The Densest Subgraph Problem with a Convex/Concave Size Function*

Yasushi Kawase[1] and Atsushi Miyauchi[2]

1 **Tokyo Institute of Technology, Tokyo, Japan**
`kawase.y.ab@m.titech.ac.jp`
2 **Tokyo Institute of Technology, Tokyo, Japan**
`miyauchi.a.aa@m.titech.ac.jp`

Abstract

Given an edge-weighted undirected graph $G = (V, E, w)$, the *density* of $S \subseteq V$ is defined as $w(S)/|S|$, where $w(S)$ is the sum of weights of the edges in the subgraph induced by S. The densest subgraph problem asks for $S \subseteq V$ that maximizes the density $w(S)/|S|$. The problem has received significant attention recently because it can be solved exactly in polynomial time. However, the densest subgraph problem has a drawback; it may happen that the obtained subset is too large or too small in comparison with the desired size of the output.

In this study, we address the size issue by generalizing the density of $S \subseteq V$. Specifically, we introduce the *f-density* of $S \subseteq V$, which is defined as $w(S)/f(|S|)$, where $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a monotonically non-decreasing function. In the *f-densest subgraph problem* (f-DS), we are asked to find $S \subseteq V$ that maximizes the f-density $w(S)/f(|S|)$. Although f-DS does not explicitly specify the size of the output subset of vertices, we can handle the above size issue using a *convex* size function f or a *concave* size function f appropriately. For f-DS with convex function f, we propose a nearly-linear-time algorithm with a provable approximation guarantee. In particular, for f-DS with $f(x) = x^\alpha$ ($\alpha \in [1, 2]$), our algorithm has an approximation ratio of $2 \cdot n^{(\alpha-1)(2-\alpha)}$. On the other hand, for f-DS with concave function f, we propose a linear-programming-based polynomial-time exact algorithm. It should be emphasized that this algorithm obtains not only an optimal solution to the problem but also subsets of vertices corresponding to the extreme points of the upper convex hull of $\{(|S|, w(S)) \mid S \subseteq V\}$, which we refer to as the *dense frontier points*. We also propose a flow-based combinatorial exact algorithm for unweighted graphs that runs in $O(n^3)$ time. Finally, we propose a nearly-linear-time 3-approximation algorithm.

1998 ACM Subject Classification G.2.2 Graph Theory: Network problems

Keywords and phrases graphs, dense subgraph extraction, densest subgraph problem, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.44

1 Introduction

Finding dense components in a graph is an active research topic in graph mining. Techniques for identifying dense subgraphs have been used in various applications. For example, in Web graph analysis, they are used for detecting communities (i.e., sets of web pages dealing with the same or similar topics) [9] and spam link farms [12]. As another example, in bioinformatics, they are used for finding molecular complexes in protein interaction networks [4] and

* The first author is supported by a Grant-in-Aid for Young Scientists (B) (No. 16K16005). The second author is supported by a Grant-in-Aid for JSPS Fellows (No. 26-11908).

identifying regulatory motifs in DNA [10]. Furthermore, they are also used for expert team formation [6, 17] and real-time story identification in micro-blogging streams [2].

To date, various optimization problems have been considered to find dense components in a graph. The densest subgraph problem is one of the most well-studied optimization problems. Let $G = (V, E, w)$ be an edge-weighted undirected graph consisting of $n = |V|$ vertices, $m = |E|$ edges, and a weight function $w : E \to \mathbb{Q}_{>0}$, where $\mathbb{Q}_{>0}$ is the set of positive rational numbers. For a subset of vertices $S \subseteq V$, let $G[S]$ be the subgraph induced by $S \subseteq V$, i.e., $G[S] = (S, E(S))$, where $E(S) = \{\{i, j\} \in E \mid i, j \in S\}$. The *density* of $S \subseteq V$ is defined as $w(S)/|S|$, where $w(S) = \sum_{e \in E(S)} w(e)$. In the (weighted) densest subgraph problem, given an (edge-weighted) undirected graph $G = (V, E, w)$, we are asked to find $S \subseteq V$ that maximizes the density $w(S)/|S|$.

The densest subgraph problem has received significant attention recently because it can be solved exactly in time polynomial in n and m. In fact, there exist a flow-based exact algorithm [13] and a linear-programming-based (LP-based) exact algorithm [7]. Moreover, Charikar [7] demonstrated that the greedy algorithm designed by Asahiro et al. [3], which is called the *greedy peeling*, obtains a 2-approximate solution[1] for any instance. This algorithm runs in $O(m + n)$ time for unweighted graphs and $O(m + n \log n)$ time for weighted graphs.

However, the densest subgraph problem has a drawback; it may happen that the obtained subset is too large or too small in comparison with the desired size of the output. To overcome this issue, some variants of the problem have often been employed. The densest k-subgraph problem (DkS) is a straightforward size-restricted variant of the densest subgraph problem. In this problem, given an additional input k being a positive integer, we are asked to find $S \subseteq V$ of size k that maximizes the density $w(S)/|S|$. Note that in this problem, the objective function can be replaced by $w(S)$ since $|S|$ is fixed to k. Unfortunately, it is known that this size restriction makes the problem much harder to solve. In fact, Khot [14] proved that DkS has no PTAS under some reasonable computational complexity assumption. The current best approximation algorithm has an approximation ratio of $O(n^{1/4+\epsilon})$ for any $\epsilon > 0$ [5].

Furthermore, Andersen and Chellapilla [1] introduced two relaxed versions of DkS. The first problem, the densest at-least-k-subgraph problem (Dal$k$S), asks for $S \subseteq V$ that maximizes the density $w(S)/|S|$ under the size constraint $|S| \geq k$. For this problem, Andersen and Chellapilla [1] adopted the greedy peeling, and demonstrated that the algorithm yields a 3-approximate solution for any instance. Later, Khuller and Saha [15] investigated the problem more deeply. They proved that Dal$k$S is NP-hard, and designed a flow-based algorithm and an LP-based algorithm. These algorithms have an approximation ratio of 2, which improves the above approximation ratio of 3. The second problem is called the densest at-most-k-subgraph problem (Dam$k$S), which asks for $S \subseteq V$ that maximizes the density $w(S)/|S|$ under the size constraint $|S| \leq k$. The NP-hardness is immediate since finding a maximum clique can be reduced to it. Khuller and Saha [15] proved that approximating Dam$k$S is as hard as approximating DkS within a constant factor.

1.1 Our Contribution

In this study, we address the above size issue by generalizing the density of $S \subseteq V$. Specifically, we introduce the *f-density* of $S \subseteq V$, which is defined as $w(S)/f(|S|)$, where $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$

[1] A feasible solution is called a *γ-approximate solution* if its objective value times γ is greater than or equal to the optimal value. An algorithm is called a *γ-approximation algorithm* if it runs in polynomial time and returns a γ-approximate solution for any instance. For a γ-approximation algorithm, γ is referred to as an *approximation ratio* of the algorithm.

is a monotonically non-decreasing function with $f(0) = 0$.[2] Note that $\mathbb{Z}_{\geq 0}$ and $\mathbb{R}_{\geq 0}$ are the set of nonnegative integers and the set of nonnegative real numbers, respectively. In the *f-densest subgraph problem* (f-DS), we are asked to find $S \subseteq V$ that maximizes the f-density $w(S)/f(|S|)$. For simplicity, we assume that $E \neq \emptyset$. Thus, any optimal solution S^* satisfies $|S^*| \geq 2$. Although f-DS does not explicitly specify the size of the output subset of vertices, we can handle the above size issue using a *convex* size function f or a *concave* size function f appropriately. In fact, we see that any optimal solution to f-DS with convex (resp. concave) function f has a size smaller (resp. larger) than or equal to that of the densest subgraph. For details, see Section 2 and Section 3.

Here we mention the relationship between our problem and DkS. Any optimal solution S^* to f-DS is a maximum weight subset of size $|S^*|$, i.e., $\mathrm{argmax}\{w(S) \mid S \subseteq V,\ |S| = |S^*|\}$, which implies that S^* is also optimal to DkS with $k = |S^*|$. Furthermore, a γ-approximation algorithm for DkS implies a γ-approximation algorithm for f-DS. Using the above $O(n^{1/4+\epsilon})$-approximation algorithm for DkS, we can obtain an $O(n^{1/4+\epsilon})$-approximation algorithm for f-DS.

We summarize our results for each of the case where f is convex and f is concave.

The case where f is convex. Let us describe our results for the case where the size function f is convex. A function $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is called *convex* if $f(x) - 2f(x+1) + f(x+2) \geq 0$ holds for any $x \in \mathbb{Z}_{\geq 0}$. We first prove the NP-hardness of f-DS with a certain convex function by a reduction from Dam$k$S. Thus, for f-DS with convex function f, one of the best possible ways is to design algorithms with a provable approximation guarantee.

To this end, we propose a $\min\left\{\frac{2f(n)/n}{f(|S^*|)-f(|S^*|-1)}, \frac{f(2)/2}{f(|S^*|)/|S^*|^2}\right\}$-approximation algorithm, where $S^* \subseteq V$ is an optimal solution to f-DS with convex function f. Our algorithm consists of the following two procedures, and outputs the better solution found by them. The first one is based on the brute-force search, which obtains an $\frac{f(2)/2}{f(|S^*|)/|S^*|^2}$-approximate solution in $O(m+n)$ time. The second one adopts the greedy peeling, which obtains a $\frac{2f(n)/n}{f(|S^*|)-f(|S^*|-1)}$-approximate solution in $O(m + n\log n)$ time. Thus, the total running time of our algorithm is $O(m + n\log n)$. Our analysis on the approximation ratio of the second procedure extends the analysis by Charikar [7] for the densest subgraph problem.

At the end of our analysis, we observe the behavior of the approximation ratio of our algorithm for three size functions. We consider size functions *between* linear and quadratic because, as we will see later, f-DS with any super-quadratic size function only produces constant-size optimal solutions. The first example is $f(x) = x^\alpha$ ($\alpha \in [1,2]$). We show that the approximation ratio of our algorithm is $2 \cdot n^{(\alpha-1)(2-\alpha)}$, where the worst-case performance of $2 \cdot n^{1/4}$ is attained at $\alpha = 1.5$. The second example is $f(x) = \lambda x + (1-\lambda)x^2$ ($\lambda \in [0,1)$). For this case, the approximation ratio of our algorithm is $(2-\lambda)/(1-\lambda)$, which is a constant for a fixed λ. The third example is $f(x) = x^2/(\lambda x + (1-\lambda))$ ($\lambda \in [0,1]$). Note that this size function is derived by density function $\lambda\frac{w(S)}{|S|} + (1-\lambda)\frac{w(S)}{|S|^2}$. The approximation ratio of our algorithm is $4/(1+\lambda)$, which is at most 4.

The case where f is concave. Next let us describe our results for the case where the size function f is concave. A function $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is called *concave* if $f(x) - 2f(x+1) +$

[2] To handle various types of functions (e.g., $f(x) = x^\alpha$ for $\alpha > 0$), we set the codomain of the function f to be the set of nonnegative *real* numbers. We assume that we can compare $p \cdot f(i)$ and $q \cdot f(j)$ in constant time for any $p, q \in \mathbb{Q}$ and $i, j \in \mathbb{Z}_{\geq 0}$.

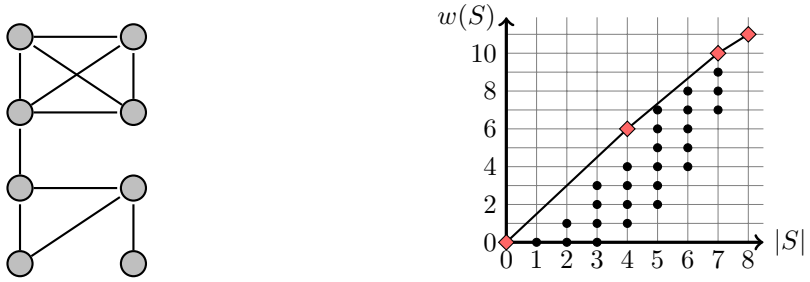

Figure 1 An example graph and corresponding points in $\mathcal{P} = \{(|S|, w(S)) \mid S \subseteq V\}$. The diamond-shaped points, i.e., $(0,0), (4,6), (7,10)$, and $(8,11)$, are the dense frontier points.

$f(x+2) \leq 0$ holds for any $x \in \mathbb{Z}_{\geq 0}$. Unlike the above convex case, f-DS in this case can be solved exactly in polynomial time.

In fact, we present an LP-based exact algorithm, which extends Charikar's exact algorithm for the densest subgraph problem [7] and Khuller and Saha's 2-approximation algorithm for Dal$k$S [15]. It should be emphasized that our LP-based algorithm obtains not only an optimal solution to f-DS but also some attractive subsets of vertices. Let us see an example in Figure 1. The graph consists of 8 vertices and 11 unweighted edges. For this graph, we plotted all the points contained in $\mathcal{P} = \{(|S|, w(S)) \mid S \subseteq V\}$. We refer to the extreme points of the upper convex hull of $\mathcal{P}$ as the *dense frontier points*. The densest subgraph is a typical subset of vertices that corresponds to a dense frontier point. Our LP-based algorithm obtains a corresponding subset of vertices for every dense frontier point.

Moreover, in this concave case, we design a combinatorial exact algorithm for unweighted graphs. Our algorithm is based on the standard technique for fractional programming. By using the technique, we can reduce f-DS to a sequence of submodular function minimizations. However, applying a submodular function minimization algorithm leads to a computationally expensive algorithm, which runs in $O(n^5(m+n) \cdot \log n)$ time. To reduce the computation time, we replace a submodular function minimization algorithm with a much faster flow-based algorithm that substantially extends a technique of Goldberg's flow-based algorithm for the densest subgraph problem [13]. The total running time of our algorithm is $O(n^3)$.

Although our flow-based algorithm is much faster than the reduction-based algorithm, the running time is still long for large-sized graphs. To design an algorithm with much higher scalability, we adopt the greedy peeling. As mentioned above, this algorithm runs in $O(m+n)$ time for unweighted graphs and $O(m + n \log n)$ time for weighted graphs. We see that the algorithm yields a 3-approximate solution for any instance.

1.2 Related Work

Tsourakakis et al. [17] introduced a general optimization problem to find dense subgraphs, which is referred to as the optimal (g, h, α)-edge-surplus problem. The problem asks for $S \subseteq V$ that maximizes $\mathsf{edge\text{-}surplus}_\alpha(S) = g(|E(S)|) - \alpha h(|S|)$, where g and h are strictly monotonically increasing functions, and $\alpha > 0$ is a constant. The intuition behind this optimization problem is the same as that of ours. In fact, the first term $g(|E(S)|)$ prefers $S \subseteq V$ that has a large number of edges, whereas the second term $-\alpha h(|S|)$ penalizes $S \subseteq V$ with a large size. Tsourakakis et al. [17] were motivated by finding near-cliques (i.e., relatively small dense subgraphs), and they derived the function $\mathsf{OQC}_\alpha(S) = |E(S)| - \alpha\binom{|S|}{2}$, which is called the OQC function, by setting $g(x) = x$ and $h(x) = x(x-1)/2$. For OQC function maximization, they adopted the greedy peeling and a simple local search heuristic.

Recently, Yanagisawa and Hara [18] introduced density function $|E(S)|/|S|^\alpha$ for $\alpha \in (1,2]$, which they called the discounted average degree. For discounted average degree maximization, they designed an integer-programming-based exact algorithm, which is applicable only to graphs with thousands of edges. They also designed a local search heuristic, which is applicable to Web-scale graphs but has no provable approximation guarantee. As mentioned above, our algorithm for f-DS with convex function f runs in $O(m + n \log n)$ time, and has an approximation ratio of $2 \cdot n^{(\alpha-1)(2-\alpha)}$ for $f(x) = x^\alpha$ $(\alpha \in [1,2])$.

2 Convex Case

In this section, we investigate f-DS with convex function f. A function $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is called *convex* if $f(x) - 2f(x+1) + f(x+2) \geq 0$ holds for any $x \in \mathbb{Z}_{\geq 0}$. We remark that $f(x)/x$ is monotonically non-decreasing for x since we assume that $f(0) = 0$. It should be emphasized that any optimal solution to f-DS with convex function f has a size smaller than or equal to that of the densest subgraph. To see this, let $S^* \subseteq V$ be an optimal solution to f-DS and $S^*_{\mathrm{DS}} \subseteq V$ be the densest subgraph. Then we have

$$\frac{f(|S^*|)}{|S^*|} = \frac{w(S^*)/|S^*|}{w(S^*)/f(|S^*|)} \leq \frac{w(S^*_{\mathrm{DS}})/|S^*_{\mathrm{DS}}|}{w(S^*_{\mathrm{DS}})/f(|S^*_{\mathrm{DS}}|)} = \frac{f(|S^*_{\mathrm{DS}}|)}{|S^*_{\mathrm{DS}}|}. \tag{1}$$

This implies the statement because $f(x)/x$ is monotonically non-decreasing.

2.1 Hardness

We first state that f-DS with convex function f contains Dam$k$S as a special case.

▶ **Theorem 1.** *For any integer $k \in [2,n]$, $S \subseteq V$ is optimal to Dam$k$S if and only if S is optimal to f-DS with (convex) function $f(x) = \max\{x,\ 2w(V) \cdot (x-k)/w(e) + k\}$, where e is an arbitrary edge.*

2.2 Our Algorithm

In this subsection, we provide an algorithm for f-DS with convex function f. Our algorithm consists of the following two procedures, and outputs the better solution found by them. Let S^* be an optimal solution to the problem. The first one is based on the brute-force search, which obtains an $\frac{f(2)/2}{f(|S^*|)/|S^*|^2}$-approximate solution in $O(m+n)$ time. The second one adopts the greedy peeling [3], which obtains a $\frac{2f(n)/n}{f(|S^*|)-f(|S^*|-1)}$-approximate solution in $O(m + n \log n)$ time. Combining these results, which will be proved later, we have the following theorem.

▶ **Theorem 2.** *Let $S^* \subseteq V$ be an optimal solution to f-DS with convex function f. For the problem, our algorithm runs in $O(m + n \log n)$ time, and it has an approximation ratio of*

$$\min\left\{\frac{2f(n)/n}{f(|S^*|) - f(|S^*|-1)},\ \frac{f(2)/2}{f(|S^*|)/|S^*|^2}\right\}.$$

2.2.1 Brute-Force Search

As will be shown below, to obtain an approximation ratio of $\frac{f(2)/2}{f(|S^*|)/|S^*|^2}$, it suffices to find the heaviest edge. Clearly, this algorithm runs in $O(m+n)$ time. However, we present a more general algorithm, which is useful for some case. Our algorithm examines all the

Algorithm 1 Brute-force search

1: **for** $i \leftarrow 2, \ldots, k$
2: Find $S_i^* \in \operatorname{argmax}\{w(S) \mid S \subseteq V,\ |S| = i\}$ by examining all the candidate subsets
3: **return** $S \in \{S_2^*, \ldots, S_k^*\}$ that maximizes $w(S)/f(|S|)$

subsets of vertices of size at most k, and then returns an optimal subset among them, where k is a constant and $k \geq 2$. For reference, we describe the procedure in Algorithm 1.

This algorithm can be implemented to run in $O((m+n)n^k)$ time because the number of subsets with at most k vertices is $\sum_{i=0}^{k} \binom{n}{i} = O(n^k)$ and the value of $w(S)/f(|S|)$ for $S \subseteq V$ can be computed in $O(m+n)$ time.

We analyze the approximation ratio of the algorithm. Let S_i^* denote a maximum weight subset of size i, i.e., $S_i^* \in \operatorname{argmax}\{w(S) \mid S \subseteq V,\ |S| = i\}$. We refer to $w(S_i^*)/\binom{i}{2}$ as the *edge density* of i vertices. The following lemma gives a fundamental property of the edge density.

▶ **Lemma 3.** *The edge density is monotonically non-increasing for the number of vertices, i.e., $w(S_i^*)/\binom{i}{2} \geq w(S_j^*)/\binom{j}{2}$ holds for any $1 \leq i \leq j \leq n$.*

Using the above lemma, we can provide the result of the approximation ratio.

▶ **Lemma 4.** *Let $S^* \subseteq V$ be an optimal solution to f-DS with convex function f. If $|S^*| \leq k$, then Algorithm 1 obtains an optimal solution. If $|S^*| \geq k$, then it holds that*

$$\frac{w(S^*)}{f(|S^*|)} \leq \frac{2 \cdot f(k)/k^2}{f(|S^*|)/|S^*|^2} \cdot \frac{w(S_k^*)}{f(k)}.$$

Proof. If $|S^*| \leq k$, then Algorithm 1 obtains an optimal solution since $S^* \in \{S_2^*, \ldots, S_k^*\}$. If $|S^*| \geq k$, then we have

$$\begin{aligned}\frac{w(S^*)}{f(|S^*|)} &\leq \frac{f(k)/\binom{k}{2}}{f(|S^*|)/\binom{|S^*|}{2}} \cdot \frac{w(S_k^*)}{f(k)} \\ &= \frac{1 - 1/|S^*|}{1 - 1/k} \cdot \frac{f(k)/k^2}{f(|S^*|)/|S^*|^2} \cdot \frac{w(S_k^*)}{f(k)} \leq \frac{2 \cdot f(k)/k^2}{f(|S^*|)/|S^*|^2} \cdot \frac{w(S_k^*)}{f(k)},\end{aligned}$$

where the first inequality follows from Lemma 3, and the last inequality follows from $|S^*| \geq k \geq 2$. ◀

From this lemma, we see that Algorithm 1 with $k = 2$ has an approximation ratio of $\frac{f(2)/2}{f(|S^*|)/|S^*|^2}$.

2.2.2 Greedy Peeling

Here we adopt the greedy peeling. For $S \subseteq V$ and $v \in S$, let $d_S(v)$ denote the weighted degree of the vertex v in the induced subgraph $G[S]$, i.e., $d_S(v) = \sum_{\{u,v\} \in E(S)} w(\{u,v\})$. Our algorithm iteratively removes the vertex with the smallest weighted degree in the currently remaining graph, and then returns $S \subseteq V$ with maximum $w(S)/f(|S|)$ over the iteration. For reference, we describe the procedure in Algorithm 2. This algorithm runs in $O(m+n)$ time for unweighted graphs and $O(m + n \log n)$ time for weighted graphs.

The following lemma provides the result of the approximation ratio.

▶ **Lemma 5.** *Let S^* be an optimal solution to f-DS with convex function f. Algorithm 2 returns a solution $S \subseteq V$ that satisfies*

$$\frac{w(S)}{f(|S|)} \geq \frac{1}{2} \cdot \frac{f(|S^*|) - f(|S^*| - 1)}{f(n)/n} \cdot \frac{w(S^*)}{f(|S^*|)}.$$

Algorithm 2 Greedy peeling

1: $S_n \leftarrow V$
2: **for** $i \leftarrow n, \dots, 2$
3: Find $v_i \in \mathrm{argmin}_{v \in S_i} d_{S_i}(v)$ and $S_{i-1} \leftarrow S_i \setminus \{v_i\}$
4: **return** $S \in \{S_1, \dots, S_n\}$ that maximizes $w(S)/f(|S|)$

Proof. Choose an arbitrary vertex $v \in S^*$. By the optimality of S^*, we have

$$\frac{w(S^*)}{f(|S^*|)} \geq \frac{w(S^* \setminus \{v\})}{f(|S^*|-1)}.$$

By using the fact that $w(S^* \setminus \{v\}) = w(S^*) - d_{S^*}(v)$, this inequality can be transformed to

$$d_{S^*}(v) \geq (f(|S^*|) - f(|S^*|-1)) \cdot \frac{w(S^*)}{f(|S^*|)}. \tag{2}$$

Let l be the smallest index that satisfies $S^* \subseteq S_l$. Note that $v_l \in S^*$. Using inequality (2), we have

$$\begin{aligned}\frac{w(S)}{f(|S|)} &\geq \frac{w(S_l)}{f(l)} = \frac{1}{2} \cdot \frac{\sum_{u \in S_l} d_{S_l}(u)}{f(l)} \geq \frac{1}{2} \cdot \frac{l \cdot d_{S_l}(v_l)}{f(l)} \geq \frac{1}{2} \cdot \frac{d_{S^*}(v_l)}{f(l)/l} \\ &\geq \frac{1}{2} \cdot \frac{f(|S^*|) - f(|S^*|-1)}{f(l)/l} \cdot \frac{w(S^*)}{f(|S^*|)} \geq \frac{1}{2} \cdot \frac{f(|S^*|) - f(|S^*|-1)}{f(n)/n} \cdot \frac{w(S^*)}{f(|S^*|)},\end{aligned}$$

where the second inequality follows from the greedy choice of v_l, the third inequality follows from $S_l \supseteq S^*$, and the last inequality follows from the monotonicity of $f(x)/x$. ◀

2.3 Examples

Here we observe the behavior of the approximation ratio of our algorithm for three convex size functions. We consider size functions *between* linear and quadratic because f-DS with any super-quadratic size function only produces constant-size optimal solutions. This follows from the inequality $\frac{f(2)/2}{f(|S^*|)/|S^*|^2} \geq 1$ (i.e., $f(2)/2 \geq f(|S^*|)/|S^*|^2$) by Lemma 4.

(i) The case where $f(x) = x^\alpha$ ($\alpha \in [1,2]$). The following corollary provides the approximation ratio of our algorithm.

▶ **Corollary 6.** *For f-DS with $f(x) = x^\alpha$ ($\alpha \in [1,2]$), our algorithm has an approximation ratio of $2 \cdot n^{(\alpha-1)(2-\alpha)}$.*

Proof. Let $s = |S^*|$. By Theorem 2, the approximation ratio is at most

$$\begin{aligned}\min\left\{\frac{2f(n)/n}{f(s)-f(s-1)}, \frac{f(2)/2}{f(s)/s^2}\right\} &= \min\left\{\frac{2n^{\alpha-1}}{s^\alpha - (s-1)^\alpha}, 2^{\alpha-1} \cdot s^{2-\alpha}\right\} \\ &\leq \min\left\{\frac{2n^{\alpha-1}}{s^{\alpha-1}}, 2 \cdot s^{2-\alpha}\right\} \leq 2 \cdot n^{(\alpha-1)(2-\alpha)}.\end{aligned}$$

The first inequality follows from the fact that $s^\alpha - (s-1)^\alpha = s^\alpha - (s-1)^{\alpha-1}(s-1) \geq s^\alpha - s^{\alpha-1}(s-1) = s^{\alpha-1}$. The last inequality follows from the fact that the first term and the second term of the minimum function are monotonically non-increasing and non-decreasing for s, respectively, and they have the same value at $s = n^{\alpha-1}$. ◀

Note that an upper bound on $2 \cdot n^{(\alpha-1)(2-\alpha)}$ is $2 \cdot n^{1/4}$, which is attained at $\alpha = 1.5$.

(ii) The case where $f(x) = \lambda x + (1-\lambda)x^2$ ($\lambda \in [0,1)$). The following corollary provides an approximation ratio of Algorithm 1, which is a constant for a fixed λ.

▶ **Corollary 7.** *For f-DS with $f(x) = \lambda x + (1-\lambda)x^2$ ($\lambda \in [0,1)$), Algorithm 1 with $k = 2$ has an approximation ratio of $(2-\lambda)/(1-\lambda)$. Furthermore, for any $\epsilon > 0$, Algorithm 1 with $k \geq \frac{2}{\epsilon} \cdot \frac{\lambda}{1-\lambda}$ has an approximation ratio of $2+\epsilon$.*

(iii) The case where $f(x) = x^2/(\lambda x + (1-\lambda))$ ($\lambda \in [0,1]$). This size function is derived by density function $\lambda \frac{w(S)}{|S|} + (1-\lambda)\frac{w(S)}{|S|^2}$. The following corollary provides an approximation ratio of our algorithm, which is at most 4.

▶ **Corollary 8.** *For f-DS with $f(x) = x^2/(\lambda x + (1-\lambda))$ ($\lambda \in [0,1)$), our algorithm has an approximation ratio of $4/(1+\lambda)$.*

3 Concave Case

In this section, we investigate f-DS with concave function f. A function $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is called *concave* if $f(x) - 2f(x+1) + f(x+2) \leq 0$ holds for any $x \in \mathbb{Z}_{\geq 0}$. We remark that $f(x)/x$ is monotonically non-increasing for x since we assume that $f(0) = 0$. Note that any optimal solution to f-DS with concave function f has a size larger than or equal to that of the densest subgraph. This follows from inequality (1) and the monotonicity of $f(x)/x$.

3.1 Dense Frontier Points

Here we define the dense frontier points and prove some basic properties. We denote by $\mathcal{P}$ the set $\{(|S|, w(S)) \mid S \subseteq V\}$. A point $(x,y) \in \mathcal{P}$ is called a *dense frontier point* if it is a unique maximizer of $y - \lambda x$ over $\mathcal{P}$ for some $\lambda > 0$. In other words, the extreme points of the upper convex hull of $\mathcal{P}$ are the dense frontier points. The densest subgraph is a typical subset of vertices corresponding to a dense frontier point. We prove that (i) for any dense frontier point, there exists some concave function f such that any optimal solution to f-DS with the function f corresponds to the dense frontier point, and conversely, (ii) for any strictly concave function f (i.e., f that satisfies $f(x) - 2f(x+1) + f(x+2) < 0$ for any $x \in \mathbb{Z}_{\geq 0}$), any optimal solution to f-DS with the function f corresponds to a dense frontier point.

We first prove (i). Note that each dense frontier point can be written as $(i, w(S_i^*))$ for some i, where S_i^* is a maximum weight subset of size i. Let $(k, w(S_k^*))$ be a dense frontier point and assume that it is a unique maximizer of $y - \hat{\lambda}x$ over $\mathcal{P}$ for $\hat{\lambda} > 0$. Consider the concave function f such that $f(x) = \hat{\lambda}x + w(S_k^*) - \hat{\lambda}k$ for $x > 0$ and $f(0) = 0$. Then, any optimal solution S^* to f-DS with the function f corresponds to the dense frontier point (i.e., $(|S^*|, w(S^*)) = (k, w(S_k^*))$ holds) because $w(S)/f(|S|)$ is greater than or equal to 1 if and only if $w(S) - \hat{\lambda}|S| \geq w(S_k^*) - \hat{\lambda}k$.

We next prove (ii). Let f be any strictly concave function. Let S_k^* be an optimal solution to f-DS with the function f, and take $\hat{\lambda}$ that satisfies $(f(k) - f(k-1)) \cdot \frac{w(S_k^*)}{f(k)} > \hat{\lambda} > (f(k+1) - f(k)) \cdot \frac{w(S_k^*)}{f(k)}$. Note that the strict concavity of f guarantees the existence of such $\hat{\lambda}$. Since f is strictly concave, we have $w(S_k^*) + \hat{\lambda}(|S| - k) \geq \frac{w(S_k^*)}{f(k)} \cdot f(|S|) \geq \frac{w(S)}{f(|S|)} \cdot f(|S|) = w(S)$ for any $S \subseteq V$, and the equalities hold only when $(|S|, w(S)) = (k, w(S_k^*))$. Thus, $(k, w(S_k^*))$ is a unique maximizer of $y - \hat{\lambda}x$ over $\mathcal{P}$, and hence is a dense frontier point.

Algorithm 3 LP-based algorithm

1: **for** $k \leftarrow 1, \ldots, n$
2: Solve LP_k and obtain an optimal solution (x^k, y^k)
3: Compute r_k^* that maximizes $w(S^k(r))/f(|S^k(r)|)$
4: **return** $S \in \{S^1(r_1^*), \ldots, S^n(r_n^*)\}$ that maximizes $w(S)/f(|S|)$

3.2 LP-Based Algorithm

We provide an LP-based polynomial-time exact algorithm. We introduce a variable x_e for each $e \in E$ and a variable y_v for each $v \in V$. For $k = 1, \ldots, n$, we construct the following linear programming problem:

$$\mathrm{LP}_k: \ \max. \sum_{e \in E} w(e) \cdot x_e \ \ \text{s.t.} \sum_{v \in V} y_v = k, \quad x_e \leq y_u, \ x_e \leq y_v \ (\forall e = \{u, v\} \in E),$$
$$x_e, y_v \in [0, 1] \ (\forall e \in E, \ \forall v \in V).$$

For an optimal solution (x^k, y^k) to LP_k and a real parameter r, we define a sequence of subsets $S^k(r) = \{v \in V \mid y_v^k \geq r\}$. For $k = 1, \ldots, n$, our algorithm solves LP_k to obtain an optimal solution (x^k, y^k), and computes r_k^* that maximizes $w(S^k(r))/f(|S^k(r)|)$. Note here that to find such an r_k^*, it suffices to check all the distinct sets $S^k(r)$ by simply setting $r = y_v^k$ for every $v \in V$. The algorithm returns $S \in \{S^1(r_1^*), \ldots, S^n(r_n^*)\}$ that maximizes $w(S)/f(|S|)$. For reference, we describe the procedure in Algorithm 3. Clearly, the algorithm runs in polynomial time.

In what follows, we demonstrate that Algorithm 3 obtains an optimal solution to f-DS with concave function f. The following lemma provides a lower bound on the optimal value of LP_k.

▶ **Lemma 9.** *For any $S \subseteq V$, the optimal value of $\mathrm{LP}_{|S|}$ is at least $w(S)$.*

We have the following key lemma.

▶ **Lemma 10.** *Let $S^* \subseteq V$ be an optimal solution to f-DS with concave function f, and let $k^* = |S^*|$. Furthermore, let (x^*, y^*) be an optimal solution to LP_{k^*}. Then, there exists a real number r such that $S^{k^*}(r)$ is optimal to f-DS with concave function f.*

Proof. For each $e = \{u, v\} \in E$, we have $x_e^* = \min\{y_u^*, y_v^*\}$ from the optimality of (x^*, y^*). Without loss of generality, we relabel the indices of (x^*, y^*) so that $y_1^* \geq \cdots \geq y_n^*$. Then we have

$$\begin{aligned}
\int_0^{y_1^*} w(S^{k^*}(r))dr &= \int_0^{y_1^*} \left(\sum_{e=\{u,v\} \in E} w(e) \cdot [y_u^* \geq r \text{ and } y_v^* \geq r] \right) dr \\
&= \sum_{e=\{u,v\} \in E} \int_0^{y_1^*} (w(e) \cdot [y_u^* \geq r \text{ and } y_v^* \geq r])\, dr \\
&= \sum_{e=\{u,v\} \in E} w(e) \cdot \min\{y_u^*, y_v^*\} \geq \sum_{e \in E} w(e) \cdot x_e^* \geq w(S^*),
\end{aligned}$$

where $[y_u^* \geq r \text{ and } y_v^* \geq r]$ is 1 if the condition in the square bracket is satisfied and 0

otherwise, and the last inequality follows from Lemma 9. Moreover, we have

$$\int_0^{y_1^*} f(|S^{k^*}(r)|)dr = \sum_{h=1}^{n} f(h)\cdot(y_h^* - y_{h+1}^*) = \sum_{h=1}^{n}(f(h)-f(h-1))\cdot y_h^*$$
$$\leq \sum_{h=1}^{k^*}(f(h)-f(h-1)) = f(k^*)-f(0) = f(k^*),$$

where we assume that $y_{n+1}^* = 0$, and the inequality holds by the concavity of f (i.e., $f(h+2)-f(h+1) \leq f(h+1)-f(h)$), $\sum_{h=1}^{n} y_h^* = k^*$, and $y_h^* \leq 1$.

Let r^* be a real number that maximizes $w(S^{k^*}(r))/f(|S^{k^*}(r)|)$ in $[0, y_1^*]$. Using the above two inequalities, we have

$$\frac{w(S^*)}{f(k^*)} \leq \frac{\int_0^{y_1^*} w(S^{k^*}(r))dr}{\int_0^{y_1^*} f(|S^{k^*}(r)|)dr} = \frac{\int_0^{y_1^*}\left(\frac{w(S^{k^*}(r))}{f(|S^{k^*}(r)|)}\cdot f(|S^{k^*}(r)|)\right)dr}{\int_0^{y_1^*} f(|S^{k^*}(r)|)dr}$$
$$\leq \frac{\int_0^{y_1^*}\left(\frac{w(S^{k^*}(r^*))}{f(|S^{k^*}(r^*)|)}\cdot f(|S^{k^*}(r)|)\right)dr}{\int_0^{y_1^*} f(|S^{k^*}(r)|)dr} = \frac{w(S^{k^*}(r^*))}{f(|S^{k^*}(r^*)|)}.$$

This completes the proof. ◀

Clearly, Algorithm 3 examines $S^{k^*}(r^*)$ as a candidate subset of the output. Therefore, we have the result.

▶ **Theorem 11.** *Algorithm 3 is a polynomial-time exact algorithm for f-DS with concave function f.*

By Lemma 10, for any concave function f, an optimal solution to f-DS with the function f is contained in $\{S^k(r) \mid k = 1, \ldots, n,\ r \in [0,1]\}$ whose cardinality is at most n^2. As shown above, for any dense frontier point, there exists some concave function f such that any optimal solution to f-DS with the function f corresponds to the dense frontier point. Thus, we have the following result.

▶ **Theorem 12.** *We can find a corresponding subset of vertices for every dense frontier point in polynomial time.*

3.3 Flow-Based Algorithm

Here we provide a combinatorial exact algorithm for unweighted graphs (i.e., $w(e) = 1$ for every $e \in E$). We first show that using the standard technique for fractional programming, we can reduce f-DS with concave function f to a sequence of submodular function minimizations. The critical fact is that $\max_{S\subseteq V} w(S)/f(|S|)$ is at least β if and only if $\min_{S\subseteq V}(\beta\cdot f(|S|) - w(S))$ is at most 0. Note that for $\beta \geq 0$, the function $\beta\cdot f(|S|) - w(S)$ is submodular because $\beta\cdot f(|S|)$ and $-w(S)$ are submodular [11]. Thus, we can calculate $\min_{S\subseteq V}(\beta\cdot f(|S|) - w(S))$ in $O(n^5(m+n))$ time using Orlin's algorithm [16], which implies that we can determine $\max_{S\subseteq V} w(S)/f(|S|) \geq \beta$ or not in $O(n^5(m+n))$ time. Hence, we can obtain the value of $\max_{S\subseteq V} w(S)/f(|S|)$ by binary search. Note that the objective function of unweighted f-DS may have at most $O(mn)$ distinct values since $w(S)$ is a nonnegative integer at most m. Thus, the procedure yields an optimal solution in $O(\log(nm)) = O(\log n)$ iterations. The total running time is $O(n^5(m+n)\cdot\log n)$.

To reduce the computation time, we replace Orlin's algorithm with a much faster algorithm that substantially extends a technique of Goldberg's flow-based algorithm for the densest

Algorithm 4 Flow-based algorithm

```
Construct the (unweighted) directed network (U, A)
{b_1, ..., b_r} = {p/f(q) | p = 0, 1, ..., m, q = 2, 3, ..., n} such that b_1 < ··· < b_r
i_min ← 1 and i_max ← r
while TRUE
    i ← ⌊(i_max + i_min)/2⌋
    Compute a minimum s–t cut (X, Y) in (U, A, w_{b_i})
    if the cost of (X, Y) is larger than w(V) then i_max ← i − 1
    else if the cost of (X, Y) is less than w(V) then i_min ← i + 1
    else return X ∩ V
```

subgraph problem [13]. The key technique is to represent the value of $\beta \cdot f(|S|) - w(S)$ using the cost of minimum cut of a certain directed network constructed from G and $\beta \geq 0$.

For a given graph $G = (V, E, w)$ and a real number $\beta \geq 0$, we construct a directed network (U, A, w_β) as follows. The vertex set U is defined by $U = V \cup P \cup \{s, t\}$, where $P = \{p_1, \ldots, p_n\}$. The edge set A is given by $A = A_s \cup A_t \cup A_1 \cup A_2$, where

$$A_s = \{(s, v) \mid v \in V\},\ A_t = \{(p, t) \mid p \in P\},$$
$$A_1 = \{(u, v), (v, u) \mid \{u, v\} \in E\},\ \text{and}\ A_2 = \{(v, p) \mid v \in V,\ p \in P\}.$$

The edge weight $w_\beta : A \to \mathbb{R}_{\geq 0}$ is defined by

$$w_\beta(e) = \begin{cases} d(v)/2 & (e = (s, v) \in A_s), \\ \beta \cdot k \cdot a_k & (e = (p_k, t) \in A_t), \\ 1/2\ (= w(\{u, v\})/2) & (e = (u, v) \in A_1), \\ \beta \cdot a_k & (e = (v, p_k) \in A_2), \end{cases}$$

where $d(v)$ is the degree of vertex v (i.e., $d(v) = |\{u \in V \mid \{u, v\} \in E\}|$), and

$$a_k = \begin{cases} 2f(k) - f(k+1) - f(k-1) & (k = 1, \ldots, n-1), \\ f(n) - f(n-1) & (k = n). \end{cases}$$

Note that $a_k \geq 0$ holds since f is a monotonically non-decreasing concave function.

The following lemma reveals the relationship between an s–t cut in (U, A, w_β) and the value of $\beta \cdot f(|S|) - w(S)$.

▶ **Lemma 13.** *Let (X, Y) be any s–t cut in the network (U, A, w_β), and let $S = X \cap V$. Then, the cost of (X, Y) is equal to $w(V) + \beta \cdot f(|S|) - w(S)$.*

From this lemma, we see that the minimum s–t cut value is $w(V) + \min_{S \subseteq V}(\beta \cdot f(|S|) - w(S))$. Therefore, for a given value $\beta \geq 0$, we can determine whether there exists $S \subseteq V$ that satisfies $w(S)/f(|S|) \geq \beta$ by checking the minimum s–t cut value is at most $w(V)$ or not. Our algorithm applies binary search for β within the possible objective values of f-DS (i.e., $\{p/f(q) \mid p = 0, 1, \ldots, m,\ q = 2, 3, \ldots, n\}$). For reference, we describe the procedure in Algorithm 4. The minimum s–t cut problem can be solved in $O(N^3/\log N)$ time for a graph with N vertices [8]. Thus, the running time of our algorithm is $O(\frac{n^3}{\log n} \cdot \log(mn)) = O(n^3)$ since $|U| = 2n + 2$. We summarize the result in the following theorem.

▶ **Theorem 14.** *Algorithm 4 is an $O(n^3)$-time exact algorithm for unweighted f-DS with concave function f.*

Finally, we remark that Algorithm 4 can be modified for weighted f-DS with concave function f so that a $(1 + \epsilon)$-approximate solution is obtained in $O(n^3 \cdot \log(1/\epsilon))$ time.

3.4 Greedy Peeling

Here we provide an approximation algorithm with much higher scalability. Specifically, we see that the greedy peeling (Algorithm 2) has an approximation ratio of 3 for f-DS with concave function f. As mentioned above, the algorithm runs in $O(m+n)$ time for unweighted graphs and $O(m + n \log n)$ time for weighted graphs. The proof of the following theorem relies on the monotonicity of $f(x)/x$ and the fact that the greedy peeling is a 3-approximation algorithm for Dal$k$S [1].

▶ **Theorem 15.** *The greedy peeling (Algorithm 2) has an approximation ratio of* 3 *for f-DS with concave function f.*

References

1 R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW'09*, pages 25–37, 2009.
2 A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. In *VLDB'12*, pages 574–585, 2012.
3 Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2):203–221, 2000.
4 G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):1–27, 2003.
5 A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *STOC'10*, pages 201–210, 2010.
6 F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *KDD'14*, pages 1316–1325, 2014.
7 M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX'00*, pages 84–95, 2000.
8 J. Cheriyan, T. Hagerup, and K. Mehlhorn. An $o(n^3)$-time maximum-flow algorithm. *SIAM J. Comput.*, 25(6):1144–1170, 1996.
9 Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *WWW'07*, pages 461–470, 2007.
10 E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. MotifCut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14):e150–e157, 2006.
11 S. Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*. Elsevier, 2005.
12 D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB'05*, pages 721–732, 2005.
13 A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.
14 S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.*, 36(4):1025–1071, 2006.
15 S. Khuller and B. Saha. On finding dense subgraphs. In *ICALP'09*, pages 597–608, 2009.
16 J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Math. Program.*, 118(2):237–251, 2009.
17 C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *KDD'13*, pages 104–112, 2013.
18 H. Yanagisawa and S. Hara. Axioms of density: How to define and detect the densest subgraph. Technical report, IBM Research – Tokyo, 2016.