Block Interaction: A Generative Summarization Scheme for Frequent Patterns

Ruoming Jin[1] Yang Xiang[2] Hui Hong[1]
[1]Kent State University, Kent, OH 44242
{jin, hhong}@cs.kent.edu
[2]The Ohio State University, Columbus, OH 43210
yxiang@bmi.osu.edu

ABSTRACT

Frequent pattern mining is an essential tool in the data miner's toolbox, with data applications running the gamut from itemsets, sequences, trees, to graphs and topological structures. Despite its importance, a major issue has clouded the frequent pattern mining methodology: the number of frequent patterns can easily become too large to be analyzed and used. Though many efforts have tried to tackle this issue, it remains to be an open problem. In this paper, we propose a novel *block-interaction* model to answer this call. This model can help summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets. At the heart of our approach is a set of core blocks, each of which is the Cartesian product of a frequent itemset and its support transactions. Those core blocks interact with each other through two basic operators (horizontal union and vertical union) to form the complexity of frequent patterns. Each frequent itemset can be expressed and its frequency can be accurately recovered through the combination of these core blocks. This is also the first complete generative model for describing the formation of frequent patterns. Specifically, we relate the problem of finding a minimal block-interaction model to a generalized set-cover problem, referred to as the graph set cover (GSC) problem. We develop an efficient algorithm based on GSC to discover the core blocks. A detailed experimental evaluation demonstrates the effectiveness of our approach.

1. INTRODUCTION

Frequent pattern mining is an essential tool in the data miner's toolbox, with data applications running the gamut from itemsets, sequences, trees, to graphs and topological structures [2, 23, 3, 29, 4]. Frequent pattern mining not only works by itself to discover hidden patterns from relational or structured data, but also serves as a basis for other data mining techniques, including association rules, clustering, classification, indexing, etc. Over the years, the data mining community has achieved great success in developing efficient and scalable mining algorithms to discover frequent patterns (See [13] for a state-of-the-art review).

However, a major issue has clouded the frequent pattern mining methodology from the very beginning: the number of frequent patterns can easily become too large to be analyzed and used. Many efforts have been made in attempting to reduce the number of frequent patterns, especially for itemsets (which can be generalized to other pattern types). An early reduction approach considered how to utilize "constraints" to filter out non-essential itemsets. Well-known examples include maximal frequent patterns [18], closed frequent patterns [20], non-derivable itemsets [9], and δ-cluster notation [26]. A weakness of this approach is that they either cannot recover the frequency of individual frequent itemsets (like maximal frequent patterns), or can still produce a very large number of patterns. Another approach, which includes top-k frequent patterns [14], top-k redundancy-aware patterns [25], and error-tolerant patterns [28], tries to rank the importance of individual patterns, or to revise the frequency concept to reduce the number of frequent patterns. However, these methods generally do not provide a good representation of the collection of frequent patterns.

The most recent approach is referred to as pattern summarization, which aims to offer 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). A key open question mentioned in this work is how to derive the support information for an frequent itemset from such summarization. Several works have applied probabilistic inference to restore or browse the frequency of frequent itemsets [16, 27, 22, 21]. However, contrary to the original goal of [1], these works do not directly contribute to the reduction of frequent patterns.

Can we summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets? In this paper, we provide a novel *block-interaction* model to answer this call. At the heart of our approach is a set of core blocks, also referred to as hyperrectangles [24], or tiles [11, 12], each of which is the Cartesian product of a frequent itemset and its support transactions. Those core blocks interact with each other through two basic operators (horizontal union and vertical union) to form the complexity of frequent patterns. Each frequent itemset can be expressed and its frequency can be accurately recovered through the combination of these core blocks. This is the first complete generative model for describing the formation of frequent patterns. An additional benefit is that this model also provides a simple explanation of each frequent itemset based on a small set of core blocks and two basic operators. Finally, this generative model naturally reduces the entire collection of frequent itemsets to a very small core set. Note that in [17], we developed an approach to provide a generative view of an entire collection of itemsets. However, that model, similar to [1], could not recover support information for an individual itemset. Besides, the optimization goal of [17] is not to reduce the number of frequent itemsets to a small core set, but to minimize the representation or storage cost of maximal itemsets.

The contributions of this paper are as follows.

1. We develop the first generative model to describe the formation of frequent itemsets. This model not only reduces the entire collection of frequent itemsets to a small number of core patterns, but also it can express each frequent itemset

and recover its frequency.

2. We relate the problem of finding a small set of core patterns to a generalized set-cover problem, referred to as the Graph Set Cover (GSC) problem. We develop an efficient algorithm utilizing GSC to compute the small set of core patterns.

3. We have performed a detailed experimental evaluation; our experimental results demonstrate the effectiveness of our approach.

2. BLOCK-INTERACTION MODEL AND PROBLEM DEFINITION

In this section, we formally introduce the *block-interaction* model for a collection of frequent itemsets F_α with α being the minimum support level.

Let the transaction database DB be represented as a binary matrix, i.e., a cell (i, j) is 1 if a transaction i contains item j; otherwise 0. For convenience, we also denote the database DB as the set of all cells which are 1, i.e., $DB = \{(i, j) : DB[i, j] = 1\}$. Let the block B be the Cartesian product of a transaction set T and an itemset I, i.e., $B = T \times I = \{(i, j) : i \in T \text{ and } j \in I\}$, such that $B \subseteq DB$. In other words, each block is an all-one submatrix of the binary matrix of DB. For a block B, we also denote $T(B)$ as the transaction set of B and $I(B)$ as the itemset of B, i.e., $T(B) = T$ and $I(B) = I$ for $B = T \times I$.

The block-interaction model contains a set of *core blocks* and two simple *block operators*, the block vertical union ($\varobar$) and the block horizontal union ($\ominus$) operators. Those core blocks and operators can provide a generative view of a collection of frequent itemsets and derive a concise explanation for each itemset.

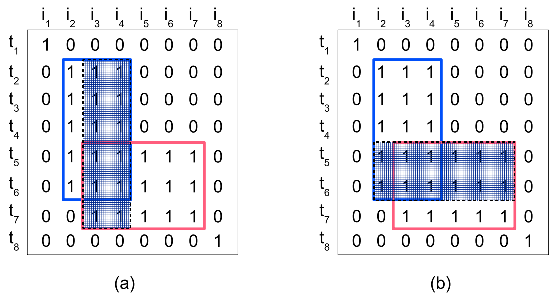
Core Blocks: Given a transaction database DB, the block-interaction model contains a small set of core blocks, denoted as $\mathcal{B} = \{B_1, B_2, \cdots, B_p\}$, where B_i is a block of transaction database DB.

Block Vertical Union ($\varobar$): Given two blocks $B_1 = T_1 \times I_1$ and $B_2 = T_2 \times I_2$, the block vertical union operator generates a new block with the itemset being the *intersection* of two itemsets $I_1 \cap I_2$ and the transaction set being the *union* of two transaction sets $T_1 \cup T_2$ (Figure 1(a)):

$$B_1 \varobar B_2 = T_1 \times I_1 \varobar T_2 \times I_2 = (T_1 \cup T_2) \times (I_1 \cap I_2)$$

Block Horizontal Union ($\ominus$): Given two blocks $B_1 = T_1 \times I_1$ and $B_2 = T_2 \times I_2$, the block horizontal union operator generates a new block with the itemset being the *union* of two itemsets $I_1 \cup I_2$ and the transaction set being the *intersection* of two transaction sets $T_1 \cap T_2$ (Figure 1(b)):

$$B_1 \ominus B_2 = T_1 \times I_1 \ominus T_2 \times I_2 = (T_1 \cap T_2) \times (I_1 \cup I_2)$$



i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8
(a)
(b)

Figure 1: Block Union

Clearly, for any two blocks B_1 and B_2 of transaction database DB, their vertical union ($B_1 \varobar B_2$) and horizontal union ($B_1 \ominus B_2$) are also blocks of DB. It is also easy to observe the following properties of these two operators (the proof is omitted for simplicity).

LEMMA 1. *Both operators satisfy the commutative, associate, and distributive properties, i.e.,*

$$B_1 \varobar B_2 = B_2 \varobar B_1, \quad B_1 \varobar (B_2 \varobar B_3) = (B_1 \varobar B_2) \varobar B_3,$$
$$B_1 \ominus B_2 = B_2 \ominus B_1, \quad B_1 \ominus (B_2 \ominus B_3) = (B_1 \ominus B_2) \ominus B_3,$$
$$B_1 \varobar (B_2 \ominus B_3) = (B_1 \varobar B_2) \ominus (B_1 \varobar B_3),$$
$$B_1 \ominus (B_2 \varobar B_3) = (B_1 \ominus B_2) \varobar (B_1 \ominus B_3)$$

Given the core blocks $\mathcal{B}$ and the two block operators, we can recursively generate a large set of blocks. Formally, we introduce the closure of $\mathcal{B}$, denoted $\mathcal{P}(\mathcal{B})$, as the set of all blocks generated by the combination of core blocks $\mathcal{B}$ using the two operators:

Block Closure of $\mathcal{B}$, $\mathcal{P}(\mathcal{B})$: 1) for any block $B \in \mathcal{B}$, $B \in \mathcal{P}(\mathcal{B})$; 2) If $B_1 \in \mathcal{P}(\mathcal{B})$ and $B_2 \in \mathcal{P}(\mathcal{B})$, then $B_1 \varobar B_2 \in \mathcal{P}(\mathcal{B})$; 3) If $B_1 \in \mathcal{P}(\mathcal{B})$ and $B_2 \in \mathcal{P}(\mathcal{B})$, then $B_1 \ominus B_2 \in \mathcal{P}(\mathcal{B})$).

Now, we relate the frequent itemset $I \in F_\alpha$ to the set of core blocks $\mathcal{B}$ and its closure $\mathcal{P}$. We define $supp(I)$ to be the support of itemset I in the transaction database DB.

DEFINITION 1. *(**Block Support**) Given an itemset I, if a block B in $\mathcal{P}(\mathcal{B})$ subsumes I, i.e.,$I \subseteq I(B)$, and if $|T(B)| \geq (1-\epsilon)supp(I)$, where ϵ is a user-preferred accuracy level for support recovery, then we say I is **supported** or **explained** by block B, denoted as $B \models I$. For a given set of frequent itemset F_α, and a set of core blocks $\mathcal{B}$, if any itemset I in F_α is supported by at least one block B in the closure $\mathcal{P}$ of $\mathcal{B}$, then we say that F_α is supported or explained by $\mathcal{B}$ or $\mathcal{P}$, denoted as $\mathcal{B} \models F_\alpha$ or $\mathcal{P} \models F_\alpha$.*

Given this, we can see that the block-interaction model provides a generative view of any frequent itemset by utilizing the two operators on top of a small set of core blocks. Indeed, each block in the block closure can be written as an expression of core blocks and block operators. However, the potential problem is that such an expression may be too complex or too unnatural to provide explanatory power. For instance, a straightforward but extreme interaction model would be to consider each single item-transaction pair in the database as a core block, i.e., $\mathcal{B} = DB$. Alternatively, we could treat each single column (each item with all its support transactions) as a core block. Clearly, these sets of core blocks neither provide a satisfactory generative view of the collection of frequent itemsets nor are able to concisely summarize them. The underlying reason for such an issue is that the core blocks do not immediately relate to the frequent itemsets and the block expression can be too complex.

In order to derive a meaningful block-interaction model, we consider constraining the complexity of each block expression for supporting or explaining an itemset. Specifically, we introduce the concepts of (2×2)-*block support* (read "two by two") and (2×2)-*block interaction model.*

DEFINITION 2. *((2×2)**-Block Support**) Given an itemset I, if a block B in the block closure $\mathcal{P}$ supports I, i,e., $B \models I$, and if this block can be expressed in the following format,*

$$B = (B_1 \ominus B_2) \varobar (B_3 \ominus B_4), \tag{1}$$

where $B_1, B_2, B_3, B_4 \in \mathcal{B}$ are core blocks, then we say I is (2×2)-block supported by $\mathcal{B}$. If each itemset I in F_α is (2×2)-block supported by $\mathcal{B}$, then we say $\mathcal{B}$ is a (2×2)-block interaction model for F_α.

Note that in our (2×2)-block support definition, the four blocks, B_1, B_2, B_3 and B_4, are not necessarily different. In other words, some or all may correspond to the same block. For instance, B_1 and B_2 can be the same: $B_1 = B_2$. Figure 2 illustrates the block expression for the 2×2 block support (and interaction model). Note that we could also define the 2×2 block support based on the block expression $(B_1 \obar B_2) \ominus (B_3 \obar B_4)$. The methodology developed in this paper can be naturally adopted for such a model as well. Due to space limitation, we focus our discussion the first model. In general, we may further relax the constraint to consider more operators in the block expression. However, relaxation will increase the expression complexity. The 2×2 model is the most concise one which allows both operators to be utilized in the block expression in a symmetric fashion.

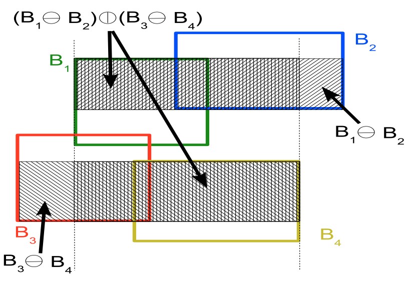


Figure 2: 2×2 **block interaction** $(B_1 \ominus B_2) \obar (B_3 \ominus B_4)$

Given the (2×2) block interaction model, our goal is to provide a generative view of an entire collection of itemsets F_α using only a small set of core blocks $\mathcal{B}$.

DEFINITION 3. **(Minimal** (2×2)**-Block Interaction Model***)* *Let the complexity of the* (2×2)*-block interaction model be defined as the number of core blocks,* $|\mathcal{B}|$*. Given a collection of frequent itemset* F_α*, we seek the most concise* (2×2)*-block interaction model to explain* F_α*, i.e.,*

$$\arg\min_{|\mathcal{B}|} \mathcal{B} \models F_\alpha$$

3. THEORETICAL BASIS

In this section, we first study the NP-hardness of the *minimal* (2×2)*-block interaction model* (Subsection 3.1) and then we introduce a new variant of the set cover problem, referred to as the *graph set cover* problem (GSC for short), which forms the basis of our algorithm to identify the minimal (2×2)-block interaction model (Subsection 3.2). In the following sections (Section 4 and Section 5, we discuss how to transform our problem into GSC problem and how to solve it efficiently.

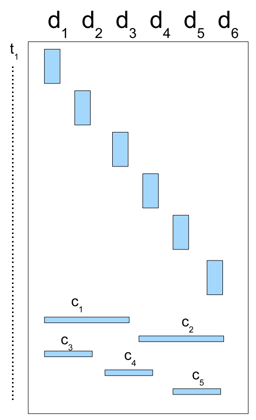
3.1 Hardness Results

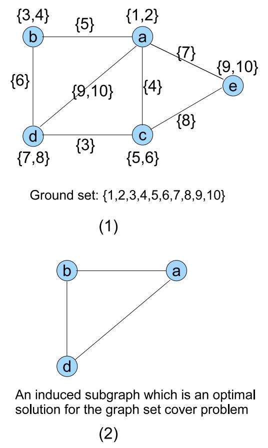
THEOREM 1. *Given a transaction database* DB *and a collection of frequent itemsets* F_α*, it is NP-hard to find a minimal* (2×2)*-block interaction model.*

Proof: To prove this theorem, we reduce the set cover problem, which is well-known to be NP-hard, to this problem.

The minimum set cover problem can be formulated as follows: Given a collection C of subsets of a finite set D, what is the $C' \subseteq C$ with minimum $|C'|$ such that every element in D belongs to at least one member of C'? We assume that there is no set $c \in C$ that covers D completely, otherwise the solution is trivial. We also assume each set c is unique in C.

The reduction utilizes the database DB, whose entire set of items is D, i.e., each element in D corresponds to a unique item in DB. For each item $d \in D$, we create k ($k > |C|$) *unique* transactions in DB such that each transaction contains only item d. These transactions with the item make up a single-item block. For each itemset $c \in C$, we create 1 *unique* transaction in DB such that it contains all items in c but nothing else. This transaction with its itemset c makes up a single-transaction block. Let $\alpha = \frac{k}{k+|C|}$ and $\epsilon = 1 - \frac{1}{k+|C|}$. Clearly the reduction takes polynomial time. A example of such reduction is illustrated in Figure 3.





{3,4}
{5}
{1,2}
b
a
{7}
{9,10}
e
{6}
{9,10}
{4}
{8}
d
{3}
c
{7,8}
{5,6}
Ground set: {1,2,3,4,5,6,7,8,9,10}
(1)
b
a
d
An induced subgraph which is an optimal solution for the graph set cover problem
(2)

Figure 3: $D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$, $C = \{c_1, c_2, c_3, c_4, c_5\}$, $c_1 = \{d_1, d_2, d_3\}$, $c_2 = \{d_4, d_5, d_6\}$, $c_3 = \{d_1, d_2\}$, $c_4 = \{d_3, d_4\}$, $c_5 = \{d_5, d_6\}$

Figure 4: An example of GSC problem

Because $\alpha = \frac{k}{k+|C|}$, we conclude that F_α of DB contains $|D|$ frequent itemsets, each of which is an element in D. We also conclude that any block $T \times I$ can support a frequent itemsets d where $d \in I$, due to $\epsilon = 1 - \frac{1}{k+|C|}$.

Then we claim that the minimal (2×2)-block interaction model for DB implies a solution for the minimum set cover problem.

This claim can be proved by combining the following lemmas:

LEMMA 2. *For any block* $T \times I$ *of* DB*, there exists a set* $c \in C$ *such that* $I \subseteq c$*.*

LEMMA 3. *For each item* d *of* DB*, there must exist a block* $T \times I$ *in the minimal* (2×2)*-block interaction model such that* $d \in I$*.*

Let 'base blocks' denote those single-item and single-transaction blocks created during the reduction. Lemma 2 can be proved by showing that block horizontal union cannot create a new block that is not covered by any base block. Lemma 3 is correct according to the fact that any single item is frequent and must be supported by some block. Due to space limits, we omit the proof details for these lemmas in this paper. □

3.2 Graph Set Cover Problem

In this subsection, we introduce a new variant of the set cover problem, which is closely related to the minimal (2×2)-block interaction model and is utilized in solving our problem.

DEFINITION 4. **(Graph Set Cover (GSC) Problem)** *Let U be the ground set and let $G = (V, E)$ be a graph, where each vertex and each edge is associated with a collection of elements in U: for a vertex v, let $S(v)$ be the elements associated with $v \in V$; for an edge (u, v), let $S(u, v)$ be the elements associated with $(u, v) \in E$ $(S(v), S(u, v) \subseteq U)$. Given a set of vertices $V_s \subseteq V$, let $G[V_s] = (V_s, E_s)$ be the subgraph induced by vertices V_s, and let $S(G[V_s])$ be the elements covered by the induced subgraph $G[V_s]$, i.e.,*

$$S(G[V_s]) = \bigcup_{v \in V_s} S(v) \bigcup_{(u,v) \in E_s} S(u, v). \qquad (2)$$

Given this, the GSC problem seeks the smallest number of vertices, such that their induced subgraph can cover all the elements in the ground set, i.e.,

$$\arg\min_{|V_s|} S(G[V_s]) = U. \qquad (3)$$

Figure 4 shows an example of GSC, where the ground set $U = \{1, 2, \cdots, 10\}$, and both vertices and edges in the graphs associate with a subset of U. The induced subgraph of vertices a, c, and d can cover U and is the optimal solution of this GSC problem.

It is not hard to see that the GSC problem is a generalization of the classical set cover problem [10]. Indeed, if we consider only the vertices are assigned with elements and the edges are not, then, this problem is a simple set cover problem. This also immediately shows the NP-hardness of this problem.

THEOREM 2. *The GSC problem is NP-hard.*

This problem is more closely related to the recently proposed *set-cover-with-pairs problem* [15].

DEFINITION 5. **(Set-Cover-with-Pairs Problem)** *Let U be the ground set and let $S = \{1, \ldots, M\}$ be a set of objects. For every $\{i, j\} \subseteq S$, let $\mathcal{C}(i, j)$ be the collection of elements in U covered by the pair $\{i, j\}$. The objective of the set cover with pairs (SCP) problem is to find a subset $S' \subseteq S$ such that*

$$\mathcal{C}(S') = \bigcup_{\{i,j\} \subseteq S'} \mathcal{C}(i, j) = U$$

with a minimum number of objects.

Note that we consider each object in SCP and each vertex in the GSC problem to be unweighted. Both can be easily reformulated as weighted versions but that is beyond scope of this work. It is easy to see that the SCP problem is also a special case of the GSC problem. Here, the input of SCP is a complete graph where each edge associates with a subset of the ground set U (not vertices). As demonstrated in [15], the approximation solution for SCP is much harder than the classical set cover problem, where a logarithmic bound approximation is available. The best known approach for the SCP is the following greedy algorithm:

Let R be the set of objects already covered, where $R = \emptyset$ initially; then at each iteration, we select the minimum of these two choices: 1) a node i such that $\frac{1}{|\mathcal{C}(R \cup \{i\}) \setminus \mathcal{C}(R)|}$ is minimum; and 2) a pair of nodes i and j such that $\frac{2}{|\mathcal{C}(R \cup \{i,j\}) \setminus \mathcal{C}(R)|}$ is minimum. We repeat such iteration until all elements are covered ($R = U$).

This greedy algorithm has been proved to yield an $O(\sqrt{N \log N})$ approximation ratio, where N is the cardinality of the ground set $N = |U|$, for the cardinality SCP problem. Note that we can directly employ this algorithm to the GSC problem as follows. We add a virtual vertex v_0 in the GSC problem, and then we link each vertex in the original graph with this virtual vertex. After that, we also move the sets of elements associated with each vertex $S(v)$ to the corresponding new edge, i.e., $S(v, v_0)$. We also assume the cost of virtual vertex v_0 is zero. Thus, we transform our GSC problem into a SCP problem.

Given this, to solve the GSC problem, we can always *1) cover the virtual vertex at the first step since there is no cost associated with it; 2) employ the greedy algorithm for SCP to complete the covering process*. Clearly, the approximation bound achieved by this procedure is no worse than $O(\sqrt{N \log N})$ (considering the optimal first step is always given by covering the virtual vertex).

4. A TWO-STAGE APPROACH

In this section, we describe an overall approach based on the GSC problem to generate a minimal (2×2)-block interaction model. Since it is hard to find the overall optimal number of core blocks for a collection of frequent itemsets F_α, we consider a two-stage algorithm for this purpose. In the first stage, we seek a minimal number of blocks which use only the $\oplus$ operator to support the entire collection of frequent itemsets. In the second stage, we seek a minimal number of blocks which use only the $\ominus$ operator to represent the blocks discovered in the first stage. Before we describe each of the two stages in details (Subsection 4.1), we first discuss several simple properties which can help simplify the search for the (2×2) block interaction model.

We will first simplify our problem by reducing both the targeted frequent itemsets and by limiting the scope of candidate core blocks. First, let the targeted collection frequent itemsets be F_α. We make the following observations:

OBSERVATION 1. *Let CF_α be the set of all closed frequent itemsets with minimal support α (an itemset is closed if no superset has as high a support level, $CF_\alpha \subseteq F_\alpha$. For a set of core blocks $\mathcal{B}$, if it can support each closed itemset in CF_α under the (2×2)-block support expression, i.e., it is a (2×2) block interaction model for CF_α, then, it is a (2×2) block interaction model for F_α (and vice versa).*

This observation allows us to focus on only the closed frequent itemsets CF_α in searching for the (2×2) block interaction model.

OBSERVATION 2. *Let $\mathcal{B}$ be the set of core blocks of an (2×2)-block interaction model for the collection of closed frequent itemsets CF_α. Then, for any block $B_i = I_i \times T_i \in \mathcal{B}$, we can always expand its transaction set to be $T(I_i)$, where $T(I_i)$ includes all the transactions containing (supporting) I_i. We refer to block $I_i \times T(I_i)$ as a* **supporting block**. *We can further expand the itemset I_i of the supporting block such that $I_i \subseteq I'_i$ and $T(I'_i) = T(I_i)$ where there is no other itemset I''_i with $I'_i \subset I''_i$ and $T(I_i) = T(I'_i) = T(I''_i)$. In other words, I'_i is a closed itemset, and we refer to the expanded block as a* **closed supporting block**.

This observation confirms that in any (2×2) block interaction model, each core block can be replaced by a *closed supporting block*. Thus, to simplify our search, the only candidate core blocks we need to consider are closed blocks. In addition, since one of the goals of this work is to reduce the collection of frequent itemsets to a very small number of them, we can further limit the candidate core blocks to those whose itemsets are closed frequent itemsets (CF_α).

Formally, let the set of all candidate core blocks be $CB = \{I \times T(I) | I \in CF_\alpha\}$. Given this, we would like to search $\mathcal{B} \subseteq CB$, such that with minimal $|\mathcal{B}|$, each closed frequent itemset can be supported or explained by a (2×2) block expression:

$$\arg\min_{|\mathcal{B}|} \mathcal{B} \models CF_\alpha, \mathcal{B} \subseteq CB$$

Note that if $B_1, B_2 \in CB$, then, $I(B_1 \ominus B_2) \in CF_\alpha^2$, where $CF_\alpha^2 = \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha\}$. This observation is used in the next subsection for discovering the core blocks of a (2×2)-block interaction model.

4.1 Vertical Union ($\oplus$) and Horizontal Union ($\ominus$) Decomposition

As mentioned earlier, in order to find the set of core blocks for a (2×2) block interaction model, we consider a two-stage algorithm, where in each stage, we utilize a single type of operator ($\ominus$ or $\oplus$) to *support* a collection of (frequent) itemsets. The rationale for this two-stage approach comes from the following observation. Basically, for any closed frequent itemsets $I \in CF_\alpha$, we need a (2×2)-block expression to support it: $(B_1 \ominus B_2) \oplus (B_3 \ominus B_4)$, where B_1, B_2, B_3 and B_4 are core blocks. The optimization goal is to minimize the total number of unique core blocks. Since this is an NP-hard problem and the direct optimization is hard to achieve, we consider the following heuristics. If the total number of unique core blocks is small, then their combinations $B_1 \ominus B_2$ and $B_3 \ominus B_4$ which are used in the (2×2)-block expression in supporting each closed frequent itemset also tend to be small. This observation inspires us to divide the overall problem into two subproblems which can be solved independently: in the first stage, we seek a minimal number of blocks which use only the $\ominus$ operator to support the entire collection of frequent itemsets; in the second stage, we seek a minimal number of blocks which use only the $\oplus$ operator to support the itemsets of the blocks discovered in the first stage. Here, we describe these two stages in detail.

Stage 1 (Minimizing Vertical Union Decomposition): In the first stage, we seek a minimal number of blocks ($\mathcal{C}$) which use only the $\oplus$ operator to support the entire collection of closed frequent itemsets (CF_α). Those blocks $\mathcal{C}$ discovered in the first stage then will be decomposed using $\ominus$ operator in the second stage. Specifically, the goal of the first stage is as follows:

DEFINITION 6. (**Subproblem 1: Minimal Vertical Union Decomposition Problem**) *Given a collection of closed frequent itemset CF_α, we seek a small set of blocks, $\mathcal{C} = \{C_1, \cdots, C_m\}$, where $C_i = I_i \times T(I_i)$ and $I_i \in CF_\alpha^2$, such that each itemset $I \in CF_\alpha$ can be supported or explained by at most two blocks C_i and C_j in $\mathcal{C}$, $C_i \oplus C_j \models I$ with respect to accuracy level ϵ_1 ($\epsilon_1 \leq \epsilon$): $I \subseteq I(C_i \oplus C_j)$ and*

$$|T(C_i \oplus C_j)| \geq (1 - \epsilon_1) supp(I).$$

Now we transform Subproblem 1 into an instance of the GSC problem (Subsection 3.2). Let $U_1 = CF_\alpha$ be the ground set to be covered. Let $G_1 = (V_1, E_1)$ be the graph we will construct.

Vertex Set Construction: Each vertex in the graph G_1 corresponds to a candidate block in $\{I_i \times T(I_i) | I_i \in CF_\alpha^2\}$, i.e., $|V| = |CF_\alpha^2|$. For each vertex $v \in V_1$, let B_v be the corresponding candidate block ($B_v \in \{I_i \times T(I_i) | I_i \in CF_\alpha^2\}$). Each vertex v is assigned with a set $S(v)$ which contains contains all the closed frequent itemsets being supported or explained by B_v:

$$S(v) = \{I \in CF_\alpha | B_v \models I\}$$

i.e., $I(B_v) \subseteq I$ and $|T(B_v)| = supp(I(B_v)) \geq (1 - \epsilon_1) supp(I)$.

Edge Set Construction: For any two vertices v_1 and v_2 in G_1, let set $S(v_1, v_2)$ include all the closed frequent itemsets which are supported by $B_{v_1} \oplus B_{v_2}$ but are not supported by B_{v_1} or B_{v_2}:

$$S(v_1, v_2) = \{I \in CF_\alpha | B_{v_1} \oplus B_{v_2} \models I\} \backslash (S(v_1) \cup S(v_2)) \quad (*)$$

i.e., $I(B_{v_1}) \cap I(B_{v_2}) \supseteq I$ and $|T(B_{v_1} \oplus B_{v_2})| \geq (1 - \epsilon_1) supp(I)$ (how to efficiently compute $|T(B_{v_1} \oplus B_{v_2})|$ will be discussed in Subsection 5.1). If $S(v_1, v_2)$ is not empty, then, these two vertices v_1 and v_2 are linked and set $S(v_1, v_2)$ is assigned to the corresponding edge.

It is easy to see that each subset of vertices V_s in G_1 which covers the ground set, i.e., $S(G_1[V_s]) = U_1$, corresponds to a set of blocks $\mathcal{C} = \{B_v | v \in V_s\}$ which supports the collection of closed frequent itemsets CF_α, i.e., $\mathcal{C} \models CF_\alpha$ (a solution for subproblem 1). Further, the optimal solution for the graph set problem is also the optimal solution for subproblem 1. Finally, we note that based on the greedy algorithm described in subsection 3.2, we obtain an approximation bound $O(\sqrt{N \log N})$ with $N = |CF_\alpha|$.

Stage 2 (Minimizing Horizontal Union Decomposition): In the second stage, we will seek a minimal number of blocks ($\mathcal{B}$) to support the blocks ($\mathcal{C}$) discovered in the first stage. Formally, the goal of this stage is formally described as follows.

DEFINITION 7. (**Subproblem 2: Minimal Horizontal Union Decomposition Problem**) *Let $\mathcal{C}$ be the set of blocks discovered in the first stage, we seek a minimal number of closed supporting blocks, $\mathcal{B} = \{B_1, \cdots, B_k\}$, where $B_i = I_i \times T(I_i), I_i \in CF_\alpha$, such that the itemset $I(C)$ of each block $C \in \mathcal{C}$ is supported or explained by at most two blocks B_i and B_j in $\mathcal{B}$, $B_i \ominus B_j \models I(C)$ with respect to accuracy level $\epsilon_2 = (\epsilon - \epsilon_1)/2$, i.e., $I(C) \subseteq I(B_i \ominus B_j)$ and*

$$|T(B_i \ominus B_j)| \geq (1 - \epsilon_2) supp(I(C)).$$

Now we transform Subproblem 2 into an instance of the GSC problem. Let $U_2 = \{I(C) | C \in \mathcal{C}\}$ be the ground set to be covered, where $\mathcal{C}$ is the collection of blocks generated in the first stage. Let $G_2 = (V_2, E_2)$ be the graph for this subproblem.

Vertex Set Construction: Each vertex v in G_2 corresponds to a closed supporting block $B = I \times T(I)$, where $I \in CF_\alpha$ and is assigned with a set $S(v)$, which contains all the frequent itemsets which are supported or explained by the corresponding block of v, denoted as B_v:

$$S(v) = \{I \in U_2 | B(v) \models I\}$$

i.e., $I(B_v) \supseteq I$ and $|T(B_v)| \geq (1 - \epsilon_2) supp(I)$.

Edge Set Construction: For any two vertices v_1 and v_2 in G_2, let set $S(v_1, v_2)$ include all the itemsets in U which are supported by $B_{v_1} \ominus B_{v_2}$, but are not supported by B_{v_1} or B_{v_2}:

$$S(v_1, v_2) = \{I \in U | B_{v_1} \ominus B_{v_2} \models I\} \backslash (S(v_1) \cup S(v_2)) \quad (**)$$

i.e., $I(B_{v_1}) \cup I(B_{v_2}) \supseteq I$ and $|T(B_{v_1} \ominus B_{v_2})| \geq (1 - \epsilon_2) supp(I)$. If $S(v_1, v_2)$ is not empty, then these two vertices v_1 and v_2 are linked and set $S(v_1, v_2)$ is assigned to the corresponding edge.

Again, we observe that each subset of vertices V_s in G_2 which covers the ground set, $S(G_2[V_s]) = U_2$, corresponds to a set of blocks $\mathcal{B} = \{B_v | v \in V_s\}$ which supports each itemset in $\mathcal{C}$, and the optimal solution for the graph set problem is also the optimal solution for subproblem 2. Using the greedy algorithm described in subsection 3.2, we can obtain an approximation bound $O(\sqrt{N \log N})$ with $N = |\mathcal{C}|$.

4.2 Correctness of the Two-Stage Approach

In this subsection, we prove that the blocks discovered in subproblems 1 and 2 form the core blocks of a (2×2) block interaction model for CF_α. Especially, we demonstrate how the user-preferred accuracy level ϵ is distributed correctly into the two stages, $\epsilon_1 \leq \epsilon$ to stage 1 and $\epsilon_2 = (\epsilon - \epsilon_2)/2$ to stage 2. The effect of different choices of ϵ_1 will be studied in Section 7. To facilitate our discussion, we first make the following observations for the cardinality of the transaction set generated by the vertical and horizontal union operators.

OBSERVATION 3. *Given any two blocks $B_1 = I_1 \times T(I_1)$ and $B_2 = I_2 \times T(I_2)$, we have $|T(I_1)| = supp(I_1)$ and $|T(I_2)| = supp(I_2)$. Then, the following properties hold:*

$$\begin{aligned}
|B_1 \oplus B_2| &= |T(I_1) \cup T(I_2)| \\
&= |T(I_1)| + |T(I_2)| - |T(I_1) \cap T(I_2)| \\
&= |T(I_1)| + |T(I_2)| - |T(I_1 \cup I_2)| \\
&= supp(I_1) + supp(I_2) - supp(I_1 \cup I_2) \\
|B_1 \ominus B_2| &= |T_1 \cap T_2| = supp(I_1 \cup I_2)
\end{aligned}$$

THEOREM 3. *The set of core blocks discovered by subproblems 1 and 2 form the basis for a (2×2) block interaction model for the collection of closed frequent itemsets CF_α.*

Proof: To prove this, we need to show that for each closed frequent itemset $I \in CF_\alpha$, there are blocks B_1, B_2, B_3 and $B_4 \in \mathcal{B}$, such that $(B_1 \ominus B_2) \oplus (B_3 \ominus B_4) \models I$. Specifically, we must show that

$$(I(B_1) \cup I(B_2)) \cap (I(B_3) \cup I(B_4)) \subseteq I \text{ and}$$
$$|(B_1 \ominus B_2) \oplus (B_3 \ominus B_4)| \geq (1-\epsilon)supp(I).$$

First, we can see that from subproblem 1, for each closed itemset I, we can identify two candidate blocks $C_i = I_i \times T(I_i)$ and $C_j = I_j \times T(I_j)$ such that $I(C_i) \cap I(C_j) \subseteq I$ and

$$|T(C_i \oplus C_j)| = |T(C_i)| + |T(C_j)| - |T(I_i \cup I_j)| =$$
$$supp(I_i) + supp(I_j) - supp(I_i \cup I_j) \geq (1-\epsilon_1)supp(I). \quad (4)$$

Then from subproblem 2, for both I_i and I_j, we can discover blocks B_1, B_2 for I_i and B_3, B_4 for I_j, such that $I(B_1) \cup I(B_2) \subseteq I_i$ and $I(B_3) \cup I(B_4) \subseteq I_j$, and

$$|T(B_1 \ominus B_2)| \geq (1-\epsilon_2)supp(I_i) \quad (5)$$
$$|T(B_3 \ominus B_4)| \geq (1-\epsilon_2)supp(I_j) \quad (6)$$

where $\epsilon_2 = (\epsilon - \epsilon_1)/2$. To show B_1, B_2, B_3 and B_4 can explain I, we can easily see

$$I((B_1 \ominus B_2) \oplus (B_3 \ominus B_4)) =$$
$$(I(B_1) \cup I(B_2)) \cap (I(B_3) \cup I(B_4)) \supseteq I_i \cap I_j \supseteq I$$

In addition, the following inequality holds:

PROPOSITION 4. $|T((B_1 \ominus B_2) \oplus (B_3 \ominus B_4))| \geq (1-\epsilon)supp(I)$

Proof:

$$\begin{aligned}
&|T((B_1 \ominus B_2) \oplus (B_3 \ominus B_4))| = \\
&|T(B_1 \ominus B_2)| + |T(B_3 \ominus B_4)| - |T(I(B_1 \ominus B_2) \cup I(B_3 \ominus B_4))| \geq \\
&\text{(See (5)\&(6))} \\
&(1-\epsilon_2)supp(I_i) + (1-\epsilon_2)supp(I_j) - \\
&|T(I(B_1 \ominus B_2) \cup I(B_3 \ominus B_4))| \geq \\
&(1-\epsilon_2)supp(I_i) + (1-\epsilon_2)supp(I_j) - |T(I_i \cup I_j)| \geq \\
&(I_i \subseteq I(B_1 \ominus B_2) \& I_j \subseteq I(B_3 \ominus B_4) \Rightarrow \\
&I_i \cup I_j \subseteq I(B_1 \ominus B_2) \cup I(B_3 \ominus B_4) \Rightarrow \\
&|T(I_i \cup I_j)| \geq |T(I(B_1 \ominus B_2) \cup I(B_1 \ominus B_2))|) \\
&(1-\epsilon_2)supp(I_i) + (1-\epsilon_2)supp(I_j) - |T(I_i \cup I_j)| = \\
&supp(I_i) + supp(I_j) - \epsilon_2 supp(I_i) - \epsilon_2 supp(I_j) - |T(I_i \cup I_j)| \geq \\
&\text{(See (4))} \\
&(1-\epsilon_1)supp(I) - \epsilon_2 supp(I_i) - \epsilon_2 supp(I_j) \geq \\
&(I_i \cap I_j \supseteq I \Rightarrow supp(I) \geq supp(I_1) \wedge supp(I) \geq supp(I_2) \& \\
&\epsilon_2 = \frac{\epsilon - \epsilon_1}{2}) \\
&(1-\epsilon_1)supp(I) - \frac{\epsilon - \epsilon_1}{2} supp(I) * 2 \\
&\geq (1-\epsilon)supp(I)
\end{aligned}$$

□

Therefore, $(B_1 \ominus B_2) \oplus (B_3 \ominus B_4) \models I$.

□

5. FAST ALGORITHM FOR CORE BLOCK DISCOVERY

In this section, we present the complete two-stage algorithm (Subsection 5.2) for discovering the core blocks of a (2×2)-block interaction model. In particular, we introduce several heuristics to deal with the scalability issues (Subsection 5.1).

5.1 Scalability Issues and Heuristics

As discussed in Section 4, our approach for discovering the minimal (2×2) block interaction model contains two stages: the vertical union decomposition and horizontal union decomposition stages. Each stage involves two major computational steps: 1) constructing the instance of a GSC problem and then 2) invoking the greedy GSC algorithm. However, both steps can be computationally expensive.

GSC construction: Let us look at the graph construction for the first stage. The vertex set V_1 corresponds to $CF_\alpha^2 = \{I_i \cup I_j | I_i, I_j \in CF_\alpha\}$ which includes the pairwise combination of any two closed frequent itemsets. To construct edge set E_1, we try to join any two vertices which involves $O(|CF_\alpha^2|^2)$ computational complexity. Even though $|CF_\alpha^2| << |CF_\alpha|^2$, this is still rather expensive.

A particular difficulty exists in computing $S(v_1, v_2)$, which needs to determine if $|T(B_{v_1} \oplus B_{v_2})| \geq (1-\epsilon_1)supp(I)$ for any frequent closed itemset I, where $B_{v_1} = I_1 \times T(I_1)$ and $B_{v_2} = I_2 \times T(I_2)$ $(I_1, I_2 \in CF_\alpha^2)$. We can utilize Observation 3 to solve this problem: $|T(B_{v_1} \oplus B_{v_2})| = supp(I_1) + supp(I_2) - supp(I_1 \cup I_2)$. Therefore, we need precompute the support for CF_α^2 and $CF_\alpha^4 = \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha^2\}$. Clearly, this can be very costly.

To deal with this problem, we employ two heuristics to effectively reduce the candidate block search space. The first heuristic considers the *support constraints* on the candidate blocks. If an itemset in CF_α^2 has lower support, then its likelihood to combine with other itemsets for recovering the supports of frequent itemsets may be smaller. To compensate, we may request each candidate itemset itself should cover at least one frequent itemset. This suggests that each itemset in CF_α^2 should have a support no less than $(1-\epsilon)\alpha$. In addition, since each candidate block can be improved by a corresponding *closed supporting block* (Observation 2), we further focus on only those closed itemsets in CF_α^2 instead of all the members. In sum, the set of candidate blocks in stage 1 is written as $CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$.

Interestingly, we note that graph construction for the second stage has a lesser scalability issue because its vertex set V_2 corresponds to CF_α, which is smaller than CF_α^2 in the first stage. In addition, only $|\mathcal{C}|$ blocks need to be covered, which can be orders of magnitude less than CF_α for the first stage.

GSC Greedy Algorithm: Now we take a look of the GSC algorithm based on the greedy algorithm for the set-cover-with-pairs (SCP) problem. In this greedy algorithm, we consider any pair of vertices in the graph as a candidate for covering. Actually, we only need to consider pairs of vertices that are adjacent (no improvement over single vertex if they are not connected), but this still results in $O(|E_1|)$ and $O(|E_2|)$ time complexity for each iteration of the GSC algorithm in the first and second stage, respectively.

It is interesting to observe the following properties on the GSC problem for the first stage.

LEMMA 4. *In graph $G_1 = (V_1, E_1)$ of the first stage, for any edge set $S(v_1, v_2)$, there is a vertex set $S(v_3)$, such that $S(v_1, v_2) \subseteq$*

$S(v_3)$.

Proof: For edge (v_1, v_2) that covers any frequent itemsets in CF_α, then $|T(I_{v_1}) \cup T(I_{v_2})| \geq (1-\epsilon)\alpha$. Since our candidate blocks in the first stage consider at least $CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$, it means there is an itemset I_{v_3} such that $I_{v_3} \subseteq I_{v_1} \cap I_{v_2}$ and $supp(I_{v_3}) = supp(I_{v_1} \cap I_{v_2}) \geq |T(I_{v_1}) \cup T(I_{v_2})|$, i.e., I_{v_3} is the corresponding closed items of $I_{v_1} \cap I_{v_2}$. □

A similar observation can be made for $G_2 = (V_2, E_2)$ for the second stage. In these types of SCP (i.e., for each edge set $S(v_1, v_2)$, there is a vertex set $S(v_3)$ that covers at least as much), we observe that the ratio between the price of choosing an optimal vertex $\frac{1}{|\mathcal{C}(R\cup\{i\})\backslash\mathcal{C}(R)|}$ and the price of choosing an optimal pair of vertices (an edge) $\frac{2}{|\mathcal{C}(R\cup\{i,j\})\backslash\mathcal{C}(R)|}$ is no more than $3/2$ (Greedy algorithm for SCP, Section 3). This basically suggests that the benefit of choosing a pair (an edge) may be limited. Therefore, we can simplify and speed up the GSC algorithm as follows:

1. *Let R be the set of objects already covered, $R = \emptyset$ initially;*
2. *At each iteration, we select a vertex v such that $|S(G[R \cup \{i\}])\backslash S(G[R])|$ is maximum;*
3. *We repeat 2) until all elements in U are covered.*

This algorithm is referred as *GraphSetCover*.

5.2 Algorithm Description

```
Algorithm 1 CoreBlockDiscovery($DB$,$CF_\alpha$, $\epsilon$)
Require: $CF_\alpha$: frequent closed itemsets in $DB$ with minimum support $\alpha$
Require: $\epsilon$: user-defined accurracy level
    {Stage 1: Block Vertical Union ($\oplus$) Decomposition}
 1: $CF_\alpha^2 \leftarrow \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha\}$;
 2: $CF \leftarrow CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$ {reducing the candidate blocks};
 3: $CF^2 \leftarrow \{I_1 \cup I_2 | I_1, I_2 \in CF\}$ ;
 4: ComputeSupport($CF^2 \backslash CF_{(1-\epsilon)\alpha}$); {Compute support for each itemset in $CF$}
    {Vertex Set Construction:}
 5: for all $I_v \in CF$ do
 6:     $V_1 \leftarrow V_1 \cup \{(I_v, supp(I_v)\}$;
 7:     $S(v) \leftarrow \{I \in CF_\alpha | I \subseteq I_v \wedge supp(I_v) \geq (1-\epsilon_1)supp(I)\}$;
 8: end for
    {Edge Set Construction:}
 9: for all $(I_1, I_2) \in CF \times CF$ do
10:     $S(v_1, v_2) \leftarrow \{I \in CF_\alpha | I_1 \cap I_2 \supseteq I \wedge supp(I_1) + supp(I_2) - supp(I_1 \cup I_2) \geq (1-\epsilon_1)supp(I)\} \backslash (S(v_1) \cup S(v_2))$;
11:     if $S(v_1, v2) \neq \emptyset$ then
12:         $E_1 \leftarrow E_1 \cup \{(v_1, v_2)\}$;
13:     end if
14: end for
15: $\mathcal{C} \leftarrow$ GraphSetCover($G_1(V_1, E_1), CF_\alpha$);
    {Stage 2: Block Horizontal Union ($\ominus$) Decomposition}
16: $U \leftarrow \{I(C) | C \in \mathcal{C}\}$;
    {Vertex Set Construction:}
17: for all $I_v \in CF_\alpha$ do
18:     $V_2 \leftarrow V_2 \cup \{(I_v, supp(I_v)\}$;
19:     $S(v) \leftarrow \{I \in U | I \subseteq I_v \wedge supp(I_v) \geq (1-\epsilon_2)supp(I)\}$;
20: end for
    {Edge Set Construction:}
21: for all $(I_1, I_2) \in CF_\alpha \times CF_\alpha$ do
22:     $S(v_1, v_2) \leftarrow \{I \in U | I_1 \cup I_2 \supseteq I \wedge supp(I_1 \cup I_2) \geq (1-\epsilon_2)supp(I)\} \backslash (S(v_1) \cup S(v_2))$;
23:     if $S(v_1, v2) \neq \emptyset$ then
24:         $E_2 \leftarrow E_2 \cup \{(v_1, v_2)\}$;
25:     end if
26: end for
27: $\mathcal{B} \leftarrow$ GraphSetCover($G_2(V_2, E_2), U$);
```

Algorithm 1 sketches the complete two-stage approach (including both vertical union ($\oplus$) and horizontal union ($\ominus$) decomposition, Section 4.1).

Lines 1 to 15 describe Stage 1 of Algorithm 1 for vertical union decomposition. We first generate the candidate itemsets and their supports in Lines 1 to 4. Note that to compute the support for candidate itemsets in $CF^2 \backslash CF_{(1-\epsilon)\alpha}$, we simply organize those itemsets in a prefix tree and then count their occurrences by enumerating the subsets in each transaction. We refer to this procedure as *ComputeSupport* (details omitted). We then construct the vertex set of G_1 (Lines 5 to 8) and its edge set (Lines 9 to 13. Once the graph G_1 is constructed, the GSC algorithm is invoked to cover CF_α using core blocks in $\mathcal{C}$.

Similarly, Stage 2 of Algorithm 1 for horizontal union decomposition is described from Lines 16 to 27. Here, the ground set U (to be covered) includes all the itemsets of blocks generated by Stage 1. We then construct the vertex set and edge set of G_2 (Lines 17 to 25) Finally, after the graph G_2 is constructed, the GSC algorithm is called to find the final core blocks $\mathcal{B}$, which is also the core blocks of the entire (2×2)-block interaction model (Line 27).

Computational Complexity: Algorithm 1 consists of four parts: (1) vertex construction of G_1, (2) edge construction of G_1, (3) vertex construction of G_2, and (4) edge construction of G_2, plus graph set cover for G_1 and G_2. The total time complexity for the four parts is $O(|CF|f(|CF_\alpha|)) + O(|CF|^2|f(CF_\alpha|)) + O(|CF_\alpha|f(|U|)) + O(|CF_\alpha|^2 f(|U|)) = O(|CF|^2 f(|CF_\alpha|) + |CF_\alpha|^2 f(|CF|))$. Here $f()$ is the cost of function to check if any itemset in CF_α is included in a vertex itemset or edge itemset. The naĩve implementation based on a linear scan yields $f(|CF_\alpha|) = |CF_\alpha|$. However, we can improve this procedure by treating CF_α as a transactional database where each transaction is an itemset. Then, we can organize it by the vertical format, i.e., for each item, which transactions contains it. Thus, the checking process can be improved significantly. The time complexity of GSC may vary according to its detailed implementation. We only need to consider Stage 1 as it is more expensive than Stage 2. A simple greedy algorithm has a time complexity proportional to the size of the the graph (in our case $O(|CF|)$) and the set of the ground set (in our case $O(CF_\alpha)$) to be covered. Putting these together, the overall time complexity of Algorithm 1 is $O(|CF|^2 f(|CF_\alpha|) + |CF_\alpha|^2 f(|CF|))$.

6. RELATED WORKS

Numerous efforts have been made to reduce the number of frequent patterns, especially for itemsets. Well-known examples include maximal frequent patterns [18], closed frequent patterns [20], free-sets [5], disjunction-free sets [7], non-derivable itemsets [9, 8, 19], and δ-cluster notation [26]. Most of these methods try to identify certain rules to filter out "non-essential" itemsets. Even though some of the rules utilize disjunctive rules or deductive rules which share a certain spirit with our block-interaction modeling, their objectives do not address utilizing a small collection of basic itemsets to "generatively" explain or recover other frequent itemsets. In our scheme, even if an itemset can be explained by other itemsets, it may still serve as a core block. We note that δ-cluster also tries to apply the set cover idea to concisely represent the collection of frequent itemsets. However, it does not provide a generative view. Indeed, the δ-cluster method can be viewed as a special case of ours: a (1×1)-block interaction model with no block union operators.

In [17], we developed an approach to provide a generative view of an entire collection of itemsets. However, that model cannot recover support information for an individual itemset. Also, the optimization goal of [17] is not to reduce the number of frequent itemsets to a small core set, but to minimize the representation or storage cost of maximal itemsets. In [17], we proposed a bipartite graph set cover problem which is equivalent to a set cover problem.

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

Table 1: Datasets Characters. $\mathcal{I}$ is the total number of items and $\mathcal{T}$ is the total number of transactions

In this work, the proposed *graph set cover* problem is much more general and more difficult than the classical set cover problem.

7. EXPERIMENTAL RESULTS

In this section, we report the effectiveness of (2×2)-block interaction modeling in summarizing the frequent itemsets on both real and synthetic datasets. Specifically, we are interested in the following questions:

1. How does our block interaction model(**B.I.**) compare with the state-of-art summarization schemes, including Maximal Frequent Itemsets [18](**MFI**), Close Frequent Itemsets [20](**CFI**), Non-Derivable Frequent Itemsets [9](**NDI**), and Representative pattern [26](**δ-Cluster**).
2. How do different parameters, including α, ϵ, and ϵ_1 affect the conciseness of the block modeling, i.e., the number of core blocks?

In order to answer the above questions, we design three groups of experiments:

Group 1: In the first group of experiments, we vary the support level α for each dataset with a fixed user-preferred accuracy level ϵ (either 5% or 10%) and fix $\epsilon_1 = \frac{\epsilon}{2}$.

Group 2: In the second group of experiments, we study how user-preferred accuracy level ϵ would affect the model conciseness (the number of core blocks). Here, we vary ϵ generally in the range from 0.1 to 0.2 with a fixed support level α and $\epsilon_1 = \frac{\epsilon}{2}$.

Group 3: In the third group of experiments, we study how the distribution of accuracy level ϵ_1 in the two stages would affect the model conciseness. We vary ϵ_1 between 0.1ϵ and 0.9ϵ with fixed support level α and the overall accuracy level ϵ.

Datasets and Experimental Environment: We conducted our evaluation on a total of 5 datasets, including four real datasets and one synthetic dataset. Three of them are dense datasets and two of them are sparse datasets. All of them are publicly available on the FIMI repository [1]. The basic characteristics of the datasets are listed in Table 1. In our experiments, we use the MAFIA algorithm [6], which is publicly available online[2], to generate frequent itemsets, closed frequent itemsets and maximal frequent itemsets. Our algorithms were implemented in C++ and run on Linux 2.6 on an Intel Xeon 3.2 GHz processor with 4GB memory.

Results from Group 1: Table 2, 3, 4, and 5 show the results of Group 1 on the real datasets *connect*, *pumsb*, *chess*, and *retail*, respectively. Table 6 shows the results on the synthetic dataset *T40I10D100K*. In these experiments, we can see clearly that the (2×2) interaction model consistently produces the most concise representation for different support levels. In particular, our results show that the number of core blocks needed to represent the entire collection of frequent itemsets, including their support information, is even less than the number of maximal frequent itemsets (MFI), one of the best summarization methods for frequent itemsets, but which does not include support information.

[1]http://fimi.cs.helsinki.fi/data/
[2]http://himalaya-tools.sourceforge.net/Mafia/

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.92	175	2212	168	178	56
0.91	192	2819	184	196	56
0.90	222	3486	199	222	72
0.89	261	4218	223	279	85
0.88	313	5106	240	332	89

Table 2: Group1.Connect: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.90	259	1465	585	259	48
0.89	348	2186	763	348	82
0.88	500	3160	501	988	88
0.87	633	4508	1200	634	99
0.86	825	6245	1470	826	262

Table 3: Group1.Pumsb: $\epsilon = 0.1$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.875	74	1059	133	83	81
0.850	119	1885	172	137	82
0.825	176	3189	218	209	126
0.800	226	5083	281	288	109
0.775	325	7679	352	426	266

Table 4: Group1.Chess: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.007	167	315	317	294	136
0.006	219	417	418	391	176
0.005	284	580	582	545	241
0.004	424	831	838	783	335
0.003	692	1393	1410	1325	538

Table 5: Group1.Retail: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.032	608	685	686	685	458
0.031	645	730	731	730	472
0.030	700	793	794	793	486
0.029	741	842	843	842	495
0.028	812	924	925	924	506

Table 6: Group1.T40I10D100K: $\epsilon = 0.1$

ϵ	MFI	CFI	NDI	δ-Cluster	B.I.
0.06	222	3486	199	225	104
0.08	222	3486	199	223	50
0.1	222	3486	199	222	40
0.12	222	3486	199	222	27
0.14	222	3486	199	222	19

Table 7: Group2.Connect: $\alpha = 0.9$

ϵ	MFI	CFI	NDI	δ-Cluster	B.I.
0.06	219	417	418	390	176
0.07	219	417	418	389	175
0.08	219	417	418	389	175
0.09	219	417	418	220	233
0.1	219	417	418	389	203

Table 8: Group2.Retail: $\alpha = 0.006$

ϵ_1	MFI	CFI	NDI	δ-Cluster	B.I.
0.01	259	1465	585	259	28
0.03	259	1465	585	259	39
0.05	259	1465	585	259	48
0.07	259	1465	585	259	87
0.09	259	1465	585	259	258

Table 9: Group3.Pumsb: $\alpha = 0.9$, $\epsilon = 0.1$

ϵ_1	MFI	CFI	NDI	δ-Cluster	B.I.
0.005	219	417	418	391	216
0.015	219	417	418	391	401
0.025	219	417	418	391	176
0.035	219	417	418	391	175
0.045	219	417	418	391	175

Table 10: Group3.Retail: $\alpha = 0.006$, $\epsilon = 0.05$

Specifically, Table 2 shows the number of core blocks (core itemsets) in B.I. is on average more than 3 times smaller than the number of patterns in MFI, NDI and δ-Cluster. It is 50 times more compact than CFI. Table 3 shows that B.I. is 5 times better than MFI and δ-Cluster, 9 and 32 times better than CFI and NDI respectively. In Table 4, the result sizes of MFI, NDI and δ-Cluster are around 1.5 times more than the number of core blocks, and B.I. is 27 times better than CFI. In Table 5, the block size in B.I. is noticeably smaller than MFI, and less than half of those results produced by other methods, while in Table 6, our B.I. method achieves 1.5 times better compactness than the others.

Results from Group 2: In the second group of experiments, we fix α for each dataset and vary ϵ between 1 % and 20%. Due to the lack of space, we only provide the experimental results on two datasets, one dense dataset (Connect) and one sparse dataset (Retail). For the dense dataset *connect*, Tahle 7 shows the number of core blocks shrinks as the user-preferred accuracy level grows. Clearly, this is consistent with the intuition that less accurate recovery needs fewer blocks and more accurate recovery needs more core blocks. Specifically, for the dense dataset (Connect), the number of core blocks (itemsets) is up to more than 10 times better than MFI, NDI, and δ-Clusters, and on average 183 times smaller than CFI. Our method also works well for the sparse dataset: Table 8 shows that the number of core blocks is smaller than that of MFI, and 2 times more compact than CFI, NDI and δ-Cluster.

Results from Group 3: For the similar reason, we only report on two datasets for the third group of experiments. We first fix α to be 0.9 for the dense dataset *Pumsb*, 0.006 for the sparse dataset *Retail*, and vary ϵ_1 from 0.1ϵ to 0.9ϵ.

For the dense dataset *Pumsb*, when $\epsilon_1 = 0.7\epsilon$, B.I. is about 3 times better than MFI and δ-Cluster, 6.7 times better than NDI, and 16.8 times better than CFI. However, when ϵ_1 decreases to 0.1ϵ, B.I. performs almost 10 times better than MFI and δ-Cluster, 20 times better than NDI, and even 52 times better than CFI. Interestingly, we can observe quite different performance of B.I. on the sparse dataset *Retail* (Table 10) when ϵ_1 decreases. This suggests that ϵ_1 has very different impacts on B.I. in different datasets. While for a given dataset, we may be able to obtain a general understanding of its performance (by looking at how each stage can effectively explain their input itemsets), an analytical method for optimizing ϵ_1 for different dataset is an interesting open question.

8. CONCLUSIONS

Can we summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets? This problem is not only of practical importance, but also of theoretical interest. It has become one of the central problems in recent frequent pattern mining research. The existing methods have mainly focused on leveraging simple and basic rules to define and then remove non-essential patterns. In this work, we take a different route by focusing on different but also fundamentally important questions: *How does the complexity of frequent patterns arise? Can the large number of frequent itemsets be generated from a small number of patterns through their interactions?* In our search for a generative view of the collection of frequent patterns, we have developed the novel *block-interaction* model to concisely summarize a collection of frequent patterns. This model not only brings a new view of the frequent patterns, but also opens a set of new research questions: Should frequent patterns be a phenomenon or the targeted knowledge? What is the underlying knowledge/rules for frequent patterns? Will such knowledge (like the core block/itemsets) be more useful than the frequent pattern themselves? In addition, this model also addresses an important challenge for data mining: Given a set of mining results, how can we explain them or visualize them to the end users? The model developed here clearly gears towards such a goal. Finally, our study also opens up a list of research questions for algorithmic research, for instance, can we develop an algorithm for graph set cover with better approximation bound?

9. 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] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[5] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22, 2003.

[6] 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.

[7] Artur Bykowski and Christophe Rigotti. Dbc: a condensed representation of frequent patterns for efficient mining. *Inf. Syst.*, 28(8):949–977, 2003.

[8] T. Calders, C. Rigotti, and J-F. Boulicaut. A survey on condensed representations for frequent sets. In J-F. Boulicaut, L. de Raedt, and H. Mannila, editors, *Constraint-Based Mining*, volume 3848 of *LNCS*. Springer, 2006.

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

[10] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[11] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[12] Aristides Gionis, Heikki Mannila, and Jouni K. Seppänen. Geometric and combinatorial tiles in 0-1 data. In *PKDD*, pages 173–184, 2004.

[13] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

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

[15] Refael Hassin and Danny Segev. Proceedings of the 25th annual conference on foundations of software technology and theoretical computer science (fsttcs). In *FSTTCS*, pages 164–176, 2005.

[16] 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.

[17] Ruoming Jin, Yang Xiang, and Lin Liu. Cartesian contour: a concise representation for a collection of frequent sets. In *KDD*, pages 417–426, 2009.

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

[19] Juho Muhonen and Hannu Toivonen. Closed non-derivable itemsets. In *PKDD*, pages 601–608, 2006.

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

[21] Ardian Kristanto Poernomo and Vivekanand Gopalkrishnan. Cp-summary: a concise representation for browsing frequent itemsets. In *KDD*, pages 687–696, 2009.

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

[23] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[24] 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.

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

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

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

[28] M. T. Yang, R. Kasturi, and A. Sivasubramaniam. An Automatic Scheduler for Real-Time Vision Applications. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.

[29] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2002.