# Cartesian Contour: A Concise Representation for a Collection of Frequent Sets

Ruoming Jin, Yang Xiang, Lin Liu
Department of Computer Science, Kent State University
Kent, OH 44242, USA
{jin,yxiang,lliu}@cs.kent.edu

## ABSTRACT

In this paper, we consider a novel scheme referred to as Cartesian contour to concisely represent the collection of frequent itemsets. Different from the existing works, this scheme provides a complete view of these itemsets by covering the entire collection of them. More interestingly, it takes a first step in deriving a generative view of the frequent pattern formulation, i.e., how a small number of patterns interact with each other and produce the complexity of frequent itemsets. We perform a theoretical investigation of the concise representation problem and link it to the biclique set cover problem and prove its NP-hardness. We develop a novel approach utilizing the technique developed in frequent itemset mining, set cover, and max k-cover to approximate the minimal biclique set cover problem. In addition, we consider several heuristic techniques to speedup the construction of Cartesian contour. The detailed experimental study demonstrates the effectiveness and efficiency of our approach.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

## General Terms

Algorithms, Theory

## Keywords

frequent itemsets, Cartesian product, set cover, concise pattern representation

## 1. INTRODUCTION

Frequent pattern discovery has initiated the acceleration of data mining research and is also becoming a core building block of data mining. Starting from frequent itemset mining [2], frequent pattern mining has grown into a rich subject in data mining, ranging over a variety of pattern domains. The powerful toolbox of frequent pattern mining not only serves as a basis for a list of other data mining techniques and more importantly, it can directly help researchers gain insight into their data [7]. Typically, domain experts try to identify the interesting frequent patterns and associate them with domain knowledge for domain specific discovery. However, a major drawback of the frequent pattern mining methodology is its often unwieldy number of discovered patterns, despite several existing efforts [11, 13, 5, 18, 8, 17]. Indeed, the collection of frequent patterns themselves are becoming the "data" which needs to be mined.

The call to better understand a large collection of frequent patterns is resonated with a list of recent efforts in pattern summarization or concise pattern representation on itemsets. Simply speaking, these methods try to help gain a global view of the whole collection of frequent itemsets. In [1], the authors propose to use $K$ itemsets as a concise representation to approximately cover the majority of the frequent itemsets. Typically, those itemsets are either maximal frequent itemsets or close to being maximal, with the condition that most of their subsets are frequent (subject to false positive constraint). Several later works apply probability inference to restore the frequency of frequent itemsets [10, 19, 15].

In this paper, we consider a novel scheme to concisely represent the collection of frequent itemsets. Different from [1], this scheme provides a complete view of these itemsets by covering the entire collection of them. More interestingly, it takes a first step in deriving a generative view of the frequent pattern formulation, i.e., how a small number of patterns interact with each other and produce the complexity of frequent itemsets. Here, the interaction is represented as the union of two frequent itemsets. Intuitively, for some *core* itemsets, if they are frequent, then all the subsets of their union are also likely to be frequent. To make this observation useful in concise pattern representation, we further introduce the *generalized Cartesian product* between any two sets of the itemsets to include all the unions between any pair of itemsets, one from each set. Each such product can be considered as a basic source for the frequent itemsets and several such products will be utilized to represent the entire collection of the frequent itemsets.

Next, we formulate our problem precisely.

### 1.1 Problem Formulation

Given a collection of frequent patterns $F_\alpha$ with $\alpha$ being the minimum support level on transactional database $D$ where $\mathcal{I}$ is the set of all items. we would like to find a concise representation to approximate the collection of frequent patterns. Our approximation scheme is as follows: Let $\mathcal{X} = X_1, X_2, \cdots, X_m$ and let $\mathcal{Y} = Y_1, Y_2, \cdots, Y_l$, where $X_i$ and $Y_j$ are the itemsets, i.e. $X_i \subseteq \mathcal{I}$ and $Y_i \subseteq \mathcal{I}$. We define a generalized Cartesian product ($\otimes$) between two sets of itemsets $\mathcal{X}$ and $\mathcal{Y}$:

$$\mathcal{X} \otimes \mathcal{Y} = \{X_1 \cup Y_1, X_1 \cup Y_2, \cdots, X_1 \cup Y_l, \cdots, X_m \cup Y_1, \cdots, X_m \cup Y_l\}$$


*KDD'09,* June 28–July 1, 2009, Paris, France.

Given this, we define the *covering sets* of a set of itemsets as

$$\mathcal{C}(\mathcal{X}) = 2^{X_1} \cup 2^{X_2} \cup \cdots \cup 2^{X_m}$$

where $2^{X_i}$ is the powerset of $X_i$. Thus, the covering set of $\mathcal{X} \otimes \mathcal{Y}$ is

$$\mathcal{C}(\mathcal{X} \otimes \mathcal{Y}) = \bigcup_{1 \le i \le m, 1 \le j \le l} 2^{X_i \cup Y_j}$$

Given this, for a collection of frequent patterns $F_\alpha$, we seek a small number of generalized Cartesian products, which can cover $F_\alpha$. Specifically, we would like to find a list of Cartesian products, i.e., $\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k$, such that

$$\bigcup_{1 \le i \le k} \mathcal{C}(\mathcal{X}_i \otimes \mathcal{Y}_i) \approx F_\alpha$$

Given this, we define the *covering cost* of a list of Cartesian product $\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k$ as the sum of covering costs from each individual Cartesian product:

$$cost(\{\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k\}) = \sum_{1 \le i \le k} cost(\mathcal{X}_i \otimes \mathcal{Y}_i)$$

The covering cost of a Cartesian product $cost(\mathcal{X}_i \otimes \mathcal{Y}_i)$ can be defined differently depending on how we want to express the conciseness. *In this paper, we define the covering cost of a Cartesian product as the sum of the sizes of its two sets,* i.e.,

$$cost(\mathcal{X}_i \otimes \mathcal{Y}_i) = |\mathcal{X}_i| + |\mathcal{Y}_i|$$

However, other cost functions are also possible. For instance, if we simply want to have the minimal number of Cartesian products, then, the cost is simply one, i.e. $cost(\mathcal{X}_i \otimes \mathcal{Y}_i) = 1$. Or if the cost needs to reflect the size of each itemset, the cost of covering by $\mathcal{X}_i \otimes \mathcal{Y}_i$ can also be defined as:

$$cost(\mathcal{X}_i \otimes \mathcal{Y}_i) = \sum_{X_j \in \mathcal{X}_i} |X_j| + \sum_{Y_j \in \mathcal{Y}_i} |Y_j|$$

The algorithms discussed in this paper can be generalized to handle these costs.

Given this, we can formally introduce the problem of constructing a concise representation of a collection of frequent itemsets.

DEFINITION 1. **(Problem 1: Exact Representation)** *Find a list of Cartesian products, $\mathcal{X}_1 \otimes \mathcal{Y}_1$, $\mathcal{X}_2 \otimes \mathcal{Y}_2$, $\cdots \mathcal{X}_k \otimes \mathcal{Y}_k$, such that $\mathcal{C}(\mathcal{X}_1 \otimes \mathcal{Y}_1)$, $\mathcal{C}(\mathcal{X}_2 \otimes \mathcal{Y}_2)$, $\ldots$, $\mathcal{C}(\mathcal{X}_k \otimes \mathcal{Y}_k)$ exactly cover $F_\alpha$:*

$$\bigcup_{1 \le i \le k} \mathcal{C}(\mathcal{X}_i \otimes \mathcal{Y}_i) = F_\alpha$$

*where the $cost(\{\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k\})$ is minimal.*

Let us consider an example of Exact Representation. For the dataset in table 1, we only need three generalized Cartesian products as shown below to exactly represent all the frequent itemsets with support $\alpha = 1/20$:

$$\{\{G\},\{I\},\{J\}\} \otimes \{\{A,B\},\{C,D\},\{E,F\}\}$$
$$\{\{M\},\{I\},\{A\}\} \otimes \{\{A,G\},\{K,L\},\{B,G\}\}$$
$$\{\{M\},\{I\}\} \otimes \{\{G,H\}\}$$

The above problem requires the covering set from the list of Cartesian products to be exactly equal to the collection of frequent itemsets. In many cases, we may want to trade the exactness with the conciseness. In other words, we can relax the condition to allow the cover to be approximate, especially, to cover some itemsets which may not be frequent itemsets, in order to reduce the covering cost. Given this, we expand the collection of frequent itemsets $F_\alpha$ to include itemsets which are nearly frequent and/or very similar to a frequent itemset. We denote the expanded collection as $\tilde{F}_\alpha$. There are several different choices for such an expanded collection. For instance, we may include the negative border, choose a slightly smaller support level, or require certain similarities. To handle all these different possibilities, we generalize the exact representation problem (problem 1) as follows.

| N | O,P,Q | R,S,T | A,B,U | C,D,V | E,F,W |
|---|---|---|---|---|---|
| A,B,G | A,B,G | A,B,I | A,B,I | A,B,J | A,B,J |
| C,D,G | C,D,G | C,D,I | C,D,I | C,D,J | C,D,J |
| E,F,G | E,F,G | E,F,I | E,F,I | E,F,J | E,F,J |
| A,G,M | A,G,M | K,L,M | K,L,M | K,L,I | K,L,I |
| K,L,A | K,L,A | A,G,I | A,G,I | G,H,M | G,H,M |
| G,H,I | G,H,I | B,G,M | B,G,M | B,G,I | B,G,I |
| A,B,G | A,G,M | A,B,I | K,L,M | A,B,J | K,L,I |
| C,D,G | K,L,A | C,D,I | A,G,I | C,D,J | G,H,M |
| E,F,G | G,H,I | E,F,I | B,G,M | E,F,J | B,G,I |

**Table 1: A toy transactional database. Each cell is a set of items contained by one transaction. There is a total of 60 transactions.**

DEFINITION 2. **(Problem 2: Approximate Representation)** *Given an expanded collection of frequent itemsets for $F_\alpha$, denoted as $\tilde{F}_\alpha \supseteq F_\alpha$, find a list of Cartesian products, $\mathcal{X}_1 \otimes \mathcal{Y}_1$, $\mathcal{X}_2 \otimes \mathcal{Y}_2$, $\cdots \mathcal{X}_k \otimes \mathcal{Y}_k$, such that $\mathcal{C}(\mathcal{X}_1 \otimes \mathcal{Y}_1)$, $\mathcal{C}(\mathcal{X}_2 \otimes \mathcal{Y}_2)$, $\ldots$, $\mathcal{C}(\mathcal{X}_k \otimes \mathcal{Y}_k)$ cover $F_\alpha$:*

$$\tilde{F}_\alpha \supseteq \bigcup_{1 \le i \le k} \mathcal{C}(\mathcal{X}_i \otimes \mathcal{Y}_i) \supseteq F_\alpha$$

*where the $cost(\{\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k\})$ is minimal.*

For example, if in the previous example we let $\tilde{F}_\alpha = F_\alpha \cup \{\{A,G,H\}\}$, then we can approximately represent the dataset of table 1 by two generalized Cartesian products:

$$\{\{G\},\{I\},\{J\}\} \otimes \{\{A,B\},\{C,D\},\{E,F\}\}$$
$$\{\{M\},\{I\},\{A\}\} \otimes \{\{A,G\},\{K,L\},\{B,G\},\{G,H\}\}$$

Note that the exact representation problem becomes a special case of the approximate representation problem if we denote $\tilde{F}_\alpha = F_\alpha$. We also refer to both the exact and approximate representation of a collection of frequent itemsets as its *Cartesian contour*.

## 1.2 Our Contribution

In this work, we make the following contributions.

1. We propose a new scheme (Cartesian contour) based on the generalized Cartesian products to concisely represent a collection of frequent itemsets.

2. We perform a theoretical investigation of this problem and link it to the minimal biclique set cover problem and prove its NP-hardness.

3. We develop a novel approach utilizing the techniques developed in frequent itemset mining, set cover, and max k-cover to approximate the minimal biclique set cover.

4. We consider several techniques to efficiently construct the Cartesian contour using our minimal biclique set cover algorithm.

5. We perform a detailed experimental study and demonstrate the effectiveness and efficiency of our approach.

## 2. THEORETICAL BASICS

In this section, we perform a theoretical investigation of the problems in finding the minimal Cartesian contour for a collection of frequent itemsets. We link this problem to the minimal biclique set cover problem and demonstrate its NP-hardness.

We first simplify our concise representation from covering the collection of frequent itemsets to covering only the maximal frequent itemsets. Let $\mathcal{M}_\alpha = \{M_1, M_2, \ldots, M_p\}$ be the set of maximal frequent itemsets under support level $\alpha$. By definition, an itemset is maximal frequent if none of its supersets are frequent. Hence it's easy to see that $\mathcal{C}(\mathcal{M}_\alpha) = F_\alpha$. Thus, we can simplify our problems by focusing on covering the maximal frequent itemsets.

Let $\mathcal{M}(\mathcal{X}_i \otimes \mathcal{Y}_i)$ record only the maximal itemsets covered by the Cartesian product:

$$\mathcal{M}(\mathcal{X}_i \otimes \mathcal{Y}_i) = \mathcal{C}(\mathcal{X}_i \otimes \mathcal{Y}_i) \cap \mathcal{M}_\alpha$$

Then, we basically want to find a list of Cartesian products, $\mathcal{X}_1 \otimes \mathcal{Y}_1$, $\mathcal{X}_2 \otimes \mathcal{Y}_2$, $\cdots \mathcal{X}_k \otimes \mathcal{Y}_k$, such that they cover all the maximal frequent itemsets $M_\alpha$:

$$\tilde{F}_\alpha \supseteq \bigcup_{1 \le i \le k} \mathcal{M}(\mathcal{X}_i \otimes \mathcal{Y}_i) \supseteq \mathcal{M}_\alpha$$

where the $cost(\{\mathcal{X}_1 \otimes \mathcal{Y}_1, \mathcal{X}_2 \otimes \mathcal{Y}_2, \cdots \mathcal{X}_k \otimes \mathcal{Y}_k\})$ is minimal.

Now we can transform our problem as a biclique set cover in the bipartite graph. Let $S$ be the ground set to be covered. Let $G = (A \cup B, E)$ be a bipartite graph where an edge in $E$ connects two vertices from $A$ and $B$, respectively. Each edge $e$ is associated with a subset $S_e \subseteq S$ and $S = \cup_{e \in E} S_e$. Let $C = (\mathcal{X}, \mathcal{Y}, \mathcal{X} \times \mathcal{Y})$ be a subgraph of $G$, where $\mathcal{X} \subseteq A$, $\mathcal{Y} \subseteq B$, $\mathcal{X} \times \mathcal{Y} \subseteq E$. Thus, $C$ is a biclique. Let $S(C)$ be the union of sets covering all it's edges, i.e., $S(C) = \cup_{e \in \mathcal{X} \times \mathcal{Y}} S_e$. Further, we associate a cost with each biclique $C$, denoted as $cost(C)$. Given this, the minimal biclique set cover problem is defined as follows.

DEFINITION 3. **(Minimal Biclique Set Cover)** *Given the ground set $S$ and a bipartite graph $G$, where $S = \cup_{e \in E} S_e$, we seek a list of bicliques, $C_1$, $C_2$, $\cdots$, $C_k$, to cover the ground set, i.e., $S = \cup_{1 \le i \le k} S(C_i)$, with the minimal cost, $\sum_{1 \le i \le k} cost(C_i)$.*

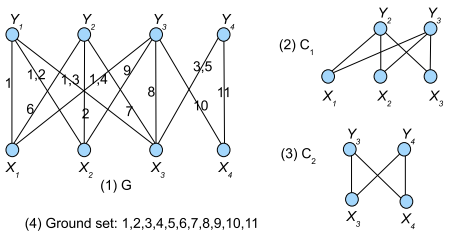


**Figure 1: $C_1$,$C_2$ of $G$ cover the ground set**

Figure 1 shows an example of minimal biclique set cover. Each edge of $G$ is associated with a subset of the ground set, as marked on that edge. We can use bicliques $C_1$ and $C_2$ from $G$ to cover the ground set.

We can transform our problem to a minimal biclique set cover problem as follows. We construct a bipartite graph, where each of its vertices corresponds to an itemset. Basically, let
$A = \{X_1, X_2, \ldots, X_m\}$ containing $m$ itemsets and
$B = \{Y_1, Y_2, \ldots, Y_l\}$ containing $l$ itemsets. At this point, for the easy of understanding of our problem, we can simply assume $A = B = F_\alpha$. Then, we construct an edge between itemsets $X_i$ and $Y_j$ if and only if $X_i \cup Y_i$ is a frequent itemsets, i.e., $X_i \cup Y_i \in \tilde{F}_\alpha$. We further assign the set of maximal frequent itemsets being covered by $X_i \cup Y_i$, to the edge $(X_i, Y_i)$ in the bipartite graph. Clearly, for the exact representation, $\tilde{F}_\alpha = F_\alpha$, each edge either has a singleton (covering only a single maximal frequent itemset), or the empty set ($\emptyset$). However, when $\tilde{F}_\alpha \supset F_\alpha$, each edge can associate a set with more than one maximal frequent itemset. In addition, we define the cost of a biclique $C = (\mathcal{X}, \mathcal{Y}, \mathcal{X} \times \mathcal{Y})$ as the cost of its corresponding Cartesian product:

$$cost(C) = cost(\mathcal{X} \otimes \mathcal{Y})$$

Thus, we have demonstrated that our problem is an instance of the minimal biclique set cover problem.

The NP-hardness of the *minimal biclique set cover* can be easily observed by noting that a single biclique can represent a candidate set. In addition, we have the following NP-hardness results of our Cartesian contour problems. The Proof is omitted due to the space limit.

THEOREM 1. *Problem 1 (Exact Representation) and 2 (Approximate Representation) are NP-hard.*

## 3. ALGORITHM FOR MINIMAL BICLIQUE SET COVER

In this section, we will develop a novel algorithmic framework to handle the minimal biclique set cover problem, which is the generalization of our Cartesian contour problem. In the next section, we will consider how to leverage this framework effectively to handle our Cartesian contour problem by considering how to construct a sparse bipartite graph and how to handle the approximation.

### 3.1 Basic Ideas

Since our problem is essentially using bicliques of a bipartite graph $G$ to cover the ground set $S$. A natural approach is the greedy set cover algorithm [6]. Let $R$ be the covered subset of $S$. For each biclique (or sub-biclique) $C = (\mathcal{X} \cup \mathcal{Y}, \mathcal{X} \times \mathcal{Y})$ in the bipartite graph $G$, we define the *price* of $C$ as:

$$\gamma(C) = \frac{|\mathcal{X}| + |\mathcal{Y}|}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}} S_e \backslash R|} \quad (1)$$

By greedy set cover algorithm, we will choose a biclique with minimum price in each iteration until the ground set $S$ is covered ($R = S$) and we will have a logarithmic approximation bound [6]. However, the problem is that the total number of bicliques in the bipartite graph $G$ is exponential ($O(2^{|A|+|B|})$). Thus, to directly apply the greedy set cover algorithm is computationally intractable as the number of candidate sets (bicliques) is too large

Our basic strategy to tackle this problem is as follows. First, we utilize the connection between frequent itemset mining with bicliques in a bipartite graph to quickly identify "one-side maximal bicliques" (defined in section 3.3), which can be much smaller than the number of total bicliques. Then, we introduce a fundamental lemma which can select a sub-biclique from a (maximal) biclique with the cheapest price linearly and with a constant approximation bound. In the following we first introduce the fundamental lemma.

### 3.2 Finding Low Price Sub-biclique from Biclique in Polynomial Time

In this subsection, we introduce a fundamental lemma and algorithm to identify a sub-biclique with cheapest price. This tool

forms a basic building block of our approach to construct the minimal biclique set cover.

DEFINITION 4. **(Cheapest Sub-biclique Problem)** *Given a clique $C = (\mathcal{X} \cup \mathcal{Y}, \mathcal{X} \times \mathcal{Y})$, where each of its edges $e \in \mathcal{X} \times \mathcal{Y}$ associates with a set $S_e$. We would like to find a sub-biclique $C' = (\mathcal{X} \cup \mathcal{Y}', \mathcal{X} \times \mathcal{Y}')$ from $C$, where $\mathcal{Y}' \subseteq \mathcal{Y}$, with the lowest covering price:*

$$\arg \min_{\mathcal{Y}' \subseteq \mathcal{Y}} \frac{|\mathcal{X}| + |\mathcal{Y}'|}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|}$$

Note that we can easily generalize this problem to consider a set of elements, denoted as $R$, which has already been covered in the biclique. In this case, we can simply treat the set associated with each edge $e$ as $S_e \backslash R$. Clearly, the solution to our above problem can directly handle such generalization. Therefore, the *price* of a sub-biclique here is defined as $\gamma(C') = \frac{|\mathcal{X}|+|\mathcal{Y}'|}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|}$ which does not consider $R$ as in Formula ( 1).

It is easy to see the brute-force algorithm cannot work here because the number of candidate sub-bicliques is $2^{|\mathcal{Y}|}$. We will introduce a polynomial time algorithm for this problem. To explain our algorithm, we first transform it into a classical SET COVER problem setting. Let $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_{|\mathcal{Y}|}\}$, and let $C_j = (\mathcal{X} \cup \{Y_j\}, \mathcal{X} \times \{Y_j\})$ be a biclique containing only a single $Y$-vertex and the entire $\mathcal{X}$ vertices, referred to as *single-$Y$-vertex biclique*. Now, we treat each single-$Y$-vertex biclique $C_j$, $1 \leq |C_j| \leq |\mathcal{Y}|$, as a candidate set, which covers all the elements associated with each of its edges, $\bigcup_{e \in \mathcal{X} \times \{Y_j\}} S_e$. Clearly, the clique $C$ has a total of $|\mathcal{Y}|$ candidate sets (as well as single-$Y$-vertex bicliques).

Now, we consider the following variant of the set cover problem, the MAX $k$-COVER problem, which seeks $k$ sets whose union has maximal cardinality. Specifically in our problem setting, Max $k$-COVER is to get $k$ single $Y$-vertex bicliques of $C$, such that they cover a maximal number of elements. Notice that the union of the $k$ single $Y$-vertex bicliques of $C$ actually forms a biclique. Then, the MAX $k$-COVER problem in this setting basically corresponds to finding a sub-biclique $(\mathcal{X} \cup \mathcal{Y}', \mathcal{X} \times \mathcal{Y}')$ of $k$ vertices in $\mathcal{Y}$, $|\mathcal{Y}'| = k$, which maximize $\cup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e$. This is equivalent to minimizing

$$\frac{|\mathcal{X}| + k}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|}$$

for any fixed $k$. It is well-known that the greedy algorithm provides a bounded $(1 - \frac{1}{e})$ approximation ratio to this problem [9]. Note that the difference between our Cheapest Sub-biclique Problem and the MAX $k$-COVER problem is that the latter needs a fixed $k$ and the former does not have such constraint. Given this, we simply need to generalize the greedy algorithm for the MAX $k$-COVER problem to help identify the cheapest sub-biclique.

Our greedy procedure for the cheapest sub-biclique is as follows. *1) We initialize an empty vertex set $\mathcal{Y}' = \emptyset$; 2) At each iteration, we choose the single-$Y$-vertex biclique $C_i$ with the maximal number of uncovered elements,*

$$\arg \max_{Y_i \in (\mathcal{Y} \backslash \mathcal{Y}')} |\bigcup_{e \in \mathcal{X} \times \{Y_i\}} S_e \backslash \bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|$$

*3) We try to add the vertex $Y_i$ to $\mathcal{Y}'$ to build a new biclique as $(\mathcal{X} \cup (\mathcal{Y}' \cup \{Y_i\}), \mathcal{X} \times (\mathcal{Y}' \cup \{Y_i\}))$. As long as the price is being reduced by the new vertex, i.e.,*

$$\frac{|\mathcal{X}| + |\mathcal{Y}'|}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|} > \frac{|\mathcal{X}| + |\mathcal{Y}'| + 1}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}' \cup \{Y_i\}} S_e|}$$

*we add $Y_i$ to $\mathcal{Y}'$. Otherwise (we find the new vertex does not reduce the price), we will stop the iteration, and return the current biclique $(\mathcal{X} \cup \mathcal{Y}', \mathcal{X} \times \mathcal{Y}')$ as the one with cheapest price in $C$.*

We refer to this procedure as *OptimalSubBiclique*. It has a worst case time complexity $O(|\mathcal{Y}|^2|\mathcal{X}|)$, as in each iteration we need to update every single-$Y$-vertex biclique in $C$, pick up a lowest-price one and add it into the current biclique. This procedure can find a $(\frac{e}{e-1})$ approximation ratio for our Cheapest Sub-biclique Problem.

LEMMA 1. *Let the $C^*$ be the lowest-price sub-biclique in $\{(\mathcal{X}, \mathcal{Y}^*) | \mathcal{Y}^* \subseteq \mathcal{Y}\}$. The OptimalSubBiclique procedure finds a sub-biclique $C' \in \{(\mathcal{X}, \mathcal{Y}') | \mathcal{Y}' \subseteq \mathcal{Y}\}$ such that $\gamma(C^*) \leq \gamma(C') \leq \frac{e}{e-1}\gamma(C^*)$.*

**Proof Sketch:** Without loss of generality, let us assume the selected sub-biclique at the $i$-th iteration by greedily choosing the single-$Y$-vertex biclique with the maximal number of uncovered elements as $C'_i$. However, our algorithm stops at the $t+1$-th iteration (i.e., $C' = C'_t$) when the price stops decreasing.

To prove this lemma, we first prove the following claim: If in our *OptimalSubBiclique* algorithm, we do not stop greedily choosing and adding single-$Y$-vertex bicliques, the price of the later formed sub-biclique $\gamma(C'_i), (i > t)$, will be greater than the price of $C'_t$. Basically, we will show:

$$\frac{|\mathcal{X}| + t}{f(C'_t)} < \frac{|\mathcal{X}| + i}{f(C'_i)}, \text{ for any } i, (t < i \leq |\mathcal{Y}|)$$

where $f(C'_i)$ is the number of elements covered by sub-biclique $C'_i$.

To show this, it is easy to observe that the coverage size of a single-$Y$-clique will never go up, i.e.,

$$f(C'_2) - f(C'_1) \geq \ldots \geq f(C'_{t+1}) - f(C'_t) \geq \ldots$$

Thus, we have for any $i > t$,

$$\frac{|\mathcal{X}| + i}{f(C'_i)} = \frac{|\mathcal{X}| + t + (i-t)}{f(C'_t) + \sum_{j=t+1}^{i}(f(C_j) - f(C_{j-1}))} \geq \frac{|\mathcal{X}| + t + (i-t)}{f(C'_t) + (i-t)(f(C_{t+1}) - f(C_t))} \geq \frac{|\mathcal{X}| + t}{f(C'_t)}$$

from $\frac{|\mathcal{X}| + t + 1}{f(C'_{t+1})} = \frac{|\mathcal{X}| + t + 1}{f(C'_t) + (f(C_{t+1}) - f(C_t))} \geq \frac{|\mathcal{X}| + t}{f(C'_t)}$

Now we prove the lemma. It is straightforward that $\gamma(C^*) \leq \gamma(C')$ because $C^*$ is the lowest-price sub-biclique. Therefore we focus on the proof of $\gamma(C') \leq \frac{e}{e-1}\gamma(C^*)$.

We consider the optimal sub-biclique of $C$ for a fixed size $k$ on the $\mathcal{Y}$ vertices, denoted as $C^*_k$:

$$C^*_k = \arg \min_{|\mathcal{Y}'|=k, \mathcal{Y}' \subseteq \mathcal{Y}} \frac{|\mathcal{X}| + k}{|\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e|}$$

Thus, we can redefine the cheapest price sub-biclique $C^*$ as

$$C^* = \arg \min_{1 \leq k \leq |\mathcal{Y}|} \gamma(C^*_k)$$

Given this, it is easy to see that our *OptimalSubBiclique* simply applies the MAX k-COVER greedy procedure to find a sub-biclique $C'_k$ at each iteration which covers a maximal number of elements $(f(C'_k))$ for $(k \leq t+1)$. This is within a $(1 - \frac{1}{e})$ approximation ratio of the optimal one:

$$f(C'_k) \geq (1 - \frac{1}{e}) f(C^*_k)$$

Then we can see

$$\gamma(C'_k) \leq \frac{e}{e-1}\gamma(C^*_k), \text{ because } \frac{|\mathcal{X}| + k}{f(C'_k)} \leq \frac{|\mathcal{X}| + k}{(1 - \frac{1}{e}) f(C^*_k)}$$

Put together, we have

$$\gamma(C') \leq \min_{1\leq k\leq t+1} \gamma(C'_k) \leq \min_{1\leq k\leq |\mathcal{Y}|} \gamma(C'_k)$$
$$\leq \min_{1\leq k\leq |\mathcal{Y}|} \frac{e}{e-1}\gamma(C^*_k) = \frac{e}{e-1}\gamma(C^*)$$

□

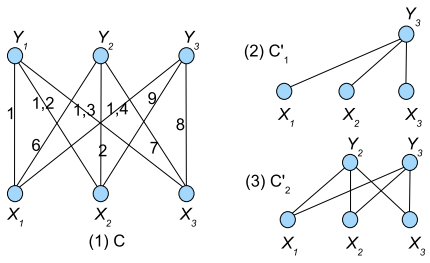


**Figure 2: An example of OptimalSubBiclique**

Figure 2 shows an example of $OptimalSubBiclique$: Given biclique $C$ as shown in (1), we first choose the single-$Y$-vertex biclique $Y_3 \cup \{X_1, X_2, X_3\}$ with the greatest number (here is 4) of uncovered elements and put it into $C'$, as shown in (2). Then $\gamma(C'_1) = 4/4 = 1$. Following the same rule, secondly we choose the single-$Y$-vertex biclique $Y_2 \cup \{X_1, X_2, X_3\}$ with 3 uncovered elements and put it into $C'$, as shown in (3). Then $\gamma(C'_2) = 5/7$. Lastly we choose the single-$Y$-vertex biclique $Y_1 \cup \{X_1, X_2, X_3\}$. Now it has only 1 uncovered element and the price $\gamma(C'_3) = 6/8$, which is greater than $5/7$. Therefore we revert the last step and the final output is the biclique $C'_2$ as shown in (3).

## 3.3 The Biclique-Covering Algorithm

Based on *OptimalSubBiclique* and Lemma 1, we can find low-price sub-biclique, where one-side is fixed, from a given biclique. To release the power of this tool, we introduce the notation of *one-side maximal biclique*. Given a bipartite graph $G = (A \cup B, E)$, for a biclique $C = (\mathcal{X} \cup \mathcal{Y}, \mathcal{X} \times \mathcal{Y})$, where $\mathcal{X} \subseteq A$, $\mathcal{Y} \subseteq B$, and $\mathcal{X} \times \mathcal{Y} \subseteq E$, if we add an extra vertex to $\mathcal{X}$, it will not be a biclique of $G$, then, we refer to it as a *$A$-side maximal biclique*. Similarly, we refer to it as $B$-side maximal biclique. In other words, for instance, a $B$-side maximal biclique of $G$ can be simply written as

$$(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X}))$$

where $B(\mathcal{X}) = \{b|(x, b) \in E \text{ for any } x \in \mathcal{X}\}$ simply includes all the vertices in $B$, which links to every vertex in $\mathcal{X}$. An $A$-side or $B$-side maximal biclique is called as a *one-side maximal biclique*. It is easy to see that if a biclique is both $A$-side maximal and $B$-side maximal, it is a *maximal biclique* [12].

Recall in the SET COVER greedy algorithm (Subsection 3.1, we need to find a biclique with lowest price, which needs an enumeration of $O(2^{|A|+|B|})$ bicliques of $G$. We can achieve such a task using the *$B$-side maximal biclique* and the *OptimalSubBiclique* procedure as follows. Let $U$ be the set of all $B$-side maximal bicliques of $G$:

$$U = \{(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X})\}$$

Then, we can visit each $B$-side maximal biclique of $U$ and invoke the *OptimalSubBiclique* procedure to find its lowest price subbiclique, and then choose the one with lowest price among these subbicliques (a total of $|U|$). However, the number of bicliques in $U$ is still in the order of $O(2^{|A|})$. To bring down the number of exponential $B$-side maximal bicliques, we propose to approximate $U$ using frequent itemset mining technique. This is quite interesting as our initial goal is to concisely represent a collection of frequent itemsets, and its solution also needs frequent itemset mining, though on a different transactional database which is derived from the bipartite graph.

**Frequent itemsets and One-side Maximal Bicliques:** To reveal the relationship between frequent itemsets and one-side maximal bicliques, we will have to represent the bipartite graph $G = (A \cup B, E)$ as a transactional database $D(G)$. Let each row correspond to a vertex in $A$ and each column corresponds to a vertex in $B$ and if an edge between a vertex $a$ and $b$, then, we have $D(a, b) = 1$. For a support level $\beta$, let $F_\beta[D(G)]$ be the collection of all frequent itemsets whose support is greater than or equal to $\beta$. Further, for a frequent itemset $I \in F_\beta[D(G)]$, let $T(I)$ be its supporting transactions, i.e., the transactions have $I$ as the subset. Then, we can see the Cartesian product $I \times T(I) = \{(i, t)|i \in I \text{ and } t \in T(I)\}$ corresponds to a $B$-side maximal biclique $(I \cup T(I), I \times T(I))$ of $G$, where $I \subseteq A$ and $T(I) \subseteq B$.

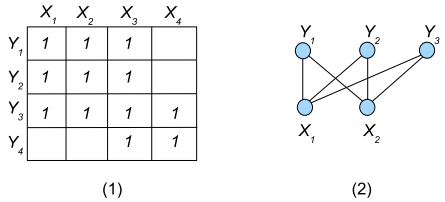


**Figure 3: Representing a bipartite graph as a transactional database**

Given this, we construct the following set of $B$-side maximal bicliques to approximate $U$:

$$U_\beta = \{(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X}))|\mathcal{X} \in F_\beta[D(G)] \cup A\}$$

Clearly, it includes only those $B$-side maximal bicliques, whose $A$-subsets either correspond to a frequent itemset in the transactional database $D(G)$, or contain a single vertex in $A$ (for the completeness of set cover). Note that this allows us to utilize the fast frequent itemset mining algorithms to construct the set $U_\beta$.

As an example, figure 3(1) is a transactional database converted from the bipartite graph shown in figure 1(1). Given $\beta = 0.5$, figure 3(2) shows a $B$-side maximal biclique (there are 9 in total).

**Overall Algorithm:** The general sketch of our biclique-covering algorithm is illustrated as Algorithm 1, which takes $G, U_\beta$ as input. In each iteration it will find the lowest-price sub-biclique (with $\mathcal{X}$ on the $A$ side) from each biclique $C(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X})) \in U_\beta$ using the *OptimalSubBiclique* procedure (Line 4). Then it selects the cheapest $C'$ from the set of selected sub-bicliques (Line 5). $C'$ will be added into the set of $Bicliques\_Found$ (Line 6). Set $R$ which records the covered elements which are associated with each edge in the bipartite graph, and will be updated accordingly (Line 7). The while loop terminates when the ground set $S = \cup_{e\in E} S_e$ is covered, i.e. $R = S$ (Line 3-8).

We note that there are well-established techniques to speedup the greedy set cover procedure [16, 14] and both our *BicliqueCover* algorithm and *OptimalSubBiclique* procedure can employ them. The basic idea of these techniques is to maintain a priority queue of the candidate sets and sort them based on their price. However, since the price of each candidate set cannot decrease (since more elements are being covered, i.e., $R$ grows), we can apply this property to prune the search space for each iteration significantly. Due to the space limitation, we omit the details here.

**Algorithm 1** BicliqueCover($G, U_\beta$)

**Require:** $G = (A \cup B, E)$ is the bipartite graph
**Require:** $U_\beta$ is the set of $B$-side maximal bicliques of $G$
1: $R \leftarrow \emptyset$;
2: $Bicliques\_Found \leftarrow \emptyset$;
3: **while** $R \neq \cup_{e \in E} S_e$ **do**
4: call OptimalSubBiclique to find $C' = (\mathcal{X} \cup \mathcal{Y}', \mathcal{X} \times \mathcal{Y}')$ of each $(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X})) \in U_\beta$ ($Y' \subseteq B(\mathcal{X})$) with minimum $\gamma(C')$;
5: choose the $C'$ with minimum $\gamma(C')$ among all the ones discovered by OptimalSubBiclique;
6: $Bicliques\_Found \leftarrow Bicliques\_Found \cup \{C'\}$;
7: $R \leftarrow R \cup (\bigcup_{e \in \mathcal{X} \times \mathcal{Y}'} S_e)$;
8: **end while**
9: **return** $Bicliques\_Found$

This algorithm has a bounded approximation ratio with respect to the candidate sets $U_\beta$ as stated in Theorem 2.

The proof is omitted due to the space limit.

THEOREM 2. *The Biclique_Cover algorithm has* $\frac{e}{e-1}(\ln(n) + 1)$ *approximation ratio with respect to the candidate set* $U_\beta$.

**Time Complexity of *BicliqueCover* Algorithm:** Here, we consider the worst case time complexity of our algorithm (the speedup techniques are not considered since they typically do not reduce the worst case time complexity.) Further the *BicliqueCover* procedure assume the frequent itemsets of the transactional database $D(G)$ transformed from the bipartite graph is given. Thus, the overall running time for the minimal biclique set cover problem includes the time of running the frequent itemsets mining algorithm, which we do not consider for the time complexity analysis of *BicliqueCover* algorithm.

The worst case time complexity of the *BicliqueCover* algorithm is $O((|F_\beta[D(G)]| + |A|)(|A||B|^2)k)$, where $k$ is the number of bicliques in the final result. Assume the while loop in Algorithm 1 runs $k$ times. Each time it chooses a $C'$ with minimum $\gamma(C')$ from $U_\beta$, which contains no more than $|F_\beta[D(G)]| + |A|$ candidates. The *OptimalSubBiclique* procedure takes $O(|B||B||A|)$ time. Since we need to do so for every biclique in $U_\alpha$, it takes $O((|F_\beta[D(G)]| + |A|)(|B||B||A|))$. In addition, we note that $k$ is bounded by $(|F_\beta[D(G)]| + |A|) \times |B|$ since each biclique in $U_\beta$ can be visited at most $|B|$ times. Thus, we conclude that our $BicliqueCover$ algorithm runs in polynomial time with respect to $|F_\beta[D(G)]|$, $|A|$ and $|B|$.

# 4. FAST CONSTRUCTION OF CARTESIAN CONTOUR

In the previous section, we present the *BicliqueCover* algorithm for the minimal biclique set cover algorithm. Following the discussion in Section 2, we will try to transform the Cartesian contour problem into a minimal biclique set cover problem. However, the time to construct the bipartite graph $G = (A \cup B, E)$ can significantly affect the efficiency of the algorithm. This is because our algorithm relies on finding the itemsets in the transactional database $D(G)$, which correspond to the bipartite graph $G$. If $G$ is very dense, then the itemsets of $D(G)$ even with a high support level may be very large. Besides, since $A$ corresponds to the items (columns) in the transactional database, we also need $A$ to be small. Otherwise, the search space for the itemsets is too large. Given this, it is clear that the simple transformation as described in Section 2, which lets $A$ and $B$ contain all frequent itemsets $F_\alpha$, and links an edge between two itemsets if their union is in $\tilde{F}_\alpha$, is prohibitive for a large number of maximal frequent itemsets.

In the following, we will introduce several techniques which help with constructing sparse bipartite graphs and present our overall algorithm for deriving a concise Cartesian Contour representation for a large number of (maximal) frequent itemsets.

## 4.1 Closed Itemsets for $A$ and $B$

In this subsection, we introduce a method to reduce the size of vertex sets $A$ and $B$ of the bipartite graph. Recall our goal is to use general Cartesian products (i.e. bicliques) to cover the collection of maximal frequent itemsets $\mathcal{M}_\alpha$ with minimum cost. The process is similar to the "factorization" of the $\mathcal{M}_\alpha$, i.e., finding the factors (the common itemsets) shared by a group of maximal itemsets. Considering a Cartesian product

$$\mathcal{X} \otimes \mathcal{Y} = \{X_1 \cup Y_1, \cdots, X_1 \cup Y_l, \cdots, X_m \cup Y_l\} \subseteq \mathcal{M}_\alpha$$

covering $l \times m$ maximal itemsets, then, each $X_i$ is contained in at least $l$ maximal itemsets and each $Y_j$ is contained in at least $m$ maximal itemsets. Thus we are interested in finding a complete set of "maximal factors" such that they are the largest subsets which are shared by a number of maximal itemsets in $\mathcal{M}_\alpha$.

DEFINITION 5. **(Maximal Factor Itemsets)** *An itemset $X$ is a maximal factor itemset of $M_\alpha$ if the following two condition holds: 1) $X \subseteq M_i \cap M_j \cap \ldots \cap M_l$ where $\{M_i, M_j, \ldots, M_l\}$ is a subset of $\mathcal{M}_\alpha$; and 2) There does NOT exist another subset $X'$ such that $X \subset X'$ and $X' \subseteq M_i \cap M_j \cap \ldots \cap M_l$.*

The definition of *maximal factor itemsets* has the following important property which enables us to reduce the size of $A$ and $B$: *Given a maximal factor itemset $X$ which is a subset of $M_1 \cap M_2 \cap \ldots \cap M_l$ and $X' \subset X$, if*

$$X' \otimes \{Y_1, Y_2, \ldots, Y_k\} = \{M_1, M_2, \ldots, M_k\}$$
$$\textit{then}\ , X \otimes \{Y_1, Y_2, \ldots, Y_k\} = \{M_1, M_2, \ldots, M_k\}$$

However, the reverse is not always true. Therefore, it is not hard to see that $X'$ is a redundant information for our Cartesian contour (exact representation) problem as we can always use $X$ to replace $X'$. Regarding efficiency, we do not consider the the approximate representation problem here (for constructing $A$ and $B$). The next subsection will deal with that.

Now the problem is how we can find the maximal factor itemsets for $M_\alpha$. The following lemma builds the connection between the notion of maximal factor itemsets and the notion of *closed itemsets*. Given a transaction database, a closed itemset is an itemset all of whose immediate supersets do not have the same support as its. In the context of frequent itemset mining, we typically give a minimum support and refer to the closed itemsets whose support is no less than the minimum support as the frequent closed itemsets. In other words, the closed itemsets are the frequent closed itemsets with support zero.

LEMMA 2. *For a collection of maximal frequent itemsets, $\mathcal{M}_\alpha$, we construct a transactional database $DB(\mathcal{M}_\alpha)$, which records each maximal itemset in $\mathcal{M}_\alpha$ as a unique transaction. Thus, the database has a number of $|\mathcal{M}_\alpha|$ transactions. Then, the closed itemsets of $DB(\mathcal{M}_\alpha)$ are maximal factor itemsets of the collection of maximal itemsets $\mathcal{M}_\alpha$.*

For simplicity, we omit the proof here. Given this, we utilize only the closed itemsets in $A$ and $B$, which is much less than the number of frequent itemsets.

## 4.2 Sparse and Dense Bipartite Graphs

Even though we have reduced the size of $A$ and $B$, we still need to consider how to reduce the edge density of the bipartite graph. Recall the straightforward solution is simply connecting two itemsets (vertices) if their union is in the expanded frequent itemsets $\tilde{F}_\alpha$. Clearly, this will make the bipartite graph very dense. However, if we construct a sparse bipartite graph, we may lose some good biclique candidates (or Cartesian products).

To deal with this problem, we introduce a novel "two-bipartite-graphs" technique. First, we construct a sparse bipartite graph $G_s$, which is used for finding the frequent itemsets with support $\beta$, $F_\beta[D(G_s)]$. Then, we construct a dense bipartite graph $G_d$, which is used to build the set of $B$-side maximal bicliques, $U_\beta$.

**Sparse Bipartite Graph $G_s$:** We construct the edge set $E_s$ for the sparse bipartite graph $G_s = (A \cup B, E_s)$ as follows: two vertices $(a, b)$ have an edge if and only if the union of their corresponding itemsets is a maximal frequent itemset.

**Dense Bipartite Graph** $G_d$: We construct the edge set $E_d$ for the sparse bipartite graph $G_d = (A \cup B, E_d)$ as follows: two vertices $(a, b)$ have an edge if and only if the union of their corresponding itemsets is in the expanded set of frequent itemsets $\tilde{F}_\alpha$.

Given this, after we find the frequent itemsets from $G_s$, we construct the candidate biclique set as

$$U_\beta = \{(\mathcal{X} \cup B(\mathcal{X}), \mathcal{X} \times B(\mathcal{X})) | \mathcal{X} \in F_\beta[D(G_s)] \cup A, \text{ and } B(\mathcal{X}) \text{ is the supporting transactions in } G_d\}$$

The idea is to reduce the number of frequent itemsets through the sparse bipartite graph ($|F_\beta[D(G_s)]|$ is small). Then, we use the dense graph to build a larger biclique by incorporating more vertices from $B$ into each candidate bicliques. Since our *Optimal-SubBiclique* performs linear search over the vertices from $B$ in each candidate biclique, we can traverse a large number of sub-bicliques without compromising the efficiency.

## 4.3 Iterative Algorithm

The last technique we consider is to handle the case where a large number of closed itemsets are being produced from $DB(\mathcal{M})$. In this case, we consider iteratively invoking the *BicliqueCover* procedure, and utilize the frequent closed itemset here. Basically, we apply a minimum support $\delta$ for $DB(\mathcal{M}_\alpha)$ and reduce it gradually in each round. Since we do not expect to cover $\mathcal{M}_\alpha$ in a single pass, we can also further shrink the size of $A$ and $B$, by splitting those frequent closed itemsets into two disjoint parts. For instance, we can put all the itemsets whose size is smaller than a threshold in one dataset and the rest in another one. Then, we always let $A$ be the part which has smaller number of itemsets, to reduce the search space of frequent itemset mining. In addition, we let $A$ and $B$ each contain an empty itemset so that coverage will be complete after several passes.

Given this, our overall procedure is as follows:

*1) Find the frequent closed itemsets in $DB(\mathcal{M}_\alpha)$;*
*2) Split those discovered itemsets into $A$ and $B$;*
*3) Construct the sparse and dense bipartite graphs, $G_s$ and $G_d$, respectively;*
*4) Find the frequent itemsets in $D(G_s)$ and build $U_\beta$, the set candidate bicliques, by using $G_d$, $A$, and $F_\beta[D(G_s)]$;*
*5) Invoke the* BicliqueCover *procedure to cover maximal frequent itemsets;*
*6) Remove the covered maximal itemsets from $\mathcal{M}_\alpha$;*
*7) Reduce the minimum support $\delta$ gradually and then repeat the above steps until $\mathcal{M}_\alpha$ is empty (all maximal itemsets are covered).*

Consider the dataset of table 1 as a running example: Given support $\alpha = 1/20$, the maximal frequent itemsets $\mathcal{M}_\alpha$ are listed in table 2. In addition, the frequent closed itemsets in $DB(\mathcal{M}_\alpha)$ are listed in table 3 given support $\delta = 1/9$. Then we construct sparse and dense bipartite graphs as shown in Figure 4(1). The dark bold edges are edges in both sparse and dense bipartite graphs, and the light thin edges are edges in dense bipartite graph only. After invoking the biclique cover procedure, we find three bicliques (i.e. three general Cartesian products) that cover all frequent itemsets $F_\alpha$ with support $\alpha = 1/20$, as shown in Figure 4(2).

| | | | | | |
|---|---|---|---|---|---|
| A,B,G | C,D,G | A,B,I | C,D,I | A,B,J | C,D,J |
| E,F,G | A,G,M | E,F,I | K,L,M | E,F,J | K,L,I |
| K,L,A | G,H,I | A,G,I | B,G,M | G,H,M | B,G,I |

**Table 2: $\mathcal{M}_\alpha$**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A,B | C,D | E,F | A,G | K,L | G,H | B,G | GI | MG |
| A,I | BI | I | J | M | A | B | G | |

**Table 3: frequent closed itemsets in $DB(\mathcal{M}_\alpha)$**

In this example our overall procedure finishes in one round. However, as an example, if $\mathcal{M}_\alpha$ contains an additional maximal frequent itemset $\{O, P, Q\}$, our overall procedure can finish in two rounds given the same support $\delta = 1/9$. In addition, here we only considered exact representation. If we let $\tilde{F}_\alpha = F_\alpha \cup \{\{A, G, H\}\}$, the edge between $\{A\}$ and $\{G, H\}$ will be added to the sparse bipartite graph, and the approximate representation will contain only two bicliques (i.e. $C_2$ and $C_3$ in figure 4(2) will merge into one biclique).

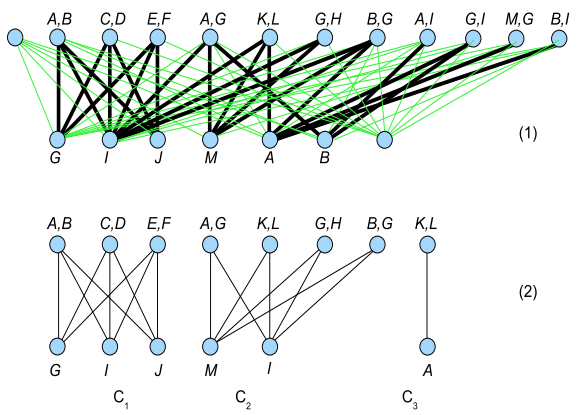


**Figure 4: (1) sparse and dense bipartite graph (2) three bicliques that cover the frequent itemsets $F_\alpha$**

# 5. RELATED WORKS

Our work is closely related to the goal defined by the spanning set approach [1]. In this work, the authors propose to use $K$ itemsets as a concise representation of a collection of frequent itemsets. Basically, the $K$ itemsets are chosen to maximally cover the collection of frequent itemsets. They consider two important cases: 1) if the $K$ itemsets themselves are frequent, i.e., being chosen from the collection, there will be no false positive coverage. Based on the down-closure property, a subset of any of the $K$ itemsets must be frequent. 2) if the $K$ itemsets can be chosen outside the collection, they require that those $K$ itemsets satisfy a certain false positive ratio. They further show that finding such $K$ itemsets corresponds to the classical set-cover problem and thus is NP-hard. The well-known greedy algorithm is applied to find the concise representation. The main differences between their work and ours are in the

| Datasets | $\mathcal{I}$ | $\mathcal{T}$ | density |
|---|---|---|---|
| chess | 75 | 3,196 | dense |
| retail | 16469 | 88162 | sparse |
| connect | 129 | 67557 | dense |
| pumsb | 7116 | 49046 | sparse |
| T40I10D100K | 1000 | 100000 | sparse |

**Table 4: Datasets Characters. $\mathcal{I}$ is the total number of items and $\mathcal{T}$ is the total number of transactions**

following three aspects:1) First, their work is a special case of our work as a spanning itemset $X$ in [1] is simply a Cartesian product between itself and an empty set, i.e., $\{X\} \times \{\emptyset\}$. Indeed, in our work, we do incorporate such cases into consideration by allowing one side of Cartesian product to be an empty set. Careful readers should have observed that in Figure 4(1) there are two non-labeled vertices which actually represent empty sets. 2) Secondly, the goal of their work is to use a small number of itemsets to maximally represent the collection of frequent itemsets. Thus, their work typically can cover only a proportion of the frequent itemsets. Specifically, if no false positive is allowed, the approach in [1] essentially need all the maximal frequent itemsets. In this sense, there will no reduction in terms of the conciseness. While in our work, we focus on representing all the frequent itemsets using a small number of itemsets. 3) Finally, our representation scheme based on the Cartesian product between itemsets is clearly different from [1] and we develop a set of novel techniques utilizing the set cover and MAX $k$-COVER approach to discover such a scheme.

Several methods consider to restoring the frequency for the collection of frequent itemsets. Yan *et al.*'s work [19] introduces a pattern-profile approach. It partitions the itemsets into $K$ clusters, and all frequent itemsets in a cluster are estimated based on the item-independence assumption. Wang *et al.* [15] propose the construction of a global Markov random field (MRF) model to estimate the frequencies of frequent itemsets. This model utilizes the dependence between items, specifically, the conditional independence, to reduce their estimation error. Jin *et al.* [10] derive a regression-based approach to cluster the frequent itemsets and restore the itemset frequency based on the regression model. However, these approaches cannot provide a global view of the collection of frequent itemsets and their power to predict which itemsets are frequent or not seems also to be limited by their frequency restoration error. Our goal here is different as we focus on the the concise representation of the collection of frequent itemsets. It is an interesting research problem as to how this approach can contribute to the itemset frequency restoration.

Finally, our problem is also related to the hyperrectangle covering problem [16]. However, as we discussed in Subsection 2, this problem is only a special case of the minimal biclique cover problem, and the method developed in [16] cannot handle our problem. Even though both works link their roots to the set cover problem, this work has developed a set of new techniques to handle the complexity of using bicliques for set cover. Especially, we develop a fundamental lemma based on the MAX k-Cover to approximate the low-price sub-bicliques from a clique and techniques to employ the minimal set cover problem for the Cartesian contour representation.

# 6. EXPERIMENTAL RESULTS

In our experimental evaluation, we will focus on answering the following questions:

1. How does Cartesian contour representation summarize the collection of frequent itemsets?
2. How does the approximate representation performs compared with the exact representation?
3. How fast can we construct such representation?

| $\alpha$ | $\lvert\mathcal{M}_\alpha\rvert$ | exact representation | | approximate representation | |
|---|---|---|---|---|---|
| | | Car. prod. | cost | Car. prod. | cost |
| 0.88 | 313 | 21 | 243 | 23 | 171 |
| 0.87 | 348 | 23 | 306 | 25 | 229 |
| 0.86 | 352 | 19 | 309 | 23 | 228 |
| 0.85 | 454 | 22 | 359 | 25 | 297 |
| 0.84 | 510 | 21 | 432 | 28 | 363 |

**Table 5: connect**

| $\alpha$ | $\lvert\mathcal{M}_\alpha\rvert$ | exact representation | | approximate representation | |
|---|---|---|---|---|---|
| | | Car. prod. | cost | Car. prod. | cost |
| 0.9 | 34 | 4 | 29 | 4 | 14 |
| 0.85 | 119 | 13 | 91 | 11 | 38 |
| 0.8 | 226 | 17 | 170 | 18 | 71 |
| 0.75 | 489 | 38 | 393 | 42 | 176 |
| 0.7 | 891 | 37 | 855 | 63 | 366 |

**Table 6: chess**

| $\alpha$ | $\lvert\mathcal{M}_\alpha\rvert$ | exact representation | | approximate representation | |
|---|---|---|---|---|---|
| | | Car. prod. | cost | Car. prod. | cost |
| 0.89 | 348 | 24 | 219 | 20 | 94 |
| 0.88 | 500 | 29 | 320 | 33 | 152 |
| 0.87 | 633 | 38 | 413 | 32 | 183 |
| 0.86 | 825 | 40 | 462 | 40 | 282 |
| 0.85 | 1080 | 76 | 684 | 49 | 415 |

**Table 7: pumsb**

Here, we report our experimental evaluation on 4 real datasets and 1 synthetic dataset. All of them are publicly available from the FIMI repository [1]. The basic characteristics of the datasets are listed in Table 4. Borgelt's implementation of the well-known Apriori algorithm [3] was used to generate frequent itemsets. The maximal frequent itemsets and closed maximal frequent itemsets are generated by MAFIA algorithm [4], which is publicly available online[2]. Our algorithms were implemented in C++ and run on Linux 2.6 on an Intel Xeon 3.2 GHz with 4GB of memory.

For each dataset, we vary the support level from high to low. We perform both the exact Cartesian contour representation and the approximate Cartesian contour representation. For the approximate representation, we specify the expanded collection of frequent itemsets $\tilde{F}_\alpha$ as follows. For each $I \in \tilde{F}_\alpha \backslash F_\alpha$, we require there is at least one maximal frequent itemset $M \in \mathcal{M}_\alpha$, such that $I$ is very close to $M$ with only one item difference.

Table 5, 6 and 7 show the results for real datasets *connect*, *chess*, and *pumsb*, respectively. We can see that on average the exact representations need around 16% to 37% less number of itemsets than the number of maximal itemsets. Most numbers of Cartesian products for the representations are on the order of low tens. The approximate representation needs much less number of itemsets to cover all the frequent itemsets. On average, they only need around 60% of the number of itemsets required by the exact representation.

Table 8 and table 9 show the experimental results for real dataset *retail* and synthetic dataset *T40I10D100K*, respectively. In both results, the cost of exact representation is a little higher than the number of Cartesian products. We found that this is because one side of the Cartesian product contains only one or few itemsets or is simply an empty set, and the other side contains a large number

[1]http://fimi.cs.helsinki.fi/data/

[2]http://himalaya-tools.sourceforge.net/Mafia/

| $\alpha$ | $\lvert\mathcal{M}_\alpha\rvert$ | exact representation | | approximate representation | |
|---|---|---|---|---|---|
| | | Car. prod. | cost | Car. prod. | cost |
| 0.009 | 100 | 8 | 108 | 5 | 86 |
| 0.008 | 122 | 7 | 129 | 5 | 101 |
| 0.007 | 167 | 5 | 172 | 7 | 133 |
| 0.006 | 219 | 6 | 225 | 5 | 172 |
| 0.005 | 284 | 4 | 285 | 5 | 223 |

**Table 8: retail**

| $\alpha$ | $\lvert\mathcal{M}_\alpha\rvert$ | exact representation | | approximate representation | |
|---|---|---|---|---|---|
| | | Car. prod. | cost | Car. prod. | cost |
| 0.02 | 2015 | 3 | 2018 | 12 | 1595 |
| 0.019 | 2372 | 4 | 2376 | 12 | 1879 |
| 0.018 | 2781 | 10 | 2791 | 23 | 2101 |
| 0.017 | 3300 | 23 | 3323 | 30 | 2429 |
| 0.016 | 4003 | 155 | 4147 | 41 | 3112 |

**Table 9: T40I10D100K**

of itemsets. However, as we allow approximation, we can reduce the representation cost significantly.

In Table 8, we can also observe that as $\alpha$ decreases, both the total number of frequent itemsets and the exact representation cost increases. However, the number of Cartesian products for the exact representation decreases. Note that the representation cost in the present work is the total number of itemsets used in all the Cartesian product, and our algorithm focuses on minimizing this cost. A reason to choose this criteria is that we can always do the exact representation with one side of the Cartesian product being an empty set and the other side being the set $\mathcal{M}_\alpha$. In this case, using a single Cartesian product, we may cover a large number of frequent itemsets, but this product does not provide much compression than simply listing each individual frequent itemsets. In general, if one side of the Cartesian product has only very few itemsets and then other side has many, the representation cost would also be high even the Cartesian product can cover a large number of itemsets. As we mentioned before, most of the Cartesian products found in both real dataset *retail* and synthetic dataset *T40I10D100K* have such characteristic. Indeed, this also explains why in Table 9 the number of Cartesian products for some approximate representation is higher than that for the exact representation, but the approximate representation cost is smaller.

In addition, our algorithm is rather efficient. On average, it takes around 277s, 52s, 116s, and 1s to generate the exact representation for datasets *connect*, *chess*, *pumsb*, and *retail*, respectively; it takes around 261s, 50s, 56s, and 1s to generate the approximate representation for datasets *connect*, *chess*, *pumsb*, and *retail*, respectively. For the synthetic dataset, it takes on an average of 140s and 424s to produce the exact and approximate representations, respectively.

# 7. CONCLUSIONS

In this paper, we introduced a Cartesian contour representation for covering the entire collection of frequent itemsets by the observation that shorter itemsets interact with each other to produce longer frequent itemsets. We use the generalized Cartesian product to formalize such interaction and allows the concise representation of a collection of frequent itemsets. We transformed our representation problem as an instance of the minimal biclique covering problem. Based on this, we first developed a general approach for the minimal bicilque covering problem. Then, we developed several techniques to adapt this general approach to the Cartesian contour construction. Our experimental evaluation shows that our approach is both effective and efficient to concisely represent the collection of frequent itemsets. In the future, we are interested in utilizing this representation for restoration of the frequent itemset frequency, and to derive generative model for frequent itemsets.

# 8. REFERENCES


[1] Foto Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In *KDD*, pages 12–19, 2004.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, May 1993.

[3] Christan Borgelt. Apriori implementation. http://fuzzy.cs.Uni-Magdeburg.de/ borgelt/Software. Version 4.08.

[4] Douglas Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Mafia: A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504, 2005.

[5] Toon Calders and Bart Goethals. Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1):171–206, 2007.

[6] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res*, 4:233–235, 1979.

[7] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[8] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, pages 211–218, 2002.

[9] Dorit S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[10] Ruoming Jin, Muad Abu-Ata, Yang Xiang, and Ning Ruan. Effective and efficient itemset pattern summarization: regression-based approaches. In *KDD*, pages 399–407, 2008.

[11] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *SIGMOD Conference*, pages 85–93, 1998.

[12] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. 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, 2007.

[13] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[14] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. Hopi: An efficient connection index for complex xml document collections. In *EDBT*, pages 237–255, 2004.

[15] Chao Wang and Srinivasan Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD*, pages 730–735, 2006.

[16] Yang Xiang, Ruoming Jin, David Fuhry, and Feodor F. Dragan. Succinct summarization of transactional databases: an overlapped hyperrectangle scheme. In *KDD*, pages 758–766, 2008.

[17] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting redundancy-aware top-k patterns. In *KDD*, 2006.

[18] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In *VLDB*, 2005.

[19] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In *KDD*, 2005.