# Deconstruct Densest Subgraphs

Lijun Chang
The University of Sydney
lijun.chang@sydney.edu.au

Miao Qiao
The University of Auckland
miao.qiao@auckland.ac.nz

## ABSTRACT

In this paper, we aim to understand the distribution of the densest subgraphs of a given graph under the density notion of average-degree. We show that the structures, the relationships and the distributions of all the densest subgraphs of a graph $G$ can be encoded in $O(L)$ space in an index called the ds-Index. Here $L$ denotes the maximum output size of a densest subgraph of $G$. More importantly, ds-Index can report all the minimal densest subgraphs of $G$ collectively in $O(L)$ time and can enumerate all the densest subgraphs of $G$ with an $O(L)$ delay. Besides, the construction of ds-Index costs no more than finding a single densest subgraph using the state-of-the-art approach. Our empirical study shows that for web-scale graphs with one billion edges, the ds-Index can be constructed in several minutes on an ordinary commercial machine.

## CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Information systems → Web mining**.

## KEYWORDS

Densest Subgraph, Graph Analytics, Query Processing

**ACM Reference Format:**
Lijun Chang and Miao Qiao. 2020. Deconstruct Densest Subgraphs. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3366423.3380033

## 1 INTRODUCTION

The dense subgraphs of a given graph are the subgraphs that maximize some notion of density. They are found to be semantically significant in real graphs in various domains. For example, dense subgraphs can be communities in social networks [17], expert teams in co-authorship graphs [9], spam links in web graphs [7, 21] and stories in social media [3]. The primary role of dense subgraphs in graph analytics has yielded extensive studies on the problem of dense subgraphs identification (see [23] and references therein).

A widely adopted notion of graph density is the *average-degree density* [5, 9, 14, 22], which is also used throughout this paper. It is defined as the ratio of the number of edges to the number of nodes in the graph. Given a graph $G$, the maximum density $\rho^*(G)$ is the largest density over all subgraphs of $G$. The research on dense subgraphs identification initially focuses on Finding a Single Densest Subgraph (FSDS) [9, 14, 22]. A solution to FSDS can terminate as soon as any subgraph $g$ of $G$ whose density $\rho(g)$ equals $\rho^*(G)$ is found. The state-of-the-art algorithm finds a densest subgraph of an undirected graph $G$ with $n$ nodes and $m$ edges in $O(nm\log(n^2/m))$ time and $O(m)$ space. This time complexity is achieved by applying the push-relabel algorithm [20] to a parametric flow network formulation of FSDS [22] (see Section 2.2 for a detailed explanation).

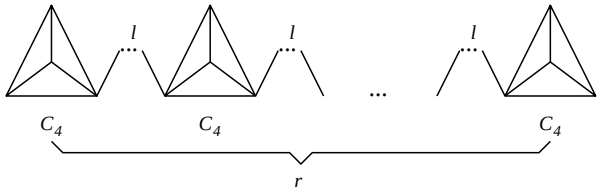


**Figure 1: A graph $G_{l,r}$ lines $r$ 4-cliques $C_4$ up with $l$ nodes between each pair of consecutive cliques. $G_{l,r}$ has $n = 4r + l(r-1)$ nodes and $m = 6r + (l+1)(r-1)$ edges.**

The objective of dense subgraphs identification has recently been expanded from FSDS to Finding Multiple Dense Subgraphs (FMDS) [5, 31]. It is observed in [5] that a densest subgraph may contain multiple minimal densest subgraphs — a *densest* subgraph is *minimal* if it is strictly denser than all of its proper subgraphs — and the minimal densest subgraphs contain disjoint set of nodes. Thus, it will be beneficial to compute all minimal densest subgraphs in a graph when we aim to find multiple subgraphs by maximizing the aggregated density while having limited overlap [5].

The enhanced objective of dense subgraphs identification, *i.e.*, to find multiple dense subgraphs, urges us to

> 'identify many (if not all) densest subgraphs, understand their distribution in the graph, and ideally determine the relationships among them.'

A valuable probe into the structure of the densest subgraphs is made by Balalau et al. [5] who introduced the concept of minimal densest subgraph. To find *each* minimal densest subgraph of a graph with $n$ nodes, Balalau et al. proposed an algorithm that triggers $O(\log n)$ instances of FSDS. Suggested by the graph in Figure 1, the total number of minimal densest subgraphs can be proportional to $n$: when $l \geq 2$, each of the $r$ 4-cliques is a minimal densest subgraph of $G_{l,r}$. Therefore, it necessitates $O(n \log n)$ instances of FSDS to find all the minimal densest subgraphs, which is extremely expensive.

**Our Contributions.** Based on the prior work [5], we formulate the problem of deconstructing densest subgraphs as follows.

Problem 1 (Deconstruct Densest Subgraphs). *Understand the relationships of the densest subgraphs of an undirected graph $G$ under the average-degree density, and propose algorithms to*

*(1) Efficiently report all the minimal densest subgraphs of $G$;*
*(2) Efficiently report all the densest subgraphs of $G$.*

We answer Problem 1 with an index called the ds-Index. The ds-Index encodes all the inner structures of the densest subgraphs of


*WWW '20, April 20–24, 2020, Taipei, Taiwan*

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380033

$G$ in $O(L)$ space. $L$ denotes the maximum size of a densest subgraph of $G$ where the size of a graph is the summation of its number of edges and its number of nodes. With the ds-Index,

(1) all the minimal densest subgraphs of $G$ can be reported in $O(L)$ time in total, and
(2) all the densest subgraphs of $G$ can be enumerated with an $O(L)$ delay.

When $G$ has only one densest subgraph (for example, when $G$ is a clique), ds-Index is optimal to Problem 1 since the time complexity of Problem 1 is bounded by its output size which is exactly $L$. When $G$ has at least $d$ (minimal) densest subgraphs for any integer $d$, ds-Index remains optimal in the worst-case. For example, let $G$ be the graph in Figure 1 with $r \geq d$ and $l = 3$. The total number of edges in $G$ is $10r - 4$ and the maximum density of $G$ is $\frac{3}{2}$. The minimal densest subgraphs of $G$ are the $r$ 4-cliques in $G$, and thus the total number of edges in the minimal densest subgraphs is $6r$. The densest subgraphs of $G$ are the power set of the $r$ 4-cliques and thus the expected/average number of edges of a densest subgraph is $3r$. It is worth pointing out that a densest subgraph is not simply a union of some minimal densest subgraphs, *e.g.*, see Figure 2.

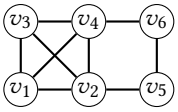


**Figure 2: The entire graph is densest, and the subgraph induced by $\{v_1, v_2, v_3, v_4\}$ is a minimal densest subgraph**

The ds-Index of a given graph $G$ provides a shortcut to *all* the (minimal) densest subgraphs of $G$. Its construction, however, costs no more than finding a *single* densest subgraph of $G$ using the state-of-the-art solution, the network-flow based solution. Besides, via a reduction of $G$ prior to the network-flow computation, web-scale graphs can be easily handled. Our empirical study shows that for a real graph with one billion edges, the ds-Index can be constructed in merely several minutes on an ordinary commercial machine.

## 2 PRELIMINARIES

The input is an undirected graph $G$. Denote by $n = |V(G)|$ and $m = |E(G)|$ the total number of nodes and edges of $G$, respectively. An undirected edge from node $u$ to node $v$ is denoted as $(u, v)$ while a directed edge $\langle u, v\rangle$. For each node $v$ in $V(G)$, denote by $d(v)$ the degree of $v$ in $G$, *i.e.*, $d(v) = |\{u \in V(G) \mid (v, u) \in E(G)\}|$. Without loss of generality, we assume that $G$ is a *connected* graph with *at least two nodes*; thus, $d(v) > 0, \forall v \in V(G)$.

### 2.1 Densest Subgraph

For any graph $g$, its node set is denoted as $V(g)$ and its edge set is denoted as $E(g)$. Its *average-degree density* is $\rho(g) = \frac{|E(g)|}{|V(g)|}$.

Definition 1 (Densest Subgraph). *A **densest subgraph** of $G$ is a subgraph $g$ of $G$ that maximizes $\rho(g)$. The **maximum density** of $G$ is $\rho^*(G) = \max_{g \text{ subgraph of } G} \rho(g)$.*

Consider a subset $S$ of $V(G)$. The *induced subgraph* of $G$ on $S$ is the graph with node set $S$ and edge set $E(S) = \{(u, v) \in E(G) \mid u, v \in S\}$. Since the induced subgraph is uniquely decided by $S$ and $G$ while $G$ is the input graph, we abuse $S$ to denote the induced subgraph of $G$ on $S$. The density of the induced subgraph $S$ is $\rho(S) = \frac{|E(S)|}{|S|}$. It is straightforward that a densest subgraph of $G$ must be an induced subgraph of $G$, *i.e.*, $\rho^*(G) = \max_{S \subseteq V(G)} \rho(S)$.

Let $S_1$ and $S_2$ be two densest subgraphs of $G$. It is known [5] that $S_1 \cup S_2$ is a densest subgraph of $G$, and $S_1 \cap S_2$ is either an empty set or a densest subgraph of $G$. Thus, the maximal densest subgraph of $G$ is unique, and the minimal densest subgraphs of $G$ are disjoint.

Definition 2 (Maximal/Minimal Densest Subgraph). *The union of all densest subgraphs of $G$ is a densest subgraph of $G$ and is called the **maximal** densest subgraph of $G$. A densest subgraph $S$ of $G$ is **minimal** if it is strictly denser than all of its proper subgraphs.*

### 2.2 Parametric Flow Network and FSDS

The state-of-the-art solution to finding a single densest subgraph (FSDS) works on flow networks [22]. Let $\lambda \geq 0$ be a parameter. The parametric **flow network**, called **$\lambda$-graph** $G_\lambda$, of $G$ is a directed graph where every edge $e$ bears a non-negative **capacity** $c(e)$. Specifically, $G_\lambda$ has $n + 2$ nodes, $V(G_\lambda) = V(G) \cup \{s, t\}$ with $s$ and $t$ being the **source node** and **target node**, respectively, and $4n + 2m$ parameterized directed edges categorized in 4 types: [1]

**One-edges** For each edge $(u, v)$ of $G$, $E(G_\lambda)$ has two edges $\langle u, v\rangle$ and $\langle v, u\rangle$, both having a capacity of 1;

**Source-edges** For each node $v$ of $G$, $E(G_\lambda)$ has an edge $\langle s, v\rangle$ bearing a capacity of $d(v)$, the degree of $v$ in $G$;

**Target-edges** For each node $v$ of $G$, $E(G_\lambda)$ has an edge $\langle v, t\rangle$ bearing a capacity of $2\lambda$;

**Zero-edges** For each node $v$ of $G$, $E(G_\lambda)$ has two edges, $\langle v, s\rangle$ and $\langle t, v\rangle$, both bearing a capacity of 0.

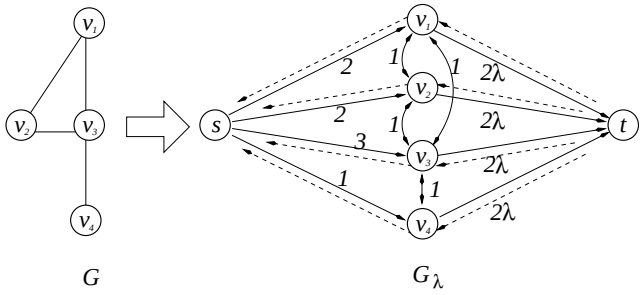


**Figure 3: An undirected graph $G$ and its $\lambda$-Graph $G_\lambda$ (zero-edges are lined in dashed arrows).**

Example 1. *Figure 3 shows the $\lambda$-graph of an undirected graph $G$. $G$ has 4 nodes and 4 edges. $G_\lambda$ has 6 nodes and 24 directed edges.*

The $\lambda$-graph of $G$ bridges the density of an induced subgraph of $G$ to the capacity of an $s$-$t$ cut of $G_\lambda$. Let $S$ be a subset of $V(G)$, and $\overline{S} = V(G) \setminus S$ be the complement of $S$. Denote by $(S \cup \{s\}, \overline{S} \cup \{t\})$ the **$s$-$t$ cut** of $G_\lambda$ that consists of all edges in $G_\lambda$ from nodes in $S \cup \{s\}$ to nodes in $\overline{S} \cup \{t\}$. The **capacity of a cut** is the summation of the capacities of edges in the cut. It can be verified that the capacity of cut $(S \cup \{s\}, \overline{S} \cup \{t\})$ is

$$c(S \cup \{s\}, \overline{S} \cup \{t\}) = \sum_{v \in \overline{S}} d(v) + 2\lambda|S| + |E(G) \cap (S \times \overline{S})|$$
$$= 2m - 2|E(S)| + 2\lambda|S| = 2m + 2(\lambda - \rho(S))|S|. \quad (1)$$

[1] This formulation slightly simplifies the one proposed by Goldberg [22].

Lemma 1. *Let $(S^* \cup \{s\}, \overline{S^*} \cup \{t\})$ be a **minimum** s-t **cut** of $G_\lambda$,* i.e.*, having the minimum capacity among all s-t cuts. Then, its capacity is decided by the relation between $\lambda$ and $\rho^*(G)$:*

*(1) If $\lambda < \rho^*(G)$ then $c(S^* \cup \{s\}, \overline{S^*} \cup \{t\}) < 2m$.*

*(2) If $\lambda \geq \rho^*(G)$ then $c(S^* \cup \{s\}, \overline{S^*} \cup \{t\}) = 2m$.*

*When $\lambda$ is close enough to $\rho^*(G)$, $S^*$ is a densest subgraph of $G$:*

*(3) If $\rho^*(G) - \frac{1}{n(n-1)} < \lambda < \rho^*(G)$ then $\rho(S^*) = \rho^*(G)$.*

The proof of Lemma 1 is largely based on Goldberg's work [22], and is given in Section A in Appendix. According to Lemma 1(3), finding a densest subgraph of $G$ boils down to finding a $\lambda$ such that the capacity of a minimum $s$-$t$ cut of $G_\lambda$ is slightly smaller than $2m$. Besides, Lemma 1(1) and (2) suggest a binary search which is terminated when $\lambda < \rho^*(G) < \lambda + \frac{1}{n(n-1)}$.

Goldberg [22] finds the desirable $\lambda$ in $\log n$ instances of the network flow computation based on the **max-flow min-cut theorem**: *the capacity of a minimum cut equals the value of a maximum flow.* Here, a **flow** of $G_\lambda$ is a mapping $f$ from each edge $e \in E(G_\lambda)$ to a real value $f(e) \in \mathbb{R}$ that satisfies three properties:

**Capacity** $f(e) \leq c(e)$ for each edge $e \in E(G_\lambda)$;

**Antisymmetry** $f(\langle v, u \rangle) = -f(\langle u, v \rangle)$ for each edge $\langle u, v \rangle$ in $G_\lambda$;

**Conservation** $\sum_{\text{edge } e \in E(G_\lambda) \text{ into } u} f(e) = \sum_{\text{edge } e \in E(G_\lambda) \text{ out } u} f(e) = 0$ for each node $u \in V(G_\lambda) \setminus \{s, t\}$.

The **value of the flow** $f$ is defined as

$$val(f) = \sum_{\text{edge } e \in E(G_\lambda) \text{ out of } s} f(e) = \sum_{\text{edge } e \in E(G_\lambda) \text{ into } t} f(e).$$

A **maximum flow** of $G_\lambda$ is a flow $f$ that maximizes its value $val(f)$.

Finally, the $\log n$ factor in the time complexity can be removed by leveraging the push-relabel algorithm [20].

Lemma 2 ([20]). *The maximum flow of $G_\lambda$ for all possible $\lambda$ can be computed in $O(nm \log(n^2/m))$ time. A densest subgraph of $G$ can be found in $O(nm \log(n^2/m))$ time.*

## 3 DECONSTRUCT DENSEST SUBGRAPHS

To find a densest subgraph, the state-of-the-art approach relies on the $\lambda$-graph of $G$ with $\lambda \in (\rho^*(G) - \frac{1}{n(n-1)}, \rho^*(G))$ (Lemma 1(3)). However, the key to the structure of all the densest subgraphs of $G$ is hidden in the $\lambda$-graph with $\lambda = \rho^*(G)$, specifically, the residual graph of this $\lambda$-graph under the maximum flow.

Denote by $\mathcal{H}$ the $\lambda$-graph of $G$ under $\lambda = \rho^*(G)$. Given any flow $f$ of $\mathcal{H}$, the **residual capacity** of an edge $e$ in $\mathcal{H}$ under $f$ is defined as $c_f(e) = c(e) - f(e)$. An edge $e$ is **saturated** if $c_f(e) = 0$. The **residual graph** of $\mathcal{H}$ under $f$ is a graph $\mathcal{H}_f$ with all nodes in $\mathcal{H}$ and all non-saturated edges, *i.e.*, $V(\mathcal{H}_f) = V(\mathcal{H})$ and $E(\mathcal{H}_f) = \{e \in E(\mathcal{H}) \mid f(e) < c(e)\}$. Let $f^*$ be a maximum flow of $\mathcal{H}$. We have the following lemma, whose proof is in Section B in Appendix.

Lemma 3. *Let $S$ be a non-empty subset of $V(G)$, and $\overline{S}$ be $V(G) \setminus S$. The following statements are equivalent.*

*(1) $S$ is a densest subgraph of $G$;*

*(2) $(S \cup \{s\}, \overline{S} \cup \{t\})$ is a minimum s-t cut of $\mathcal{H}$;*

*(3) There is no edge from $S \cup \{s\}$ to $\overline{S} \cup \{t\}$ in $\mathcal{H}_{f^*}$.*

We call $\mathcal{H}_{f^*}$ the **critical residual graph**. To enumerate all densest subgraphs by following Lemma 3, we decompose $\mathcal{H}_{f^*}$, by treating it as a simple directed graph, into strongly connected components (**SCC**s). Denote by $\mathcal{C}(\mathcal{H}_{f^*})$ the **component set** (*i.e.*, the set of all SCCs) of $\mathcal{H}_{f^*}$. For each node $v \in V(\mathcal{H}_{f^*})$, denote by $scc(v)$ the unique SCC in $\mathcal{C}(\mathcal{H}_{f^*})$ that contains $v$. By contracting each SCC of $\mathcal{H}_{f^*}$ into a super-node, we obtain the **critical component graph**, denoted as $\mathcal{H}^C$, which is a directed acyclic graph (DAG).

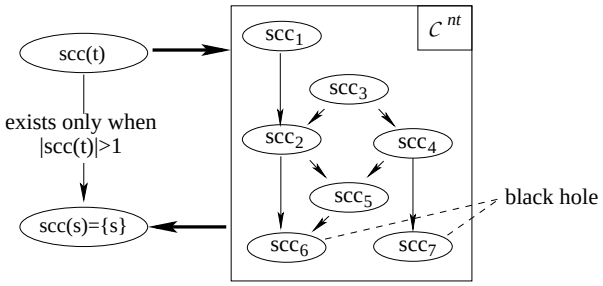


**Figure 4: The critical component graph $\mathcal{H}^C$**

The critical component graph $\mathcal{H}^C$ is conceptually sketched in Figure 4, and is formally analyzed in Section D in Appendix. We further introduce the notion of "non-trivial" to the components. Specifically, a component $C$ of $\mathcal{H}_{f^*}$ is **non-trivial** if $C$ includes neither the source node $s$ nor the target node $t$, *i.e.*, $C \notin \{scc(s), scc(t)\}$. The set of all non-trivial components, called **non-trivial component set**, is denoted as $\mathcal{C}^{nt} = \mathcal{C}(\mathcal{H}_{f^*}) \setminus \{scc(s), scc(t)\}$. The set of all nodes of $G$ in non-trivial components, called the **non-trivial node set**, is denoted as $S^{nt} = \cup_{C \in \mathcal{C}^{nt}} C$. Now, we are ready to define the ds-Index for indexing densest subgraphs.

Definition 3. *The* ds-Index *of $G$ records*

- *The non-trivial components $\mathcal{C}^{nt}$ of critical residual graph $\mathcal{H}_{f^*}$;*
- *The subgraph of the critical component graph $\mathcal{H}^C$ on $\mathcal{C}^{nt}$;*
- *The induced subgraph of $G$ on the non-trivial node set $S^{nt}$.*

For example, the ds-Index for the critical component graph in Figure 4 focuses on the $\mathcal{H}^C$ in the scope of the rectangle and the corresponding part of the underlying critical residual graph.

Denote by $L$ the size, the summation of the total number of nodes and the total number of edges, of the maximal densest subgraph of $G$. Note that $L \leq m + n$. Theorem 1 shows the space and time complexity of enumerating all the (minimal) densest subgraphs using the ds-Index.

Theorem 1. *For a given graph $G$ whose maximal densest subgraph is of size $L$, the* ds-Index *takes $O(L)$ space. With the* ds-Index*,*

- *all densest subgraphs of $G$ can be enumerated in $O(L)$ delay,*
- *all minimal densest subgraphs of $G$ can be computed in $O(L)$ time in total.*

We prove Theorem 1 in three parts in the following three subsections: the space complexity of ds-Index, the time complexity of enumerating all densest subgraphs and that of enumerating all minimal densest subgraphs of $G$.

### 3.1 Space Complexity of ds-Index

The size of ds-Index is dominated by the size of the induced subgraph of $G$ on $S^{nt}$. This size, as Theorem 2 shows, however, is the very size $O(L)$ of the maximal densest subgraph of $G$.

Theorem 2 (Maximal Densest Subgraph). *The non-trivial node set $S^{nt}$ is the maximal densest subgraph of $G$.*

*The Proof of Theorem 2.* We start with proving the following two properties of the critical residual graph $\mathcal{H}_{f^*}$.

Lemma 4. *For any non-source node $v \in V(\mathcal{H}_{f^*}) \setminus \{s\}$, the critical residual graph $\mathcal{H}_{f^*}$ has no edge from the source node $s$ to $v$,* i.e., *$\langle s, v\rangle \notin E(\mathcal{H}_{f^*})$. For any node $u \in V(\mathcal{H}_{f^*})\setminus\{s,t\}$, the critical residual graph has an edge from $u$ to $s$,* i.e., *$\langle u, s\rangle \in E(\mathcal{H}_{f^*})$.*

Proof. Recall that in the $\lambda$-graph $\mathcal{H}$, $s$ and $t$ have no edges between them, therefore, $t$ has no edge from/to $s$ in the critical residual graph $\mathcal{H}_{f^*}$. Thus, it suffices to consider nodes in $V(\mathcal{H}_{f^*}) \setminus \{s,t\} = V(G)$. Let $v$ be an arbitrary node in $V(G)$. Based on the assumption in Section 2.1 that $G$ is a connected graph with at least two nodes, the degree of $v$ has $d(v) > 0$ in $G$. Besides, the cut $(\{s\}, V(G) \cup \{t\})$ is a minimum $s$-$t$ cut since (i) its capacity is $2m$ and (ii) the capacity of a minimum $s$-$t$ cut of $\mathcal{H}$ is $2m$ (Lemma 1(2)). By following the proof of Lemma 3, we know that $f^*(\langle s, v\rangle) = c(\langle s, v\rangle) = d(v) > 0$. According to the antisymmetry property of a flow, edge $\langle v, s\rangle$ has therefore, $f^*(\langle v, s\rangle) = -f^*(\langle s, v\rangle) = -d(v) < 0 = c(\langle v, s\rangle)$. Thus, $\langle s, v\rangle$ is saturated and $\langle v, s\rangle$ is unsaturated under $f^*$, and $\langle s, v\rangle \notin E(\mathcal{H}_{f^*})$ and $\langle v, s\rangle \in E(\mathcal{H}_{f^*})$. □

Lemma 5. *For every node $v \in V(\mathcal{H}_{f^*}) \setminus \{s,t\} = V(G)$, there is a path from $t$ to $v$ in the critical residual graph $\mathcal{H}_{f^*}$.*

Proof. The general idea of proving Lemma 5 is that there must exist a directed path from $s$ to $v$ and then to $t$ in $\mathcal{H}$ such that $f^*(e) > 0$ for every edge $e$ in the path. This is because Lemma 4 states that $f^*(\langle s, u\rangle) > 0$ for every $u \in V(\mathcal{H}_{f^*} \setminus \{s,t\}$. The full proof of Lemma 5 is in Section C in Appendix. □

To prove Theorem 2, we first prove that $S^{nt}$ is a densest subgraph of $G$. Firstly, according to Lemma 4, $(\{s\} \cup S^{nt}, scc(t))$ is an $s$-$t$ cut of $\mathcal{H}$, as $scc(s) = \{s\}$. Secondly, according to Lemma 5, there is no edge in the residual graph $\mathcal{H}_{f^*}$ from a node in $\{s\} \cup S^{nt}$ to a node in $scc(t)$. Therefore, according to Lemma 3, $(\{s\} \cup S^{nt}, scc(t))$ is a minimum $s$-$t$ cut of $\mathcal{H}$, and $S^{nt}$ is a densest subgraph of $G$.

Now we prove that, for any node $v \in scc(t)$ with $v \neq t$, there is no densest subgraph of $G$ containing $v$. Suppose $v$ is in a densest subgraph $S$ of $G$. Then, there is no edge from $\{s\} \cup S$ to $\overline{S} \cup \{t\}$ in $\mathcal{H}_{f^*}$ (Lemma 3); this contracts the facts that $v \in S$ and $v \in scc(t)$. Thus, $v$ does not belong to any densest subgraph of $G$. Therefore, $S^{nt}$ is the maximal densest subgraph of $G$. This completes the proof of Theorem 2. ■

## 3.2 Enumerate All Densest Subgraphs

To enumerate all densest subgraphs, we define the non-trivial descendants and ancestors of a component and introduce the independence among components.

Definition 4 (nt-descendant and nt-ancestor). *For a non-trivial component $C$, the set of non-trivial descendants of $C$ is $des^{nt}(C) = \{C' \in \mathcal{C}^{nt} \setminus \{C\} \mid C \rightsquigarrow C' \text{ in } \mathcal{H}^C\}$ where $C \rightsquigarrow C'$ denotes that there exists a directed path in $\mathcal{H}^C$ from $C$ to $C'$. The set of non-trivial ancestors of $C$ is $anc^{nt}(C) = \{C' \in \mathcal{C}^{nt} \setminus \{C\} \mid C' \rightsquigarrow C \text{ in } \mathcal{H}^C\}$.*

Definition 5 (Independent Component Set). *Two distinct non-trivial components $C_1$ and $C_2$ are independent if $C_1 \notin des^{nt}(C_2)$ and $C_2 \notin des^{nt}(C_1)$. A non-trivial component set $Z \subseteq \mathcal{C}^{nt}$ is independent if every pair of the components in $Z$ are independent.*

For example, in Figure 4, the nt-descendants of $scc_1$ are $scc_2$, $scc_5$, and $scc_6$; the nt-ancestor of $scc_4$ is $scc_3$. $\{scc_2, scc_4\}$ is independent, while $\{scc_1, scc_6\}$ is not. Denote by $des^{nt}(Z)$ the union of the non-trivial descendants of the components in $Z$ for a subset $Z$ of $\mathcal{C}^{nt}$, *i.e.*, $des^{nt}(Z) = \cup_{C\in Z} des^{nt}(C)$. Then, we have the following theorem for enumerating all densest subgraphs.

Theorem 3. *For a graph $G$, by enumerating all the independent component sets $Z \subseteq \mathcal{C}^{nt}$ and outputting the induced subgraph of $G$ with nodes in components in $Z \cup des^{nt}(Z)$,* i.e., *$\cup_{C\in Z\cup des^{nt}(Z)} C$, one can enumerate every densest subgraph of $G$ exactly once.*

*The Proof of Theorem 3.* To enumerate all the densest subgraphs, we show that a graph is a densest subgraph if and only if it can be disjointly partitioned into components in $Z \cup des^{nt}(Z)$ of an independent component set $Z$.

We call a subset $Z$ of $\mathcal{C}^{nt}$ as **non-trivial descendant closed (d-closed)** if $des^{nt}(Z) \subseteq Z$. We first prove that there is a bijection between densest subgraphs and d-closed component sets.

Lemma 6. *For any d-closed subset $Z$ of $\mathcal{C}^{nt}$, $S = \cup_{C\in Z} C$ is a densest subgraph of $G$.*

Proof. A d-closed subset $Z$ has no edge, in the critical component graph $\mathcal{H}^C$, to $\mathcal{C}^{nt} \setminus Z$. According to Lemmas 3, 4 and 5, $S$ is a densest subgraph of $G$. □

Lemma 7. *For any non-trivial component $C$ of the critical component graph $\mathcal{H}^C$ of $G$, a densest subgraph $S$ of $G$ contains either none or all of the nodes in the component $C$.*

Proof. This directly follows from Lemma 3 and the definition of strongly connected component. □

Following Lemma 3 and 7, for every densest subgraph $S$ of $G$, there is a d-closed subset $Z$ of $\mathcal{C}^{nt}$ such that $S = \cup_{C\in Z} C$. It remains to build up the bijection between the independent component sets and the d-closed component sets.

Lemma 8. *For each d-closed component set $Z'$ there is a unique independent component set $Z$ such that $Z' = Z \cup des^{nt}(Z)$.*

Proof. Let $Z$ be the set of components in $Z'$ with no incoming edges in $\mathcal{H}^C$ from any component in $Z'$. $Z$ is uniquely determined by $Z'$. Now we prove that $Z$ is independent by contradiction. Suppose there exist $C_1, C_2 \in Z \subseteq Z'$ and $C_1 \in des^{nt}(C_2)$. Then there is a directed path from $C_2$ to $C_1$ in $\mathcal{H}^C$. Let $C'$ be the immediate predecessor of $C_1$ on the path. Since $Z'$ is d-closed, $C' \in Z'$. The edge from $C'$ to $C_1$ contradicts the construction of $Z$. □

By concatenating the above two bijections, we can non-repeatedly enumerate all densest subgraphs by enumerating all independent component subsets. This completes the proof of Theorem 3. ■

Following Theorem 3, all the densest subgraphs of $G$ can be non-repeatedly enumerated with an $O(L)$ delay by calling $\mathsf{EnumAll}(\emptyset, \mathcal{C}^{nt})$ in Algorithm 1. The algorithm recursively selects the independent component set $Z_1$ while keeping, in $Z_2$, the components that are independent with all the components of $Z_1$. When a component $C$ is moved from $Z_2$ to $Z_1$, all the descendants and ancestors of $C$ and $C$ itself are removed from $Z_2$ and then a recursion is invoked. Note that every recursion (Line 5) costs $O(L)$ time and produce one

**Algorithm 1:** EnumAll

**Input:** Two sets of non-trivial components $Z_1$ and $Z_2$
**Output:** All densest subgraphs $S$ with $S \subseteq \bigcup_{C' \in des^{nt}(Z_1 \cup Z_2) \cup Z_1 \cup Z_2} C'$

1 **if** $Z_1 \neq \emptyset$ **then** Output the induced subgraph of $\bigcup_{C' \in Z_1 \cup des^{nt}(Z_1)} C'$;
2 **if** $Z_2 \neq \emptyset$ **then**
3  **for** *each component* $C$ *in* $Z_2$ **do**
4  $Z_2 \leftarrow Z_2 \setminus \{C\}$;
5  EnumAll $(Z_1 \cup \{C\}, Z_2 \setminus des^{nt}(C) \setminus anc^{nt}(C))$;

densest subgraph (Line 1), therefore, Algorithm 1 pays $O(L)$ time for each densest subgraph of $G$.

### 3.3 Enumerate Minimal Densest Subgraphs

To explain how to use ds-Index to enumerate all the minimal densest subgraphs of $G$, we specialize a type of components in $\mathcal{C}^{nt}$.

Definition 6 (Black Hole Component). *A non-trivial component is a black hole component if it has no outgoing edges in the critical component graph to any other non-trivial components.*

For example, the critical component graph in Figure 4 has two black hole components, $scc_6$ and $scc_7$.

Theorem 4 (Minimal Densest Subgraph). *Let $S$ be a subset of $V(G)$. $S$ is a minimal densest subgraph of $G$ if and only if $S$ is a black hole component of the critical component graph $\mathcal{H}^C$ of $G$.*

Proof. This theorem directly follows from Theorem 3 and the definition of minimal densest subgraph. □

According to Theorem 4, the black hole components are the minimal densest subgraphs of $G$, which can be reported by ds-Index in $O(L)$ total time.

### 3.4 Scale the Index Construction

To improve the practical efficiency of ds-Index construction, we propose a reduction step to safely reduce the size of $G$, by inexpensively finding a supergraph of the maximal densest subgraph of $G$. The correctness of the reduction is based on the observation below.

Lemma 9. *Let $G$ be a connected graph with at least two nodes. For a densest subgraph $S$ of $G$, every node $v$ in $S$ has its degree $d_S(v)$ on $S$ no less than $\lceil \rho^*(G) \rceil$.*

The proof of Lemma 9 is given in Section E in Appendix. When a direct computation of $\rho^*(G)$ is costly, there is an inexpensive computation of the upper and lower bounds of $\rho^*(G)$.

Lemma 10 ([14]). *Let $\widetilde{\rho}$ be the highest density among the $n$ subgraphs that are generated in an iterative peeling from $G$ the node with the lowest degree. $\rho^*(G)/2 \leq \widetilde{\rho} \leq \rho^*(G)$.*

**Two-Step Construction.** Denote by $S_m$ the maximal densest subgraph of $G$. According to Lemma 9, the degree of any node in $S_m$ is no less than $\lceil \rho^*(G) \rceil$ and then no less than $\lceil \widetilde{\rho} \rceil$ (Lemma 10). Thus, we can safely reduce the graph $G$ before the flow computation using the $\lceil \widetilde{\rho} \rceil$**-core** [6]. Note that, the computation of $\widetilde{\rho}$ and the $\widetilde{\rho}$-core of $G$ takes only linear time [6]. We construct ds-Index in two steps.

**Reduction** Compute $\widetilde{\rho}$, the highest density among the $n$ subgraphs that are generated in an iterative peeling from $G$ the node with the lowest degree. Compute the $\lceil \widetilde{\rho} \rceil$-core of $G$ and replace $G$ with the $\lceil \widetilde{\rho} \rceil$-core.

**Flow** Perform the parametric network flow algorithm on the $\lambda$-graph of $G$. Generate $\rho^*(G)$, and the critical component graph. Construct the ds-Index based on Definition 3.

## 4 RELATED WORKS

The problem of dense subgraph identification has been widely studied [23]. A subgraph with the largest average degree can be computed exactly in $O(nm \log(n^2/m))$ time by parametric maximum flow [20, 22], and a 2-approximation result can be computed in linear time by iteratively removing the vertex with the smallest degree [14]. Besides, dense subgraph identification in streaming and dynamic environment is studied [4, 8, 18].

Recently, the research interests regarding dense subgraph identification have been devoted to finding multiple dense subgraphs [5, 16, 25, 29, 31]. For example, computing all *locally* densest subgraphs, which form a nested structure, is studied in [16, 29]. Here, all the subgraphs in the nested structure have different densities, and only the inner-most subgraph is a (globally) densest subgraph while other subgraphs are locally but not globally densest subgraphs (*i.e.*, their densities are smaller than $\rho^*(G)$). The problem of computing top-$k$ dense subgraphs with limited overlap is studied in [5], which iteratively computes minimal densest subgraphs and removes a certain proportion of its nodes from the input graph. As we will show, our new solution to the problem of computing all the minimal densest subgraphs significantly outperforms the existing one in [5].

Higher-order variants of the densest subgraph problem has also been studied in the literature. Tsourakakis [30] aims to maximize the average number of $h$-cliques (*i.e.*, divided by the number of nodes) in the result subgraph for a parameter $h$, which generalizes the densest subgraph problem as an edge is a 2-clique. Recently, Fang et al. [19] further generalize the problem by considering an arbitrary pattern graph, and aim to maximize the average number of occurrences of the pattern in the result subgraph. Nevertheless, the solutions of [19, 30] are still based on the parametric flow network as described in Section 2.2. Thus, our techniques can be straightforwardly extended to deconstruct the densest subgraphs for these settings.

Dense subgraph identification based on other notions of density, usually referred to as cohesive subgraph identification, has also been extensively studied [12], for example, minimum-degree based $k$-core computation [28], triangle based $k$-truss computation [15, 26, 32], edge-connectivity based $k$-edge connected component computation [2, 13], clique [11] and its relaxations (e.g., $k$-plex [27], $n$-clique [10], $n$-clans [24]).

## 5 EXPERIMENTS

We arrange the empirical study in two parts. The first part evaluates the construction efficiency of ds-Index and the second part examines the performance of plugging ds-Index as a building block into an existing solution to top-$k$ dense subgraphs identification [5].

The experiments were carried out on 22 real graphs downloaded from the Stanford Network Analysis Platform[2], the Laboratory

[2] http://snap.stanford.edu/

**Table 1: Construct ds-Index on real graphs and apply ds-Index on a problem of FMDS [5].**

| Graph $G$ | $\|V(G)\|$ | $\|E(G)\|$ | $\rho^*(G)$ | $\lceil\widetilde{\rho}\rceil$ | Reduced graph $G^-$ | | | Index time of ds-Index (seconds) | | | TopkDS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $\|V(G^-)\|$ | $\|E(G^-)\|$ | $\frac{\|E(G^-)\|}{\|E(G)\|}$ | Reduction | Flow | Total | (seconds) |
| CA-CondMat | 21,363 | 91,286 | 13.37 | 13 | 719 | 7,744 | 8.48% | 0.002 | 0.004 | 0.006 | 0.15 |
| Email-EuAll | 224,832 | 339,925 | 32.92 | 33 | 527 | 17,346 | 5.10% | 0.007 | 0.06 | 0.067 | 0.207 |
| Epinions | 75,877 | 405,739 | 60.25 | 61 | 999 | 60,192 | 14.84% | 0.008 | 0.24 | 0.248 | 0.67 |
| slashdot | 77,350 | 468,554 | 42.13 | 43 | 205 | 8,637 | 1.84% | 0.009 | 0.014 | 0.023 | 0.854 |
| dblp | 317,080 | 1,049,866 | 56.57 | 57 | 280 | 13,609 | 1.30% | 0.049 | 0.021 | 0.07 | 0.635 |
| web-Stanford | 281,903 | 1,992,636 | 59.39 | 60 | 1,370 | 78,797 | 3.95% | 0.083 | 0.15 | 0.233 | 1.9 |
| com-youtube | 1,134,890 | 2,987,624 | 45.60 | 46 | 2,269 | 103,342 | 3.46% | 0.193 | 0.807 | 1 | 5.1 |
| web-Google | 875,713 | 4,322,051 | 28.04 | 28 | 787 | 16,641 | 0.39% | 0.336 | 0.11 | 0.446 | 6.2 |
| WikiTalk | 2,388,953 | 4,656,682 | 114.14 | 115 | 1,384 | 157,968 | 3.39% | 0.195 | 0.935 | 1.13 | 4.5 |
| youtube-growth | 3,223,585 | 9,375,374 | 77.47 | 78 | 1,219 | 94,427 | 1.01% | 0.982 | 0.768 | 1.75 | 18 |
| as-skitter | 1,694,616 | 11,094,209 | 89.40 | 90 | 915 | 73,480 | 0.66% | 0.644 | 0.286 | 0.93 | 12 |
| soc-flickr-und | 1,715,255 | 15,555,041 | 468.83 | 469 | 3,135 | 1,469,797 | 9.45% | 0.566 | 19.434 | 20 | 80 |
| patent | 3,774,768 | 16,518,947 | 40.13 | 41 | 730 | 25,697 | 0.16% | 2.88 | 0.6 | 3.48 | 35 |
| soc-pokec | 1,632,803 | 22,301,964 | 41.13 | 42 | 8,974 | 368,613 | 1.65% | 1.87 | 6.53 | 8.4 | 480 |
| LiveJournal | 4,843,953 | 42,845,684 | 229.85 | 228 | 3,639 | 661,891 | 1.54% | 4.65 | 1.72 | 6.37 | 67 |
| twitter-mpi | 9,862,152 | 99,940,317 | 602.44 | 603 | 8,448 | 5,089,428 | 5.09% | 5.36 | 164.64 | 170 | 624 |
| tech-p2p | 5,792,297 | 147,829,887 | 750.18 | 751 | 7,641 | 5,732,158 | 3.88% | 19 | 283 | 302 | 1,671 |
| uk-2002 | 18,459,128 | 261,556,721 | 471.50 | 472 | 3,429 | 1,231,751 | 0.47% | 9.6 | 2.4 | 12 | 142 |
| uk-2005 | 39,252,879 | 781,439,892 | 485.75 | 429 | 51,784 | 15,037,470 | 1.92% | 26 | 77 | 103 | 515 |
| webbase | 115,554,441 | 854,809,761 | 816.92 | 804 | 9,990 | 6,631,895 | 0.78% | 61 | 36 | 97 | 813 |
| it-2004 | 41,290,577 | 1,027,474,895 | 2008.19 | 2,009 | 4,279 | 8,593,024 | 0.84% | 30 | 53 | 83 | 1,523 |
| twitter-2010 | 41,652,230 | 1,202,513,046 | 1643.30 | 1,644 | 11,619 | 17,996,107 | 1.50% | 143 | 320 | 463 | 10,205 |

of Web Algorithmics[3], the Koblenz Network Collection[4], and the network repository[5]. The statistics of the graphs are included in Table 1. In particular, the largest graph twitter-2010 has 42 million nodes and 1.2 billion undirected edges. A machine with an Intel(R) Xeon(R) 3.4GHz CPU and 16GB main memory under Linux System (64bit Debian) were employed to run the codes written in C++. [6]

**Eval-I: Construction Time of ds-Index.** The construction time is shown in Table 1 in two steps: the reduction step produces a reduced graph $G^-$; the flow step solves a parametric network-flow problem on a flow network derived from $G^-$. The size of ds-Index is bounded by the size of $G^-$. As Table 1 shows, the reduction step reduces an average of 97% of the total number of edges from a graph using an average of 14 seconds. The dramatic reduction of the graph size enables (i) the algorithm of parametric network-flow in the flow step to be completed in an average of 45 seconds and (ii) the ds-Index to be stored in the main memory under general memory settings. In particular, on the largest graph with 1.2 billion edges, the reduced graph has only 18 million edges while the ds-Index was constructed within 8 minutes in total.

**Eval-II: Top-$k$ Dense Subgraphs Identification.** As an application of ds-Index, we consider a concrete problem of finding multiple dense subgraphs that was proposed by Balalau et al. [5]. This problem aims at computing $k$ subgraphs that maximize the summation of the subgraph densities while ensuring that the Jaccard similarity between the node sets of every pair of subgraphs is at most $\alpha$. Here $\alpha$ is a user-given threshold. Balalau et al. proved the NP-hardness of this problem and proposed a MinAndRemove framework to find $k$ optimal subgraphs approximately. Let $G$ be the input graph. MinAndRemove reports $k$ subgraphs in $k$ iterations. In each iteration, MinAndRemove reports a minimal densest subgraph $S$ of $G$ and then removes a fraction of $(1-\alpha)$ vertices of $S$ from $G$. We plug the ds-Index into this framework to compute the minimal densest subgraph of $G$ in each iteration and denote this algorithm as TopkDS. Note that, since the graph $G$ is changing in each iteration, ds-Index is constructed on $k$ distinct graphs.

The results of running TopkDS on real graphs to compute top-10 dense subgraphs with $\alpha = 0.1$ are illustrated in Table 1. The last column shows the total running time of TopkDS. We can see that our ds-Index algorithm can scale the MinAndRemove framework to graphs with over one billion edges. The original MinAndRemove algorithm reported in [5] took 0.3 hour, 0.54 hour, 2.15 hours, 1.5 hours, and 1.29 hours, respectively, for processing graphs of web-Stanford, com-youtube, web-Google, youtube-growth, and as-skitter; in comparison, TopkDS took less than 18 seconds on these graphs. The superiority of TopkDS over the existing algorithm in [5] attributes to our efficient computation of minimal densest subgraphs using ds-Index.

## 6 CONCLUSION

This paper unveils the distributions and relationships of the densest subgraphs of a graph $G$ under the density notion of average-degree. Denote by $L$ the size of the maximal densest subgraph of $G$. There is an index called the ds-Index that encodes the structure of all the densest subgraphs of $G$ in $O(L)$ space. With the ds-Index, all the (minimal) densest subgraphs of $G$ can be enumerated optimally. Furthermore, the construction of ds-Index is no more expensive than the state-of-the-art solution for finding a single densest subgraph.

**Acknowledgements.** Lijun Chang is supported by ARC DP160101513 and FT180100256. Miao Qiao is supported by Marsden Fund UOA1732, Royal Society of New Zealand.

[3] http://law.di.unimi.it/datasets.php
[4] http://konect.uni-koblenz.de/
[5] http://networkrepository.com/
[6] Our source code is publicly available at https://github.com/LijunChang/Cohesive_subgraph_book/densest_subgraph

## REFERENCES


[1] [n.d.]. full version, https://lijunchang.github.io/pdf/dds.pdf.
[2] T. Akiba, Y. Iwata, and Y. Yoshida. 2013. Linear-time enumeration of maximal K-edge-connected subgraphs in large networks by random contraction. In *Proc. of CIKM'13.*
[3] A. Angel, N. Koudas, N. Sarkas, and D. Srivastava. 2012. Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification. *PVLDB* 5, 6 (2012), 574–585.
[4] B. Bahmani, R. Kumar, and S. Vassilvitskii. 2012. Densest Subgraph in Streaming and MapReduce. *PVLDB* 5, 5 (2012), 454–465.
[5] O. D. Balalau, F. Bonchi, T.-H. H. Chan, F. Gullo, and M. Sozio. 2015. Finding Subgraphs with Maximum Total Density and Limited Overlap. In *Proc. of WSDM'15.* 379–388.
[6] V. Batagelj and M. Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR* (2003).
[7] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. 2013. CopyCatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proc. of WWW'13.* 119–130.
[8] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. 2015. Space- and Time-Efficient Algorithm for Maintaining Dense Subgraphs on One-Pass Dynamic Streams. In *Proc. of STOC'15.* 173–182.
[9] N. Biggs, E. K. Lloyd, and R. J. Wilson. 1986. *Graph Theory, 1736-1936.* Clarendon Press.
[10] C. Bron and J. Kerbosch. 1973. Finding All Cliques of an Undirected Graph (Algorithm 457). *CACM* 16, 9 (1973), 575–576.
[11] Lijun Chang. 2019. Efficient Maximum Clique Computation over Large Sparse Graphs. In *Proc. of KDD'19.* 529–538.
[12] Lijun Chang and Lu Qin. 2018. *Cohesive Subgraph Computation over Large Sparse Graphs.* Springer Series in the Data Sciences.
[13] L. Chang, J. X. Yu, L. Q., X. Lin, C. Liu, and W. Liang. 2013. Efficiently computing k-edge connected components via graph decomposition. In *Proc. of SIGMOD'13.*
[14] M. Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop.* 84–95.
[15] J. Cohen. 2008. Trusses: Cohesive Subgraphs for Social Network Analysis.
[16] M. Danisch, T.-H. H. Chan, and M. Sozio. 2017. Large Scale Density-friendly Graph Decomposition via Convex Programming. In *Proc. of WWW'17.* 233–242.
[17] Y. Dourisboure, F. Geraci, and M. Pellegrini. 2007. Extraction and classification of dense communities in the web. In *Proc. of WWW'07.* 461–470.
[18] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. 2015. Efficient Densest Subgraph Computation in Evolving Graphs. In *Proc. of WWW'15.* 300–310.
[19] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V. S. Lakshmanan, and Xuemin Lin. 2019. Efficient Algorithms for Densest Subgraph Discovery. *PVLDB* 12, 11 (2019), 1719–1732.
[20] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. 1989. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM J. of Comp.* 18, 1 (1989), 30–55.
[21] D. Gibson, R. Kumar, and A. Tomkins. 2005. Discovering Large Dense Subgraphs in Massive Graphs. In *PVLDB.* 721–732.
[22] A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph.* Technical Report. Berkeley, CA, USA.
[23] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. 2010. A Survey of Algorithms for Dense Subgraph Discovery. In *Managing and Mining Graph Data.* 303–336.
[24] Robert Mokken. 1979. Cliques, clubs and clans. *Quality & Quantity* 13 (1979), 161–173.
[25] L. Qin, R. H. Li, L. Chang, and C. Zhang. 2015. Locally Densest Subgraph Discovery. In *Proc. of KDD'15.* 965–974.
[26] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek. 2015. Finding the Hierarchy of Dense Subgraphs using Nucleus Decompositions. In *Proc. of WWW'15.* 927–937.
[27] S. B. Seidman and B. L. Foster. 1978. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology* 6 (1978), 139–154.
[28] M. Sozio and A. Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proc. of KDD'10.* 939–948.
[29] N. Tatti and A. Gionis. 2015. Density-friendly Graph Decomposition. In *Proc. of WWW'15.* 1089–1099.
[30] Charalampos E. Tsourakakis. 2015. The K-clique Densest Subgraph Problem. In *Proc. of WWW'15.* 1122–1132.
[31] E. Valari, M. Kontaki, and A. N. Papadopoulos. 2012. Discovery of Top-k Dense Subgraphs in Dynamic Graph Collections. In *Proc. of SSDBM'12.* 213–230.
[32] J. Wang and J. Cheng. 2012. Truss Decomposition in Massive Networks. *PVLDB* 5, 9 (2012).


## A THE PROOF OF LEMMA 1

According to Equation 1, the capacity of a minimum $s$-$t$ cut $(S^* \cup \{s\}, \overline{S^*} \cup \{t\})$ is

$$c(S^* \cup \{s\}, \overline{S^*} \cup \{t\}) = \min_{S \subseteq V(G)} c(S \cup \{s\}, \overline{S} \cup \{t\})$$
$$= 2m + 2 \min_{S \subseteq V(G)} (\lambda - \rho(S))|S|. \quad (2)$$

We prove the correctness of Lemma 1 in three cases, respectively.

(1) Let $S'$ be a densest subgraph of $G$. Since $G$ is connected and has at least two nodes, $|S'| > 1$. If $\lambda < \rho^*(G) = \rho(S')$, then $\min_{S \subseteq V(G)}(\lambda - \rho(S))|S| \leq (\lambda - \rho(S'))|S'| < 0$. According to Equation 2, $c(S^* \cup \{s\}, \overline{S^*} \cup \{t\}) < 2m$.
(2) If $\lambda \geq \rho^*(G)$ then for any $S \subseteq V(G)$, $(\lambda - \rho(S))|S| \geq 0$. Besides, when $S = \emptyset$, $(\lambda - \rho(S))|S| = 0$. Therefore, $\min_{S \subseteq V(G)}(\lambda - \rho(S))|S| = 0$ and thus $c(S^* \cup \{s\}, \overline{S^*} \cup \{t\}) = 2m$.
(3) Consider any induced subgraph $S''$ of $G$ with $\rho(S'') < \rho^*(G)$. Since both $\rho^*(G)$ and $\rho(S'')$ take values from all possible fractions between an integer in $[0, m]$ and an integer in $[0, n]$, $\rho^*(G) - \rho(S'') \geq \frac{1}{n(n-1)}$. Therefore, there is no subgraph of $G$ having a density in range $(\rho^*(G) - \frac{1}{n(n-1)}, \rho^*(G))$. When $\lambda \in (\rho^*(G) - \frac{1}{n(n-1)}, \rho^*(G))$, $\min_{S \subseteq V(G)}(\lambda - \rho(S))|S|$ takes the minimum value only when $S$ is the densest subgraph of $G$ with the largest cardinality. According to Equations 1 and 2, $S^*$ is the densest subgraph of $G$ with the maximum cardinality.

## B THE PROOF OF LEMMA 3

Recall that, the capacity of cut $(S \cup \{s\}, \overline{S} \cup \{t\})$ is $c(S \cup \{s\}, \overline{S} \cup \{t\}) = 2m + 2(\lambda - \rho(S))|S|$ (Equation 1). Besides, the critical flow network $\mathcal{H}$ has $\lambda = \rho^*(G)$, and $S \neq \emptyset$. Therefore, $\rho(S) = \lambda = \rho^*(G)$ if and only if $c(S \cup \{s\}, \overline{S} \cup \{t\}) = 2m$. Since the capacity of the minimum $s$-$t$ cut of $\mathcal{H}$ is $2m$ (Lemma 1), $S$ is a densest subgraph $G$ if and only if $(S \cup \{s\}, \overline{S} \cup \{t\})$ is a minimum cut of $\mathcal{H}$.

The equivalence between (2) and (3) directly follows from the following extension to the max-flow min-cut theorem.

Lemma 11. *Given a flow network $\vec{G}$, let $f$ be a maximum flow on $\vec{G}$ and let $(X, Y)$ be an $s$-$t$ cut of $\vec{G}$. $(X, Y)$ is a minimum $s$-$t$ cut if and only if all edges from $X$ to $Y$ are saturated under $f$.*

Proof. The lemma is proved in two directions.

(1) We prove that if $(X, Y)$ is a minimum $s$-$t$ cut then all edges from $X$ to $Y$ must be saturated by contradiction. Assume that there is an edge $e$ from $X$ to $Y$ with $f(e) < c(e)$, then the value of the flow from $X$ to $Y$ is strictly smaller than the capacity of cut $(X, Y)$, that is, $\Sigma_{e \text{ from } X \text{ to } Y} f(e) < \Sigma_{e \text{ from } X \text{ to } Y} c(e) = c(X, Y)$. Therefore, the value of the flow is

$$val(f) = \Sigma_{v \in V(\vec{G}) \setminus \{s\}, \langle s, v \rangle \in E(\vec{G})} f(\langle s, v \rangle)$$
$$= \Sigma_{u \in X \setminus \{s\}, \langle s, u \rangle \in E(\vec{G})} f(\langle s, u \rangle) + \Sigma_{v \in Y, \langle s, v \rangle \in E(\vec{G})} f(\langle s, v \rangle)$$

Due to the conservation property of a flow,

$$\Sigma_{u \in X \setminus \{s\}, v \in V(\vec{G}), \langle u, v \rangle \in E(\vec{G})} f(\langle u, v \rangle) = 0.$$

Therefore,

$$\begin{aligned} val(f) =& \Sigma_{u\in X\setminus\{s\},\langle s,u\rangle\in E(\vec{G})} f(\langle s,u\rangle) \\ &+ \Sigma_{v\in Y,\langle s,v\rangle\in E(\vec{G})} f(\langle s,v\rangle) \\ &+ \Sigma_{u\in X\setminus\{s\}} \Sigma_{v\in V(\vec{G}),\langle u,v\rangle\in E(\vec{G})} f(\langle u,v\rangle) \qquad (3) \\ &\qquad \text{split the Term (3) into three cases} \\ =& \Sigma_{u\in X\setminus\{s\},\langle s,u\rangle\in E(\vec{G})} f(\langle s,u\rangle) \qquad (4) \\ &+ \Sigma_{v\in Y,\langle s,v\rangle\in E(\vec{G})} f(\langle s,v\rangle) \\ &+ \Sigma_{u\in X\setminus\{s\}} \Sigma_{v\in X\setminus\{s\},\langle u,v\rangle\in E(\vec{G})} f(\langle u,v\rangle) \qquad (5) \\ &+ \Sigma_{u\in X\setminus\{s\}} \Sigma_{v\in Y,\langle u,v\rangle\in E(\vec{G})} f(\langle u,v\rangle) \\ &+ \Sigma_{u\in X\setminus\{s\},\langle u,s\rangle\in E(\vec{G})} f(\langle u,s\rangle) \qquad (6) \end{aligned}$$

Term (4) cancels Term (6) due to the antisymmetry property of a flow, Term (5) is 0 due to the conservation property of a flow. Therefore,

$$\begin{aligned} val(f) =& \Sigma_{v\in Y,\langle s,v\rangle\in E(\vec{G})} f(\langle s,v\rangle) + \Sigma_{u\in X\setminus\{s\}} \Sigma_{v\in Y,\langle u,v\rangle\in E(\vec{G}))} f(\langle u,v\rangle) \\ =& \Sigma_{e \text{ from X to Y}} f(e) < \Sigma_{e \text{ from } X \text{ to } Y} c(e) = c(X,Y) \end{aligned}$$

According to the max-flow min-cut theorem, the capacity of a minimum cut equals the value of the maximum flow, $c(X,Y) = val(f)$, contradiction.

(2) If all edges $e$ from $X$ to $Y$ are saturated, then $val(f) = c(X,Y)$. According to max-flow min-cut theorem, since $f$ is a maximum flow, there is no $s$-$t$ cut with a capacity smaller than $val(f)$. $(X,Y)$ is therefore a minimum cut. □

## C THE PROOF OF LEMMA 5

Denote by $E^+ = \{e \in E(\mathcal{H}) | f^*(e) > 0\}$ the set of edges with positive flow valued in the critical flow network $\mathcal{H}$. To prove Lemma 5, it suffices to show that for every node $v \in V(G)$, there is a path from $v$ to $t$ in the critical flow network $\mathcal{H}$ with only $E^+$ edges. This is because that if an edge $\langle u,v\rangle$ has $f^*(\langle u,v\rangle) > 0$, then the critical residual graph $\mathcal{H}_{f^*}$ must include its reverse edge $\langle v,u\rangle$ since the residual capacity of the reverse edge $c_{f^*}(\langle v,u\rangle) = c(\langle v,u\rangle) - f^*(\langle v,u\rangle) = c(\langle v,u\rangle) + f^*(\langle u,v\rangle) > 0$.

Let $v^*$ be an arbitrary node in $V(G)$. Let $S$ be the union of $\{v^*\}$ and the set of all nodes in the critical flow network $\mathcal{H}$ that are reachable from $v^*$ with only $E^+$ edges. The definition of $S$ implies that for every node $u$ in $S$, there is a path from $v^*$ to $u$ with only $E^+$ edges; besides, any edge $e$ in $\mathcal{H}$ from a node in $S$ to $V(\mathcal{H}) \setminus S$ has $f^*(e) \le 0$. We prove by contradiction that $t \in S$.

We first prove that $s \notin S$. For any $u \in V(G)$, we know from Lemma 4 that the edge from $s$ to $u$ is saturated, thus, $f^*(u,s) = -f^*(s,u) = -d(u) < 0$. Besides, there is no edge in $\mathcal{H}$ between $s$ and $t$. Therefore, all edges $e$ to $s$ has negative value of the flow, that is, $s$ cannot be reached by $v^*$ via edges in $E^+$.

If $t \notin S$, since $s \notin S$, then $S \subseteq V(G)$. Denote $\overline{S} = V(G) \setminus S$. For an edge $e$ that is not in the critical flow network $\mathcal{H}$, we define $f^*(e) = 0$ for convenience. For a node $w \in S$, due to the conservation property of a flow, the flow to $w$ has

$$\begin{aligned} \Sigma_{u\in V(\mathcal{H})} f^*(\langle u,w\rangle) \quad &= \Sigma_{u\in S} f^*(\langle u,w\rangle) + \Sigma_{u\in\overline{S}} f^*(\langle u,w\rangle) \\ &\quad + f^*(\langle t,w\rangle) + f^*(\langle s,w\rangle) = 0. \end{aligned}$$

Aggregate over $w$, we have

$$\begin{aligned} \Sigma_{w\in S,u\in V(\mathcal{H})} f^*(\langle u,w\rangle) \quad &= \quad \Sigma_{w\in S,u\in\overline{S}} f^*(\langle u,w\rangle) \\ &\quad + \Sigma_{w\in S} f^*(\langle t,w\rangle) + \Sigma_{w\in S} f^*(\langle s,w\rangle) \\ &\quad + \Sigma_{w,u\in S} f^*(\langle u,w\rangle) = 0. \end{aligned}$$

Due to the antisymmetry property, term $\Sigma_{w,u\in S} f^*(\langle u,w\rangle) = 0$. Besides, $\Sigma_{w\in S} f^*(\langle s,w\rangle) = \Sigma_{u\in S} d(u) \ge d(v^*) > 0$ because $v^* \in S$. Therefore, $\Sigma_{w\in S,u\in\overline{S}} f^*(\langle u,w\rangle) + \Sigma_{w\in S} f^*(\langle t,w\rangle) < 0$. However, the construction of $S$ indicates that, for every $w \in S$, all edges from $w$ to $V(\mathcal{H}) \setminus S$ are non-positive and therefore, $\Sigma_{u\in\overline{S}} f^*(\langle u,w\rangle) \ge 0$ and $f^*(\langle t,w\rangle) \ge 0$, contradiction.

## D PROPERTIES OF THE CRITICAL COMPONENT GRAPH

Lemma 12 (Critical Properties). *The critical component graph $\mathcal{H}^C$ of the critical residual graph $\mathcal{H}_{f^*}$ has the following properties:*

*(1) the component $scc(t)$ of the target node $t$ has no incoming edge;*
*(2) the component $scc(s)$ of the source node $s$ has no outgoing edge;*
*(3) $scc(t)$ has an edge to $scc(s)$ if and only if the component $scc(t)$ has a node other than $t$,* i.e., *$|scc(t)| > 1$;*
*(4) $scc(s)$ has no nodes other than $s$ itself,* i.e., *$scc(s) = \{s\}$.*

*Besides, for each component $C \in C(\mathcal{H}_{f^*})$ with neither $s$ nor $t$ included,*

*(5) there is an edge from $scc(t)$ to $C$, and*
*(6) there is an edge from $C$ to $scc(s)$.*

Proof. The correctness of this lemma depends on Lemma 4 and Lemma 5.

*The proof of Properties 2, 3, 4, 6.* Proved directly by Lemma 4.

*The proof of Property 1.* If there is an incoming edge in the critical component graph from a component $C$ to $scc(t)$, let $v_c$ be any node in $C$, then $v_c \ne t$ and $v_c$ can reach $t$ in the critical residual graph, that is, $v_c \rightsquigarrow t$ on $\mathcal{H}_{f^*}$. Since $scc(s)$ has no outgoing edge (Property (2)), $v_c \ne s$. Therefore, $v_c \in V(G)$. According to Lemma 5, $t \rightsquigarrow v_c$ in the critical residual graph $\mathcal{H}_{f^*}$. $v$ and $t$ are then in the same strongly connected component, contradicting that $v_c \in C \ne scc(t)$. Therefore, $scc(t)$ has no incoming edge.

*The proof of Property 5.* Let $v$ be a non-trivial node in $V(G) \setminus scc(t) \setminus scc(s)$. Lemma 5 shows that $t \rightsquigarrow v$ in the critical residual graph $\mathcal{H}_{f^*}$. If $v \rightsquigarrow t$, then $v \in scc(t)$, contradiction. Therefore, $v \not\rightsquigarrow t$ in $\mathcal{H}_{f^*}$. Since the critical flow network $\mathcal{H}$ has an edge $\langle v,t\rangle$ whose capacity is $2\rho^*(G) > 0$, $v \not\rightsquigarrow t$ in the critical residual graph $\mathcal{H}_{f^*}$ means that edge $\langle v,t\rangle$ is saturated under $f^*$. Therefore, the residual capacity of the reverse edge $\langle t,v\rangle$ is positive, that is, there is an edge from $t$ to $v$ in $\mathcal{H}_{f^*}$. Therefore, there is an edge from $scc(t)$ to all the components other than $scc(s)$ in the component graph. This completes the proof of Lemma 12. □

## E THE PROOF OF LEMMA 9

We prove the lemma by contradiction. Since $G$ is a connected graph with at least two nodes, $S$ has at least two nodes. Let $v$ be the node in $S$ with $d_S(v) < \lceil \rho^*(G) \rceil$, where $d_S(v)$ is the degree of $v$ in the subgraph $S$. Then, the density of $S \setminus \{v\}$ is

$$\rho(S \setminus \{v\}) = \frac{\rho^*(G) \times |S| - d_S(v)}{|S| - 1} > \rho^*(G),$$

contradicting the maximality of the density of $\rho^*(G)$. Thus, the lemma holds.