Fair densest subgraph across multiple graphs

Chamalee Wickrama Arachchi ✉ and Nikolaj Tatti

HIIT, University of Helsinki, firstname.lastname@helsinki.fi

Abstract. Many real-world networks can be modeled as graphs. Finding dense subgraphs is a key problem in graph mining with applications in diverse domains. In this paper, we consider two variants of the densest subgraph problem where multiple graph snapshots are given and the goal is to find a fair densest subgraph without over-representing the density among the graph snapshots. More formally, given a set of graphs and input parameter α, we find a dense subgraph maximizing the sum of densities across snapshots such that the difference between the maximum and minimum induced density is at most α. We prove that this problem is **NP**-hard and present an integer programming based, exact algorithm and a practical polynomial-time heuristic. We also consider a minimization variant where given an input parameter σ, we find a dense subgraph which minimizes the difference between the maximum and minimum density while inducing a total density of at least σ across the graph snapshots. We prove the **NP**-hardness of the problem and propose two algorithms: an exponential time algorithm based on integer programming and a greedy algorithm. We present an extensive experimental study that shows that our algorithms can find the ground truth in synthetic dataset and produce good results in real-world datasets. Finally, we present case studies that show the usefulness of our problem.

1 Introduction

The problem of dense subgraph discovery is an important tool in graph mining with applications in temporal pattern mining in financial markets [7], social network analysis [20], and biological system analysis [8]. On the other hand, multi-layer graph networks naturally exist in real-world, complex networks and have gained a significant amount of attention [3, 9, 12, 18, 20].

Among many definitions of a dense component, the ratio between the number of induced edges and the number of nodes has been popular since Goldberg [10] proposed a polynomial-time, exact algorithm to find the densest subgraph of a single graph and Charikar [4] proposed a greedy approximation algorithm.

Given a graph sequence, a natural way to find the densest subgraph is to find a common subgraph that maximizes the sum of densities in individual snapshots. This is done by first flattening the graphs into a single weighted graph and solving the weighted densest graph problem [20].

While this approach finds the densest subgraph, it may be the case that the densities are unevenly distributed. That is, the sum is dominated by the density of a single or few snapshots while the densities in the remaining snapshots are unfairly low or even 0.

In this paper, we consider a fair variant of the densest subgraph problem, where we require that the densities are close to each other across the snapshots. The topic of fairness is relatively new within the data mining and machine learning community which has been studied with classical problems like clustering, community detection, and anomaly detection [1, 2, 14, 15]. The fairness in the context of the densest subgraph problems is not well-studied yet.

More specifically, we consider two variants of a fair dense subgraph problem. In our first problem, we find a fair subgraph that maximizes the sum of the densities while being evenly distributed. To ensure that the total density is fairly distributed among the graph snapshots, we constrain the difference between the maximum and minimum density induced by the subgraph via an input parameter that specifies the maximum allowed density difference. In our second problem, given a pre-defined, minimum value for the total density as an input parameter, our goal is to minimize the gap between the maximum and minimum induced density over the graph sequence.

We show that both of our problems are **NP**-hard, and that the second problem is inapproximable. To solve our problems, we first propose two exact algorithms based on integer linear programming which run in exponential time. We also consider two heuristics that run in polynomial time in the size of the input graph.

The remainder of the paper is organized as follows. In Section 2, we provide preliminary notation along with the formal definitions of our optimization problems. Next, we prove **NP**-hardness of our problems in Section 3. All our algorithms and their running times are presented in Section 4. Related work is discussed in Section 5. In Section 6 we present an extensive experimental study both with synthetic and real-world datasets followed by a case study. Finally, Section 7 summarizes the paper and provides directions for future work.

2 Preliminary notation and problem definition

We begin by providing preliminary notation and formally defining our problem.

Our input is a sequence of graphs $\mathcal{G} = (G_1, \ldots, G_r)$, where each snapshot $G_i = (V, E_i)$ is defined over the same set of nodes. We often denote the number of nodes and edges by $n = |V|$.

Given a graph $G = (V, E)$, and a set of nodes $S \subseteq V$, we define $E(S) \subseteq E$ to be the subset of edges having both endpoints in S. We also write $m(S) = |E(S)|$. Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$ with $G_i = (V, E_i)$, we write $E(S, G_i) = E_i(S)$ and $m(S, G_i) = |E(S, G_i)|$.

As mentioned before, our goal is to find dense subgraphs in a temporal network, and for that we need to quantify the density of a subgraph. More formally, assume an unweighted graph $G = (V, E)$, and let $S \subseteq V$. We define the *density* $d(S, G)$ of a graph G_i induced by node set S, and extend this definition for a sequence of graphs $\mathcal{G} = (G_1, \ldots, G_k)$ as

$$d(S, G_i) = \frac{|E(S, G_i)|}{|S|} \qquad \text{and} \qquad d(S, \mathcal{G}) = \sum_{i=1}^{r} d(S, G_i) \quad .$$

We first state the problem of finding a common subgraph in a graph sequence which maximizes the sum of the densities proposed by Semertzidis et al. [20].

Problem 1 (Total densest subgraph problem (TDS)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$, find a common subset of vertices S, such that $d(S, \mathcal{G})$ is maximized.

This problem can be solved by first flattening the graph sequence into one weighted graph, where an edge weight is the number of snapshots in which an edge occurs. The problem is then a standard (weighted) densest subgraph problem that can be solved using the exact method given by Goldberg [10] in $\mathcal{O}(n(n+m)(\log n + \log r))$ time.

Next, we introduce our main problem where an additional fairness constraint of the induced densities is considered. To this end, we denote the difference between the maximum and minimum induced density as

$$\Delta(S, \mathcal{G}) = \max_i d(S, G_i) - \min_i d(S, G_i) \quad .$$

Given a sequence of graph snapshots and an input parameter α, our goal is to find a subset of vertices, such that the sum of the densities of subgraphs is maximized while maintaining the difference $\Delta(S, \mathcal{G})$ at most α.

Problem 2 (Fair densest subgraph problem (FDS)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$ and real number α, find a subset of vertices S, such that $d(S, \mathcal{G})$ is maximized and $\Delta(S, \mathcal{G}) \leq \alpha$.

Note that for $\alpha = 0$, the FDS problem is equivalent to finding a subgraph which induces exactly the same density over each snapshot while maximizing the total density. On the other hand, setting $\alpha = \infty$ reduces FDS to TDS.

Next, we present a minimization variant of FDS problem where given an input parameter σ, the goal is to find a subset of vertices S that minimizes the difference $\Delta(S, \mathcal{G})$ while inducing a total density of at least σ.

Problem 3 (The smallest difference densest subgraph problem (SDS)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$ and real number σ, find a common subset of vertices S, such that the density induced by S over $\mathcal{G}$ is at least σ and $\Delta(S, \mathcal{G})$ is minimized.

Finally, we state the minimum densest subgraph problem [12] where the goal is to find a common subgraph which maximizes the minimum density.

Problem 4 (Minimum densest subgraph (MDS)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$, find a common subset of vertices S, such that $\min_i d(S, G_i)$, the minimum density induced by S over $\mathcal{G}$ is maximized.

3 Computational complexity

In this section, we show that both of our problems are **NP**-hard. We saw in the previous section that if we set $\alpha = \infty$, then FDS reduces to TDS, which can be solved exactly in polynomial time. However, FDS is **NP**-hard when $\alpha = 0$.

Proposition 1. *FDS is* ***NP****-hard.*

Proof. We prove the hardness from k-Clique, a problem where, given a graph H, we are asked if there is a clique of size at least k.

Assume that we are given a graph $H = (V, E)$ with n nodes, $n \geq k$. We set $\alpha = 0$. The graph snapshot G_1 consists of the graph H and an additional set of k singleton vertices U. G_2 consists of a k-clique connecting the vertices in U.

We claim that there is a subset S yielding $d(S, \mathcal{G}) = (k-1)/2$ if and only if there is an k-clique in H.

Assume that there is a subset S yielding $d(S, \mathcal{G}) = (k-1)/2$. Since the value of objective is $(k-1)/2$, we have $d(S, G_1) = d(S, G_2) = (k-1)/4$. Let $S = W \cup T$ where $W \subseteq V$ and denotes the subset of vertices from H and $T \subseteq U$ is the subset of vertices from U in S.

Assume that $|W| < |T|$. Since $|T| \leq k$, $|W| < k$. The density induced on G_1 is bounded by $d(S, G_1) \leq \frac{\binom{|W|}{2}}{|T|+|W|} < \frac{\binom{|W|}{2}}{2|W|} < \frac{k-1}{4}$, which is a contradiction. Assume that $|W| > |T|$. Then the density induced on G_2 is bounded by $d(S, G_2) = \frac{\binom{|T|}{2}}{|T|+|W|} < \frac{\binom{|T|}{2}}{2|T|} = \frac{|T|-1}{4} \leq \frac{k-1}{4}$, again a contradiction. Therefore, $|W| = |T|$.

Consequently, $d(S, G_2) = (|T| - 1)/4$, implying that $|T| = k$. Finally, $\frac{k-1}{4} = d(S, G_1) = \frac{|E(S)|}{2k}$ implies that $|E| = \binom{k}{2}$, that is, W is a k-clique in H.

On the other hand, assume there is a clique C of size k in H. Set $S = C \cup U$. Immediately, $d(S, \mathcal{G}) = (k-1)/2$ proving the claim. □

A similar proof will show that SDS is **NP**-hard, and inapproximable.

Proposition 2. *SDS is* ***NP****-hard. Unless* ***P*** $=$ ***NP****, there is no polynomial-time algorithm with multiplicative approximation guarantee for SDS.*

Proof. We use the same reduction from k-Clique as in the proof of Proposition 1. We also set $\sigma = (k-1)/2$. If there is a clique C in H, then selecting $S = C \cup U$ yields $\Delta(S, \mathcal{G}) = 0$. On the other hand, if $\Delta(S, \mathcal{G}) = 0$, then $d(S, G_1) = d(S, G_2) = (k-1)/4$, and the argument in the proof of Proposition 1 shows that there must be a k-clique in H. In summary, the difference $\Delta(S, \mathcal{G}) = 0$ for a solution S if and only if there is a k-clique in H.

This also immediately implies that there is no polynomial-time algorithm with multiplicative approximation guarantee since this algorithm can be then used to test whether there is a set S with $\Delta(S, \mathcal{G}) = 0$. □

4 Algorithms

In this section, we present our algorithms for FDS and SDS problems. We present two exact algorithms based on fractional programming [6] and integer linear programming. Both algorithms may require exponential time but in practice can provide a solution in reasonable time for moderately-sized graphs. In addition, we propose two polynomial time heuristics.

4.1 Exact algorithm for FDS

We will present an integer programming based algorithm algorithm that finds an exact or near-optimal solution for FDS problem. To this end, let us first define the *edge* difference as

$$b(S, \mathcal{G}) = \max_i m(S, G_i) - \min_i m(S, G_i) \quad .$$

To solve FDS we consider the following auxiliary problem.

Problem 5 (FDS(γ)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$, and two numbers α, γ, find a subset of vertices $S \subseteq V$ maximizing $\sum_{i=1}^{r} m(S, G_i) - \gamma|S|$ such that $b(S, \mathcal{G}) \leq \alpha|S|$.

Next, we show the relationship between FDS(γ) and FDS. This connection is an example of fractional programming [6].

Proposition 3. *Let $S(\gamma)$ be the subgraph solving FDS(γ). Similarly, let S^* be the optimal solution for FDS. Write $\gamma^* = d(S^*, \mathcal{G})$. If $\gamma > \gamma^*$, then $S(\gamma) = \emptyset$. If $\gamma < \gamma^*$, then $S(\gamma) \neq \emptyset$ and $d(S(\gamma), \mathcal{G}) > \gamma$.*

Proof. Let $f(S) = \sum_{i=1}^{r} m(S, G_i)$. We know that

$$f(S(\gamma)) - \gamma|S(\gamma)| \geq f(\emptyset) - \gamma|\emptyset| = 0. \tag{1}$$

Let us first assume that $\gamma > \gamma^*$. If $S(\gamma) \neq \emptyset$, then Eq. 1 implies that $d(S(\gamma), \mathcal{G}) \geq \gamma > \gamma^*$, which contradicts the optimality of γ^*. Thus, $S(\gamma) = \emptyset$.

Next, let us assume that $\gamma < \gamma^*$. Then

$$f(S(\gamma)) - \gamma|S(\gamma)| \geq f(S^*) - \gamma|S^*| > f(S^*) - \gamma^*|S^*| = 0.$$

Thus $f(S(\gamma)) > \gamma|S(\gamma)|$, and consequently $S(\gamma) \neq \emptyset$ and $d(S(\gamma), \mathcal{G}) > \gamma$. □

Next, we use Proposition 3 to solve our main problem FDS. We find the (almost) largest γ for which FDS(γ) yields a non-empty solution. Then FDS(γ) for such γ yields (almost) optimal solution.

We can solve FDS(γ) with an integer linear program,

$$\begin{aligned}
\text{MAXIMIZE} \quad & \sum_{k=1}^{r} \sum_{ij \in E_k} x_{ij} - \gamma \sum_{i=1}^{n} y_i & & \\
\text{SUBJECT TO} \quad & x_{ij} \leq y_i & ij \in E \quad & (2) \\
& x_{ij} \leq y_j & ij \in E \quad & (3) \\
& x_{ij} \geq y_i + y_j - 1, & ij \in E \quad & (4) \\
& \sum_{ij \in E_k} x_{ij} - \sum_{ij \in E_\ell} x_{ij} \leq \alpha \sum_{i=1}^{n} y_i & k, \ell = 1, \ldots, r \quad & (5) \\
& x_{ij}, y_j \in \{0, 1\} \quad . & &
\end{aligned}$$

To see why this program solves FDS(γ), let $S \subseteq V$ be a solution to our FDS(γ), The indicator variable y_i denotes whether the node $i \in S$, and the indicator variable x_{ij} denotes whether both endpoints of edge ij are in S. Note that Constraints 2–4 force $x_{ij} = \min(y_i, y_j)$.

Furthermore, Constraint 5 ensures that $b(S, \mathcal{G}) \leq \alpha |S|$.

Proposition 3 allows to maximize γ with a binary search. Here, we choose the initial interval (L, U) by setting lower threshold $L = 0$ and upper threshold $U = r\frac{n-1}{2}$, and keep halving the interval until $|U - L| \leq \epsilon L$, where $\epsilon > 0$ is an input parameter. Finally, we return the solution of FDS(L) as the final solution to FDS. We refer to this algorithm as FDS-IP. Next, we show that FDS-IP yields $1/(1+\epsilon)$ approximation guarantee, or exact solution if ϵ is small enough.

Proposition 4. *Assume a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, $\alpha \geq 0$, and $\epsilon > 0$. Let γ be the score of the solution returned by FDS-IP and let γ^* be the optimal score of FDS. Then $\gamma \geq \gamma^*/(1+\epsilon)$. If $\epsilon \leq \frac{1}{rn^3}$, then $\gamma = \gamma^*$.*

Proof. Let L and U be the values of the interval when binary search is terminated. Note that $\gamma \geq L$ due to Proposition 3. We know that $U - L \leq \epsilon L$ and $L \leq \gamma^* \leq U$. Thus, $\gamma^* - L \leq U - L \leq \epsilon L$, or $\gamma^* \leq (1+\epsilon)L \leq (1+\epsilon)\gamma$.

To prove the second claim, note that $d(S, \mathcal{G})$ is a rational number with a numerator of at most n. Thus, either $\gamma = \gamma^*$ or $\gamma^* - \gamma > n^{-2}$. If $\epsilon \leq \frac{1}{rn^3}$, then $\gamma^* - \gamma \leq U - L \leq n^{-2}$. Consequently, $\gamma = \gamma^*$. □

FDS-IP requires $\mathcal{O}(\log rn - \log \epsilon)$ rounds, solving an integer linear program in each round. Note that solving an integer program is **NP**-hard [19].

We should point out that fractional programming was used by Goldberg [10] for finding the densest subgraph in a single graph. The key difference is the fairness constraint: without it FDS(γ) reduces to a minimum cut, which can be solved exactly in polynomial time.

4.2 Exact algorithm for SDS

Next, we will propose an exact algorithm to solve SDS. The approach is similar to the solver for FDS, that is, we will define an auxiliary problem, which can be solved with an integer program, and then use binary search to find the solution for SDS. More formally, we have the following auxiliary problem SDS(γ), and its relation to SDS.

Problem 6 (SDS(γ)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$, and two numbers α, γ, find a subset of vertices $S \subseteq V$ minimizing $b(S, \mathcal{G}) - \gamma |S|$, such that $m(S, \mathcal{G}) \geq \alpha |S|$.

Proposition 5. *Let $S(\gamma)$ be the subgraph solving SDS(γ). Similarly, let S^* be the optimal solution for SDS. Write $\gamma^* = \Delta(S^*, \mathcal{G})$. If $\gamma < \gamma^*$, then $S(\gamma) = \emptyset$. If $\gamma > \gamma^*$, then $S(\gamma) \neq \emptyset$ and $\Delta(S(\gamma), \mathcal{G}) < \gamma$.*

Proof. Since an empty set satisfies the constraints, we have $b(S(\gamma), \mathcal{G}) - \gamma |S(\gamma)| \leq 0$. Let us first assume that $\gamma < \gamma^*$. If $S(\gamma) \neq \emptyset$, then $\Delta(S(\gamma), \mathcal{G}) \leq \gamma < \gamma^*$ and $d(S(\gamma), \mathcal{G}) \geq \alpha$, which contradicts the optimality of γ^*. Thus, $S(\gamma) = \emptyset$.

Next, let us assume that $\gamma > \gamma^*$. Then

$$b(S(\gamma)) - \gamma |S(\gamma)| \leq b(S^*) - \gamma |S^*| < b(S^*) - \gamma^* |S^*| = 0.$$

Thus $b(S(\gamma)) < \gamma |S(\gamma)|$, consequently $S(\gamma) \neq \emptyset$ and $\Delta(S(\gamma), \mathcal{G}) < \gamma$. □

We can solve SDS(γ) with an integer linear program,

$$\begin{array}{lrl}
\text{MINIMIZE} & u - \ell - \gamma \sum_{i=1}^{n} y_i & \\
\text{SUBJECT TO} & x_{ij} \leq y_i & ij \in E \\
 & x_{ij} \leq y_j & ij \in E \\
 & x_{ij} \geq y_i + y_j - 1 & ij \in E \\
 & \sum_{ij \in E_k} x_{ij} \geq \ell & k = 1, \ldots, r \\
 & \sum_{ij \in E_k} x_{ij} \leq u & k = 1, \ldots, r \\
 & \sum_{k=1}^{r} \sum_{ij \in E_k} x_{ij} \geq \sigma \sum_{i=1}^{n} y_i & \\
 & x_{ij}, y_j \in \{0, 1\} & \\
 & u, \ell \geq 0 \quad . &
\end{array}$$

Here we introduce upper and lower threshold variables u and ℓ such that the difference $u - \ell$ matches to $b(S, \mathcal{G})$. Similar to FDS-IP, we search for the smallest γ such that the solution is non-empty, that is, we run a binary search and stop when $U - L \leq (1 + \epsilon)L$, and then use SDS($U$) as the final result. We refer to our algorithm as SDS-IP. The algorithm yields the following guarantees.

Algorithm 1: SDS-Grd($\mathcal{G}, \alpha$), finds greedily a subgraph S which minimizes $\Delta(S, \mathcal{G})$ while satisfying $d(S, \mathcal{G}) \geq \sigma$.

```
1 $S \leftarrow$ The solution of TDS;
2 while changes to $\Delta(S)$ do
3     Find a vertex $v$ which minimizes $\Delta(S)$ either by adding or removing while
        satisfying density constraint;
4     Add or remove $v$ from the set $S$;
5 return $S$;
```

Proposition 6. *Assume a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, $\sigma > 0$, and $\epsilon > 0$. Let γ be the score of the solution returned by SDS-IP and let γ^* be the optimal score of SDS. Then $\gamma \leq (1 + \epsilon)\gamma^*$. If $\epsilon \leq \frac{1}{n^3}$, then $\gamma = \gamma^*$.*

The proof is similar to the proof of Proposition 4, and therefore is omitted.

4.3 Greedy algorithm to find a good solution for SDS

A standard technique in designing a polynomial-time algorithm is to start from the integer program, in this case SDS-IP, relax the integrality constraints, solve the resulting linear program, and then devise a rounding step. In our experimentation, this approach was problematic as the resulting sets rarely satisfied the constraints. Consequently, we propose a greedy algorithm for SDS.

We start by solving TDS; let S be the resulting set. Note since $d(S, \mathcal{G})$ is optimal, $d(S, \mathcal{G}) \geq \sigma$ or there is no set satisfying the constraint.

We continue by trying to minimize $\Delta(S, \mathcal{G})$ either by adding or removing a vertex greedily, while satisfying the density constraint. We repeat the same process until the algorithm converges. The pseudo-code for the algorithm is given in Algorithm 1.

4.4 Greedy algorithm to find a good solution for FDS

Next, we present a two-phase greedy algorithm to find a good solution for FDS. Given an input parameter α, in the first phase of the algorithm, we search for a set S such that $\Delta(S, \mathcal{G}) \leq \alpha$.

More concretely, we repeatedly run SDS-Grd by varying σ between 0 and d_{tds}, the density of the solution for TDS, and choose a feasible set with the highest total density. As a candidate set for σ, we used $\left\{ \frac{i}{k} d_{tds} \mid i = 0, \ldots, k \right\}$, where k is a user parameter.

In the second phase of the algorithm, we start from our feasible set returned in the first phase, and at each iteration, we try to improve our density score by picking the best vertex to either add or delete until convergence. We refer to our algorithm as FDS-Grd.

4.5 Exact algorithm for MDS

Although heuristics have been proposed to MDS problem [12, 20], we propose an integer programming-based algorithm to find an exact or near-optimal solution which we refer to this algorithm as MDS-IP.

We introduce the following auxiliary problem.

Problem 7 (MDS(γ)). Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$, with $G_i = (V, E_i)$, find a common subset of vertices S, such that $\min_i m(S, G_i) - \gamma|S|$ is maximized.

We solve MDS(γ) with the following integer program

$$\begin{array}{lrll}
\text{MAXIMIZE} & & z - \gamma \sum_{i=1}^{n} y_i & \\
\text{SUBJECT TO} & & x_{ij} \leq y_i & ij \in E \\
& & x_{ij} \leq y_j & ij \in E \\
& \sum_{i,j \in E_k} & x_{ij} \geq z & k = 1, \ldots, r \\
& & x_{ij}, y_j \in \{0, 1\} & \\
& & z \geq 0 \quad . &
\end{array}$$

To see why this program solves MDS(γ), note that $x_{ij} = \min(y_i, y_j)$ and $z = \min_k m(S, G_k)$, where $S = \{i \mid y_i = 1\}$.

We search for maximal γ that induces non-empty solution with a binary search. Here, we set the initial interval (L, U) to $L = 0$ and $U = \frac{n-1}{2}$, and keep halving the interval until $|U - L| \leq \epsilon L$, where $\epsilon > 0$ is an input parameter. Finally, we return the solution of MDS(L) as the final solution to MDS.

The following result states the approximation guarantees of MDS-IP.

Proposition 7. *Assume a graph sequence $\mathcal{G} = (G_1, \ldots, G_r)$ and $\epsilon > 0$. Let γ be the score of the solution returned by MDS-IP and let γ^* be the optimal score of MDS. Then $\gamma \geq \gamma^*/(1 + \epsilon)$. If $\epsilon \leq \frac{1}{n^3}$, then $\gamma = \gamma^*$.*

The proof of the proposition is essentially the same as the proofs for FDS-IP, and is therefore omitted.

5 Related work

Given an undirected graph, the problem of finding the densest subgraph can be solved in polynomial time. To solve the problem Goldberg [10] proposed an exact algorithm based on fractional programming, binary search, and minimum-cut computations. Moreover, Charikar [4] proposed a linear program solving the problem, and a linear time, greedy 1/2-approximation algorithm.

Instead of a single graph, the densest subgraph problem has been extended to the case of multiple graphs. Jethava and Beerenwinkel [12] considered finding a common that maximizes the minimum density, an **NP**-hard problem as shown by Charikar et al. [5]. We refer to this problem as MDS problem. While LP-based and greedy algorithms proposed by Jethava and Beerenwinkel[12] and the greedy-like algorithms proposed by Semertzidis et al. [20] do not provide any theoretical approximation guarantees for all inputs, we propose an exact algorithm for MDS problem as an additional result.

Another problem variant for multiple graphs has been considered by Semertzidis et al. [20] where their objective was to maximize the sum of the densities. This problem is solvable in polynomial time by first flattening the graph sequence into one weighted graph and then considering it as the standard weighted densest subgraph instance. Arachchi and Tatti [3] and Rozenshtein et al. [18] considered extensions of the problem where the dense subgraph is allowed to vary across the snapshots.

Anagnostopoulos et al. [2] considered a single binary node-colored graph and constrained the subgraph to have a fair distribution of colors. Even with two colors, the problem is shown to be **NP**-hard [2]. Next, Miyauchi et al. [16] considered finding a fair densest subgraph of a node-colored graph with multiple colors. Note that our problem can be viewed as finding fair densest subgraph in an edge-colored graph. Here, we color each edge with a color unique to a snapshot, and collapse the sequence into one (multi-edged) graph. Since the constraints and the input information in these problems are different the existing methods cannot be used to solve our problem.

Alternative measures for the density have been also studied. One option is to use the proportion of edges, that is, $m(S)/\binom{|S|}{2}$. The issue with this measure is that a single edge yields the highest density of 1. Moreover, finding the largest graph with the edge proportion of 1 is equal to finding the largest clique, an inapppproximable problem [11]. Tsourakakis et el [23] proposed a score $m(S) - \alpha\binom{|S|}{2}$, leading to an **NP**-hard optimization problem. In addition, Tsourakakis [22] proposed a ratio of triangles and the nodes as the density measure. Like the density based on edges, optimizing this measure can be done in polynomial time but requires computing the existing triangles in a graph. Adopting these measures with fairness constraints provides a future line of work.

6 Experimental evaluation

The goal of this section is to experimentally evaluate our algorithms. To this end, we first generate a synthetic dataset with planted, fair dense component in each snapshot and test how well our algorithms discover the ground truth. Next, we study the performance of the algorithm on real-world temporal datasets. Finally, we present interpretative results from a case study.

We implemented the algorithms in Python[1] and performed the experiments using a 2.4GHz Intel Core i5 processor and 16GB RAM. In our experimental

[1] The source code is available at `https://version.helsinki.fi/dacs/`.

Table 1: Computational statistics from the experiments with the synthetic datasets. Let d_{tds} be the solution of TDS. Here, *constr.* relates to the input parameter α in FDS or the normalized parameter σ_{nrm} in SDS, d_{dis} is the discovered sum of densities, d_{min}, d_{max}, and d_a are the minimum, maximum, and average density induced by the discovered subgraph over the graph sequence, respectively, i gives the number of iterations of the algorithms, *Jacc.* is the Jaccard index between the solution set and ground truth set, $|S|$ provides the size of the solution, and *time* gives the computational time in seconds.

Algorithm	constr.	d_{dis}	d_{min}	d_{max}	d_a	Δ	i	$\|S\|$	Jacc.	time
FDS-IP	3.9	51.12	10.84	14.74	12.78	3.9	8	100	1	127
FDS-GRD	3.9	50.53	10.73	14.58	12.63	3.85	13	98	0.96	81
SDS-IP	0.69	51.12	10.84	14.74	12.78	3.9	8	100	1	159
SDS-GRD	0.69	51.17	10.59	15.03	12.79	4.44	97	103	0.95	9

evaluation, we used the Gurobi optimization solver in Python to solve the integer programs associated with FDS-IP and SDS-IP algorithms by setting $\epsilon = 0.01$. As densities vary across the datasets, for SDS problem, we use $\sigma = \sigma_{nrm} \times d_{tds}$ where d_{tds} is the solution of TDS. Recall that the greedy algorithm FDS-GRD requires finding a feasible set in the first phase, as discussed in Section 4.4. The search is done with SDS-GRD and varying $\sigma_{nrm} \in \left\{\frac{i}{k} \mid i = 0, \ldots, k\right\}$. We initially set $k = 20$. If we did not find a feasible set, we tested $k = 100$. This strategy was sufficient for our experiments.

We compare our results with the exact solution of the densest common subgraph problem which maximizes $\min_i d(S, G_i)$ [12]. We refer to this problem as MDS. We also compare our results to the total density induced by TDS [20], and the sum of densities of individual dense subgraphs [10].

6.1 Synthetic dataset

The synthetic dataset was generated as follows. The dataset consists of 4 graphs given as $\{G_1, \ldots, G_4\}$ with 200 vertices. We split the vertices into two groups U and W. For G_1 and G_2 we randomly placed $1500p_i$ edges connecting U, where p_i was sampled uniformly from the interval $[0.4, 1]$. We also randomly connected vertices W with 200 edges, as well as vertices U and W also with 200 edges. For G_3 and G_4 we randomly connected vertices with 6000 edges. The resulting graph sequence had 14 866 edges. Moreover, the total density and the density difference for the dense fair component U were $d(U, \mathcal{G}) = 51.12$ and $\Delta(U, \mathcal{G}) = 3.9$.

6.2 Results for the synthetic dataset

The densest subgraph, that is, the solution to TDS, contained the whole graph with the density 74.33 and the density difference of 22.84. We tested our algorithm by setting the constraints to match the fair dense component, that is, we

Table 2: Characteristics of real-world datasets. Here, n and m are the number of vertices and edges, r is the number of snapshots, d_{tds} is the solution of TDS, Δ_{tds} is the difference between the maximum and minimum density of the TDS solution, d_{ind} gives the sum of densities of individual densest subgraph from each graph snapshot, and d_{mds} and Δ_{mds} give the total density and the difference between the maximum and minimum density of the MDS solution, respectively.

Data	n	m	r	d_{ind}	d_{tds}	d_{mds}	Δ_{tds}	Δ_{mds}
Twitter-#	806	1 518	15	38.8	9.8	5.5	2	0.21
Hospital	75	1 885	5	41.68	30.29	17.76	6.93	2.47
Airports	417	3 588	37	83.75	24.54	16.34	2.98	1.26
Students	889	5 329	122	118.01	26.32	17.15	0.68	0.26
Tumblr	1 980	5 812	89	103.99	55.83	43.62	1.33	0.77
Facebook	4 117	8 646	104	88.65	14	5.75	0.5	0.07
Twitter-user	4 605	10 155	93	90.63	23	12.81	0.5	0.17

set $\alpha = 3.9$ for FDS and $\sigma_{nrm} = 0.69$ so that $\sigma = 51.12$ for SDS. We report our results in Table 1.

The Jaccard indices in Table 1 indicate that the exact algorithms FDS-IP and SDS-IP algorithms discovered the underlying fair dense subgraph. Next, we see that both FDS-Grd and SDS-Grd achieved a Jaccard index of 0.95 and 0.96, respectively, while being faster than their exact counterparts. The exact algorithms yielded solutions with slightly better scores than the greedy algorithms.

Finally, solving MDS yielded the fair dense component U since U for this data also had the largest minimum degree $\min_i d(S, G_i)$.

6.3 Real-world datasets

We consider 7 publicly available, real-world datasets. The details of the datasets are shown in Table 2. *Twitter-#* [21][2] is a hashtag network where nodes correspond to hashtags and edges corresponds to the interactions where two hashtags appear in a tweet. This dataset contains 15 such daily graph snapshots in total. *Facebook* [24][3] is a network of Facebook users in New Orleans regional community. It contains a set of Facebook wall posts among these users from 9th of June to 20th of August, 2006. *Students*[4] is an online message network at the University of California, Irvine. It spans over 122 days. *Twitter-user* [18][5] is a network of twitter users in Helsinki 2013. It contains a set of tweets that appear in each others' names. *Tumblr* [13][6] contains phrase or quote mentions appeared

[2] `https://github.com/ksemer/BestFriendsForever-BFF-`
[3] `https://networkrepository.com/fb-wosn-friends.php`
[4] `https://toreopsahl.com/datasets/\#online_social_network`
[5] `https://github.com/polinapolina/segmentation-meets-densest-subgraph`
[6] `http://snap.stanford.edu/data/memetracker9.html`

Table 3: Computational statistics from the experiments with real-world datasets using FDS-IP and FDS-GRD algorithms which are denoted as IP and GR, respectively. Here, α is the input parameter in FDS problem which is the minimum value of the allowed induced density difference, d_{sum} is the discovered sum of densities, Δ gives the difference between the minimum and maximum density induced by the solution set, i gives the number of iterations, *size* gives the size of the discovered subgraph, and *time* gives the computational time in seconds.

		d_{sum}		Δ		time		size			
Data	α	IP	GR	IP	GR	IP	GR	IP	GR	i_{IP}	i_{GR}
Twitter-#	0.3	6.45	1.75	0.3	0.25	14	2.44	20	4	18	3
	0.5	7.58	5	0.5	0.5	3	2.33	12	2	18	1
	0.7	7.92	5	0.69	0.5	4	2.19	13	2	18	1
Hospital	0.3	12.21	4.71	0.29	0.21	23	4.11	14	14	12	2
	0.5	13.86	5	0.5	0.5	23	1.48	14	16	12	5
	0.7	14.53	8.33	0.67	0.67	12	2.8	15	18	12	5
Airports	0.3	14.96	6.3	0.3	0.3	1 006	25.31	70	50	17	5
	0.5	17.02	9.88	0.5	0.5	127	30.95	54	40	17	8
	0.7	18.43	12.27	0.69	0.7	106	26.32	49	33	17	1
Students	0.3	22.65	17.83	0.3	0.29	28	102.35	40	24	19	2
	0.5	25.64	25.17	0.5	0.5	5	102.45	36	18	19	2
	0.7	26.32	26.32	0.68	0.68	4	92.65	19	19	19	5
Tumblr	0.3	29.41	12.86	0.3	0.29	74	14.34	27	7	20	2
	0.5	38.56	26	0.5	0.5	14	13.37	18	8	19	1
	0.7	45.92	37	0.69	0.67	7	13.19	13	3	19	1
Facebook	0.3	11.14	7	0.29	0.25	14	26.05	7	4	22	1
	0.5	14	14	0.5	0.5	11	29.37	2	2	22	1
	0.7	14	14	0.5	0.5	12	29.61	2	2	22	1
Twitter-user	0.3	23	11.5	0.5	0.25	15	47.46	4	4	21	1
	0.5	23	23	0.5	0.5	14	42.82	4	2	21	1
	0.7	23	23	0.5	0.5	14	44.91	4	2	21	1

in blogs and news media. It contains author and meme interactions of users over 3 months from February to April in 2009. Finally, the datasets, *Hospital* and *Airport* [17][7] represent multi-layer networks.

6.4 Results for the real-world datasets

First, let us compare the scores produced by the baselines TDS and MDS, given in Table 2. By definition, the density d_{tds} is always larger than d_{mds} as TDS optimizes the total density. Moreover, Δ_{tds} is always greater than Δ_{mds}, which is a side-effect of MDS optimizing the minimum density.

Next, we report our results for the FDS problem in Table 3. First, we see that the discovered scores d_{sum} by both algorithms increase as α increases. This

[7] `https://gitlab.com/densest/diverse`

Table 4: Computational statistics from the experiments with real-world datasets using using SDS-IP and SDS-Grd algorithms which are denoted as IP and GR, respectively. Here, σ is the input parameter in SDS where $\sigma = \sigma_{nrm} \times d_{tds}$, d_{sum} is the sum of densities induced by the discovered subgraph, Δ gives the difference between the minimum and maximum density, *size* gives the size of the discovered subgraph, *time* gives the computational time in seconds, and the columns i_{IP} and i_{GR} give the number of iterations for respective algorithms.

			Δ		d_{sum}		time		size			
Data	σ_{nrm}	σ	IP	GR	IP	GR	IP	GR	IP	GR	i_{IP}	i_{GR}
Twitter-#	0.3	2.94		0.33		3.33		0.12		3		5
	0.5	4.9	0.06	0.5	4.91	5	92	0.11	35	2	21	4
	0.7	6.86	0.36	0.75	7	7.25	26	0.11	14	4	18	4
Hospital	0.3	9.09	0	1.58	9.67	9.13	297	0.19	15	24	34	21
	0.5	15.14	0.93	2.73	15.2	15.17	32	0.14	15	30	13	13
	0.7	21.2	2.59	4.15	21.23	21.24	28	0.11	22	34	12	9
Airports	0.3	7.36		0.43		7.4		1.81		40		34
	0.5	12.27		0.7		12.27		1.03		33		19
	0.7	17.18	0.53	1.33	17.19	17.22	458	0.79	57	36	17	14
Students	0.3	7.89		0.13		7.93		3.72		40		30
	0.5	13.16		0.26		13.16		0.97		19		9
	0.7	18.42	0.17	0.3	18.42	18.48	556	1.02	64	23	19	9
Tumblr	0.3	16.75		0.38		17.63		0.92		8		7
	0.5	27.92	0.27	0.57	27.95	28.29	154	0.48	22	7	20	4
	0.7	39.08	0.53	0.8	39.16	39.6	61	0.27	19	5	19	2
Facebook	0.3	4.2		0.14		4.29		1.69		7		6
	0.5	7	0.07	0.25	7.02	7	962	0.84	60	4	23	3
	0.7	9.8	0.21	0.33	9.86	10	43	0.5	14	3	21	2
Twitter-user	0.3	6.9		0.14		7.14		1.67		7		6
	0.5	11.5		0.25		11.5		0.77		4		3
	0.7	16.1	0.19	0.5	16.12	23	132	0.48	26	2	22	2

is because as we increase α we let the difference in the maximum and minimum density vary within a larger interval so that our objective score may increase. We see that Δ is close or equal to α, and that FDS-IP is almost always larger than FDS-Grd. This is expected since FDS-IP searches for the densest subgraph more aggressively.

Next, we compare the scores and time of our exact and greedy algorithms. As expected, FDS-IP achieves a higher score than FDS-Grd. We can see that FDS-Grd runs faster than FDS-IP, but only for about half of the cases. The reason for FDS-Grd being slower is the search for a feasible set as the algorithm uses SDS-Grd as a subroutine. Both algorithms required a low number of iterations. The binary search in FDS-IP required solving at most 22 integer programs. The number of iterations for FDS-Grd is also low, at most 8.

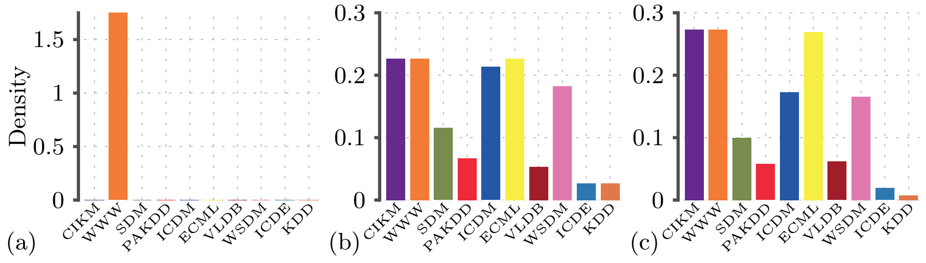

Fig. 1: Dense subgraphs among different conferences in *DBLP-CS*: (1a) TDS solution with density value of 1.75, (1b) FDS solution with density value of 1.36 for $\alpha = 0.2$, and (1c) SDS solution with density value of 1.4 for $\sigma = 0.8d_{tds}$.

Next, we report our results for the SDS problem in Table 4. Unlike with FDS-IP, in several cases, SDS-IP required an extensive amount of time: we stopped searching for the exact solution, if SDS-IP did not finish in 1 hour. First, we see that the exact solution tends to have smaller densities d_{sum} than the greedy since the exact algorithm optimizes Δ more aggressively.

Next, we compare the scores and running times of our exact and greedy algorithms as shown in the Δ and time columns, respectively. As expected the objective increases as we tighten the density constraint by increasing σ_{nrm}. Moreover, SDS-IP achieves smaller scores than SDS-GRD at the cost of computational time. Finally, we find that the number of iterations is reasonable in practice with both algorithms as shown in i_{IP} and i_{GR} columns.

6.5 Case study

DBLP-CS [17][7] is a co-authorship network where each edge represents a co-authorship attributed with one of the top data mining conferences. The graph contains 15 308 number of nodes and 10 publication venues. We sampled 12 000 edges from the original dataset.

We first solved TDS subgraph and observed that the densest subgraph represents only one conference which is the Web Conference (Fig. 1a). We are interested in finding the densest subgraph that contains publications of diverse publication venues without badly under-representing any venue. To that end, we applied FDS-IP($\alpha = 0.2$) algorithm. In Figure 1b we see that the densest subgraph attained by FDS-IP contains the edges from all publication venues in contrast to Figure 1a. The density of the unconstrained version was 1.75 while our algorithm achieved a fair dense subgraph with $\Delta = 0.2$ at the cost of having a density value of 1.36. Next, we applied SDS-IP($\sigma = 0.8d_{tds} = 1.4$) algorithm. As shown in Figure 1d, it achieves a density value of 1.4 while having with $\Delta = 0.27$. Finally, we solve MDS, yielding a solution with a density value of 1.18 and $\Delta = 0$. Note that MDS does not maximize the density $d(S, \mathcal{G})$, and does not have a mechanism for controlling the trade-off between the $\Delta(S, \mathcal{G})$ and

$d(S, \mathcal{G})$. On the other hand, by changing the parameters σ or α we can regulate the trade-off.

7 Concluding remarks

In this paper, we introduced two variants of dense subgraph discovery problem for graphs with multiple snapshots that take fairness into account.

More specifically, given an input parameter α, the goal of our first variant is to find a dense subgraph maximizing the sum of densities across snapshots such that the difference between the maximum and minimum induced density is at most α. We considered also the dual problem where given an input parameter σ, we find a subgraph that minimizes the gap between the maximum and minimum density induced by the subgraph while inducing at least σ amount of total density over the graph sequence.

We proved that both problems are **NP**-hard and proposed two exponential time, exact algorithms based on integer programming. We also proposed two polynomial time heuristics. We experimentally showed that our algorithms could find the ground truth in synthetic dataset and perform reasonably well in real-world datasets. Finally, we performed a study to show the usefulness of our problems.

The paper introduces several interesting directions for future work. In this paper, we considered the difference between the maximum and minimum density as the measure of fairness. However, we can use an alternative constraint that forces the density induced in each snapshot to be at least a portion of the average density. Another possible direction is adopting a different kind of density definition for our problem setting.

References

[1] Ahmadian, S., Epasto, A., Kumar, R., Mahdian, M.: Clustering without over-representation. In: KDD. pp. 267–275 (2019)
[2] Anagnostopoulos, A., Becchetti, L., Fazzone, A., Menghini, C., Schwiegelshohn, C.: Spectral relaxations and fair densest subgraphs. In: CIKM. pp. 35–44 (2020)
[3] Arachchi, C.W., Tatti, N.: Jaccard-constrained dense subgraph discovery. arXiv preprint arXiv:2308.15936 (2023)
[4] Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: APPROX. pp. 84–95 (2000)
[5] Charikar, M., Naamad, Y., Wu, J.: On finding dense common subgraphs (2018). https://doi.org/10.48550/ARXIV.1802.06361, `https://arxiv.org/abs/1802.06361`
[6] Dinkelbach, W.: On nonlinear fractional programming. Management science **13**(7), 492–498 (1967)

[7] Du, X., Jin, R., Ding, L., Lee, V.E., Thornton, J.H.: Migration motif: A spatial-temporal pattern mining approach for financial markets. In: KDD. pp. 1135–1144 (2009)

[8] Fratkin, E., Naughton, B.T., Brutlag, D.L., Batzoglou, S.: Motifcut: regulatory motifs finding with maximum density subgraphs. Bioinformatics **22**(14), e150–e157 (2006)

[9] Galimberti, E., Bonchi, F., Gullo, F., Lanciano, T.: Core decomposition in multilayer networks: Theory, algorithms, and applications. TKDD **14**(1), 1–40 (2020)

[10] Goldberg, A.V.: Finding a maximum density subgraph (1984)

[11] Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. In: STOC. pp. 627–636 (1996)

[12] Jethava, V., Beerenwinkel, N.: Finding dense subgraphs in relational graphs. In: ECMLPKDD. pp. 641–654 (2015)

[13] Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: KDD. pp. 497–506 (2009)

[14] Mehrabi, N., Morstatter, F., Peng, N., Galstyan, A.: Debiasing community detection: the importance of lowly connected nodes. In: ASONAM. pp. 509–512 (2019)

[15] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM computing surveys (CSUR) **54**(6), 1–35 (2021)

[16] Miyauchi, A., Chen, T., Sotiropoulos, K., Tsourakakis, C.E.: Densest diverse subgraphs: How to plan a successful cocktail party with diversity. In: KDD (2023)

[17] Oettershagen, L., Wang, H., Gionis, A.: Finding densest subgraphs with edge-color constraints. arXiv preprint arXiv:2402.09124 (2024)

[18] Rozenshtein, P., Bonchi, F., Gionis, A., Sozio, M., Tatti, N.: Finding events in temporal networks: segmentation meets densest subgraph discovery. KAIS **62**(4), 1611–1639 (2020)

[19] Schrijver, A.: Theory of Linear Integer Programming. John Wiley & Sons (Jun 1998)

[20] Semertzidis, K., Pitoura, E., Terzi, E., Tsaparas, P.: Finding lasting dense subgraphs. DMKD **33**(5), 1417–1445 (2019)

[21] Tsantarliotis, P., Pitoura, E.: Topic detection using a critical term graph on news-related tweets. In: EDBT/ICDT Workshops. pp. 177–182 (2015)

[22] Tsourakakis, C.: The k-clique densest subgraph problem. In: WWW. pp. 1122–1132 (2015)

[23] Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: KDD. pp. 104–112 (2013)

[24] Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: WOSN. pp. 37–42 (2009)