Finding Itemset-Sharing Patterns in a Large Itemset-Associated Graph

Mutsumi Fukuzaki[1], Mio Seki[1], Hisashi Kashima[2], and Jun Sese[1]

[1] Dept. of Computer Science, Ochanomizu Univ. 2-1-1 Otsuka, Bunkyo, Tokyo, Japan
[2] Dept. of Math. Informatics, Univ. of Tokyo. 7-3-1 Hongo, Bunkyo, Tokyo, Japan

Abstract. Itemset mining and graph mining have attracted considerable attention in the field of data mining, since they have many important applications in various areas such as biology, marketing, and social network analysis. However, most existing studies focus only on either itemset mining or graph mining, and only a few studies have addressed a combination of both. In this paper, we introduce a new problem which we call *itemset-sharing subgraph (ISS) set enumeration*, where the task is to find sets of subgraphs with common itemsets in a large graph in which each vertex has an associated itemset. The problem has various interesting potential applications such as in side-effect analysis in drug discovery and the analysis of the influence of word-of-mouth communication in marketing in social networks. We propose an efficient algorithm *ROBIN* for finding ISS sets in such graph; this algorithm enumerates connected subgraphs having common itemsets and finds their combinations. Experiments using a synthetic network verify that our method can efficiently process networks with more than one million edges. Experiments using a real biological network show that our algorithm can find biologically interesting patterns. We also apply ROBIN to a citation network and find successful collaborative research works.

1 Introduction

Since the origin of the field of data mining, frequent pattern mining has been one of the main topics of interests for researchers in this field. These researchers initially worked on itemset patterns [1, 2] in the context of market basket analysis; these studies were later extended to event sequence patterns [3]. Recently, graph-structured data have attracted considerable attention [4–6] because graph pattern mining can be applied to many interesting application areas such as biological networks, social networks, and the Web. While itemset mining seeks frequent combinations of items in a set of tuples, graph mining seeks frequent subgraphs in a set of graphs. Most of the prior studies have addressed only one type of their patterns, and only a few studies have considered combinatorial mining of two types of data structures [7–9].

In this paper, we consider a new combinatorial mining problem of itemsets and subgraphs, which we call the *itemset-sharing subgraph (ISS) set* enumeration problem. Let us assume that we have a graph in which each vertex is associated with an itemset (Fig. 1(A)). We refer to this graph as an *itemset-associated graph*. Our

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 147–159, 2010.

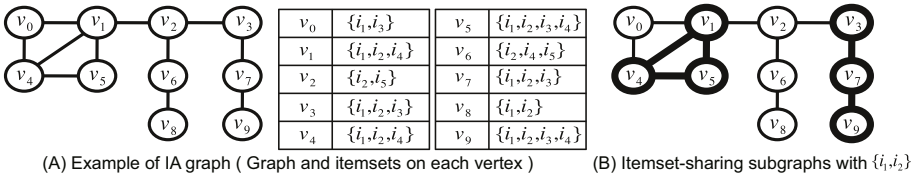


Fig. 1. An example of itemset-sharing subgraph

task is to enumerate the patterns that we call the ISS set, which is a set of large subgraphs in which all vertices share a large common itemset. The ISS set shown in Fig. 1(B) consists of two subgraphs depicted by the bold lines. All vertices in the ISS set share the itemset $\{i_1, i_2\}$. Although the subgraph consisting of only v_8 also shares the itemset, it is not included in the ISS set since it is quite small. Similarly, although the subgraph consisting of $\{v_0, v_1, v_4, v_5\}$ shares $\{i_1\}$, it is not sufficient to be an ISS set since the shared itemset is very small. The ISS set enumeration problem differs from other graph mining problems in the sense that the subgraphs included in an ISS set need not be identical.

We now illustrate how ISS sets are used in drug discovery. Let us consider the metabolic pathway networks, which describe biochemical processes occurring within a cell. A pathway is represented as a graph, where vertices denote genes and chemical compounds and edges denote chemical reactions among the genes and the compounds. The pathway networks play a considerably important role in drug discovery, because by finding a sub-pathway that is closely related to a disease, we can determine the target genes or chemical compounds on which the drug candidate should act. However, the drug candidate can affect several different pathways simultaneously, which may lead to unexpected outcomes. Such phenomena are called side effects. We would like to not only find the drug targets but also predict the side effects that may be caused by the action of the drug on the targets. Taking the drugs into account, we considered a pathway network to be an itemset-associated graph (Fig. 1(A)), where each vertex (a gene or a chemical compound) is associated with an itemset that indicates the set of drugs activating the gene or the compound ($\{i_1, i_2, \ldots, i_5\}$ shown in Fig. 1(A)). In the above context, an ISS set (Fig. 1(B)) corresponds to a set of sub-pathways that share the common activation drug; this implies that there are hidden or unknown connections among the sub-pathways and that the drugs designed to target genes or compounds in one sub-pathway might also act on the other sub-pathways. In Fig. 1(B), the sub-pathway consisting of only v_8 is also activated by the drugs $\{i_1, i_2\}$. However, this sub-pathway is very small, which implies that the activation would result from accidental observations. Large sub-pathways are more reliable and indicate that the side effects are more serious because these effects cover a wide range of pathway networks. Similarly, we expect that as the size of the set of common activation drugs increases, the possibility of the occurrence of side effects increases. Therefore, networks that consist of the large sub-pathways with a large set of activation drugs are important clues in predicting side effects for drug discovery and biological experimental design.

Let us now consider a marketing scenario in social network analysis. In social networks, vertices are considered to be participants, and edges are considered to be the relationships between these participants (e.g., friendships). Let us assume that each participant (vertex) is associated with the items that he or she has bought. The network can be considered to be an itemset-associated graph, and the subgraphs with large common itemsets can be regarded as underlying communities. Further, in a social network, common itemsets shared by many communities are considered as the (sets of) products that can be easily marketed through word-of-mouth communication; hence, the products with common features would be suitable for social marketing.

In order to solve the ISS set enumeration problem, one approach is to use an itemset mining technique [1, 2] to obtain all the frequent itemsets and then checking the connections between the itemsets in the networks. However, the itemset mining in real dataset are very long computation time because the supports (frequencies) of the items included in the ISS sets are usually low. To overcome the computation time problem, we propose an efficient algorithm called *ROBIN*. The ROBIN algorithm consists of two stages; it enumerates subgraphs that are larger than a specified threshold value at the first stage, and then it combines them at the second stage. By introducing effective pruning techniques in both the stages, we can enumerate the graphs very efficiently.

Finally, the efficiency of the proposed algorithm is shown in the experiments by using a synthetic dataset. ROBIN can solve problems with more than 100K vertices and 1,000K edges for about a half hour. In the experiments using a real biological network, we discover hidden connections in metabolic pathways; this suggests the practical utility of ROBIN in the context of drug discovery. Furthermore, by applying ROBIN to a citation network, we find interesting patterns indicating successful collaborative works containing well-known database research topics. In both of the real dataset experiments, we show that execution time of ROBIN are faster than that of the method which first enumerates the itemset and then checks their connectivity.

Our contributions are summarized as follows:

1. We introduce the ISS set enumeration problem, which has sound potential applications in various real-world problems including finding side effects in drug discovery and estimating effects of word-of-mouth communication in marketing in social networks.
2. We propose a very efficient algorithm called ROBIN to solve the ISS set enumeration problem; In the ISS enumeration stage, we develop a novel pruning technique using hash tables called a *visited itemset table*, which stores itemsets shared by generated subgraphs, and allows us to enumerate ISSes very efficiently. In the ISS combination stage, we propose an efficient algorithm for finding ISS sets using an extend depth-first search (DFS) tree called *ISS tree*, which enables us to generate combinations of ISSes without unnecessary checking of graph inclusion.
3. We conduct experiments using two real-world network data, a biological network and a citation network, and show scenarios where the ISS set

enumeration problem is useful. The results also show that ROBIN is much faster than the itemset-enumeration approach.

2 Itemset-Sharing Subgraph (ISS) Set Enumeration Problem

In this section, we introduce a novel data mining problem for analyzing itemset-associated graphs, which we refer to as the ISS set enumeration problem.

Let G be an undirected[1], unlabelled, and unweighted graph with an itemset on each vertex. We refer to this graph as an *itemset-associated graph (IA graph)*. Let $V(G)$, $E(G)$ and $\mathcal{I}(G)$ respectively signify a set of the vertices in G, a set of edges in G and a set of itemsets on vertices in G. Note that the size of graph G is given as the number of edges, i.e., $|G| = |E(G)|$.

We next define subgraphs whose vertices share itemsets.

Definition 1. *(Shared Itemset) Let G' be a connected subgraph of an IA graph G, where G' is also an IA graph. We define $I(G')$ as $I(G') = \bigcap_{v \in V(G')} I(v)$, and refer to $I(G')$ as a shared itemset of G'.*

Among the subgraphs having a shared itemset, we focus on an important subset, which cannot be expanded while retaining the currently shared itemsets.

Definition 2. *(Itemset-Sharing Subgraph (ISS)) We call G' an itemset-sharing subgraph (ISS) with $I(G')$ if $I(G') \neq \phi$ and $I(v) \not\supseteq I(G')$ for any vertex v in the neighbor vertices of G'.*

Note that the itemset shared by an ISS is defined without reference to its edges.

Now, we define the sets of ISSes that we want to enumerate in our task. As described in Section 1, sets of ISSes are useful in the context of drug discovery and marketing in social networks.

Definition 3. *(ISS Set) Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ be a set of ISSes, where each G_i is an ISS. Define $I(\mathcal{G})$ as $I(\mathcal{G}) = \bigcap_{G \in \mathcal{G}} I(G)$. Note that $I(\mathcal{G}) = \bigcap_{G \in \mathcal{G}} \bigcap_{v \in G} I(v)$. We call $\mathcal{G}$ an ISS set with $I(\mathcal{G})$, if all of the following conditions are satisfied: (1) $V(G_i) \cap V(G_j) = \phi$ for any G_i and G_j $(i \neq j)$ in $\mathcal{G}$. (2) $I(v) \not\supseteq I(\mathcal{G})$ for any vertex v in the neighbor vertices of $G' \in \mathcal{G}$. (3) $|G_i| \geq \theta_S$, where θ_S is a user-specified value. (4) No ISS G' with $I(\mathcal{G})$ exists except in $\mathcal{G}$.*

The first two conditions are an extension of the definition of ISS for dealing with multiple ISSes. The third condition gives the minimum size of the obtained ISSes because larger ISSes are of greater interest to us. The last condition ensures the maximality of the found ISS sets. Let $|\mathcal{G}|$ indicate the number of disconnected components of G, and hence, $|\mathcal{G}| = n$.

Finally, we define our new data mining problem where the task is to enumerate all ISS sets from a given IA graph.

[1] Although for simplicity, we assume that G is undirected, our method can be used for directed graphs in the same manner.

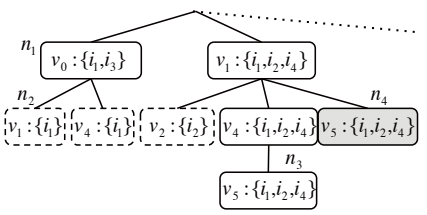
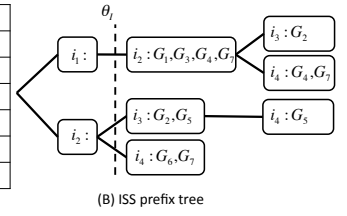


Fig. 2. The DFS itemset tree for Fig. 1(A)

$\{i_1, i_2\}$	$\{G_1\}$
$\{i_1, i_2, i_3\}$	$\{G_2\}$
$\{i_1, i_2\}$	$\{G_3\}$
$\{i_1, i_2, i_4\}$	$\{G_4\}$
$\{i_2, i_3, i_4\}$	$\{G_5\}$
$\{i_2, i_4\}$	$\{G_6\}$
$\{i_1, i_2, i_4\}$	$\{G_7\}$

(A) ISS table



(B) ISS prefix tree

Fig. 3. An ISS table and an itemset-graph prefix tree

Definition 4. *(ISS Set Enumeration Problem) Given an IA graph and user-specified values θ_S, θ_I and θ_F, from the IA graph, enumerate all ISS sets* $\mathbf{G}$ *satisfying $|\mathcal{G}| \geq \theta_F$, $|I(\mathcal{G})| \geq \theta_I$ for any ISS set $\mathcal{G} \in \mathbf{G}$, and $|G| \geq \theta_S$ for any ISS $G \in \mathcal{G}$ in any ISS set $\mathcal{G} \in \mathbf{G}$.*

3 Proposed Method

In this section, we propose an efficient algorithm called *ROBIN* (RelatiOn Between Items and Networks) for solving the ISS set enumeration problem. To solve the problem, one strategy first enumerates all the itemsets such as Apriori [1] and FP-trees [2], and then check the connectivity between the itemsets. The other strategy first enumerates the subgraphs, and then check the conditions of the subgraphs. Here, we use the latter method. We will show that the computing time of the former method requires longer than the latter method using real dataset in Section 4.

Robin consists of two stages. In the first stage, we enumerate all the ISSes efficiently by introducing *DFS itemset tree* and *visited itemset table*. Their details are described in Section 3.1. In the second stage, we generate ISS sets by combining the ISSes according to Section 3.2. In order to enumerate the ISS sets efficiently, we introduce an *ISS prefix tree* that contains the prefix of itemsets and their associated ISSes.

3.1 ISS Enumeration

In the first stage of the ROBIN, we enumerate ISSes from the given IA graph. We introduce efficient techniques for the enumeration of ISSes in this section. In the second stage of ROBIN, the obtained ISSes are combined with the generated ISS sets (Section 3.2).

We use a depth-first search (DFS) tree for enumerating ISSes $\mathcal{G}$ where $|G| \geq \theta_S$ and $|I(G)| \geq \theta_I$ for $G \in \mathcal{G}$. Each node of the tree contains a vertex and an itemset related to the path from the root to the node. We denote the tree as a *DFS itemset tree*. On the DFS itemset tree, we do not need to maintain edges because $I(G)$ can be computed from vertices and their itemsets.

The generation of the subgraphs itself is considered to be a simplified version of the DFS lexicographic order used in the gSpan algorithm [6], and hence, this

DFS itemset tree can avoid duplicate generation of identical graphs. Fig. 2 shows the DFS itemset tree for the IA graph in Fig. 1(A). Each node in the DFS itemset tree contains a vertex and an itemset. The vertices included in the path from the root to the tree node represent the vertices of the subgraph.

Thanks to the following monotonic property of ISS about itemset size, we can prune subtrees in the DFS itemset tree, which dramatically reduces the search space.

Property 1. *Let us denote two ISSes by G' and G'', and let $V(G') \supset V(G'')$. Then, $I(G') \subseteq I(G'')$ holds.*

The tree nodes indicated by dotted boxes in Fig. 2 can be pruned by using this property when $\theta_I = 2$.

The next theorem allows us to avoid generating subgraphs that have the same vertices as those of already generated subgraphs and have itemsets that are subsets of itemsets associated with the already generated graphs.

Theorem 1. *Let n_1 and n_2 be a pair of nodes of the DFS itemset tree, where n_1 was generated before n_2. If vertices associated with n_1 and n_2 are identical, and $I(n_1) \supseteq I(n_2)$, no ISS exists in a descendant of n_2.*

This theorem implies that if we visit one of already visited vertices and the common itemset of the current path is identical to or a subset of one of the itemsets of the previously visited vertices, we can prune the subtree rooted by the current node in the DFS itemset tree. Therefore, this property is useful for avoiding unnecessary exploration of subgraphs.

Theorem 1 prompts us to make the hash table from nodes to their related itemsets for efficient pruning of subgraphs. We call the hash table a *visited itemset table* and build it while constructing a DFS itemset tree.

Using the DFS itemset tree, we can generate all ISSes whose subgraph size is greater than θ_S and common itemset size is greater than θ_I. Fig. 3(A) illustrates the ISSes and their associated itemsets. We refer to this table as the *ISS table*. In the next section, in order to enumerate ISS sets efficiently, we introduce an efficient method of generating combinations of the ISSes.

3.2 ISS Set Combination

In this section, we introduce an efficient method for enumerating ISS sets from the ISS table created in the previous section. One simple method for computing the ISS sets is to generate combinations of all the ISSes. However, this procedure is quite redundant because different combinations of ISSes may result in the same shared itemset. Our method generates ISS sets efficiently by grouping ISSes by shared itemsets. Once we fix one itemset, an ISS set sharing the itemset is uniquely determined. Therefore, one approach to enumerating all ISS sets is to generate all itemsets that can be associated with ISS sets. For the efficient generation of the itemsets, we use the depth first search.

Definition 5. *(ISS Tree) Let T_I be a tree, each of whose node n contains itemset $I(n)$ and a set of ISSes $\mathcal{G}(n)$ which shares $I(n)$. The root of T_I contains an*

itemset including all items and a vacant set of ISSes. Let n_1 and n_2 be a pair of nodes of T_I. When n_1 is an ascendant of n_2, $I(n_1) \supseteq I(n_2)$.

We call the tree *ISS tree*. Nodes closer to the root contain larger itemset. A child of a node in the ISS tree can be generated by adding an ISS to a parent node's ISSes, and its shared itemset can be computed. Thanks to the monotonic property of itemset size, we can prune subtrees in the ISS tree.

Although we can enumerate all the combinations of ISSes by the simple DFS method, the size of ISS tree may increase considerably especially when the number of ISSes is large. In order to efficiently generate the ISS tree, we add a group of ISSes sharing an itemset to ISSes in its parent node.

We here divide ISS sets into two types: explicit ISS sets and implicit ISS sets. Explicit ISS sets are associated with itemset appeared in an ISS table, while implicit ISS sets are associated with itemset which is a subset of itemsets appeared in the ISS table. We first generate explicit ISS sets quickly using prefix tree structure, and then produce implicit ISS sets by the combinations of explicit ISS sets.

Definition 6. *(Explicit and Implicit ISS Set) Let $\mathcal{G}$ be all the ISSes in an ISS table, and $\mathcal{I}(\mathcal{G})$ be itemsets associated with ISSes in $\mathcal{G}$. Let $\mathcal{G}_I$ be an ISS set with I. When $I \in \mathcal{I}(\mathcal{G})$, we call $\mathcal{G}_I$ an explicit ISS set; otherwise we call $\mathcal{G}_I$ an implicit ISS set.*

Basis of the above definition, any ISS set can be classified as explicit or implicit.

For the efficient generation of all ISS sets, we first extract all of the explicit ISS sets, and then generate ISS sets by removing overlapping ISSes. The following theorem guarantees us to generate all the ISS sets.

Theorem 2. *Let $\mathcal{G}_I$ be $\mathcal{G}_C - \{G \mid G \subseteq G'$ where $G, G' \in \mathcal{G}_c\}$. Then, $\mathcal{G}_I$ is an ISS set with I.*

This theorem allows us to generate the explicit ISS sets with I. All the explicit ISS sets can be generated by computing $\mathcal{G}_I$ for all the itemsets in the ISS table. However, the procedure requires many checks related to the inclusion relations among graphs. Here, we introduce an efficient way to generate explicit ISS sets by using a prefix tree representing itemsets.

Definition 7. *(ISS Prefix Tree) Let T_P be a tree, each of whose nodes n contains an item i_n and an ISS set $\mathcal{G}(n)$. Let denote two nodes in T_P by n_1 and where n_1 is an ascendant of n_2. Then, $i_{n_1} < i_{n_2}$ holds. Any itemset I in the ISS table is represented by a path in T_P. The ISS set in node n in T_P shares an itemset represented by a corresponding path from the root to n.*

We call the tree an *ISS prefix tree*. Using the ISS prefix tree, we represent all the associations between the itemsets contained in the ISS table and the ISSes. Fig. 3(B) represents the ISS prefix tree of Fig. 3(A). We put no ISSes to nodes whose depth is less than θ_I because none of the nodes generate ISS sets. Thanks to this prefix tree structure, we can accelerate the finding of the associations between itemsets and explicit ISS sets.

Table 1. Parameters for ROBIN

Name	Description	Default
Parameters for ISS sets		
$\|I\|$	Size of the itemset shared by an ISS set	10
S	ISS size	7
F	Size of ISS set	5
P	Number of ISS sets patterns	10
Q	Number of ISSes not included in ISS sets	$\|V\|/30$

Name	Description	Default
Parameters for graphs and itemsets		
$\|V\|$	Number of vertices	15,000
$\|E\|$	Number of edges	$10 \times \|V\|$
N	Number of items	100
$\|T\|$	Avg. size of itemsets in a vertex	10
User-specified thresholds		
θ_S	Minimum ISS size	$S-1$
θ_I	Minimum shared itemset size	$\|I\|-1$
θ_F	Minimum size of ISS set	$P-1$

We here generate itemsets shared by implicit ISS sets by using the combination of two explicit ISS sets. From the itemset, we generate ISS sets by using the ISS prefix tree. The following theorem guarantees that the combinations can enumerate all of the implicit ISS sets.

Theorem 3. *Any itemset shared by an implicit ISS set is represented by the intersection of the itemsets shared by some of the explicit ISS sets.*

On the basis of this theorem, we can generate implicit ISS sets by using combinations of the itemsets shared by explicit ISS sets. Therefore, we generate a DFS tree each of whose nodes contains an itemset and an ISS set. We can prune the branches in the ISS tree from the monotonic property in Definition 5. Furthermore, the following property substantially reduces the search space.

Property 2. *Let node n contain an itemset $I(n)$ and an ISS set $\mathcal{G}(n)$. If $I(n)$ and an itemset I' of an existing node are identical, we need not traverse the branch rooted by n.*

To use these pruning techniques, we need not calculate inclusion relations between graphs in ISSes.

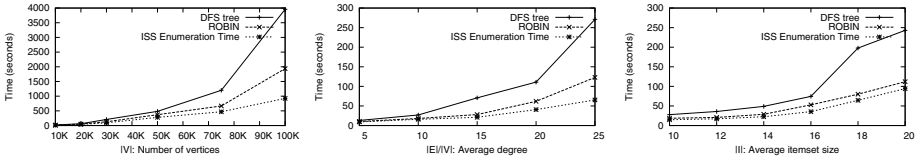
4 Experiments

In this section, we present the results of our experiments using a synthetic dataset and two real-world datasets.

4.1 Results for a Synthetic Network

We generated a synthetic network dataset in order to evaluate the performance of the ROBIN algorithm. The parameters for ROBIN and their default values are presented in Table 1. We generated synthetic datasets having $|V|$ vertices and $|E|$ edges. Each dataset includes P ISS sets whose shared itemset size is $|I|$ and size of ISS set is F. Moreover, we add the fake itemsets whose size is $1.7 \times |I|$. The detail procedure is omitted due to the space limitation. All experiments were performed using a 3.2 GHz AMD® Opteron™ machine with 1 GB memory running on Linux kernel 2.6. We implemented ROBIN in Java™ 5.

We investigated the efficiency of the ROBIN algorithm by using the synthetic network data and varying the size of the network, the average size of itemsets, and the parameters for ROBIN.



(A) Graph size (B) Average degree (C) Average itemset size

Fig. 4. Performance study with respect to overall graph size

In order to investigate the efficiency of enumerating combinations of the ISSes in ROBIN, we measured the execution times in the case of ROBIN (labeled as "ROBIN") and the times in the case of the algorithm in which we replace the ISS tree and the groups of ISSes generated by ISS prefix tree with the standard DFS tree by adding single ISS to its parent node to generate combinations of ISSes (labeled as "DFS tree"). We also show the execution times required for enumerating ISSes (Section 3.1) because these times are independent of the approach we choose. The differences between the execution times of ROBIN and ISS enumeration and those of the DFS tree approach and ISS enumeration indicate the computing time required to enumerate the combinations of ISSes.

Fig. 4(A) presents the execution times by varying the number of nodes in the network. This figure depicts that our method is more scalable than the alternative approach. The largest network in this experiment has 100K vertices and one million edges. The execution times increase quadratically with respect to the increase in the number of vertices. In particular, the larger the graph becomes, the larger is the execution time difference between the two approaches. Because the support (ratio of the number of vertices in ISS sets to the total number of vertices) was $F \times S/|V| = 0.0023$, when the values were set to the default values, it is difficult to find itemset patterns using the Apriori algorithm [1] and the FP-trees [2]. In contrast, ROBIN can work with such a low support and can still find important itemsets because it uses subgraphs that connect the itemsets.

Fig. 4(B) shows the execution times by varying the number of degrees in the network. In general, the execution time increases rapidly according to the density of the graph, because we need to check many neighbor vertices. However, our result demonstrates that the execution time of ROBIN increases rather gradually. We can observe that as the degree of the graph increases, the difference between the execution times of the two methods increases. This observation verifies the computational efficiency of ROBIN.

Next, we investigate the performance of ROBIN by varying the itemset size shared in ISSes. The dependence of the execution time on the itemset size is shown in Fig. 4(C). As shown in the figure, the average itemset size is not significant impact to ROBIN. Note that the algorithms succeeded in finding ISS sets with relatively large itemsets (more than 10 items). This result is in contrast to that of the existing studies on mining long patterns [10], in which finding low-frequency itemset patterns efficiently is difficult.

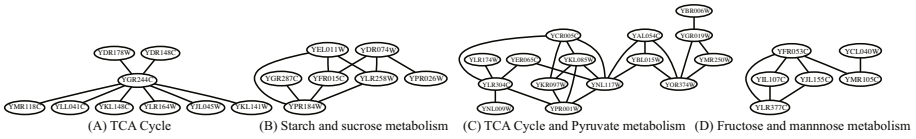


Fig. 5. One of the ISS sets found in the real biological network. All of the vertices sharing five stress conditions.

Table 2. All of the ISS sets found in the DBLP citation network

No.	Authors	# of ISSes	# of refs.	# of papers
1	Rajeev Rastogi, Abraham Silberschatz	3	30	23
2	Amr El Abbadi, Divyakant Agrawal	2	40	25
3	Riccardo Torlone, Paolo Atzeni	2	13	11
4	Marc Gyssens, Dirk Van Gucht	2	12	11
5	Ling Liu, Calton Pu	2	11	11
6	Raghu Ramakrishnan, Praveen Seshadri	2	11	11

4.2 Results for a Biological Network

We applied ROBIN to a real metabolic pathway dataset with 6,152 vertices and 3,318 edges; here, the vertices and edges represent genes and chemical interactions, respectively. The dataset was obtained under 173 different stressed conditions [11] by using yeast microarrays. Each of the conditions causes stimuli to cells, and finding stimuli associated with treatments of diseases is a good starting point for development of new drugs. Therefore, we used the set of the conditions as the items. In biological systems, highly expressed genes play an important role within the cells. Therefore, we converted the quantitative values into Boolean values using a threshold t. We set the parameters as $t = 1.5$, $\theta_S = 7$, $\theta_I = 5$ and $\theta_F = 4$. The average itemset size in the dataset was 4.78, and its execution time was 35.9 seconds. We extracted eight ISS sets in total. One of the ISS sets depicted in Fig. 5 was associated with the conditions of 8 hours, 10 hours, 1 day, 2 days and 3 days grown under YPD condition at 30 degree Celsius; all of these conditions are high-nutrition and high-temperature conditions. Consequently, our algorithm could extract biologically consistent conditions automatically. The four connected graphs were associated with four biological metabolic pathways. Some genes in Fig. 5(C) are known as the activator of the TCA cycle including Fig. 5(A). Also associated pathways with Fig 5(D) are related to TCA cycle, and hence, the relationship between these two ISS sets is biologically reasonable.

4.3 Results for a Citation Network

We applied ROBIN to a citation network consisting of academic papers to demonstrate that ROBIN can extract successful collaborative researches automatically.

We create a citation network from the DBLP dataset [12], which is a snapshot of the DBLP as of April 12, 2006. Each vertex in the network corresponds to a

paper and is associated with an itemset representing the author of the paper. Each edge indicates a citation. The DBLP network has 22,178 vertices (papers), 112,304 edges (citations), and 16,638 items (authors). All papers have at least one author and one reference. The average number of authors for a paper is 2.29. We set the parameters for ROBIN as $\theta_I = 2$, $\theta_S = 10$, and $\theta_F = 2$.

Table 2 summarizes the six ISS sets found by ROBIN. The columns represent the ISS sets number, co-authors, numbers of disconnected networks, number of references in the ISS set, and number of papers in the ISSes. For example, the ISS set No.1 consists of three different ISSes, and the ISS set contains 23 papers and 30 references.

The research topics corresponding to the three ISSes in the ISS set No.1 are *multi-databases*, *video-on-demand storage*, and *main memory databases*. This result implies that Rajeev Rastogi and Abraham Silberschatz have successfully collaborated on three different research topics.

5 Related Work

In recent years, graph mining has received increasing interest from researchers. Frequent subgraph discovery methods [4–6, 13] enable us to enumerate all frequent common-structured subgraphs in a graph database. In this study, we are not concerned about the structure of the subgraph, and the existing methods cannot handle itemsets on subgraphs, hence we cannot apply the existing methods to our problem directly.

For the discovery of ISS sets, one straightforward approach might be to use the frequent pattern or closed itemset mining methods [1, 2, 14, 15] and then to check the connection among the found itemsets in the networks. However, in Section 4, we demonstrated that this approach is not efficient and requires huge amount of memory, which implies the effectiveness of ROBIN's approach which enumerates all subgraphs first.

The combinatorial mining of networks with numerical vectors has been studied in constrained clustering [7, 9]. The studies attempt to find the simultaneous clustering of the vertices in a network and the numerical vectors associated with the vertices. One significant difference between our problem and these problems is that the associated features on every vertex are discrete values in our problem. This property makes it difficult to apply the constrained clustering methods to our problem. MATISSE [16] and CoPaM [17] study the combinatorial mining of networks with feature vectors. Both methods find dense subgraphs whose vertices having similar features. However, we are not concerned about the density of the subgraph, and our method can find the *sparse* hub network shown in Fig. 5(A). Hashimoto *et al.* [8] proposed a combinatorial mining of sequence structured data and tree structured data. Their approach can be naturally extended to handle graph structured data, but the goal of our problem is not the enumeration of frequent subgraphs. Seki and Sese [18] introduced a problem to find the largest *connected* common pattern graph. However in the present paper, we focus on enumerating frequent *disconnected* graphs.

6 Concluding Remarks

In this paper, we introduced a novel problem called *ISS set enumeration problem*, which enumerates a set of large disconnected subgraphs in which all vertices share a large common itemset. The problem has wide application such as in side effect analysis for drug discovery and in viral-marketing effect investigations. However, it is difficult to find the graphs because of the difficulty of handling itemsets and a graph structure simultaneously. We designed a novel algorithm called *ROBIN* in order to solve this problem efficiently. Our demonstration with synthetic data showed that our algorithm is effective even in the case of a large and dense graph. Using our method, we found interesting graphs and itemsets from both a biological network and a citation network. From a biological network, we demonstrated the applicability in biological research and drug discovery. From a citation network, we found interesting patterns indicating successful collaborative works.

The problem of finding ISS sets is quite general and applicable to other itemset-associated graphs, and we are going to extend the applications to the others such as marketing in social networks and text analyses with Web links.

Acknowledgement

This work was partially supported by KAKENHI (Grant-in-Aid for Scientific Research) on Priority Areas "Systems Genomics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan. We thank Dr. Tsuyoshi Kato for fruitful discussions.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp. 487–499 (1994)
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD '00, pp. 1–12 (2000)
3. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. 1(3), 259–289 (1997)
4. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
5. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001, pp. 313–320 (2001)
6. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM '02, pp. 721 (2002)
7. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD '04, pp. 59–68 (2004)
8. Hashimoto, K., Takigawa, I., Shiga, M., Kanehisa, M., Mamitsuka, H.: Incorporating gene functions as priors in model-based clustering of microarray gene expression data. Bioinformatics 24(16), i167–i173 (2008)

9. Shiga, M., Takigawa, I., Mamitsuka, H.: A spectral clustering approach to optimally combining numerical vectors with a modular network. In: KDD '07, pp. 647–656 (2007)
10. Bayardo, R.: Efficiently mining long patterns from databases. In: SIGMOD '98, pp. 85–93 (1998)
11. Gasch, A.P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 11(12), 4241–4257 (2000)
12. Knowledge Discovery Laboratory, University of Massachusetts Amherst: The Proximity DBLP database, `http://kdl.cs.umass.edu/data/dblp/dblp-info.html`
13. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: KDD '04, pp. 581–586 (2004)
14. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in knowledge discovery and data mining, pp. 307–328 (1996)
15. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE TKDE 17(4), 462–478 (2005)
16. Ulitsky, I., Shamir, R.: Identification of functional modules using network topology and high throughput data. BMC Systems Biology 1 (2007)
17. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM '09 (2009)
18. Seki, M., Sese, J.: Identification of active biological networks and common expression conditions. In: BIBE '08 (2008)