Netw Model Anal Health Inform Bioinforma (2012) 1:103–116
DOI 10.1007/s13721-012-0009-3

ORIGINAL ARTICLE

Merging network patterns: a general framework to summarize biomedical network data

Yang Xiang · David Fuhry · Kamer Kaya · Ruoming Jin · Ümit V. Çatalyürek · Kun Huang

Received: 19 December 2011 / Revised: 29 April 2012 / Accepted: 8 May 2012 / Published online: 16 June 2012

Abstract The ability to summarize a large number of network patterns discovered from biomedical data provides valuable information for use in many applications. We show that several variants of the problem are all NP-hard, and merging network patterns is a practical solution for these applications. In this work, we propose an algorithmic framework for merging network patterns. We have developed fast algorithms under this general framework which supports several types of biomedical network data. In addition, our empirical study demonstrates that our algorithms are efficient in merging a large number of biomedical network patterns and can be configured for various knowledge discovery purposes.

Keywords Network patterns · Summarization · Graph theory · Cliques · Bicliques · Frequent itemset mining

Y. Xiang (✉) · K. Kaya · Ü. V. Çatalyürek · K. Huang
Department of Biomedical Informatics, The Ohio State University, Columbus, OH 43210, USA
e-mail: yxiang@bmi.osu.edu

K. Kaya
e-mail: kamer@bmi.osu.edu

Ü. V. Çatalyürek
e-mail: umit@bmi.osu.edu

K. Huang
e-mail: kun.huang@osumc.edu

D. Fuhry
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA
e-mail: fuhry@cse.ohio-state.edu

R. Jin
Department of Computer Science, Kent State University, Kent, OH 44242, USA
e-mail: jin@cs.kent.edu

1 Introduction

Biomedical data are frequently modeled in the form of networks, i.e., graphs. For example, gene-coexpression data, which show the correlation between genes, can be modeled as a graph $G = (V, E)$ where each vertex represents a gene and each edge represents the correlation between two genes. Furthermore, we can also model the disease–gene relationship data as a bipartite graph $G = (V_A, V_B, E)$ where the vertex sets V_A and V_B represent the sets of diseases and genes, respectively, such that an edge exists between $d \in V_A$ and $g \in V_B$ if and only if the disease d and the gene g are related. In general, we can roughly classify these networks constructed from such biological data into two categories: ones which are modeled as graphs with a single vertex set V and others which are modeled as bipartite graphs using two vertex sets V_A and V_B. Both types of graphs can be weighted or unweighted. That is, the edge weights can be any real numbers, or just binary values.

One of the most important tasks for the biomedical networks is to find significant patterns. Often, dense subgraphs in these networks correspond to important biomedical structures. For example, dense subgraphs in gene-coexpression networks often contain genes that have similar biological functions with prognostic powers in patient diagnosis (Uppalapati et al. 2010; Zhang et al. 2010). For the disease–gene data, the dense subgraphs have been successfully used in predicting unknown disease–gene relations (Xiang et al. 2012b), an application comparable to link prediction in social networks (Almansoori et al. 2012). The definition of the density varies with respect to the application and graph model. For an unweighted graph, the density can be defined as the ratio of the edge count in the subgraph to the number of its possible edges. For a weighted

graph, the density can be defined as the ratio of the total weight in the subgraph to the number of possible edges. It is easy to see that cliques and bicliques of unweighted graphs have a density of 1 for the above definitions.

In this paper, we will use $\delta(G)$ to denote the density of a network G, and we use $G_i \subseteq G$ to denote that G_i is a subgraph of G. Since there are often too many subgraph patterns to focus on, summarizing a large number of patterns into a small number of informative patterns is desirable for many applications. We formally define the basic summarization problem as follows:

Problem 1 Let $\mathcal{P} = \{G_1, G_2, \ldots, G_p\}$ be a set of network patterns of G such that G_i is a subgraph of G for all $1 \leq i \leq p$. The summarization problem is defined as finding another set of network patterns $\mathcal{G}' = \{G'_1, G'_2, \ldots, G'_q\}$ with minimum cardinality q such that for each $G_i \in \mathcal{P}$, there exists a $G'_j \in \mathcal{G}'$ where $G_i \subseteq G'_j$ and $\delta(G'_j) \geq \beta$.

The restriction on the density function δ ensures that a network pattern is a highly connected component. There are several ways to measure the density of a network pattern. In this work, we measure a network's density based on its matrix representation and edge weights are made nonnegative. Thus, edge directions and signs (if any) are ignored. Since both graphs and bipartite graphs can be represented as a matrix, we consider A to be the set of rows, and B to be the set of columns of a matrix M. To facilitate our measurement, we consider every matrix entry (an edge weight) to be normalized between 0 and 1, and every diagonal entry on a symmetric matrix to be 1. Thus, the density of a bipartite graph induced by $V_A \subseteq A$ and $V_B \subseteq B$ is:

$$\delta(G(V_A, V_B)) = \frac{\sum_{v_i \in V_A, v_j \in V_B} M(v_i, v_j)}{|V_A||V_B|}$$

As a unified measurement, in this work the density of a graph (i.e., a symmetric matrix) is defined in the same manner except that V_A and V_B are required to be identical. The unified measurement significantly simplifies our algorithm design as we can always treat an input graph as a bipartite graph (or a matrix). In an unweighted (bipartite) graph, an edge always has a weight of 1. Thus, the unweighted (bipartite) graph is a (bi) clique if and only if its density is 1.

In this work, we propose a general framework to summarize network patterns for biomedical applications. Our main contributions are:

- We provide theoretical analysis to illustrate that the problem and several variations are NP-hard, and subsequently we propose a subproblem and a general algorithm framework for summarizing network patterns in real biomedical applications (Sect. 3).
- We design two efficient merge algorithms to fit the summarization framework. By carefully designing the network pattern merge operation, we show that the time complexity of our summarization framework is even comparable to simple hierarchical clustering algorithms in some circumstance (Sect. 4).
- We demonstrate that our algorithms are very efficient in handling real datasets, and through batch processing, our algorithms can summarize very large datasets (Sect. 5.1).
- Finally, by a comprehensive application study, we demonstrate that our algorithms can be configured to achieve various purposes in analyzing biomedical network data. Our study includes mining network patterns that are significant in patient prognosis, macro-level network clustering, and visual analysis (Sect. 5.2).

2 Related work

There are many methods that can be used to obtain network patterns. Our work, as compared to these methods, has several advantages: (1) It can easily control the summarization scale by adjusting the density threshold β, and it accepts various network data formats. Thus, it can be applied to various summarization scenarios (micro level or macro level). (2) It can guarantee the density of each network pattern in the final result. (3) Each network pattern in the original input will be covered by at least one network pattern in the summarized output. (4) It is highly scalable and can be applied to summarize very large biomedical datasets. Finally, our work can be applied to summarize network patterns obtained from other methods. Thus it has wide applicability in biomedical network analysis. In the following we discuss the difference between related methods and our approach.

2.1 Network partition

These methods partition a network into several non-overlapping sub-networks. Clustering (Berkhin 2006), coclustering (or biclustering) (Hartigan 1972; Mirkin 1996), and graph partitioning techniques (e.g., Stoer 1997) are typical examples. However, in many biomedical applications, researchers are interested in finding small and highly connected network patterns rather than network partitions. These network patterns may connect to potential biomarkers and lead to novel scientific discovery (Kutalik et al. 2008; Mushlin et al. 2009; Ravetti 2008; Zhang et al. 2010). As a result, methods in this category are not sufficient for these types of applications.

2.2 Clique or biclique enumeration

A clique (or biclique) is a fully connected network component. Different from network partitions, two different cliques of a graph may overlap. Listing all maximal cliques

is a classical problem which has been well studied in the past (Bron and Kerbosch 1973; Tsukiyama et al. 1977; Makino 2004; Johnson et al. 1988). The task for mining maximal bicliques was often accomplished by mining closed itemsets (Du et al. 2008; Li et al. 2007; Xiang et al. 2012a). In addition to algorithms that list cliques or bicliques, there are many algorithms focusing on identifying quasi-cliques (Abello et al. 2002; Li et al. 2008; Seidman 1983; Seidman and Foster 1978) or quasi-bicliques (Geerts et al. 2004; Jin et al. 2010). The quasi-cliques or quasi-bicliques can be described as the dense components which are close to cliques or bicliques with a small amount of edges missing. However, the methods identifying dense network patterns often produce more patterns than biologists can focus on. Moon and Moser (1965) proved that a graph with n vertices can have up to $3^{n/3}$ maximal cliques if $n \equiv 0$ (*mod* 3). Thus, clique or biclique enumeration methods alone are not good choices for analyzing biomedical data.

2.3 Pattern summarization

A few pattern summarization methods have been proposed to tackle the pattern exploration problem in clique (or biclique) enumeration and related problems. Our approach in this work falls into this category. However, the past pattern summarization methods have two major problems. First, methods such as Hyper (Xiang et al. 2011) claim to produce summarized patterns but still output too many patterns for biological analysis. Second, available summarization methods (Lucchese et al. 2010; Miettinen et al. 2008), Hyper+ in (Xiang et al. 2011) to our knowledge solve the problem heuristically and without a quality (density) guarantee on individual network patterns in the final output.

It is also worthwhile to note an additional stratification of graph/network summarization work (e.g., Li et al. 2011) that focuses on summarizing graph patterns over a set of graphs. Our work, in contrast, centers on developing summarization techniques over network patterns discovered from a single graph. Although it is possible to extend our approach to multiple graphs, we limit our context to summarizing patterns from a single graph only in this work.

2.4 Pattern growing

Some recent work (e.g. Xiang et al. 2012b; Ou and Zhang 2007) obtain network patterns from a weighted graph by a pattern growing approach. Although technical details vary slightly, their work flows are similar: Start a cluster with one edge. In each iteration, add a new vertex to the cluster that adds maximal weight to the cluster. Stop adding vertices when the increased weight resulted from adding the new vertex is less than a threshold. To reduce computational time as well as the number of final patterns, a covered edge (Xiang et al. 2012b) [or an edge with two covered vertices (Ou 2007)] will not be selected again as a start point for a new cluster. This simple greedy approach can guarantee the density of the generated pattern. However, it is easy to show that it may miss an important dense pattern as a result of being separately covered by several other patterns. This issue can be avoided by dropping the coverage check. However, if doing so, the algorithm may generate too many patterns, and the worst case time complexity is $\mathcal{O}(n^5)$, too high for real graphs.

3 An algorithmic framework for summarizing network patterns

Let us first show that Problem 1 is intractable.

Theorem 1 *Problem 1 is NP-hard.*

Proof We use a reduction from the well known minimum clique cover problem, which is NP-hard (Karp 1972). The problem is defined as follows: given an unweighted graph $G = (V, E)$, find a minimum number of cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ such that each edge $e \in E$ is covered by at least one clique in $\mathcal{C}$. The polynomial time reduction is given below.

(1) Create a set of networks $\mathcal{P}$ where each network in $\mathcal{P}$ corresponds to an edge in E. That is, each edge in E with its endpoints is a network in $\mathcal{P}$.
(2) $\beta = 1$.

According to this definition and that of Problem 1, $\mathcal{P}'$ is a clique cover for G. Hence, the solution for Problem 1, which requires minimum $|\mathcal{P}'|$, is also the solution for the minimum clique cover problem described above. □

After proving Problem 1 is NP-hard, we consider a simplified version. Since merging several networks in $\mathcal{P}$ usually creates a larger one that covers these networks, we focus on summarizing network patterns by merging. The merge operation is defined as

$$\begin{aligned}&\text{NetMerge}((V_{A_1}, V_{B_1}, E_1), \ldots, (V_{A_k}, V_{B_k}, E_k))\\ &= G\left(\bigcup_{i=1}^{k} V_{A_i}, \bigcup_{i=1}^{k} V_{B_i}\right)\end{aligned}$$

where $G\left(\bigcup_{i=1}^{k} V_{A_i}, \bigcup_{i=1}^{k} V_{B_i}\right)$ is the (bipartite) graph induced by the rows $\bigcup_{i=1}^{k} V_{A_i}$ and columns $\bigcup_{i=1}^{k} V_{B_i}$ of the matrix M.

Let us define a problem similar to Problem 1 which necessitates using the operation NetMerge for network pattern summarization.

Problem 2 Let $\mathcal{P} = \{G_1, G_2, \ldots, G_p\}$ be a set of network patterns of G such that G_i is a subgraph of G for all $1 \leq i \leq p$. The summarization problem is defined as finding another set of network patterns $\mathcal{P}' = \{G'_1, G'_2, \ldots, G'_q\}$ with minimum cardinality q such that for each $G_i \in \mathcal{P}$, there exists a $G'_j \in \mathcal{P}'$ where $G_i \subseteq G'_j, \delta(G'_j) \geq \beta$, and

$$G'_j = \text{NetMerge}(G_{i_1}, \ldots, G_{i_m})$$

is obtained by merging a set of patterns $G_{i_\ell} \in \mathcal{P}$ for $1 \leq \ell \leq m$.

Let us prove that the cardinalities of the solutions for Problems 1 and 2 are equal in the following case.

Lemma 1 *If $\beta = 1$ is applied for both Problems 1 and 2, the summarized sets of their solutions have the same cardinality.*

Proof Let q and q' be the cardinality of the sets obtained by solving Problems 1 and 2, respectively, for the pattern set $\mathcal{P}$. Since the latter problem is a restricted version of the former, $q \leq q'$.

Given a solution $\mathcal{P}' = \{G'_1, G'_2, \ldots, G'_q\}$ for Problem 1, a solution with the same cardinality q can be constructed for Problem 2 as follows: let $C(G'_i) = \{G_j : G_j \subseteq G'_i\}$ be the set of network patterns in $\mathcal{P}$ which are subgraphs of G'_i. Note that $\bigcup_{i=1}^{q} C(G'_i) = \mathcal{P}$. Then the summarized set

$$\mathcal{P}'' = \{\text{NetMerge}(C(G'_1)), \ldots, \text{NetMerge}(C(G'_q))\}$$

is a solution for Problem 2 with the same cardinality q since merge operations cover any pattern in $\mathcal{P}$ and $\delta(\text{NetMerge}(C(G'_i))) = 1$ for all $1 \leq i \leq q$. □

After Lemma 1, the following corollary is immediate:

Corollary 1 *Problem 2 is NP-hard.*

Since Problem 2 is intractable, let us consider a simplified version. In this version, we remove a network pattern from $\mathcal{P}$ after it is merged into a new network pattern. Hence, the number of network patterns decreases after each merge operation. This "merge and delete" strategy can be useful in practice by leading us to more efficient solutions for the summarization problem. This strategy also corresponds to finding a non-overlapping *partition* of the given network pattern set $\mathcal{P}$. Note that here the non-overlapping partition of $\mathcal{P}$ does not imply a non-overlapping partition of G. A non-overlapping partition of $\mathcal{P}$ is denoted as $\Pi = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ where

- parts are pairwise disjoint, i.e., $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for all $1 \leq i < j \leq k$,
- each part $\mathcal{P}_i$ is a nonempty subset of $\mathcal{P}$, i.e., $\mathcal{P}_i \subseteq \mathcal{P}$ and $\mathcal{P}_i \neq \emptyset$ for $1 \leq i \leq k$,
- union of k parts is equal to $\mathcal{P}$, i.e., $\bigcup_{i=1}^{k} \mathcal{P}_i = \mathcal{P}$.

We define a more restricted version of Problem 2 as follows:

Problem 3 Let $\mathcal{P} = \{G_1, G_2, \ldots, G_p\}$ be a set of network patterns of G such that G_i is a subgraph of G for all $1 \leq i \leq p$. The summarization problem is defined as finding another set of network patterns $\mathcal{P}' = \{G'_1, G'_2, \ldots, G'_q\}$ with minimum cardinality q where for each $G'_j \in \mathcal{P}', \delta(G'_j) \geq \beta$, and there exists a partition $\Pi = \{\mathcal{P}_1, \ldots, \mathcal{P}_q\}$ of $\mathcal{P}$ such that $G'_j = \text{NetMerge}(\mathcal{P}_j)$.

Problem 3 is a restricted version of Problem 2 since each network pattern in $\mathcal{P}$ can only be merged into one pattern in $\mathcal{P}'$. That is, a network pattern disappears after being merged into a new network pattern. This problem is also intractable.

Theorem 2 *Problem 3 is NP-hard.*

Proof We use the same reduction from the minimum clique cover problem as in the proof of Theorem 1 to the network summarization problem. That is, we create the initial network pattern set $\mathcal{P}$ by constructing a network pattern for each edge in E, and again we set $\beta = 1$. Assume that $\mathcal{P}'$ is the solution to this problem. Due to the definition of δ, each network pattern in $\mathcal{P}'$ is a clique. If $\mathcal{P}'$ corresponds to a solution with minimum cardinality for the clique cover, we are done. Otherwise, there must exist a solution $\{C_1, C_2, \ldots, C_k\}$ for the clique cover problem with $k < q$. Let E_i be the edge set of the clique C_i, and let $G(E_i)$ be the graph induced by endpoints of edges in E_i. We can obtain a solution $\mathcal{P}'' = \{G''_1, \ldots, G''_k\}$ for the summarization problem where

$$G''_1 = G(E_1),$$

$$G''_i = \text{NetMerge}\left(G(\{e\}) : e \in E_i \setminus \bigcup_{j=1}^{i-1} E_j\right)$$

for all $1 < i \leq k$. Note that each G''_i is a clique since the merge operation uses edges in the original graph G. This is a contradiction since $\mathcal{P}'$ and $\mathcal{P}''$ are both solutions for Problem 3 and $|\mathcal{P}''| < |\mathcal{P}'|$. □

After proving the NP-hardness of Problem 3, we look for other solutions. Since a merge operation on k networks is equivalent to $k - 1$ merge operations on 2 networks, it is reasonable to solve Problem 3 by repetitively merging a set of network pairs until a summarized network pattern set with desired properties is obtained. The following is a subproblem we need to solve repeatedly during this process.

Problem 3s (Subproblem of Problem 3) Let $\mathcal{P} = \{G_1, G_2, \ldots, G_p\}$ be a set of network patterns of G such that G_i is a subgraph of G for all $1 \leq i \leq p$. The problem is defined as finding another set of network patterns

$\mathcal{P}' = \{G'_1, G'_2, \ldots, G'_q\}$ with $q < p$, where there exists a partition $\Pi = \{\mathcal{P}_1, \ldots, \mathcal{P}_q\}$ of $\mathcal{P}$ such that $|\mathcal{P}_j| \leq 2, G'_j = \text{NetMerge}(\mathcal{P}_j)$, and $\delta(G'_j) \geq \beta$ for all $1 \leq j \leq q$.

Algorithm 1 SUMNETWORK($G, \mathcal{P} = \{G_1, G_2, \ldots, G_p\}, \delta, \beta$)

```
1: P' ← G
2: for all G'_i, G'_j ∈ P' do
3:     G'_ij ← NETPATTERNMERGE(G'_i, G'_j)
4:     if δ(G'_ij) ≥ β then
5:         Q ← Q ∪ {(δ(G'_ij), G'_i, G'_j)}
6:     end if
7: end for
8: while true do
9:     ▶ MERGE solves an instance of Problem 3s at each
       iteration, updates P' accordingly, and returns the set
       of new network patterns M.
10:    M ← MERGE(P', Q)
11:    if M = ∅ then
12:        break
13:    end if
14:    for all G'_i ∈ M do
15:        for all G'_j ∈ P' \ {G'_i} do
16:            G'_ij ← NETPATTERNMERGE(G'_i, G'_j)
17:            if δ(G'_ij) ≥ β then
18:                Q ← Q ∪ {(δ(G'_ij), G'_i, G'_j)}
19:            end if
20:        end for
21:    end for
22: end while
23: return P' = {G'_1, G'_2, ..., G'_q}
```

A generic framework for an algorithm using this approach is given in Algorithm 1. At each iteration of the while loop in the algorithm, the MERGE operation constructs a solution for Problem 3s by selecting a set of network pairs and merging them simultaneously. MERGE takes two inputs: the current summarized network pattern set $\mathcal{P}'$ and a queue $\mathcal{Q}$ which contains a set of triplets formed by a pair of networks G'_i, G'_j, and the density of $\delta(G'_{ij})$ where G'_{ij} is the network obtained by merging G'_i and G'_j. All network pairs in $\mathcal{Q}$ are *feasible*. That is, $\delta(G'_{ij}) \geq \beta$. When MERGE chooses a triplet from $\mathcal{Q}$, it removes the two corresponding networks from $\mathcal{P}'$ and inserts the one obtained by the NETPATTERNMERGE operation, which includes the previously defined NETMERGE operation *plus calculating the density of the merged pattern*.

MERGE returns the set of these new network patterns. After MERGE returns, the framework updates $\mathcal{Q}$ by adding all new feasible pairs into it. Note that when we remove a pattern G'_i from $\mathcal{P}'$, the triplets in $\mathcal{Q}$ which have G'_i as one of the networks become obsolete. MERGE ignores such triplets and removes them from $\mathcal{Q}$ upon visiting.

As mentioned above, MERGE in Algorithm 1 solves an instance of Problem 3s. Note that Problem 3s does not have an objective function. Here we discuss three objective functions which can be optimized in polynomial time.

1. *Minimize q*: Choosing the largest possible $\mathcal{M}$ at each iteration optimizes this objective function. Given a pattern set $\mathcal{P}'$, consider a graph $G' = (V', E')$ where there exists a vertex $v \in V'$ for each network pattern in $\mathcal{P}'$, and an edge $uv \in E'$ if the pair of network patterns corresponding to u and v is feasible. A matching in G' is a set E'' of edges $(E'' \subseteq E')$ such that each vertex in V' is an endpoint of at most one edge in E''. A matching is called *maximum* if it has the maximum possible cardinality. The maximum number of feasible network pairs in $\mathcal{P}'$ that can be merged pairwise simultaneously is equal to the cardinality of the maximum matching in G'. There are several graph matching algorithms in the literature. The first matching algorithm is proposed by Edmonds with $\mathcal{O}(|V|^4)$ complexity (Edmonds 1965). Later, Micali and Vazirani 1980 proposed another algorithm with $\mathcal{O}(\sqrt{|V|}|E|)$ complexity. The best theoretical complexity $\mathcal{O}(|V|^{2.38})$ is obtained by Mucha and Sankowski (2004) with a randomized algorithm based on fast matrix multiplication.
2. *Minimize q and then maximize* $\sum\delta(G'_j)$: For this objective, consider the graph G' in the previous heuristic but this time each edge uv has a weight $w(uv)$ which is equal to the density of network pattern obtained by merging the corresponding patterns of u and v. Again, the maximum matching in G' has the same cardinality as the maximum number of feasible network pattern pairs in $\mathcal{P}'$. Note that there can be more than one set $\mathcal{M}$ of pairs having this cardinality. Hence, choosing the one with maximum $\sum_{(G'_i,G'_j)\in\mathcal{M}} \delta(\mathcal{P}'_{ij})$ value is promising to maximize the overall density of the summarized network patterns in the output. This problem is equivalent to finding a *maximum (minimum) weighted maximum cardinality matching* in G'. Note that this problem is as hard as the maximum cardinality matching problem described above. We refer the reader to (Duan et al. 2011) for a latest result and a brief survey about the complexities of maximum weighted matching algorithms. Since the number of initial network patterns is huge in practice, we do not prefer to adopt this heuristic considering the time complexity of the existing algorithms for the maximum weighted matching problem.
3. *Maximize* $\sum\delta(G'_j)$ **with** $q = p - 1$: In this objective function, we want to maximize $\sum\delta(G'_j)$ with only one merge operation on 2 network patterns. To maximize this objective function, one can implement the MERGE operation by choosing one pair which achieves the maximum density after the merge operation. Note that on the contrary, the merge operations in the previous objective functions were simultaneous. Compared with simultaneous merging, this single merge operation is

expected to perform better for maximizing the overall density because it has a larger search space. For example, in simultaneous merging, if network G_1 and G_2 are merged, and G_3 and G_4 are merged in the same iteration, then the possibility of merging G_1 and G_2 with G_3 will not be considered. However, this is not the case for the single merge operation. On the other hand, the single operation can be more expensive since the number of network patterns is reduced only one for each execution of MERGE. The experimental results in Sect. 5 confirm these predictions.

4 Fast algorithms for network summarization

In the previous section we discussed several problem formulations for summarizing network patterns and provided a general algorithm framework for Problem 3, a restricted version of Problems 1 and 2. The framework iteratively solves Problem 3s for which optimal solutions exist with the three different objectives discussed in the previous section. Since finding optimal solutions for the second objective function is too expensive, we will focus on the other two functions.

For the first objective, using existing matching algorithms for the MERGE operation in Algorithm 1 may not be efficient considering their complexities. For example, if we use the maximum matching algorithm of Micali and Vazirani (Micali and Vazirani 1980), the overall complexity of Algorithm 1 becomes $\mathcal{O}(p^{3.5})$ where p is the number of initial network patterns which is large in practice. If we use the algorithm of Mucha and Sankowski (2004), it slightly reduces the worst case complexity with a potential cost of the large memory consumption of the fast matrix multiplication. Since minimizing q in the first objective function does not lead to a global optimal result for Problem 3, it may be a good idea to relax the problem by using *maximal matchings* instead of maximum ones. A matching in a graph G is maximal if there is no edges in G that can be added to M. With this relaxation, we can obtain an efficient implementation of the MERGE operation for the first objective as shown in Algorithm 2.

Algorithm 2 $\mathcal{M} = \text{MULTIMERGE}(\mathcal{P}', \mathcal{Q})$

$\mathcal{M} \leftarrow \emptyset$
while $\mathcal{Q}$ has more triplets **do**
  Let $(\delta(G'_{ij}), G'_i, G'_j)$ be the first triplet in $\mathcal{Q}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{(\delta(G'_{ij}), G'_i, G'_j)\}$
  if $G'_i \in \mathcal{P}'$ **and** $G'_j \in \mathcal{P}'$ **then**
    $G'_{ij} \leftarrow \text{NETPATTERNMERGE}(G'_i, G'_j)$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{G'_{ij}\} \setminus \{G'_i, G'_j\}$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{G'_{ij}\}$
  end if
end while
return $\mathcal{M}$

MULTIMERGE, given in Algorithm 2, merges a maximal set of disjoint network pairs from the current set $\mathcal{P}'$. These pairs are obtained from the triplets in $\mathcal{Q}$. If both networks of a triplet are in $\mathcal{P}'$ then MULTIMERGE merges them and replaces them in $\mathcal{P}'$ with the one obtained after the merge operation. But, if one of the networks is not in $\mathcal{P}'$, the triplet is ignored.

There are two main implementation choices for $\mathcal{Q}$: the first one is the heap implementation where the first triplet in $\mathcal{Q}$ is always the one with the maximum density value. In this case, the best possible merge with the maximum density value will be realized each time and the add and remove operations for $\mathcal{Q}$ take logarithmic time. On the other hand, if $\mathcal{Q}$ is implemented as a simple queue, the triplets will be processed with a FIFO (first in first out) strategy and queue operations take constant time. The following lemma gives the time complexity of the framework given in Algorithm 1 for both cases.

Lemma 2 *If* MULTIMERGE *is used for merging, and* $\mathcal{Q}$ *is implemented as a min-heap*, SUMNETWORK *runs in* $\mathcal{O}(p^2 \log p) * \mathcal{O}(\mathcal{T})$ *time where* $p = |\mathcal{P}|$ *is the number of initial network patterns and* $\mathcal{O}(\mathcal{T})$ *is the worst case time complexity of* NETPATTERNMERGE *function. Furthermore, if* $\mathcal{Q}$ *is implemented as an unordered list of triplets the time complexity is* $\mathcal{O}(p^2) * \mathcal{O}(\mathcal{T})$.

Proof Let us first show that throughout the execution, the number of different possible triplets in $\mathcal{Q}$ is $\mathcal{O}(p^2)$. Since each merge operation reduces (i.e., delete 2 and add 1) the size of the pattern set by one, there can be $\mathcal{O}(p)$ merge operations. Hence, there are $\mathcal{O}(p)$ different network patterns which are either added, deleted, or kept in $\mathcal{P}'$. Each pair of networks uniquely defines a triplet. That is, there are $\mathcal{O}(p^2)$ triplets we can see throughout the execution of Algorithm 1. Note that if a triplet is removed from $\mathcal{Q}$, it will never be added to $\mathcal{Q}$ again. Hence, the total cost for queue updates is $\mathcal{O}(p^2 \log p^2) = \mathcal{O}(p^2 \log p)$ if a min-heap implementation is used. On the other hand, if a simple queue is used with constant time add-remove operations, the cost will be $\mathcal{O}(p^2)$.

Algorithm 3 $\mathcal{M} = \text{SINGLEMERGE}(\mathcal{P}', \mathcal{Q})$

$\mathcal{M} \leftarrow \emptyset$
while $\mathcal{Q}$ has more triplets **do**
  Let $(\delta(G'_{ij}), G'_i, G'_j)$ be the first triplet in $\mathcal{Q}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{(\delta(G'_{ij}), G'_i, G'_j)\}$
  if $G'_i \in \mathcal{P}'$ **and** $G'_j \in \mathcal{P}'$ **then**
    $G'_{ij} \leftarrow \text{NETPATTERNMERGE}(G'_i, G'_j)$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{G'_{ij}\} \setminus \{G'_i, G'_j\}$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{G'_{ij}\}$
    break
  end if
end while
return $\mathcal{M}$

For the last objective function, we implemented SINGLEMERGE as shown in Algorithm 3. The implementation of SINGLEMERGE is almost the same as that of MULTIMERGE. The only difference is after selecting a triplet and merging the corresponding network pair, SINGLEMERGE returns. Hence, the following corollary is immediate.

Corollary 2 *SUMNETWORK with SINGLEMERGE has the same time complexity as SUMNETWORK with MULTIMERGE.*

Quite interestingly, if we ignore the NETPATTERNMERGE time $\mathcal{O}(\mathcal{T})$, the running time of SUMNETWORK is comparable to SLINK (Sibson 1973) and CLINK (Defays 1977), two classic hierarchical clustering algorithms both with $\mathcal{O}(n^2)$ running time. This is understandable because our approach is analogous to "hierarchically clustering" network patterns, and this process can be illustrated with a hierarchical clustering tree as we will show in Sect. 5.

However, $\mathcal{O}(\mathcal{T})$, the worst-case running time of NETPATTERNMERGE, equals $\mathcal{O}(|A||B|)$. Recalling $|A|$ is the number of columns (or number of vertices in part A) and $|B|$ is the number of rows (or number of vertices in part B), the complexity $\mathcal{O}(|A||B|)$ is prohibitively high.

To tackle this computational challenge, we propose a very efficient way to realize the NETPATTERNMERGE function. Our idea was motivated by the fact that we can utilize the prior knowledge to calculate the density for merging two network patterns, as stated in the following lemma.

Lemma 3 *Let $G_i = G(A_i, B_i)$ and $G_j = G(A_j, B_j)$ be two bipartite subgraph patterns. Let $A' = A_i \cap A_j$ and $B' = B_i \cap B_j$. Then, we conclude that $weight(NetPatternMerge(G_i, G_j)) = (weight(G_i) + weight(G_j) - weight(G(A', B')) + weight(G(A_i - A', B_j - B')) + weight(G(A_j - A', B_i - B'))$, and $\delta(NetPatternMerge(G_i, G_j)) = weight(NetPatternMerge(G_i, G_j))/(|A_i \cup A_j||B_i \cup B_j|)$, where $G(X, Y)$ is a bipartite subgraph induced by vertex sets X and Y, and $weight(G(X, Y))$ is the total edge weight of the graph $G(X, Y)$.*

Although the proof of Lemma 3 is quite straightforward as shown in Fig. 1, it implies a huge reduction in the NETPATTERNMERGE computation. That is, if we keep the weight of each network pattern, then we do not need to start from the beginning in calculating the weight and density of a merged pattern. In order to calculate $weight(NetPatternMerge(G_i, G_j))$, we only need to calculate $weight(G(A', B'))$, $weight(G(A_i - A', B_j - B'))$, and $weight(G(A_j - A', B_i - B'))$, if $weight(G_i)$ and $weight(G_j)$ are known. Calculating $weight(G(A', B'))$ involves visiting overlapped edges between G_i and G_j, while calculating $weight(G(A_i - A', B_j - B'))$ and $weight(G(A_j - A', B_i - B'))$ involving visit edges that have not been visited before.

Thus, in the case where no network patterns have overlapped vertices, we have the following corollary.

Corollary 3 *Given a set of network patterns $\mathcal{P} = \{G_1, G_2, \ldots, G_p\}$ of G such that no two network patterns share a common vertex, SUMNETWORK runs in $\mathcal{O}(p^2 \log p) + \mathcal{O}(n^2)$ if $\mathcal{Q}$ is implemented as a min-heap, or $\mathcal{O}(p^2) + \mathcal{O}(n^2)$ if $\mathcal{Q}$ is implemented as an unordered list of triplets, where n is the number of vertices in G.*

Corollary 3 shows an impressive result: SUMNETWORK for summarizing non-overlapping network patterns has time complexity comparable to SLINK (Sibson 1973) and CLINK (Defays 1977) for hierarchically clustering simple data. However, when overlap occurs, Corollary 3 no longer holds, and the analysis needs to be amended by adding the total time for calculating the weight of the overlapping part for every merge operation (i.e., $weight(G(A', B'))$ in Fig. 1).

We observed that SUMNETWORK running time performance is very acceptable for handling real large datasets. By using batch processing, it can handle very large datasets as discussed in Sect. 5.

5 Empirical study

In this section, we empirically study the proposed network merging algorithms on two typical sets of data, an unweighted bipartite graph (the gene-to-phenotype dataset (Robinson et al. 2008, 2010) and a weighted graph (the gene coexpression data, a symmetric matrix, built on microarray dataset GSE2034 (Carroll et al. 2006; Wang et al. 2005). Following the two typical examples, it is easy to extend our algorithms to handle the other two types of datasets, unweighted graphs (i.e., symmetric (0,1) matrices) and weighted bipartite graphs.

Although MULTIMERGE and SINGLEMERGE are actually the names of the algorithms devised for the subproblem in Sect. 3, to facilitate our discussion, we will use them to denote the SUMNETWORK algorithms equipped with them.

We implemented network merging algorithms in C++ and tested them on computer clusters equipped with 2.4 GHz AMD Opteron processors and a Linux 2.6 Kernel. The survival test was implemented in Matlab and was carried out on a mainstream Windows PC. In addition, we also used the latest network visualization tool Gephi (Bastian et al. 2009) (version 0.8.1-beta for windows), to visualize network patterns.

5.1 Performance study on merging unweighted bipartite network patterns

The gene-phenotype dataset (Robinson et al. 2008, 2010)[1] is a typical unweighted bipartite graph where one set of

[1] Publicly available at: http://human-phenotype-ontology.org. The data for this work, downloaded on Feb 23, 2012, contains 1854 genes and 5220 phenotypes.

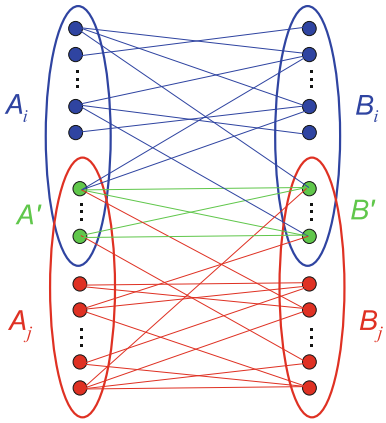

Fig. 1 Merge network patterns

vertices are genes and the other are phenotypes. This type of datasets can be represented as (0,1) matrices, and are well known as transactional data (Han et al. 2011). We revised HYPER (Xiang et al. 2011) code, which takes closed itemsets generated by MAFIA (Burdick et al. 2005) as input, to generate bipartite network patterns. We set minimum support to be 0.05 % for MAFIA. Under this minimum support level, MAFIA is supposed to generate all closed itemsets, which correspond to all maximal bicliques for the gene-phenotype dataset. Subsequently, our program completed the conversion from closed itemsets to their corresponding maximal bicliques.

To study the performance of MULTIMERGE and SINGLEMERGE, we created small sets of network patterns by selecting the first k (k ranges from 1,000 to 10,000) bipartite network patterns from the above output. In addition, we fixed β to be 90 and 70 %, two typical density ratios, for MULTIMERGE and SINGLEMERGE.

We applied MULTIMERGE and SINGLEMERGE on these network patterns. Figure 2 shows the running time of MULTIMERGE and SINGLEMERGE. Figure 3 shows the number of summarized network patterns by MULTIMERGE and SINGLEMERGE.

As expected, the running time in Fig. 2 and the number of summarized network patterns in Fig. 3 increase as the number of input network patterns increases. However, the increase rates are not sharp. The time (between 2 and 6 min) for summarizing 10,000 network patterns is also acceptable.

In addition, SINGLEMERGE (MULTIMERGE) with $\beta = 0.7$, takes longer time and results in a smaller number of summarized network patterns, than SINGLEMERGE (MULTIMERGE) with $\beta = 0.9$. This is quite understandable as a smaller β implies more merge operations, resulting in more compact network patterns as shown in Fig. 3.

Interestingly, we can see in Fig. 2, MULTIMERGE is substantially faster than SINGLEMERGE, although their worst case time complexities are the same, as indicated by Corollary 2. This constant-factor difference is understandable as SINGLEMERGE tends to maintain a larger number of records in $\mathcal{Q}$, while MULTIMERGE clears out $\mathcal{Q}$ in each iteration.

In Figs. 2 and 3, one can see that, in general, SINGLEMERGE produces a slightly smaller number of network patterns than MULTIMERGE at the cost of a longer running time. This suggests that the third objective function discussed in Sect. 3 for Problem 3s is a good heuristic for Problem 3. If fast running time is more desirable than a slightly more compact summarization result, MULTIMERGE will be more preferable than SINGLEMERGE.

However, in this experiment when we continue increasing input network patterns, SINGLEMERGE and MULTIMERGE reach memory limitation before the running time becomes unacceptably long. Thus, in order to summarize the complete set of maximal bicliques, we use batch processing, a strategy similar to what we have used for

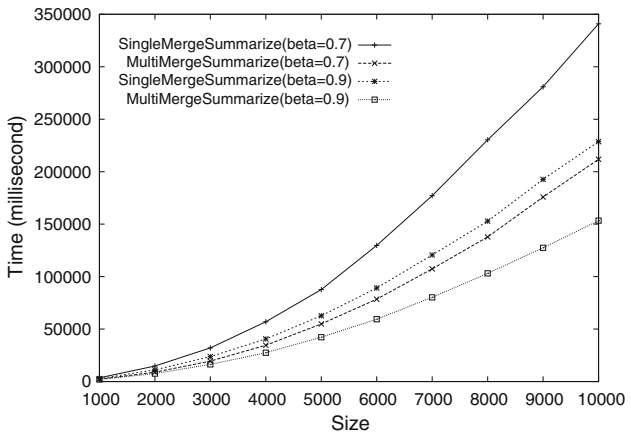

Fig. 2 Running time of MULTIMERGE and SINGLEMERGE for summarizing 1,000–10,000 patterns

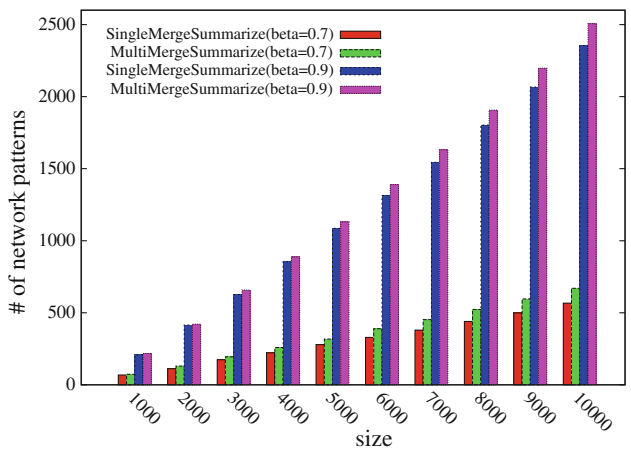

Fig. 3 Number of summarized (outputted) network patterns by MULTIMERGE and SINGLEMERGE under various β values

handling large datasets in (Jin et al. 2010, 2012). In the batch processing for this experiment, we separate the complete set of maximal bicliques into subsets such that each subset contains k bicliques or less. Then, we apply SingleMerge or MultiMerge separately on these subsets. We can iteratively apply this approach on summarized results until either the summarized network patterns are compact enough (e.g., smaller than the batch size), or there are only a few (or no) pattern number reductions between two consecutive iterations.

For example, by applying the above batch processing of SingleMerge ($\beta = 0.7$, batch size $k = 10{,}000$), the complete set of maximal biclique patterns of gene-phenotype dataset (148,001 maximal bicliques) was summarized into 41,402 network patterns with total summarization time less than 45 min. In the second round of summarization, SingleMerge was able to take all these 41,402 network patterns on our computing platform and they were summarized into 14,882 with total summarization time less than 10 min. Among the 14,882 network patterns, we found many interesting bipartite network patterns that may lead to novel hypotheses between gene and phenotype relations. In Table 1 we list three bipartite network pattern examples that involve three common diseases.

5.2 Application study on merging weighted network patterns

In the following we study our network merging approach on weighted network patterns. Different from the above study on unweighted bipartite patterns, we will focus on the bioinformatics application work flow using our network merging approach.

Our target is a gene coexpression network built on the microarray dataset GSE2034 (Carroll et al. 2006; Wang et al. 2005) by applying Spearman correlation on gene expressions. To make our study less affected by noisy information, we selected 7,446 gene symbols (and their expressions) from 13,785 gene symbols in the GSE2034. A gene is selected if it does not contain '///' and has a unique entry in the gene expression table. Thus, our gene coexpression network is a $7{,}446 \times 7{,}446$ symmetric matrix with each entry value ranging between 0 and 1 (all values were turned into absolute values).

In the following we will use some clique mining algorithm to generate clique patterns for our summarization study. For this purpose, we need to turn the graph into an unweighted graph by setting up a threshold. By sorting edge weights in ascending order as shown in Fig. 4, we observe that the edge weight increase rate changes sharply around 0.2–0.6. A majority of edges have weight less than 0.2, while a small portion of edges have weight more than 0.6. This suggests that 0.2–0.6 is a good range for choosing a threshold to turn the gene coexpression network into an unweighted graph. In this experiment we choose 0.6 as the threshold.

There are a few maximal clique mining algorithms available (e.g. Bron and Kerbosch 1973; Tsukiyama et al. 1977; Makino and Uno 2004; Johnson et al. 1988), among which the Bron–Kerbosch algorithm (1973) is often

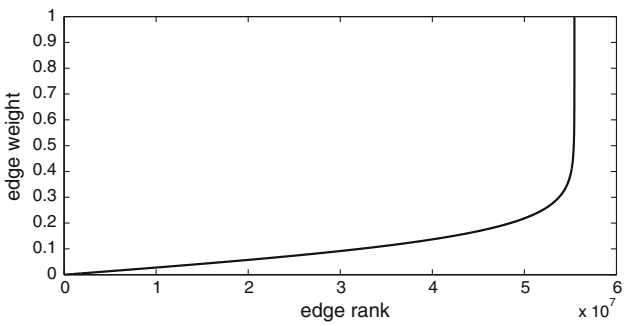

Fig. 4 Edge weights listed in ascending order

Table 1 Three bipartite network pattern examples selected from the result of applying batch processing of SingleMerge on all maximal biclique patterns of gene-phenotype datasets

	Density	Gene list	Phenotype list
A network pattern with breast carcinoma	0.704545	APC, CASP8, CTNNB1, MET, PDGFRL, PIK3CA, TP53, AXIN1	Abnormality of metabolism/homeostasis, autosomal dominant inheritance, autosomal recessive inheritance, breast carcinoma, conical teeth, decreased IgA, hepatocellular carcinoma, heterogeneous, micronodular cirrhosis, somatic mutation, subacute progressive viral hepatitis
A network pattern with colon cancer	0.73	ALPL, AQP2, BRAF, BTD, CASR, DGUOK, KRAS, MAP2K1, MAP2K2, SMPD1	Autosomal dominant inheritance, colon cancer, constipation, failure to thrive, feeding difficulties, muscular hypotonia, nystagmus, seizures, splenomegaly, vomiting
A network pattern with osteoarthritis	0.727273	COL1A1, COL1A2, COL3A1, COL5A1, FBN1, TNXB	Autosomal dominant inheritance, autosomal recessive inheritance, bruising susceptibility, conical incisor, hyperextensible skin, inguinal hernia, joint dislocation, joint laxity, mitral valve prolapse, osteoarthritis, soft skin

considered one of the best in practice. However, our implementation suggested that it is not scalable enough to handle this dataset in a reasonable time frame. Thus, we resort to frequent itemset mining techniques again. By treating the unweighted dataset as a transactional data, we use MAFIA (minimum support 0.1 %) to obtain a set of closed frequent itemsets and use our program to turn these closed frequent itemsets into bicliques. Since this dataset is actually a graph corresponding to a symmetric matrix, we need to extract cliques from these bicliques. If the largest clique in a biclique (the complete set of vertices that are in common in both sets of a biclique) contains three or more vertices, it is considered nontrivial and saved in our clique mining results. Eventually, we obtained 633,724 nontrivial cliques for the following study.

Then, we applied the batch processing of MultiMerge ($\beta = 0.7$, batch size $k = 10{,}000$) on this large amount of nontrivial cliques, and we surprisingly summarized them into only 1,451 network patterns. We further applied SingleMerge on these patterns and summarized them into 1,130 network patterns. It is important to note that during the summarization process, the density calculation is based on the original weighted coexpression matrix. Thus, edge weight information are fully considered in the summarization process, and the cliques obtained above can be regarded as "network backbones" guiding the summarization.

We measure the biological significance of each summarized pattern (that contains at least 10 genes) with respect to breast cancer prognosis through survival tests as described in the following. The expressions[2] of genes in a cluster are used as features to separate a given group of patients into two subgroups by K-means algorithm ($K = 2$, distance = cityblock, repeating 100 times). We use log-rank test[3] to determine the statistical significance (p value) of patient survival time difference between these subgroups.

The survival tests[4] were conducted on two additional datasets, the breast cancer dataset NKI (van de Vijver et al. 2002; van't Veer et al. 2002) (295 patients) and GSE1456 (159 patients). Since many available gene-signatures are not significant in subtype patient prognosis, especially for the ER-negative subtype (Desmedt et al. 2008; Reyal et al. 2008; van't Veer et al. 2002; Wirapati et al. 2008), we also conduct survival tests on two subtypes of patients in the NKI datasets, i.e., the lymph node positive (LN-positive) subtype (144 patients) and the estrogen receptor negative (ER-negative) subtype (69 patients), and ER-negative subtype patients of GSE2034[5] (77 patients). In summary, we conduct our survival tests on five cohorts of patients: GSE1456, NKI, NKI LN-positive, NKI ER-negative, GSE2034 ER-negative.

Among the 1,130 summarized network patterns, 242 are significant ($p < 0.05$) in at least one cohort survival test. Among them, we have observed gene lists more significant in NKI subtype (LN-positive, ER-negative) patient prognosis than the well-known 70-gene signature (van't Veer et al. 2002). For example, Figs. 5 and 6 show two summarized network patterns whose gene lists demonstrate better LN-positive patient prognosis and ER-negative patient prognosis, respectively, for the NKI dataset.

In order to understand the relationship between the 242 summarized network patterns, we continue merging them using SingleMerge. Since the process of SingleMerge is analogous to the hierarchical clustering, we visualized its dendrogram in Fig. 7. We can obtain the macro clusters from SingleMerge by either specifying a density threshold as we did in the above, or specifying the exact number of final clusters (slightly revising SumNetwork). By applying SingleMerge ($\beta = 0.4$) on the 242 summarized network patterns, we obtained 4 networks (listed in Table 2).

We further queried the gene list of each network via ToppGene[6], and found that each network is highly enriched with one or more gene ontology terms. The top enriched gene ontology terms are listed in Table 2. It is interesting to observe that the largest network is highly enriched with immune system process, and the second largest network is highly enriched with cell cycle process.

To further understand the macro relationship among the four networks, we visualize them by Gephi (Bastian et al. 2009) , a state-of-the-art network visualization tool. Since the gene coexpression network is fully connected, directly visualizing the whole network is not helpful due to the visual clutters caused by the excessive number of edges. Thus, we use two parameters (ε, ϵ) to reduce visual clutters as well as to highlight the four discovered network patterns. An edge is visualized if and only if its weight is no less than ε. In addition, if a visualized edge connects two vertices within a network (cluster), its original weight is used for visualization, otherwise, ϵ is used as the edge weight for visualization.

[2] If a gene name has multiple entries in the microarray, only an entry with the maximum express level is considered. Gene names without an entry (due to alias or other reasons) in the microarray will be omitted.

[3] Publicly available at: http://www.mathworks.com/matlabcentral/fileexchange/20388.

[4] For GSE1456 and GSE2034, there is not survival data but relapse data. Therefore "no-relapse" is considered as survival for our statistical study in this work.

[5] All patients in GSE2034 are LN-Negative.

[6] Website: http://toppgene.cchmc.org, visited on April 26, 2012, "correction" set to be "Bonferroni".

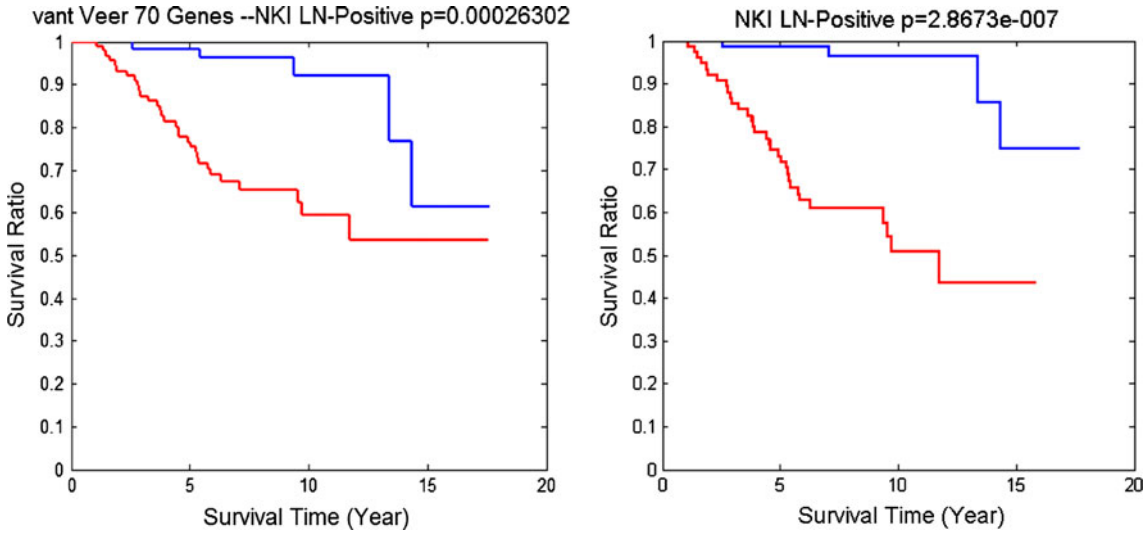

Fig. 5 *Left* the Kaplan–Meier curves for the NKI LN-positive patients which are separated into two subgroups by the 70-gene signature. *Right* the Kaplan–Meier curves for the same group of patients which are separated into two subgroups by applying *K*-means on the gene-expressions of {MYBL2, MCM6, MCM2, FOXM1, CCNB2, CDC20, UBE2C, SPAG5, EZH2, BUB1B, DLGAP5, ZWINT, TRIP13, NDC80, PKMYT1, KIF11, EXO1, TTK, MELK, CENPE, GINS1, KIF14, TPX2, PRC1, ASF1B, TACC3, KIF4A, CEP55, DTL, HJURP, KIF20A, ORC6L, PBK, KIF15, ASPM, CDCA8, KIF18B, RACGAP1} (this gene list belongs to Network 3 in Table 2)

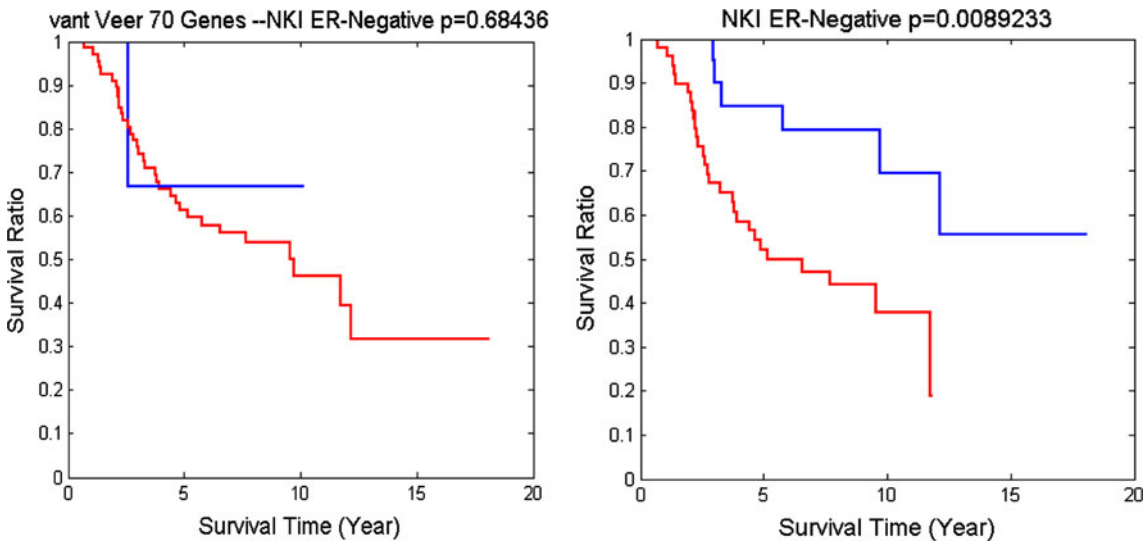

Fig. 6 *Left* the Kaplan–Meier curves for the NKI ER-negative patients which are separated into two subgroups by the 70-gene signature. *Right* the Kaplan–Meier curves for the same group of patients which are separated into two subgroups by applying *K*-means on the gene-expressions of {CAPNS1, PPP2R1A, PPP1CA, RHOC, KDELR1, PLD3, ASNA1, CYBA, UBE2M, EIF2S3, BSG, TXLNA, MGAT4B, SH3BGRL3} (this gene list belongs to Network 4 in Table 2)

We use "ForceAtlas 2"[7], a built-in layout of Gephi 0.8.1-beta, to visualize these networks and their interactions. Since ForceAtlas 2 simulates repulsion and attraction among network vertices based on topology and edge information, it is a helpful layout tool to study network interactions in our case. Figure 8 shows the visualization result under parameters ($\varepsilon = 0.2$, $\epsilon = 0.01$). From Fig. 8 we can observe that network 2 (cyan) is highly connected both within itself and with other networks. This suggests

[7] Described on Gephi official blog: https://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout/, visited April 29, 2012.

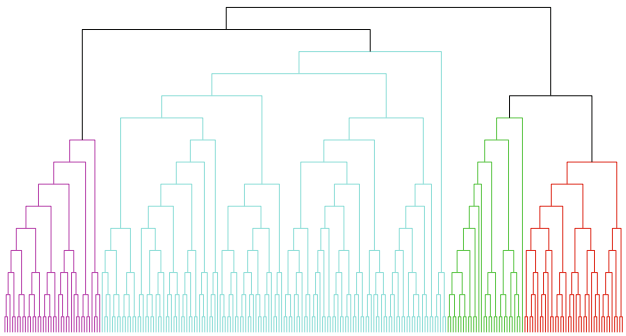

Fig. 7 The dendrogram of SINGLEMERGE on 242 summarized network patterns

Fig. 8 The four network patterns visualized under parameters ($\varepsilon = 0.2$, $\epsilon = 0.01$)

Fig. 9 The four network patterns visualized under parameters ($\varepsilon = 0.3$, $\epsilon = 0.01$)

that immune system process could be a central biological process that plays a very important role in breast carcinoma. Interestingly, although network 1 has the highest density value (determined by the summed edge weight and its size) in Table 2, it looks the sparsest in Fig. 8. Our intuition is further confirmed by the visualization shown in Fig. 9 ($\varepsilon = 0.3$, $\epsilon = 0.01$). This suggests that compared with the other three networks, the gene network enriched by the extracellular matrix is more unevenly correlated within itself.

6 Conclusion and future work

We showed that several variants of the network pattern summarization problem such as Problems 1, 2, and 3 are all NP-hard. To solve the problem efficiently, we proposed an algorithmic framework which is based on optimal solutions of a subproblem with respect to different objective functions. We implemented novel algorithms for two of these objective functions with an efficient merge operation. We evaluated the performance of these algorithms through extensive empirical study. The application study demonstrates that our methods are suitable for multipurpose network analysis in biomedical informatics. The proposed algorithms are generic and can be applied to various biomedical datasets which can be modeled as weighted or unweighted graphs or bipartite graphs. In the future, we plan to use our algorithms in many biomedical applications and perform in-depth analysis on the biomedical significance of summarized patterns.

Table 2 Characters of the four networks

	Density	# of unique genes	# of network patterns	Toppgene enrichment (p value)
Network 1 (red)	0.509707	67	39	CC: extracellular matrix (1.113E−17)
Network 2 (cyan)	0.495775	153	135	BP: immune system process (5.433E−62)
Network 3 (green)	0.474597	80	38	BP: cell cycle process (9.662E−40)
Network 4 (magenta)	0.44619	66	30	BP: translational termination (2.368E−20), CC: cytosolic ribosome (1.211E−22)

"# of unique genes" is the number of genes obtained by merging all network patterns associated with this network (cluster)

CC cellular component, *BP* biological process

Acknowledgments This work was supported by the National Science Foundation under Grant #1019343 to the Computing Research Association for the CIFellows Project, and by the National Cancer Institute under Grant NCI R01CA141090.

References

Abello J, Resende MGC, Sudarsky S (2002) Massive quasi-clique detection. In: LATIN, pp 598–612

Almansoori W, Gao S, Jarada TN, Elsheikh AM, Murshed AN, Jida J, Alhajj R, Rokne J (2012) Link prediction and classification in social networks and its application in healthcare and systems biology. Netw Model Anal Health Inform Bioinforma. doi: 10.1007/s13721-012-0005-7

Bastian M, Heymann S, Gephi MJ (2009) An open source software for exploring and manipulating networks. In: ICWSM

Berkhin P (2006) A survey of clustering data mining techniques. Grouping Multidimensional Data

Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Commun ACM 16(9):575–577

Burdick D, Calimlim M, Flannick J, Gehrke J, Yiu T (2005) Mafia: a maximal frequent itemset algorithm. IEEE Trans Knowl Data Eng 17(11):1490–1504

Carroll JS, Meyer CA, Song J, Li W et al (2006) Genome-wide analysis of estrogen receptor binding sites. Nat Genet 38(11):1289–1297

Defays D (1977) An efficient algorithm for a complete link method. Comput J 20(4):364–366

Desmedt C, Haibe-Kains B, Wirapati P et al (2008) Biological processes associated with breast cancer clinical outcome depend on the molecular subtypes. Clin Cancer Res 14(16):5158–5165

Du N, Wang B, Wu B, Wang Y (2008) Overlapping community detection in bipartite networks. In: Web intelligence, pp 176–179

Duan R, Pettie S, Su H-H (2011) Scaling algorithms for approximate and exact maximum weight matching

Edmonds J (1965) Paths, trees, and flowers. Can J Math 17(3):449–467

Geerts F, Goethals B, Mielikäinen T (2004) Tiling databases. In: Discovery science, pp 278–289

Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques, 3rd edn. Elsevier

Hartigan JA (1972) Direct clustering of a data matrix. J Am Stat Assoc 67(337):123–129

Jin R, Hong H, Wang H, Ruan N, Xiang Y (2010) Computing label-constraint reachability in graph databases. In: SIGMOD conference, pp 123–134

Jin R, Ruan N, Xiang Y, Lee VE (2012) A highway-centric labeling approach for answering distance queries on large sparse graphs. In: SIGMOD conference

Jin R, Xiang Y, Hong H, Huang K (2010) Block interaction: a generative summarization scheme for frequent patterns. In: UP '10 proceedings of the ACM SIGKDD workshop on useful patterns, pp 55–64

Johnson DS, Yannakakis M, Papadimitriou CH (1988) On generating all maximal independent sets. Inf Process Lett 27(3):119–123

Karp R (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum Press, New York, pp 85–103

Kutalik Z, Beckmann JS, Bergmann S (2008) A modular approach for integrative analysis of large-scale gene-expression and drug-response data. Nat Biotechnol 26(5):531–539

Li J, Liu Y, Gao H (2011) Efficient algorithms for summarizing graph patterns. IEEE Trans Knowl Data Eng 23(9):1388–1405

Li J, Liu G, Li H, Wong L (2007) Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: a one-to-one correspondence and mining algorithms. IEEE Trans Knowl Data Eng 19(12):1625–1637

Li J, Sim K, Liu G, Wong L (2008) Maximal quasi-bicliques with balanced noise tolerance: concepts and co-clustering applications. In: SDM, pp 72–83

Lucchese C, Orlando S, Perego R (2010) A generative pattern model for mining binary datasets. In: SAC, pp 1109–1110

Makino K, Uno T (2004) New algorithms for enumerating all maximal cliques. In: SWAT, pp 260–272

Micali S, Vazirani VV (1980) An $\mathcal{O}(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. In: FOCS, pp 17–27

Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. IEEE Trans Knowl Data Eng 20(10):1348–1362

Mirkin B (1996) Mathematical classification and clustering. Kluwer Academic Publishers, Dordrecht

Moon JW, Moser L (1965) On cliques in graphs. Israel J Math 3(1):23–28

Mucha M, Sankowski P (2004) Maximum matchings via gaussian elimination. In: FOCS, pp 248–255

Mushlin RA, Gallagher S, Kershenbaum A, Rebbeck TR (2009) Clique-finding for heterogeneity and multidimensionality in biomarker epidemiology research: the chamber algorithm. PloS one 4(3):4862

Ou Y, Zhang C (2007) A new multimembership clustering method. J Ind Manag Optim 3(4):619

Ravetti MG, Moscato P (2008) Identification of a 5-protein biomarker molecular signature for predicting alzheimer's disease. PLoS One 3(9):3111

Reyal F, van Vliet MH, Armstrong NJ et al (2008) A comprehensive analysis of prognostic signatures reveals the high predictive capacity of the proliferation, immune response and rna splicing modules in breast cancer. Breast Cancer Res 10(6):R93

Robinson PN, Köhler S, Bauer S, Seelow D, Horn D, Mundlos S (2008) The human phenotype ontology: a tool for annotating and analyzing human hereditary disease. Am J Hum Genet 83(5):610–615

Robinson PN, Mundlos S (2010) The human phenotype ontology. Clin Genet 77(6):525–534

Seidman SB (1983) Network structure and minimum degree* 1. Social Netw 5(3):269–287

Seidman SB, Foster BL (1978) A graph-theoretic generalization of the clique concept. J Math Sociol 6(1):139–154

Slink RS (1973) An optimally efficient algorithm for the single-link cluster method. Comput J 16(1):30–34

Stoer M, Wagner F (1997) A simple min-cut algorithm. J ACM 44(4):585–591

Tsukiyama S, Ide M, Ariyoshi H, Shirakawa I (1977) A new algorithm for generating all the maximal independent sets. SIAM J Comput 6:505

Uppalapati P, Xiang Y, Huang K (2010) Predicting prognostic markers for glioma using gene co-expression network analysis. In: Proceedings of the first ACM international conference on bioinformatics and computational biology, pp 546–551

van de Vijver MJ, He YD, van 't Veer LJ et al (2002) A gene-expression signature as a predictor of survival in breast cancer. New Engl J Med 347(25):1999–2009

van't Veer LJ, Dai H, van de Vijver MJ et al (2002) Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871):530–536

Wang Y, Klijn PGM, Zhang Y et al (2005) Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. Lancet 365(9460):671–679

Wirapati P, Sotiriou C, Kunkel S et al (2008) Meta-analysis of gene expression profiles in breast cancer: toward a unified

understanding of breast cancer subtyping and prognosis signatures. Breast Cancer Res 10(4):R65

Xiang Y, Zhang CQ, Huang K (2012) Predicting glioblastoma prognosis networks using weighted gene co-expression network analysis on tcga data. BMC Bioinform 13(Suppl 2):S12

Xiang Y, Jin R, Fuhry D, Dragan FF (2011) Summarizing transactional databases with overlapped hyperrectangles. Data Min Knowl Discov 23(2):215–251

Xiang Y, Payne P, Huang K (2012) Transactional database transformation and its application in prioritizing human disease genes. IEEE/ACM Trans Comput Biol Bioinform 9(1):294–304

Zhang J, Xiang Y, Ding L et al (2010) Using gene co-expression network analysis to predict biomarkers for chronic lymphocytic leukemia. BMC Bioinform 11(Suppl 9):S5