Mining Dual Networks: Models, Algorithms and Applications

YUBAO WU, Electrical Engineering and Computer Science, Case Western Reserve University
XIAOFENG ZHU, Epidemiology and Biostatistics, Case Western Reserve University
LI LI, Family Medicine and Community Health, Case Western Reserve University
WEI FAN, Baidu Research Big Data Lab
RUOMING JIN, Computer Science, Kent State University
XIANG ZHANG, Electrical Engineering and Computer Science, Case Western Reserve University

Finding the densest subgraph in a single graph is a fundamental problem that has been extensively studied. In many emerging applications, there exist *dual* networks. For example, in genetics, it is important to use protein interactions to interpret genetic interactions. In this application, one network represents *physical* interactions among nodes, e.g., protein-protein interactions, and another network represents *conceptual* interactions, e.g., genetic interactions. Edges in the conceptual network are usually derived based on certain correlation measure or statistical test measuring the strength of the interaction. Two nodes with strong conceptual interaction may not have direct physical interaction.

In this paper, we propose the novel dual network model and investigate the problem of finding the densest connected subgraph (DCS) which has the largest density in the conceptual network and is also connected in the physical network. Density in the conceptual network represents the average strength of the measured interacting signals among the set of nodes. Connectivity in the physical network shows how they interact physically. Such pattern cannot be identified using the existing algorithms for a single network. We show that even though finding the densest subgraph in a single network is polynomial time solvable, the DCS problem is NP-hard. We develop a two-step approach to solve the DCS problem. In the first step, we effectively prune the dual networks while guarantee that the optimal solution is contained in the remaining networks. For the second step, we develop two efficient greedy methods based on different search strategies to find the DCS. Different variations of the DCS problem are also studied. We perform extensive experiments on a variety of real and synthetic dual networks to evaluate the effectiveness and efficiency of the developed methods.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data Mining; G.2.2 [**Graph Theory**]: Graph Algorithms; Network Problems; E.1 [**Data Structures**]: Graphs and Networks

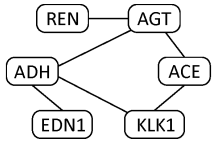
General Terms: Design, Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Dual networks, densest connected subgraphs

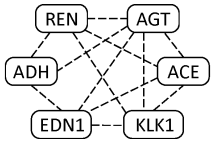
ACM Reference Format:
Yubao Wu, Xiaofeng Zhu, Li Li, Wei Fan, Ruoming Jin, and Xiang Zhang. 2015. Mining dual networks:

This work was partially supported by the National Science Foundation grants IIS-1218036, IIS-1162374, IIS-0953950, the National Basic Research Program of China (No. 2014CB340401), the NIH grant R01 HG003054, the NIH/NIGMS grant R01 GM103309, and the OSC (Ohio Supercomputer Center) grant PGS0218.
Authors' addresses: Y. Wu and X. Zhang, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106; emails: {yubao.wu, xiang.zhang}@case.edu; X. Zhu, Department of Epidemiology and Biostatistics, Case Western Reserve University, Cleveland, OH 44106; email: xiaofeng.zhu@case.edu; L. Li, Department of Family Medicine and Community Health, Case Comprehensive Cancer Center, Case Western Reserve University, Cleveland, OH 44106; email: lxl62@case.edu; W. Fan, Baidu Research Big Data Lab, 1050 Enterprise Way, Sunnyvale, CA 94089; email: wei.fan@gmail.com; R. Jin, Department of Computer Science, Kent State University, Kent, OH 44240; email: jin@cs.kent.edu.

DOI: http://dx.doi.org/10.1145/0000000.0000000



(a) Protein interaction network



(b) Genetic interaction network

Fig. 1. An example of dual biological networks

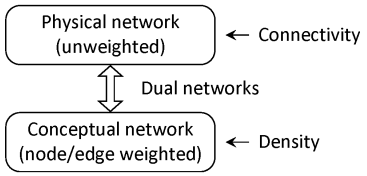


Fig. 2. Finding DCS in dual networks

models, algorithms and applications. *ACM Trans. Knowl. Discov. Data.* 0, 0, Article 00 (2015), 36 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Finding the densest subgraph in a single graph is a key primitive with a wide range of applications, such as modularity detection in biological networks [Saha et al. 2010] and community detection in social networks [Chen and Saad 2012]. Given a graph $G(V, E)$, the goal is to find the subgraph with maximum average edge weight [Lee et al. 2010]. This problem can be solved in polynomial time [Gallo et al. 1989]. For large graphs, approximation algorithms have also been developed [Asahiro et al. 2000; Bahmani et al. 2012; Charikar 2000].

In many real-life applications, we can often observe dual networks representing *physical* and *conceptual* interactions among a set of nodes respectively. For example, in genetics, it is crucial to examine interaction between genetic variants since many diseases are caused by the joint effect of multiple genetic factors [Phillips 2008]. The interacting strength between two genetic variants is usually measured by statistical tests such as likelihood ratio test or analysis of variance [Prabhu and Pe'er 2012]. These statistical interactions are conceptual. Even two genetic variants have strong statistical interaction, their corresponding protein products may not have direct physical interaction. On the other hand, protein-protein interaction network represents physical interactions among proteins and can be used to uncover physical mechanisms behind statistical genetic interactions [Sun and Kardia 2010].

Figure 1 shows an example of dual biological networks, where Figure 1(a) shows the physical protein interaction network among a set of nodes and Figure 1(b) shows the statistical genetic interaction network. The set of nodes have high *density* in the genetic interaction network demonstrating that the statistical interactions among them are strong. Moreover, this set of nodes are *connected* in the protein-protein interaction network, which provides biological interpretation on how they interact with each other through signal transduction among proteins [Ulitsky and Shamir 2007]. Note that we will use solid (dotted) lines to represent the edges in the physical (conceptual) network throughout the paper.

Dual research interest and collaboration networks can be constructed using bibliographic information such as the DBLP dataset [Tang et al. 2008]. Dual networks can also be found in social recommender systems. Please see Section 10 for a detailed description of various real-life dual networks.

In this paper, we study the problem of finding the densest connected subgraph (DCS) in dual networks. Given two graphs $G_a(V, E_a)$ and $G_b(V, E_b)$ representing the physical

and conceptual networks respectively, the DCS consists of a subset of nodes $S \subseteq V$ such that the induced subgraph $G_a[S]$ is connected and the density of $G_b[S]$ is maximized. Density in the conceptual network indicates strong interacting signals among the nodes, and connectivity in the physical network explains the signal transduction process.

Figure 2 summarizes our DCS problem setting. Note that our problem is different from finding co-dense subgraphs [Kelley and Ideker 2005; Pei et al. 2005] or coherent dense subgraphs [Hu et al. 2005; Li et al. 2012a], whose goal is to find the dense subgraphs preserved across multiple networks of the *same* type. In our problem, the physical and conceptual networks complement each other and require different treatments.

We show that the DCS problem is NP-hard and develop a two-step approach to solve the DCS problem. In the first step, we effectively prune the dual networks while guarantee that the optimal solution is contained in the remaining networks. For the second step, we develop two efficient greedy approaches based on different search strategies to find the DCS. The first approach finds the densest subgraph in the conceptual network first and then refines it according to the physical network to make it connected. Although finding the densest subgraph in a single graph can be solved in polynomial time, its actual computational cost is still high and becomes prohibitive for large graphs. We show how to effectively remove nodes in the conceptual network while still retaining the densest subgraph in it. The second approach keeps the target subgraph connected in the physical network while deleting low degree nodes in the conceptual network. We further study two variations of the DCS problem. One problem aims to find the DCS with fixed number of nodes. Another problem requires a set of input seed nodes to be included in the identified subgraph. Both problems are of practical interests. For the DCS problem with input seed nodes, we design an efficient heuristic local search algorithm.

Based on the basic DCS problem, we further study several extensions. First, we observe that the conceptual network has node weights in some applications. Thus we study the DCS problem with both node and edge weights in the conceptual network. All the developed algorithms above can be readily extended to solve the new problem. Second, we formulate a more general problem, where there are multiple types of networks. Third, we provide the MapReduce implementation of the proposed algorithms.

We perform extensive empirical study using real-life biological, social and co-author networks to demonstrate the usefulness of the identified patterns and evaluate the efficiency of the developed algorithms.

Compared to the previous conference version [Wu et al. 2015], several significant improvements have been made in this paper.

(1) In Section 8, we study the densest connected subgraph problem with both node and edge weights in the conceptual network.
(2) We provide a more general problem formulation and the MapReduce implementation of the proposed algorithms in Section 9.
(3) We develop a heuristic local search algorithm to efficiently find the densest connected subgraph with input seed nodes in Section 7.
(4) In experimental studies, Subsection 10.1.2 shows the effectiveness evaluation on a new biological dataset, which demonstrates that the discovered biological signal can be replicated using an independent dataset. Comprehensive results are provided to further evaluate the effectiveness and efficiency of the proposed methods.

2. RELATED WORK

Finding the densest subgraph is an important problem with a wide range of applications [Saha et al. 2010; Chen and Saad 2012] and has attracted intensive research interests. Most of the existing work focuses on a single network, i.e., given a graph[1] $G(V, E)$, find the subgraph with maximum density (average edge weight) [Lee et al. 2010]. This problem can be solved in polynomial time using parametric maximum flow [Gallo et al. 1989]. However, its complexity $O(nm \log(n^2/m))$ is prohibitive for large graphs, where n is the number of nodes and m is the number of edges. For large graphs, efficient approximation algorithms have been developed. A 2-approximation algorithm is proposed in [Asahiro et al. 2000; Charikar 2000]. The basic strategy is deleting the node with minimum degree. This idea can be traced back to [Kortsarz and Peleg 1994], which shows that the density of the maximum core of a graph is at least half of the density of the densest subgraph. Recently, an improved $2(1 + \epsilon)$ approximation greedy node deletion algorithm has been proposed [Bahmani et al. 2012]. The algorithm takes $O(\log_{1+\epsilon} n)$ iterations. In each iteration, it deletes a set of nodes with degree smaller than $2(1 + \epsilon)$ times the density of the remaining subgraph.

Variations of the densest subgraph problem have also been studied. The densest k subgraph problem aims to find the densest subgraph with exactly k nodes, which has been shown to be NP-hard [Bhaskara et al. 2010]. The problem of finding the densest subgraph with seed nodes requires that a set of input nodes must be included in the resulting subgraph, which can be solved in polynomial time [Saha et al. 2010].

In biomedical domain, the densest subgraph has been used to analyze the gene annotation graph [Saha et al. 2010]. The idea can be generalized to analyze multiple networks. For example, in [Hu et al. 2005; Li et al. 2012a], the authors aim to find coherent dense subgraphs whose edges are not only densely connected but also frequently occur in multiple gene co-expression networks. Finding co-dense subgraphs that exist in multiple gene co-expression or protein interaction networks are studied in [Kelley and Ideker 2005; Pei et al. 2005]. The underlying assumption of these works is that the set of networks under study are of the same type.

The network-based methods have shown to be promising in integrating different datasets in systems biology. In [Ideker et al. 2002], the authors use the gene expression data to weight the nodes in the protein interaction networks. Their goal is to find the maximum score connected subgraph. The maximum score connected subgraph requires that the subgraph is connected and also maximizes certain objective function. This approach has also been applied to integrate genome-wide association study datasets and protein interaction networks [Baranzini et al. 2009; Jia et al. 2011]. Since the problem of finding the maximum score connected subgraph is NP-hard, heuristics are developed to find approximate solutions [Baranzini et al. 2009; Jia et al. 2011]. All these methods aim to find dense subgraphs from a single network.

3. THE DCS PROBLEM

We adopt the classic graph density definition [Asahiro et al. 2000; Bahmani et al. 2012; Charikar 2000; Gallo et al. 1989] to formulate the DCS problem. Table I lists the main symbols and their definitions.

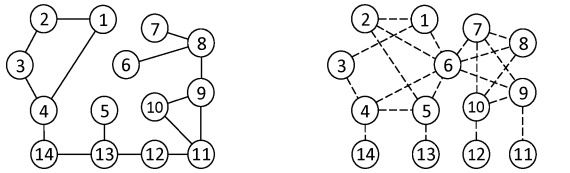
Definition 3.1. Given a graph $G(V, E)$ and $S \subseteq V$, density $\rho(S)$ is defined as

$$\rho(S) = \frac{e(S)}{|S|},$$

[1] In this paper, we use network and graph interchangeably.

Table I. Main Symbols

Symbols	Definitions
$G(V, E)$	graph G with node set V and edge set E
$G(V, E_a, E_b)$	dual networks $G_a(V, E_a)$ and $G_b(V, E_b)$
$n; m_a; m_b$	number of nodes; number of edges in G_a; number of edges in G_b
S	node set $S \subseteq V$
$G[S]$	subgraph induced by S in graph G
$G_a[S], G_b[S]$	subgraph induced by S in graph G_a, G_b
$\|S\|$	number of nodes in S
$e(u, v)$	weight of edge (u, v)
$e(S)$	sum of the weights of edges in subgraph $G[S]$
$e_b(S)$	sum of the weights of edges in subgraph $G_b[S]$
$e(S, T)$	sum of edge weights, $e(S,T) = \sum_{u \in S, v \in T} e(u, v)$
$r(u)$	weight of node u
$r(S)$	sum of node weights, $r(S) = \sum_{u \in S} r(u)$
$\rho(S)$	density of subgraph $G[S]$
$N(u)$	the set of neighbor nodes of node u in graph G
$\delta(S)$	boundary of S, $\delta(S) = \{u \in S \mid \exists v \in N(u) \cap (V \setminus S)\}$
$w_G(u)$	degree of node u in graph G
$w'_G(u)$	sum of node degree and node weight, $w'_G(u) = w_G(u) + r(u)$



(a) Physical network G_a (b) Conceptual network G_b

Fig. 3. An example of dual networks

where $e(S)$ is the sum of the weights of edges in subgraph $G[S]$, and $|S|$ is the number of nodes in $G[S]$.

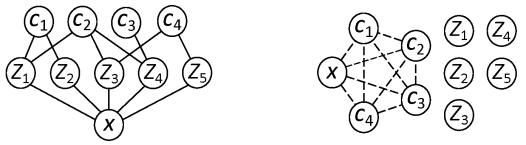
Let $G_a(V, E_a)$ be an unweighted graph representing the physical network and $G_b(V, E_b)$ be an edge weighted graph representing the conceptual network. We denote the subgraphs induced by node set $S \subseteq V$ in the physical and conceptual networks as $G_a[S]$ and $G_b[S]$ respectively. For brevity, we also use $G(V, E_a, E_b)$ to represent the dual networks. Let $e_b(S)$ denote the sum of the weights of edges in subgraph $G_b[S]$.

Definition 3.2. Given dual networks $G(V, E_a, E_b)$, the densest connected subgraph (DCS) consists of a set of nodes $S \subseteq V$ such that $G_a[S]$ is connected and the density of $G_b[S]$ is maximized.

An example is shown in Figure 3. In this example, the DCS consists of nodes $S = \{6, 7, 8, 9, 10\}$. Its induced subgraph $G_a[S]$ is connected in the physical network and $G_b[S]$ has the largest density in the conceptual network. Note that the dense component consisting of nodes $\{1, 2, 3, 4, 5, 6\}$ in G_b is not connected in G_a.

THEOREM 3.3. *Finding the DCS in dual networks is NP-hard.*

PROOF. We show that the DCS problem can be reduced from the set cover problem [Karp 1972]. Let $Z = \{Z_1, \cdots, Z_l\}$ be a family of sets with $C = \{c_1, \cdots, c_h\} = \bigcup_{i=1}^{l} Z_i$



(a) Physical network G_a (b) Conceptual network G_b

Fig. 4. Dual networks construction from an instance of the set cover problem

being the elements. The set cover problem aims to find a minimum subset $Z_{\text{opt}} \subseteq Z$, such that each element c_j is contained in at least one set in Z_{opt}.

The dual networks can be constructed as follows. Let the node set $V = \{x, c_1, \cdots, c_h, Z_1, \cdots, Z_l\}$. In the physical network G_a, node x is connected to every node $Z_i \in Z$, and every node $c_j \in C$ is connected to node Z_i if $c_j \in Z_i$ in the set cover problem. The conceptual network G_b is constructed by creating a unit edge weight clique among nodes $\{x, c_1, \cdots, c_h\}$ and leaving nodes $\{Z_1, \cdots, Z_l\}$ isolated.

Figure 4 gives an example of the dual networks constructed from an instance of the set cover problem with $Z_1 = \{c_1, c_2\}, Z_2 = \{c_1\}, Z_3 = \{c_2, c_4\}, Z_4 = \{c_2, c_3\}, Z_5 = \{c_4\}$.

Let $Z_{\text{opt}} \subseteq Z$ be the optimal solution to the set cover problem and $|Z_{\text{opt}}| = l^* \leq l$. Denote $X = \{x, c_1, \cdots, c_h\}$. The subgraph induced from $S = X \cup Z_{\text{opt}}$ is connected in G_a, and has density $\frac{h(h+1)/2}{h+l^*+1}$ in G_b. Let S' denote any node set, where $G_a[S']$ is connected. Next, we prove that the density of $G_b[S]$ is no less than that of $G_b[S']$.

First, we consider the case when S' contains all nodes in X. S' must contain a set of nodes $Z' \subseteq Z$ to be connected in G_a. Thus $S' = X \cup Z'$, $|S'| = h + 1 + |Z'|$, and $e_b(S') = h(h+1)/2$. Since Z_{opt} has the minimum number of sets (nodes) among all subsets of Z that cover all elements in C, the density of $G_b[S]$ is no less than that of $G_b[S']$.

Second, we consider the case when S' contains a subset of nodes $X' \subset X$. S' must contain a set of nodes $Z' \subseteq Z$ to be connected in G_a. Thus $S' = X' \cup Z'$. Let $|X'| = h'$ and $|Z'| = l' \geq 1$. The density of $G_b[S']$ is $\frac{h'(h'-1)/2}{h'+l'}$. Next, we show that adding nodes in $X \setminus X'$ to S' will only increase its density.

If $x \notin S'$, after adding x to S', the resulting subgraph has density $\frac{h'(h'-1)/2+h'}{h'+l'+1} > \frac{h'(h'-1)/2}{h'+l'}$ in G_b, and is also connected in G_a since x is connected to every $Z_i \in Z$. To add a node $c_j \in X \setminus X'$ to S' and make it still connected, we need to add at most one node Z_i, where $c_j \in Z_i$. The density of the resulting subgraph is at least $\frac{h'(h'-1)/2+h'}{h'+l'+2} > \frac{h'(h'-1)/2}{h'+l'}$. We can repeat this process by adding remaining nodes to S' until it contains all the nodes in X. During this process, the density of the resulting subgraph will keep increasing. In the first case, we already prove that the density of $G_b[S]$ is no less than that of $G_b[S']$ when $X \subset S'$. This completes the proof for the second case.

Therefore, the subgraph induced from $S = X \cup Z_{\text{opt}}$ is the DCS, and it gives an optimal solution to the set cover problem.

Let's continue the example in Figure 4. The subgraph induced from $S = \{x, c_1, c_2, c_3, c_4, Z_1, Z_3, Z_4\}$ is the DCS which is connected in G_a and has maximum density 1.25 in G_b. $Z_{\text{opt}} = \{Z_1, Z_3, Z_4\}$ is an optimal solution to the set cover problem. □

The DCS with size constraint (DCS_k) and input seed nodes (DCS_*seed*) can be defined as follows.

Definition 3.4. Given dual networks $G(V, E_a, E_b)$ and an integer k, the DCS_k consists of a set of nodes $S \subseteq V$ such that $|S| = k$, $G_a[S]$ is connected and the density of $G_b[S]$ is maximized.

Definition 3.5. Given dual networks $G(V, E_a, E_b)$ and an input query node set $Q \subseteq V$, the DCS_*seed* consists of a set of nodes $S \subseteq V$ such that $Q \subseteq S$, $G_a[S]$ is connected and the density of $G_b[S]$ is maximized.

The DCS_k and DCS_*seed* problems are also NP-hard. The proofs are omitted.

4. OPTIMALITY PRESERVING PRUNING

In this section, we introduce a pruning step, which removes the *low degree leaf nodes* from the dual networks and still guarantees that the optimal DCS is contained in the resulting networks.

Definition 4.1. Given dual networks $G(V, E_a, E_b)$, suppose that its DCS consists of a set of nodes S. Let $\rho(S)$ represent its density in G_b, i.e., $\rho(S) = \rho(G_b[S])$. A node $u \in V$ is a low degree leaf node if (1) u is a leaf node in G_a, i.e., $w_{G_a}(u) = 1$, and (2) its degree in G_b is less than $\rho(S)$, i.e., $w_{G_b}(u) < \rho(S)$.

LEMMA 4.2. *The DCS in dual networks does not contain any low degree leaf node.*

PROOF. Suppose otherwise. We remove u from S and let S' be the remaining set of nodes. Since $G_a[S]$ is connected and u is a leaf node in G_a, so after deleting u, $G_a[S']$ is still connected. However, its density $\rho(S') = \frac{e_b(S')}{|S'|} = \frac{e_b(S) - w_{G_b[S]}(u)}{|S|-1} > \frac{e_b(S)}{|S|} = \rho(S)$, since $w_{G_b[S]}(u) \leq w_{G_b}(u) < \rho(S) = \frac{e_b(S)}{|S|}$. This contradicts the assumption. □

Even though the density of DCS ($\rho(S)$) is unknown beforehand, we can still effectively prune many low degree leaf nodes as follows. Let $G_0 = G$ be the original dual networks. We remove all low degree leaf nodes (using density $\rho(G_b[V])$) in the physical network G_0^a and conceptual network G_0^b respectively. That is, we remove all the nodes that have degree one in G_a and have degree less than $\rho(G_b[V])$ in G_b from the dual networks. Let the resulting dual networks be $G_1(V_1, E_a(V_1), E_b(V_1))$, where $E_a(V_1)$ and $E_b(V_1)$ represents the edge sets induced by V_1 in network G_a and G_b respectively. We then continue to remove the low degree leaf nodes using density $\rho(V_1)$ in G_1. That is, we remove all the nodes that have degree one in $G_a[V_1]$ and have degree less than $\rho(G_b[V_1])$ in G_b from the dual networks. We repeat this process until no such nodes left.

Let $\{G_0, G_1, \cdots, G_l\}$ represent the sequence of dual networks generated by this process and $\{v_j^i\}$ represent the set of nodes deleted in iteration i ($0 \leq i \leq l$). The following theorem shows that the DCS is retained in this process.

THEOREM 4.3. *Iteratively removing low degree leaf nodes will not delete any node in the DCS.*

PROOF. Consider two adjacent dual networks G_i and G_{i+1} in the sequence $\{G_0, G_1, \cdots, G_l\}$. From G_i to G_{i+1}, we delete a set of nodes $\{v_j^i\}$. For a node $u \in \{v_j^i\}$, it is a leaf node in G_i^a, and its degree in G_i^b is $w_{G_i^b}(u) < \rho(G_i^b)$. Let S_i be the node set of the DCS in G_i. We have that $\rho(G_i^b) \leq \rho(S_i)$. Thus $w_{G_i^b}(u) < \rho(S_i)$. Therefore, node u is a low degree leaf node with respect to the DCS in G_i. From the proof of Lemma 4.2, node u must not exist in S_i. By induction, we have that the DCS of the original dual networks is retained in the low degree leaf nodes removing process. □

Using this pruning strategy, we can safely remove the nodes that are not in the DCS, thus reduce the overall search space. Experimental results on real graphs show that 40% to 60% of the nodes can be pruned using this method.

Complexity: Let m_a and m_b be the numbers of edges in the graphs G_a and G_b respectively. At each iteration, the algorithm will delete a set of nodes and their adjacent edges in both G_a and G_b. For each deleted edge, the algorithm needs to update the

node degree if the other endpoint still exists. Each update takes $O(1)$ time. Thus, the running time of the algorithm is $O(m_a + m_b)$.

Next, we introduce two greedy algorithms, DCS_RDS (Refining the Densest Subgraph) and DCS_GND (Greedy Node Deletion), to find the DCS from the size reduced dual networks.

5. THE DCS_RDS ALGORITHM

The DCS_RDS algorithm first finds the densest subgraph in G_b, which usually is disconnected in G_a. It then refines the subgraph by connecting its disconnected components in G_a. Although the densest subgraph can be identified in polynomial time by the parametric maximum flow method [Gallo et al. 1989], its actual complexity $O(nm \log(n^2/m))$ is prohibitive for large graphs (n and m are the number of nodes and edges in the graph respectively). Next, we first introduce an effective procedure that can dramatically reduce the cost of finding the densest subgraph in a single graph.

5.1. Fast Densest Subgraph Finding in Conceptual Network

To find the densest subgraph in a single network, greedy node deletion algorithms [Asahiro et al. 2000; Charikar 2000] and peeling algorithms [Alvarez-Hamelin et al. 2005; Bahmani et al. 2012; Batagelj and Zaversnik 2003] keep deleting the nodes with low degree. However, these methods do not guarantee that the densest subgraph is contained in the identified subgraph.

We introduce an approach which effectively removes nodes in G_b and still guarantees to retain the densest subgraph. Our node removal procedure is based on the following key observation.

LEMMA 5.1. *Let $\rho(T)$ be the density of the densest subgraph $G[T]$. Any node $u \in T$ has degree $w_{G[T]}(u) \geq \rho(T)$.*

PROOF. Suppose there exists a node $u \in T$ with $w_{G[T]}(u) < \rho(T)$. Then the subgraph $G[T'] = G[T \setminus \{u\}]$ has density $\rho(T') = \frac{e(T)-w_{G[T]}(u)}{|T|-1} > \frac{e(T)}{|T|} = \rho(T)$. Thus we find a subgraph $G[T']$ whose density is larger than that of $G[T]$. This contradicts the assumption that $G[T]$ is the densest subgraph. □

The lemma says that the degree of any node in the densest subgraph $G[T]$ must be no less than its density $\rho(T)$. Since the density of $G[T]$ is also equivalent to half of the average degree in $G[T]$, i.e., $\rho(T) = \overline{w}_{G[T]}/2$, this is equivalent to say that any node should have degree more than $\overline{w}_{G[T]}/2$. Note that this is a necessary condition for characterizing the densest subgraph. It is also related to the concept of d-core.

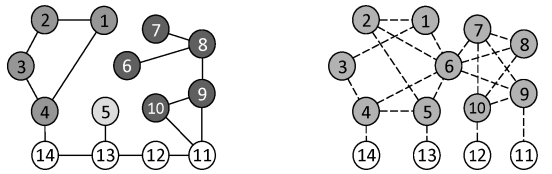
Definition 5.2. The d-core D of G is the maximal subgraph of G such that for any node u in D, $w_D(u) \geq d$.

Note that the d-core of a graph is unique and may consist of multiple connected components. It is easy to see that any subgraph in which every node's degree is no less than d is part of the d-core.

THEOREM 5.3. *The densest subgraph $G[T]$ of G is a subgraph of the d-core D of G ($G[T] \subseteq D$) when $d \leq \rho(T)$.*

PROOF. From Lemma 5.1, any node $u \in T$ has degree $w_{G[T]}(u) \geq \rho(T)$. Since $d \leq \rho(T)$, any node $u \in T$ has degree $w_{G[T]}(u) \geq d$. Thus $G[T]$ is a subgraph of the d-core D. □

LEMMA 5.4. *Let $\alpha = \rho(T)/d$. The d-core subgraph D is a 2α-approximation of the densest subgraph T.*



(a) Physical network G_a (b) Conceptual network G_b

Fig. 5. Refining the densest subgraph

PROOF. Let $D.V$ represent the node set in D. Since the density of the d-core is $\rho(D) = \frac{e(D.V)}{|D.V|} = \frac{\sum_{u \in D.V} w_D(u)}{2|D.V|} \geq \frac{\sum_{u \in D.V} d}{2|D.V|} = \frac{d}{2} = \frac{\rho(T)}{2\alpha}$, we have that $\rho(T) \leq 2\alpha\rho(D)$. □

From Theorem 5.3 and Lemma 5.4, if we can find a density value d ($d \leq \rho(T)$), then we have both: 1) $G[T] \subseteq D$, and 2) D is a $2\rho(T)/d$ approximation of the densest subgraph $G[T]$. Therefore, if we use the density of the 2-approximation subgraph generated by the greedy node deletion algorithm [Asahiro et al. 2000; Charikar 2000] for d-core, we obtain a 4-approximation ratio ($2\alpha \leq 2(2d)/d = 4$) of the densest subgraph $G[T]$. Note that d-core can be generated by iteratively removing all nodes with degree less than d until every node in the remaining graph has degree no less than d [Bahmani et al. 2012].

To sum up, we use the following three-step procedure to find the exact densest subgraph from G: (1) Find a 2-approximation of the densest subgraph in G, where the density of the discovered subgraph $d \geq \rho(T)/2$; (2) Find the d-core D of G; (3) Compute the exact densest subgraph from D.

Empirical results show that after applying this approach, the remaining subgraph can be orders of magnitude smaller than the original graphs. It can significantly speed up the process of finding the exact densest subgraph. Moreover, we have shown that the density of the remaining subgraph is a 4-approximation of the density of the densest subgraph.

Complexity: Let n and m_b be the number of nodes and edges in the original graph G_b, and n' and m'_b be the number of nodes and edges in the d-core D. The first and second steps run in $O(m_b + n \log n)$ and $O(m_b)$ respectively. To find the exact densest subgraph from D, the parametric maximum flow algorithm runs in $O(n'm'_b \log(n'^2/m'_b))$. Note that n' (m'_b) can be orders of magnitude smaller than n (m_b).

5.2. Refining Subgraph in Physical Network

Suppose that the densest subgraph of G_b consists of node set T and is denoted as $G_b[T]$. The induced subgraph in the physical network $G_a[T]$ is typically disconnected. Given dual networks $G(V, E_a, E_b)$ and the densest subgraph $G_b[T]$, we use $\{G_a[V_1], G_a[V_2], \cdots, G_a[V_\kappa]\}$ to represent all connected components in $G_a[T]$, where $T = V_1 \cup V_2 \cup \cdots \cup V_\kappa$.

Example 5.5. In Figure 5, the densest subgraph in the conceptual network consists of nodes $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Its corresponding connected components in the physical network are $V_1 = \{6, 7, 8, 9, 10\}$, $V_2 = \{1, 2, 3, 4\}$, and $V_3 = \{5\}$.

In the next, we discuss how to refine the subgraph $G_a[T]$ to make it connected in the physical network G_a while still preserving its high density in G_b. Specifically, we consider the following dense subgraph refinement problem.

Definition 5.6. Given dual networks $G(V, E_a, E_b)$ and the densest subgraph $G_b[T]$ of G_b, the problem of refining the densest subgraph aims to find a nonempty subset

Algorithm 1: Refining the densest subgraph

Input: $G(V, E_a, E_b)$, nodes T (densest subgraph in G_b)
Output: node set $\hat{S}$ of DCS

1: Find all κ connected components $\{G_a[V_i]\}$ in $G_a[T]$;
2: Sort $G_a[V_i]$ by the density $\rho(G_b[V_i])$ in descending order;
3: Weigh the node u in G_a by $(w_{G_b}(u))^{-1}$;
4: $S_1 \leftarrow V_1$;
5: **for** $i \leftarrow 1$ to $\kappa - 1$ **do**
6: Compute the shortest path $H_i(S_i, V_{i+1})$ in G_a;
7: $S_{i+1} \leftarrow S_i \cup V_{i+1} \cup H_i$;
8: $j \leftarrow \text{argmax}_i\, \rho(G_b[S_i])$; **return** S_j;

of $\{G_a[V_1], G_a[V_2], \cdots, G_a[V_\kappa]\}$ with node set Y and a node set $X \subseteq V \setminus T$, such that $G_a[Y \cup X]$ is connected and the density of $G_b[Y \cup X]$ is maximized.

The problem of refining the densest subgraph is also NP-hard, which can be proved using similar reduction method as in the proof of Theorem 3.3.

We introduce a greedy heuristic procedure to refine the densest subgraph as outlined in Algorithm 1. The algorithm puts the node set T to G_a, and finds all the connected components $\{G_a[V_i]\}$. It then sorts $\{G_a[V_i]\}$ by their density $\rho(G_b[V_i])$ in descending order. It weights the nodes in G_a by the reciprocal of its degree in G_b. The intuition is that we want to select nodes that have high degree in G_b to connect $\{G_a[V_i]\}$. The algorithm merges the connected components in G_a iteratively. In each iteration, it merges two components by adding the nodes on the node weighted shortest path connecting two components. The density of the newly merged component is calculated after each iteration. The component with the largest density is returned as the DCS.

Example 5.7. Continue the example in Figure 5. The densities of the connected components in the physical network are $\rho(V_1 = \{6, 7, 8, 9, 10\}) = 1.6$, $\rho(V_2 = \{1, 2, 3, 4\}) = 0.75$, and $\rho(V_3 = \{5\}) = 0$. Initially, the subgraph induced by $S_1 = V_1$ has density $\rho(S_1) = 1.6$. Algorithm 1 first connects S_1 and V_2 through the shortest path $H_1 = \{11, 12, 13, 14\}$. The subgraph induced by $S_2 = S_1 \cup V_2 \cup H_1$ has density $\rho(S_2) = 1.31$. After merging V_3, the subgraph induced by S_3 has density $\rho(S_3) = 1.5$. Therefore, the subgraph induced by S_1 has the largest density in G_b and is returned as the DCS.

The approximation ratio of the DCS_RDS algorithm can be estimated as $\alpha = \rho(T)/\rho(\hat{S})$, where $\rho(T)$ is the density of the densest subgraph in the conceptual network. Experimental results show that the approximation ratio is usually around 1.5∼2 using real networks.

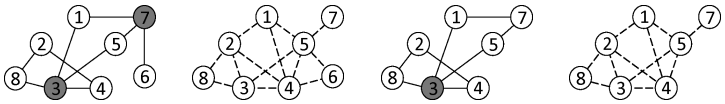
Complexity: Algorithm 1 runs in $O(m_a + n \log n)$ as we can easily modify Dijkstra's algorithm to find the shortest path in node weighted graph by transforming each node as an edge.

6. THE DCS_GND ALGORITHM

The basic DCS_GND algorithm keeps deleting nodes with low degree in the conceptual network, while avoiding disconnecting the physical network.

Definition 6.1. A node is an articulation node if removing this node and the edges incident to it disconnects the graph.

Articulation nodes can be identified in linear time by the depth first search [Tarjan 1972]. The basic DCS_GND algorithm deletes one node in each iteration. The deleted



(a) $G_a[V_0]$ (b) $G_b[V_0]$ (c) $G_a[V_1]$ (d) $G_b[V_1]$

Fig. 6. Greedy node deletion example

node has the minimum degree in the conceptual network among all the non-articulation nodes in the physical network. Since in each iteration, only one non-articulation node is deleted, the remaining physical network will keep connected. Note that as long as the graph is not empty, there always exists a non-articulation node in the graph. Thus, the DCS_GND algorithm can always find a non-articulation node to delete until the graph becomes empty. Density of the subgraphs generated in this process is recorded and the subgraph with the largest density is returned as the identified DCS.

Example 6.2. Suppose that the input physical and conceptual networks are as shown in Figures 6(a) and 6(b) respectively. Nodes $\{3, 7\}$ in gray color are articulation nodes and the remaining ones are non-articulation nodes. Node 6, which has the minimum degree 2 among all the non-articulation nodes, will be deleted. The resulting dual networks are shown in Figures 6(c) and 6(d), where node 3 is the only articulation node.

To further improve the efficiency, we can delete a set of low degree non-articulation nodes in each iteration. However, not all non-articulation nodes can be deleted simultaneously, since deleting one non-articulation node may make another non-articulation node to become an articulation node. Thus we need to find the subset of non-articulation nodes that can be deleted together.

Definition 6.3. A set of non-articulation nodes are independent if the deletion of them does not disconnect the graph.

LEMMA 6.4. *Let $\{B_i\}$ represent the set of biconnected components of graph G such that each B_i has at least one non-articulation node. If we select one non-articulation node from each B_i, the set of selected nodes are independent non-articulation nodes.*

PROOF. Suppose that we delete one non-articulation node $v_i \in B_i$. The deletion of node v_i does not disconnect B_i since it is biconnected. Since two distinct biconnected components share at most one articulation node, deleting node v_i does not disconnect any other biconnected components. So if we delete one non-articulation node from each component in $\{B_i\}$, every component is still connected. Therefore, the remaining subgraph is still connected. □

Algorithm 2 illustrates the algorithm based on deleting independent non-articulation nodes iteratively. Parameter γ is used to control the degree of the non-articulation nodes to be deleted. γ is usually set between $0 \sim 2$. Since $2\rho(G_b)$ is the average node degree, there are about half of the nodes whose degree is smaller than the threshold $2\rho(G_b)$. More nodes are deleted in each iteration when larger γ value is used. Please refer to experimental evaluation for further discussion on the effect of γ. If all low degree nodes are articulations nodes, the algorithm picks the non-articulation node with the minimum degree to delete.

Example 6.5. Let's continue the example in Figure 6. Suppose that $\gamma = 1.5$. We have that $\gamma \cdot \rho(G_b[V_0]) = 2.44$. The fast DCS_GND method will delete nodes $\{6, 8\}$ simultaneously, since they have degree $2 < 2.44$ and are independent non-articulation nodes.

Algorithm 2: Fast DCS_GND algorithm

Input: $G(V, E_a, E_b)$, parameter $\gamma > 0$
Output: node set $\hat{S}$ of DCS

1: $V_0 \leftarrow V$; $i \leftarrow 0$;
2: **while** $|V_i| > 0$ **do**
3: Compute the articulation nodes A in $G_a[V_i]$; $A \leftarrow V_i \setminus A$;
4: Select a set of nodes $L \subseteq A$ such that the nodes in L are independent non-articulation nodes and have low degrees, i.e., for any $u \in L$, $w_{G_b[V_i]}(u) \leq \gamma \cdot \rho(G_b[V_i])$;
5: **if** $|L| = 0$ **then** $L \leftarrow \{u | u = \text{argmin}_{v \in A}\, w_{G_b[V_i]}(v)\}$;
6: $V_{i+1} \leftarrow V_i \setminus L$; $i \leftarrow i + 1$;
7: $j \leftarrow \text{argmax}_i\, \rho(G_b[V_i])$; **return** V_j;

Complexity: It takes $O(nm_a)$ time to find the non-articulation nodes and biconnected components in line 3 by depth first search. It takes $O(m_b)$ time to find the low degree nodes in line 4. It takes $O(m_b + n \log n)$ time to select the node with minimum degree in line 5. Thus, the DCS_GND algorithm runs in $O(nm_a + m_b + n \log n)$.

We can estimate the approximation ratio of DCS_GND as follows. When deleting a node v, we assign its incident edges in G_b to it. Let $edg(v)$ denote the sum of the edge weights, and $edg_{\max}$ represent the maximum $edg(v)$ among all nodes (deleted in order by the algorithm). Let S be the node set of the optimal DCS, T be the node set of the densest subgraph in the conceptual network G_b. We have the following inequality.

LEMMA 6.6. $\rho(S) \leq \rho(T) \leq edg_{\max}$.

PROOF. It is easy to see that $\rho(S) \leq \rho(T)$. Next, we show that $\rho(T) \leq edg_{\max}$. Each edge in $G_b[T]$ must be assigned to a node in T in the node deletion process. Thus we have that $e_b(T) \leq \sum_{u \in T} edg(u) \leq \sum_{u \in T} edg_{\max} = |T| \cdot edg_{\max}$. This means that $\rho(T) = \frac{e_b(T)}{|T|} \leq edg_{\max}$. □

LEMMA 6.7. *Let $\hat{S}$ be the node set of the DCS identified by the DCS_GND algorithm. The approximation ratio of the algorithm is $\alpha = edg_{\max}/\rho(\hat{S}) \geq \rho(S)/\rho(\hat{S})$.*

Based on Lemma 6.7, we can estimate the approximation ratio α from the results returned by the algorithm. Empirical study shows that α is usually around 2 in real networks.

The DCS_GND algorithm can be easily extended to solve the DCS_k and DCS_*seed* problems. For the DCS_k problem, we can keep deleting low degree non-articulation nodes until there are k nodes left. For the DCS_*seed* problem, we avoid deleting the seed nodes during the process. The approximation ratio analysis discussed above also applies to these variants.

7. THE DCS_MAS ALGORITHM FOR THE DCS_SEED PROBLEM

In this section, we introduce a heuristic algorithm, DCS_MAS, for the DCS_*seed* problem. DCS_MAS uses a local search procedure, which iteratively include one more node with the maximum adjacency value to the visited nodes.

The algorithm is outlined in Algorithm 3. Given the query nodes, we first compute the Steiner tree in the physical network by the Mehlhorn's algorithm [Mehlhorn 1988]. Thus the query nodes become connected.

Next, the algorithm begins a local search process. The Steiner tree is used as the initial subgraph. In each iteration (lines 4-5), the algorithm adds the node $u \in \delta_a(V \setminus V_i)$ with the maximum adjacency value $e_b(\{v\}, V_i)$ to V_i. Here, $\delta_a(S)$ denotes the boundary

Algorithm 3: Maximum adjacency search (DCS_MAS) algorithm for the DCS_*seed* problem

Input: $G(V, E_a, E_b)$, query nodes Q, parameter K
Output: node set $\hat{S}$ of DCS_*seed*

1: Compute the Steiner tree of Q in G_a, and let S be its node set;
2: $V_0 \leftarrow S$; $i \leftarrow 0$;
3: **while** $|V_i| < K$ **do**
4: $\quad u \leftarrow \text{argmax}_{v \in \delta_a(V \setminus V_i)} e_b(\{v\}, V_i)$;
5: $\quad V_{i+1} \leftarrow V_i \cup \{u\}$; $i \leftarrow i + 1$;
6: $j \leftarrow \text{argmax}_i \rho(G_b[V_i])$; **return** V_j;

of the node set S in the physical network. The intuition is that the node u is adjacent to V_i in the physical network, and has the maximum adjacency value to V_i in the conceptual network. The subgraph during the local search process is always connected in the physical network. The subgraph with the maximum density in the conceptual network during the local search process is returned. Parameter K is used to control the search space. When the number of nodes in V_i is equal to K, the algorithm will terminate.

The local search process needs at most K iterations. Let $\overline{N}_a$ be the average number of neighbors in the physical network and $\overline{N}_b$ be the average number of neighbors in the conceptual network. Then, in the ith iteration, it takes $O(|V_i| \cdot \overline{N}_a)$ to find the node with the maximum adjacency value in line 4, and it takes $O(\overline{N}_b)$ to add this node and update the adjacency values of its neighborhood nodes in the conceptual network. Thus the local search process runs in $O(\sum_i(|V_i| \cdot \overline{N}_a + \overline{N}_b)) = O(K^2\overline{N}_a + K\overline{N}_b)$. Mehlhorn's algorithm runs in $O(m_a + n \log n)$ [Mehlhorn 1988].

8. FINDING DCS WITH BOTH NODE AND EDGE WEIGHTS IN THE CONCEPTUAL NETWORK

In the conceptual network, in addition to edge weights, we can often have node weights as well. For example, in the genetic interaction network, in addition to measuring genetic interactions, i.e., the edge weights, we can also apply statistical tests to measure node weights [Jia and Zhao 2014]. A node weight represents the strength of the association between a single genetic factor and the disease trait. In this section, we study the DCS problem when there are both edge and node weights in the conceptual network. To integrate both node and edge weights, we revise the classic density definition as follows.

Definition 8.1. Given a graph $G(V, E)$ and a set of nodes $S \subseteq V$, the node and edge weighted density $\rho(S)$ is defined as

$$\rho(S) = \frac{e(S) + r(S)}{|S|},$$

where $e(S)$ is the sum of the weights of edges in the subgraph $G[S]$, $r(S) = \sum_{u \in S} r(u)$ is the sum of the node weights, and $r(u)$ is the weight of node u. For brevity, the node and edge weighted density is also referred to as density when there is no ambiguity.

If we need to tune the composition ratio of the node and edge weights, we can reweight the nodes and edges in a pre-processing step. Let $\eta (0 \leq \eta \leq 1)$ be a constant. For each node u, its weight is changed to $(1-\eta) \cdot r(u)$. For each edge (u, v), its weight is changed to $\eta \cdot e(u, v)$. Then, the node and edge weighted density of subgraph $G[S]$ is changed to $(\eta \cdot e(S) + (1-\eta) \cdot r(S))/|S|$. Thus, we can tune the composition ratio of the two parts through the node and edge reweighting step.

The node and edge weighted density is adopted in the work [Goldberg 1984]. When all nodes have unit weights, we have that $r(S) = |S|$ and $\rho(S) = \frac{e(S)}{|S|} + 1$, which degrades to the classic density as in Definition 3.1. The DCS problem with both node and edge weights is also NP-hard.

Next, we show how to extend the techniques developed before to solve the DCS problem following Definition 8.1.

8.1. Optimality Preserving Pruning

In this section, we extend the optimality preserving pruning step. We need to re-define the *low degree leaf nodes*. We still use $w_G(u)$ to denote the degree of node u in graph G. Let $w'_G(u)$ denote the sum of node degree and node weight of node u, i.e., $w'_G(u) = w_G(u) + r(u)$.

Definition 8.2. Given dual networks $G(V, E_a, E_b)$, suppose that its DCS consists of a set of nodes S. Let $\rho(S)$ represent the node and edge weighted density in G_b, i.e., $\rho(S) = \rho(G_b[S])$. A node $u \in V$ is a low degree leaf node if (1) u is a leaf node in G_a, i.e., $w_{G_a}(u) = 1$, and (2) u satisfies that $w'_{G_b}(u) < \rho(S)$.

LEMMA 8.3. *The DCS in dual networks does not contain any low degree leaf node.*

PROOF. Suppose otherwise. We remove u from S and let S' be the remaining set of nodes. Since $G_a[S]$ is connected and u is a leaf node in G_a, after deleting u, $G_a[S']$ is still connected. However, its density $\rho(S') = \frac{e_b(S')+r(S')}{|S'|} = \frac{e_b(S)+r(S)-w'_{G_b[S]}(u)}{|S|-1} > \frac{e_b(S)+r(S)}{|S|} = \rho(S)$, since $w'_{G_b[S]}(u) \leq w'_{G_b}(u) < \rho(S) = \frac{e_b(S)+r(S)}{|S|}$. This contradicts the assumption that $G[S]$ is the optimal solution to the DCS problem. □

Therefore, when we iteratively remove the low degree leaf nodes from the dual networks, the DCS is retained in the resulting networks.

THEOREM 8.4. *Iteratively removing low degree leaf nodes will not delete any node in the DCS.*

PROOF. The proof is similar to that of Theorem 4.3. □

The optimality preserving pruning procedure still runs in $O(m_a + m_b)$.

8.2. The DCS_RDS Algorithm

8.2.1. The Greedy Node Deletion Algorithm. In this section, we extend the greedy node deletion algorithm to find a 2-approximation for the densest subgraph problem with Definition 8.1.

Algorithm 4 shows the greedy node deletion algorithm. It keeps deleting the node with the minimum $w'_{G[V_i]}(u)$ value. After deleting one node, we compute the density of the remaining subgraph. Then the subgraph with the maximum density during the node deletion process is returned. This algorithm still finds a 2-approximation solution.

LEMMA 8.5. *For any node u in the densest subgraph $G[T]$, $w'_{G[T]}(u) \geq \rho(T)$.*

PROOF. Since $G[T]$ is the densest subgraph, $\rho(T) \geq \rho(T \setminus \{u\})$. Therefore, we have that $\frac{e(T)+r(T)}{|T|} \geq \frac{e(T)+r(T)-w'_{G[T]}(u)}{|T|-1}$. Then we can prove this lemma. □

LEMMA 8.6. *Algorithm 4 obtains a 2-approximation solution to the densest subgraph problem with Definition 8.1.*

Algorithm 4: Greedy node deletion algorithm for the densest subgraph problem with the node and edge weighted density

Input: $G(V, E)$
Output: node set $\hat{S}$

1: $V_0 \leftarrow V$; $i \leftarrow 0$;
2: **while** $|V_i| > 0$ **do**
3: $\quad u \leftarrow \text{argmin}_{v \in V_i} w'_{G[V_i]}(v)$;
4: $\quad V_{i+1} \leftarrow V_i \setminus \{u\}$; $i \leftarrow i + 1$;
5: $j \leftarrow \text{argmax}_i \rho(G[V_i])$; **return** V_j;

PROOF. Let $S = V_i$. That is, S represents the remaining nodes in the ith iteration. We have that $\sum_{u \in S} w'_{G[S]}(u) = 2|S| \cdot \rho(S) - r(S) \leq 2|S| \cdot \rho(S)$, which means that the average $w'_{G[S]}(u)$ value is no greater than $2\rho(S)$. If a node u has the minimum $w'_{G[S]}(u)$ value, we have that $w'_{G[S]}(u) \leq 2\rho(S)$.

Now, consider the first time in the iteration when a node u from the densest subgraph $G[T]$ is deleted. Clearly, $S \supseteq T$. Thus we have that $\rho(T) \leq w'_{G[T]}(u) \leq w'_{G[S]}(u) \leq 2\rho(S)$, where we use Lemma 8.5 in the first inequality. This implies that $\rho(S) \geq \rho(T)/2$ and hence the algorithm gives a 2-approximation solution. □

Algorithm 4 runs in $O(m_b + n \log n)$ time when using Fibonacci heaps [Cormen et al. 2001].

8.2.2. Removing Low Degree Nodes. In this section, we show that the densest subgraph is contained in the node and edge weighted d-core. Thus, we can prune the search space by first finding the node and edge weighted d-core.

Definition 8.7. The node and edge weighted d-core D of G is the maximal subgraph of G such that for any node u in D, $w'_D(u) \geq d$.

THEOREM 8.8. *The densest subgraph $G[T]$ of G is a subgraph of the node and edge weighted d-core D of G ($G[T] \subseteq D$) when $d \leq \rho(T)$.*

PROOF. The proof is similar to that of Theorem 5.3. □

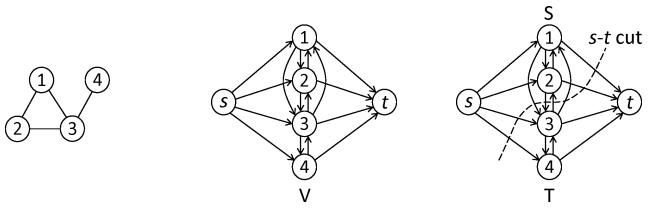
To compute the node and edge weighted d-core, we can iteratively remove the nodes with low $w'_G(u)$ values, i.e., $w'_G(u) < d$. This procedure still runs in $O(m_b)$.

8.2.3. The Parametric Maximum Flow Method. In this section, we show how to apply the parametric maximum flow method to exactly solve the densest subgraph problem with the node and edge weighted density. Given an undirected graph G, the flow network can be constructed as follows.

(1) Replace each edge (u, v) of G by two oppositely directed edges $\langle u, v\rangle$ and $\langle v, u\rangle$ of capacity $e(u, v)$;
(2) Add a source node s and a sink node t;
(3) Create directed edge $\langle s, u\rangle$ of capacity $w_G(u) + 2r(u)$ for each node u;
(4) Create directed edge $\langle u, t\rangle$ of capacity 2λ for each node u;

Figure 7 shows one example of constructing the flow network. Figure 7(a) shows the original undirected graph, and Figure 7(b) shows the flow network. We change the undirected edge into two directed edges, add the source node s and the edges from s to each node, and add the sink node t and the edges from each node to t.

By computing the parametric maximum flow on the flow network, we can solve the densest subgraph problem exactly. Suppose that any s-t cut partitions the node set V



(a) Original network (b) Flow network (c) s-t cut

Fig. 7. Constructing the flow network from the original network

into two parts S and T, where S is on the s side and T is on the t side. For example, in Figure 7(c), the s-t cut is indicated by the dotted curve, and we have that $S = \{1, 2\}$ and $T = \{3, 4\}$. The capacity of an s-t cut is

$$\begin{aligned} &\textstyle\sum_{u\in T}(w_G(u) + 2r(u)) + e(S, T) + 2\lambda|S| \\ &= 2e(V) + 2r(V) - 2(e(S) + r(S) - \lambda|S|). \end{aligned}$$

Therefore, minimizing the capacity of an s-t cut is equivalent to maximizing the quantity $e(S) + r(S) - \lambda|S|$. Thus we can maximize the ratio $\frac{e(S)+r(S)}{|S|}$ by searching for the largest λ value [Gallo et al. 1989].

8.2.4. Refining Subgraph in Physical Network. We can simply change the line 3 of Algorithm 1 to the following line.

3: Weigh the node u in G_a by $(w'_{G_b}(u))^{-1}$;

The analysis of the approximation ratio and complexity is the same as that of the original algorithm.

8.3. The DCS_GND Algorithm

To extend the DCS_GND algorithm to handle the node and edge weighted conceptual network, we can change the lines 4 and 5 in Algorithm 2 to the following two lines.

4: Select a set of nodes $L \subseteq A$ such that the nodes in L are independent non-articulation nodes and have low $w'_{G_b[V_i]}(u)$ values, i.e., for any $u \in L$, $w'_{G_b[V_i]}(u) \leq \gamma \cdot \rho(G_b[V_i])$;

5: **If** $|L| = 0$ **then** $L \leftarrow \{u \,|\, u = \text{argmin}_{v\in A}\, w'_{G_b[V_i]}(v)\}$;

Next, we derive the approximation ratio. We still use S to denote the node set of the optimal DCS, T to denote the node set of the densest subgraph in the conceptual network G_b, and $edg(u)$ to denote the sum of the weights of edges, which are deleted together with node u during the node deletion process. We have the following lemmas.

LEMMA 8.9. $\rho(S) \leq \rho(T) \leq \max_{u\in V}(edg(u) + r(u))$.

PROOF. It is easy to see that $\rho(S) \leq \rho(T)$. Next, we show that $\rho(T) \leq \max_{u\in V}(edg(u) + r(u))$. Each edge in $G_b[T]$ must be assigned to a node in T in the node deletion process. Thus we have that $e_b(T) \leq \sum_{u\in T} edg(u)$. Therefore, $e_b(T)+r(T) \leq \sum_{u\in T} edg(u)+\sum_{u\in T} r(u) \leq |T|\cdot\max_{u\in V}(edg(u)+r(u))$. This means that $\rho(T) = \frac{e_b(T)+r(T)}{|T|} \leq \max_{u\in V}(edg(u) + r(u))$. □

LEMMA 8.10. *Let $\hat{S}$ be the node set of the DCS identified by the DCS_GND algorithm. The approximation ratio of the algorithm is $\alpha = \max_{u\in V}(edg(u)+r(u))/\rho(\hat{S}) \geq \rho(S)/\rho(\hat{S})$.*

The complexity of the modified DCS_GND algorithm is the same as that of the original algorithm.

Table II. Complexities of the Methods

Methods (with/without node weights)		Complexities
Optimality preserving pruning		$O(m_a + m_b)$
DCS_RDS	Greedy node deletion	$O(m_b + n \log n)$
	Removing low degree nodes	$O(m_b)$
	Parametric maximum flow	$O(n' m'_b \log(n'^2/m'_b))$
	Refining densest subgraph	$O(m_a + n \log n)$
Basic/fast DCS_GND		$O(nm_a + m_b + n \log n)$
DCS_MAS	Maximum adjacency search	$O(K^2 \overline{N}_a + K\overline{N}_b)$
	Mehlhorn's algorithm	$O(m_a + n \log n)$

8.4. The DCS_MAS Algorithm for the DCS_*seed* Problem

We can simply change line 4 in Algorithm 3 to the following line.

4: $u \leftarrow \text{argmax}_{v \in \delta_a(V \setminus V_i)}(e_b(\{v\}, V_i) + r(v))$;

The complexity is the same as that of the original algorithm.

Table II lists the complexities of the methods. The complexity of each method for the networks with node weights is the same as that of the corresponding method for the networks without node weights.

9. FURTHER EXTENSIONS

In this section, we discuss two extensions to the DCS problem: one from the theoretical perspective, and one from the implementation perspective.

9.1. A More General Problem Formulation

The basic DCS problem can be treated as a special case of a more general problem formulation, where the edges in the network may have different labels [Kivelä et al. 2013]. Specifically, we can define the graph with multiple edge labels as a triple $G(V, E, L)$, where V is a set of nodes, L is a set of labels, and E is a set of labeled edges. One triple $(u, v, l) \in E$ with $u, v \in V$ and $l \in L$ represents one edge (u, v) with edge label l. An edge label induced subgraph, $G_l(V, E_l)$, consists of the edges E_l of a particular label $l \in L$. Different optimization objective functions or constraints (such as edge density, k-edge or k-vertex connectivity, diameter of the subgraph, etc.) can be defined on different edge label induced subgraphs.

In the DCS problem, there are two edge labels, i.e., conceptual and physical edges. On the conceptual edge induced subgraph, the objective function is to maximize the density among a subset of nodes. On the physical edge induced subgraph, the constraint is the connectivity among the nodes. In the future work, we will explore the generalized subgraph discovery problem and their applications in the real-world.

9.2. MapReduce Implementation

Recently, there have been a lot of interests using MapReduce for processing large graphs [Bahmani et al. 2012]. In the following, we show that the DCS_RDS and DCS_GND methods can be easily implemented on top of the MapReduce framework. The DCS_MAS method is a local search heuristic, thus we do not discuss its implementation on MapReduce. In the discussion, we assume that the classic density in Definition 3.1 is used in the DCS problem. All the implementations can also be readily extended to solve the DCS problem with the node and edge weighted density in Definition 8.1.

9.2.1. DCS_RDS on MapReduce. The DCS_RDS method has four steps: computing the 2-approximation densest subgraph; finding d-core; computing the exact densest subgraph from the d-core; refining the densest subgraph in the physical network.

In the first step, we can directly apply the MapReduce algorithm developed in [Bahmani et al. 2012] to compute a $2(1+\epsilon)$-approximation densest subgraph, where ϵ is a small constant. Note that this algorithm gives a relaxed approximation ratio. The density of the discovered subgraph will be used as the threshold for finding d-core in the second step. The densest subgraph is still guaranteed to be contained in the d-core, since we are using a slightly smaller d value. In the second step, we can apply the same strategy when removing low degree nodes to find the d-core. In the third step, we can use the MapReduce implementation of the Ford-Fulkson method [Halim et al. 2011] as a subroutine of the parametric maximum flow method [Gallo et al. 1989] to compute the densest subgraph. In the fourth step, we can apply the MapReduce implementation of single-source shortest path algorithm [Lin and Dyer 2010] to connect the disconnected components in the physical network.

9.2.2. DCS_GND on MapReduce. The DCS_GND method has four steps: computing the articulation nodes of G_a; computing the density $\rho(G_b)$; selecting the low degree independent non-articulation nodes; removing those nodes.

To compute the density $\rho(G_b)$ and remove nodes, we can still apply the method in [Bahmani et al. 2012]. To compute the articulation nodes, we can use the algorithm in [Ausiello et al. 2012]. Let $\pi(u)$ denote whether node u is an articulation node. Let $\theta(u)$ denote the identifier of the biconnected component which contains the node u. Note that $\theta(u)$ may contain multiple identifiers if node u is an articulation node. The output of this step has the form $\langle u; \pi(u), \theta(u)\rangle$.

To select the low degree independent non-articulation nodes, we need two passes on MapReduce. In the first pass, we duplicate each edge (u, v) and its weight $e(u, v)$ to two $\langle key; value\rangle$ pairs $\langle u; e(u, v)\rangle$ and $\langle v; e(v, u)\rangle$ in the mapping step. We also add $\langle u; \pi(u), \theta(u)\rangle$ for each node in the mapping step. The input to the reduce task is of the form $\langle u; \pi(u), \theta(u), e(u, v_1), e(u, v_2), \ldots, e(u, v_k)\rangle$ where $v_1, v_2, \ldots, v_k$ are the neighbors of u. The reducer will do nothing if node u is an articulation node. Otherwise, the reducer will sum up the associated edge weights for each key, and output $\langle \theta(u); u, w_{G_b}(u)\rangle$ if $w_{G_b}(u) \leq \gamma \cdot \rho(G_b)$. In the second pass, we just emit the key-value pairs in the form of $\langle \theta(u); u, w_{G_b}(u)\rangle$ in the mapping step. The reducer will pick one node v with the minimum degree from each $\theta(u)$, and output $\langle v; \$\rangle$, which denotes that the node v will be deleted.

10. EXPERIMENTAL RESULTS

In this section, we perform comprehensive experiments to evaluate the effectiveness and efficiency of the proposed methods using a variety of real and synthetic datasets. All the programs are written in C++. All experiments are performed on a server with 32G memory, Intel Xeon 3.2GHz CPU, and Redhat OS.

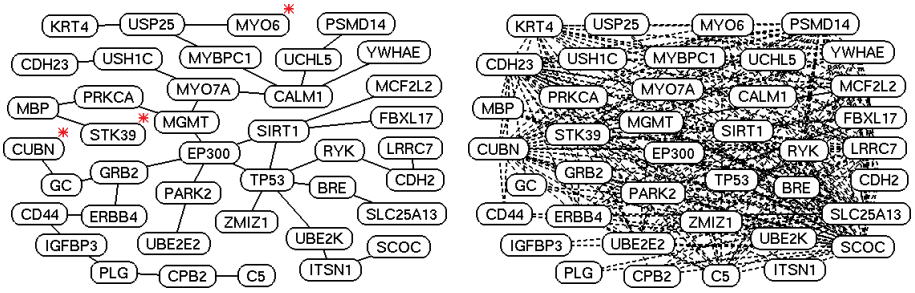
10.1. Effectiveness Evaluation in the Biological Application Domain

We first evaluate the effectiveness of the DCS method in the biological application domain. The dual biological networks include the physical protein interaction network and the conceptual genetic interaction network. The protein interaction network is downloaded from the BioGRID database (*http://thebiogrid.org/*). After filtering out duplicate interactions, the network contains 8,468 proteins and 25,715 unique physical bonding interactions.

The first genetic interaction network is generated by performing chi-square test on genetic marker pairs in the Wellcome Trust Case Control Consortium (WTCCC) hy-

Table III. Statistics of the Dual Biological Networks

Dual networks	Abbr.	#nodes	#edges in G_a	#edges in G_b
WTCCC	WT	$8,468$	$25,715$	$67,744$
ARIC	AR	$8,468$	$25,715$	$81,810$



(a) Subgraph in protein interaction network (b) Subgraph in genetic interaction network

Fig. 8. The DCS_k ($k = 40$) identified from the WTCCC dataset

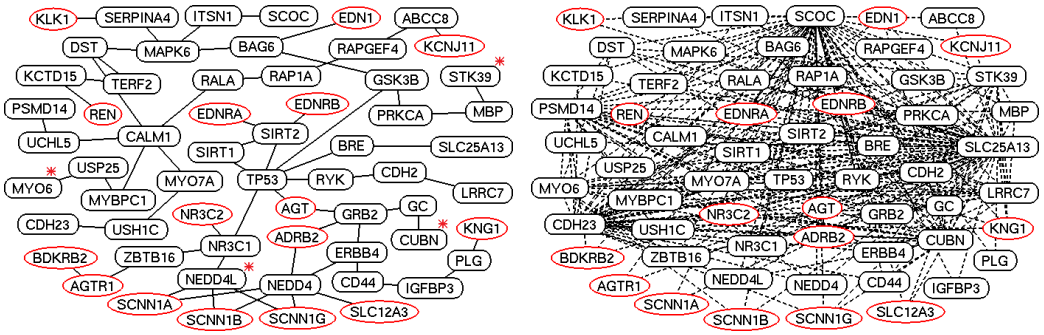
pertension dataset [Burton et al. 2007]. The WTCCC dataset includes $4,890$ European adults. The most significant interactions between genes are used to weight the edges in the genetic interaction network, which has $67,744$ edges. Note that we use half of the samples in the WTCCC dataset to construct the dual networks. Another half is used for significance evaluation of the identified DCS.

The second genetic interaction network is generated in a similar way from the atherosclerosis risk in communities (ARIC) study dataset downloaded from dbGaP [Levy et al. 2009; the ARIC Investigators 1989]. The ARIC dataset includes $15,792$ African American and European American adults. We focus on the $9,319$ European American adults. We study hypertension and calculate genetic interaction using the chi-square test. The resulting genetic interaction network has $81,810$ edges. Note that the WTCCC and ARIC datasets are independent. We use the WTCCC dataset to evaluate the significance of the DCS identified from the ARIC dataset. Table III shows the basic statistics of the dual biological networks constructed from the WTCCC and ARIC datasets.

10.1.1. The DCSs Identified from the WTCCC Dataset. The DCS identified in the dual biological networks has 211 nodes. The figures are omitted because of the large size. The set of nodes are sparsely connected in the protein interaction network, while the subgraph in the genetic interaction network has high density. Specifically, the DCS has 282 edges in the protein interaction network and $4,258$ edges in the genetic interaction network.

Note that the densest subgraph of the genetic interaction network is not connected in the protein interaction network. There are 73 nodes in the densest subgraph of the genetic interaction network. Only 2 of them are connected in the protein interaction network. This demonstrates that dual networks can help to uncover pattern that cannot be identified in individual networks. Such pattern cannot be identified by finding dense subgraphs preserved in both networks either. There are 68 overlapping nodes between the densest subgraph in the genetic interaction network and the DCS identified by DCS_RDS.

Figure 8 shows the identified DCS_k with $k = 40$. From the figure, it is clear that the identified subgraph is connected in the protein interaction network and highly dense in the genetic interaction network. Several genes in this subgraph have been reported to be associated with hypertension. For example, MYO6 encodes an actin-based molecular motor involved in intracellular vesicle and organelle transport, and has been



(a) Subgraph in protein interaction network (b) Subgraph in genetic interaction network

Fig. 9. The DCS_*seed* identified from the WTCCC dataset (renin pathway genes are in red ellipses)

shown to have association with hypertension [Slavin et al. 2011]. The CUBN gene is associated with albuminuria, which is an important factor for cardiovascular disease [McMahon et al. 2013]. The STK39 gene has been reported many times as a hypertension susceptibility gene [Wang et al. 2009]. This gene encodes a serine/threonine kinase that is thought to function in the cellular stress response pathway. These genes are highlighted by stars in the figure. Other genes in the identified subgraph are potential hypertension candidate genes or important for signal transduction in hypertension related pathways.

To identify the DCS_*seed*, we use a set of 16 genes in renin pathways known to be associated with hypertension as the input seed nodes [Yue et al. 2006]. Renin pathway, also called renin-angiotensin system, is a hormone system that regulates blood pressure. The resulting subgraphs are shown in Figure 9. The input seed genes are in red ellipses and the remaining nodes represent the newly added genes. As can be seen from the figure, the seed nodes are originally not directly connected in the protein interaction network. The newly added genes tend to have large degree in the genetic interaction network. In addition to the genes discussed above, we can see the NEDD4L gene is connected to multiple seed genes. It has been reported that NEDD4L is involved in the regulation of plasma volume and blood pressure by controlling cell surface expression of the kidney epithelial Na^+ channel [Luo et al. 2009].

To evaluate the statistical significance of the discovered DCSs, we apply 4 widely used pathway evaluation methods : the GenGen method, the gene set ridge regression (GRASS) method, the Plink set-based test method, and the hybrid set-based test (HYST) method [Wang et al. 2010; Li et al. 2012b]. Given a set of genes, these methods evaluate the significance of the association between the set of genes and the disease phenotype. These methods adopt the null hypothesis that none of the genes in a gene set harbor genetic markers associated with the disease risk. The alternative hypothesis is that at least one gene harbors genetic markers associated with the disease risk. Different methods adopt different strategies to perform the tests. The GenGen method assigns the best test statistic among genetic markers in or near a gene to represent the gene level signal, then calculates the Kolmogorov-Smirnov-like enrichment score for a pathway [Wang et al. 2007]. The GRASS method first uses the regularized regression to select representative genetic markers for each gene, then assesses their joint association with the disease risk [Chen et al. 2010]. The Plink set-based test method selects the independent and significant genetic markers in the pathway, and then calculates the average of the test statistics as the pathway enrichment score [Purcell et al. 2007]. The HYST method combines the extended Simes' test and the scaled chi-square test to

Table IV. P-values of the DCSs Identified from the WTCCC Dataset (without node weights, tested on half of the WTCCC dataset)

Methods	GenGen	GRASS	Plink	HYST
DCS (1)	2.4×10^{-6}	1.0×10^{-6}	2.3×10^{-6}	1.1×10^{-9}
DCS (2)	1.6×10^{-5}	2.8×10^{-5}	4.6×10^{-5}	5.6×10^{-7}
DCS (3)	4.8×10^{-5}	7.4×10^{-5}	9.5×10^{-5}	8.2×10^{-7}
DCS_k	5.6×10^{-5}	1.3×10^{-6}	4.6×10^{-6}	3.7×10^{-8}
DCS_*seed*	8.5×10^{-5}	4.9×10^{-6}	1.5×10^{-5}	2.4×10^{-6}
DS	0.36	0.47	0.33	0.17
MSCS	0.15	0.13	0.21	0.12

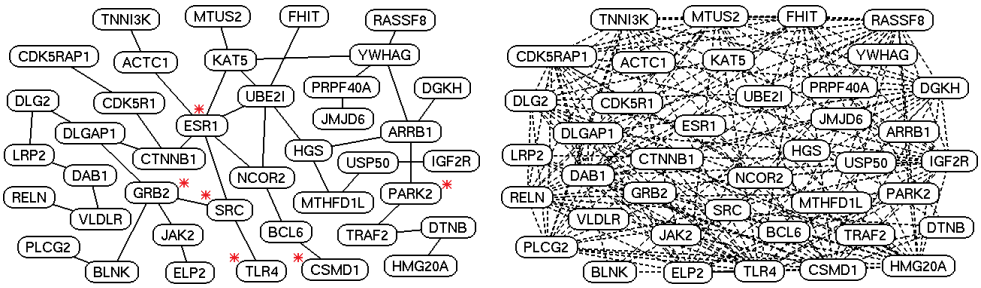
assess the overall significance of the association in a set of genetic markers [Li et al. 2012b]. Note that the test dataset consists of the samples that are not used for constructing the dual networks to ensure the independence between pattern discovering and significance evaluation.

Table IV shows the p-value of the identified DCSs. DCS (1), (2) and (3) represent the top-3 DCSs. The top DCSs are identified iteratively: after the top-1 DCS is identified, we remove its nodes and edges from the dual networks; the DCS_GND algorithm is then applied to each connected component to find the next DCS in the remaining graph. As can be seen from the table, the DCSs are highly significant. In the table, we also show the results of two other methods for finding pathways in biological networks. One method finds the densest subgraph (DS) in the protein interaction network. Another method aims to find the maximum-score connected subgraph (MSCS) in the protein interaction network [Ideker et al. 2002]. The DS method uses the most significant genetic interactions between genes to weight the edges in the protein interaction network. The MSCS method uses the most significant chi-square test statistics to weight the nodes in the protein interaction network. As we can see, the subgraphs identified by these two methods are not as significant as the DCSs. This indicates the importance of integrating the complementary information encoded in the physical protein interaction network and the conceptual genetic interaction network.

10.1.2. The DCSs Identified from the ARIC Dataset. Using the genetic interaction network generated from the ARIC dataset, the identified DCS has 184 nodes. The figures are omitted because of the large size. The set of nodes are sparsely connected in the protein interaction network, while the subgraph in the genetic interaction network has high density. Specifically, the DCS has 246 edges in the protein interaction network and $4,135$ edges in the genetic interaction network.

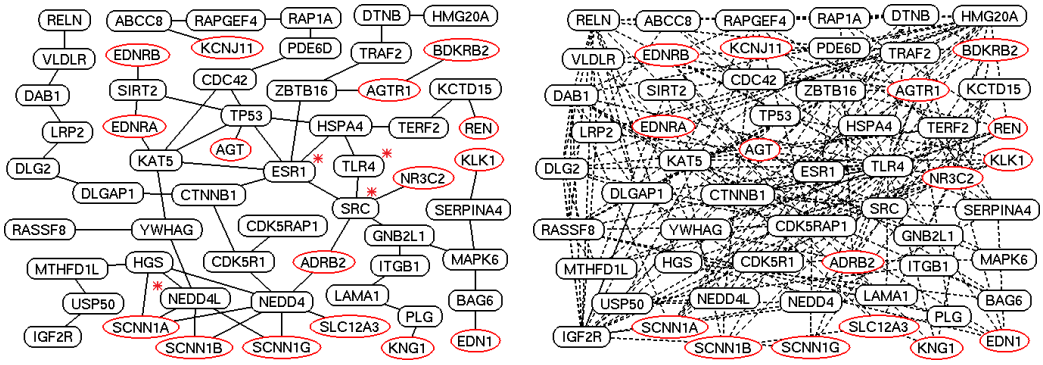
Similar to the results from the WTCCC dataset, the densest subgraph of the genetic interaction network is not connected in the protein interaction network. The densest subgraph of the genetic interaction network consists of 89 nodes, however, the induced subgraph in the protein interaction network only contains 6 edges.

Figure 10 shows the identified DCS_k with $k = 40$. Several genes in the identified DCS have been reported to be associated with hypertension. The CSMD1 gene encodes a transmembrane protein and is a potential tumor suppressor. It has been shown to have association with hypertension [Hong et al. 2009]. The ESR1 gene encodes an estrogen receptor and has been shown to have association with pregnancy-induced hypertension [Tamura et al. 2008]. Considerable evidence has been accumulated suggesting the involvement of receptor tyrosine kinases in the pathogenesis of pulmonary arterial hypertension [Pullamsetti et al. 2012]. The SRC gene encodes a tyrosine-protein kinase, and has been shown to be associated with pulmonary arterial hypertension [Pullamsetti et al. 2012]. The TLR4 gene encodes toll-like receptor 4, which contributes to blood pressure regulation and vascular contraction in spontaneously hypertensive rats [Bomfim et al. 2012] and was also reported to be associated with hypertension in



(a) Subgraph in protein interaction network (b) Subgraph in genetic interaction network

Fig. 10. The DCS_k ($k = 40$) identified from the ARIC dataset



(a) Subgraph in protein interaction network (b) Subgraph in genetic interaction network

Fig. 11. The DCS_*seed* identified from the ARIC dataset (renin pathway genes are in red ellipses)

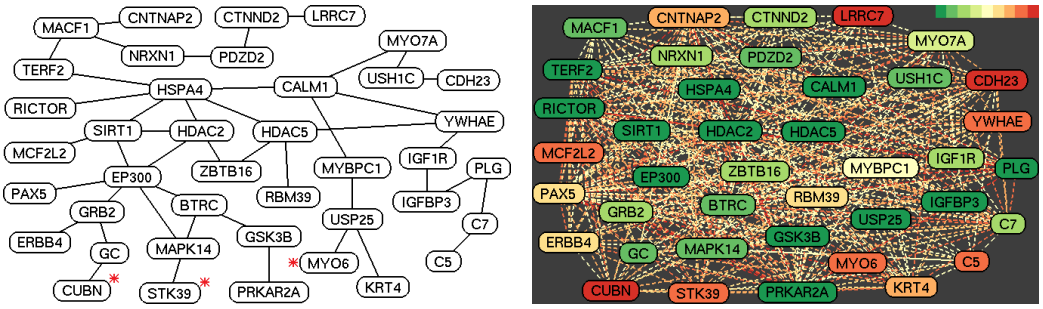
human [Zhu et al. 2010]. The association study in 199 Nigerian families reveals that the PARK2 gene is significantly associated with the risk for hypertension [Tayo et al. 2009]. This result is replicated in the Korean population [Jin et al. 2011]. The PARK2 and GRB2 genes exist in both the DCS_*seed*s identified from the WTCCC and ARIC datasets, which are shown in Figure 8 and 10 respectively. The GRB2 gene, together with the SRC gene, is interacting with the platelet-derived growth factor, which has been implicated in the pathobiology of vascular remodeling [Humbert et al. 2013]. These genes are highlighted by stars in the figure. Other genes in the identified subgraph are potential hypertension candidate genes or important for signal transduction in hypertension related pathways.

To identify the DCS_*seed*, we still use the set of 16 genes in the renin pathways as the input seed nodes [Yue et al. 2006]. The resulting subgraphs are shown in Figure 11. The newly added genes tend to have large degree in the genetic interaction network. We observe the ESR1, TLR4, and SRC genes, which have been discussed above. We also observe the NEDD4L gene, which was observed in Figure 9. In Figure 9, the DCS_*seed* contains 44 newly added genes in addition to the 16 seed genes from the renin pathways; in Figure 11, the DCS_*seed* contains 42 newly added genes. Between these two sets of newly added genes, there are 14 overlapping genes, such as NEDD4L, NEDD4, TP53, SIRT2, etc.

We also apply the GenGen, GRASS, Plink, and HYST methods to evaluate the statistical significance of the discovered DCSs. Note that we use the WTCCC dataset as the test dataset, which is independent of the ARIC dataset. Table V shows the p-value of the identified DCSs. DCS (1), (2) and (3) represent the top-3 DCSs. As can be seen

Table V. P-values of the DCSs Identified from the ARIC dataset (without node weights, tested on the WTCCC dataset)

Methods	GenGen	GRASS	Plink	HYST
DCS (1)	7.9×10^{-6}	5.7×10^{-6}	8.3×10^{-6}	3.1×10^{-8}
DCS (2)	5.2×10^{-5}	6.9×10^{-5}	9.4×10^{-5}	1.3×10^{-6}
DCS (3)	8.3×10^{-5}	1.2×10^{-4}	2.6×10^{-4}	6.4×10^{-6}
DCS_k	9.4×10^{-5}	2.1×10^{-5}	2.0×10^{-5}	4.5×10^{-7}
DCS_*seed*	7.3×10^{-4}	1.7×10^{-5}	6.8×10^{-4}	7.2×10^{-5}
DS	0.21	0.32	0.37	0.26
MSCS	0.12	0.14	0.23	0.09



(a) Subgraph in protein interaction network (b) Subgraph in genetic interaction network

Fig. 12. The DCS_k ($k = 40$) with node and edge weighted density identified from the WTCCC dataset

from the table, the DCSs are highly significant. In the table, we also show the results of the DS and MSCS methods. As we can see, the subgraphs identified by these two methods are not as significant as the DCSs.

10.1.3. The DCSs with Node and Edge Weighted Density. In this section, we study the effectiveness of the DCS method when the conceptual network has both node and edge weights as discussed in Section 8.

We perform single-marker chi-square test on the genetic markers in the WTCCC dataset. The single-marker test statistics are used as the node weights. Then, we apply the algorithms developed for the DCS problem with node and edge weighted density. The DCS identified in the dual networks has 176 nodes. The figures are omitted because of the large size. The set of nodes are sparsely connected in the protein interaction network, while the subgraph in the genetic interaction network has high node and edge weighted density. Specifically, the DCS has 217 edges in the protein interaction network and $3,928$ edges in the genetic interaction network.

Figure 12 shows the DCS_k with $k = 40$ identified from the WTCCC dataset. To better visualize the weights, in Figure 12(b), the node and edge weights are indicated by the colors in the color bar. The red color represents the maximum weight and the green color represents the minimum weight. The CUBN, STK39, MYO6 genes are also observed in this subgraph, which have been discussed before since they are also observed in Figure 8. These genes are highlighted by stars in the figure. In Figure 12(b), we can see that some genes, such as the PRKAR2A gene, have green node color, which means that they have weak single-marker association. However, they have strong interaction with other genes. If only the node weights are used, such as in the MSCS method, we will miss these important interactions.

We also use the set of 16 genes in the renin pathways as the input seed nodes, and discover the DCS_*seed* with the node and edge weighted density. The results are similar to that in Figure 9 and omitted here.

Table VI. P-values of the DCSs Identified from the WTCCC Dataset (with node weights, tested on the ARIC dataset)

Methods	GenGen	GRASS	Plink	HYST
DCS	5.8×10^{-6}	4.6×10^{-6}	6.7×10^{-6}	1.4×10^{-8}
DCS_k	8.2×10^{-5}	8.7×10^{-6}	9.4×10^{-6}	1.5×10^{-7}
DCS_*seed*	4.1×10^{-4}	7.7×10^{-6}	7.4×10^{-5}	9.1×10^{-6}
DS	0.15	0.26	0.23	0.25
MSCS	0.14	0.07	0.25	0.14

Table VII. P-values of the DCSs Identified from the ARIC Dataset (with node weights, tested on the WTCCC dataset)

Methods	GenGen	GRASS	Plink	HYST
DCS	4.2×10^{-6}	2.9×10^{-6}	3.6×10^{-6}	2.3×10^{-8}
DCS_k	6.9×10^{-5}	7.4×10^{-6}	6.6×10^{-6}	1.8×10^{-7}
DCS_*seed*	5.1×10^{-4}	8.5×10^{-6}	6.5×10^{-5}	8.3×10^{-6}
DS	0.21	0.32	0.18	0.35
MSCS	0.06	0.09	0.14	0.23

Table VIII. P-values of the DCSs Identified from the WTCCC Dataset (without node weights, tested on the ARIC dataset)

Methods	GenGen	GRASS	Plink	HYST
DCS (1)	7.2×10^{-6}	8.2×10^{-6}	7.6×10^{-6}	4.8×10^{-8}
DCS (2)	3.8×10^{-5}	3.2×10^{-5}	6.1×10^{-5}	3.5×10^{-7}
DCS (3)	6.5×10^{-5}	8.1×10^{-5}	9.8×10^{-5}	7.9×10^{-7}
DCS_k	8.9×10^{-5}	1.6×10^{-5}	2.2×10^{-5}	4.5×10^{-7}
DCS_*seed*	6.3×10^{-4}	2.1×10^{-5}	9.3×10^{-5}	1.8×10^{-5}
DS	0.27	0.36	0.38	0.21
MSCS	0.13	0.15	0.19	0.16

We further evaluate the statistical significance of the discovered DCSs. Note that to ensure the independence between the training and test datasets, when the DCSs are discovered from the WTCCC dataset, we use the ARIC dataset as the test dataset; when the DCSs are discovered from the ARIC dataset, we use the WTCCC dataset as the test dataset.

Tables VI and VII show the p-values of the DCSs identified from the WTCCC and ARIC datasets respectively. As can be seen from the table, the DCSs are highly significant. The subgraphs identified by the DS and MSCS methods are not as significant as the DCSs.

To compare the methods with and without node weights, we evaluate the significance of the DCSs, which are discovered from the WTCCC dataset, using the ARIC dataset. The results are shown in Table VIII. Previously, in Table IV, the DCSs discovered from the WTCCC dataset are evaluated using the other half of the WTCCC dataset. However, in Table VI, the results with node weights are evaluated using the ARIC dataset. To make a fair comparison, we compare the results evaluated using the same ARIC dataset. Comparing Tables VI and VIII, we can see that the p-values are smaller when node weights are integrated in the method. Comparing Tables VII and V, we also observe that the p-values are smaller when node weights are integrated. These results demonstrate that the integration of node weights gives better performance.

10.1.4. Gene Set Enrichment Analysis. To further understand the biological meaning of the discovered DCSs, we apply the standard gene set enrichment analysis [Bauer et al. 2008] to evaluate their significance. In particular, for each DCS (gene set) S, we identify the most significantly enriched KEGG (Kyoto Encyclopedia of Genes and Genomes) pathways (downloaded from *http://www.genome.jp/kegg/*). The significance (p-value) is determined by the Fisher's exact test. The raw p-values are further calibrated to cor-

Table IX. Gene Set Enrichment Analysis (WTCCC Dataset)

DCSs		KEGG pathways	p-values	Ref.
Only edge weights	DCS (1)	Neurotrophin signaling pathway	3.2×10^{-6}	[Smith et al. 2015]
	DCS (2)	ErbB signaling pathway	2.6×10^{-5}	[Matsukawa et al. 2011]
	DCS (3)	Glioma	7.5×10^{-5}	–
	DCS_k	Neurotrophin signaling pathway	9.3×10^{-6}	[Smith et al. 2015]
	DCS_*seed*	Renin-angiotensin system	2.8×10^{-5}	[Kobori et al. 2007]
Node and edge weights	DCS	Neurotrophin signaling pathway	1.4×10^{-6}	[Smith et al. 2015]
	DCS_k	ErbB signaling pathway	5.7×10^{-6}	[Matsukawa et al. 2011]
	DCS_*seed*	Renin-angiotensin system	8.3×10^{-6}	[Kobori et al. 2007]

Table X. Gene Set Enrichment Analysis (ARIC Dataset)

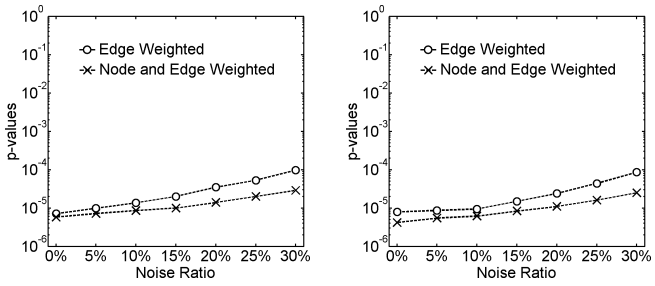
DCSs		KEGG pathways	p-values	Ref.
Only edge weights	DCS (1)	Calcium signaling pathway	4.7×10^{-6}	[Makani et al. 2011]
	DCS (2)	Neurotrophin signaling pathway	8.4×10^{-6}	[Smith et al. 2015]
	DCS (3)	ErbB signaling pathway	6.3×10^{-5}	[Matsukawa et al. 2011]
	DCS_k	MAPK signaling pathway	7.2×10^{-6}	[Bao et al. 2007]
	DCS_*seed*	Renin-angiotensin system	1.8×10^{-5}	[Kobori et al. 2007]
Node and edge weights	DCS	Calcium signaling pathway	1.6×10^{-6}	[Makani et al. 2011]
	DCS_k	Insulin signaling pathway	3.9×10^{-6}	[Carvalho-Filho et al. 2007]
	DCS_*seed*	Renin-angiotensin system	8.5×10^{-6}	[Kobori et al. 2007]

rect for the multiple testing problem [Westfall and Young 1993]. To compute calibrated p-values for each S, we perform a randomization test, wherein we apply the same test to 10^7 randomly created gene sets that have the same number of genes as S.

Tables IX and X show the most significantly enriched pathways and the corresponding p-values for DCSs identified from the WTCCC and ARIC datasets respectively. We can see that all the pathways have low p-values and are significantly enriched. We further study the existing literature and find that most of these pathways have been previously reported to be associated with hypertension. For example, the renin-angiotensin system is known to be associated with hypertension [Kobori et al. 2007]. The MAPK signaling pathway interacts with the angiotensin system [Bao et al. 2007]. The Neuregulin-1/ErbB signaling in rostral ventrolateral medulla is involved in blood pressure regulation as an antihypertensive system [Matsukawa et al. 2011]. The brain-derived neurotrophic factor may be a compensatory mechanism for the high blood pressure in Africans [Smith et al. 2015]. The calcium signaling pathway is reported to interact with the renin-angiotensin system [Makani et al. 2011], and it may also contribute to the hypertension pathogenesis. The insulin signaling pathway is also reported to interact with the renin-angiotensin system [Carvalho-Filho et al. 2007].

Moreover, we can observe that the DCSs identified from the node and edge weighted dual networks have more significant p-values than those identified from the dual networks with only edge weights do. This indicates that integrating node weights can further increase the significance of the detected patterns.

10.1.5. Robustness Evaluation. The protein interaction network is usually noisy. In this section, we perform simulation study to evaluate the robustness of our method. Specifically, given a noise ratio $\tau\%$, we randomly remove $\tau\%$ edges from the protein interaction network, and then randomly add the same number of edges. Thus we get a noisy protein interaction network. We then find the DCS from these dual networks, and evaluate the significance of the discovered DCS by the GenGen method. Note that to ensure the independence between the training and test datasets, when the DCSs are discovered from the WTCCC dataset, we use the ARIC dataset as the test dataset;



(a) WTCCC dataset (b) ARIC dataset

Fig. 13. Robustness analysis

when the DCSs are discovered from the ARIC dataset, we use the WTCCC dataset as the test dataset.

Figure 13(a) shows the p-values of the DCSs identified from the WTCCC dataset. We can see that the p-values of the discovered DCSs slightly increase when we increase the noise ratio. When the noise ratio is 30%, our method can still find significant patterns. Figure 13(b) shows the p-values of the DCSs identified from the ARIC dataset. A similar trend can be observed. These results demonstrate that our method is robust to the noises in the protein interaction network. We can also observe that the method with node and edge weights is more robust than the method with only edge weights. This indicates that integrating node weights can further increase the robustness of our method.

10.2. Effectiveness Evaluation in Other Application Domains

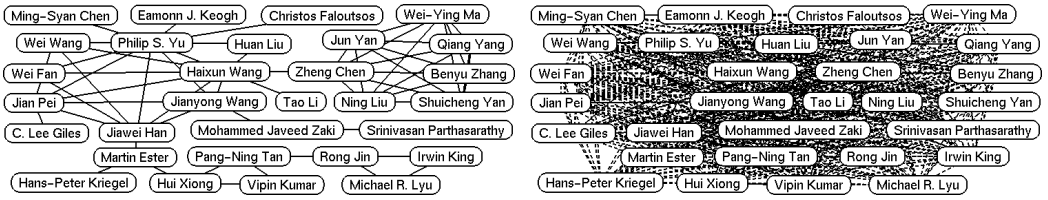
In addition to the biological application, we further evaluate the effectiveness of the DCS method in two other application domains. One application is about the bibliographic information analysis, and the other is about the social recommender system.

We use the DBLP dataset [Tang et al. 2008] to build two dual networks, one for data mining research community and one for database research community. To construct the dual networks for the data mining community, we extract a set of papers published in 5 data mining conferences: KDD, ICDM, SDM, PKDD and CIKM. The dataset contains 4,284 papers and 7,169 authors. The physical network is the co-author network with authors being the nodes and edges representing two authors have co-authored a paper. The conceptual research interest similarity network among authors is constructed based on the similarity of the terms in the paper titles of different authors. The shrunk Pearson correlation coefficient is used to compute the research interest similarity between authors [Koren 2008]. The dual networks for the database community are constructed in a similar way based on papers published in SIGMOD, VLDB and ICDE.

We construct two dual networks using recommender system datasets, Flixster [Jamali and Ester 2010] and Epinions [Massa and Avesani 2007]. In the original Flixster dataset, the physical network has 786,936 nodes (users) and 7,058,819 edges representing their social connectivity. The user-item rating matrix consists of 8,184,462 user ratings for 48,791 items with rating scale from 1 to 5 with 0.5 increment. We construct the conceptual interest similarity network by measuring the correlation coefficients of the common ratings between users [Ma et al. 2011]. Note that we only calculate the correlation coefficient between two users with more than 5 common ratings. The constructed interest similarity network has 2,713,671 edges. The trust network in Epinions dataset has 49,288 nodes and 487,002 edges. The user-item rating matrix consists of 664,811 user ratings for 139,737 items with rating scale from 1 to 5 with 1 incre-

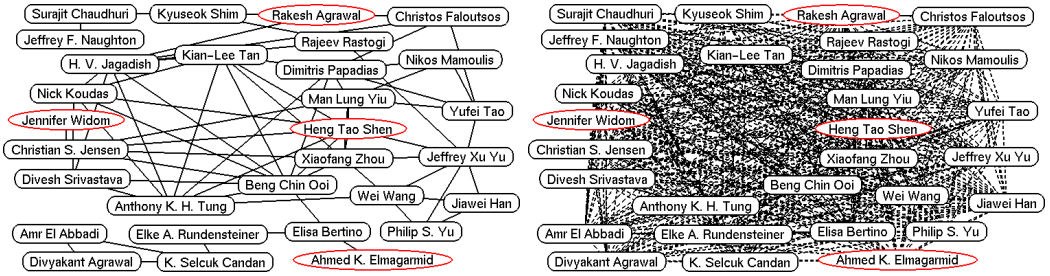
Table XI. Statistics of the Dual Networks in Other Application Domains

Dual networks	Abbr.	#nodes	#edges in G_a	#edges in G_b
Research-DM	DM	7,169	14,526	30,000
Research-DB	DB	6,131	17,940	30,000
Recom-Epinions	EP	49,288	487,002	313,432
Recom-Flixster	FX	786,936	7,058,819	2,713,671



(a) Subgraph in co-author network (b) Subgraph in research interest similarity network

Fig. 14. The DCS_k ($k = 30$) identified from the dual co-author (data mining) networks



(a) Subgraph in co-author network (b) Subgraph in research interest similarity network

Fig. 15. The DCS_*seed* identified from the dual co-author (database) networks

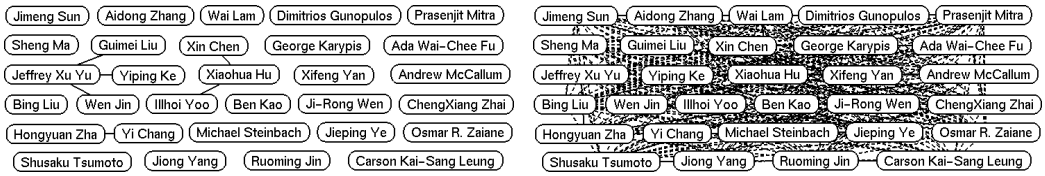
ment. The interest similarity network is constructed in a similar way as the one in the Flixster dataset. It has 313,432 edges. Table XI shows the basic statistics of the dual networks in these applications.

10.2.1. Research Interest Similarity and Co-Author Dual Networks. The DCS identified in the dual co-author networks consists of hundreds of nodes. Here we only show the identified DCS_k for data mining in Figure 14 and DCS_*seed* for database in Figure 15.

Figures 14(a) and 14(b) shows the DCS_k ($k = 30$) identified in the dual networks of the data mining research community. The subgraph in the co-author network is sparsely connected and highly dense in the research interest similarity network. This indicates that the set of researchers have very close research interest. The subgraph in the co-author network shows their collaboration pattern.

Figures 15(a) and 15(b) shows the DCS_*seed* identified in the dual networks of the database research community. The names of the 4 input seed authors are in red ellipses. The researchers with similar interest and their collaboration patterns are clearly shown in the two subgraphs. The 4 seed authors do not have direct co-authorship with each other. Through the resulting DCS_*seed*, we uncover the connected community of common interests.

Note that a dense subgraph in the research interest similarity network may not be connected in the co-author network. One example is shown in Figure 16. Figure 16(b) shows a dense subgraph identified from the research interest similarity network for data mining. Figure 16(a) shows the induced subgraph in the co-author network. We



(a) Induced subgraph in co-author network (b) Dense subgraph in research interest similarity network

Fig. 16. The dense subgraph in the research interest similarity network of the dual co-author (data mining) networks

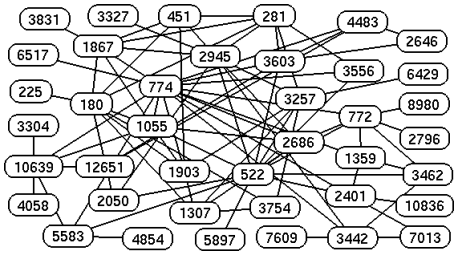
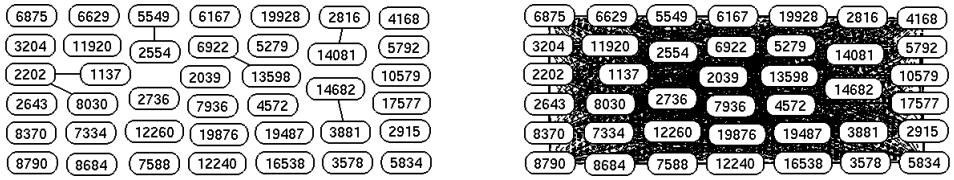


Fig. 17. The social connectivity network of the DCS_k ($k = 40$) identified in the Epinions dataset



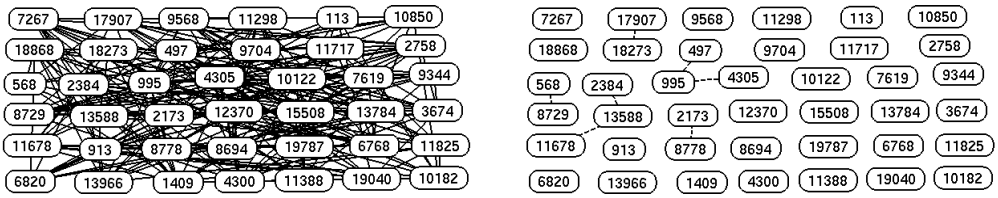
(a) Induced subgraph in social connectivity network (b) Dense subgraph in interest similarity network

Fig. 18. The dense subgraph in the interest similarity network of the Epinions dataset

can see that very few authors are connected. Thus finding dense subgraphs in a single network may miss important information presented in the other network.

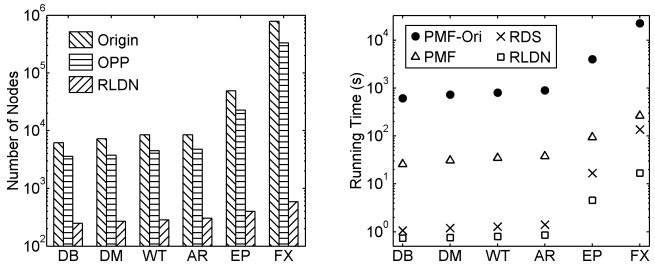
10.2.2. User Interest Similarity and Social Connectivity Dual Networks. The DCS_k ($k = 40$) identified in the dual network constructed from the Epinions dataset is shown in Figure 17. The subgraph in the interest similarity network is a dense component and not shown here. In this figure, each node is a user, whose name is not shown because of the privacy issue. Because this group of users have high interest similarity and also have social connection, if one of the users receives an advertisement of an interested product, this information is likely to be propagated to the rest of the group.

To demonstrate the effectiveness of the DCS pattern, we compare it with the dense subgraphs discovered from a single network. Figure 18(b) shows a dense subgraph in the interest similarity network. Figure 18(a) shows its induced subgraph in the social connectivity network. We can see that this set of users have no social connectivity even though they have high interest similarity. Figure 19(a) shows a dense subgraph in the social connectivity network. Figure 19(b) shows its induced subgraph in the interest similarity network. We can see that the subgraph in the interest similarity network is very sparse. This indicates that a group of users having high social connectivity may not have similar interest. Similar observations can be made in the dual networks constructed from the Flixster dataset.



(a) Dense subgraph in social connectivity network (b) Induced subgraph in interest similarity network

Fig. 19. The dense subgraph in the social connectivity network of the Epinions dataset



(a) Pruning effect (b) Running time

Fig. 20. Pruning effect and running time of the DCS_RDS algorithm

10.3. Efficiency Evaluation on Real Networks

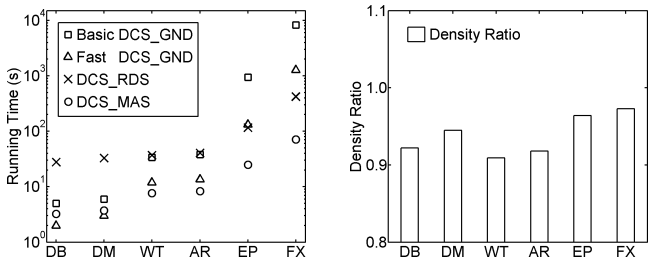
In this subsection, we evaluate the efficiency of the proposed DCS_RDS, DCS_GND and DCS_MAS algorithms using real networks.

The DCS_RDS algorithm has three major components: removing low degree nodes (RLDN) in the conceptual network, finding the densest subgraph in the remaining graph by parametric maximum flow (PMF), and refining the densest subgraph (RDS) to make it connected in the physical network.

We first evaluate the pruning effect of the RLDN step. Figure 20(a) shows the number of nodes in the original graph and the number of nodes remained after pruning. It can be seen that the RLDN step can reduce the number of nodes by 2 to 4 orders of magnitude. Moreover, the pruning effect becomes larger for larger graphs. This indicates that the RLDN step is more effective when graph size increases. The effect of the optimality preserved pruning (OPP) approach discussed in Section 4 is also shown in this figure. We can see that the OPP step can prune 40% to 60% nodes in the 6 real graphs. Since the real graphs are scale-free, there are many leaf nodes in the physical networks and the OPP step has large pruning ratio.

Figure 20(b) shows the running time of each step in DCS_RDS. We also run the parametric maximum flow method on the original graph (PMF-Ori) to see the performance improvement of our method. From the results, we can see that the RLDN and RDS steps run efficiently. The most time consuming part is to use parametric maximum flow to find the densest subgraph. Because of the pruning effect of RLDN, finding the densest subgraph after the RLDN step is about 2 orders of magnitude faster than directly finding it in the original graph.

Figure 21(a) shows the running time of the basic and fast DCS_GND methods. We can see that the fast DCS_GND method runs about 1 order of magnitude faster than the basic method, even though they have the same theoretical complexity. This demonstrates the effectiveness of simultaneously deleting independent non-articulation nodes. The running time of DCS_RDS is also shown in the figure for comparison. We can observe that DCS_GND runs faster on smaller graphs and DCS_RDS runs faster on larger graphs. The reason is that when the graph becomes larger, the depth first



(a) Running time (b) Density ratio

Fig. 21. (a) Running time of DCS_RDS, basic and fast DCS_GND, and DCS_MAS; (b) Density ratio of the subgraphs identified by DCS_RDS and basic DCS_GND

Table XII. Approximation Ratios on Real Networks

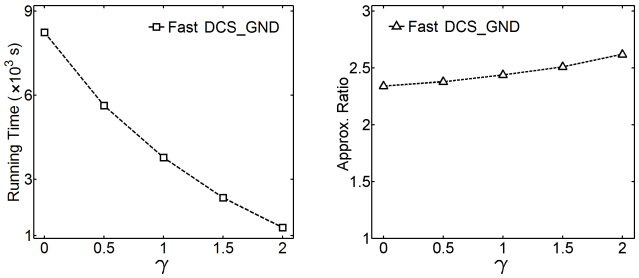
Datasets →	DB	DM	WT	AR	EP	FX
DCS_RDS	1.48	1.42	1.94	1.83	1.23	2.25
Basic DCS_GND	1.53	1.44	2.11	1.97	1.26	2.34
Fast DCS_GND	2.21	2.10	2.35	2.14	1.87	2.62
DCS_MAS	3.45	4.21	5.16	5.28	6.39	6.72

search procedure in DCS_GND will take longer time. On the other hand, more nodes will be removed by DCS_RDS for larger graphs as demonstrated in Figure 20(a). The running time of DCS_MAS is also shown in the figure. In the DCS_MAS method, we randomly select 8 seed nodes initially and set the parameter $K = 2000$. We can see that DCS_MAS takes about one hundred seconds on large graphs. Since DCS_MAS searches the whole graph to compute the Steiner tree, DCS_MAS has increasing running time when the graph size increases.

Figure 21(b) shows the ratio of the density of the subgraphs identified by DCS_RDS and DCS_GND. It can be seen that the densities of the subgraphs identified by these two methods are very similar. The DCS_GND method always results slightly larger density value. The reason is that the densest subgraph in the conceptual network may not have large overlap with the DCS. The exact DCS solution may contain other dense components instead of the densest subgraph because of the connectivity constraint in the physical network.

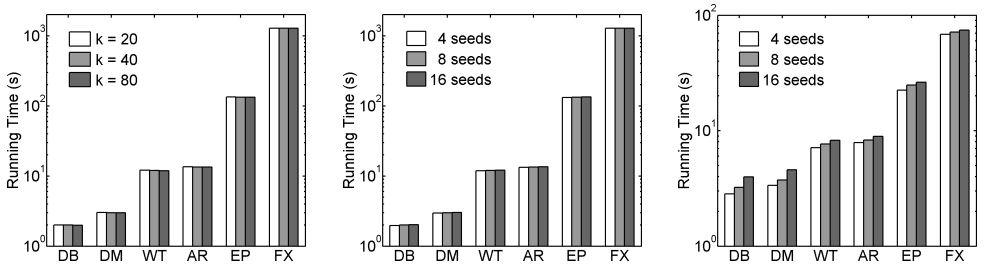
Table XII shows the estimated approximation ratio of the proposed methods on different datasets. In the fast DCS_GND method, we set $\gamma = 2.0$. From the table, we can see that the approximation ratio of DCS_RDS is tighter than that of DCS_GND. The reason is that DCS_RDS uses the exact densest subgraph in its first step. We can also observe that the approximation ratio of the basic DCS_GND method is always smaller than that of the fast DCS_GND method. This is because the fast DCS_GND method is greedier, which deletes a set of low degree nodes in each iteration. The basic DCS_GND method only deletes the node with the minimum degree. From the table, we can also see that the approximation ratio of both methods is around 2, which is the theoretical approximation ratio of the greedy node deletion algorithm for finding the densest subgraph in a single graph. Since we compute the density of the densest subgraph in the DCS_RDS method, we can compute the approximation ratio of the DCS_MAS method. The approximation ratios of DCS_MAS are also shown in Table XII. Compared with the approximation ratios of the other methods, the approximation ratios of the DCS_MAS method are larger. The reason is that the local dense subgraph, which DCS_MAS aims to search for, may have smaller density than the global densest subgraph.

Figure 22(a) shows the running time of the fast DCS_GND method on the Flixster dataset when varying γ. When $\gamma = 0$, it degrades to the basic DCS_GND method. When increasing γ, the fast DCS_GND method will delete more nodes in each iteration.



(a) Running time (b) Approx. ratio

Fig. 22. Effects of γ in fast DCS_GND



(a) Varying the size k (b) Varying the number of seeds (c) Varying the number of seeds

Fig. 23. Running time of fast DCS_GND (a,b) and DCS_MAS (c)

Thus the running time decreases. Figure 22(b) shows the approximation ratio when varying γ. We can see that the approximation ratio slightly increases when increasing γ. This indicates that the approximation ratio of the fast DCS_GND method is not very sensitive to γ.

Figure 23(a) and Figure 23(b) show the running time of the fast DCS_GND method when varying the output size k in DCS_k problem and varying the number of seeds in DCS_*seed* problem respectively. From the results, we can see that fast DCS_GND has almost constant running time when varying k and the number of seeds. This is because the DCS_GND method keeps deleting nodes from the dual networks and is not sensitive to k and the number of seeds. Figure 23(c) shows the running time of the DCS_MAS method when varying the number of seeds in the DCS_*seed* problem. We can see that the running time slightly increases when increasing the number of seeds. It may costs more time to find the Steiner tree when there are more seed nodes.

We further evaluate the DCS_RDS, DCS_GND and DCS_MAS methods for the DCS problem following the node and edge weighted density in Definition 8.1. The running time results are similar to that in Figure 21(a). The approximation ratio results are similar to that in Table XII. These results are omitted.

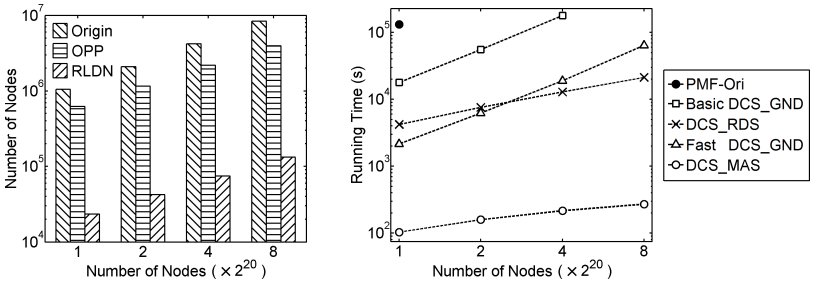
10.4. Efficiency Evaluation on Large Synthetic Networks

To further evaluate the scalability of the proposed methods, we generate a series of synthetic dual networks. Both the physical and conceptual networks are scale-free graphs based on the R-MAT model [Chakrabarti et al. 2004]. We use the graph generator from *http://www.cse.psu.edu/∼madduri/software/GTgraph/*. The statistics of the generated graphs are shown in Table XIII.

Figure 24(a) shows the pruning ratio of the OPP and RLDN steps in the DCS_RDS method. The OPP step can prune about 50% nodes, while the RLDN step can further reduce the number of nodes by 1$\sim$2 orders of magnitude.

Table XIII. Statistics of Synthetic Dual Networks

#nodes	1×2^{20}	2×2^{20}	4×2^{20}	8×2^{20}
#edges in G_a, G_b	1×10^7	2×10^7	4×10^7	8×10^7



(a) Pruning ratio (b) Running time of different methods

Fig. 24. Results on synthetic dual networks

Table XIV. Approximation Ratios on Synthetic Networks

#nodes	1×2^{20}	2×2^{20}	4×2^{20}	8×2^{20}
DCS_RDS	1.53	1.48	1.44	1.41
Basic DCS_GND	1.58	1.54	1.51	–
Fast DCS_GND	1.72	1.69	1.67	1.63
DCS_MAS	5.86	5.63	5.42	5.16

Figure 24(b) shows the running time of the DCS_RDS, DCS_GND and DCS_MAS methods. DCS_RDS has slower increasing rate. The reason is that the RLDN step has larger pruning ratio on larger graphs. The fast DCS_GND method runs about 1 order of magnitude faster than the basic DCS_GND method. This figure also shows the running time of the PMF method on the original conceptual network. PMF cannot be applied to large networks because of its long running time. DCS_MAS has increasing running time when the graph size increases. This is because it will take more time to find the Steiner tree when the graph size increases.

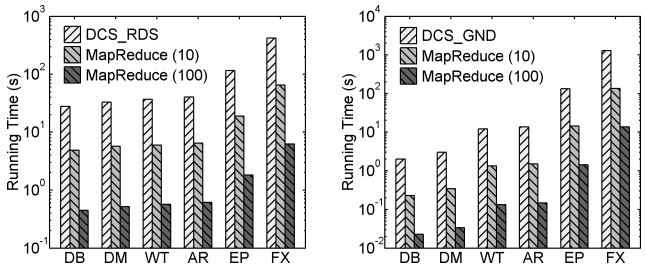
Table XIV shows the approximation ratios of the DCS_RDS, DCS_GND and DCS_MAS methods. The approximation ratio becomes tighter when the graph size increases.

To evaluate the scalability of the methods for the DCS problem following the node and edge weighted density in Definition 8.1, we randomly generate node weights on the synthetic conceptual networks. The running time results are similar to that in Figure 24(b). The approximation ratio results are similar to that in Table XIV. These results are omitted.

10.5. Efficiency Evaluation of the MapReduce Implementation

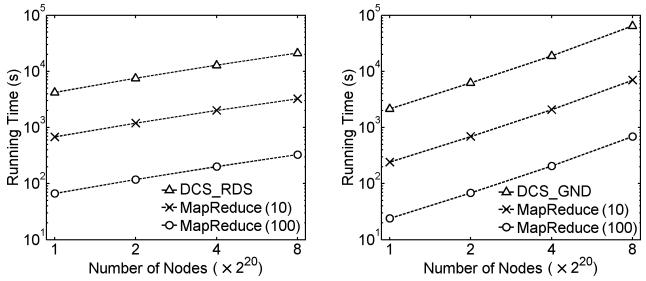
In this subsection, we evaluate the efficiency of the MapReduce implementation of the DCS_RDS and DCS_GND algorithms. We rent 101 nodes from the Amazon's Elastic Compute Cloud. The parameter γ in the DCS_GND method is set to 2.0.

Figure 25(a) shows the running time of the DCS_RDS method on real networks. The legends "MapReduce (10)" and "MapReduce (100)" denote that 10 and 100 worker nodes are used in the MapReduce implementation respectively. The MapReduce implementation with 10 worker nodes is about 6.1 times faster than the sequential algorithm. The MapReduce implementation with 100 worker nodes is about 64.5 times faster than the sequential algorithm. Figure 25(b) shows the running time of the DCS_GND method on real networks. The MapReduce implementation with 10 worker nodes is about 9.2



(a) DCS_RDS (b) DCS_GND

Fig. 25. Running time on real networks



(a) DCS_RDS (b) DCS_GND

Fig. 26. Running time on synthetic networks

times faster than the sequential algorithm. The MapReduce implementation with 100 worker nodes is about 91.5 times faster than the sequential algorithm. Comparing Figures 25(a) and 25(b), we can see that the MapReduce implementation of the DCS_GND method gets larger speed improvement than that of the DCS_RDS method does. The reason is that the DCS_GND method adopts the simple greedy node deletion strategy, which is quite suitable for the distributed computing environment.

Figures 26(a) and 26(b) show the running time of the DCS_RDS and DCS_GND methods on large synthetic networks respectively, whose statistics are shown in Table XIII. Similar patterns can be observed.

11. CONCLUSION

Dual networks exist in many real-life applications, where the physical and conceptual networks encode complementary information. In this paper, we study the problem of finding the densest connected subgraph in dual networks. A dense subgraph in the conceptual network that is also connected in the physical network can unravel interesting patterns that are invisible to the existing methods. We formulate the DCS problem and prove it is NP-hard. To find the DCS, we first introduce an effective optimality pruning strategy to remove the nodes that are not in the optimal solution. Then, we develop two efficient greedy algorithms to find the DCS. We also develop an efficient local search heuristic for the DCS problem with input seed nodes. We further study the DCS problem when there are node weights in the conceptual network, and extend the algorithms to solve this new problem. Extensive experimental results on real and synthetic datasets demonstrate the interestingness of the identified patterns and the efficiency of the proposed algorithms.

REFERENCES

José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2005. K-core decomposition: a tool for the visualization of large scale networks. *arXiv preprint cs/0504107* (2005).

Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. 2000. Greedily finding a dense subgraph. *Journal of Algorithms* 34, 2 (2000), 203–221.

Giorgio Ausiello, Donatella Firmani, Luigi Laura, and Emanuele Paracone. 2012. *Large-scale graph biconnectivity in MapReduce*. Technical Report.

Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Densest subgraph in streaming and MapReduce. *PVLDB* 5, 5 (2012), 454–465.

Weike Bao and others. 2007. Effects of p38 MAPK Inhibitor on angiotensin II-dependent hypertension, organ damage, and superoxide anion production. *Journal of cardiovascular pharmacology* 49, 6 (2007), 362–368.

Sergio E Baranzini, Nicholas W Galwey, Joanne Wang, and others. 2009. Pathway and network-based analysis of genome-wide association studies in multiple sclerosis. *Human Molecular Genetics* 18, 11 (2009), 2078–2090.

Vladimir Batagelj and Matjaz Zaversnik. 2003. An $O(m)$ algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

Sebastian Bauer, Steffen Grossmann, Martin Vingron, and Peter N Robinson. 2008. Ontologizer 2.0 – a multifunctional tool for GO term enrichment analysis and data exploration. *Bioinformatics* 24, 14 (2008), 1650–1651.

Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. 2010. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *STOC*. 201–210.

Gisele F Bomfim, Rosangela A Dos Santos, Maria Aparecida Oliveira, and others. 2012. Toll-like receptor 4 contributes to blood pressure regulation and vascular contraction in spontaneously hypertensive rats. *Clinical Science* 122, 11 (2012), 535–543.

Paul R Burton, David G Clayton, Lon R Cardon, and others. 2007. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature* 447, 7145 (2007), 661–678.

Marco A de Carvalho-Filho and others. 2007. Insulin and angiotensin II signaling pathways cross-talk: implications with the association between diabetes mellitus, arterial hypertension and cardiovascular disease. *Arquivos Brasileiros de Endocrinologia & Metabologia* 51, 2 (2007), 195–203.

Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *SDM*. 442–446.

Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*. 139–152.

Jie Chen and Yousef Saad. 2012. Dense subgraph extraction with application to community detection. *TKDE* 24, 7 (2012), 1216–1230.

Lin S Chen, Carolyn M Hutter, John D Potter, Yan Liu, Ross L Prentice, Ulrike Peters, and Li Hsu. 2010. Insights into colon cancer etiology via a regularized approach to gene set analysis of GWAS data. *The American Journal of Human Genetics* 86, 6 (2010), 860–871.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2001. *Introduction to algorithms*. MIT Press.

Giorgio Gallo, Michael D Grigoriadis, and Robert E Tarjan. 1989. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* 18, 1 (1989), 30–55.

Andrew V Goldberg. 1984. *Finding a maximum density subgraph*. Technical Report.

Felix Halim, Roland HC Yap, and Yongzheng Wu. 2011. A MapReduce-based maximum-flow algorithm for large small-world network graphs. In *ICDCS*. 192–202.

KW Hong, MJ Go, HS Jin, and others. 2009. Genetic variations in ATP2B1, CSK, ARSG and CSMD1 loci are related to blood pressure and/or hypertension in two Korean cohorts. *Journal of Human Hypertension* 24, 6 (2009), 367–372.

Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* 21, suppl 1 (2005), i213–i221.

Marc Humbert, Oleg V Evgenov, and Johannes-Peter Stasch. 2013. *Pharmacotherapy of pulmonary hypertension*. Vol. 218. Springer.

Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F Siegel. 2002. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics* 18, suppl 1 (2002), S233– S240.

Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*. 135–142.

Peilin Jia and Zhongming Zhao. 2014. Network-assisted analysis to prioritize GWAS results: principles, methods and perspectives. *Human Genetics* 133, 2 (2014), 125–138.

Peilin Jia, Siyuan Zheng, Jirong Long, Wei Zheng, and Zhongming Zhao. 2011. dmGWAS: dense module searching for genome-wide association studies in protein–protein interaction networks. *Bioinformatics* 27, 1 (2011), 95–102.

Hyun-Seok Jin, Kyung-Won Hong, Bo-Young Kim, and others. 2011. Replicated association between genetic variation in the PARK2 gene and blood pressure. *Clinica Chimica Acta* 412, 17 (2011), 1673–1677.

Richard M Karp. 1972. *Reducibility among combinatorial problems*. Springer.

Ryan Kelley and Trey Ideker. 2005. Systematic interpretation of genetic interactions using protein networks. *Nature Biotechnology* 23, 5 (2005), 561–566.

Mikko Kivelä, Alexandre Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. 2013. Multilayer networks. *arXiv preprint arXiv:1309.7233* (2013).

Hiroyuki Kobori and others. 2007. The intrarenal renin-angiotensin system: from physiology to the pathobiology of hypertension and kidney disease. *Pharmacological reviews* 59, 3 (2007), 251–287.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.

Guy Kortsarz and David Peleg. 1994. Generating sparse 2-spanners. *Journal of Algorithms* 17, 2 (1994), 222–236.

Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, 303–336.

Daniel Levy, Georg B Ehret, Kenneth Rice, and others. 2009. Genome-wide association study of blood pressure and hypertension. *Nature Genetics* 41, 6 (2009), 677–687.

Miao-Xin Li, Johnny SH Kwan, and Pak C Sham. 2012b. HYST: A hybrid set-based test for genome-wide association studies, with application to protein-protein interaction-based association analysis. *The American Journal of Human Genetics* 91, 3 (2012), 478–488.

Wenyuan Li, Haiyan Hu, Yu Huang, Haifeng Li, Michael R Mehan, Juan Nunez-Iglesias, Min Xu, Xifeng Yan, and Xianghong Jasmine Zhou. 2012a. Pattern mining across many massive biological networks. In *Functional Coherence of Molecular Networks in Bioinformatics*. Springer, 137–170.

Jimmy Lin and Chris Dyer. 2010. *Data-intensive text processing with MapReduce*.

Fang Luo, Yibo Wang, Xiaojian Wang, Kai Sun, Xianliang Zhou, and Rutai Hui. 2009. A functional variant of NEDD4L is associated with hypertension, antihypertensive response, and orthostatic hypotension. *Hypertension* 54, 4 (2009), 796–801.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*. 287–296.

Harikrishna Makani and others. 2011. Effect of renin-angiotensin system blockade on calcium channel blocker-associated peripheral edema. *The American journal of medicine* 124, 2 (2011), 128–135.

Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *RecSys*. 17–24.

Ryuichi Matsukawa and others. 2011. Neuregulin-1/ErbB signaling in rostral ventrolateral medulla is involved in blood pressure regulation as an antihypertensive system. *Journal of hypertension* 29, 9 (2011), 1735–1742.

Gearoid M McMahon, Conall M O'Seaghdha, Shih-Jen Hwang, James B Meigs, and Caroline S Fox. 2013. The association of a single-nucleotide polymorphism in CUBN and the risk of albuminuria and cardiovascular disease. *Nephrology Dialysis Transplantation* (2013), gft386.

Kurt Mehlhorn. 1988. A faster approximation algorithm for the Steiner problem in graphs. *IPL* 27, 3 (1988), 125–128.

Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *KDD*. 228–238.

Patrick Phillips. 2008. Epistasis - the essential role of gene interactions in the structure and evolution of genetic systems. *Nature Review Genetics* 9, 11 (2008), 855–867.

Snehit Prabhu and Itsik Pe'er. 2012. Ultrafast genome-wide scan for SNP-SNP interactions in common complex disease. *Genome Research* 22, 11 (2012), 2230–2240.

Soni Savai Pullamsetti, Eva Maria Berghausen, Swati Dabral, and others. 2012. Role of Src tyrosine kinases in experimental pulmonary hypertension. *Arteriosclerosis, Thrombosis, and Vascular Biology* 32, 6 (2012), 1354–1365.

Shaun Purcell and others. 2007. PLINK: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics* 81, 3 (2007), 559–575.

Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. 2010. Dense subgraphs with restrictions and applications to gene annotation graphs. In *RECOMB*. 456–472.

Thomas P Slavin, Tao Feng, Audrey Schnell, Xiaofeng Zhu, and Robert C Elston. 2011. Two-marker association tests yield new disease associations for coronary artery disease and hypertension. *Human Genetics* 130, 6 (2011), 725–733.

AJ Smith and others. 2015. Attenuated brain-derived neurotrophic factor and hypertrophic remodelling: the SABPA study. *Journal of human hypertension* 29, 1 (2015), 33–39.

Yan V. Sun and Sharon L.R. Kardia. 2010. Identification of epistatic effects using a protein-protein interaction database. *Human Molecular Genetics* 19, 22 (2010), 4345–4352.

Masaaki Tamura, Tomohiro Nakayama, Ichiro Sato, and others. 2008. Haplotype-based case-control study of estrogen receptor α (ESR1) gene and pregnancy-induced hypertension. *Hypertension Research* 31, 2 (2008), 221–228.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *KDD*. 990–998.

Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1, 2 (1972), 146–160.

Bamidele O Tayo, Amy Luke, Xiaofeng Zhu, Adebowale Adeyemo, and Richard S Cooper. 2009. Association of regions on chromosomes 6 and 7 with blood pressure in Nigerian families. *Circulation: Cardiovascular Genetics* 2, 1 (2009), 38–45.

the ARIC Investigators. 1989. The atherosclerosis risk in communities (ARIC) study: design and objectives. *American Journal of Epidemiology* 129, 4 (1989), 687–702.

Igor Ulitsky and Ron Shamir. 2007. Pathway redundancy and protein essentiality revealed in the Saccharomyces cerevisiae interaction networks. *Molecular Systems Biology* 3 (2007), 104.

Kai Wang, Mingyao Li, and Maja Bucan. 2007. Pathway-based approaches for analysis of genomewide association studies. *The American Journal of Human Genetics* 81, 6 (2007), 1278–1283.

Kai Wang, Mingyao Li, and Hakon Hakonarson. 2010. Analysing biological pathways in genome-wide association studies. *Nature Review Genetics* 11, 12 (2010), 843–854.

Ying Wang, Jeffrey R O'Connell, Patrick F McArdle, and others. 2009. Whole-genome association study identifies STK39 as a hypertension susceptibility gene. *PNAS* 106, 1 (2009), 226–231.

Peter H Westfall and Stanley S Young. 1993. *Resampling-based multiple testing*. Wiley, New York.

Yubao Wu, Ruoming Jin, Xiaofeng Zhu, and Xiang Zhang. 2015. Finding Dense and Connected Subgraphs in Dual Networks. In *ICDE*.

Peng Yue, Eugene Melamud, and John Moult. 2006. SNPs3D: candidate gene and SNP selection for association studies. *BMC Bioinformatics* 7, 1 (2006), 166.

Xiaofeng Zhu, Tao Feng, Yali Li, Qing Lu, and Robert C Elston. 2010. Detecting rare variants for complex traits using family and unrelated data. *Genetic Epidemiology* 34, 2 (2010), 171–187.

Received October 2014; revised March 2015; accepted May 2015