

# Finding Dense Subgraphs in Relational Graphs

Vinay Jethava<sup>(✉)</sup> and Niko Beerenwinkel

Department of Biosystems Science and Engineering, ETH Zürich, Zurich, Switzerland  
{vinay.jethava,niko.beerenwinkel}@bsse.ethz.ch

**Abstract.** This paper considers the problem of finding large dense subgraphs in relational graphs, i.e., a set of graphs which share a common vertex set. We present an approximation algorithm for finding the densest common subgraph in a relational graph set based on an extension of Charikar’s method for finding the densest subgraph in a single graph. We also present a simple greedy heuristic which can be implemented efficiently for analysis of larger graphs. We give graph dependent bounds on the quality of the solutions returned by our methods. Lastly, we show by empirical evaluation on several benchmark datasets that our method out-performs existing approaches.

## 1 Introduction

Finding dense subgraphs is a key subtask in many applications [see 21, forasurvey]. In many contexts, there exist several graphs encoding different relationships between the same set of actors. Then, a subset of actors having high degree of interconnections (dense) which recur in multiple graphs (frequent) often have a rich interpretation in the application domain. For example, dense recurrent subgraphs in multiple gene co-expression networks have been shown to correspond to known functional/transcriptional modules or protein complexes as well as phenotype-specific modules [12, 24].

Several data-mining methods have addressed the problem of enumerating dense recurrent subgraphs [see, e.g., 6, 12, 13, 24, 26, 31, 33]. Most existing approaches enumerate all frequent quasCliques depending on parameters such as minimum support threshold and minimum relative density. This results in exponential growth of search space with increasing size of the returned subgraph, making the methods unsuitable for identifying large dense subgraphs in multiple graphs. The approach of [24] yields a non-convex cubic programming problem which is solved approximately using multi-stage convex relaxation [34] and used in the analysis of co-expression networks in order to identify small biologically relevant modules. [13] present a method to identify large dense subgraphs based on solving a Multiple Kernel Learning (MKL) problem [3, 27] with precomputed kernels.

On the other hand, there is a rich body of work on approximation algorithms which addresses the problem of finding the densest subgraph (DS) [7, 11] and its size-constrained variants (*DkS*, *Da1kS*, *DamkS*) [1, 5, 20] including greedy approaches [2, 9, 28], truncated power method [32], linear programming (LP) based methods [7, 20] and semidefinite programming (SDP) based methods

[8, 29]. We note that given a relational graph set <sup>1</sup>,  $\mathbb{G} := \{G^{(m)} = (V, E^{(m)})\}_{m=1}^M$ , it is possible to construct integrated graphs, e.g.,  $G_{\cup}$ ,  $G_{\cap}$  or  $G_{\geq t}$  having edge sets respectively,  $\bigcup E^{(m)}$ ,  $\bigcap E^{(m)}$  or  $\{e \mid \text{supp}(e, \mathbb{G}) \geq t\}$  <sup>2</sup>; and identify dense subgraphs in the integrated graph using these methods. However, as noted by [16], a dense subgraph in the integrated graph may either not be dense in one or more of the original graphs, e.g.,  $G_{\cup}$  and  $G_{\geq t}$ ; or, be too pessimistic in size of the returned subgraph, e.g.,  $G_{\cap}$ .

In this paper, we formalize the notion of Densest Common Subgraph (i.e., a subset of nodes which maximizes the density of the induced subgraph in each of the graphs in the relational graph set) which was previously discussed in [13]. We present an approximation algorithm (DCS\_LP) for finding the densest common subgraph in a relational graph set based on an extension of Charikar’s LP based approach [7, 20] for finding the densest subgraph (DS) in a single graph. We also present a simple greedy heuristic (DCS\_GREEDY) which can be implemented efficiently for analysis of larger graphs. We give graph dependent bounds on the quality of the solutions returned by DCS\_LP and DCS\_GREEDY. Lastly, we show by empirical evaluation on several benchmark datasets and real-world datasets that our methods out-perform prior approaches.

*Notation.* We represent vectors using lower case bold letters  $\mathbf{a}, \mathbf{b}, \dots$ , etc., and matrices using upper case bold letters  $\mathbf{A}, \mathbf{B}, \dots$  etc.; with  $a_i$  referring to  $i^{\text{th}}$  element of  $\mathbf{a}$ , and similarly  $A_{ij}$  referring to  $(i, j)^{\text{th}}$  entry of matrix  $\mathbf{A}$ . We use notation  $[n]$  to denote the set  $\{1, 2, \dots, n\}$ . For a vector in  $\mathbb{R}^d$ , we denote the Euclidean norm by  $\|\cdot\|$  and the  $p$ -norm by  $\|\cdot\|_p$ . The inequality  $\mathbf{a} \geq 0$  is true if it holds element-wise.

Let  $G = (V, E)$  be a simple undirected graph of order  $n$  with vertex set  $V = [n]$  and edge set  $E \subseteq V \times V$ . Let  $\mathbf{A} \in \mathbf{S}_n$  denote the adjacency matrix of  $G$  where  $A_{ij} = 1$  if edge  $(i, j) \in E$ , and 0 otherwise. We use shorthand notation  $ij$  to mean  $(i, j)$  whenever clear from context. Let  $\bar{G}$  denote the complement graph of  $G$ . The adjacency matrix of  $\bar{G}$  is  $\bar{\mathbf{A}} = \mathbf{e}\mathbf{e}^T - \mathbf{I} - \mathbf{A}$ , where  $\mathbf{e} = [1, 1, \dots, 1]^T$  is a vector of length  $n$  containing all 1’s, and  $\mathbf{I}$  denotes the identity matrix. We denote the indicator vector for some set  $S \subseteq V$  as  $\mathbf{e}_S$  which is one for all  $i \in S$  and zero in other co-ordinates.

We use notation  $\text{deg}_G(i)$  to denote the degree of node  $i$  in graph  $G$ , and  $\text{deg}_G(i, S)$  to denote the degree of node  $i$  in the subgraph  $(S, E(S))$  induced by vertex set  $S \subseteq V$  in graph  $G$ . The *density*  $\delta_G(S)$  and *relative density*  $\rho_G(S)$  of subgraph  $(S, E(S))$  induced by vertex set  $S \subseteq V$  in graph  $G = (V, E)$  are given by  $\delta_G(S) := \frac{|E(S)|}{|S|}$  and  $\rho_G(S) := |E(S)| / \binom{|S|}{2}$  respectively. The induced subgraph  $(S, E(S))$  is an  $\alpha$ -*quas clique* if  $|E(S)| \geq \alpha \binom{|S|}{2}$ , i.e., if the relative density  $\rho_G(S)$  of the induced subgraph exceeds a threshold parameter  $\alpha \in (0, 1)$ . Let  $\delta_G^* := \max_{S \subseteq V} \delta_G(S)$  denote the density of the densest subgraph (DS) in  $G$ .

<sup>1</sup> A relational graph set is defined as a set of simple undirected graphs which share a common vertex set.

<sup>2</sup> The support of an edge  $e$  in the relational graph set  $\mathbb{G}$  denoted by  $\text{supp}(e, \mathbb{G}) := \#\{m : e \in E^{(m)}\}$  is the number of graphs  $G^{(m)} \in \mathbb{G}$  which contain this edge.

Given a relational graph set  $\mathbb{G} = \{G^{(m)} = (V, E^{(m)})\}_{m=1}^M$ , we use shorthand notation  $\text{deg}_m(i)$ ,  $\text{deg}_m(i, S)$ ,  $\delta_m(S)$ ,  $\rho_m(S)$  and  $\delta_m^*$  to denote  $\text{deg}_{G^{(m)}}(i)$ ,  $\text{deg}_{G^{(m)}}(i, S)$ ,  $\delta_{G^{(m)}}(S)$ ,  $\rho_{G^{(m)}}(S)$  and  $\delta_{G^{(m)}}^*$ , respectively, whenever clear from context. We use  $n_{\mathbb{G}} := \sum_m |E^{(m)}|$  to denote the total number of edges in the relational graph set. The support of a subgraph  $H = (V_H, E_H)$ ,  $V_H \subseteq V$ ,  $E_H \subseteq V \times V$  in  $\mathbb{G}$  is given by  $\text{supp}(H, \mathbb{G}) := \#\{m : E_H \subseteq E^{(m)}\}$ .

## 2 Related Work

In this section, we review prior work on finding dense subgraphs and quasicliques. Section 2.1 discusses approximation algorithms (with known worst-case bounds) that find the subgraph with maximum density (DS) or its size-constrained variants (DkS, DalkS, DamkS). In Section 2.2 reviews methods for enumerating frequent dense subgraphs present in a relational graph set.

### 2.1 Finding a Dense Subgraph in a Single Graph

The problem of finding maximum density subgraph was studied by Goldberg [11] who introduced a maximum-flow algorithm for this problem (see also [10]). Kannan and Vinay [18] studied the problem for directed graphs and introduced an  $O(\log n)$ -approximation algorithm. Charikar [7] presented a linear programming relaxation from which the optimal solution can be recovered. They also showed that the related greedy algorithm of [2] yields 2-approximation for the problem. The work of Khuller and Saha [20] simplified the analysis in [7]. Bahmani et al. [4] obtained  $O(2 + \epsilon)$ -approximation algorithm for DS in the streaming model which makes  $O(\frac{1}{\epsilon} \log n)$  passes over the data. Recently, Tsourakakis et al. [30] defined the notion of optimal  $\alpha$ -quasiclique (OQC) which maximizes the edge surplus given by  $f_{\alpha}(S) := (|E(S)| - \alpha \binom{|S|}{2})$ ; and, obtained a 2-approximation algorithm for finding optimal quasicliques analogous to Charikar’s algorithm [7]. Comparing their approach ( $\alpha = 1/3$ ) with Charikar’s algorithm on several real-world graphs, the authors argue that OQC yields better results in terms of lower diameter, higher relative density and higher triangle density of the extracted subgraphs.

When there is a constraint on the size of the subgraph ( $|S| = k$ ), the problem of finding the densest  $k$  subgraph (DkS) is NP-Hard [2, 9]. Feige et al. [9] gave an  $O(n^{1/3-\epsilon})$ -approximation algorithm for DkS. The best known current bound for DkS is given by Bhaskara et al. [5] who obtain an  $O(n^{1/4+\epsilon})$ -approximation algorithm for DkS with any  $\epsilon > 0$  having run time  $n^{O(1/\epsilon)}$ . Khot [19] showed that under reasonable complexity assumptions, DkS cannot be approximated within an arbitrary constant factor. Recently, Papailiopoulos et al. [25] obtained graph-dependent bounds for DkS based on low-rank approximation of the adjacency matrix, and experimentally showed that their bounds are tight for several large real-world graphs.

Two variants of the DkS problem where the size constraint is relaxed were introduced in [1], namely, densest at-most- $k$  subgraph (DamkS,  $|S| \leq k$ ) and

densest at-least- $k$  subgraph ( $\text{Dal}k\text{S}$ ,  $|S| \geq k$ ). Andersen and Chellapilla [1] gave an 2-approximation algorithm for  $\text{Dal}k\text{S}$  which was subsequently improved in running time by [20]. Khuller and Saha [20] also showed that  $\text{Dam}k\text{S}$  is as hard as  $\text{D}k\text{S}$ , specifically, an  $\alpha$ -approximation for  $\text{Dam}k\text{S}$  implies a  $4\alpha$ -approximation for  $\text{D}k\text{S}$ .

## 2.2 Finding Cross-Graph Quasicliques

The problem of identifying dense recurrent subgraphs has been studied in several guises with differing terminologies. Early works on frequent subgraph mining were based either on the apriori principle or pattern-growth approach [see 15, forarecentsurvey]. One key bottleneck in general frequent subgraph mining is handling graph (and subgraph) isomorphism; which is absent in relational graph sets since the graphs share a common vertex set.

Yan et al. [31] enumerated frequent *dense* subgraphs in a relational graph set where each edge has support greater than some threshold (frequent) and the subgraph has large minimum cut (dense). Zeng et al. [33] studied the problem of mining frequent quasicliques in a database of vertex labeled graphs by considering  $\gamma$ -quasicliques induced by node sets  $S^{(a)}$  and  $S^{(b)}$  in graphs  $G^{(a)}$  and  $G^{(b)}$  respectively “ $\gamma$ -isomorphic” if there is a one-to-one mapping between  $S^{(a)}$  and  $S^{(b)}$  which preserves their vertex labels. Boden et al. [6] present a method for enumerating frequent quasicliques (minimum support) in edge-labelled graphs.

Hu et al. [12] presented a method for identifying *coherent* dense subgraphs from gene microarray expression datasets by finding dense subgraphs in the *summary graph*  $G_S = (V_S, E_S)$  where  $V_S \subset V \times V$  and nodes  $(u, v)$  and  $(w, z)$  have an edge in  $E_S$  if their support sets in  $\mathbb{G}$  have high Jaccard similarity. Pei et al. [26] presented an exhaustive approach for enumerating all cross-graph quasicliques defined as follows: given relational graph set  $\mathbb{G}$ , parameters  $\gamma^{(1)}, \gamma^{(2)}, \dots, \gamma^{(M)}$  and  $\text{min\_sup} \in (0, 1]$ ; a vertex set  $S$  is a frequent cross-graph quasiclique (fCGQC) if it has relative density  $\rho_m(S) \geq \gamma^{(m)}$  in at least  $(M \cdot \text{min\_sup})$  graphs.

Li et al. [24] present a method for identifying recurrent dense subgraphs in weighted graphs based on a non-convex optimization problem which is solved approximately using multi-state convex relaxation [34]. This is used to identify several recurrent heavy subgraphs in multiple co-expression networks.

Jethava et al. [13] present a parameter-less algorithm based on a multiple kernel learning (MKL) [3, 27] formulation for finding an ordering of vertices which maximizes the minimum relative density (across all graphs) of the induced subgraph. Their approach also provides weak graph-dependent bounds on the density of the induced subgraphs [see 14, Lemma 11 and Theorem 12]. However, their method has  $O(n^3)$  complexity which cannot scale to large graphs.

## 3 Methods

We are interested in finding a set of nodes which induces a dense subgraph in each of the graphs in  $\mathbb{G}$ . We formalize this notion by defining the problem

of densest common subgraph (DCS). Let  $\delta_{\mathbb{G}}(S) := \min_{G^{(m)} \in \mathbb{G}} \delta_m(S)$  denote the density of the subgraph having minimum density among the subgraphs induced by  $S$  in graphs  $G^{(m)} \in \mathbb{G}$ . In the sequel, we refer to  $\delta_{\mathbb{G}}(S)$  as the *common density* of vertex set  $S \subseteq V$  in graph set  $\mathbb{G}$ .

**Definition 1 (Densest Common Subgraph).** *Given relational graph set  $\mathbb{G}$ , the densest common subgraph is given by:*

$$S_{\text{DCS}} := \arg \max_{S \subseteq V} \delta_{\mathbb{G}}(S), \tag{1}$$

We use shorthand notation  $\delta_{\mathbb{G}} := \delta_{\mathbb{G}}(S_{\text{DCS}})$  to denote the minimum density of any subgraph induced by  $S_{\text{DCS}}$  in the graph set  $\mathbb{G}$ .

The DCS problem is related to the  $k$ -multicut (following Goldberg’s construction [11]) which is known to be NP-Hard for  $k \geq 3$ , and therefore, DCS is suspected to be NP-Hard. However, a formal proof of hardness is non-trivial and it constitutes an interesting problem for future research.

We can define the LP relaxation (DCS\_LP) of the DCS problem as:

$$\begin{aligned} \max \quad & t \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \leq 1 \\ & \sum_{ij \in E^{(m)}} x_{ij}^{(m)} \geq t \quad \forall E^{(m)}, m \in [M] \\ & x_{ij}^{(m)} \leq y_i, x_{ij}^{(m)} \leq y_j \quad \forall ij \in E^{(m)} \\ & x_{ij}^{(m)} \geq 0, y_i \geq 0 \quad \forall ij \in E^{(m)}, \forall i \in [n], \end{aligned} \tag{2}$$

where  $(t, \{x_{ij}^{(m)}\}_{ij \in E^{(m)}}, \{y_i\}_{i \in [n]})$  denote the primal variables. We make the following observations:

**Lemma 1.** *For any  $S \subseteq V$ , the optimal value of DCS\_LP in (2) is at least  $\delta_{\mathbb{G}}(S)$ . In particular, the optimal value of DCS\_LP is an upper bound on  $\delta_{\mathbb{G}}$ .*

*Proof.* Suppose  $|S| = k$ . We construct a feasible solution as follows:

$$y_i = \begin{cases} \frac{1}{k} & \text{if } i, j \in S \\ 0 & \text{otherwise} \end{cases}, \quad x_{ij}^{(m)} = \begin{cases} \frac{1}{k} & \text{if } i, j \in S \\ 0 & \text{otherwise} \end{cases}.$$

Then,  $\sum_{ij \in E^{(m)}} x_{ij}^{(m)} = \frac{1}{k} \sum_{ij \in E^{(m)}(S)} 1 = \delta_{G^{(m)}}(S)$  and  $t = \delta_{\mathbb{G}}(S)$ . □

We consider an algorithm which solves DCS\_LP and returns  $S(r) = \{i : y_i^* \geq r\}$  which maximizes  $\delta_{\mathbb{G}}(S(r))$ . Note that there are at most  $n$  sets to consider corresponding to distinct values of  $y_i^*$ . The following theorem provides a lower bound on the quality of the returned subgraph.

**Lemma 2.** *Let  $(t^*, x_{ij}^{(m)*}, y_i^*)$  denote optimal solution for DCS\_LP (2) and let  $S := \{i : y_i^* > 0\}$  denote the set of vertices with non-zeros  $y_i^*$ ’s with  $k := |S|$  and  $y_{\min} = \min_{i \in S} y_i^*$ . The following hold:*

(a) *If  $y_i^* = \frac{1}{k} \forall i \in S$ , then  $S$  is a densest common subgraph, i.e.,  $\delta_{\mathbb{G}}(S) = \delta_{\mathbb{G}}$ .*

- (b) For each graph  $G^{(m)} \in \mathbb{G}$ , there exists  $S^{(m)} \subseteq S$  which induces a subgraph in  $G^{(m)}$  with density at least  $t^*$ , i.e.,  $\delta_{G^{(m)}}(S^{(m)}) \geq t^*$ .
- (c) The density of the subgraph induced by  $S$  in any graph  $G^{(m)}$  is at least  $\frac{t^* \lceil 2t^* + 1 \rceil}{k}$ , i.e.,  $\delta_{\mathbb{G}}(S) \geq \frac{t^* \lceil 2t^* + 1 \rceil}{k}$ .

*Proof (of Lemma 2).* Without loss of generality, assume  $x_{ij}^{(m)*} = \min(y_i^*, y_j^*)$  since for any feasible solution  $(x, y, t)$ , we can construct another solution by setting  $\bar{x}_{ij}^{(m)} := \min(y_i, y_j) \geq x_{ij}^{(m)}$  with  $\bar{t} \geq t$ .

- (a) If  $\mathbf{y}^* = \frac{1}{K} \mathbf{e}_S$ , then  $\bar{t} = \delta_{\mathbb{G}}(S)$ . By Lemma 1,  $t^* \geq \delta_{\mathbb{G}}$  and the result follows.
- (b) Following the analysis of Charikar [7, Lemma 2], we define collection of sets indexed by a parameter  $r \geq 0$ . Let  $S(r) := \{i : y_i^* \geq r\}$  and  $E^{(m)}(r) := \{ij \in E^{(m)} : x_{ij}^{(m)*} \geq r\}$ . Since  $x_{ij}^{(m)*} = \min(y_i^*, y_j^*)$  by construction,

$$ij \in E^{(m)}(r) \Leftrightarrow i, j \in S(r).$$

Now,  $\int_0^\infty |S(r)| dr = \sum_i y_i^* \leq 1$  and  $\int_0^\infty |E^{(m)}(r)| dr = \sum_{ij \in E^{(m)}} x_{ij}^{(m)*} \geq t^*$ . Then, for each  $G^{(m)} \in \mathbb{G}$ , there exists  $r^{(m)}$  such that  $\frac{|E^{(m)}(r^{(m)})|}{|S(r^{(m)})|} \geq t^*$ . Otherwise, it leads to the following contradiction:

$$t^* \leq \int_0^\infty |E^{(m)}(r)| dr < t^* \int_0^\infty |S(r)| dr \leq t^*$$

We define  $S^{(m)} := S(r^{(m)}) \subseteq S$  which induces a subgraph of density at least  $t^*$  in graph  $G^{(m)} \in \mathbb{G}$ .

- (c) We observe  $|S^{(m)}| \geq \lceil 2t^* + 1 \rceil$  since  $\delta_m(S^{(m)}) \geq t^*$ . Consequently, for each graph  $G^{(m)} \in \mathbb{G}$ ,

$$\delta_m(S) = \frac{|E^{(m)}(S)|}{k} \geq \frac{|E^{(m)}(S^{(m)})|}{k} = \frac{|S^{(m)}|}{k} \delta_m(S^{(m)}) \geq \frac{\lceil 2t^* + 1 \rceil}{k} t^*. \quad \square$$

The LP optimal  $t^*$  is at most  $\delta_{\mathbb{G}}^{\min} := \min_{G^{(m)}} \delta_{G^{(m)}}$  where  $\delta_{G^{(m)}}$  denotes the density of the densest subgraph in  $G^{(m)}$ . This yields an upper bound on the integrality gap of DCS\_LP given by  $\frac{\delta_{\mathbb{G}}^{\min}}{\delta_{\mathbb{G}}}$ . The LP solution is particularly interesting whenever  $\mathbf{y}^* = \frac{1}{|S|} \mathbf{e}_S$  since we get proof of optimality in that case. In the general case, DCS\_LP can exhibit both integrality gap higher than 1 and an approximation ratio lower than 1 (obtained by  $\delta_{\mathbb{G}}(S)$ ) in contrast to Charikar’s LP in the densest subgraph problem (Section 4.1).

Furthermore, solving DCS\_LP has running time polynomial in the number of edges in the graph set  $(n_{\mathbb{G}}^{O(1)})$  which cannot be scaled to large graphs having more than a few hundred thousand edges. In the next section, we consider a simple greedy algorithm for finding common dense subgraphs in large graphs.

---

**Algorithm 1**  $(S, r) = \text{DCS\_GREEDY}(\mathbb{G})$

---

```

1: Initialize  $V_1 := V$ 
2: while  $\delta_{\mathbb{G}}(V_t) > 0$  do
3:    $m_t := \arg \min_m \delta_m(V_t)$             $\{G^{(m)}$  with min. density induced subgraph $\}$ 
4:    $V' := \{i \in V_t : \text{deg}_{m_t}(i, V_t) > 0\}$     $\{\text{Non-isolated nodes in } G^{(m_t)}(V_t)\}$ 
5:    $i_t := \arg \min_{i \in V_t} \text{deg}_{m_t}(i, V_t)$         $\{\text{minimum degree node in } G^{(m_t)}(V_t)\}$ 
6:   For all  $m$ , assign all edges  $(i_t, j) \in E^{(m)}$  and  $(j, i_t) \in E^{(m)}$  to node  $i_t$ .
7:   Set  $\text{deg}_t^+ := \max_m \text{deg}_m(i_t, V_t)$           $\{\text{Max. degree of } i_t \text{ in any } G^{(m)}(V_t)\}$ 
8:   Set  $\text{deg}_t^- := \min_{i \in V'} \text{deg}_{m_t}(i, V')$       $\{\text{Min. non-zero degree in } G^{(m_t)}(V_t)\}$ 
9:   Set  $k_t = \frac{\text{deg}_t^+}{\text{deg}_t^-}$  and  $r_t = k_t \frac{|V_t|}{|V'|}$             $\{\text{deg}_t^+ \leq 2r_t \delta_{\mathbb{G}}(V_t)\}$ 
10:   $V_{t+1} \leftarrow V_t \setminus i_t$ 
11:   $t \leftarrow t + 1$ 
12: end while
13:  $S' := V_t$                                       $\{\text{Clean-up phase}\}$ 
14: while  $|S'| > 0$  do
15:   Choose random  $i' \in S'$ 
16:    $\forall m$ , assign edges  $(i', j) \in E^{(m)}(S')$  and  $(j, i') \in E^{(m)}(S')$  to  $i$ .
17:    $S' \leftarrow S' \setminus i'$ 
18: end while
19: return  $S := \arg \max_{V_t} \delta_{\mathbb{G}}(V_t)$  and  $r := \max_t r_t$ 

```

---

### 3.1 A Greedy Algorithm for Densest Common Subgraph

In this section, we consider a peeling algorithm (DCS\_GREEDY) for solving densest common subgraph problem. The algorithm iteratively constructs vertex set  $V_{t+1}$  at each time  $t$  by removing the node  $i_t$  from  $V_t$  where  $i_t$  is the minimum degree node in the subgraph  $G^{(m_t)}(V_t)$  where  $G^{(m_t)}(V_t)$  has the minimum density among the subgraphs induced by  $V_t$  in graph set  $\mathbb{G}$ . All edges  $(i_t, j)$  and  $(j, i_t)$  present in induced subgraphs  $G^{(m)}(V_t)$  are assigned to node  $i_t$ . The set  $V_t$  which maximizes  $\delta_{\mathbb{G}}(V_t)$  is returned as solution. Algorithm 1 gives the complete pseudocode for DCS\_GREEDY.

We now consider an analysis of the above algorithm. We have the following invariant: after each iteration  $t$ , the set of edges  $E^{(m)} \cap (V_t \times V_t)$  are unassigned while all other edges are assigned to a node in  $V \setminus V_t$ . At the termination of the algorithm, either all edges  $(i, j) \in E^{(m)} \forall m$  are assigned to  $i$ , or all edges  $(i, j) \in E^{(m)} \forall m$  are assigned to  $j$ .

Let  $d_{max}^{(m)}$  denote the maximum number of edges  $(i, j)$  or  $(j, i)$  assigned to any node  $i$  in graph  $G^{(m)}$  and let  $d_{minmax}$  denote the minimum value of  $d_{max}^{(m)}$  among all graphs  $G^{(m)} \in \mathbb{G}$ . The following holds:

**Lemma 3.** *For any  $S \subseteq V$ , the value  $\delta_{\mathbb{G}}(S)$  is at most  $d_{minmax}$ . In particular,  $d_{minmax}$  is an upper bound on  $\delta_{\mathbb{G}}$ .*

*Proof (Sketch).* Each edge in  $E^{(m)}(S)$  is assigned to a vertex in  $S$ , and therefore,  $|E^{(m)}(S)| \leq d_{max}^{(m)} \cdot |S|$ . Consequently,  $\delta_{\mathbb{G}}(S) \leq d_{minmax}$ . □

The greedy algorithm of Charikar [7] yields a 2-approximation for the densest subgraph problem due to the following property: in the densest subgraph, the degree of the minimum degree subgraph is at least the average degree otherwise it can safely be removed to yield an even denser subgraph. This property is not true for DCS. The following result gives an lower bound on the quality of the solution returned by DCS\_GREEDY.

**Lemma 4.** *The value  $d_{minmax}$  is at most  $(2r \cdot \delta_{\mathbb{G}}(S))$  where  $(S, r)$  is the solution returned by the DCS\_GREEDY algorithm. In particular,  $\delta_{\mathbb{G}} \leq (2r \cdot \delta_{\mathbb{G}}(S))$ .*

*Proof.* Let  $m' = \arg \min d_{max}^{(m)}$  and  $i'$  denote the node in graph  $G^{(m')}$  which has the maximum number of edges assigned to it. Let  $t'$  denote the iteration in which node  $i'$  is removed from  $V_{t'}$  during the execution of DCS\_GREEDY. By construction, edges are assigned to any node only when it is removed from the graph and therefore,  $d_{minmax} = \deg_{t'}^+$ . Further, by definition of  $\deg_t^-$ , we have  $\deg_t^- \leq 2 \frac{|E^{(m_t)}(V_t)|}{|V_t|} = 2 \delta_{\mathbb{G}}(V_t)$ . Combining, we get

$$d_{minmax} = \deg_{t'}^+ \leq 2r_{t'} \delta_{\mathbb{G}}(V_{t'}) \leq 2r \delta_{\mathbb{G}}(S). \quad \square$$

The DCS\_GREEDY algorithm can be efficiently implemented in  $O(n + n_{\mathbb{G}})$  time by maintaining a list of node degrees for each graph  $G^{(m)}$  and updating the neighbours of a node  $v$  in the degree list whenever node  $v$  is removed.

### 3.2 Densest Common at Least- $k$ Subgraph (DCalkS)

We next consider the Densest Common at least- $k$  Subgraph (DCalkS) problem which imposes constraint on the minimum size of the induced subgraphs.

**Definition 2 (Densest Common at Least- $k$  Subgraph).** *Given relational graph set  $\mathbb{G}$ , the densest common at least- $k$  subgraph (DCalkS) is given by,*

$$H := \arg \max_{S \subseteq V: |S| \geq k} \delta_{\mathbb{G}}(S). \tag{3}$$

In the sequel, we use the shorthand notation  $n_h := |H|$  and  $\delta_{\geq k} := \delta_{\mathbb{G}}(H)$  to denote the cardinality of the DCalkS subgraph and the minimum density of any subgraph by  $H$  induced in any graph  $G^{(m)} \in \mathbb{G}$  respectively.

We note that DCalkS is NP-Hard (by restriction to DalkS whenever  $|\mathbb{G}| = 1$ ) since DalkS is known to be NP-Hard [20, Theorem 3.1]. Consider the linear program P2(c) given by:

$$\begin{aligned} & \max t \\ & \text{s.t. } \sum_{i=1}^n y_i = 1 \\ & \sum_{ij \in E^{(m)}} x_{ij}^{(m)} \geq t \quad \forall E^{(m)}, m \in [M] \\ & x_{ij}^{(m)} \leq y_i, x_{ij}^{(m)} \leq y_j \quad \forall ij \in E^{(m)} \\ & x_{ij}^{(m)} \geq 0, y_i \geq 0, y_i \leq \frac{1}{c} \quad \forall ij \in E^{(m)}, \forall i \in [n]. \end{aligned} \tag{4}$$

where  $c \geq k$  is our guess for the size of the optimal DCalkS solution ( $n_h$ ). The following result is a direct consequence of Lemma 1.



**Lemma 5.** For any  $S \subseteq V$  with  $c := |S| \geq d$ , the optimal value of  $P2(d)$  is at least  $\delta_G(S)$ . In particular, the optimal value  $P2(k)$  is an upper bound on  $\delta_{\geq k}$ .

We consider an algorithm (DCalkS-LP) which solves  $P2(k)$  and returns  $\{i : y_i^* > 0\}$ . Then, the following holds:

**Lemma 6.** Let  $(t^*, x_{ij}^{(m)*}, y_i^*)$  denote optimal solution for  $P2(k)$  in (4) and let  $S := \{i : y_i^* > 0\}$  denote the set of vertices with non-zeros  $y_i^*$ 's with  $l := |S| \geq k$ . Then,

- (a) if  $y_i^* = \frac{1}{l} \forall i \in S$  and zero otherwise, then  $\delta_G(S) = \delta_{\geq k}$ , i.e.,  $S$  is a densest common at least- $k$  subgraph; and,
- (b) the density of the subgraph induced by  $S$  in any graph  $G^{(m)}$  is at least  $\frac{t^* \lceil 2t^* + 1 \rceil}{l}$ , i.e.,  $\delta_G(S) \geq \frac{t^* \lceil 2t^* + 1 \rceil}{l}$ .

*Proof (Sketch).* The proof is analogous to the proof for Theorem 2. □

## 4 Experiments

We compare our approach (DCS\_LP and DCS\_GREEDY) with MKL-based formulation (MKL) in [13] and the greedy algorithm (CHARIKAR) of [7]. In order to compare with [13], we construct the sets  $S(r) = \{i \in V : \alpha_i^* \geq r\}$  where  $\alpha^*$  is the solution of the MKL optimization in [13] and choose the set which maximizes  $\delta_G(S(r))$ , i.e.,  $S_{MKL} := \arg \max_{S(r)} \delta_G(S(r))$ . We run the greedy algorithm (CHARIKAR) to find a dense subgraph in the intersection graph  $G_{\cap}$ .

The tensor-based approach in [24] is not considered since it has significantly higher computational complexity (e.g., the authors report a running time of 200 hours on a high-performance computing node for analysis of co-expression networks [See 24, SupplementaryText,S6]).

All experiments were implemented in MATLAB R2014A and run on a laptop having 2.8GHz Intel Core i5 processor and 16GB RAM. The greedy algorithm was coded in MATLAB while the linear programs were solved using Gurobi 6.0 solver using its MATLAB API. Gurobi <sup>3</sup> is a state-of-the-art commercial solver for linear and non-linear optimization implemented in C programming language with APIs for several other programming languages.

### 4.1 Synthetic Dataset

We investigate the integrality gap and approximation ratio of DCS\_LP by considering small graphs for which we can find the DCS solution by exhaustive search. Each graphset consists of  $m$  graphs each of which is generated independently as follows: We generate an Erdős-Renyi random graph  $G(n, p)$  and then plant a clique of size  $k$  randomly. We generate a dataset having 100 such graphsets with the following parameters:  $m = 3$ ,  $n = 20$ ,  $p \sim Uniform(0, 0.5)$  and  $k \sim Uniform[1, \dots, n/2]$ .

<sup>3</sup> <http://www.gurobi.com>

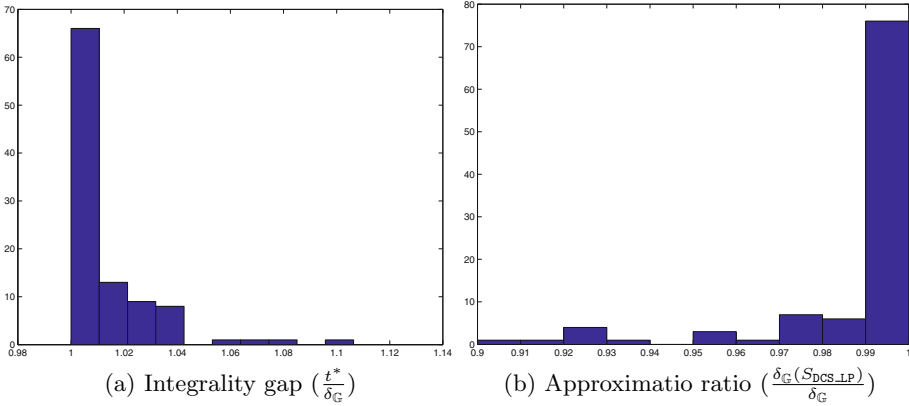


Fig. 1. Properties of DCS\_LP

Figure 1 shows the histogram of (a) the LP objective value  $t^*$ , and, (b) the common density of subgraph obtained by rounding the LP solution  $S = \{i : y_i^* > 0\}$  relative to the optimum solution  $\delta_G$ , respectively. DCS\_LP recovers the optimum solution in 70% of the cases but in general, it can and does exhibit both integrality gap  $t^* > \delta_G$  and sub-optimal rounding  $\delta_G(S) < \delta_G$ .

### 4.2 Real-World Datasets

We consider the following datasets for evaluation of our methods:

- DIMACS. The DIMACS 1994 dataset <sup>4</sup> is a comprehensive benchmark for testing of clique finding and related algorithms. Each of the graph families in DIMACS (*brock*, *c-fat*, *p-hat*, *san*, *sanr*) is motivated by carefully selected real world problems e.g. fault diagnosis (*c-fat*), etc.; thus covering a wide range of practical scenarios [17]. This was used for experimental evaluation in [13] and we repeat their experimental setup.
- SNAP-AS. We consider the graph sets *Oregon-1* and *Oregon-2* from SNAP network database <sup>5</sup> [22, 23] related to autonomous systems. Each dataset consists of  $M = 9$  graphs having  $\sim 11000$  nodes and  $20000 - 30000$  edges.
- SNAP-AMAZON. This dataset consists of directed graphs *amazon0312*, *amazon0505* and *amazon0601* available from SNAP network database which consist of Amazon product co-purchasing networks on specific dates. We make the graphs undirected (by introducing edges  $(j, i)$  whenever  $(i, j)$  is present) and consider the nodes which are present in all three graphs. This yields a graph set having  $n = 400727$  nodes and  $n_G = 7157921$  edges in total.

<sup>4</sup> <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/cliq/>

<sup>5</sup> <http://snap.stanford.edu/data/>

**Table 1.** Comparison of DCS\_LP and DCS\_GREEDY with MKL [13] and CHARIKAR [7] for finding DCS in DIMACS and SNAP-AS datasets.  $M$ ,  $n$ ,  $n_G$  denote resp. the number of graphs, number of nodes in each graph, and the total number of edges summed over all the graphs in the graph set.  $|S|$  and  $\delta_G(S)$  denote the size and common density of the induced subgraph found by each method.  $\delta_{\cap}(S)$  (in CHARIKAR) is the density of the subgraph in the intersection graph  $G_{\cap}$  and  $t^*$  denotes the optimum of DCS LP. X marks the instances where the DCS LP failed to converge within 2 hours.

$\mathbb{G}$	$M$	$n$	$n_G$	MKL [13]		DCS_LP			DCS_GREEDY		CHARIKAR [7]		
				$ S $	$\delta_G(S)$	$ S $	$\delta_G(S)$	$t^*$	$ S $	$\delta_G(S)$	$ S $	$\delta_G(S)$	$\delta_{\cap}(S)$
<i>c-fat200</i>	3	200	13242	100	7.93	101	8.01	8.08	102	8.04	199	7.63	1.81
<i>c-fat500</i>	4	500	83416	140	9.65	140	9.65	9.65	140	9.65	140	9.65	9.65
<i>brock200</i>	4	200	29753	200	25.33	200	25.33	25.33	200	25.33	149	19.31	1.93
<i>brock400</i>	4	400	80245	400	50.04	400	50.04	50.04	400	50.04	110	14.08	1.20
<i>brock800</i>	4	800	447753	800	139.29	X	X	X	800	139.29	783	136.41	5.99
<i>p.hat300</i>	3	300	66251	300	36.44	286	36.65	36.65	286	36.65	286	36.65	36.65
<i>p.hat500</i>	3	500	188315	500	63.14	489	63.21	63.21	489	63.21	489	63.21	63.21
<i>p.hat700</i>	3	700	365737	700	87.14	X	X	X	679	87.27	679	87.27	87.27
<i>p.hat1000</i>	3	1000	738798	1000	122.25	X	X	X	973	122.41	973	122.41	122.41
<i>p.hat1500</i>	3	1500	1701127	1500	189.95	X	X	X	1478	190.05	1478	190.05	190.05
<i>san200</i>	5	200	17910	200	9.95	200	9.95	9.95	200	9.95	2	0.50	0.50
<i>san400</i>	3	400	119700	400	19.95	400	19.95	19.95	400	19.95	176	9.14	0.66
<i>sanr200</i>	2	200	8069	200	10.19	199	10.19	10.19	199	10.19	170	9.23	3.17
<i>sanr400</i>	2	400	63747	400	59.83	400	59.83	59.83	400	59.83	399	59.71	29.97
<i>Oregon-1</i>	9	11492	203127	54	11.37	76	12.03	12.03	76	12.03	64	11.28	10.09
<i>Oregon-2</i>	9	11806	284031	173	22.35	X	X	X	119	22.45	115	21.23	17.57

Table 1 shows the results for DIMACS and SNAP-AS datasets. In all instances (except *c-fat200*) where DCS\_LP finishes within time, the DCS\_LP solution is optimal as verified by  $t^* = \delta_G(S)$ . Further, DCS\_GREEDY also finds the optimal solution in these instances. Both our methods out-perform MKL and CHARIKAR in all graph sets. This is especially striking in the case of *san* and *sanr* graph sets where the greedy algorithm (CHARIKAR) yields very poor results – highlighting the fact that taking the intersection of the graphs is unsuitable.

We note that the high computational complexity of DCS\_LP ( $n_G^{O(1)}$ ) and MKL ( $O(n^3)$ ) prohibits their use for finding the densest common subgraph in the SNAP-AMAZON graph set. The DCS\_GREEDY algorithm finds a subgraph with  $\delta_G(S) = 5.90251$  while CHARIKAR finds a subgraph with  $\delta_G(S) = 2.5$ .

## 5 Discussion

This paper formalizes the Densest Common Subgraph (DCS) problem which extends the notion of densest subgraph to a relational graph set. We present an extension of Charikar’s linear programming approach [7] to the problem of finding the densest common subgraph in a relational graph set.

The LP-based approach recovers the densest common subgraph in many cases (with proof of optimality). In other cases, it provides an upper bound on the common density (e.g. in the case of *c-fat200* graph set in the experiments) and a good starting point for further heuristic search approaches. We note that in the worst-case, the approximation guarantee is  $O(\frac{n}{\delta_G})$  – which can be trivially obtained by taking original vertex set  $V$  as a solution. A tighter analysis of the LP relaxation can reveal more insight into the problem.

Our greedy algorithm `DCS.GREEDY` can be scaled to large graphs which is not possible with existing methods. Further, it substantially improves over the greedy approach (`CHARIKAR`) which only considers the integrated graph by taking intersection of the different edge sets. We note that the `DCS.GREEDY` algorithm is closely related to the dual LP. Designing a combinatorial primal-dual algorithm can lead to better results and will be addressed in future work.

**Acknowledgments.** We thank Chiranjib Bhattacharyya, Devdatt Dubhashi and Jack Kuipers for valuable comments and suggestions. Vinay Jethava is funded by the MERiC project as part of the European Research Council (ERC) synergy grant 2013.

## References

1. Andersen, R., Chellapilla, K.: Finding dense subgraphs with size bounds. In: Avrachenkov, K., Donato, D., Litvak, N. (eds.) WAW 2009. LNCS, vol. 5427, pp. 25–37. Springer, Heidelberg (2009)
2. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding a dense subgraph. *Journal of Algorithms* **34**(2), 203–221 (2000)
3. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 6 (2004)
4. Bahmani, B., Kumar, R., Vassilvitskii, S.: Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment* **5**(5), 454–465 (2012)
5. 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: Proceedings of the Forty-Second ACM Symposium on Theory of Computing (STOC), pp. 201–210. ACM (2010)
6. Boden, B., Günnemann, S., Hoffmann, H., Seidl, T.: Mining coherent subgraphs in multi-layer graphs with edge labels. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1258–1266. ACM (2012)
7. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: Jansen, K., Khuller, S. (eds.) APPROX 2000. LNCS, vol. 1913, pp. 84–95. Springer, Heidelberg (2000)
8. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms* **41**(2), 174–211 (2001)
9. Feige, U., Peleg, D., Kortsarz, G.: The dense  $k$ -subgraph problem. *Algorithmica* **29**(3), 410–421 (2001)
10. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing* **18**(1), 30–55 (1989)

11. Goldberg, A.V.: Finding a maximum density subgraph. University of California at Berkeley, Berkeley (1984)
12. Hu, H., Yan, X., Huang, Y., Han, J., Zhou, X.J.: Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* **21**(suppl 1), i213–i221 (2005)
13. Jethava, V., Martinsson, A., Bhattacharyya, C., Dubhashi, D.: The lovasz  $\theta$  function, svms and finding large dense subgraphs. In: *Neural Information Processing Systems (NIPS)*, pp. 1169–1177 (2012)
14. Jethava, V., Martinsson, A., Bhattacharyya, C., Dubhashi, D.: Lovasz theta function, SVMs and finding dense subgraphs. *Journal of Machine Learning Research* **14**, 3495–3536 (2014)
15. Jiang, C., Coenen, F., Zito, M.: A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review* **28**(01), 75–105 (2013)
16. Jiang, D., Pei, J.: Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data (TKDD)* **2**(4), 16 (2009)
17. Johnson, D.S., Trick, M.A.: Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11–13, 1993, vol. 26. Amer Mathematical Society (1996)
18. Kannan, R., Vinay, V.: Analyzing the structure of large graphs. Rheinische Friedrich-Wilhelms-Universität Bonn (1999)
19. Khot, S.: Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing* **36**(4), 1025–1071 (2006)
20. Khuller, S., Saha, B.: On finding dense subgraphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettseas, S., Thomas, W. (eds.) *ICALP 2009, Part I. LNCS*, vol. 5555, pp. 597–608. Springer, Heidelberg (2009)
21. Lee, V.E., Ruan, N., Jin, R., Aggarwal, C.: A survey of algorithms for dense subgraph discovery. *Managing and Mining Graph Data* 303–336 (2010)
22. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection (June 2014). <http://snap.stanford.edu/data>
23. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp. 177–187. ACM (2005)
24. Li, W., Liu, C.-C., Zhang, T., Li, H., Waterman, M.S., Zhou, X.J.: Integrative analysis of many weighted co-expression networks using tensor computation. *PLoS computational biology* **7**(6), e1001106 (2011)
25. Papailiopoulos, D., Mitliagkas, I., Dimakis, A., Caramanis, C.: Finding dense subgraphs via low-rank bilinear optimization. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1890–1898 (2014)
26. Pei, J., Jiang, D., Zhang, A.: Mining cross-graph quasi-cliques in gene expression and protein interaction data. In: *Proceedings of the 21st International Conference on Data Engineering. ICDE 2005*, pp. 353–356. IEEE (2005)
27. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: Simplemkl. *Journal of Machine Learning Research* **9**, 2491–2521 (2008)
28. Ravi, S.S., Rosenkrantz, D.J., Tayi, G.K.: Heuristic and special case algorithms for dispersion problems. *Operations Research* **42**(2), 299–310 (1994)
29. Srivastav, A., Wolf, K.: Finding dense subgraphs with semidefinite programming. In: Jansen, K., Rolim, J.D.P. (eds.) *APPROX 1998. LNCS*, vol. 1444, pp. 181–191. Springer, Heidelberg (1998)

30. Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 104–112. ACM (2013)
31. Yan, X., Zhou, X., Han, J.: Mining closed relational graphs with connectivity constraints. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 324–333. ACM (2005)
32. Yuan, X.-T., Zhang, T.: Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research* **14**(1), 899–925 (2013)
33. Zeng, Z., Wang, J., Zhou, L., Karypis, G.: Coherent closed quasi-clique discovery from large dense graph databases. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 797–802. ACM (2006)
34. Zhang, T.: Multi-stage convex relaxation for non-convex optimization. Technical report, Rutgers University (2009)