Dense and well-connected subgraph detection in dual networks

Tianyi Chen* Francesco Bonchi† David Garcia-Soriano‡ Atsushi Miyauchi§
Charalampos E. Tsourakakis¶

Abstract

Dense subgraph discovery is a fundamental problem in graph mining whose goal is to extract a dense subgraph from a given graph, and it has a wide range of applications [18]. However, numerous real-world applications, ranging from computational biology and computational neuroscience to computational social science, take as input a *dual* graph, namely a pair of graphs on the same set of nodes. Despite the large number of such applications, research on dense subgraph discovery has focused on a single graph input, with few notable exceptions [9, 22, 35, 36]. In this work, we contribute to this line of research by studying the following novel algorithmic problem:

> Given a pair of graphs G, H on the same set of nodes V, how do we find a subset of nodes $S \subseteq V$ that induces a well-connected subgraph in G and a dense subgraph in H?

Our formulation generalizes previous research [11, 44, 45], by enabling to *control* the connectivity constraint on G. We propose a mathematical formulation and prove that it is solvable exactly in polynomial time. We compare our method to state-of-the-art competitors and find empirically that controlling the connectivity constraint enables the practitioner to obtain information that is otherwise inaccessible. Finally, we show that our proposed mining tool can be used to better understand how users interact on Twitter and connectivity aspects of human brain networks with and without Autism Spectrum Disorder (ASD).

Keywords: dense subgraph discovery, k-edge connectivity, dual graph, algorithm design, graph mining

1 Introduction

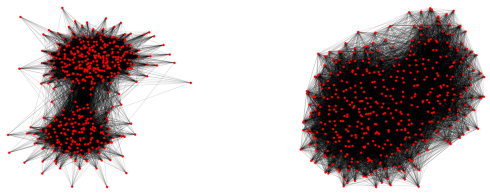
Dense subgraph discovery (DSD) is a major graph mining area of research that has mostly focused on extracting a dense cluster from a single graph given as (part of the) input [18]. However, numerous real-world settings involve a pair of graphs on the same vertex set. For example, in neuroscience, an important pair of brain networks is given by the *structural connectivity* and *functional connectivity* graphs, defined on the basis of correlation on the time series of activity of the different brain regions (co-activation) [16, 27]. In bioinformatics, differential coexpression network analysis of gene expression data is used to analyze gene-to-gene coexpression jointly across two different networks, especially in cancer research [20]. In computational social science, we are interested in understanding better how users interact on social media such as Twitter, where for instance, users may retweet each other, and may favorite certain tweets of other users. These interactions naturally induce two graph layers, the *retweet* layer and the *favorite layer*. Qi et al. [34] use two Flickr graph topologies to mine online communities; one topology is naturally induced by online friendships between users, while the second one is created from log files by adding an edge between two users if there is a photo liked by both of them. In reality mining, we are interested in understanding relationships defined by Bluetooth scans and phone calls respectively [13, 14]. Furthermore, dual graphs are a special case of multilayer networks [29], when the number of layers is equal to 2. Despite the large amount of research on DSD, only few works study the problem of dense subgraph discovery across more than one graphs [9, 22, 25, 35, 36, 44, 45].

In this work we introduce the following problem that generalizes the works of Wu et al. [44, 45] and Cui et al. [11]:

Problem 1. *Given two simple, unweighted, undirected graphs $G = (V, E_G)$ and $H = (V, E_H)$, find a set of nodes $S \subseteq V$ such that $G[S]$ is well-connected and $H[S]$ is dense.*

In those works [11, 44, 45] the goal is to ensure $G[S]$ is (simply) connected and $H[S]$ is dense. To the best of our knowledge, our work is the first contribution towards enabling full control over the connectivity constraint, a powerful feature of our method. Specifically, our

*Boston University. ctony@bu.edu
†ISI Foundation, Turin. francesco.bonchi@isi.it
‡ISI Foundation, Turin. d.garcia.soriano@isi.it
§University of Tokyo. miyauchi@mist.i.u-tokyo.ac.jp
¶Boston University and ISI Foundation. tsourolampis@gmail.com



(a) (b)

Figure 1: Visualization of our algorithm's output on the Twitter retweet (G) and reply (H) graphs, with connectivity requirement $k = 8$. (a) 8-edge-connected retweet subgraph. (b) Dense reply subgraph with min degree 51.

contributions include the following:

• **Problem formulation.** We carefully formalize Problem 1 using for the connectivity and density constraints the well-established graph theoretic notions of k-edge-connectivity and minimum degree respectively. Specifically, we study algorithmically the following problem:

PROBLEM 2. *Given two simple, unweighted, undirected graphs $G = (V, E_G)$ and $H = (V, E_H)$, and positive integer k, find a set of nodes $S \subseteq V$ such that $G[S]$ is k-edge connected and the minimum degree in $H[S]$ is maximized.*

We shall also refer to k value as the connectivity requirement. Perhaps surprisingly, the choice of the minimum degree over the average degree as the density measure allows to solve Problem 2 efficiently. In contrast the average degree objective leads to an intractable formulation (see Section 2).

• **Algorithm design.** We prove that Problem 2 is solvable in polynomial time. Our proof is constructive, i.e., we design a polynomial-time exact algorithm that scales to large networks. Figure 1 shows the output of our algorithm on two layers of the Twitter network, namely the retweet and the reply layers. Specifically, Figure 1(a) depicts the 8-edge-connected retweet subgraph, and Figure 1(b) the resulting dense subgraph on the reply layer, both induced by the optimal set of nodes found by our algorithm. For more details, see Section 4.

• **Ranging connectivity.** A key contribution of our work over prior methods is our ability to control the connectivity requirement. Figure 2 shows a preview of the range of connectivity values on the dual Twitter graph (reply, quote) that the existing approaches BFF [36] and k-CCO [11] output. The plot for BFF is annotated by the specific version of the BFF problem (see Section 2 for more details). Observe that maximizing the average degree over the union of the two graphs results in a densest subgraph that is 1-edge-connected. k-CCO also outputs a simply connected graph on G, that can be disconnected by the removal of a single edge. While BFF-MM achieves high edge connectivity, only our method can *control* fully the connectivity requirement. This is a powerful feature of our work that enables the practitioner to obtain insights, not accessible by other competitors.

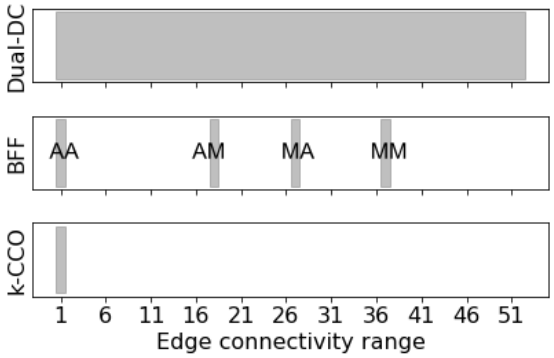


Figure 2: The four variants (AA, AM, MA, and MM) of Best Friends Forever (BFF) [36] result in four different increasing connectivity values, and k-CCO [11] yields a simply connected subgraph on the dual Twitter graph (reply, quote). Only our Dual-DC method allows to *fully control* the connectivity constraint.

• **Application #1: Mining multilayer networks.** We use our algorithmic primitive to mine different layers of the Twitter network. The characteristics of the two subgraphs induced by the optimal set of nodes provide insights into the different types of interactions of users on Twitter.

• **Application #2: Mining human brain networks.** We use our algorithmic primitive on 101 human children brain datasets available from the Brain Imaging Data Exchange (ABIDE) project [10]. Among these datasets, 52 correspond to typically developed (TD) children, and 49 to children with Autism Spectrum Disorder (ASD). We show that our algorithm can be used to extract a clear signal of separation on average between a pair of brain networks corresponding to two typically developed (TD) kids, and a pair of networks each corresponding to a TD child and a child suffering from ASD.

2 Related Work

We briefly review work that lies closest to ours. All graphs considered in this paper are simple, unweighted, and undirected.

Table 1: Comparison of our proposed framework Dual-DC to other prior work. Here, for a graph G and a subset of nodes $S \subseteq V_G$, quantities $d_G(S), \delta_G(S)$ are the average degree of $G[S]$ and the minimum induced degree $\text{min-deg}_G(S) := \min_{v \in S} \deg_{G[S]}(v)$ respectively.

Methods	Density measure (max)	Connectivity constraint on $G[S]$	Hardness
Wu et al. [44]	Avg. degree $d_H(S)$	Connected	NP-hard
Cui et al. [11]	Maximal k-core	Connected	P
BFF-MM [36]	$\min(\delta_G(S), \delta_H(S))$	N/A	P [36]
BFF-AM	$\delta_G(S) + \delta_H(S)$	N/A	NP-hard [9]
BFe-MA	$\min(d_G(S), d_H(S))$	N/A	NP-hard [9]
BFF-AA	$d_G(S) + d_H(S)$	N/A	P [8, 36]
Dual-DC (Ours)	$\delta_H(S)$	k-edge connected	P

Mining dual graphs. Closest to our work lies the work of Wu et al. [44, 45], who pose the question of finding a subset of nodes $S \subseteq V$ such that $G[S]$ forms a connected graph and $H[S]$ maximizes the average degree; they prove that this problem is **NP**-hard and design a heuristic. There is a large amount of research work related to multilayer graphs for other problems such as core decomposition [17] and community detection [25]. Yang et al. [46], Tsourakakis et al. [42], and Lanciano et al. [27] study the problem of finding a subset of nodes that induces a dense subgraph on G and a sparse subgraph on H. Also related to our work from an experimental point of view is the work of Lanciano et al. [27] who use human brain networks to generate easy-to-interpret graph features that can be used to diagnose autism disorders.

Despite the existence of numerous heuristics for mining dual graphs, and more generally multi-layer networks, there are significantly fewer results related to optimizing concrete mathematical objectives simultaneously over two or more graphs on the same set of nodes. Notably, Bhangale et al. recently designed a near-optimal algorithm for the simultaneous Max-Cut problem [4], and proved a hardness result [3] that shows that simultaneous optimization of Max-Cut over more than one graphs is harder than a single graph in terms of approximation.

Semertzidis et al. introduce the Best Friends Forever (BFF) problem [36]; see also Charikar et al. [9] for improved complexity and algorithmic results. Specifically, Semertzidis et al. [36] propose four different formulations for finding a subset of nodes $S \subseteq V$ that induces a dense subgraph across a collection of graphs with vertex set V. They use the minimum and the average degree to measure edge density, and then maximize either the average or the minimum measure of edge density across the collection of graphs. This results in four variants abbreviated as MM, MA, AM, and AA. The second letter indicates the density measure, while the first whether we consider the average (A) or the minimum (M) across the collection. For example, AM aims to maximize the average induced minimum degree over all graphs, whereas AA aims to maximize the average induced average degree over all graphs. It is worth emphasizing that BFF variants impose no connectivity constraint, but according to the classic Mader's theorem we know that every graph of average degree (at least) $4k$ has a k-connected subgraph [28].

k-edge connectivity. A major connectivity notion is *k-edge-connectivity*. An unweighted graph G is said to be k-edge-connected if the removal of any $k-1$ or fewer edges leaves G connected, or equivalently, by Menger's theorem (see [12]), if there are at least k edge-disjoint paths between every pair of distinct vertices.

Dense subgraph discovery. One of the most popular optimization models in DSD is the densest subgraph problem, which asks to find a subset of nodes $S \subseteq V$ that maximizes the average degree of $G[S]$. Unlike most of the other models, this problem is known to be polynomial-time solvable [19]. In addition to its original form, there are a large number of problem variations. The most well-studied variants are the size-restricted ones [2, 5, 15, 24]. For example, in the densest k-subgraph problem [15], given a graph G and a positive integer k, we are asked to find $S \subseteq V$ that maximizes the average degree of $G[S]$ subject to the size constraint $|S| = k$. It is known that such a restriction makes the problem much harder to solve; in fact, the densest k-subgraph problem is **NP**-hard and the best known approximation ratio is $O(|V|^{1/4+\epsilon})$ for any $\epsilon > 0$ [5]. The average degree has recently been generalized in various ways to obtain more sophisticated structures [23,31,39,40,43]. Bonchi et al. [6] very recently proposed a family of algorithms for finding densest k-connected subgraphs on the single network topology.

Partitioning a graph into well-connected components. A line of research closely related to DSD aims to partition a graph into well-connected compo-

nents. For instance, Zhou et al. [47] recently studied the problem to find a family of maximal k-edge-connected subgraphs. To find that, a naive approach is to iteratively compute a minimum cut on each of connected components until every connected component is either k-edge-connected or a singleton. However, such an approach is computationally quite expensive and thus prohibitive for large graphs. To overcome this issue, Zhou et al. [47] devised some techniques and incorporated them into the naive approach. Later, Akiba et al. [1] and Chang et al. [7] developed much more efficient algorithms. The algorithm by Akiba et al. [1] runs in $O(|E| \log |V|)$ time, whereas Chang et al.'s algorithm [7] runs in $O(hl|E|)$ time, where h is the height of the so-called decomposition tree and l is the number of iterations of some subroutine, both of which are instance-dependent parameters typically bounded by a small constant.

3 Proposed Method

We design a polynomial-time exact algorithm for Problem 2. Our algorithm is shown in pseudocode as Algorithm 1, which takes as input the two graphs G, H on the same vertex set V and a parameter k that specifies the k-edge connectivity constraint on G. The algorithm is recursive, and carefully breaks down G in its k-edge connected components. Once the vertex set S induces a k-edge connected component in G, the algorithm removes a node $v \in S$ having the lowest degree in H and returns as its output the best between S and the result of a recursive invocation of the algorithm on $S \setminus \{v\}$. The idea is to maintain the invariant that any solution which is k-connected in G and with higher minimum H-degree than the current best must be entirely contained in one of the subgraphs that we recurse into.

Our main theoretical result concerns the correctness of our algorithm, and is stated as the next theorem.

Algorithm 1: Dual-DC($G(V, E_G), H(V, E_H), k$)

Input: $G = (V, E_G)$, $H = (V, E_H)$, $k \geq 1$ (edge connectivity)

Output: S^* such that the minimum degree in $H[S^*]$ is maximized subject to $G[S^*]$ being k-edge connected; or NO SOLUTION

```
1  if G is not k-edge-connected then
2      Let F ← {C_1, ..., C_r} be the maximal
         k-edge-connected components of G;
3      if r = 0 then
4          return NO SOLUTION
5      else
6          for i = 1 to r do
7              S_i ← Dual-DC(G[C_i], H[C_i], k);
8          t ← argmax{min-deg_H(S_i) | i ∈ [r] ∧ S_i ≠
             NO SOLUTION};
9          return S_t
10 else if |V| > 1 then
11     Let v be the vertex in V of minimum degree
         in H[V] (ties broken arbitrarily);
12     T ← Dual-DC(G[V \ {v}], H[V \ {v}], k);
13     if T = NO SOLUTION or
         min-deg_H(T) ≤ min-deg_H(V) then
14         return V
15     else
16         return T
17 else
18     return V
```

Theorem 3.1. *Algorithm 1 outputs an optimal solution for Problem 2.*

Proof. We argue by induction on the size of the common vertex set V. Let S^* denote the optimal solution for (G, H). If $|V| \leq 1$, then $S^* = V$ (Line 18). So assume $|V| \geq 2$.

Consider first the case where G is not k-connected. Any k-connected subgraph of G is entirely contained in one of the maximal k-connected components $C_1, \ldots, C_r$, so one of them contains S^*, say C_j. By induction, the optimal solution within C_j is computed in Line 7 (when $i = j$), and no other solution computed in Line 7 is feasible and has a higher minimum degree in H than S^* (otherwise S^* would not be optimal). Thus the algorithm returns an optimal solution with the same minimum degree as S^* in Line 9.

If G is k-connected, let $v \in V$ be a vertex with the minimum degree in H. We distinguish the following two cases:

Case (i): V is an optimal solution (i.e., $\text{min-deg}_H(V) = \text{min-deg}_H(S^*)$). Then V is returned in Line 14.

Case (ii): V is not an optimal solution. In this case $\text{min-deg}_H(V) < \text{min-deg}_H(S^*)$ holds. But this implies that no optimal solution includes v, because the degree of v in $H[S^*]$ cannot be larger than its degree in H. Hence Line 12 computes an optimal solution for the pair $(G[V], H[V])$ by the induction hypothesis, which is returned in Line 16.

□

Time complexity. An efficient implementation

of Algorithm 1 runs in time $O(nm \log n)$ in the RAM model, where n is the number of nodes and $m > 1$ is the maximum number of edges in G and H. To see this, observe that in Line 11 of Algorithm 1, after removing the minimum degree vertex of degree d, we can iteratively remove all nodes of degree d in the induced subgraph, i.e., find the set of nodes C in the $(d+1)$-core of H and replace G and H with $G[C]$ and $H[C]$ respectively. This reduces the number of k-connected component computations in the algorithm and doesn't affect its correctness because in Case (ii) above, none of the additional vertices removed can be part of an optimal solution: the minimum degree of an optimal solution must be at least $d+1$, so it must be contained in the $(d+1)$-core of H.

Let d be the minimum degree of the optimal solution in H (or 0 if none exists). We argue inductively that the running time of Algorithm 1 (with the aforementioned faster implementation) is $O(n + (d+1)m \log n) = O(nm \log n)$. Indeed, the total running time spent between Lines 11 and 16, excluding recursive calls, is $O(n+m)$, and the maximum depth of recursive calls is at most d. Each computation of k-edge-connected components in a subgraph with $n' \leq n$ vertices and $m' \leq m$ edges takes time $O(m' \log n)$ by [1]. In Line 2, r subgraphs are found with $m'_1, \ldots, m'_r$ and $\sum_i m'_i \leq m'$, $\sum_i n'_i \leq n'$. Each recursive invocation takes time $O(n'_i + (d+1)m'_i \log n)$ by induction, for a total time of $O(\sum_i (n'_i + (d+1)m'_i \log n)) = O(n + (d+1)m \log n)$, as claimed. We summarize the above analysis with the following theorem statement.

THEOREM 3.2. *Algorithm 1 can be implemented to run in $O(nm \log n)$ time, where $n = |V|$ and $m = \max(|E_G|, |E_H|)$.*

Remarks. Our algorithm naturally extends to the following version of Problem 2, where we are given a graph G and a collection of graphs $\mathcal{H} = \{H_1, \ldots, H_T\}$ and our goal is to find a set of nodes $S \subseteq V$ such that $G[S]$ is k-edge connected, and the minimum degree across the collection $\mathcal{H}$ is maximized. The only difference in the algorithm is that the peeling in Line 11 is done across the set of graphs $\mathcal{H}$, i.e., the node being removed is the node that has the smallest degree across $H_1, \ldots, H_T$. The proof follows our inductive proof and the argument in Proposition 1 [36].

Finally, if instead of using the minimum degree $\delta_H(S)$, we use the average degree $d_H(S)$, the formulation of Problem 1 becomes NP-hard. This is a direct corollary of Wu et al. [44] for the special case of our problem with 1-edge connectivity constraint.

Table 2: Statistics of dual graphs from Twitter multilayer networks and Enron (mail, cc) networks.

# Dual graphs (G, H)	# common nodes	# edges in G	# edges in H
(Reply, Quote)	0.15M	0.46M	0.48M
(Reply, Retweet)	0.23M	0.61M	2.14M
(Retweet, Follow)	0.32M	2.39M	3.49M

4 Experimental results

4.1 Setup

Datasets. For our synthetic experiments, we generate graphs with stochastic block models [21]. The real-world Twitter datasets we use in our experiments are summarized in Table 2. We experiment with Twitter multilayer networks [37] crawled from Twitter traffic generated during the month of February 2018 by Greek-speaking users using a publicly available crawler TWAWLER [33]. Specifically, we use four layers each corresponding to the type of interaction *reply*, *quote*, *retweet*, and *follow*. For each pair of graphs we test, we report the number of common nodes (i.e., Twitter accounts) that appear in both graphs, and the number of edges in each graph.

We also use brain network datasets [27] preprocessed from the public dataset released by the Autism Brain Imagine Data Exchange project. Experiments are done on 101 brain networks of children patients, 52 Typically Developed (TD) and 49 suffering from Autism Spectrum Disorder (ASD). Each network is undirected and unweighted with 116 nodes, summarizing patient's brain activity.

Machine specs. The experiments were performed on a single machine, with Intel i7-10850H CPU @ 2.70GHz and 32GB of main memory.

Implementation and competitors. Our code is available at `https://github.com/tsourakakis-lab/dense-kedge-connected`. We use the code of Akiba et al. [1] to decompose our graph into k-edge connected components. Although we introduce Problem 2 for the first time, two competitors, i.e., BFF[1] and k-CCO[2], are considered in this section.

4.2 Ranging connectivity requirement

An important aspect of our proposed framework is the fact that we can range the connectivity requirement in contrast to other methods that use different formulations

[1] We use the code from `https://github.com/ksemer/BestFriendsForever-BFF-`

[2] The original code was not provided to us by the authors, so we provide our own implementation of k-CCO in Python3.

or heuristics to mine dense subgraphs or communities from multilayer networks. We illustrate the power of our framework by comparing to the elegant Best-Friends-Forever (BFF) formulations [36], as well as the k-CCO model [11]. We generate two random graphs G and H on the same node set according to the stochastic block model. Both graphs contain five blocks $B_1, \ldots, B_5$, where each block has 50 nodes. The internal edge density of each block B_i in graph G, i.e., the probability any two nodes within B_i are connected, is $0.1 \cdot i$, $i = 1, \ldots, 5$; edges across blocks are generated with low probability 2×10^{-4} in order to ensure that the graph is connected, but not well-connected. The edge probability of block B_i in graph H is $0.1 + 0.1 \cdot (5 - i)$, $i = 1, \ldots, 5$; edges across blocks are generated with probability 0.1. Note the densities of blocks in G are increasing from B_1 to B_5, while they are decreasing in H.

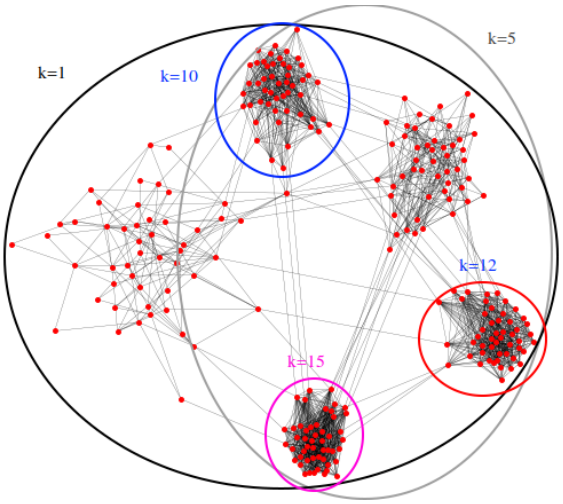


Figure 3: Blocks found by our method for different connectivity requirement k values, visualized on graph G.

The BFF algorithm for the AA formulation (see Section 2 and [36]) returns the whole graph, while the other three formulations (AM, MM, and MA) return the subgraph induced by blocks $B_3 \cup B_4 \cup B_5$. The k-CCO algorithm always returns the whole graph given different k core value constraint on H. However, our method can mine the interesting connectivity structure for different k values on G, and return the cluster has the highest core value on H. Figure 3 visualizes the different blocks on G obtained for different k values, and Figure 4 shows our method has more comprehensive k-edge connectivity control when comparing with benchmarks.

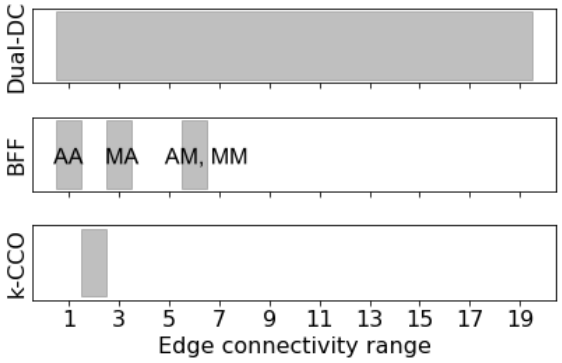


Figure 4: Subgraph connectivity on G ranged by all methods.

4.3 Mining Twitter Table 3 shows our results for some pairs of Twitter graphs, and for various values of the connectivity requirement k. We show the output difference by using the pairs (Reply, Quote), (Reply, Retweet), and (Retweet, Follow) and their reverse ordering. Recall that for a given pair (G, H) we impose the connectivity requirement on G. Table 3 shows the number of nodes in the optimal solution S^*, as well as some basic graph statistics of the subgraph $H[S^*]$. We observe that $|S^*|$ is largest for the *follow* and *retweet* pairs of interactions. If this were not the case, this would have been surprising since the corresponding 2-layered graph for *follow* and *retweet* shares more nodes (0.32M) than the other two pairs of interactions. The average shortest path among all induced subgraphs on H is always less than 2, and as can be seen by comparing $|S^*|$ and the maximum degree, there is typically a hub node connecting almost all pairs via its ego-network. Furthermore, as we can see by the average degree in $H[S^*]$, those induced subgraphs are quasi-clique-like [41]. Figure 5 visualizes the output for our algorithm for the pairs (*follow, retweet*), (*quote,reply*) when $k = 2$.

The problem of finding large-near cliques in a single graph is NP-hard, but in recent years tools that work efficiently on large-scale networks have been proposed (see [26, 30, 40, 41]). It is worth emphasizing an interesting side-effect of our algorithm that appears to hold on real-world dual graphs. Once you are able to find a well-connected subgraph across two networks, it appears in practice that it is a large near-clique. Our findings on the large near-cliques we find on H, agree with the theorems of Konar and Sidiropoulos [26] concerning the existence of large-near cliques in the ego-networks of certain nodes.

4.4 Mining brain networks Finally we apply our algorithm to all possible dual graphs defined by typ-

Table 3: Twitter dual graph results. Statistics are calculated on subgraphs of H.

Graph pair	k	# of nodes	min deg	max deg	avg deg	diameter	# of triangles	avg shortest path
(Reply, Quote)	2	369	84	286	133.2	2	452293	1.64
(Reply, Quote)	6	306	79	238	121.7	2	345408	1.6
(Quote, Reply)	2	368	51	222	81.8	3	148437	1.78
(Quote, Reply)	6	334	50	203	78.8	3	130708	1.76
(Reply, Retweet)	2	470	275	468	362.9	2	8.4M	1.23
(Reply, Retweet)	6	501	256	494	360.8	2	8.6M	1.28
(Retweet, Reply)	2	359	52	217	82.5	3	149782	1.77
(Retweet, Reply)	6	515	51	276	88	3	214812	1.84
(Retweet, Follow)	2	1030	219	966	356.6	2	9.1M	1.65
(Retweet, Follow)	6	931	198	874	324.9	2	6.9M	1.65
(Follow, Retweet)	2	620	285	613	415.1	2	13.5M	1.33
(Follow, Retweet)	6	612	285	606	413	2	13.2M	1.32

ically developed (TD) children and children suffering from Autism Spectrum Disorder (ASD). We report our findings for the 52×51 possible (i.e., ordered) pairs of brain graphs of TD children, and 49×52 possible pairs of children suffering from ASD and TD children.

Figure 6 shows the box plot of the output sizes for $k = 14$. Our results are stable over the choice of k in the range we tried (i.e., k values between 10 and 20). Despite the existence of several outlier pairs of (TD,TD) graphs with respect to their output size (e.g., plenty of (TD,TD) pairs share about 40 nodes as the joint optimal solution), there is still a separation of the averages; for $k = 14$ the respective average value of $|S^*|$ is 95 and 90 for (TD,TD), (ASD,TD) dual graphs respectively. We observe that even if the range of values is similar for the two types of dual graphs, the medians are also separated as shown by the box plot. We conclude that there exists a weak but measurable signal that indicates that healthy individual brains are at least on average well connected across a larger subset of nodes, a finding that agrees with [27] in spirit, and is consistent with other studies in the context of other diseases that argue that "better connected brains, healthier brains"; see, e.g., [32, 38] and references therein. We highlight the importance of ranging connectivity to observe such phenomenon, as it is invisible from the results of k-CCO that always returns the whole graph.

4.5 Scalability analysis Figure 7 shows the running time of the four variants of BFF, our method for $k =$ 5, 20, 35, 50, and k-CCO algorithms on the Twitter (reply, quote) dual graph respectively. We observe that controlling the connectivity requirement comes at the cost of the run time, compared to the competitor methods. Dual-DC runs in at most 25 minutes for $k = 5$. As k grows, the depth of the recursion decreases, and thus we observe that the total run time decreases and becomes comparable to the competitor methods. For instance, for $k = 5$, 50 the depth of the recursion (i.e., the number of calls to Dual-DC) is equal to 294 and 20 respectively. The bar plot illustrates the run time required for the k-connected components computation for this specific dataset. In general, our method handles the graphs from Table 2 in at most thirty minutes on a single machine for any k connectivity value, with the single exception of the largest pair (Retweet, Follow), for which our code requires 5 hours to execute.

5 Conclusions

In this work we introduced a new problem on a dual graph (G, H), that aims to find a set of nodes that induces a well-connected subgraph on G and a dense subgraph on H. We proved that our formulation admits an exact solution, and we propose an algorithm that runs in polynomial time. In practice, our algorithm scales on graphs with several millions of edges on a simple laptop, and runs reasonably fast. Compared to competitor methods, our Dual-DC method enables to control the connectivity constraint on H. Designing a faster algorithm is an interesting open question. We show that our method can be used in practice to mine layers of Twitter and human brain networks.

Our work opens various interesting directions. Optimizing objectives over dual graphs can be harder than a single graph, not only from a formal complexity point of view (e.g., [3]), but even for designing heuristics. Can we design approximation algorithms when we choose the average degree as density measure? Another interesting direction is to choose a different connectivity notion, such as the graph conductance. Can we extend the analysis simultaneously to multiple graphs, except than just two? Finally, and more broadly, how do we design algorithms that mine dual graphs efficiently?

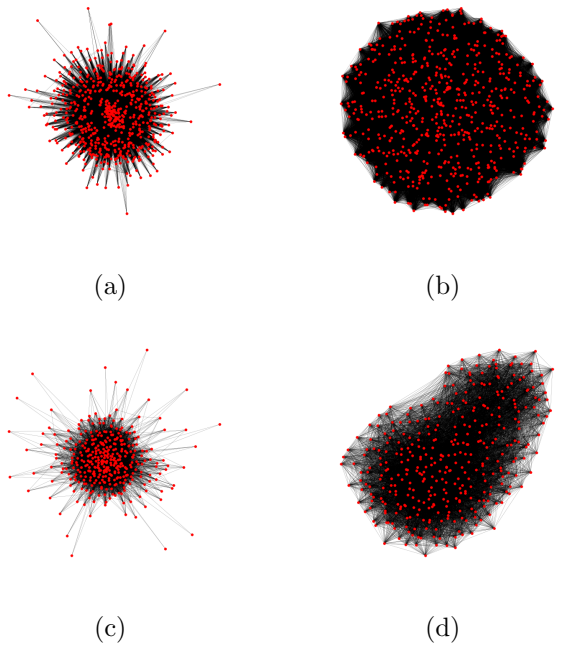


Figure 5: Visualization of our algorithm's output on the Twitter graphs, with connectivity requirement $k = 2$. (a), (b) Follow[S^*], Retweet[S^*], (c),(d) Quote[S^*], Reply[S^*].

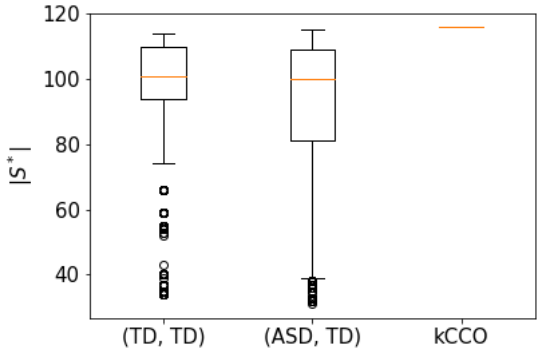


Figure 6: Box plots for the size $|S^*|$, over all possible (TD, TD) and (TD, ASD) dual graphs. The average size of $|S^*|$ is 95 and 90 respectively. The competitor k-CCO always returns the whole graph.

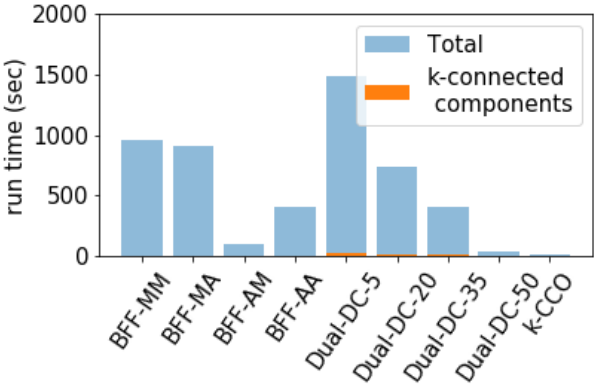


Figure 7: Running time(sec) of Our algorithm with ranged k value, together with four variants of BFF and k-CCO.

References

[1] T. Akiba, Y. Iwata, and Y. Yoshida. Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction. In *Proc. CIKM '13*, pages 909–918, 2013.

[2] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *Proc. WAW '09*, pages 25–37, 2009.

[3] A. Bhangale and S. Khot. Simultaneous max-cut is harder to approximate than max-cut. In *Proc. CCC '20*, pages 9:1–9:15, 2020.

[4] A. Bhangale, S. Khot, S. Kopparty, S. Sachdeva, and D. Thiruvenkatachari. Near-optimal approximation algorithm for simultaneous max-cut. In *Proc. SODA '18*, pages 1407–1425, 2018.

[5] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *Proc. STOC '10*, pages 201–210, 2010.

[6] F. Bonchi, D. García-Soriano, A. Miyauchi, and C. E. Tsourakakis. Finding densest k-connected subgraphs. *Discrete Applied Mathematics*, 305:34–47, 2021.

[7] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang. Efficiently computing k-edge connected components via graph decomposition. In *Proc. SIGMOD '13*, pages 205–216, 2013.

[8] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proc. APPROX '00*, pages 84–95, 2000.

[9] M. Charikar, Y. Naamad, and J. Wu. On finding dense common subgraphs. *arXiv preprint arXiv:1802.06361*, 2018.

[10] C. Craddock, Y. Benhajali, C. Chu, F. Chouinard, A. Evans, A. Jakab, B. S. Khundrakpam, J. D. Lewis, Q. Li, M. Milham, C. Yan, and P. Bellec. The neuro bureau preprocessing initiative: Open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7, 2013.

[11] L. Cui, L. Yue, D. Wen, and L. Qin. K-connected cores computation in large dual networks. *Data Science and Engineering*, 3(4):293–306, 2018.

[12] R. Diestel. *Graph Theory*. Springer, 2005.

[13] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov. Clustering with multi-layer graphs: A spectral

perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831, 2012.

[14] N. Eagle and A. S. Pentland. Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[15] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[16] K. J. Friston. Functional and effective connectivity: A review. *Brain Connectivity*, 1(1), 2011.

[17] E. Galimberti, F. Bonchi, and F. Gullo. Core decomposition and densest subgraph in multilayer networks. In *Proc. CIKM '17*, pages 1807–1816, 2017.

[18] A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: KDD 2015 tutorial. In *Proc. KDD '15*, pages 2313–2314, 2015.

[19] A. V. Goldberg. *Finding a maximum density subgraph.* University of California Berkeley, 1984.

[20] L. B. H. Differential coexpression network analysis for gene expression data. *Methods in Molecular Biology*, 1754:155–165, 2018.

[21] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[22] V. Jethava and N. Beerenwinkel. Finding dense subgraphs in relational graphs. In *Proc. ECML PKDD '15*, pages 641–654, 2015.

[23] Y. Kawase and A. Miyauchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018.

[24] S. Khuller and B. Saha. On finding dense subgraphs. In *Proc. ICALP '09*, pages 597–608, 2009.

[25] J. Kim and J.-G. Lee. Community detection in multilayer graphs: A survey. *ACM SIGMOD Record*, 44(3):37–48, 2015.

[26] A. Konar and N. D. Sidiropoulos. Mining large quasi-cliques with quality guarantees from vertex neighborhoods. In *Proc. KDD '20*, pages 577–587, 2020.

[27] T. Lanciano, F. Bonchi, and A. Gionis. Explainable classification of brain networks via contrast subgraphs. In *Proc. KDD '20*, pages 3308–3318, 2020.

[28] W. Mader. Existenzn-fach zusammenhängender teilgraphen in graphen genügend großer kantendichte. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 37, pages 86–97. Springer, 1972.

[29] M. Magnani and L. Rossi. The ML-model for multilayer social networks. In *Proc. ASONAM '11*, pages 5–12, 2011.

[30] M. Mitzenmacher, J. Pachocki, R. Peng, C. E. Tsourakakis, and S. C. Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proc. KDD '15*, pages 815–824, 2015.

[31] A. Miyauchi and N. Kakimura. Finding a dense subgraph with sparse cut. In *Proc. CIKM '18*, pages 547–556, 2018.

[32] S. Navlakha, A. L. Barth, and Z. Bar-Joseph. Decreasing-rate pruning optimizes the construction of efficient and robust distributed networks. *PLoS Computational Biology*, 11(7):e1004347, 2015.

[33] P. Pratikakis. twawler: A lightweight twitter crawler. *arXiv preprint arXiv:1804.07748*, 2018.

[34] G.-J. Qi, C. C. Aggarwal, and T. Huang. Community detection with edge content in social media networks. In *Proc. ICDE '12*, pages 534–545, 2012.

[35] A. Reinthal, A. Törnqvist, A. Andersson, E. Norlander, P. Stållhammar, and S. Norlin. Finding the densest common subgraph with linear programming. B.S. thesis, Chalmers University of Technology & University of Gothenburg, 2016.

[36] K. Semertzidis, E. Pitoura, E. Terzi, and P. Tsaparas. Finding lasting dense subgraphs. *Data Mining and Knowledge Discovery*, 33(5):1417–1445, 2019.

[37] K. Sotiropoulos, J. W. Byers, P. Pratikakis, and C. E. Tsourakakis. Twittermancer: Predicting interactions on twitter accurately. *arXiv preprint arXiv:1904.11119*, 2019.

[38] K. Supekar, V. Menon, D. Rubin, M. Musen, and M. D. Greicius. Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS Computational Biology*, 4(6):e1000100, 2008.

[39] C. E. Tsourakakis. A novel approach to finding near-cliques: The triangle-densest subgraph problem. *arXiv preprint arXiv:1405.1477*, 2014.

[40] C. E. Tsourakakis. The k-clique densest subgraph problem. In *Proc. WWW '15*, pages 1122–1132, 2015.

[41] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proc. KDD '13*, pages 104–112, 2013.

[42] C. E. Tsourakakis, T. Chen, N. Kakimura, and J. Pachocki. Novel dense subgraph discovery primitives: Risk aversion and exclusion queries. In *Proc. ECML PKDD '19*, pages 378–394. Springer, 2019.

[43] N. Veldt, A. R. Benson, and J. Kleinberg. The generalized mean densest subgraph problem. In *Proc. KDD '21*, page 1604–1614, 2021.

[44] Y. Wu, R. Jin, X. Zhu, and X. Zhang. Finding dense and connected subgraphs in dual networks. In *Proc. ICDE '15*, pages 915–926, 2015.

[45] Y. Wu, X. Zhu, L. Li, W. Fan, R. Jin, and X. Zhang. Mining dual networks: Models, algorithms, and applications. *ACM Transactions on Knowledge Discovery from Data*, 10(4):40:1–40:37, 2016.

[46] Y. Yang, L. Chu, Y. Zhang, Z. Wang, J. Pei, and E. Chen. Mining density contrast subgraphs. In *Proc. ICDE '18*, pages 221–232, 2018.

[47] R. Zhou, C. Liu, J. X. Yu, W. Liang, B. Chen, and J. Li. Finding maximal k-edge-connected subgraphs from a large graph. In *Proc. EDBT '12*, pages 480–491, 2012.