with time complexity $O(|V||E|log(|V|^2/|E|))$. This is the cost to guarantee the connectivity of the returned subgraphs. We will improve the efficiency of Algorithm 1 in our future work.

5.2. *Comparison of Basic and Improved GSDS Algorithms*

We then show the effectiveness of Algorithm 3 compared to Algorithm 2. Recall that Algorithm 3 removes unnecessary edges and vertices. Table 3 shows the comparisons of the number of vertices, the number of edges, and the memory size of the datasets before and after the reduction. In order to show the reduction performances clearly, we draw Figures 5(a), (b) and (c) that show the ratio of the number of vertices,



(a) The number of vertices (b) The number of edges

(c) The memory size (d) The running time

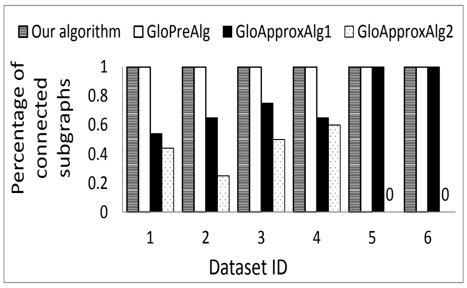
Fig. 5. The performance of the improved GSDS algorithm compared to the basic GSDS algorithm

the number of edges and the memory in Algorithm 3 compared to Algorithm 2. From the table and figures, we see that the reduction in Algorithm 3 is significant. Especially for the big dataset 7 and big dataset 8 which have a GB level data size, the number of vertices and edges, and memory occupation are reduced to about 2% (for average) of the initial size, which makes it possible to run the polynomial algorithm on one PC.

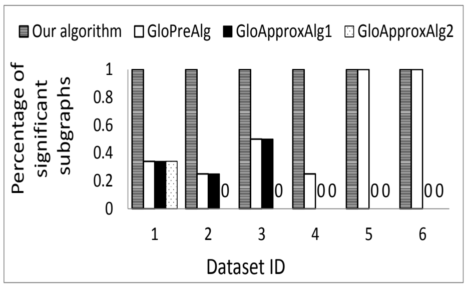
Figure 5(d) shows the percentage of running time of Algorithm 2 compared to Algorithm 3. Unsurprisingly, the running time of Algorithm 3 is much faster than Algorithm 2 since the data sizes after reduction are much smaller than the initial sizes due to the significant size reduction in Algorithm 3. Since dataset 7 and dataset 8 are too large to be computed by Algorithm 2, we only estimate the results based on the theoretical time complexity. We see the actual running time of Algorithm 3 are only 0.01% of the estimated running time of Algorithm 2.

5.3. *Performance Evaluation of the Improved GSDS Algorithm*

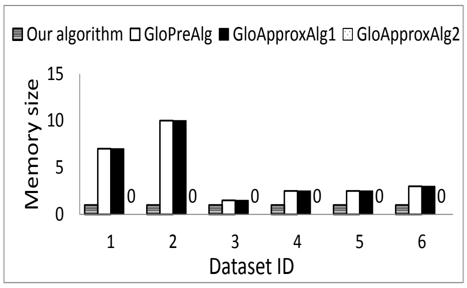
In this section, we compare the performance of discovering GSDSs between Algorithm 3 and *GloPreAlg*, *GloAppxAlg1* and *GloAppxAlg2*. First, we use Algorithm 3 to discover all the GSDSs. Then, we use *GloPreAlg*, *GloAppxAlg1* and *GloAppx-Alg2* to find the same number of densest subgraphs by recursively running these algorithms in the remaining graph. Then, we check whether these dense subgraphs are connected or significant. We repeat 100 experiments on each dataset in Table 1.



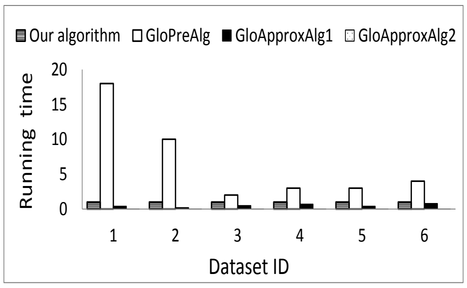
(a) Discovering connected subgraphs



(b) Discovering significant subgraphs



(c) The memory size



(d) The running time

Fig. 6. The performance of the improved GSDS algorithm compared to previous algorithms

Figure 6(a) shows the percentage of connected subgraphs of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We find that for *GloAppxAlg1* and *GloAppxAlg2*, there are a lot of disconnected subgraphs in the results, while for Algorithm 3, all the output subgraphs are connected. Also, we find *GloAppxAlg1* performs a little better than *GloAppxAlg2* since *GloAppxAlg1* greedily searches the densest subgraph by deleting the vertices one by one, while *GloAppxAlg2* greedily searches the densest subgraph by deleting the vertices batch by batch. *GloPreAlg* does not have the connectivity problem since it is a precise algorithm. The results confirm that *GloAppxAlg1* and *GloAppxAlg2* neglect the connectivity problem and Algorithm 3 can solve it.

Figure 6(b) shows the percentage of significant subgraphs in *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We find that for *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg*, there are a lot of insignificant subgraphs in the results, while for Algorithm 3, all the output subgraphs are significant. Also, similar as the connectivity comparison results, we find *GloAppxAlg1* performs a little better than *GloAppxAlg2*. Although *GloPreAlg* is a precise algorithm and does not have the connectivity problem, it still cannot guarantee that its discovered subgraphs are significant. The results confirm that *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* cannot guarantee that the discovered subgraphs are significant and Algorithm 3 can solve it.

Figure 6(c) shows the ratios of memory sizes of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. We see that Algorithm 3 needs much small-

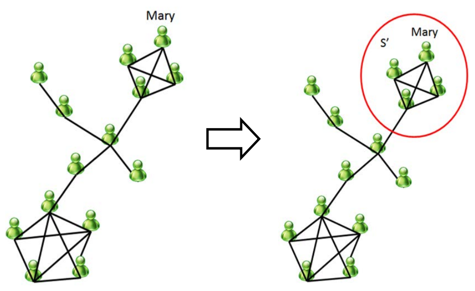


Fig. 7. Example of task 1

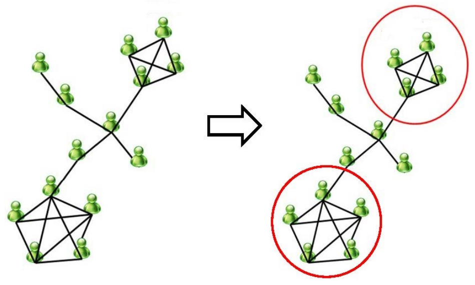


Fig. 8. Example of task 2

er memory size than *GloPreAlg* and *GloAppxAlg1* since it significantly reduces the data size. *GloAppxAlg2* almost does not need any memory since it is a streaming algorithm. Figure 6(d) shows the ratio of running time of *GloAppxAlg1*, *GloAppxAlg2* and *GloPreAlg* compared to Algorithm 3. Algorithm 3 is much faster than *GloPreAlg*, while *GloAppxAlg1* and *GloAppxAlg2* are much faster than Algorithm 3. This is because Algorithm 3 is precise algorithm but *GloAppxAlg1* and *GloAppxAlg2* are approximate algorithms applying a greedy strategy. Although *GloAppxAlg1* is more time efficient and *GloAppxAlg2* is both more time and memory efficient than Algorithm 3, both the algorithms cannot guarantee that the returned subgraphs are connected and significant as shown previously. Algorithm 3 can handle these problems with the capability of handling big data.

6. Applications and comparison with the general community detection algorithm

There are many works which study the community detection. For example, Newman proposed a community detection algorithm which has good performances on many datasets. An improved version of this algorithm can be used for large natural graphs. However, all these algorithms are proposed to detect general communities in graphs. However, when it comes to the real applications of data mining, advertisement recommendation and so on, the function of detecting general communities in graphs cannot satisfy their demands.

In order to prove that our algorithms have better performances than general community detection algorithm on different applications, in this section, firstly, we propose two data mining tasks which are important in the real applications. Then, we apply our local and global densest subgraph discovering algorithm on the two data mining tasks, respectively. At the same time, we compare the performances of our algorithms with the general community detection algorithm [].

6.1. *Specific Community Mining on the Natural Graphs*

6.1.1. *Task description*

The task of the first application is mining the significant subgraphs from natural graphs. The task of the second application is to find a specific node's community by local densest subgraph discovering algorithm.

For example as shown in Figure 7, this is a social network. Usually we hope to find all the communities. However, suppose there is a person Mary. Now, in order to recommend Mary with advertisements which she may be interested in, we hope to find the community Mary is in. Then, based on information of Mary's community, we can have a deeper understanding of Mary's interests.

In this task, we select Dataset 1 as our test dataset and randomly generate a specific node 50 times. In each time, we try to find the community which includes the specific node by LCDS algorithm and general community detection algorithm separately. For our algorithms, we can directly choose local densest subgraph discovering algorithm to find Mary's community. For the general community detection algorithm, we have to follow the following steps to find Mary's community since the general community detection algorithm is not designed directly for such a task:

(1) There are many factors which can influence the performances of different routing methods. It's impossible for us to design corresponding strategy for each of them. Furthermore, even if we can list as many as factors and design corresponding strategy for each of them, it still will not be the best way since the situation for each vehicle pair is a combination of different factors.

Finally we compare the performances of LCDS algorithm and general community detection algorithm by the density of the communities found and the time efficiency of the algorithms.

6.1.2. *Result and analysis*

First, we compare the density of the community we find by LCDS algorithm and general community detection algorithm as shown in Figure 7. From Figure 7, we can see that since LCDS algorithm is designed to find the densest community which includes the specific node and ignores the rest of the graph, LCDS algorithm can always find a denser community which includes the specific node than general community detection algorithm. On the contrary, general community detection algorithm focuses on all the communities at the same time and lack the capability to pay more attention to a community which includes a specific node. Then, we compare the time efficiency of the two algorithms as shown in Figure 7. From Figure 7, we can see that since we reduced most of the nodes in the reduction process, our algorithm can be much faster than the general community detection algorithm.

6.1.3. *Motivation to apply LCDS algorithm*

From the simple analysis above, we find that the LCDS problem is different from traditional community detecting in that LCDS algorithm focuses on the specific part of the graph which is defined based on the real demands and therefore, can perform better than general community detection algorithm on finding a particular community. On the contrary, the traditional community detecting methods [14, 24, 27] (e.g. modularity) return all the subgraphs and lack the capacity to purify the significant parts. Therefore, GSDS can be treated as a good complement in real applications when there is a desire to focus on the core parts of the graph. For example, we can apply GSDS to detect important academic circles in a research field, major metropolitan areas of a country, important functional module groups in a protein interaction network and so on.

6.2. *Significant Subgraph Mining on the Natural Graphs*

6.2.1. *Task description*

Instead of focusing on every trivial detail of the graph, in reality, we only want to focus on the important parts of the graph and study the features of the important parts when we study the graph from a macro-scope. The trivial parts of the graph may only bring the noise into the analysis. For example as shown in Figure 8, we only focus on the two completed connected subgraphs in the graph since the graph mainly consists of two completed connected subgraphs and the rest parts are trivial.

Therefore, in this section, our task is mining the important parts of natural graphs and analyze their physical significance of Datasets 1 to 6.

6.2.2. *Result and analysis*

Table 4 shows the actual number of GSDSs discovered by our algorithm. Datasets 1 and 2 are paper collaboration networks from the categories of gr-qc and hep-th in ArXiv, respectively. For the papers in each category, they have sub-categories. Table 5 shows the top-10 most frequent sub-categories of 1000 randomly selected papers in each of the categories gr-qc and hep-th, which correspond to datasets 1 and 2, respectively. We find that the number of GSDSs in datasets 1 and 2 can precisely reflect the number of the most frequent sub-categories (emphasized with bold type) of their corresponding categories. Dataset 3 is the power grid network of western states. Interestingly, there are just 31 major metropolitan areas in western states [5], which is very close to the number of GSDSs (28) in dataset 3. The protein network in dataset 4 has 13 GSDSs, while there should be hundreds of functional modules in the network [8]. The massive functional modules are highly clustered into several GSDSs separately. We are interested in the reasons behind such a clustering of functional modules to different GSDSs. The email network (dataset 5) from University Rovira i Virgili and metabolic network (dataset 6) from C.elegans both only have 1 GSDS, while there are 13 and 10 modularities [10] in each of the datasets, respectively. which means they are highly centralized to single important circle (e.g. a university or a simple organism).

ID	# of GSDS	# of modularities	ID	# of GSDS	# of modularities
Dataset 1	3	17	Dataset 4	13	38
Dataset 2	4	24	Dataset 5	1	13
Dataset 3	28	54	Dataset 6	1	10

Dataset 1		Dataset 2	
Sub-category	Frequency	Sub-category	Frequency
hep-th	**327**	**hep-ph**	**184**
astro-ph	**304**	**gr-qc**	**167**
math-ph	**175**	**math-ph**	**91**
quant-ph	31	**astro-ph**	**85**
cond-mat	28	hep-lat	39
physics.atom-ph	27	math.DG	35
stat-mech	11	nlin.SI	21
nucl-th	7	nucl-th	19

6.2.3. *Motivation to apply GSDS algorithm*

From the simple analysis above, we find that the GSDS problem is different from traditional community detecting in that GSDS problem naturally ignore the unimportant parts and focus on the important parts of the graph (while the concepts of important parts may depend on the specific knowledge from the corresponding domains). On the contrary, the traditional community detecting methods [14,24,27] (e.g. modularity) return all the subgraphs and lack the capacity to purify the significant parts. Therefore, GSDS can be treated as a good complement in real applications when there is a desire to focus on the core parts of the graph. For example, we can apply GSDS to detect important academic circles in a research field, major metropolitan areas of a country, important functional module groups in a protein interaction network and so on.

7. Related Work

The densest subgraph problem was first formally introduced by Goldberg [14]. He gave an algorithm that requires $O(\log n)$ running time (n is the number of vertexes in the graph) to find the optimal solution by reducing the problem to a series of min-cut max-flow computations. Later on, different subproblems of the dense subgraph problem were proposed. Feige *et al.* [11] defined and studied the densest k-subgraph problem, which is to find a subgraph with the maximum density among subgraphs containing k vertices. Asahiro *et al.* [2] defined and studied the problem of discovering a k-vertex subgraph of a given graph G that has at least $f(k)$ edges. Saha *et al.* [25] defined the densest subgraph problems with a distance restriction or a specific subset restriction, and provided algorithms for these subproblems. However, this work neglects the connectivity of the returned graphs.

Various approximate and heuristic algorithms have been proposed to improve the time and space complexity of the initial algorithms for big data. Charikar [6] presented a simple greedy algorithm that leads to a 2-approximation to the optimum. This algorithm was improved by Bahmani *et al.* [3] in a MapReduce framework, which can lead to a $2(1+\varepsilon)$-approximation of the optimum. The most important problem in these previous algorithms [3, 6] is that they neglect the connectivity of the returned densest subgraph. Some heuristic algorithms [7, 13] for discovering dense subgraphs were also proposed based on different techniques such as shingling and matrix blocking.

The applications of dense subgraph problem are accompanied by theoretical works. Kumar *et al.* [20] proposed an approach to identify web page communities in the Internet based on dense subgraphs. Gibson *et al.* [13] applied the solution for discovering dense subgraphs problem to detect the link spam in World Wide Web. These works use a threshold to determine the returned subgraphs. These algorithms also neglect the connectivity problem of the detected subgraphs. Also, there are no previous works that find all dense subgraphs which do not contain denser subgraphs or are contained in denser subgraphs, which however are needed in many applications.

In addition to the neglect of the connectivity, there has been no previous works that find all significant dense subgraphs. Compared to previous works, our study of dense subgraphs is novel in that i) we define two new subproblems of the dense subgraph problem, which consider the connectivity of the outputs, and find all significant dense subgraphs, and ii) our algorithms can be easily applied for handling GB-level natural graphs with no approximations in one PC with high time and memory efficiency.

8. Conclusion

In this paper, we revealed two problems existing in previous studies, which are important in various applications in different domains. One problem is the neglect of the connectivity of returned subgraphs in previous approximate and precise algorithms for finding the densest subgraph without and with the restriction of containing a vertex subset, respectively. The other problem is the lack of an algorithm for finding multiple connected and significant dense subgraphs. To handle these problems, we defined two subproblems of discovering dense subgraphs: the LCDS and GSDS problems, and proposed algorithms to solve the problems in polynomial time. Also, based on the feature of natural graphs, we provided an improved algorithm to reduce the time and space complexity of the basic GSDS algorithm, which can easily handle data with GB-level size in one PC. In the experiments, we applied our algorithms on massive natural graphs, evaluated the efficiencies of our algorithms in comparison with previous algorithms, and analyzed the structure of these natural graphs. The experimental results showed the high effectiveness and efficiency of our algorithms in solving the problems. In the future, we will focus on improving the time complexity of the algorithms. Also, we will find more evidences to reveal the physical significance of GSDSs in natural graphs from different domains.

References

1. Xin Ai, Vikram Srinivasan, and Chen-Khong Tham. Wi-sh: A simple, robust credit based wi-fi community network. In *Proc. of INFOCOM*, 2009.
2. Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 2002.
3. Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest Subgraph in Streaming and MapReduce. *PVLDB*, 2012.
4. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 1999.
5. United States Census Bureau. Annual estimates of the population of metropolitan and micropolitan statistical areas. *2011 Population Estimates*, 2012.
6. Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. *APPROX*, 2000.
7. Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Trans. Knowl. Data Eng.*, 2012.
8. Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 2006.
9. Boris V. Cherkassky and Andrew V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 1997.
10. J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 2005.
11. Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 2001.
12. Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *Proc. of KDD*. 2000.
13. David Gibson, Ravi Kumar, and Andrew Tomkins. Pregel: a system for large-scale graph processing. In *Proc. of VLDB*, 2005.
14. A. V. Goldberg. Finding a maximum subgraph. *Technical Report*, 1984.
15. R. Guimer, L. Danon, A. Daz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 2003.
16. Cunqing Hua and Rong Zheng. Starvation modeling and identification in dense 802.11 wireless community networks. In *Proc. of INFOCOM*, 2008.
17. H. Jeong, S.P. Mason, A.L. Barabasi, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 2001.
18. David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 1996.
19. Samir. Khuller and Barna Saha. On finding dense subgraphs. *ICALP*, 2009.
20. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *Proc. of WWW*, 1999.
21. Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. of KDD*, 2005.
22. Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 2007.
23. S. Thomas McCormick, M. Rammohan Rao, and Giovanni Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming*, 2003.
24. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 2004.
25. Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Proc. of RECOMB*, 2010.

26. Mechthild Stoer and Frank Wagner. A simple min cut algorithm. *ESA*, 1994.
27. Li Wan, Bin Wu, Nan Du, Qi Ye, and Ping Chen. A new algorithm for enumerating all maximal cliques in complex network. In *ADMA*, 2006.
28. D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.
29. Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *CoRR*, 2012.