Mining Dense Subgraphs with Similar Edges

Polina Rozenshtein[1]✉, Giulia Preti[2], Aristides Gionis[3], and Yannis Velegrakis[4]

[1] Institute of Data Science, National University of Singapore, Singapore
`idspoli@nus.edu.sg`
[2] ISI Foundation, Italy
[3] KTH Royal Institute of Technology, Sweden
[4] Utrecht University, The Netherlands

Abstract. When searching for interesting structures in graphs, it is often important to take into account not only the graph connectivity, but also the metadata available, such as node and edge labels, or temporal information. In this paper we are interested in settings where such metadata is used to define a similarity between edges. We consider the problem of finding subgraphs that are *dense* and whose edges are *similar* to each other with respect to a given similarity function. Depending on the application, this function can be, for example, the Jaccard similarity between the edge label sets, or the temporal correlation of the edge occurrences in a temporal graph.
We formulate a Lagrangian relaxation-based optimization problem to search for dense subgraphs with high pairwise edge similarity. We design a novel algorithm to solve the problem through parametric MIN-CUT [15, 17], and provide an efficient search scheme to iterate through the values of the Lagrangian multipliers. Our study is complemented by an evaluation on real-world datasets, which demonstrates the usefulness and efficiency of the proposed approach.

1 Introduction

Searching for densely-connected structures in graphs is a task with numerous applications [1, 7, 10, 23] and extensive theoretical work [4, 16, 19]. A densely-connected subset of nodes may represent a community in a social network, a set of interacting proteins, or a group of related entities in a knowledge base. Given the relevance of the problem in different applications, a number of measures have been used to capture graph density, including average degree [19], quasi-cliques [23], and k-clique subgraphs [22].

Often, however, real-world graphs have attributes associated with their edges, which describe how nodes are related with each other. This is common in social networks, where we can distinguish multiple types of relationships between individuals (friends, family, class-mates, work, etc.), as well as several types of interactions (likes, messages, and comments). Similarly, a communication network records information that describes the communication patterns between its nodes, the volume of data exchanged between two nodes, or the level of congestion at a given link, as a function of time.

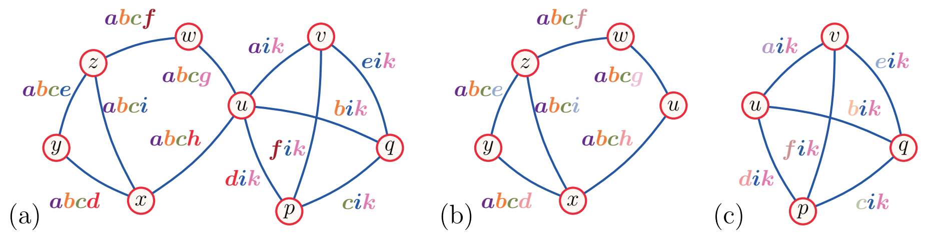

Fig. 1. (a) Input graph G with edge labels; (b) subgraph G_B of users $B = \{x, y, z, w, u\}$, where each edge pair shares 3 out of 4 labels; (c) clique G_C of users $C = \{u, v, p, q\}$, where each edge pair shares 2 out of 3 labels.

Incorporating this rich information into standard graph-mining tasks, such as dense-subgraph mining, can provide a better understanding of the graph, and enable the discovery of clear, cohesive, and homogeneous groups and patterns [14]. For instance, a group of hashtags that form a dense subgraph in the Twitter's hashtag co-occurence network becomes more meaningful for a social scientist if those hashtags are also correlated in time, as they likely indicate a recurrent topic of discussion, or an emerging trend.

In this paper, we study a general graph-mining problem where the input is a graph $G = (V, E)$ and a function $s : E \times E \to \mathbb{R}_{\geq 0}$ that measures similarity between pairs of edges. We do not restrict the choice of edge similarity s, meaning that it can be defined using any type of information that is available about the edges. For example, for a graph with edge labels, the similarity of two edges can be defined as the Jaccard similarity between their label sets, while for a temporal graph, where edges are active in some timestamps and inactive in others, the similarity can be defined as the temporal correlation between the edge time series. Given a similarity function, we are interested in finding *dense* subgraphs whose edges are *similar* to each other. Consider the following example.

Example. As a toy example, Figure 1(a) illustrates a portion of a social network, where a set of labels is available for each connection, describing the topics on which the two users have interacted with each other. Figures 1(b) and 1(c) highlight two dense subgraphs G_B and G_C, represented by the sets of users $B = \{x, y, z, w, u\}$ and $C = \{u, v, p, q\}$, respectively. The graph G_C is denser than G_B (G_C is a clique), meaning that the users in C have interacted more. However, the edges of G_B have more labels in common than those of G_C (3 out of 4 per edge pair, versus 2 out of 3), meaning that the users in B share more topics of interest. This example shows that when multiple metrics of interest are taken into consideration, some solutions may optimize some of the metrics, while other solutions may optimize the other metrics. For example, an advertiser may be interested in finding both tighter groups of users and highly similar groups of users, because the first ones have more connections and thus they can influence more other users in the group, while the second ones have more interests in common and thus they are more likely to like similar products. □

The previous example brings an interesting trade-off: some subgraphs have higher density, while other have higher edge similarity. This is a typical situation in *bi-criteria optimization* [12]. A common approach to study such problems is by using a Lagrangian relaxation, i.e., combining the two objectives into a weighted sum and solving the resulting optimization problem for different weights. We adopt this approach and combine the density and the edge-similarity of the subgraph induced by an edge set. Then, we reformulate the problem and design a novel efficient algorithm to solve the relaxation based on *parametric minimum cut* [15, 17]. We explore possible density-similarity trade-offs and provide an efficient search procedure through the values of the Lagrangian multipliers.

We demonstrate experimentally that our method finds efficiently a set of solutions on real-world datasets. A wide range of the weighting parameter effectively controls the trade-off between similarity and density. Additionally, we present a case study where we explore the properties of the discovered subgraphs.

All omitted proofs can be found in the Supplementary Material.

2 Problem Formulation

We consider an undirected graph $G = (V, E)$ with node set V and edge set E. All our algorithms extend to weighted graphs, but for simplicity of presentation we discuss the unweighted case. To avoid degenerate cases, we assume that G has at least 2 edges. We consider subsets of edges and edge-induced subgraphs:

Definition 1 (Edge-induced subgraph). *Let $G = (V, E)$ be an undirected graph and X a subset of edges. The subgraph $G(X) = (V(X), X)$ of G is induced by X, where $V(X)$ contains all the nodes that are endpoints of edges in X.*

We define the density of an edge-induced subgraph as the standard half of average degree or the number of edges divided by the number of nodes [9, 19]:

Definition 2 (Density). *Given an undirected graph $G = (V, E)$ and a set of edges $X \subseteq E$, the density of the edge-induced graph $G(X) = (V(X), X)$ is defined as*

$$D(G(X)) = \frac{1}{2}\frac{\sum_{u \in V(X)} deg(u)}{|V(X)|} = \frac{|X|}{|V(X)|},$$

where $deg(u)$ denotes the degree of a node $u \in V$. We refer to $D(G(X))$ as the density of the set of edges $X \subseteq E$, and denote it by $D(X) = D(G(X))$.

We assume that the graph G is equipped with a non-negative *edge similarity function* $s : E \times E \to \mathbb{R}_{\geq 0}$. We define the *total edge similarity* of an edge e as $s_{total}(e, X) = \sum_{e_i \in X \wedge e \neq e_i} s(e, e_i)$. We then define the *subgraph edge similarity* of an edge-induced subgraph as half of the average total edge similarity:

Definition 3 (Subgraph edge similarity). *The similarity of a set of edges X with at least 2 edges is defined to be*

$$S(X) = \frac{1}{2}\frac{\sum_{e \in X} s_{total}(e, X)}{|X|} = \frac{1}{|X|}\sum_{\{e_i, e_j\} \in X^2} s(e_i, e_j),$$

where X^2 is the set of all the unordered pairs of edges in X, i.e., $X^2 = \{\{e_i, e_j\} \mid e_i, e_j \in X$ with $e_i \neq e_j\}$. If $|X| \leq 1$, we set $S(X) = 0$.

In this paper we look for edge-induced subgraphs that have *high density* and *high subgraph edge similarity.* Note that the more common definition of node-induced subgraphs is not suitable for our problem setting, because a solution to our problem is not defined by a node set. Indeed, excluding some edges from a node-induced subgraph may lead to a subgraph, which is less dense, but have edges more similar to each other.

As shown in Figure 1, there may not exist solutions that optimize the two objectives at the same time. One possible approach is to search for subgraphs whose density and subgraph edge similarity exceed given thresholds. However, setting meaningful thresholds requires domain knowledge, which may be expensive to acquire. Here, we rely on a common approach to cope with bi-criteria optimization problems, namely to formulate and solve a Lagrangian relaxation:

*Problem 1 (*DSS*).* Given an undirected graph $G = (V, E)$ with an edge-similarity function $s : E \times E \to \mathbb{R}_{\geq 0}$ and a non-negative real number $\mu \geq 0$, find a subset of edges $X \subseteq E$, that maximizes the objective $O_\mu(X \mid \mu) = S(X) + \mu\, D(X)$.

3 Proposed Method

We start describing our solution by reformulating the DSS problem. The reformulation will allow us to use efficient algorithmic techniques. We alter the DSS objective by substituting the density term D with the inverse negated term $-1/D$. Without loss of generality, we require that the solution edge set X contains at least one edge. The resulting problem is the following.

*Problem 2 (*DSS-INV*).* Given an undirected graph $G = (V, E)$ and a non-negative real number $\lambda \geq 0$, find a subset of edges $X \subseteq E$, with $|X| \geq 1$, that maximizes the objective $O_\lambda(X \mid \lambda) = S(X) - \lambda/D(X)$.

For shorthand, we denote $-1/D$ as $\bar{D}$. We first show that DSS can be mapped to DSS-INV, so that optimal solutions for the one problem can be found by solving the other, with parameters μ and λ appropriatelly chosen. Then, we focus on solving the DSS-INV problem.

Proposition 1. *An edge set X^* is an optimal solution for* DSS *with parameter μ if and only if X^* is an optimal solution for* DSS-INV *with $\lambda = D^2(X^*)\mu$.*

The mapping provided in Proposition 1 guarantees that a solution to DSS with a parameter μ can be found by solving DSS-INV with a corresponding parameter λ. A drawback is that to construct an DSS-INV instance for a given DSS instance with a fixed μ we need to know the density of DSS's solution $D(X^*)$. However, in general, the Lagrangian multiplier μ is often not known in advance, and the user needs to experiment with several values and select the setting leading to an interesting solution. In such cases, arguably, there is no difference between experimenting with μ for DSS or with λ for DSS-INV. Furthermore, if the value of μ is given, we will show that our solution provides an efficient framework to explore the solution space of DSS-INV for all possible values of λ, and identify the solutions for the given value of μ.

3.1 Fractional Programming

Following the connection established in the previous section, our goal is therefore to solve problem DSS-INV for a given value of λ. We use the technique of *fractional programming*, based on the work by Gallo et al. [15]. For completeness, we review the technique. We first define the *fractional programming* (FP) problem:

*Problem 3 (*FP*).* Given an undirected graph $G = (V, E)$, and edge set functions $F_1 : 2^E \to \mathbb{R}$ and $F_2 : 2^E \to \mathbb{R}_{\geq 0}$, find a subset of edges $X \subseteq E$ so that $c(X) = \frac{F_1(X)}{F_2(X)}$ is maximized.

The following problem, which we call Q, is closely related to the FP problem:

*Problem 4 (*Q*).* Given an undirected graph $G = (V, E)$, edge set functions $F_1 : 2^E \to \mathbb{R}$ and $F_2 : 2^E \to \mathbb{R}_{\geq 0}$, and a real number $c \in \mathbb{R}$, find a subset of edges $X \subseteq E$ so that $Q(X \mid c) = F_1(X) - cF_2(X)$ is maximized.

The key result of fractional programming [15] states that:

Proposition 2 (Gallo et al. [15]). *A set X^* is an optimal solution to an instance of the* FP *problem with solution value $c(X^*)$, if and only if X^* is an optimal solution to the corresponding* Q *problem with $c = c(X^*)$ and $Q(X^* \mid c(X^*)) = 0$.*

Proposition 2 provides the basis for the following iterative algorithm (FP-algo), which finds a solution to FP by solving a series of instances of Q problems [15].

<u>Algorithm FP-algo</u>:

```
1. Select some $X_0 \subseteq E$. Set $c_0 \leftarrow F_1(X_0)/F_2(X_0)$, and $k \leftarrow 0$.
2. Compute $X_{k+1}$ by solving the Q problem:
       $Q(X_{k+1} \mid c_k) \leftarrow \max_{X \subseteq E}\{F_1(X) - c_k F_2(X)\}$.
3. If $Q(X_{k+1} \mid c_k) = 0$, then return $X^* \leftarrow X_k$.
   Otherwise, set $c_{k+1} \leftarrow F_1(X_{k+1})/F_2(X_{k+1})$, $k \leftarrow k + 1$, and go to Step (2).
```

It can be shown [15] that the sequence (c_k) generated by FP-algo is increasing, and that if F_2 is an integer-valued set function (and we will see that this is our case), then the number of iterations of FP-algo is bounded by the number of elements in the underlying set, which in our case is the edgeset E.

We formulate DSS-INV as an instance of FP. As DSS-INV is parameterized by $\lambda \geq 0$, we introduce such parameter in FP and set

$$F_1(X \mid \lambda) = \sum_{\{e_i, e_j\} \in X^2} s(e_i, e_j) - \lambda |V(X)|, \text{ and } F_2(X) = |X|.$$

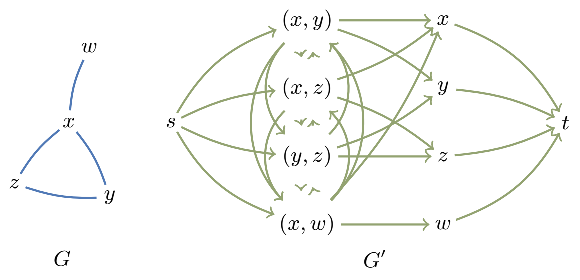

Fig. 2. A graph G (left), and the corresponding flow graph G' (right) used for solving Q with parametric MIN-CUT techniques, as described in Section 3.2. The edge weights in G' are not shown to avoid clutter.

Now, DSS-INV becomes an instance of FP and algorithm FP-algo can be applied. As $F_2(X)$ is the number of edges in the solution, the algorithm FP-algo is guaranteed to halt after solving $\mathcal{O}(|E|)$ instances of the Q problem.

Each instance of the Q problem at Step (2) of FP-algo can be solved efficiently by a parametric preflow/minimum cut algorithm [15]. The construction of the flow graph is presented in the next section.

Since we introduced the parameter λ, we need to write the objectives of FP and Q as $c(X \mid \lambda)$ and $Q(X \mid c, \lambda)$, respectively, but we will omit the dependency on λ when it is clear. We denote the optimal values of FP and Q as $c^*(\lambda) = c(X^* \mid \lambda)$ and $Q^*(c, \lambda) = Q(X^* \mid c, \lambda)$, respectively.

3.2 Parametric MIN-CUT

In this section we show how to solve instances of the Q problem by using a mapping to the MIN-CUT problem. A similar approach has been used, among others, by Goldberg [16] to solve the densest-subgraph problem.

Let the input of Q be a graph $G = (V, E)$ with edge similarity S and parameters $c \in \mathbb{R}$ and $\lambda \in \mathbb{R}_{\geq 0}$. We construct the following directed weighed network $G' = (U', E', w')$. The set U' is defined as $U' = U_E \cup U_V \cup \{s, t\}$, where $U_E = \{u_e \mid e \in E\}$ contains a node u_e for each edge $e \in E$, $U_V = \{u_v \mid v \in V\}$ contains a node u_v for each node $v \in V$, and the nodes s and t are additional source and sink nodes. The nodes in U_E are pairwise connected by bidirectional edges (u_e, u_d) with weight $\frac{1}{2}S(e, d)$, whereas the nodes in U_V are not connected to each other. Additionally, there is a directed edge (u_e, u_v) for each $v \in V$ that is an endpoint of $e \in E$ with weight $w'(u_e, u_v) = +\infty$. Finally, the source s is connected to all the nodes in U_E by directed edges with weight $w'(s, u_e) = \frac{1}{2}\sum_{d \in E, d \neq e} s(e, d) - c$, and each node in U_V is connected to t by a directed edge with weight λ. The construction of G' for a given G is clearly polynomial. An example of the construction of G' is shown in Figure 2.

We now solve the (s, t)-MIN-CUT problem on the graph $G' = (U', E', w')$, parameterized with c and λ. Let $(\{s\} \cup U^*, \{t\} \cup \overline{U}^*)$ be the minimum cut in G', and let $\mathsf{C}^*(c, \lambda)$ be its value. The next proposition establishes the connection between the optimal values of MIN-CUT on G' and the Q problem on G, and describes how the solution edge set for the Q problem can be derived from the solution cut set of MIN-CUT.

Proposition 3. *The value of the (s,t)-MIN-CUT in the graph $G' = (U', E', w')$ for given parameters c and λ corresponds to the optimum value for the Q problem with the same values of c and λ. The solution edge set X^* for Q problem on G can be reconstructed from the minimum-cut set $\{s\} \cup U^* \subseteq U'$ in G' as $X^* = U^* \cap U_E$.*

To summarize, in the previous sections we have established the following:

Proposition 4. *An instance of DSS-INV for a given parameter λ can be solved by mapping it to Problem FP and applying the FP-algo. Problem Q in the iterative step of FP-algo can be solved by mapping it to the parametric MIN-CUT problem, as shown in Proposition 3.*

Let us evaluate the time and space complexity of the proposed solution. In FP-algo we iteratively search for optimal values in the Q problem by solving MIN-CUT problems. In each iteration, only the source link capacities are updated as c_k changes, and, as mentioned before, sequence (c_k) grows monotonically. This setting can be handled efficiently in the parametric MIN-CUT framework, which incrementally updates the solution from the previous iteration. The state-of-the-art algorithm for parametric MIN-CUT [17] requires $\mathcal{O}(mn \log n + kn)$ time and $\mathcal{O}(m)$ space for a graph with n nodes, m edges, and k updates of edge capacities (iterations in FP-algo). Recall that the number of iterations is bounded by $\mathcal{O}(|E|)$, and thus, solving DSS-INV for a fixed λ requires $\mathcal{O}(|E|^3 \log |E|)$ time and $\mathcal{O}(|E|^2)$ space.

3.3 λ-Exploration

Having discussed how to solve the DSS-INV problem for a fixed λ, we now introduce a framework to efficiently enumerate the solutions for all possible values of λ. The goal is to identify the ranges of values of λ that yield identical solutions and exclude them from the search.

First, we show the monotonicity of the optimal solution value of DSS-INV, the optimal subgraph similarity and density values with respect to λ.

Proposition 5. *The optimal solution value of DSS-INV is a monotonically non-increasing function of λ. The density of the optimal edge set is a monotonically non-decreasing function of λ. The subgraph edge similarity of the optimal edge set is a monotonically non-increasing function of λ.*

From the definition of optimality and Proposition 5, it follows that:

Corollary 1. *Given two solutions X_1 and X_2 to DSS-INV for λ_1 and λ_2 with $\lambda_1 < \lambda_2$, either (i) $S(X_1) = S(X_2)$ and $D(X_1) = D(X_2)$ or (ii) $S(X_1) > S(X_2)$ and $D(X_1) < D(X_2)$.*

The monotonicity of the optimal values of the objective functions and Corollary 1 will guide our exploration of the λ ranges.

Note that the shown monotonic properties are not strict and it is indeed easy to construct an example input graph, where different values of λ lead to solutions to DSS-INV with the same values of subgraph edge similarity and density. To avoid a redundant search, we would like to solve DSS-INV for all the values of λ that lead to distinct combinations of density and similarity values. Such redundant values of λ can be pruned by observing that when two values λ_1 and λ_2 give solutions with the same values $S_1 = S_2$ and $D_1 = D_2$, then all $\lambda \in [\lambda_1, \lambda_2]$ must also lead to the same optimal density and subgraph edge similarity, and thus the interval $[\lambda_1, \lambda_2]$ can be discarded from further search. This result follows from the monotonicity of the optimal values of density and similarity (Proposition 5).

The proposed approach to search for different values of λ is a breadth-first iterative algorithm. At the beginning, the set of distinct solutions $\mathcal{P}$ is empty, and $\lambda_\ell = \lambda_{\min}$ and $\lambda_u = \lambda_{\max}$. The algorithm maintains a queue of candidate search intervals T, which is initially empty. To avoid clutter, we denote the solution values of subgraph edge similarity and density (S, D) for a given λ as $t(\lambda)$.

Algorithm λ-exploration:

```
1. Compute set $X^*(\lambda_\ell)$ and add it to $\mathcal{P}$.
2. Compute set $X^*(\lambda_u)$.
3. If $t(\lambda_u) \neq t(\lambda_\ell)$, then push $(\lambda_\ell, \lambda_u, t(\lambda_\ell), t(\lambda_u))$ to the queue $T$ and add
   $X^*(\lambda_u)$ to $\mathcal{P}$.
4. While $Q$ is not empty:
   (a) Pop $(\lambda_\ell, \lambda_u, t(\lambda_\ell), t(\lambda_u))$ from $T$.
   (b) Set $\lambda_m = (\lambda_\ell + \lambda_u)/2$ and compute $X^*(\lambda_m)$.
   (c) If $t(\lambda_m) \neq t(\lambda_\ell)$, then push $(\lambda_\ell, \lambda_m, t(\lambda_\ell), t(\lambda_m))$ to $T$.
   (d) If $t(\lambda_m) \neq t(\lambda_u)$, then push $(\lambda_m, \lambda_u, t(\lambda_m), t(\lambda_u))$ to $T$.
   (e) If $t(\lambda_m) \neq t(\lambda_\ell)$ and $t(\lambda_m) \neq t(\lambda_u)$, add $X^*(\lambda_m)$ to $\mathcal{P}$.
```

To bound the number of calls of λ-search, we need to lower bound the difference between two consecutive values of λ that lead to two different solutions. This lower bound is given in the next proposition, together with upper and lower bounds for λ values.

Proposition 6. *To obtain all the distinct solutions in the λ-exploration algorithm, a lower bound for a value of λ is $\lambda_{min} = s_{min}/2|E|$, an upper bound is $\lambda_{max} = s_{max}|E|^2/2$, and a lower bound for the difference between two values of λ leading to solutions with distinct density and subgraph edge similarity values is $\delta_\lambda = s_{min}/2|E|$. Here $s_{min} = \min_{\{e_1,e_2\}\in X^2} s(e_1, e_2)$ and $s_{max} = \max_{\{e_1,e_2\}\in X^2} s(e_1, e_2)$.*

Given the bounds in Proposition 6, an upper bound on the number of different values of λ that we need to try is $I_\lambda = (\lambda_{max} - \lambda_{min})/\delta_\lambda \leq |E|^3 \frac{s_{max}}{s_{min}}$, where s_{max} and s_{min} are the largest and the smallest non-zero values of edge similarity between two edges in the input graph. Thus, for a complete exploration of all the possible λ values leading to different values of subgraph edge similarity and density of the solution graph, we need $\mathcal{O}(|E|^3)$ iterations. This estimate is pessimistic and assumes no subranges of λ are pruned during the exploration. As we will see later, on practice the exploration typically requires around $|E|$ number of iterations.

4 Related Work

In this paper we consider the problem of finding subgraphs that maximize both a density measure and a similarity measure. The problem of finding dense structures in graphs has been extensively studied in the literature, as it finds applications in many domains such as community detection [10, 13], event detection [1], and fraud detection [18]. Existing works have addressed the task of finding the best solution that satisfies the given constraints, such as, the densest subgraph [16, 18], the densest subgraph of k vertices [4], the densest subgraph in a dual network [25], or the best α-quasi-clique [23]. Other works have aimed at retrieving a set of good solutions, such as top-k densest subgraphs in a graph collection [24], k diverse subgraphs with maximum total aggregate density [2], or k-cores with maximum number of common attributes [14]. However, these works optimize a single measure, i.e., the density, thus ignoring other properties of the graph, or find a solution that depends on an input query.

There are a few works focusing on edge similarity. The closest to our work, Boden et al. [5], considers edge-labeled multilayer graph and looks for vertex sets that are densely connected by edges with similar labels in a subset of the graph layers. They set a pairwise edge similarity threshold for a layer and look for 0.5 quasi-cliques, which persist for at least 2 layers. In contrast, our approach does not require any preset parameters and offers a comprehensive exploration of the space of dense and similar subgraphs.

Motivated by applications in fraud detection, Shin et al. [18] propose a greedy algorithm that detects the k densest blocks in a tensor with N attributes, with guarantees on the approximation. The framework outputs k blocks by greedy iterative search. Yikun et al. [26] propose a novel model and a measure for dense fraudulent blocks detection. The measure is tailored for the fraud detection in multi-dimensional entity data, such as online product reviews, but could be possibly adapted to capture other types of node and edge similarities. They propose an efficient algorithm, which outputs several graph with some approximation guarantees. In contrast to the approaches above, our work offers exploration of *exact* solutions with different trade-offs.

Multi-objective optimization for interesting graph structures search was studied in the context of frequent pattern mining [6, 21] and graph partitioning [3]. However, most of the frequent pattern works do not consider the density as an objective function, but depend on the notion of frequency in the graph and cannot be extended to our case. Carranza et al. [6], instead, define a conductance measure in terms of an input pattern and then recursively cut the graph into partitions with minimum conductance. The result, however, depends on the input. Graph partitioning approaches optimize quality functions based on modularity and node/edge attributes, but focus on a complete partition of the input graph [11, 20] and do not guarantee to the quality of each individual partition.

5 Experimental Evaluation

We evaluate the proposed method using real-world multiplex networks from the CoMuNe lab database[5] and the BioGRID datasets[6]. An implementation of our method is publicly available[7]. In the following, we refer to our approach as DenSim. For the parametric MIN-CUT problem, we use Hochbaum's algorithm [17] and its open-source C implementation [8]. The experiments are conducted on a Xeon Gold 6148 2.40 GHz machine.

Datasets. We use the following real-world datasets: *CS-Aarhus* is a multiplex social network consisting of five kinds of online and offline relationships (Facebook, leisure, work, co-authorship, and lunch) between the employees of the Computer Science department at Aarhus University. *EU-Air* is a multilayer network composed by 37 layers, each one corresponding to a different airline operating in Europe. *Neuronal C.e.* is the C. elegans connectome multiplex that consists of layers corresponding to different synaptic junctions: electric, chemical monadic, and polyadic. *Genetic C.e.* and *Genetic A.th.* are multiplex networks of genetic interactions for C. elegans and Arabidopsis thaliana. Table 1 summarizes the main characteristics of the datasets.

All datasets are multilayer networks $G_M = (V, E, \ell)$ with a labeling function $\ell : E \to 2^L$, where L is the set of all possible labels. We omit edge directionality if it is present in the dataset. The edge similarity function is defined as the Jaccard coefficient of the labellings: $s(e_1, e_2) = |\ell(e_1) \cap \ell(e_2)|/|\ell(e_1) \cup \ell(e_2)|$. If the pairwise edge similarity is 0 for a pair of edges, we do not materialize the corresponding edge in the MIN-CUT problem graph.

We note that, while the datasets are not large in terms of number of nodes and edges, the number of edges co-appearing in at least one layer is significant. Furthermore, the subgraphs edge similarity is high.

Baselines. We compare DenSim with two baselines, BLDen and BLSim.

Algorithm BLDen optimizes the density directly, but takes into account edge-similarity indirectly. It outputs the set of edges of the densest weighted subgraph in the complete graph $G_D = (V, E_D, w^D)$, which has the same nodes as the input multiplex network G_M. The edge-weighting function w^D has two components, i.e., $w^D = (w_1^D, w_2^D)$. The weight $w_1^D(u, v)$ captures the graph topology, i.e., $w_1^D(u, v) = 1$ if $(u, v) \in E$ for some layer and 0 otherwise. The weight $w_2^D(u, v)$ captures the similarity of node activity across layers: we first define the node labels as the set of all layers where a node appears in some edge $\ell(u) = \cup_{u \in e} \ell(e)$, and then define $w_2^D(u, v)$ to be the Jaccard index between the sets $\ell(u)$ and $\ell(v)$.

[5] `https://comunelab.fbk.eu/data.php`
[6] `https://thebiogrid.org`
[7] `https://github.com/polinapolina/dense-subgraphs-with-similar-edges`

Table 1. Network characteristics. $|V|$: number of vertices; $|E|$: the number of edges; L: the number of layers; $|E|_{avg}$: number of edges per layer; $|E_{mult}|$: number of edges in the multiplex (same edges on different layers are counted as distinct); $|E_{meta}|$: number of unordered edge pairs co-appeared at least in one layer; D: density of the network; D_{avg}: average density across the layers; S: similarity of the network's edge set $S(E)$; ℓ_{avg}: average participation of an edge to a layer.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	L	$\lvert E\rvert_{avg}$	$\lvert E_{mult}\rvert$	$\lvert E_{meta}\rvert$	D	D_{avg}	S	ℓ_{avg}
CS-Aarhus	61	353	5	124.00	620	39565	5.78	2.60	57.44	1.75
EU-Air	417	2953	37	96.97	3588	360082	7.08	1.56	94.64	1.21
Neuronal C.e.	279	2290	3	1036.0	5863	1762756	8.20	3.86	534.70	1.35
Genetic C.e.	3879	7908	6	1338.66	8182	17249444	2.03	1.23	2141.88	1.01
Genetic A.th.	6980	16713	7	2499.57	18655	91782863	2.39	1.18	5156.65	1.04

The final weight of an edge is a weighted sum $w^D(u,v) = w_1^D(u,v) + \gamma w_2^D(u,v)$, where γ regulates the importance of the components. By tuning γ we can obtain a trade-off between topological density and subgraph edge similarity.

Algorithm BLSim is the counterpart of BLDen, which optimizes the edge-similarity directly, but accounts for density indirectly. BLSim finds the densest weighted subgraph of the complete graph $G_S = (E, E_S, w^S)$ with $w^S = (w_1^S, w_2^S)$. Here weight $w_1^S(e_1,e_2)$ is the edge similarity in the multiplex network G_M, i.e., $w_1(e_1,e_2) = s(e_1,e_2)$ and $w_2^S(e_1,e_2)$ represents the topological information, i.e., $w_2^S(e_1,e_2) = 1$ if e_1 and e_2 have a common node in the original graph, and 0 otherwise. Again, the final edge weight is a weighted sum $w^S(e_1,e_2) = w_1^S(e_1,e_2) + \gamma w_2^S(e_1,e_2)$. When $\gamma = 0$, finding the densest weighted subgraph is equivalent to finding the set of edges in the original graph with the largest similarity. We tune γ to obtain a trade-off between subgraph edge similarity and topological density.

As with DenSim, we do not materialize 0-weight edges in the baselines. Both baselines search for the densest subgraph. Similarly to DenSim, we use the parametric MIN-CUT framework [15].

Experimental Results. Figure 3 shows the different characteristics of solutions discovered in the datasets during λ-exploration. We observe that the baselines are extremely sensitive to the values of γ: it is hard to find a set of γ values that lead to distinct solutions. Moreover, the range and granularity of γ depend on the datasets, and it is up to the end-user to decide their values. To provide a somewhat unified comparison, we allow γ to range from 0 to 10 with step 0.1.

The first column in Figure 3 shows the values of density and subgraph edge similarity of the solutions found. The solutions discovered by DenSim cover the space of possible values of similarity and density rather uniformly, providing a range of trade-offs. The solutions discovered by the baselines are mostly grouped around the same values and often dominated by solutions of DenSim. Note that the baselines successfully find the points with the largest density or similarity. By design, these solutions correspond to values of $\gamma = 0$, and they also correspond to the solutions of DenSim for $\lambda = \lambda_{min}$ and $\lambda = \lambda_{max}$.

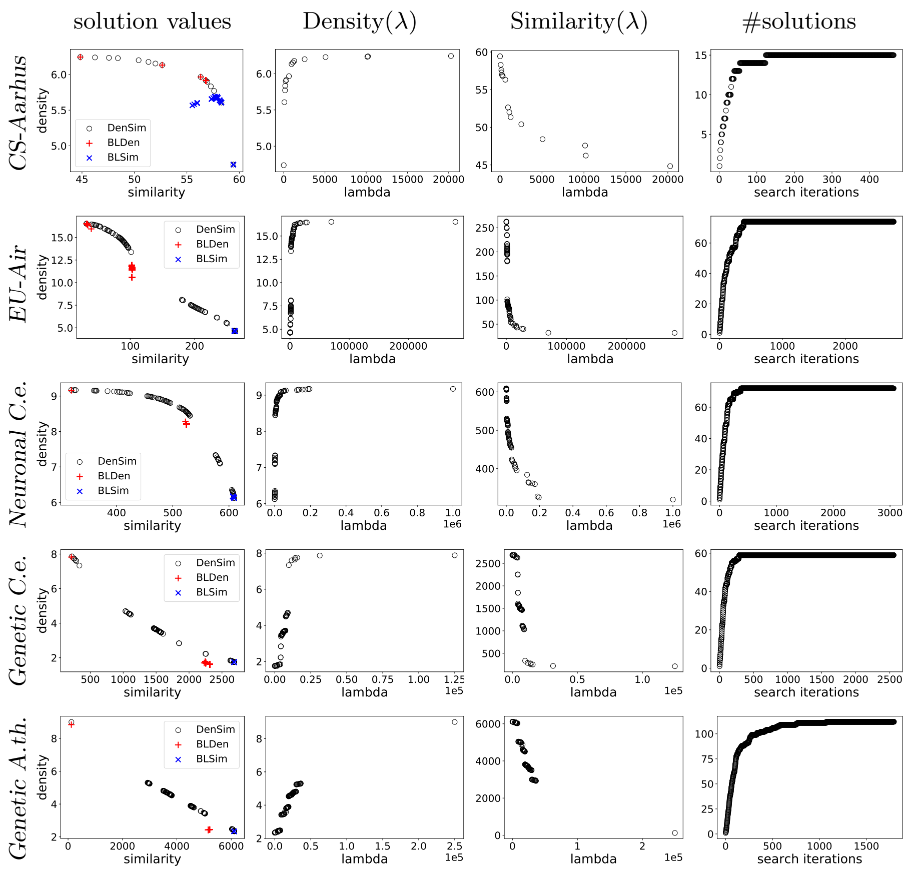

Fig. 3. Experimental results for our method DenSim, and the baselines BLDen and BLSim on real-world datasets. Each row represents one dataset. The first column shows the values of subgraph edge similarity and density of the discovered solutions. Column 1 also show the values of solutions, discovered by the baselines. Columns 2 and 3 show D and S as a function of λ. Column 4 shows how the number of discovered unique optimal solutions grows with the number of iterations in λ-exploration.

Columns 2 and 3 show optimal density D and subgraph edge similarity S as functions of λ. As expected, larger values of λ correspond to solutions with larger density and smaller similarity. The range of λ that gives unique optimal solutions is dataset-dependent and not uniform. However, due to the monotonicity property we can search for these values efficiently, in contrast to the naïve search for the baselines. The last column of Figure 3 shows the efficiency of λ-exploration. All unique solutions are found after 200 to 1000 iterations.

In Table 2 we report the number of the unique optimal solutions and running times. The total running time varies from seconds to hours. We should highlight,

Table 2. Number of subgraphs and running time characteristics. $|\mathcal{P}|$: number of discovered optimal solutions; t(s): total time (seconds) for the search; I_λ: number of tested values of λ; t_λ(s): average time to test one value of λ; I_{MC}: average number of MIN-CUT problems solved for one λ (i.e., average number of iterations in FP-algo).

Dataset	$\lvert\mathcal{P}\rvert$	t(s)	I_λ	t_λ(s)	I_{MC}
CS-Aarhus	15	2.26	465	0.003	2.89
EU-Air	74	314	2770	0.069	6.00
Neuronal C.e.	72	1015	3064	0.244	4.60
Genetic C.e.	59	10159	2561	3.075	4.72
Genetic A.th.	112	43200	1794	20.540	5.68

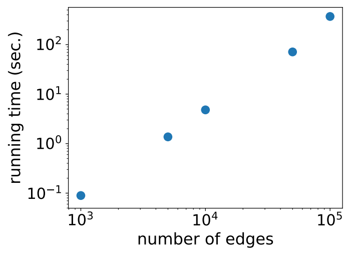

Fig. 4. Running time in seconds to calculate 10 first solutions. The input graph is a $G_{n,m}$ random graph with $n = 1000$ nodes, m edges, the probability that a pair of edges has a non-zero similarity is set to 0.001.

however, that finding a solution for a single value of λ takes on average 20 seconds for the largest dataset. Thus, if the search progresses fast (as shown in the last row of Figure 3) and a sufficient number of optimal solutions have been found, we can terminate the search. As we discussed before, we implement λ-exploration as a BFS, so that at any point we have a diverse set of λ tested. It is worth noting that FP-algo converges in about 5 iterations on average, and thus the MIN-CUT algorithm is not run many times.

Scalability. In order to test the scalability of DenSim we generate a number of random graphs with $n = 1000$ nodes and varying m number of edges. We draw the graphs from a random graph model $G_{n,m}$. The similarities between the edges are random values from $(0, 1]$, and the probability that a pair of edges has a non-zero similarity is set to 0.001. We run DenSim until it found 10 solutions, and the running time is reported in Figure 4. It took 6 minutes to find 10 solutions for the largest graph with 100000 edges.

Case Study. We run DenSim on the *CS-Aarhus* dataset. We pick three of the solutions discovered: one for λ_{min}, one for λ_{max}, and one for the median value λ_{med}. Recall that λ_{min} gives a solution with maximum subgraph edge similarity, while density is ignored; while λ_{max} gives a solution with maximum density.

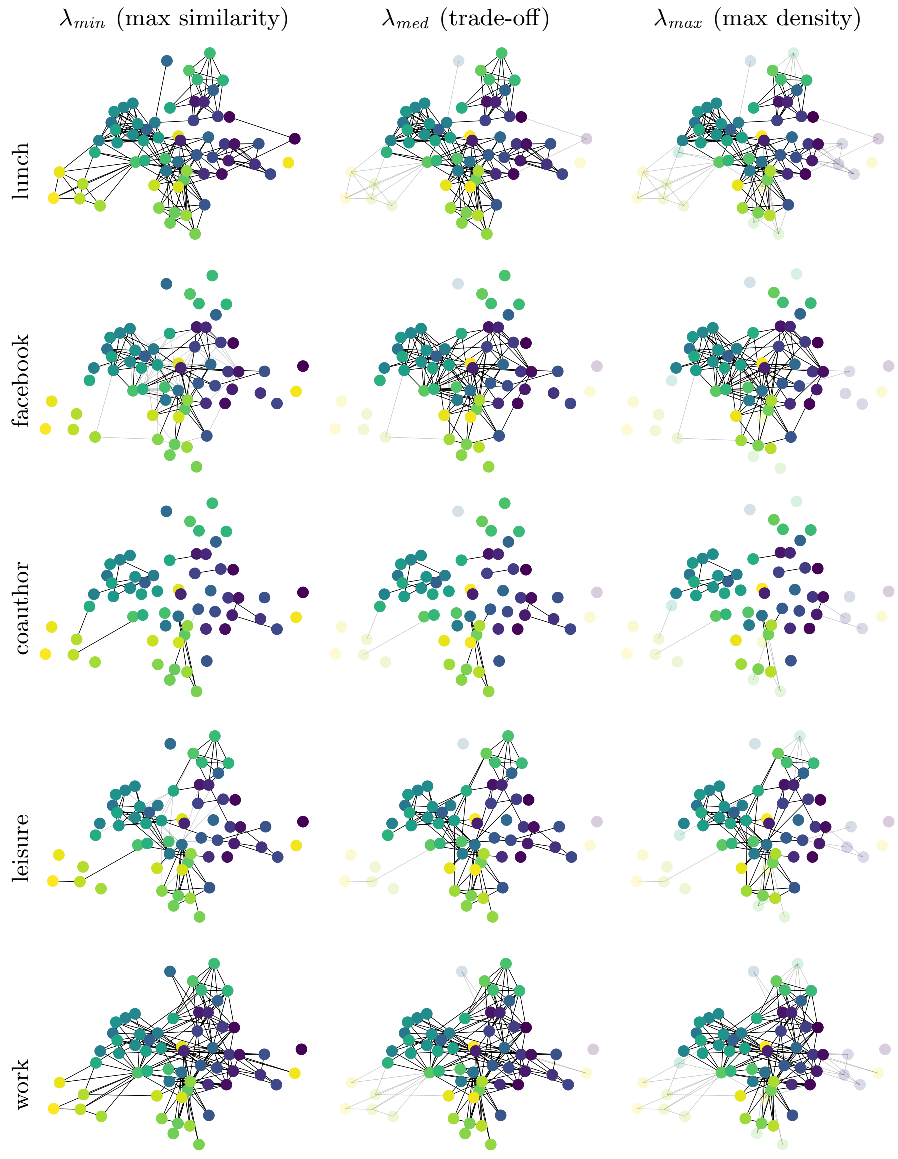
λ_{min} (max similarity)
λ_{med} (trade-off)
λ_{max} (max density)
lunch
facebook
coauthor
leisure
work

Fig. 5. Some solutions output by DenSim for *CS-Aarhus* dataset. Each column corresponds to a solution with the smallest (left), largest (right), and median (center) value of λ. Each row corresponds to a layer in the dataset. The nodes and edges, included into the solution are shown with the bright colors, the rest of the nodes and edges on each layer are drawn transparently.

ignoring edge similarity. Any other λ_{med} should provide some balance between these extremes. The solutions are visualized in Figure 5.

The graph maximizing the subgraph similarity (λ_{min}) includes all the edges from the layers of "work" and "lunch." Since the dataset contains relationships between the employees of the same university department, it is intuitive that these two layers define the edge set with the largest subgraph edge similarity. All network nodes are included in this solution, as all these people share similar interactions at work and lunch. Facebook and leisure interactions, not overlapping with "work" and "lunch", are excluded, as they are localized in their layers. The resulting graph contains 61 nodes and 289 edges, while the subgraph edge similarity is 59.43 and the density is 4.73.

The graph maximizing the density (λ_{max}) includes the edges of the densest subgraph from the "work" layer, and reinforces it by adding edges from other layers. The graph contains 45 nodes and 281 edges, and it is the smallest of the three. The subgraph edge similarity is 44.83 and the density is 6.24.

The trade-off graph (λ_{med}) selects 325 edges, more than the other two, while it has 53 nodes. Its subgraph similarity is 52.64 and its density 6.13. The graph resembles the one for λ_{max}, but adds interactions that decrease the density while increasing the subgraph similarity.

6 Concluding Remarks

In this paper we study a novel graph-mining problem, where the goal is to find a set of edges that maximize the density of the edge-induced subgraph and the subgraph edge similarity. We reformulate the problem as a non-standard Lagrangian relaxation and develop a novel efficient algorithm to solve the relaxation based on parametric minimum-cut [15, 17]. We provide an efficient search strategy through the values of Lagrangian multipliers. The approach is evaluated on real-world datasets and compared against intuitive baselines.

Acknowledgments

This research was partially supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-GC-2019-001). Aristides Gionis is supported by three Academy of Finland projects (286211, 313927, 317085), the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project "SoBigData++" (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Angel, A., Sarkas, N., Koudas, N., Srivastava, D.: Dense subgraph maintenance under streaming edge weight updates for real-time story identification. VLDB pp. 175–199 (2012)

2. Balalau, O.D., Bonchi, F., Chan, T., Gullo, F., Sozio, M.: Finding subgraphs with maximum total density and limited overlap. In: WSDM. pp. 379–388 (2015)
3. Baños, R., Gil, C., Montoya, M., Ortega, J.: A new pareto-based algorithm for multi-objective graph partitioning. In: ISCIS. pp. 779–788 (2004)
4. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In: STOC. pp. 201–210 (2010)
5. Boden, B., Günnemann, S., Hoffmann, H., Seidl, T.: Mining coherent subgraphs in multi-layer graphs with edge labels. In: SIGKDD. pp. 1258–1266 (2012)
6. Carranza, A.G., Rossi, R.A., Rao, A., Koh, E.: Higher-order spectral clustering for heterogeneous graphs. arXiv preprint arXiv:1810.02959 (2018)
7. Chan, H., Han, S., Akoglu, L.: Where graph topology matters: the robust subgraph problem. In: SIAM. pp. 10–18 (2015)
8. Chandran, B.G., Hochbaum, D.S.: A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. Operations research pp. 358–376 (2009)
9. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: APPROX. pp. 84–95 (2000)
10. Chen, J., Saad, Y.: Dense subgraph extraction with application to community detection. TKDE pp. 1216–1230 (2012)
11. Combe, D., Largeron, C., Géry, M., Egyed-Zsigmond, E.: I-louvain: An attributed graph clustering method. In: IDA. pp. 181–192 (2015)
12. Ehrgott, M.: Multicriteria optimization. Springer (2005)
13. Falih, I., Grozavu, N., Kanawati, R., Bennani, Y.: Community detection in attributed network. In: Companion Proceedings of WWW. pp. 1299–1306 (2018)
14. Fang, Y., Cheng, R., Luo, S., Hu, J.: Effective community search for large attributed graphs. VLDB pp. 1233–1244 (2016)
15. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. J. on Comp. pp. 30–55 (1989)
16. Goldberg, A.V.: Finding a maximum density subgraph. University of California Berkeley, CA (1984)
17. Hochbaum, D.S.: The pseudoflow algorithm: A new algorithm for the maximum-flow problem. Operations Research pp. 992–1009 (2008)
18. Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: SIGKDD. pp. 895–904 (2016)
19. Khuller, S., Saha, B.: On finding dense subgraphs. In: ICALP. pp. 597–608 (2009)
20. Sánchez, P.I., Müller, E., Korn, U.L., Böhm, K., Kappes, A., Hartmann, T., Wagner, D.: Efficient algorithms for a robust modularity-driven clustering of attributed graphs. In: SIAM. pp. 100–108 (2015)
21. Shelokar, P., Quirin, A., Cordón, O.: Mosubdue: a pareto dominance-based multi-objective subdue algorithm for frequent subgraph mining. KAIS pp. 75–108 (2013)
22. Tsourakakis, C.: The k-clique densest subgraph problem. In: WWW (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: SIGKDD. pp. 104–112 (2013)
24. Valari, E., Kontaki, M., Papadopoulos, A.N.: Discovery of top-k dense subgraphs in dynamic graph collections. In: SSDBM. pp. 213–230 (2012)
25. Wu, Y., Jin, R., Zhu, X., Zhang, X.: Finding dense and connected subgraphs in dual networks. In: ICDE. pp. 915–926 (2015)
26. Yikun, B., Xin, L., Ling, H., Yitao, D., Xue, L., Wei, X.: No place to hide: Catching fraudulent entities in tensors. In: The World Wide Web Conference (2019)