Extracting Densest Sub-hypergraph with Convex Edge-weight Functions

Yi Zhou¹,², Shan Hu¹, and Zimo Sheng¹

¹ School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
² Zhejiang Jinggong Steel Building Group, Shaoxing 312030, Zhejiang, China

Abstract. The densest subgraph problem (DSG) aiming at finding an induced subgraph such that the average edge-weights of the subgraph is maximized, is a well-studied problem. However, when the input graph is a hypergraph, the existing notion of DSG fails to capture the fact that a hyperedge partially belonging to an induced sub-hypergraph is also a part of the sub-hypergraph. To resolve the issue, we suggest a function \( f_e : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0} \) to represent the partial edge-weight of a hyperedge \( e \) in the input hypergraph \( \mathcal{H} = (V, E, f) \) and formulate a generalized densest sub-hypergraph problem (GDSH) as \( \max_{S \subseteq V} \sum_{e \in E} f_e(\lvert S \cap e \rvert) \rvert S \rvert \). We demonstrate that, when all the edge-weight functions are non-decreasing convex, GDSH can be solved in polynomial-time by the linear program-based algorithm, the network flow-based algorithm and the greedy \( \frac{1}{r} \)-approximation algorithm where \( r \) is the rank of the input hypergraph. Finally, we investigate the computational tractability of GDSH where some edge-weight functions are non-convex.

Keywords: Densest subgraph problem · Hypergraph · Convex function.

1 Introduction

The densest subgraph problem (DSG) is a well-known problem in research communities of operations research, combinatorial optimization, data mining and so on. Given an edge-weighted graph \( \mathcal{G} = (V, E, w) \) with a vertex set \( V \), an edge set \( E \) and an edge-weight function \( w : E \to \mathbb{R}_{\geq 0} \), DSG asks us to maximize the density of the subgraph induced by a vertex set \( S \subseteq V \), i.e., \( \max_{S \subseteq V} \frac{\sum_{e \in E} w(e)}{\lvert S \rvert} \). Applications of DSG range from web community detection [8,6], network motif clustering [318] to information recommendation [17]. For solving DSG, there exists a network flow-based exact algorithm by Goldberg [95], a linear program-based algorithm by Charikar [4] and a linear-time \( \frac{1}{2} \)-approximation algorithm in [113].

* This work is partially supported by the National Natural Science Foundation of China under Grant No. 61802049.
On the other hand, hypergraph is attracting increasing attentions in recent years. The hypergraph is a generalization of the normal graph in which a hyperedge consists of arbitrary positive number of vertices. An edge-weighted hypergraph is defined as $H = (V, E, w)$ where $V$ is a vertex set, $E$ is a hyperedge set, and $w : E \rightarrow \mathbb{R}_{\geq 0}$ is an edge-weight function assigning each hyperedge a positive weight. The densest subgraph problem in an edge-weighted hypergraph, i.e. the densest sub-hypergraph problem (DSH), is known to be formulated as follows.

**Problem 1 (DSH as in [18,11])**. Given a hypergraph $H = (V, E, w)$, DSH asks for a sub-hypergraph induced by $S \subseteq V$ such that the average edge-weight of the sub-hypergraph i.e., $\frac{\sum_{e \in S} w(e)}{|S|}$, is maximized.

As far as we know, DSH was initially appeared as a generalization of the densest $r$-clique problem (D$^r$C) in [18]. Given a graph $G = (V, E)$, D$^r$C asks for a subset of vertices $S$ such that the average number of $r$-cliques (a clique of size $r$) induced by $S$ is maximized. Clearly, D$^r$C can be reduced to DSH by building a $r$-uniform hypergraph in which each hyperedge represents a $r$-clique. For D$^r$C, a polynomial exact algorithm and a $\frac{1}{r}$-approximation algorithm were introduced in [18], and a sampling algorithm was given in [16]. In [11], Hu et al finally remove the assumption that the input hypergraph is $r$-uniform and formalize DSH as Problem 1. They demonstrated that the linear program, network flow and approximation algorithms for DSH. Recently, a much faster $(1 - \epsilon)$ approximation algorithm based on max flow is given in [5] for DSH.

In this paper, we study a more generalized version of the densest sub-hypergraph problem rather than continue working with the existing model. We observed that in Problem 1 a hyperedge $e$ is counted as a part of sub-hypergraph induced by $S$ only when $e$ is a subset of $S$. However, in some graph applications like [15], if a hyperedge $e$ intersects with $S$, i.e., $e \not\subseteq S$ and $e \cap S \neq \emptyset$, $e$ is partially belong to the sub-hypergraph induced by $S$. Therefore, the fact that the weight of a sub-hypergraph induced by $S$ should contain a partial weight of the hyperedge that intersects with $S$ is not captured by the definition of DSH. In order to fix this issue, we introduce an edge-weight function $f_e : \{0, ..., |e|\} \rightarrow \mathbb{R}_{\geq 0}$ for each $e \in E$, and define the following generalized densest sub-hypergraph problem (GDSH).

**Problem 2 (GDSH in this paper)**. Given an edge-weighted hypergraph $H = (V, E, f)$ with vertex set $V$ and hyperedge set $E$, $f_e : \{0, ..., |e|\} \rightarrow \mathbb{R}_{\geq 0}$ being an edge-weight function for each $e \in E$, GDSH asks for a set of vertices $S \subseteq V$ such that i.e., $\frac{\sum_{e \in E} f_e(|e| \cap S)}{|S|}$, is maximized.

It is clear that GDSH generalizes the DSH problem. For example, if $H = (V, E, w)$ is the input hypergraph for DSH, we can build a hypergraph $H' = (V, E, f')$ such that $f'_e(i) = 0$ when $i < |e|$ and $f'_e(|e|) = w(e)$. Then, it is clear that the solution of GDSH with input graph $H'$ is the same as DSH with input graph $H$. In this sense, GDSH also generalizes existing densest subgraph problems like DSG and DrCP.
Since convex functions are ubiquitous in many applications, in the remaining of the paper, we investigate GDSH with focus on cases where all edge-weight functions are non-decreasing convex. It is clear that all the above problems like DSG, DrC and DSH are special cases of GDSH with non-decreasing convex edge-weight functions. We will use $n$ to denote the vertex number $|V|$, $m$ to denote the edge number $|E|$, $p$ to denote $\sum_{e \in E} |e|$ and $r$ to denote the rank $\max_{e \in E} |e|$ of the input hypergraph $\mathcal{H} = (V, \mathcal{E}, f)$. Note that $p = \sum_{v \in V} \deg_H(v)$ where $\deg_H(v)$ is the degree of vertex $v \in V$. We also use $\Psi = \sum_{e \in E} f_e(|e|)$ to denote the whole edge-weight of the hypergraph. Our main contributions for GDSH when all edge-weight functions are non-decreasing convex functions are summarized as follows.

- A linear program whose optimal value is equal to the maximum density of GDSH. The linear program has $O(mn!)$ inequalities but efficient oracles exist for separation. We show that an optimal solution of GDSH can be easily obtained by solving the linear program.

- A network flow-based algorithm which runs in $O(\mincut(p, pr) \log \Psi)$ time, $\mincut(N, M)$ representing the time of solving minimum s, t-cut in directed flow network with $N$ vertices and $M$ arcs. We also show the technique to obtain a $O(\mincut(p, pr) \log(\epsilon^{-1} \log(rm)))$ time $(1-\epsilon)$ approximation algorithm which removes the $\log \Psi$ factor.

- A greedy $\frac{1}{r}$ approximation algorithm with much faster running time $O(pr \log n)$. With a little relaxation of the greedy strategy, the greedy approximation algorithm can also run in logarithmic iterations under the parallel computing settings.

It is worth mentioning that the above three algorithms extend the linear program algorithm, network flow algorithm and greedy algorithm, respectively, in [11,4]. However, the extension is not trivial as. We only assume the non-decreasing and convexity properties of the edge-weight function in this work, contrary to the existing work that edge-weight functions are uniform and specifically given.

For completeness, we lastly study the computational tractability of GDSH when some edge-weight functions are non-convex. It turns out that when all edge-weight functions are non-decreasing concave, GDSH can be simply solved by selecting a (densest) vertex, when some edge-weight function are concave, GDSH is shown to be NP-hard by reduction from the max-cut problem.

2 Properties of Edge-weight Functions

Given $\mathcal{H} = (V, \mathcal{E}, f)$, the edge weight function $f_e$ is defined on discrete domain $0, \ldots, |e|$. We first assume that $f_e$ has non-decreasing properties for any $e \in \mathcal{E}$.

Property 1 (Non-decreasing). $f_e(i) \leq f_e(i + 1), \forall i \in \{0, \ldots, |e| - 1\}$
This property is a clearly natural in practice. Without loss of generality, we assume that $f_e(0) = 0$. If $f_e(0) \neq 0$, we can use $f'_e(i) = f_e(i) - f_e(0)$ to replace $f_e$ without changing the optimal solution of GDSH.

Aside from the non-decreasing property, we also discuss the convexity and concavity properties. As we know, convexity and concavity are common properties for many functions. They play important roles in characterizing the hardness of underlying optimization problems.

**Property 2 (Convexity).**

$$f_e(i) - f_e(i - 1) \leq f_e(i + 1) - f_e(i), \forall i \in \{1, ..., |e| - 1\}.$$

**Property 3 (Concavity).**

$$f_e(i) - f_e(i - 1) \geq f_e(i + 1) - f_e(i), \forall i \in \{1, ..., |e| - 1\}.$$

Given an $S \subseteq V$, we use $F(S) = \sum_{e \in E} f_e(|e \cap S|)$ to represent the weight of sub-hypergraph induced by $S$. Clearly, if $\forall e \in E, f_e$ is non-decreasing convex (concave), then $F(S)$ is a monotone supermodular (submodular) function in finite set $V$ (because submodularity and supermodularity are closed under non-negative linear combination). Let us recall the definitions of supermodularity and submodularity as bellow.

**Property 4 (Supmodularity).**

$$F(S \cup \{v\}) - F(S) \leq F(T \cup \{v\}) - F(T), \forall S \subseteq T \subset 2^V \text{ and } \forall v \in S, v \notin T$$

**Property 5 (Submodularity).**

$$F(S \cup \{v\}) - F(S) \geq F(T \cup \{v\}) - F(T), \forall S \subseteq T \subset 2^V \text{ and } \forall v \in S, v \notin T$$

Lastly, we assume that $f_e(i)$ is computed in constant time for any $i \in \{0, ..., |e|\}$. Thus, for any set $S \subseteq V$, $f_e(|e \cap S|)$ is computed in time $O(\min\{|e|, |S|\})$ and $F(S)$ is computed in time $O(p)$.

### 3 GDSH with Convex Edge-weight Functions

In this section, we investigate algorithms for solving GDSH when every edge-weight function is non-decreasing convex. Specifically, we show a linear program, a parametric network flow-based algorithm, and a fast greedy approximation in Section 3.1, 3.2 and 3.3 respectively.

#### 3.1 A Linear Program Approach

For a hyperedge $e \in E$, let $P_e$ be the set of all permutations of $e$. Given a permutation $\pi \in P_e$, $\pi(i) = v$ means that the $i$th vertex of permutation $\pi$ is $v$ and $v \in e$. Then, the linear program for GDSH, i.e., LP-GDSH, is given as follows.
maximize \( \sum_{e \in \mathcal{E}} y_e \) \hspace{1cm} \text{(LP-GDSH)}

s.t. \( \sum_{i=1}^{\lvert e \rvert} (f_e(i) - f_e(i-1))x_{\pi(i)} \geq y_e \) \hspace{1cm} \forall e \in \mathcal{E}, \forall \pi \in \mathcal{P}_e \quad (1)

\( \sum_{v \in V} x_v \leq 1 \)

\( x_v \geq 0, y_e \geq 0 \) \hspace{1cm} \forall v \in V, e \in \mathcal{E} \quad (2)

**Lemma 1.** Let \( \mathbf{x} \) be a feasible solution of LP-GDSH. Then, for any \( e \in \mathcal{E} \), we have \( \min_{\pi \in \mathcal{P}_e} \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi(i)} = \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi^*(i)} \) where \( \pi^* \in \mathcal{P}_e \) is a permutation that \( x_{\pi^*(1)} \geq x_{\pi^*(2)} \geq \ldots \geq x_{\pi^*(l(e))} \).

**Proof.** We justify the statement by contradiction. Assume that \( \pi' \in \mathcal{P}_e \) is a minimum permutation, i.e., \( \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi'(i)} = \min_{\pi \in \mathcal{P}_e} \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi(i)} \), but there exists \( i, j \) that \( 1 \leq i < j \leq l(e) \), \( x_{\pi'(i)} < x_{\pi'(j)} \). As \( f_e \) is a non-decreasing convex function, we have \( 0 \leq f_e(i) - f_e(i-1) \leq f_e(j) - f_e(i-1) \). Then, we have \( (f_e(i) - f_e(i-1))x_{\pi'(i)} + (f_e(j) - f_e(i-1))x_{\pi'(j)} < (f_e(i) - f_e(i-1))x_{\pi'(i)} + (f_e(j) - f_e(j-1))x_{\pi'(j)} \). In other words, we can decrease \( \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi'(i)} \) by exchanging \( \pi'(i) \) and \( \pi'(j) \), which contradicts the assumption that \( \pi' \) is the minimum permutation.

**Theorem 1.** The following statements hold for LP-GDSH.

1. For any \( S \subseteq V \), there is a feasible solution \( (\mathbf{x}, \mathbf{y}) \) of LP-GDSH such that \( \sum_{e \in \mathcal{E}} y_e = \frac{F(S)}{|S|} \).
2. Let \( \gamma^* \) be the optimal objective value of LP-GDSH. Then, there is a vertex \( \gamma \in V \) such that \( \gamma^* \leq \frac{F(S)}{|S|} \).

Therefore, the optimal solution of LP-GDSH is equal to the maximum density of GDSH.

**Proof.** **Proof of the first statement.** For any \( S \subseteq V \), we construct a \( \mathbf{x} \) such that \( x_v = \frac{1}{|S|} \) if \( v \in S \) and \( x_v = 0 \) otherwise. Clearly, \( \mathbf{x} \) satisfies Inequality \( \Box \).

We also construct \( \mathbf{y} \) with \( y_e = \frac{F(S)}{|S|} \) for any \( e \in \mathcal{E} \). Then \( \sum_{e \in \mathcal{E}} y_e \) is equal to \( \frac{F(S)}{|S|} \). Now, let use verify that this \( (\mathbf{x}, \mathbf{y}) \) satisfies Inequality \( \Box \). By Lemma 1, the left-hand side of Inequality \( \Box \) is at least \( \sum_{i=1}^{l(e)} (f_e(i) - f_e(i-1))x_{\pi^*(i)} \) where
\(\pi^* \in P_e\) satisfies \(\pi^*(i) > \pi^*(i + 1)\) for \(1 \leq i \leq n - 1\). Therefore,

\[
\sum_{i=1}^{\vert e \vert} (f_e(i) - f_e(i - 1))x_{\pi(i)} \geq \sum_{i=1}^{\vert e \vert} (f_e(i) - f_e(i - 1))x_{\pi^*(i)} = \frac{1}{\vert S \vert} \sum_{i=1}^{\vert e \cap S \vert} (f_e(i) - f_e(i - 1)) = y_e.
\]

Hence, the first statement holds.

**Proof of the second statement.** Let \((x^*, y^*)\) be an optimal solution of LP-GDSH. Define \(S_r = \{v : x_v^* \geq r\}\). We claim that there exists \(r \in [0, 1]\) such that \(\int F(S_r) d|S_r| \geq \gamma^*\). Assume that there is no such \(r\). Then we have \(\int F(S_r) d|S_r| < \gamma^*\) for any \(r \in [0, 1]\). That is to say, \(\int_0^\infty F(S_r) dr < \gamma^* \int_0^\infty |S_r| dr\).

On the other hand, we have

\[
\gamma^* \int_0^1 |S_r| dr = \gamma^* \sum_{v \in V} x_v^* \quad \text{and} \quad \int_0^1 F(S_r) dr = \sum_{e \in E} \int_0^1 f_e(|e \cap S|) dr = \sum_{e \in E} \left( \int_{i=1}^{\vert e \vert} (f_e(i) - f_e(i - 1))x_{\pi^*(i)}^* \right) = \sum_{e \in E} y_e^*.
\]

Note that the last equation is from the fact that \(y_e^*\) is equal to the minimum of \(\sum_{i=1}^{\vert e \vert} (f_e(i) - f_e(i - 1))x_{\pi^*(i)}^*\) for any permutation \(\pi \in P_e\).

Hence, we have \(\sum_{e \in E} y_e^* < \gamma^* \sum_{v \in V} x_v^*\) by assumption. However, this contradicts the condition that \(\gamma^*\) is optimal value. Therefore, we conclude that we can definitely find a \(r \in [0, 1]\) such that \(\int F(S_r) d|S_r| \geq \gamma^*\).

By the proof of Theorem 1, we can obtain the optimal solution to GDSH from the optimal LP solution \((x^*, y^*)\) by simply solving \(\max_{r \in [0, 1]} \frac{F(S_r)}{\vert S_r \vert}\). The number of inequalities in LP-GDSH is \(O(mr!)\), but this linear program can be
still solved in polynomial time because Inequality 1 can be efficiently separated by Lemma 1.

**Remark.** It is clear that LP-GDSH generalizes Charikar’s linear program [4] for DSG and Hu’s linear program [11] for DSH (Problem 1). A very recent work in [5] showed that the linear program technique can be also used for solving the densest supermodular subset problem which maximizes a supermodular set function of $S$ over $|S|$. Our GDSH can be a special case of this problem as $F(S)$ is supermodular if $f_e$ is convex. It is also interested to see that our LP-GDSH can be reduced to their linear program by summarizing Inequality 1 over all $e \in E$.

### 3.2 A Network Flow Algorithm

In this section, we introduce a parametric network flow-based approach algorithm, *GDSH-Flow*, for solving GDSH when $f_e$ is non-decreasing convex. *GDSH-Flow* is a standard binary search algorithm which finds the optimal density within range $[lb, ub]$. Initially, $lb = 0$ and $ub = \Psi$. *GDSH-Flow* testifies if there is a sub-hypergraph of density $\lambda = \frac{lb+ub}{2}$ by computing $\min_{S \subseteq V} (\gamma |S| - F(S))$. If $\min_{S \subseteq V} (\gamma |S| - F(S)) \leq 0$, there exists a sub-hypergraph of density $\lambda$, then we set $lb$ as $\lambda$. Otherwise, it indicates that $\lambda$ is larger than the optimal, we then decrease $ub$ to $\lambda$. In order to compute $\min_{S \subseteq V} (\gamma |S| - F(S))$ for any $\lambda$, we make use of the minimum cut from a directed network flow $G = (U, A, \lambda)$ where $U$ and $A$ are vertex set and arc set, respectively.

**Algorithm 1: **Exact network flow algorithm for GDSH.

```
1 GDSH-Flow(H)
2 begin
3  lb ← 0, ub ← \Psi
4 while ub > lb do
5    \lambda ← \frac{lb + ub}{2}
6    Build directed flow network $G = (U, A, \lambda)$
7    if the cost of min-cut $(X, Y)$ in $G$ is larger than $\Psi$ then
8      \text{ub} ← \lambda
9    else
10      \text{lb} ← \lambda
11 build directed flow network $G = (U, A, \lambda)$
12 compute minimum cut $(X, Y)$ from $G$
13 return $X \cap V$
```

To illustrate how to build $G = (U, A, \lambda)$, we need to first assume that for any $e \in E$, $f_e(i)$ returns integers for $i = 0, ..., |e|$. This restriction does not impose any loss of generality because we can always obtain integer values by
simultaneously scaling the edge-weight functions with an enough large value $M$ which is a multiple of 10. Then, $\mathcal{G}$ is built by the following steps.

- Build a source $s$, a sink $t$ in $\mathcal{G}$, and make a copy of every vertex of $V$ in $\mathcal{G}$.
- For a vertex $v \in U \setminus \{s,t\}$, add an arc $(v, t)$ with capacity $\lambda$ to $\mathcal{G}$.
- For a hyperedge $e$ in $\mathcal{H}$, assume $e = \{v_0, ..., v_{|e|-1}\}$. Then, add $|e|$ vertices $u_0^e, ..., u_{|e|-1}^e$ to $\mathcal{G}$. Also, add the following arcs to $\mathcal{G}$.
  - For each $i = 0, ..., |e|-1$, add an arc $(s, u_i^e)$ with capacity $(|e|-i)\alpha_i^e$.
  - For each $i = 0, ..., |e|-1$, $j = 0, ..., |e|-1$, add an arc $(u_i^e, v_j)$ with capacity $\alpha_i^e$ that
    \[
    \alpha_i^e = \begin{cases} 
    f_e(1) - f_e(0) & \text{if } i = 0, \\
    f_e(i+1) + f_e(i-1) - 2f_e(i) & \text{if } 0 < i < |e|.
    \end{cases}
    \]

An illustrative example of $\mathcal{G}$ is shown in Fig. 1. Clearly, $\alpha_i^e \geq 0$ for any integer $0 \leq i < |e|$ because $f_e$ is non-decreasing convex. We have the following statement for $\mathcal{G}$.

**Lemma 2.** Let $(X,Y)$ be a minimum $s,t$-cut in network $\mathcal{G} = (U,A,\lambda)$ such that $s \in X$ and $t \in Y$. Denote $S = X \cap V$. Then, the cost of $(X,Y)$ is equal to $\Psi + \lambda|S| - F(S)$.

**Proof.** First, by the definition of $\alpha_i^e$, we have,

\[
\sum_{i=0}^{j} \alpha_i^e = f_e(1) - f_e(0) + \sum_{i=1}^{j} ((f_e(i+1) - f_e(i)) - (f_e(i) - f_e(i-1)))
\]

\[
= f_e(j+1) - f_e(j)
\]

for any integer $j < |e|$.

Second, in network $\mathcal{G}$, $(X,Y)$ is a minimum $s,t$-cut, $S = X \cap V$ and an edge $e = \{v_0, ..., v_{|e|-1}\}$. Then, if $|S \cap e| > i$, then $u_i^e \in X$, otherwise, $(X,Y)$ is not a
minimum \( s, t \)-cut. In contrary, if \(|S \cap e| < i\), then \( u^e_c \in Y\). If \(|S \cap e| = i\), \( u^e_c \) can be either in \( X \) or \( Y \) without changing the cost of cut \((X,Y)\).

Let us denote \( s_e = |e \cap S| \) for simplicity. With the above observations, we finally get the cost of cut \((X,Y)\) as

\[
\sum_{e \in \mathcal{E}} (|e| - s_e) \sum_{i=0}^{s_e-1} a_i + \sum_{i=s_e}^{|e|-1} (|e| - i) a_i + \lambda |S|
\]

\[
= \sum_{e \in \mathcal{E}} (\sum_{i=0}^{s_e} a_i + \ldots + \sum_{i=0}^{|e|-1} a_i) + \lambda |S|
\]

\[
= \sum_{e \in \mathcal{E}} (f_e(s_e) + 1 - f_e(s_e) + \ldots + f_e(|e|) - f_e(|e| - 1)) + \lambda |S|
\]

\[
= \sum_{e \in \mathcal{E}} (f_e(|e|) - f_e(s_e)) + \lambda |S|
\]

\[
= \sum_{e \in \mathcal{E}} f_e(|e|) - F(S) + \lambda |S|
\]

, which ends the proof.

For a given \( \lambda \geq 0 \), Lemma 2 indicates that the cost of minimum \( s, t \)-cut \((X,Y)\) is \( \Psi + \min_{S \subseteq V} (\lambda |S| - F(S)) \). Thus, we can decide whether there exits an \( S \subseteq V \) such that \( \lambda |S| - F(S) < 0 \) by checking whether the cost of minimum \( s, t \)-cut of \( G \) is smaller than \( \Psi \). Therefore, the correctness of Alg. 1 is straightforward due to Lemma 2.

**Theorem 2.** If \( \forall e \in \mathcal{E} \) in \( \mathcal{H} = (V, \mathcal{E}, f) \), \( f_e \) is a non-decreasing convex function, then Alg. 1 solves GDSH in \( O(\min cut(p, pr) \log \Psi) \) time where \( \min cut(N, M) \) is the time of finding minimum \( s, t \)-cut from a directed flow graph with \( N \) vertices and \( M \) edges.

For any parameter \( \lambda \), the number of vertices and edges in \( G = (U, \mathcal{A}, \lambda) \) is \( n+2 \) and \( n+2+pr \), respectively. Therefore, the running time of this flow based algorithm is \( O(\min cut(p, pr) \log \Psi) \). For example, if we use the minimum \( s, t \)-cut algorithm in [10], which has running time \( \Theta(NM \log N^2 + N^4 \log N) \) and space \( O(M) \), the flow based algorithm runs in time \( O(npr \log \frac{(n+p)^2}{n+2+pr}) \) and space \( O(pr) \).

**Remark.** Readers who are familiar with submodular optimization can realize that \( h_\lambda(S) \) is monotone submodular when \( f_e \) is non-decreasing convex for any \( e \in \mathcal{E} \). Therefore \( \min_{S \subseteq V} h_\lambda(S) \) can be also solved via Submodular Function Minimization algorithms. The best-known submodular function minimization algorithm runs in time \( O(N^3(\log N)EO + N^4 \log(N)) \) where \( N \) is the number of elements and \( EO \) is the maximum time of evaluating the submodular function [14]. In our case, \( N = n \) and \( EO = O(p) \), the overall time is \( O(n^3 \log^2 n + n^4 \log(n)) \) which is not as efficient as our network flow based approach.
Further removing the log $\Psi$ factor Inspired by the technique in [12], we can obtain an algorithm with time polynomial to the size of input graph and $\epsilon^{-1}$ by a little modification of $GDSH-\text{Flow}$. The algorithm, as shown in Alg. 2, is named $GDSH-\text{Flow-}\epsilon$, which is $(1 - \epsilon)$ approximation.

Algorithm 2: $(1 - \epsilon)$ approximation algorithm for GDSH.

```plaintext
1 GDSH-Flow-$\epsilon$(H)
2 begin
3 Let $e_m = \arg \max_{e \in E} f_e(|e|)$
4 $lb \leftarrow f_{em}(|e_m|), ub \leftarrow \frac{\Psi f_{em}(|e_m|)}{\epsilon}$
5 while $lb < (1 - \epsilon)ub$ do
6 $\lambda \leftarrow \sqrt{lb \ast ub}$
7 Build directed flow network $G = (U, A, \lambda)$
8 if the cost of min-cut $(X, Y)$ in $G$ is larger than $\sum_{e \in E} f_e(|e|)$ then
9 $ub \leftarrow \lambda$
10 else
11 $lb \leftarrow \lambda$
12 build directed flow network $G = (U, A, lb)$
13 compute minimum cut $(X, Y)$ from $G$
14 return $X \cap V$
```

Theorem 3. If $\forall e \in E$ in $H = (V, E, f)$, $f_e$ is a non-decreasing convex function, then $GDSH-\text{Flow-}\epsilon$ is a $(1 - \epsilon)$ approximation algorithm with running time $O(\text{mincut}(p, pr) \log(\epsilon^{-1} \log(rm)))$ for DHSP.

Proof. The approximation ratio is clearly guaranteed by the stop condition of the algorithm. We mainly show that the number of while iterations is bounded by $\log(\epsilon^{-1} \log(rm))$. The crux is that $\frac{ub}{lb}$ is shrunk by a square root after every iteration. Let $i \in 1, ..., i^*$ denote the iteration number of the algorithm, $i^*$ is the last iteration number. In the $i$th iteration, let $lb^i$ and $ub^i$ be the lower and upper bound respectively.

First, it is clear that $\frac{ub^i}{lb^i} = \frac{|e_{m}|\Psi}{f_{em}(|e_m|)} \leq rm$. Then, we have

$$\frac{ub^{i+1}}{lb^{i+1}} \leq \max \left( \frac{ub^i}{\lambda^{i+1}}, \frac{\lambda^i}{lb^i} \right)$$

$$= \max \left( \frac{ub^i}{\sqrt{lb^i \ast ub^i}}, \frac{\sqrt{lb^i \ast ub^i}}{lb^i} \right)$$

$$= \sqrt{\frac{ub^i}{lb^i}}$$
Hence, we have $\frac{ub_{i+1}}{lb_{i+1}} \leq \left(\frac{ub^*_i}{lb^*_i}\right)^{\frac{1}{2}}$. On the other hand, we have $\frac{ub^*_i}{lb^*_i} \leq \frac{1}{1 - \epsilon}$. Therefore, $i^* \in O(\log(\log(rm)))$. As $\lim_{\epsilon \to 0} \frac{\epsilon}{\log(1/1-\epsilon)} = 1$, we have $i^* \in O(\log(\epsilon^{-1} \log(rm)))$.

Therefore, the the overall running time of is bounded by $O(\mincut(p, np) \log(\epsilon^{-1} \log(rm)))$.

### 3.3 A Fast $\frac{1}{r}$-approximation Algorithm

We further introduce $GDSH$-$\text{Approx}$ in Alg. 3 to approximate $GDSH$ when all the edge-weight functions are non-decreasing convex. By a little sacrifice on the accuracy, $GDSH$-$\text{Approx}$ is much faster than the above approaches.

**Algorithm 3:** Find the densest hyper-subgraph approximately.

```
1 \text{GDSH-Approx}(\mathcal{H})
2 \text{begin}
3 \quad S \leftarrow V, S' \leftarrow V
4 \quad \text{for } S \neq \emptyset \text{ do}
5 \quad \quad \text{Find } v \in \arg\min_{v \in S} (F(S) - F(S \setminus \{v\}))
6 \quad \quad S' \leftarrow S \setminus \{v\}
7 \quad \quad F(S') > \frac{F(S)}{|S|} \text{ then}
8 \quad \quad \quad S' \leftarrow S
9 \quad \text{return } S'
```

$GDSH$-$\text{Approx}$ maintains a subset of vertices $S$. In each of the consequent iterations, $GDSH$-$\text{Approx}$ identifies $v$, a vertex by which is removed from $S$, the decrease to the total edge-weight of the sub-hypergraph induced by $S$ is minimized. The algorithm starts with $S = V$ and stops when $S$ becomes empty. Of all the sets $S$ during the iterations, the one maximizing $\frac{F(S)}{|S|}$ is returned. To show the approximate ratio of $GDSH$-$\text{Approx}$, we first need the following observation.

**Lemma 3.** Given any $S \subseteq V$ in hypergraph $\mathcal{H} = (V, E, f)$, $F(S) \geq \frac{1}{r} \sum_{u \in S} (F(S) - F(S \setminus \{u\}))$.

**Proof.** For any hyperedge $e$, it is clear that if vertex $u \in e \cap S$, $f_e([e \cap S]) - f_e([e \cap (S \setminus \{u\})]) \leq f([e \cap S])$ and if $u \notin e \cap S$, $f_e([e \cap S]) - f_e([e \cap (S \setminus \{u\})]) = 0$. Hence, the following inequality holds.

$$\sum_{u \in S} (f_e([e \cap S]) - f_e([e \cap (S \setminus \{u\})])) \leq |S \cap e| f_e([e \cap S]) \leq rf_e([e \cap S])$$

By summarizing the above inequalities for all $e \in E$, we have

$$\sum_{u \in S} (F(S) - F(S \setminus \{u\}) \leq rF(S)$$

which completes the proof.
Then, we have the following result for $GDSH$-Approx.

**Theorem 4.** If $\forall e \in E$ in $H = (V, E, f)$, $f_e$ is a non-decreasing convex function, $GDSH$-Approx is a $\frac{1}{r}$-approximate algorithm.

**Proof.** Assume $S^* \subseteq V$ is a set of density $\lambda^*$ in $H$. Due to the optimality of $S^*$, for any $v \in S^*$,

$$\lambda^* = \frac{F(S^*)}{|S^*|} \geq \frac{F(S^* \setminus \{v\})}{|S^*| - 1} = \frac{F(S^*) - (F(S^*) - F(S^* \setminus \{v\}))}{|S^*| - 1}.$$  

With simple elementary transformations of the above inequality, we have $F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$.  

Now, let us consider the iteration of $GDSH$-Approx before the first vertex of $S^*$, say $v$, is removed. Call the current set of this iteration $S'$. So, $S^* \subseteq S'$. We have

$$\forall u \in S', F(S') - F(S' \setminus \{u\}) \geq F(S') - F(S' \setminus \{v\})$$

$$\geq F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$$

where the first inequality follows from greedy strategy in the algorithm and the second inequality follows from the supermodularity of $F(S)$ (since all edge-weight functions are convex). Now, combining Lemma 3, we conclude that

$$F(S') \geq \sum \frac{\lambda^*}{r} \geq \frac{|S'| \lambda^*}{r}$$

Therefore, $\frac{F(S')}{|S'|} \geq \frac{\lambda^*}{r}$. Since the algorithm returns a set of maximum density of all the iterations, the approximation ratio follows.

The number of iterations of Alg. 3 is $n$, the time to evaluate $F(S)$ is $O(p)$ in each iteration. Therefore, the running time of a simple implementation of this algorithm can be $O(n^2 p)$. Using a minimum-heap to maintain the vertices in $S$, we can reduce the time to $O(pr \log n)$.

**Further reducing the number of iterations** Currently, the number of iterations of $GDSH$-Approx is clearly $\Theta(n)$. Motivated by the work in [2], we provide a method of revising Alg. 3 such that the number of iterations reduces to the logarithmic scale. The new approximation algorithm is called $GDSH$-Para which is described in Alg. 4. $GDSH$-Para would be very efficient in processing large hy-
pergraphs in the parallel processing system because it only have a small number of dependable iterations.

**Algorithm 4:** A parallel algorithm to find the densest sub-hypergraph.

1. $GDSH$-Para($H$)
2. begin
3. $S \leftarrow V$, $S' \leftarrow V$
4. for $S \neq \emptyset$ do
5.  $\Delta \leftarrow \{v \in S : F(S) - F(S \setminus \{v\}) \leq r(1 + \epsilon) \frac{F(S)}{|S|}\}$
6.  $S \leftarrow S \setminus \Delta$
7.  if $\frac{F(S)}{|S|} > \frac{F(S')}{|S'|}$ then
8.  $S' \leftarrow S$
9. return $S'$

**Theorem 5.** If $\forall e \in E$ in $H = (V, E, f_e)$, $f_e$ is a non-decreasing convex function, then $GDSH$-Para is a $\frac{1}{r(1+\epsilon)}$ approximation with $O(\log \frac{1}{1+\epsilon} n)$ iterations.

**Proof.** By the proof of Theorem 4, for any vertex $v$ in an optimal solution $S^*$, $F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$ where $\lambda^*$ is the maximum density. Let us consider the pass before a first vertex $v$ from $S^*$ is removed in the algorithm. Denote the set as $S'$. Similar to the proof of Theorem 4, we still have $F(S') - F(S' \setminus \{v\}) \geq F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$ due to the supermodularity of $F$ and the optimality of $S^*$. Then,

$$\frac{F(S')}{|S'|} \geq \frac{F(S') - F(S' \setminus \{v\})}{r(1 + \epsilon)} \geq \frac{\lambda^*}{r(1 + \epsilon)}$$

where the first inequality is a direct result of the strategy in Line 5 in the algorithm. Hence, we obtain the approximation ratio.

We now estimate the maximum number of iterations. At each iteration, for the current set $S$,

$$F(S) \geq \sum_{u \in S} \frac{F(S) - F(S \setminus \{u\})}{r}$$

$$= \sum_{u \in \Delta} \frac{F(S) - F(S \setminus \{u\})}{r} + \sum_{u \notin \Delta} \frac{F(S) - F(S \setminus \{u\})}{r}$$

$$> r(1 + \epsilon) \frac{F(S)}{|S|} \frac{|S \setminus \Delta|}{r}$$

where the first inequality follows from Lemma 3, the second follows from the fact that any $u \in S \setminus \Delta$ satisfies $F(S) - F(S \setminus \{u\}) > r(1 + \epsilon) \frac{F(S)}{|S|}$. Thus, $|S \setminus \Delta| < \frac{1}{1+\epsilon} |S|$, indicating that the size of $S$ decreases by a factor at least $\frac{1}{1+\epsilon}$ during each iteration. Therefore, the algorithm stops in $O(\log \frac{1}{1+\epsilon} n)$ iterations.

Like GDSH-Approx, the space consumption of $GDSH$-Para is $O(n)$ if the minimum-heap data structure is used.
4 Non-convex Edge-weight Functions

In this section, we investigate GDSH when some of the edge-weight functions are not non-decreasing convex.

**Theorem 6.** If \( \forall e \in \mathcal{E} \) in \( \mathcal{H} = (V, \mathcal{E}, f) \), \( f_e \) is a non-decreasing concave function, the solution of GDSH-Approx is \( \{v\} \) where \( v = \arg\max_{u \in V} \sum_{e \in \mathcal{E}, u \in e} f_e(1) \).

**Proof.** If for every \( e \in \mathcal{E} \), \( f_e \) is non-decreasing concave, then \( F(S) \) is a monotone submodular function. Besides, \( F(\emptyset) = 0 \) because \( f_e(0) = 0 \). We first claim that, for any unit vertex set \( S \) that \( |S| = 1 \), \( S \subseteq T \subseteq V \), \( F(S) \geq \frac{F(T)}{|T|} \) holds. To verify the claim, let us assume \( S = \{v_1\} \) and \( T = \{v_1, ..., v_p\} \) without loss of generality, where \( p \geq 1 \) is the size of \( T \). By submodularity, we have

\[
F(T) - F(\{v_1, ..., v_{p-1}\}) \leq F(S) - F(\emptyset)
\]

\[
F(\{v_1, ..., v_p\}) - F(\{v_1, ..., v_{p-2}\}) \leq F(S) - F(\emptyset)
\]

\[
\vdots
\]

\[
F(\{v_1\}) - F(\emptyset) \leq F(S) - F(\emptyset)
\]

By adding up the above inequalities, we obtain \( F(T) - F(\emptyset) \leq p(F(S) - F(\emptyset)) \).

As \( F(\emptyset) = 0 \), we have \( F(S) \geq \frac{F(T)}{p} = \frac{F(T)}{|T|} \).

Now, it is not hard to see that the optimal solution to GDSH is a set with one vertex \( v = \arg\max_{u \in V} \sum_{e \in \mathcal{E}, u \in e} f_e(1) \).

On the other hand, if there are some edge-weight function \( f_e \) that is (non-monotonic) concave in \( \mathcal{H} = (V, \mathcal{E}, f) \), then we have the following NP-hardness result.

**Theorem 7.** Given a hypergraph \( \mathcal{H} = (V, \mathcal{E}, f) \), if for some \( e \in \mathcal{E} \), \( f_e \) is concave and for other \( e \in \mathcal{E} \), \( f_e \) is non-decreasing convex, then GDSH is NP-hard.

**Proof.** We reduce the well-known NP-hard problem, max-cut, to GDSH that edge-weight functions contain both convex and concave functions.

Given an unweighted graph \( \mathcal{H} = (V, \mathcal{E}) \) where \( n = |V| \), the max-cut problem asks to find \( T \subseteq V \) such that \( \text{cut}_{\mathcal{H}}(T) = |\{e \in \mathcal{E}: e \cap T = \emptyset\}| \) is maximized. To show the reduction, we build an edge-weighted hypergraph \( \mathcal{H}' = (V', \mathcal{E}', f') \) which includes both concave and convex edge-weight functions.

- Make two disjoint copies of \( \mathcal{H} \) of the same vertex and (hyper)edge sets.
  Denote the two copies as \( \mathcal{H}' = (V', \mathcal{E}', f') \) and \( \mathcal{H}'' = (V'', \mathcal{E}'', f'') \). For each hyperedge \( e \) in both \( \mathcal{H}' \) and \( \mathcal{H}'' \), set \( f_e(i) = 1 \) if \( i = 1 \) and \( f_e(i) = 0 \) for all \( i \neq 1 \).

- For vertex \( v \in V \), insert a hyperedge \( e_v = \{v', v''\} \) where \( v' \) and \( v'' \) are the two copies of \( v \) in \( \mathcal{H}' \) and \( \mathcal{H}'' \), respectively. For each hyperedge \( e_v \), set \( f_{e_v}(i) = n^2 \) if \( i = 1 \) and \( f_{e_v}(i) = 0 \) for all \( i \neq 1 \). Denote the set of these hyperedges as \( \mathcal{E}'' \).
– Add a hyperedge \( e_n = V' \cup V'' \) and assign the edge-weight function

\[
    f_{e_n}(i) = \begin{cases} 
        0, & \text{if } 0 \leq i < n \\
        n^2(i - n + 1), & \text{if } n \leq i \leq 2n.
    \end{cases}
\]

to hyperedge \( e_n \).

In summary, \( H^* \) includes a set of 2n vertices and four sets of hyperedges, \( E', E'', E''' \) and \( \{e_n\} \). The edge-weights functions of hyperedges in \( E', E'', E''' \) are concave but \( f_{e_n} \) is convex. Given \( H^* \), GDSH is to find a set \( S \subseteq V^* \) such that \( \frac{F(S)}{|S|} = \frac{\text{cut}_{H'}(S) + \text{cut}_{H''}(S) + n^2 \{|e \in E'': |e \cap S| = 1\}|}{|S|} + \frac{\text{cut}_{H'}(S) + \text{cut}_{H''}(S)}{|S|} + n^2 \frac{|S|}{n} + n^2 \)

On the other hand, if \( |S^*| = n \), \( \frac{F(S^*)}{|S^*|} = n^2 + n + \frac{\text{cut}_{H'}(S^*) + \text{cut}_{H''}(S^*)}{n} \) which is not smaller than \( n + n^2 \). Hence, \( |S^*| \geq n \).

Now suppose \( |S^*| > n \). Then,

\[
    \frac{F(S^*)}{|S^*|} = \frac{\text{cut}_{H'}(S^*) + \text{cut}_{H''}(S^*) + n^2 \{|e \in E'': |e \cap S^*| = 1\}| + n^2(|S^*| - n + 1)}{|S^*|} \leq \frac{\text{cut}_{H'}(S^*) + \text{cut}_{H''}(S^*) + n^2(2n - |S^*|) + n^2(|S^*| - n + 1)}{|S^*|} < n \frac{|S^*|}{|S^*|} + \frac{n^2(n + 1)}{|S^*|} \leq n + n^2
\]

Clearly, the density when \( |S^*| > n \) is still smaller than the density when \( |S^*| = n \). Therefore, the size of optimal solution \( S^* \) is n.

Now, we show that for any vertex \( u \in V \), the two copies \( u', u'' \in V^* \) satisfy either \( u' \in S^* \) and \( u'' \notin S^* \) or \( u' \notin S^* \) and \( u'' \in S^* \). Suppose that \( u' \) and \( u'' \) are both in \( S^* \). Then for hyperedge \( e = \{u', u''\} \), \( f_e(S^* \cap e) = 0 \). By removing \( u' \) or \( u'' \) from \( S^* \), we can get a larger density for set \( S^* \), which contradicts the fact that \( S^* \) is optimal. If we assume neither \( u' \) nor \( u'' \) in \( S^* \), we can also find a contradiction by adding \( u' \) or \( u'' \) to \( S^* \). Thus, either \( u' \) or \( u'' \) is in \( S^* \) but not both.
With the above two properties of the optimal solution $S^*$, we can state that the optimal solution $\frac{f(S^*)}{|S^*|} = n^2 + n + \frac{2\max_{S \subseteq V} \text{cut}_H(S)}{n}$. Therefore, the max-cut problem can be reduced to GDSH where the input graph includes both convex and concave edge-weight functions.

5 Conclusion

It is known that the (edge-weighted) densest sub-hypergraph problem is important in many data-mining applications. In this paper, we studied this problem with respect to different properties of the edge weight functions and formalized the Generalized Densest Sub-Hypergraph problem (GDSH). We show that GDSH with non-decreasing convex edge-weight functions can be solved efficiently by a linear program-based approach, a network flow-based approach and a fast greedy approximation algorithm. We also investigated GDSH for some other cases where edge-weight function are not always non-decreasing convex.

In the future, it it would be interesting to extend the study from multiple dimensions. First, one could consider more properties about the edge weight functions like submodularity, or, one could also investigate faster algorithms when the edge-weight functions are identical. Besides, the GDSH problem under some constraints would be another interesting topic. For example, the problem of finding densest subgraph with at least $k$ vertices is NP-hard, but 2-approximated was given in [13]. So, it could be possible to investigate the GDSH with different size constraint.

References

1. Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. Journal of Algorithms, 34(2):203–221, 2000.
2. Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. Proceedings of the VLDB Endowment, 5(5):454–465, 2012.
3. Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. Science, 353(6295):163–166, 2016.
4. Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 84–95. Springer, 2000.
5. Chandra Chekuri, Kent Quanrud, and Manuel R Torres. Densest subgraph: Submodularity, iterative peeling, and flow. In Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1531–1555. SIAM, 2022.
6. Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. IEEE Transactions on Knowledge and Data Engineering, 24(7):1216–1230, 2010.
7. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2009.
8. Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense communities in the web. In Proceedings of the 16th International Conference on World Wide Web, pages 461–470. ACM, 2007.
9. A. V. Goldberg. Finding a maximum density subgraph. *Tech. Report, Berkeley, CA, USA*, 1984.

10. Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

11. Shuguang Hu, Xiaowei Wu, and TH Chan. Maintaining densest subsets efficiently in evolving hypergraphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 929–938. ACM, 2017.

12. Yasushi Kawase and Atsushi Miyauuchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018.

13. Samir Khuller and Barna Saha. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 597–608. Springer, 2009.

14. Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

15. Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems*, pages 2308–2318, 2017.

16. Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824. ACM, 2015.

17. Md Farhadur Rahman, Weimo Liu, Saad Bin Suhaim, Saravanan Thirumuruganathan, Nan Zhang, and Gautam Das. Density based clustering over location based services. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 461–472. IEEE, 2017.

18. Charalampos Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1122–1132. International World Wide Web Conferences Steering Committee, 2015.