Vietnam J Comput Sci (2017) 4:261–277
DOI 10.1007/s40595-017-0096-2

REGULAR PAPER

Efficiently mining association rules based on maximum single constraints

Anh Tran[1] · Tin Truong[1] · Bac Le[2]

Received: 26 August 2016 / Accepted: 29 April 2017 / Published online: 31 May 2017
© The Author(s) 2017. This article is an open access publication

Abstract A serious problem encountered during the mining of association rules is the exponential growth of their cardinality. Unfortunately, the known algorithms for mining association rules typically generate scores of redundant and duplicate rules. Thus, we not only waste CPU but also encounter difficulties saving, managing and using the results of these algorithms. The present paper focuses on the discovery of association rules in which the left-handed and right-handed sides contain in two user-supplied maximum single constraints. If the constraints appear on or differ from a lattice of closed itemsets (together with their typically under-sized generators and supports) that have been mined and saved once, we quickly extract the corresponding frequent sub one. Using an equivalence relation based on the closure of the two rule sides, the association rule set with maximum single constraints is partitioned into disjoint equivalence classes. Without loss of generality, it is necessary to consider mining each class independently. This helps avoid the wasteful generation of numerous candidates, reduces the burden of storing the support and confidence of rules in the same class and establishes a foundation for mining algorithms in parallel and distributed environments. In each class, the rules are represented as unique and explicit via the corresponding closed itemsets and generators. Due to the low cardinality and size of the generators, mining based on these representations, which does not generate duplicates, is very efficient. In the present paper, all these theoretical results are proven mathematically and used to construct the MAR_MaxSC algorithm. The efficiency of MAR_MaxSC compared with post-processing methods for mining association rules with maximum single constraints is then verified on several characteristic databases.

Keywords Association rules · Frequent itemsets · Closed frequent itemsets · Closed lattice · Generators · Constraints

1 Introduction

Mining association rules is useful for applications that examine how often two or more items of interest co-occur. For example, in market basket analysis, we find that the set of milk, bread and eggs occur in 80% of all transactions. This customer behavior gives us clues regarding the store placement of milk, bread and eggs. Further, we can discover rules such as “the proportion of customers who buy eggs among those who bought milk and bread is 90%”. This rule can be applied toward marketing strategies; for example, the promotion of milk and bread to increase the sale of eggs. Formally, the problem of mining association rules [2] is stated as follows: Given database $\mathcal{T} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$, m $= |\mathcal{A}|$ (where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$), n $= |\mathcal{O}|$ and minimum support and confidence thresholds $s_0, c_0 \in (0; 1]$, the task is to mine association rules satisfying s_0, c_0. This problem can be solved in two steps: (1) extracting the frequent itemsets with s_0 and (2) generating association rules from these sets for c_0.

The method used to solve the above second step is simple. We first enumerate all the nonempty, proper subsets of Z: $\emptyset \subset X \subset Z$. Then, we obtain association rules of the

✉ Anh Tran
anhtn@dlu.edu.vn

Tin Truong
tintc@dlu.edu.vn

Bac Le
lhbac@fit.hcmus.edu.vn

[1] Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam

[2] Department of Computer Science, University of Science, VNU-HCMC, Ho Chi Minh, Vietnam

form X → $Z\backslash X$, compute their confidences and filter out those satisfying c_0. Hence, most researchers concentrate on mining frequent itemsets (step 1). The Apriori method proposed in [1] and a similar independently developed approach [27] were the first algorithms proposed for mining frequent itemsets. Apriori and its variants (Apriori-Hybrid [1], DHP [30]) show reliable performance on sparse databases with simple itemsets such as market databases; however, on complex databases such as those consisting of bio-sequences and telecommunication networks, they typically generate numerous candidates or require several database passes. Recently, algorithms based on frequent pattern trees (FP trees) have been developed [15,20], wherein the original database is compressed into an FP-Tree or similar tree structure. Using the divide-and-conquer and depth-first search methods, all the large itemsets are mined from the frequent 1-itemsets without requiring a second database pass. However, in interactive or incremental mining systems, where users often change the minimum support required as well as insert new transactions into the original database, FP-Tree-inspired structures are unsuitable because the trees must be rebuilt. All these algorithms work with horizontal formatted databases. In addition, the Eclat algorithm proposed by [39] executes a transaction identification set (tidset) intersection approach using a vertical data format. A modification of Eclat with "diffsets" called Declat [40] is often applied to solve frequent itemset mining tasks. For an experimental comparison between several of the frequent itemset mining algorithms, see [18].

The search space of frequent itemsets is frequently vast and grows exponentially with the number of items. In addition, the generation of frequent itemsets produces significant duplication. In particular, a low minimum support threshold can generate a huge number of frequent itemsets. For example, a frequent itemset with m items might produce 2^{m-1} subsets. Hence, mining databases that produce several long, frequent itemsets is an impossible task due to its associated computational and storage requirements. An alternative approach is to utilize condensed, lossless representations of frequent itemsets. These representations both reduce CPU and memory requirements and enable the efficient management and storage of the results generated. Two types of condensed representations are maximal and closed itemsets (and their generators). The GenMax algorithm described in [19] mines maximal itemsets using a tidset intersection approach. An Apriori-based alternative algorithm called MaxMiner [10] uses extremely effective itemset pruning with a support lower bound. Other examples can be found in [11,17]. Having low cardinality, maximal itemsets can be used to reproduce all the frequent itemsets. Unfortunately, we only know their lower support bounds. In addition, the generation of frequent itemsets from maximal itemsets may result in an intractable number of duplicates. Thus, maximal itemsets are unsuitable for association rule mining from frequent itemsets. Hence, it is necessary to find an objective solution. Indeed, the mining of closed itemsets (which is based on the lattice theoretic framework of formal concept analysis [16,23,38]) has received great attention for two reasons. First, the number of closed frequent itemsets is greater than the number of maximal frequent itemsets while typically being orders of magnitude lower than the total number of frequent itemsets. Thus, their discovery can help purge redundant itemsets. Second, the set of all closed frequent itemsets is a condensed representation because we can determine whether an itemset X is frequent as well as the exact support of X. In other words, we can generate all the frequent itemsets based on the closed frequent itemsets. This generation is very effective if we also use closed frequent itemset generators. Charm_L [42], MinimalGenerator [41], Touch [34] and GenClose [8] are some typical algorithms for mining them.

Constraint-based association rule mining A serious problem encountered during the mining of frequent itemsets and association rules is that, in the worst case ($0 < s_0 = c_0 \le 1/|n|$), the cardinalities of frequent itemset class $\mathcal{FS}(s_0)$ and association rule set $\mathcal{ARS}(s_0, c_0)$ can become unwieldy (e.g., $\mathrm{Max}(\#\mathcal{FS}(s_0)) = 2^m - 1 = \mathcal{O}(2^m)$, $\mathrm{Max}(\#\mathcal{ARS}(s_0, c_0)) = 3^m - 2^m + 1 = \mathcal{O}(3^m)$). In addition, their generation typically produces an intractable number of duplicates (included in both candidates and solutions) that must then be eliminated. Thus, we not only waste computational and storage resources, but it is difficult to save, manage and use the results generated. Hence, in the interest of increased practicality, it is preferable to mine a suitable number of association rules subjected to user constraints.

One common rule mining approach is to filter the generated rule set via constraints on 'interestingness' measures until its size becomes manageable [12,24,35]. An alternative approach is to use inference methods to prune association rules that can be derived from other rules [14,25]. Zaki [41] proposes an algorithm for mining the "most general" rules with minimal antecedents and consequents (in terms of the subset relation) in a collection of rules with identical support and confidence. The rule for listing all the remaining rules is given in [37]. Pasquier et al. [32] and Tin et al. [36] concentrate on methods to discover rules with minimal antecedents and maximal consequents in rule classes with identical closures.

Several recent studies have focused on the discovery of frequent itemsets and association rules based on constraints (the reader is referred to [29] for details). Indeed, [28] added constraints such as monotone, anti-monotone, etc., to the mining process. The problem of integrating Boolean constraints, referring to the presence or absence of items in rules, was considered by [35]. In contrast, [10] proposed mining with a minimum "improvement" threshold. They also considered association rules with constraints in their right-handed sides. The concept of tree boundaries has been proposed to reduce the running times of the aforementioned mining methods. In addition, algorithms for mining multi-dimension association rules are given in [26].

1.1 Problem statement

We have recently concentrated on frequent itemset and association rule mining with frequently modified constraints, which directly involve support and confidence thresholds in addition to items. For example, online users who know frequent keyword sets contained in a class of keyword sets on a given subject might be interested in association rules between two given subjects. In [4,5], we solved the problem of finding frequent itemsets that are contained in a set given constraint $\mathcal{C}$ or contain at least one of its items. Hai et al. [21] applied double constraints to the problem. The discovery of association rules with various constraint types (the two-side union contained in a constraint, the intersection of a rule side with a constraint is not empty, the left-handed and right-handed sides contain two constraints, respectively) is considered in [6,9,22].

The present paper focuses on the mining of association rules based on maximum single constraints on both rule sides, which is stated as follows. Given four thresholds, minimum support s_0, maximum support s_1, minimum confidence c_0 and maximum confidence c_1, such that $0 < s_0 \leq s_1 \leq 1$, $0 < c_0 \leq c_1 \leq 1$ and two nonempty constraint itemsets in accordance with the two rule sides: $\varnothing \subset L_1, R_1, \subseteq \mathcal{A}$, the task is to discover the association rules $r : L' \rightarrow R'$ whose support and confidence are sandwiched by two pairs, (s_0, s_1), (c_0, c_1), and whose sides are contained in L_1 and R_1, respectively. More formally, we need to determine the set

$$\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \{r : L' \rightarrow R' \, \mathcal{ARS}(s_0, c_0) \\ \mid \text{supp}(r) \leq s_1, \text{conf}(r) \leq c_1, L' \subseteq L_1, R' \subseteq R_1\},$$

where $\mathcal{ARS}(s_0, c_0) = \{r : L' \rightarrow R' \mid \varnothing \neq L', R' \subseteq \mathcal{A}, L' \cap R' = \varnothing, S' \equiv L' + R', s_0 \leq \text{supp}(r), c_0 \leq \text{conf}(r)\}$ includes the association rules r with the standard meaning (the support of r and its confidence are written as supp(r) and conf(r)). For $s_1 = c_1 = 1$ and $L_1 = R_1 = \mathcal{A}$, we return the traditional mining problem. For smaller values of s_1 and greater values of c_0, we receive robust rules from unusual itemsets that are valuable in special cases.

1.2 Related work and approach

The traditional approach to generating association rules solves the problem in two phases: (1) discover frequent itemsets with constraints and (2) generate association rules with constraints from them. Srikant et al. [35] proposed a three-phase algorithm to mine association rules with item constraints. Apriori-based generation creates candidates containing given constraint items. A database pass allows for the computation of the supports of all the subsets of frequent itemsets with constraints. These frequent itemsets, together with their subsets and supports, are used to enumerate all the constrained association rules. However, similar to the different variants of the Apriori algorithm, it produces a large number of frequent itemset candidates as well as duplicates (D1, D2). Han et al. [20] introduced the idea of integrating constraints into the initialization of FP-trees. Pei et al. [33] proposed the concept of convertible constraints and combined them with FPGrowth for mining constrained frequent itemsets. Unfortunately, if the constraints change, the algorithms must be re-executed. Hence, they are unsuitable for user-interactive systems.

Post-processing approach In this approach, the constrained rule set $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ is discovered after the following two phases are completed: (1) determining the set of association rules without constraints $\mathcal{ARS}(s_0, c_0)$ and (2) checking and filtering out those of the form $r : L' \rightarrow R'$ satisfying the constraints, i.e., $\text{supp}(r) \ \leq s_1, \text{conf}(r) \ \leq c_1$ and $L' \subseteq L_1, R' \subseteq R_1$.

As discussed in the Introduction, we can identify $\mathcal{ARS}$ (s_0, c_0) by (1) finding the frequent itemset class $\mathcal{FS}(s_0)$ using algorithms such as Apriori, dEclat or FPGrowth, and then (2) for each $S' \in \mathcal{FS}(s_0)$, listing all the rules $r : L' \rightarrow R' \in \mathcal{ARS}(s_0, c_0)$, with $\varnothing \neq L' \subset S', R' \equiv S' \backslash L'$ (using the algorithms proposed by [3] or [31]). However, we encounter the same aforementioned difficulties when using this method.

A more efficient method, which mines $\mathcal{ARS}(s_0, c_0)$, is based on a lattice $\mathcal{LC}$ of frequent closed itemsets [7,32,36,37,41]. Rather than extracting all the frequent itemsets, we only extract the closed itemsets and determine the resulting lattice structure. Based on this lattice, the set $\mathcal{ARS}(s_0, c_0)$ is split into disjoint equivalence classes of identical closures of left-hand-side and two-side unions. The elements in each class have identical support and confidence and are computed once. Using frequent closed itemset generators (lattice $\mathcal{LCG}$), [32] proposed algorithms for mining rule classes. However, these algorithms generated redundancies and duplicates. In [7,36,37], we completely pruned the generation of duplicates using unique rule representations (based on effective set techniques) in each class. We also discovered rules wherein the two-side union of each rule adheres to a given constraint (see [6]). This approach is very efficient because (1) we compute $\mathcal{LCG}$ once (using well-known algorithms such as CHARM_L and MinimalGegenators), and (2) it is suitable for use with frequently modified support and confidence thresholds.

Because the cardinality of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ is typically small compared with $\mathcal{ARS}(s_0, c_0)$, these post-processing algorithms consume significant computing resources to both discover the rules of $\mathcal{ARS}(s_0, c_0)$ and filter out (using set operators) those in $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$. Even in special cases, this solution set can be empty. In addition, if online users modify the support and confidence constraints, $\mathcal{ARS}(s_0, c_0)$ must be re-computed, which decreases the speed of mining. Because the size and cardinality of $\mathcal{ARS}(s_0, c_0)$ with $s_0 = c_0 = 1/n$ are prohibitive, it becomes too complicated to mine and maintain.

1.3 Our approach

We use the lattice $\mathcal{LCG}$ of closed itemsets and generators because the cardinality of closed itemsets is typically orders of magnitude smaller than that of the total itemsets $\mathcal{FS}(s_0)$ (approximately 100 times smaller, as shown in [42]), and the ratio of the number of generators to the number of closed itemsets is approximately 1:2 (see [34]; thus, we only mine and save the lattice once. The frequent sub-lattice $\mathcal{LCG}$, with respect to the frequent closed itemsets satisfying specified constraints, can be quickly extracted from $\mathcal{LCG}$ whenever constraints emerge or change. To considerably decrease the number of duplicated candidates, it is necessary to partition the association rule set into disjoint classes. Using an equivalence relation on the closures of the two rule sides, $(L \equiv h(L') \subseteq S \equiv h(L' + R'))$, the constrained association rule set $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ is partitioned into disjoint equivalence rule classes $\mathcal{AR}^{+}_{\subseteq L_1, \subseteq R_1}(L, S)$ for each (L, S) in $\mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$, which contains all the closed itemset pairs satisfying the constraints. This prunes most of the duplicates produced during the generation of candidate rules and reduces the storage of the support and confidence of rules in the same class. In addition, it is also lays a foundation for designing efficient algorithms in parallel and distributed environments. The rules in each rule class, $\mathcal{AR}^{+}_{\subseteq L_1, \subseteq R_1}(L, S)$, are uniquely represented via the two closed itemsets L, S and their corresponding sets of generators $\mathcal{G}(L), \mathcal{G}(S)$. These representations help us understand the rule structure and prevent duplicates. We propose an algorithm called MAR_{MaxSC}, which mines a complete set of constrained rules in a negligible amount of time compared with post-processing methods.

1.4 Organization

The remainder of the paper is organized as follows. Section 2 covers the basic concepts of frequent itemset mining, association rule mining and closed itemset lattices. In Sect. 3, we first describe the partitioning of the constrained association rule set followed by the structure and unique representation of the rules of identical classes via closed itemsets and their generators. Based on theoretical results, the efficient MAR_MaxSC algorithm is proposed to completely and non-repeatedly mine a complete set of association rules given specific constraints. Section 4 describes the performance of the proposed algorithm compared with two post-processing algorithms. Finally, the conclusions of the study and future work are given in Sect. 5.

2 Preliminaries

Let $\mathcal{T} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ be a binary database where $\mathcal{O}$ is a nonempty set of objects (transactions), $\mathcal{A}$ are attributes (items) appearing in the objects and $\mathcal{R}$ is a binary relation on $\mathcal{O} \times \mathcal{A}$. A subset A of $\mathcal{A}$ is called an itemset. We consider the operator $\lambda : 2^{\mathcal{O}} \rightarrow 2^{\mathcal{A}}$ from the class of all object sets to the class of all itemsets and the operator $\rho : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{O}}$ from the class of all itemsets to the class of all object sets as follows:

$$\begin{aligned}\forall O, A : \varnothing \neq O \subseteq \mathcal{O},\ \varnothing \neq A \subseteq \mathcal{A}, \lambda(O) \\ = \{a \in \mathcal{A} | (o, a) \in \mathcal{R},\ o \in O\}, \rho(A) \\ = \{o \in \mathcal{O} \,|\, (o, a)\mathcal{R},\ \forall a \in A\},\end{aligned}$$

(per convention: $\lambda(\varnothing) = \mathcal{A}$, $\rho(\varnothing) = \mathcal{O}$). Itemset $\lambda(O)$ is the common itemset of all the objects in O, and $\rho(A)$ is the set of the objects included in A. We define the closure operator h on $2^{\mathcal{A}}$ as the union mapping of λ and ρ: $h = \lambda \text{ o } \rho$. Then, $h(A) = \lambda(\rho(A))$ is called the closure of A. Itemset A is a closed itemset if and only if h(A)=A [31].

The support of an itemset A is defined as the frequency of occurrence of the objects containing A, $\text{supp}(A) \equiv |\rho(A)|/|\mathcal{O}|$. The minimum and maximum support thresholds are designated s_0 and s_1, respectively, with $0 < 1/n \leq s_0 \leq s_1 \leq 1$ and $n = |\mathcal{O}|$. We only consider the non-trivial items in $\mathcal{A}$, $\mathcal{A}^F \equiv \{a \in \mathcal{A} : \text{supp}(\{a\})s_0\}$. The class of all closed itemsets is referred to as $\mathcal{CS}$. As the normal subset containment relation "$\supseteq$" on the subsets of $\mathcal{A}$ generates an order $\leq$ on $\mathcal{CS}$, $\mathcal{LC} \equiv (\{(S, \text{supp}(S)|S \in \mathcal{CS}\}, \leq)$ is the lattice of closed itemsets together with their support, which is represented by a Hass diagram. If the support of a nonempty itemset A (for $A \subseteq \mathcal{A}^F$) is greater than or equal to s_0 and less than or equal to s_1, i.e., $s_0 \leq \text{supp}(A) \leq s_1$, A is called a frequent itemset. By convention, s_1 is identical to 1. Let $\mathcal{FS}(s_0, s_1) \equiv \{L' : \varnothing \neq L' \subseteq \mathcal{A},\ s_0 \leq \text{supp}(L') \leq s_1\}$ and $\mathcal{FCS}(s_0, s_1) \equiv \mathcal{FS}(s_0, s_1) \cap \mathcal{CS}$ be the classes of all the frequent and closed itemsets, respectively.

For two nonempty itemsets $G, A : \varnothing \neq G \subseteq A \subseteq \mathcal{A}$, G is called a generator [32] of A if and only if[1] $h(G) = h(A)$ and $(h(G') \subset h(G), \forall G' : \varnothing \neq G' \subset G)$. We denote $\mathcal{G}(A)$ as the class of all generators of A. Because $\mathcal{G}(A)$ is not empty and finite [8] and $|\mathcal{G}(A)| = k$, all its elements can be numbered as follows: $\mathcal{G}(A) = \{G_1, G_2, \ldots, G_k\}$. Let

[1] We write "if and only if" as simply "iff".

$\mathcal{LCG} \equiv (\{< S, supp(S), \mathcal{G}(\mathrm{S}) > | S \in \mathcal{C}\mathrm{S}\}, \le)$ be the lattice of all the closed itemsets together their generators and supports, and let $\mathcal{FLCG}(\mathrm{s}_0, \mathrm{s}_1) \equiv (\{< S, supp(S), \mathcal{G}(\mathrm{S}) > | S \in \mathcal{FCS}(\mathrm{s}_0, \mathrm{s}_1)\}, \le)$ be the sub lattice of the frequent itemsets.

For frequent itemset $\mathrm{S}' \in \mathcal{FS}(\mathrm{s}_0, \mathrm{s}_1)$, we remove a proper, nonempty subset L' of S' ($\varnothing \neq \mathrm{L}' \subset \mathrm{S}'$) and assign $\mathrm{R}' \equiv \mathrm{S}' \backslash \mathrm{L}'$. Thus, an implication $\mathrm{r} : \mathrm{L}' \to \mathrm{R}'$ is called a rule created by L', R' (or L', S'). The support and confidence of r are defined as $\mathrm{supp(r)} \equiv \mathrm{supp(S')}$ and $\mathrm{conf(r)} \equiv \mathrm{supp(S')/supp(L')}$, respectively. For the given minimum and maximum confidences $\mathrm{c}_0, \mathrm{c}_1 (0 < \mathrm{c}_0 \le \mathrm{c}_1 \le 1)$, we consider r to be an association rule iff $\mathrm{s}_0 \le \mathrm{supp(r)} \le \mathrm{s}_1$ and $\mathrm{c}_0 \le \mathrm{conf(r)} \le \mathrm{c}_1$. When $\mathrm{s}_1 = 1$ and $\mathrm{c}_1 = 1$, we return the traditional concept of an association rule. The set of all association rules satisfying the thresholds of $\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1$ is written as

$$\mathcal{ARS}_{(}\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1) = \{\mathrm{r} : \mathrm{L}' \to \mathrm{R}' : \mathrm{L}', \mathrm{R}' \neq \varnothing, \\ \mathrm{L}' \cap \mathrm{R}' = \varnothing, \mathrm{L}' + \mathrm{R}'^{\,2} \in \mathcal{FS}(\mathrm{s}_0, \mathrm{s}_1), \mathrm{c}_0 \le \mathrm{conf(r)} \le \mathrm{c}_1\}.$$

The set of all rules that satisfy the two constraint itemsets $\mathrm{L}_1, \mathrm{R}_1 \subseteq \mathcal{A}$, is denoted by

$$\mathcal{ARS}_{\subseteq \mathrm{L}_1, \subseteq \mathrm{R}_1}(\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1) = \{\mathrm{r} : \mathrm{L}' \to \mathrm{R}' \\ \in \mathcal{ARS}(\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1) | \mathrm{L}' \subseteq \mathrm{L}_1, \mathrm{R}' \subseteq \mathrm{R}_1\}.$$

This also refers to the class of association rules with maximum single constraints, the association rule set with constraints or the constrained association rule.

3 Mining association rules based on maximum single constraints

3.1 Partitioning an association rule set with maximum single constraints

3.1.1 Rough partitioning

To considerably decrease the number of duplicated candidates, it is necessary to partition the association rule set into disjoint classes using a suitable equivalence relation. Based on the beautiful properties of operator h on lattice $\mathcal{LCG}$, we propose the following two equivalence relations on $\mathcal{FS}(\mathrm{s}_0, \mathrm{s}_1)$ and $\mathcal{ARS}(\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1)$:

Definition 1 (*Two equivalence relations on* $\mathcal{FS}(\mathrm{s}_0, \mathrm{s}_1)$ *and* $\mathcal{ARS}(\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1)$).

(a) $\forall A, B \in \mathcal{FS}(s_0, s_1), A \sim_{\mathcal{A}} B \Leftrightarrow h(A) = h(B)$.
(b) $\forall r_k : L_k \to R_k \in \mathcal{ARS}(s_0, s_1, c_0, c_1), k = 1, 2,$

$$r_1 \sim_r r_2 \Leftrightarrow [h(L_1) = h(L_2) \text{ and } h(L_1 + R_1) \\ = h(L_2 + R_2)]$$

It follows from Definition 1 that $\sim_{\mathcal{A}}$ and $\sim_{\mathrm{r}}$ are two equivalence relations. Let $[\mathrm{L}]_{\mathcal{A}} \equiv \{\mathrm{L}' \subseteq \mathrm{L} : \mathrm{L}' \neq \varnothing, \mathrm{h(L')} = \mathrm{L}\}$ be the equivalence class of the frequent closed itemsets having the same closure L where $\mathrm{L} \in \mathcal{FCS}(\mathrm{s}_0, \mathrm{s}_1)$. For $\mathrm{L}, \mathrm{S} \in \mathcal{FCS}(\mathrm{s}_0, \mathrm{s}_1), \varnothing \neq \mathrm{L} \subseteq \mathrm{S}, \mathrm{supp(S)/supp(L)} \in [\mathrm{c}_0; \mathrm{c}_1]$, the class $\mathcal{AR}(\mathrm{L}, \mathrm{S}) \equiv \{\mathrm{r} : \mathrm{L}' \to \mathrm{R}' | \mathrm{L}' \in [\mathrm{L}]_{\mathcal{A}}, \mathrm{R}' \neq \varnothing, \mathrm{L}' \cap \mathrm{R}' = \varnothing, \mathrm{S}' \equiv \mathrm{L}' + \mathrm{R}' \in [\mathrm{S}]_{\mathcal{A}}\}$ contains the rules $\mathrm{r} : \mathrm{L}' \to \mathrm{R}'$ such that $\mathrm{h(L')} = \mathrm{L}, \mathrm{h(L' + R')} = \mathrm{S}$.

Remark 1 (a) Using the properties of h, it is simple to show that $\forall L \in \mathcal{FCS}(s_0, s_1), supp(L') = supp(L), \forall L' \in [L]_{\mathcal{A}}$. In other words, every frequent itemset of the same class $[L]_{\mathcal{A}}$ has identical support $supp(L)$.

(b) For every $r : L' \to R' \ \mathcal{ARS}(s_0, s_1, c_0, c_1)$, we denote $L \equiv h(L'), S' \equiv L' + R', S \equiv h(S')$. We then have:$\varnothing \neq L \subseteq S, supp(S') = supp(S) \in [s_0, s_1], conf(r) = supp(S')/supp(L') = supp(S)/sup\ p(L) \in [c_0, c_1]$ and $(L, S) \in \mathcal{NFCS}\,(s_0, s_1, c_0, c_1)$, where

$$\mathcal{NFCS}\,(s_0, s_1, c_0, c_1) \equiv \{(L, S) \in \mathcal{CS}^2 | \\ S \in \mathcal{FCS}(s_0, s_1), \\ \varnothing \neq L \subseteq S, supp(S)/supp(L)\ [c_0, c_1]\}.$$

Consider $(L, S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$. Thus, every rule in equivalence class $\mathcal{AR}\,(L, S)$ has identical support and confidence, $supp(S)$, $supp(S)/supp(L)$, respectively. This fact considerably reduces the storage of the support and confidence of frequent itemsets and association rules.

(c) The following is a partition of the set $\mathcal{ARS}(s_0, s_1, c_0, c_1)$:

$$\mathcal{ARS}(s_0, s_1, c_0, c_1) = \Sigma_{(L,S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)} \mathcal{AR}(L, S).$$

As $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) \subseteq \mathcal{ARS}(s_0, s_1, c_0, c_1)$, it is straightforward to construct the following rough partition on the rule set with constraints $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$.

Proposition 1 (Roughly partitioning the constrained association rule set). *We have*

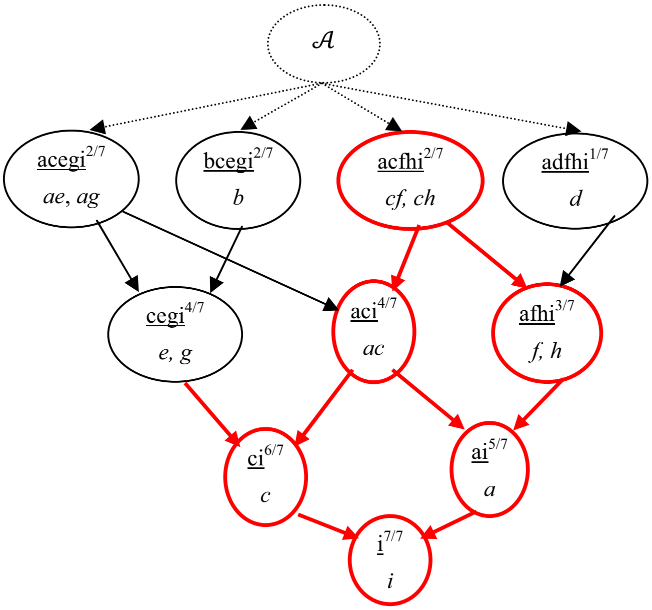
$$\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) \\ = \Sigma_{(L,S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)} \mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S),$$

where $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S) \equiv \{r : L' \to R' \in \mathcal{AR}(L, S) | L' \subseteq L_1, R' \subseteq R_1^{(t)}\}$.

Based on this rough partition (obtained from the lattice $\mathcal{FLCG}$ of frequent closed itemsets), association rules with constraints can be discovered in two steps. For each pair $(L, S) \in \mathcal{NFCS}\,(s_0, s_1, c_0, c_1)$, the rules $r : L' \to R'$ in $\mathcal{AR}(L, S)$ are non-repeatedly listed using the two

Object	Items								
1	a		c		e		g		i
2	a		c			f		h	i
3	a			d		f		h	i
4		b	c		e		g		i
5	a		c		e		g		i
6		b	c		e		g		i
7	a		c			f		h	i

(a) Example database $\mathcal{T}$



(b) The lattice of frequent closed itemsets (underlined) together with their generators (italicized) and support (superscripted).

Fig. 1 **a** Example database and **b** lattice of frequent closed itemsets and their generators

derivation functions $\mathcal{FS}(L)$, $\mathcal{FS}(S\backslash L')_{L'}$ proposed by [9]. Then, we check whether the rules satisfy two constraints, $L' \subseteq L_1$ and $R' \subseteq R_1$. The rules passed over by the check are retained. The corresponding algorithm is named $PP\_MAR\_MaxSC\_2$. It is worth noting that there are many instances of constraints where the corresponding constrained rule set $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ is empty, as well as many closed itemset pairs $(L,\ S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$ in which the corresponding classes with constraints $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}$ $(L,\ S)$ are empty. Moreover, despite the fact that $\varnothing \neq \mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L,\ S) \subseteq \mathcal{AR}(L,\ S)$, the size of $\mathcal{AR}(L,\ S)$ might remain prohibitive and contain numerous redundant rules.

Example 1 (Illustrating the weakness of $PP\_MAR\_MaxSC\_2$). In the remainder of the paper, we always consider the database $\mathcal{T}$ in Fig. 1a. Given $s_1 = 5/7$, $c_0 = 1/3$ and $c_1 = 0.9$. For $s_0 = 1/7$, $Charm\_L$ and $MinimalGenerators$ produce the lattice of frequent closed itemsets and their generators and support shown in Fig. 1b. We can see that $PP\_MAR\_MaxSC\_2$ exploits $|\mathcal{NFCS}(s_0, s_1, c_0, c_1)| = 19$ rule classes and generates 302 rules in $\mathcal{ARS}(s_0, s_1, c_0, c_1)$; however, it is unaware that these rules only satisfy the constraints of support and confidence.

(a) For the maximum single constraints of $L_1 = c$, $R_1 = i$, $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \varnothing$!
(b) Given $L_1 = ceg$, $R_1 = ai$, there are only two (per 19) rule classes with respect to two pairs, $(cegi,\ acegi)$ and $(ci,\ aci)$, that contain association rules satisfying the constraints, and their cardinality is only 12+2=14 (per 302)! Moreover, $\mathcal{AR}(cegi,\ acegi)$ includes 45 rules but we can only retrieve 12 of these desired rules, $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(cegi,\ acegi) = \{e \to a, e \to ai, ce \to a, ce \to ai, eg \to a, ge \to ai, ceg \to a, ceg \to ai, g \to a,\ g \to ai, cg \to a, cg \to ai\}$!
(c) For $L_1 = a$, $R_1 = cfhi$, there are only two rule classes containing the rules with constraints $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(ai,\ acfhi) = \{a \to cf, a \to cfi, a \to cfh, a \to cfhi, a \to ch, a \to chi\}$ and $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(ai, afhi) = \{a \to f, a \to fi, a \to h, a \to hi, a \to fh, a \to fhi\}$.

To overcome the above shortcomings, we propose two necessary condition groups. The first group addresses the nonemptiness of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$. The second group eliminates the specific pairs $(L,\ S) \in \mathcal{NFCS}(s_0, s_1, c_0$ $, c_1)$ for which the corresponding rule class $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L,\ S)$ is empty. Next, we describe the rules of $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S)$ (for pairs $(L,\ S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$ that pass the above condition) via $\mathcal{AR}^{+}_{\subseteq L_1, \subseteq R_1}(L,\ S)$. Based on this description, a smoother partition of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ is proposed.

3.1.2 Necessary conditions for the nonemptiness of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ and $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S)$

We denote the following:

$.S_1^* \equiv L_1 \cup R_1, C_1 \equiv L_1,$

$s_0^* \equiv max(s_0; c_0.supp(C_1)), s_1^* \equiv s_1;$
$.S' \equiv L' + R',\ S \equiv h(S'), S_{S_1^*} \equiv S \cap S_1^*,$
$\mathcal{G}_{S_1^*}(S) \equiv \{S_k \in \mathcal{G}(S) \,|\, S_k \subseteq S_1^*\}$
$.\,\mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*) \equiv \{S_{S_1^*} \equiv S \cap S_1^* \,|\, S$
$\in \mathcal{FCS}(s_0^*, s_1^*), \mathcal{G}_{S_1^*}(S) \neq \emptyset\};$
$.\, s_0' \equiv s_0'(S) \equiv supp(S)/c_1,\ s_1' \equiv s_1'(S)$
$\equiv min(1;\ supp(S)/c_0),$
$L \equiv h(L'), L_{C_1} \equiv L \cap C_1 = L \cap C_1,$
$\mathcal{G}_{C_1}(L) \equiv \{L_i \in \mathcal{G}(L) \,|\, L_i \subseteq C_1\},\ \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$
$\equiv \{L_{C_1} \equiv \mathrm{L} \cap C_1 | L \in \mathcal{FCS}(s_0', s_1'), \mathcal{G}_{C_1}(L) \neq \emptyset\};$
$.\,\mathcal{FS}_{\subseteq L_{C1}} \equiv \{L' \subseteq L_{C_1} \,|\, L' \neq \emptyset,\ h(L') = h(L_{C_1})\};$
$.\ R_1^* \equiv R_1^*(L') \equiv (S \cap R_1)\backslash L',\ \mathcal{FS}\left(S_{S_1^*}\backslash L'\right)_{L',\subseteq R_1^*}$
$\equiv \{R' \,|\emptyset \neq R' \subseteq R_1^*, h(L' + R') = h(S_{S_1^*})\};$
$.\,\mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1) \equiv \{(L,\ S) \in \mathcal{CS}^2 | S_{S_1^*}$
$\in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*),\ \emptyset \neq L \subseteq S,\ L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')\};$
$.\,\forall (L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1),$
$\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S) \equiv \{r : L' \to R' \,|\, L' \in \mathcal{FS}_{\subseteq L_{C1}},$
$R' \in \mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}\}.$

Proposition 2 (Necessary conditions for the nonemptiness of $\mathcal{ARS}_{\subseteq \mathrm{L}_1,\subseteq \mathrm{R}_1}(\mathrm{s}_0, \mathrm{s}_1, \mathrm{c}_0, \mathrm{c}_1)$, $\mathcal{AR}_{\subseteq \mathrm{L}_1,\subseteq \mathrm{R}_1}(\mathrm{L}, \mathrm{S})$, and a different representation of $\mathcal{AR}_{\subseteq \mathrm{L}_1,\subseteq \mathrm{R}_1}(\mathrm{L}, \mathrm{S})$).

(a) *If* $r : L' \to R' \in \mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1) \neq \emptyset$*:*

– $(L, S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1), r \in \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S) \neq \emptyset$ *with* $L = h(L'), S = h(L' + R')$ *and*

– *the following necessary conditions are satisfied:*

$$s_0^* \leq s_1^*, supp(S_1^*) \leq s_1^*. \qquad (H_1)$$

Henceforth, we always assume that (H_1) *is satisfied.*

(b) *For each* $(L, S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$*,*

– *For any* $r : L' \to R' \in \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S) \neq \emptyset$*:*

$$S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*), L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1'),$$
$$L' \in \mathcal{FS}_{\subseteq L_{C1}}, R' \in \mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}.$$

Then, $(L, S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1) \neq \emptyset$ *and* $\mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S) \subseteq \mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S)$*.*

– $\mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S) \subseteq \mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$*.*

(c) For each $(L, S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1) \neq \emptyset, \exists L' \in \mathcal{FS}_{\subseteq L_{C_1}}$ *and*

$$\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S) = \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S).$$

Proof (a) If $r :\ L' \to R' \in \mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$, we have $L',\ R' \neq \emptyset$, $L' \cap R' = \emptyset,\ S' \equiv L' + R',\ L' \subseteq L_1,\ R' \subseteq R_1$.

Denote $L \equiv h(L'), S \equiv h(L' + R')$. Because $L' \neq \emptyset$, $\emptyset \neq L \subseteq S$ (if $L = \emptyset, \emptyset \subset L' \subseteq h(L') \subseteq h(L) = \emptyset!$,), $supp(S) = supp(S') \in [s_0, s_1], supp(S)/supp(L) = supp(S')/supp(L') = conf(r)\ [c_0, c_1]$. Thus, $(L,\ S) \in \mathcal{NFCS}\ (s_0, s_1, c_0, c_1)$ and $r \in \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L,\ S)$.

Moreover, $S' \subseteq S_1^*,\ L' \subseteq L \cap C_1 = L_{C_1} \subseteq C_1, supp(C_1) \leq supp(L'), supp(C_1).c_0 \leq supp(L').c_0$ $supp(S')$ and $s_0^* \leq supp(S') = supp(S) \leq s_1^*$. Hence, $s_0^* \leq s_1^*$ and $supp(S_1^*) \leq supp(S') \leq s_1^*$.

(b) For every $(L,\ S) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$, $\forall r : L' \to R' \in \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L, S)$, we have $\emptyset \neq L \subseteq S,\ L',\ R' \neq \emptyset,\ L' \cap R' = \emptyset,\ h(L') = L,\ h(S') = S,\ s_0 \leq supp(S') = supp(S) \leq s_1,\ c_0 \leq supp(S)/supp(L) \leq c_1$ and $R' \subseteq R_1,\ L' \subseteq L_1$.

It is easy to know that $L' \in FS_{\subseteq L_{C1}}$ as $L' \subseteq L_{C_1} \subseteq L$ and $L = h(L') = h(L_{C_1})$. In addition, $supp(S)/c_1 \leq supp(L') = supp(L) \leq supp(S)/c_0,\ s_0' \leq supp(L') \leq s_1',\ S \supseteq S_{S_1^*} \supseteq S' \equiv L' + R'$ and $supp(C_1).c_0 \leq supp(L').c_0 \leq supp(S') = supp(S)$. Then, $s_0^* \leq supp(S) \leq s_1^*$ and $h(S') = h(S_{S_1^*}) = S$.

Take $L_i \in \mathcal{G}(L') \subseteq \mathcal{G}(L), S_k \in \mathcal{G}(S') \subseteq \mathcal{G}(S)$ (as $\mathcal{G}(L') \neq \emptyset,\ \mathcal{G}(S') \neq \emptyset$ [8]). We have $L_i \subseteq L' \subseteq C_1, S_k \subseteq S' \subseteq S_1^*$, i.e., $\mathcal{G}_{C_1}(L) \neq \emptyset$, $\mathcal{G}_{S_1^*}(S) \neq \emptyset$, $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1'),\ S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$ or $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$.

Moreover, the fact that $R' = S'\backslash L' \subseteq (S\backslash L') \cap R_1 = R_1^*$ means that $R' \in \mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*},\ r \in \mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S)$ and $\mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L,\ S) \subseteq \mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S)$.

As $c_0 \leq conf(r) = supp(S')/supp(L') = supp(S)/supp(L) \leq c_1$, we have $r \in \mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$. Therefore, $\mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L,\ S) \subseteq \mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$.

(c) Indeed, for $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$, we have $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1'), \exists L_i \in \mathcal{G}(\mathrm{L}) : L_i \subseteq C_1$ with $L' \equiv L_i, \emptyset \subset L_i \subseteq L'$. Then, $L_i = L' \subseteq L_{C_1} \subseteq L,\ L = h(L_i) = h(L') = h(L_{C_1})$. Hence, $L' \in \mathcal{FS}_{\subseteq L_{C_1}} \neq \emptyset$.

As $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1) \subseteq \mathcal{NFCS}\ (s_0, s_1, c_0, c_1)$, statement (b) shows that $\mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L,\ S) \subseteq \mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S)$. Thus, we must prove the reverse, i.e., $\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S) \subseteq \mathcal{AR}_{\subseteq L_1,\subseteq R_1}(L,\ S)$.

In fact, because $\forall r : L' \to R' \in \mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S), L',\ R' \neq \emptyset$, $L' \in \mathcal{FS}_{\subseteq L_{C_1}}, R' \in \mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$, we have $L' \subseteq L_{C_1} \subseteq C_1,\ h(L') = h(L_{C1}),\ R' \subseteq R_1^* = (S \cap R_1)\backslash L'R_1,\ L' \cap R' = \emptyset,\ h(L' + R') = h(S_{S_1^*})$. Because

```
$\mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1)$    $MFCS\_FromLattice$ ($\mathcal{LCG}^S$, $C_1$, $s'_0$, $s'_1$):
$\mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1) := \varnothing$;
if (($s'_0 > s'_1$) or (supp($C_1$) > $s'_1$)) then return $\varnothing$;
for each (<L, supp(L), $\mathcal{G}$(L)> $\in \mathcal{LCG}^S$) do
  if ($s'_0 \le$ supp(L) $\le s'_1$) then
    if ($\exists$ $L_i \in \mathcal{G}$(L) and $L_i \subseteq C_1$) then {
      $L_{C_1} = L \cap C_1$;
      $\mathcal{G}_{C_1}(L) = \{L_i \in \mathcal{G}(L) \mid L_i \subseteq C_1\}$;
      $\mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1) := \mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1)$ + <$L_{C_1}$, supp($L_{C_1}$) $\equiv$ supp(L), $\mathcal{G}_{C_1}$(L)>;
    }
return $\mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1)$;
```

Fig. 2 The $MFCS\_FromLattice$ procedure

$S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$, $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1)$, $\exists L_i \in \mathcal{G}(L) : L_i \subseteq C_1$, $\exists S_k \in \mathcal{G}(S) : S_k \subseteq S_1^*$, i.e., $L_i \subseteq L_{C_1} \subseteq L$, $L = h(L_i) = h(L_{C_1}) = h(L')$ and $S_k \subseteq S_{S_1^*} \subseteq S$, $S = h(S_k) = h(S_{S_1^*}) = h(S')$. Then, r $\in \mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S)$, i.e., $\mathcal{AR}^+_{\subseteq L_1, \subseteq R_1}(L, S) \subseteq \mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S) \neq \varnothing$. □

Consequence 1 (The necessary and sufficient condition for the nonemptiness of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$).

(a) *If at least one condition of* (H_1) *is violated, then* $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \varnothing$.
(b) $r{:}L' \to R' \in \mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) \neq \varnothing \Leftrightarrow$ *there exists* $(L, S) \in \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$, $L' \in \mathcal{FS}_{\subseteq L_{C_1}}$, $R' \in \mathcal{FS}(S_{S_1^*} \backslash L')_{L', \subseteq R_1^*}$, *and* $r : L' \to R' \in \mathcal{AR}^+_{\subseteq L_1, \subseteq R_1}(L, S) \neq \varnothing$.

Based on Proposition 2 and Consequence 1, we thus have a partition of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ that is smoother than that in Proposition 1.

3.1.3 Smoothly partitioning the association rule set with maximum single constraints

Theorem 1 (Smoothly partitioning the constrained association rule set) *Assuming that* (H_1) *is satisfied, we have*

$$\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \Sigma_{(L,S) \in \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)} \mathcal{AR}^+_{\subseteq L_1, \subseteq R_1}(L, S).$$

This partition establishes a foundation for independently mining each equivalence rule class $\mathcal{AR}^+_{\subseteq L_1, \subseteq R_1}(L, S)$. *Thus, it represents an original instance of using equivalence relations to obtain algorithms in parallel and distributed environments.*

Example 2 (Illustrating the emptiness of $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$ when at least one necessary condition given in (H$_1$) is not satisfied, and that of $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S)$ when $(L, S) \notin \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$).

(a) In Example 1(a), $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \varnothing$. Indeed, $S_1^* = ci$, $C_1 = c$, $s_1^* = 5/7$, $supp(S_1^*) = 6/7$. As the condition $supp(S_1^*) \le s_1^*$ is violated, we immediately conclude that $\mathcal{ARS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \varnothing$ without generating $|\mathcal{ARS}(1/7,\ 5/7,\ 1/3,\ 0.9) = 302$ rules and then eliminating all of them.
(b) For the constraints given in Example 1(b), we first find that most of the frequent closed itemset pairs (L, S) of $\mathcal{NFCS}(s_0, s_1, c_0, c_1)$ are redundant, i.e., they are not in $\mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1) = \{(cegi,\ acegi), (ci,\ aci)\}$. For example, there are 17 redundant pairs (per 19) that contain no association rules satisfying the constraints. Let us consider pair $(L, S) = (cegi,\ bcegi)$ in $\mathcal{NFCS}(s_0, s_1, c_0, c_1)$. It is easy to see that $S \in \mathcal{FCS}(s_0^*, s_1^*)$ as $s_0^* = 1.33/7\, supp(S) = 2/7 \le s_1^* = 5/7$. However, because $\mathcal{G}$(S) = {b} and {b} $\nsubseteq S_{S_1^*} = S \cap S_1^* = cegi$, the necessary condition $S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$ is not satisfied, i.e., $(L, S) \notin \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$). Hence, we immediately conclude that $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(cegi,\ bcegi) = \varnothing$ without listing all the rules of $\mathcal{AR}(cegi,\ bcegi)$. If the necessary condition tests are executed for all pairs of $\mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$, we can prune 17 rule classes, i.e., we avoid the generation of $254(302 - |\mathcal{AR}(cegi,\ acegi)| - |\mathcal{AR}(ci,\ aci)| = 302 - 45 - 3)$ $(302 - |\mathcal{AR}(cegi,\ acegi)| - |\mathcal{AR}(ci,\ aci)| = 302 - 45 - 3)$ corresponding redundant rule candidates. Observing the satisfied rule classes, we see that they retain many redundant candidates that are duplicates or were missed by the constraints. Through Example 3, we will show that all the redundant rule candidates can be completely pruned.

The $MFCS\_FromLattice(\mathcal{LCG}^S, C_1, s'_0, s'_1)$ procedure shown in Fig. 2 finds the class $\mathcal{FCS}_{\subseteq C_1}(s'_0, s'_1)$ of frequent closed itemsets satisfying the constraints from $\mathcal{LCG}^S$—the sub lattice of $\mathcal{LCG}$ with root S. To determine $\mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$, we call the procedure with the input parameters $\mathcal{LCG}$, S_1^*, s_0^* and

s_1^*. More formally, $\mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*) = MFCS\_FromLattice(\mathcal{LCG}, S_1^*, s_0^*, s_1^*)$. For example, if $S = acfhi$, the sub lattice $\mathcal{LCG}^{\mathrm{s}}$ is drawn by the lines of red color in Fig. 1b.

It is important to note that for $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$, the two sides of each rule $r : L' \rightarrow R'$ of rule class $\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L,\ S) = \{r : L' \rightarrow R' | L' \in \mathcal{FS}_{\subseteq L_{C1}},\ R' \in \mathcal{FS}(S_{S_1^*} \backslash L')_{L', \subseteq R_1^*}\}$ have not yet been explicitly represented, and their generation can contain numerous duplicates and redundant candidates.

3.2 Non-repeatedly producing all association rules satisfying the constraints in each class $\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S)$

For $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1, \subseteq R_1}(s_0, s_1, c_0, c_1)$, based on the two generator sets $\mathcal{G}(L)$ and $\mathcal{G}(S)$, we propose a unique, explicit representation for the constrained rules in$\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S)$. This representation leads to the distinct and complete production of all the constrained rules in each rule class, which is described in the algorithm $MAR\_MaxSC\_OneClass$.

3.2.1 The unique structure and representation of the equivalence class of frequent sub itemsets restricted on X with upper bound Z_1

The unique structure and representation of the equivalence class of frequent sub itemsets restricted on X with upper bound Z1 proposed in this section are used to make the unique structure and representation for the right-hand $R' \in \mathcal{FS}(S \backslash L')_{L', \subseteq R_1^*}$ and left-hand sides $L' \in \mathcal{FS}_{\subseteq L_{C_1}}$ of the rules $r : L' \rightarrow R'$ in each class$\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S)$.

For any $X,\ Y,\ Z_1\ \mathcal{A}$,

$$X \cap Y = \varnothing, \varnothing \neq Z_1 \subseteq Y \qquad (H_2)$$

(where Z_1 is an upper bound, X is a restriction). Let us call,

$$.\mathcal{FS}(Y)_{X, \subseteq Z_1} \equiv \{R' | \varnothing R' \subseteq Z_1, h(X + R') = h(X + Y)\}$$

(Note that if $X \cap Y = \varnothing$ and $Z_1 \subseteq Y$, then the necessary condition to $\mathcal{FS}(Y)_{X, \subseteq Z_1} \neq \varnothing$ is that $Y \neq \varnothing,\ Z_1 \neq \varnothing, X \cap Z_1 = \varnothing$, with $Y = \varnothing$ or or $Z_1 = \varnothing$ or $X \cap Z_1 \neq \varnothing$, we have $\mathcal{FS}(Y)_{X, \subseteq Z_1} \equiv \varnothing$.)

$.R_{\min} \equiv Minimal\{R_k \equiv S_k \backslash X, S_k \in \mathcal{G}(X + Y), R_k \subseteq Z_1\}$, $R_U^k \equiv U_{R_j \in R_{\min}, j \leq k} R_j$, $R_{U,k} \equiv \begin{cases} R_U^{k-1} \backslash R_k, & if\ k \geq 2 \\ \varnothing, & if\ k = 1 \end{cases}$, with $R_k \in R_{\min}$, $R_{-,k} \equiv Z_1 \backslash R_U^k$, Then, we denote

$$\mathcal{FS}^*(Y)_{X, \subseteq Z_1} \equiv \begin{cases} \{R' \equiv R_k + R'_k + R^{\sim}_k | R_k \in R_{min}, \\ \quad R'_k \subseteq R_{U,k}, R^{\sim}_k \subseteq R_{-,k}, \\ (R_j \not\subset R_k + R'_k, \\ \quad \forall R_j \in R_{min} : 1 \leq j < k), R' \neq \varnothing\}, & if\ R_{\min} \neq \varnothing \\ \varnothing & if\ R_{\min} = \varnothing \end{cases} \qquad (*)$$

Proposition 3 *(*Uniquely representing frequent itemsets in *$\mathcal{FS}(Y)_{X, \subseteq Z_1} \neq \varnothing$ by $\mathcal{FS}^*(Y)_{X, \subseteq Z_1}$). $\forall X, Y, Z_1 \subseteq \mathcal{A}$: $X \cap Y = \varnothing, \varnothing \neq Z_1 \subseteq Y$:*

(a) *The frequent itemsets in $\mathcal{FS}^*(Y)_{X, \subseteq Z_1}$ are distinctly enumerated.*
(b) $\mathcal{FS}(Y)_{X, \subseteq Z_1} = \mathcal{FS}^*(Y)_{X, \subseteq Z_1}$.
(c) $\mathcal{FS}^*(Y)_{X, \subseteq Z_1} \neq \varnothing \Leftrightarrow \mathcal{G}_{X+Z_1}(X + Y) \neq \varnothing$. *$(H_3)$*

Proof (a) We establish this by the method of contradiction. Assume that there exist two identical sets R'^1, R'^2 in $\mathcal{FS}^*(Y)_{X, \subseteq Z_1}$, so that $R'^1 = R'^2$, i.e.,$\exists k_2 > k_1 \geq 1$, $R'^j \equiv Z_0 + R_{k_j} + R'_{k_j} + R^{\sim}_{k_j}$, $R_{k_j} \in R_{\min}$, $R'_{k_j} \subseteq R_{U,kj}$, $R^{\sim}_{k_j} \subseteq R_{-,k_j}, \forall j = 1, 2$. Then, $R_{k_1} \subseteq R_{k_1} + R'_{k_1} + R^{\sim}_{k_1} = R_{k_2} + R'_{k_2} + R^{\sim}_{k_2}$ Because $R_{k_1} \cap R^{\sim}_{k_2} \subseteq R_{k_1} \cap R_{-,k_2} \subseteq R_{k_1} \cap R_{-,k_1} = \varnothing$ and R_{k_1}, R_{k_2}, are two different minimal elements of $R_{\min}$, we have $R_{k_1} \subset R_{k_2} + R'_{k_2}$, which contradicts the selection of ${R'_{k_2}}^{(*)}$.

(b) . "$\subseteq$": First, we consider the case that $R_{\min} \neq \varnothing$.
For $R' \mathcal{FS}(Y)_{X, \subseteq Z_1} \neq \varnothing$, $R' \subseteq Z_1$, $R' \neq \varnothing$. As $R' \subseteq Z_1 \subseteq Y$, $Y \cap X = \varnothing$, we have $R' \cap X = \varnothing$, $S' \equiv X + R'$, $h(S') = h(X + Y)$, $X \subseteq S' \subseteq X + Y$.
From $S' \neq \varnothing$, take $S_k \in \mathcal{G}(S') \subseteq \mathcal{G}(X + Y)$ (see [8]), $S_k \subseteq S'$, we have $R_k \equiv S_k \backslash X \subseteq S' \backslash X = R' \subseteq Z_1$.

Let $B \equiv \{R_i \equiv S_i \backslash X : S_i \in \mathcal{G}(S'), R_i \subseteq Z_1\}$, $C \equiv \{R_i \equiv S_i \backslash X : S_i \in \mathcal{G}(X + Y), R_i \subseteq Z_1\}$. Thus, $R_k \in B$. Since $B,\ C$ are finite and $\varnothing \neq B \subseteq C$, there exists a minimal set $R_{\min, S'} \equiv Minimal(B) \neq \varnothing$, $R_{\min} \equiv Minimal(C) \neq \varnothing$. Thus, we always acquire the minimum index k of sets R_i in $R_{\min, S'}\quad Minimal(B)$.

On the contrary, assume that $R_k \notin R_{\min}$. Then, $\exists R_j \in R_{\min} : R_j \subset R_k$, with $R_j \equiv S_j \backslash X$, $S_j \in \mathcal{G}(X + Y)$ and $h(S_j) = h(X + Y)$, $S_j \subseteq X \cup S_j = X + R_j \subseteq X + R_k \subseteq X + R' = S' \subseteq X + Y$, $h(X + Y) = h(S_j) = h(S')$. Hence, $S_j \in \mathcal{G}(S')$, $R_j \subseteq S' \backslash X = R' \subseteq Z_1$, $R_j \in B \cap R_{\min}$. We then have $R_j \in R_{\min, S'}$ and $R_j \subset R_k \in R_{\min, S'}$. This is impossible as the assumption is that R_k is the minimal set in B! Therefore, $R_k \in R_{\min} \neq \varnothing$.

This implies that, if $R_{\min} = \varnothing$, $\mathcal{FS}(Y)_{X, \subseteq Z_1} = \varnothing \mathcal{FS}^*(Y)_{X, \subseteq Z_1}$.

We also have $S' = S_k + S''_k$, for $S''_k \equiv S' \backslash S_k$. It follows from $S' \supseteq X$ that $S' = X + R_k + R'_k + R^{\sim}_k = X + R'$, with $R' \equiv R_k + R'_k + R^{\sim}_k$, $R_k \equiv S_k \backslash X\ R_{\min}$, $R'_k \equiv (S''_k \backslash X) \cap R_U^k = [(S' \backslash X) \backslash S_k] \cap$

$R_U^{k-1} \subseteq R_U^{k-1}\backslash S_k \subseteq R_U^{k-1}\backslash R_k \equiv R_{U,k}$ (as $R_k \cap [(S'\backslash X)\backslash S_k] \subseteq R_k\backslash S_k = \varnothing$), $R_k^{\sim} \equiv (S_k''\backslash X)\backslash R_U^k \subseteq (S'\backslash X) \backslash R_U^k \subseteq Z_1\backslash R_U^k \equiv R_{-,k}$.

We now suppose that $\exists R_j \equiv S_j\backslash X\ R_{\min} : 1 \leq j < k$ and $R_j \subset R_k + R_k'$. Thus, $h(S_j) = h(X+Y)$, $R_j \subseteq Z_1$, $S_j \subseteq X \cup S_j = X + R_j \subseteq X + R_k + R_k' \subseteq X + R' \equiv S' \subseteq X + Y$, $h(X + Y) = h(S_j) = h(S')$. Then, $S_j \in \mathcal{G}(S')$ and $R_j \in B \cap R_{\min}$. Hence, $R_j \in R_{\min,S'}$, i.e., $j < k$: a contradiction on how to choose index k! We can conclude that $R' \in \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$.

"$\supseteq$": For any $R' \in \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$, $R' = R_k + R_k' + R_k^{\sim}$ where $R_k \equiv S_k\backslash X \in R_{\min}$, $S_k \in \mathcal{G}(X+Y)$, $h(S_k) = h(X+Y)$, $R_k \subseteq Z_1$, $R' \neq \varnothing$. Furthermore, $R_k' \subseteq R_{U,k} \subseteq Z_1$, $R_k^{\sim} \subseteq R_{-,k} \subseteq Z_1$. Then, $R' \subseteq Z_1 \subseteq Y$ and $R' \cap X = \varnothing$. Otherwise, because $X+Y \supseteq X+R' \supseteq X+R_k = X \cup S_k \supseteq S_k$, we have $h(S_k) = h(X+R') = h(X+Y)$. Hence, $'$ $\mathcal{FS}(Y)_{X,\subseteq Z_1}$.

$$\text{(c)}{+}R_{\min} \neq \varnothing \Leftrightarrow \exists S_k \in \mathcal{G}(X+Y):\ R_k \equiv S_k\backslash X \subseteq Z_1 \quad \text{(c.1)}$$

$$\Leftrightarrow \exists S_k \in \mathcal{G}(X+Y): S_k \subseteq X+Z_1 (\text{or} \mathcal{G}_{X+Z_1}(X+Y)) \neq \varnothing \quad \text{(c.2)}$$

In fact, if $S_k\backslash X \subseteq Z_1$, $S_k \subseteq X \cup S_k = X + R_k \subseteq X + Z_1$ (as $X \cap Z_1 = \varnothing$). In contrast, if $S_k \subseteq X + Z_1$, $S_k\backslash X \subseteq Z_1$.
+ If $\mathcal{FS}^*(Y)_{X,\subseteq Z_1} \neq \varnothing$, $R_{\min} \neq \varnothing$. We immediately have (c.2).
+ Suppose that (c.2) is true. Thus, $R_{\min} \neq \varnothing$. For $R^* \equiv Z_1$, we have $\varnothing \neq R^* \subseteq Y$. Take an arbitrary $R_k \equiv S_k\backslash X \in R_{\min} : S_k \in \mathcal{G}(X+Y)$, $S_k \subseteq X+Y$, $R_k \subseteq Z_1 = R^*$. Then, $S_k \subseteq S_k \cup X = X + R_k \subseteq X + R^* \subseteq X + Y$. This implies that $h(X+Y) = h(S_k) = h(X+R^*)$. Hence, $\exists R^* \in \mathcal{FS}(Y)_{X,\subseteq Z_1} = \mathcal{FS}^*(Y)_{X,\subseteq Z_1} \neq \varnothing$. □

This concludes the proof of Proposition 3, which implies the following remark.

Remark 2 (a) If $R_{\min} \neq \varnothing$, $\mathcal{FS}(Y)_{X,\subseteq Z_1} \neq \varnothing \equiv \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$.

(b) $\mathcal{G}_{X+Z_1}(X+Y) \neq \varnothing \Leftrightarrow \exists S_k \in \mathcal{G}(X+Y) : S_k \subseteq X + Z_1 \Leftrightarrow R_{\min} \neq \varnothing$.

(c) Let us consider $\forall R' \in \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$ such that $R' \equiv R_k + R_k' + R_k^{\sim}$. If $\exists R' = \varnothing$, then $\exists S_k \in G(X + Y) : R_k \equiv S_k\backslash X = \varnothing$. Therefore, $S_k \subseteq X \subseteq X + Y$ and $h(X+Y) = h(X) = h(S_k)$. Furthermore, $R_{\min} \equiv \{R_1 \equiv \varnothing\}$, $R_{U,1} = R_U^1 = \varnothing$, $R_{-,1} \equiv Z_1 \neq \varnothing$ and $R' = R_1^{\sim} \subseteq R_{-,1}$. Hence, R' is empty if $h(X+Y) = h(X)$ and $R_{\min} = \{\varnothing\}$. Then, $R' \in \mathcal{FS}^*(Y)_{X,\subseteq Z_1} \Leftrightarrow \varnothing \subset R' \subseteq Z_1 \neq \varnothing$.

For practical purposes, when computing $R_{\min}$, we consider the following two cases:

- If $R_{\min} = \{\varnothing\}$, then $R_{U,1} = R_U^1 = \varnothing$, $R_{-,1} \equiv Z_1 \neq \varnothing$ and

$$\mathcal{FS}^*(Y)_{X,\subseteq Z_1} \equiv \{R' | \varnothing \neq R' \equiv R_1^{\sim},\ R_1^{\sim} \subseteq Z_1\}$$
$$= \{R' | \varnothing\ R' \subseteq Z_1\}$$

- If $R_{\min} = \{\varnothing\}$:

$$\mathcal{FS}^*(Y)_{X,\subseteq Z_1} \equiv \{R' \equiv R_k + R_k' + R_k^{\sim} \mid R_k \in R_{\min}, R_k' \subseteq R_{U,k},\ R_k^{\sim} \subseteq R_{-,k},\ (R_j \not\subset R_k + R_k', \forall R_j \in R_{\min} : 1 \leq j < k)\}.$$

It is worth noting that, in this case, we are not required to check whether $R' \neq \varnothing$ in generating R' because it is always true.

(d) (The advantage of $^{(*)}$ for exponentially decreasing redundancy). Assume that we are currently forming the sets R$'$. Starting with R_k, we grow subsets $R_k' \subseteq R_{U,k}$ and then $R_k^{\sim} \subseteq R_{-,k}$ to complement R'; if $^{(*)}$ is incorrect, it is unnecessary to consider the approximately $(2^{|R_{U,k}\backslash R_k'|} - 1)$ supersets R'' of $R_k'(R_k' \subset R'' \subseteq R_{U,k})$ and add all $(2^{|R_{-,k}|})$ subsets $R_k^{\sim}$ of $R_{-,k}$to R'. Essentially, we have eliminated approximately $(2^{|R_{U,k}\backslash R_k'|} - 1).(2^{|R_{-,k}|})$ redundant subset candidates for R'. Next, we consider the remaining sets $R_k'' \subseteq R_{U,k}$ (such that $R_k' \not\subset R_k'' \subseteq R_{U,k}$) or the subsequent sets R_k in $R_{\min}$. Using the necessary and sufficient condition (*), we can perfectly eliminate duplicates when generating the rules $r : L' \rightarrow R'$ in each class $\mathcal{AR}_{\subseteq L_1, \subseteq R_1}(L, S)$ based solely on minimal sets or generators. Due to their low cardinality and size, the algorithms applied during this generation are fast and efficient.

(e) (Modifying the computation of the upper bound sets $R_{U,k}$ and $R_{-,k}$ of R_k' and $R_k^{\sim}$, respectively). We can see that, for each $k > 1$, the operations of $R_U^{k-1} = R_U^{k-2} \cup R_{k-1}$, $R_{U,k} \equiv R_U^{k-1}\backslash R_k$and $R_{-,k} \equiv Z_1\backslash R_U^k$ must be executed on sets that are potentially non-disjoint. To conserve calculation time, it is important to observe that

$$\begin{aligned} R_{U,k} &= [(R_U^{k-2}\backslash R_{k-1}) + R_{k-1}]\backslash R_k \\ &= (R_{U,k-1} + R_{k-1})\backslash R_k, \\ R_{-,k} &= R_{-,k-1}\backslash R_k, \forall k \geq 2 \text{ and } R_{U,1} \\ &\equiv \varnothing,\ R_{-,1} \equiv Z_1\backslash R_1. \end{aligned}$$

In other words, $R_{U,k} = \begin{cases} (R_{U,k-1}+R_{k-1})\backslash R_k, & if\ k\geq 2 \\ \varnothing, & if\ k=1 \end{cases}$,

$$R_{-,k} = \begin{cases} R_{-,k-1}\backslash R_k, & if\ k \geq 2 \\ Z_1\backslash R_1, & if\ k = 1 \end{cases}.$$

Thus, for each $k \geq 2$, we compute the disjoint union $R_{U,k} = R_{U,k-1} + R_{k-1}$ where $R_{U,k-1} \subseteq R_U^{k-2}$ and the difference $R_{-,k} = R_{-,k-1}\backslash R_k$ where $R_{-,k-1} \subseteq Z_1$, $R_k \subseteq R_U^k$. It is readily apparent that this new calculation is faster than the old one.

For special values of Y, X and Z_1 in $\mathcal{FS}(Y)_{X,\subseteq Z_1}$, we have two structures, $\mathcal{FS}_{\subseteq LC1}$ and $\mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$.

3.2.2 *Structure and unique representation of the $\mathcal{FS}_{\subseteq LC1}$ and $\mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$ itemsets*

Suppose that $(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$: $S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$, $\varnothing \neq L \subseteq S$, $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$ and L' $\in \mathcal{FS}_{C_0 \subseteq LC1}$. As $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$, we have $\mathcal{G}_{C_1}(L) \neq \varnothing$, $\exists L_i \in \mathcal{G}(L) : \varnothing \subseteq L_i \subseteq L_{C_1}$ and $L_{C_1} \neq \varnothing$.

Corollary 1 *$\forall\ L,\ C_1 \subseteq \mathcal{A}$, if $L_{C_1} \equiv L \cap C_1 \neq \varnothing$ and $\mathcal{G}_{C_1}(L) \neq \varnothing$, then $\mathcal{G}(L_{C_1}) = \mathcal{G}_{C_1}(L)$.*

Structure and unique representation of the itemsets of $\mathcal{FS}_{\subseteq LC1}$ For $Y \equiv L_{C_1}$, $X \equiv \varnothing$ and $Z_1 = L_{C_1}$. As $L_{C_1} \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$, we know from Corollary 1 that $\mathcal{G}_{C_1}(L) = \mathcal{G}(L_{C_1}) \neq \varnothing$ and $\forall L_i \in \mathcal{G}_{C_1}(L) : \varnothing \subset L_i \subseteq L_{C_1}$. Thus,

$$\mathcal{FS}(L_{C_1})_{\varnothing,\ \subseteq L_{C_1}} = \{L' |\ \varnothing \neq L' \subseteq L_{C_1}, h(L') = h(L_{C_1})\} \equiv \mathcal{FS}_{\subseteq L_{C_1}}.$$

Based on the representation of $R_{\min}$ in $\mathcal{FS}^*(Y)_{X,\subseteq Z_1}$, $K_{\min} \equiv Minimal\{L_i,\ L_i \in \mathcal{G}_{C_1}(L)\} = \mathcal{G}_{C_1}(L)$, $L_U^i \equiv \cup_{L_k \mathcal{G}_{C_1}(L), k \leq i} L_k$,$L_{U,i} \begin{cases} L_U^{i-1}\backslash L_i, & if\ i \geq 2 \\ \varnothing, & if\ i = 1 \end{cases}$, $L_{-,i} \equiv L_{C_1}/L_U^i$ and

$$\mathcal{FS}^*_{\subseteq L_{C_1}} \equiv \{\ L' \equiv L_i + L_i' + L_i^{\sim} | L_i \in \mathcal{G}_{C_1}(L), L_i' \subseteq L_{U,i},\ L_i^{\sim} \subseteq L_{-,i}, (L_k \not\subset L_i + L_i', \forall L_k \in \mathcal{G}_{C_1}(L) : 1 \leq k < i),\ L' \neq \varnothing\ \}. \quad (1)$$

Because $\mathcal{G}_{C_1}(L) \neq \varnothing$ and $L_{C_1} \neq \varnothing$, it follows from Proposition 3(c) that $\mathcal{FS}^*_{\subseteq L_{C_1}} \neq \varnothing$.

Structure and unique representation of the itemsets of $\mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$ For $Y \equiv S_{S_1^*}\backslash L'$, $X = L'$, $Z_1 = R_1^* = (S \cap R_1)\backslash L' : Z_1 \subseteq Y$. Based on the fact that $S_{S_1^*} \in \mathcal{FCS}_{\subseteq S_1^*}$ (s_0^*, s_1^*) and Corollary 1, we have $\mathcal{G}\left(S_{S_1^*}\right) = \mathcal{G}_{S_1^*}(\mathrm{S}) \neq \varnothing$. Then,

$$\mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} \equiv \{R' | \varnothing \neq R' \subseteq R_1^*,\ h(L' + R') = h(S_{S_1^*})\}.$$

We denote $R_{\min} \equiv Minimal\{R_k \equiv S_k\backslash L',\ S_k \in \mathcal{G}_{S_1^*}(S),\ R_k \subseteq R_1^*\}$, $R_U^k \equiv \cup_{R_j \in R_{\min}, j \leq k} R_j$,, $R_{U,k} \equiv \begin{cases} R_U^{k-1}\backslash R_k, & if k \geq 2 \\ \varnothing, & if k = 1 \end{cases}$, $R_{-,k} \equiv R_1^*\backslash(L' + R_U^k) = (S \cap R_1)\backslash(L' + R_U^k)$ and

$$\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} \equiv \{R' \equiv R_k + R_k' + R_k^{\sim} | R_k \in R_{\min}, R_k' \subseteq R_{U,k},\ R_k^{\sim} \subseteq R_{-,k}, \left(R_j \not\subset R_k + R_k', \forall R_j \in R_{\min} : 1 \leq j < k\right),\ R' \neq \varnothing\}. \quad (2)$$

By Proposition 3(c), $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} \neq \varnothing \Leftrightarrow [\mathcal{G}_{L'\cup R_1}(S) \neq \varnothing$ and $(S \cap R_1)\backslash L' \neq \varnothing$]. In fact, for $\forall S_k \in \mathcal{G}(S)$, we have $S_k \subseteq L' + (S \cap R_1)\backslash L' = (S \cap R_1)\, L' = S \cap \left(L' \cup R_1\right) \Leftrightarrow S_k \subseteq L' \cup R_1$ as $S_k \subseteq S$.

The following is a consequence of Proposition 3.

Consequence 2 (Unique representation and distinct generation of the two sides of the rules in $\mathcal{AR}^+_{\subseteq L_1,\ \subseteq R_1}(L,\ S)$). $\forall(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$:

(a) *The itemsets in $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$ and $\mathcal{FS}^*_{\subseteq L_{C_1}}$ are generated distinctly.*

(b) $\mathcal{FS}(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} = \mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$, $FS_{\subseteq L_{C_1}} = \mathcal{FS}^*_{\subseteq L_{C_1}}$.

(c) $\mathcal{FS}^*_{\subseteq L_{C_1}} \neq \varnothing$.

(d) $\forall L' \mathcal{FS}^*_{\subseteq L_{C_1}}$, *then* $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} \neq \varnothing \Leftrightarrow [\mathcal{G}_{L'\cup R_1}(S) \neq \varnothing$ *and* $(S \cap R_1)\backslash L' \neq \varnothing]$.

The general procedure $MFS\_RestrictMaxSC(Y,\ X,\ Z_1,\ \mathcal{G}(X+Y))$ completely and distinctly generates the itemsets of $\mathcal{FS}^*(Y)_{X,\subseteq Z_1}$(shown in Fig. 3)

$$\mathcal{FS}^*(Y)_{X,\subseteq Z_1} = MFS\_RestrictMaxSC\ (Y,\ X,\ Z_1,\ \mathcal{G}(X+Y)).$$

Based on Remark 2, we can add Lines 4–7 to the procedure. Furthermore, at Line 21, we do not check if $\mathrm{R_k} + \mathrm{R'_k} + R_k^{\sim} \neq \emptyset$ (because it is obvious). The special cases of this procedure produce the results shown in Table 1.

In each rule class, cases 1 and 2 are used to distinctly enumerate the left-hand sides L' of $\mathcal{FS}^*_{\subseteq L_{C_1}}$ and the right-hand sides $R'$of $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$, respectively. Moreover, cases 3 and 4 give us two efficient procedures for generating $L'\mathcal{FS}(L)$ and $R'\mathcal{FS}(S\backslash L')_{L'}$ in each class $\mathcal{AR}(L, S)$ (used in [9]). They are also used to generate the rules via the $PP\_MAR\_MaxSC\_2$ post-processing approach as discussed in Sect. 3.1.1.

3.2.3 *Structure and unique representation of rule class $\mathcal{AR}^+_{\subseteq L_1, \subseteq R_1}(L, S)$*

For $\forall(L,\ S) \in \mathcal{NFCS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$, let us denote

$$\mathcal{AR}^*_{\subseteq L_1,\ \subseteq R_1}(L,\ S) \equiv \{r : L' \to R' |\ L' \in \mathcal{FS}^*_{\subseteq L_{C_1}}, R' \in \mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}\},$$

$$Suff\_\mathcal{FS}^*_{\subseteq L_{C_1}}(S,\ R_1) \equiv \{L' \in \mathcal{FS}^*_{\subseteq L_{C_1}} \mid \mathcal{G}_{L'\cup R_1}(S) \neq \varnothing \text{ and } (S \cap R_1)\backslash L' \neq \varnothing\ \}.$$

```
$\mathcal{FS}^*(Y)_{X,\subseteq Z_1}$    $MFS\_RestrictMaxSC$(Y, X, $Z_1$, $\mathcal{G}$(X+Y)): // $Z_1 \subseteq$ Y
$\mathcal{FS}^*(Y)_{X,\subseteq Z_1} = \varnothing$;
// testing the necessary conditions for the nonemptiness of $FS^*(Y)_{X,\subseteq Z_1}$:
/* if ((X∩Y ≠ ∅) or // can be deleted for association rule mining, as $(L,S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)$.
        $(Z_1 = \varnothing)$) // can be deleted for association rule mining, as $Z_1 = (S \cap R_1)\backslash L' \neq \varnothing$ in Line 3 of Fig. 4
          then return ∅; */    // in general, we need such conditions!
if (($R_{min}$ = Minimal{$R_k = S_k \backslash X \mid S_k \in \mathcal{G}(X+Y), R_k \subseteq Z_1$}) = ∅) then return ∅;  //$\mathcal{G}_{X+Z_1}(X+Y) = \varnothing$
if ($R_{min}$ = {∅}) then {                                          // Remark 2.c
        for each (R'' | ∅ ⊂ R'' ⊆ $Z_1$) do
                $\mathcal{FS}^*(Y)_{X,\subseteq Z_1} = \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$ + {R''};
}
else
{   $R_0 = \varnothing$; $R_{U,0} = \varnothing$; $R_{-,0} = Z_1$;                                  // Remark 2.e
    for (k=1; $R_k \in R_{min}$; k++) do {
        $R_{U,k} = (R_{U,k-1} + R_{k-1}) \backslash R_k$;
        for each ($R'_k \subseteq R_{U,k}$) do {
            IsDuplicate = false;
              for (j=1; $R_j \in R_{min}$, j < k; j++) do
                  if ($R_j \subset R_k + R'_k$) then {
                      IsDuplicate = true; break;
                  }
            if (not(IsDuplicate)) then {
                  $R_{-,k} = R_{-,k-1} \backslash R_k$;
                  for each ($R_k^{\sim} \subseteq R_{-,k}$) do   // for $R_{min} \neq \{\varnothing\}$, we surely know that $R_k + R'_k + R_k^{\sim} \neq \varnothing$
                        $\mathcal{FS}^*(Y)_{X,\subseteq Z_1} = \mathcal{FS}^*(Y)_{X,\subseteq Z_1}$ + {$R_k + R'_k + R_k^{\sim}$};
              }
        } // end for each $R'_k$
    } // end for k
}
return $\mathcal{FS}^*(Y)_{X,\subseteq Z_1}$;
```

Fig. 3 The $MFS\_RestrictMaxSC$ procedure

Table 1 Special results of $\mathcal{FS}^*(Y)_{X,\subseteq Z_1}$

Case	Y	X	Z_1	$\mathcal{G}(X+Y)$	Result set
1	L_{C_1}	$\varnothing$	L_{C_1}	$\mathcal{G}(L_{C_1})$	$\mathcal{FS}^*_{\subseteq L_{C_1}}$
2	$S_{S_1^*}\backslash L'$	L'	R_1^*	$\mathcal{G}(S_{S_1^*})$	$\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$
3	L	$\varnothing$	$\mathcal{A}$	$\mathcal{G}(L)$	$\mathcal{FS}(L)$
4	$S\backslash L'$	L'	$\mathcal{A}$	$\mathcal{G}(S)$	$\mathcal{FS}(S\backslash L')_{L'}$

The following consequence is deduced from Consequence 2.

Consequence 3 (Necessary and sufficient conditions for the nonemptiness of $\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S)$ and its representation) $\forall(L, S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$:

(a) *The rules in* $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S)$ *are enumerated nonrepeatedly.*

(b) $\mathcal{AR}^+_{\subseteq L_1,\subseteq R_1}(L, S) = \mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S)$.

(c) $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S) \neq \varnothing \Leftrightarrow Suff\_FS^*_{\subseteq L_{C_1}}(S, R_1) \neq \varnothing$.

(d) $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S) = \sum_{L' \in Suff\_FS^*_{\subseteq L_{C_1}}(S,R_1)}\{r : L' \to R' : R' \in \mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}\}$.

Remark 3 For $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$, if $S \equiv L \in \mathcal{G}(L)$, $\exists! L' \equiv L \in [L]$ Then, $Z_1 = (L \cap R_1)\backslash L' = \varnothing$ and $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} = \varnothing$! Hence, when $L \equiv S$, we always assume that $L \notin \mathcal{G}(L)$.

The $MAR\_MaxSC\_OneClass$ algorithm given in Fig. 4 distinctly generates all the association rules with the constraints $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S)$ for each pair $(L, S) \in \mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0, s_1, c_0, c_1)$.

Example 3 (Illustrating the advantage of distinctly generating all the rules in $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S)$ by $MAR\_MaxSC\_$

```
$\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S)$    $MAR\_MaxSC\_OneClass\ (L_{C_1}, \mathcal{G}(L_{C_1}), R_1, S_{S_1^*}, \mathcal{G}(S_{S_1^*}), S)$
1. $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S) = \varnothing$;
2. $\mathcal{FS}^*_{\subseteq L_{C_1}} = MFS\_RestrictMaxSC(L_{C_1}, \varnothing, L_{C_1}, \mathcal{G}(L_{C_1}))$;
3. for each (L' $\in \mathcal{FS}^*_{\subseteq L_{C_1}}$ and $(S \cap R_1)\backslash$L' $\neq \varnothing$) do {   // Line 2 in Fig. 3 can be deleted
4.         $R_1^* = (S \cap R_1)\backslash$L';
5.         $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*} = MFS\_RestrictMaxSC(S_{S_1^*}\backslash L', L', R_1^*, \mathcal{G}(S_{S_1^*}))$;
6.         for each R' $\in \mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$ do
7.             $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S) = \mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S)$ + {r:L'→R'};
8. }
9. return $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S)$;
```

Fig. 4 The *MAR_MaxSC_OneClass* algorithm

OneClass). Let us consider the mining of association rules with constraints on database $\mathcal{T}$ with $s_0 = 1/7, s_1 = 5/7, c_0 = 1/3, c_0 = 0.9$

(a) For the constraints $L_1 = ceg,\ R_1 = ai$, we have $S_1^* = ceagi, C_1 = ceg, s_0^* = 1.33/7, s_1^* = 5/7$ and $supp(S_1^*) = 2/7$. Then, $\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1) \neq \varnothing$. Now, consider the rule class with respect to $(L,\ S) = (cegi,\ acegi) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$ in Example 1(b): $\mathcal{G}$(L) $= \{e,\ g\}, \mathcal{G}(S) = \{ae,\ ag\},\ supp(L) = 4/7$ and $supp(S) = 2/7$. We have $S_{S_1^*}{=}ceagi \in \mathcal{FCS}_{\subseteq S_1^*}(s_0^*, s_1^*)$ as $S \in \mathcal{FCS}(s_0^*, s_1^*)$ and $\mathcal{G}_{S_1^*}(S) \equiv \{ae,\ ag\} \neq \varnothing$. Furthermore, because $s_0' = supp(S)/c_1 = 2.22/7,\ s_1'\ min(1;\ supp(S)/c_0) = 6/7$, so $L \in \mathcal{FCS}(s_0', s_1')$. In addition, $\mathcal{G}_{C_1}(L) = \{e, g\} \neq \varnothing$. Then, $L_{C_1} = ceg \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$. First, we consider the formation of $\mathcal{FS}^*_{\subseteq L_{C_1}}$(Line 4). For $L_1 = e$, because $L_U^1 = e,\ L_{U,1} = \varnothing$ and $L_{-,1} = cg$, we have the following left-hand sides: $e + \varnothing + \varnothing,\ e + \varnothing + c,\ e + \varnothing + g,\ e + \varnothing + cg$. For $L_1 = g$, we have $L_U^2 = eg,\ L_{U,2} = e$ and $L_{-,2} = c$. Hence, the $MFS\_RestrictMaxSC$ procedure generates the new left-hand sides $g + \varnothing + \varnothing,\ g + \varnothing + c$. Note that we did not generate $g + e + \varnothing,\ g + e + c$ again as $L_1 \subset g + e$. Next, we concentrate on the generation of the right-hand sides in accordance with left-hand side e (at Line 5). We have $R_1^* = ai$ and $\mathcal{G}_{S_1^*}(S) = \{ae,\ ag\}$. Hence, $R_{\min} = \{a\}$. For a, we have $R_U^1 = a, R_{U,1} = \varnothing, R_{-,1} = ai\backslash(e + a) = i$. $MFS\_RestrictMaxSC$ distinctly generates all the right-hand sides in $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$: a, ai. Thus, we receive two rules: $e{\rightarrow}a,\ e{\rightarrow}ai$. Continuing with the left-hand sides of $ce,\ eg,\ ceg, g$ and gc, we receive ten additional rules of $\mathcal{AR}^*_{\subseteq L_1,\ \subseteq R_1}(L,\ S)$ without any duplicates (see Example 1(b) for the full results).

(b) For the constraints $L_1 = a,\ R_1 = cfhi$, we have $S_1^* = acfhi, C_1 = a, s_0^* = 1.66/7, s_1^* = 5/7$and $supp$ $(S_1^*) = 2/7$. Thus, $\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1) \neq \varnothing$. Let $(L,\ S) = (ai,\ acfhi) \in \mathcal{NFCS}(s_0, s_1, c_0, c_1)$ with $\mathcal{G}(L) = \{a\}, \mathcal{G}(S) = \{cf,\ ch\},\ supp(L) = 5/7$, and $supp(S) = 2/7$. We have $S_{S_1^*} = acfhi \in FCS_{\subseteq S_1^*}(s_0^*, s_1^*)$ because $S \in \mathcal{FCS}(s_0^*, s_1^*)$ and $\mathcal{G}_{S_1^*}(S) = \{cf, ch\} \neq \varnothing$. As $s_0' = supp(S)/c_1 = 2.11/7$ and $s_1' \equiv \min(1; supp(S)/c_0) = 6/7$, we have $L \in \mathcal{FCS}(s_0', s_0^1)$. Furthermore, $\mathcal{G}_{C_1}(L) = \{\text{a}\} \neq \varnothing$. Then, $L_{C_1} = a \in \mathcal{FCS}_{\subseteq C_1}(s_0', s_1')$. For $G_1 = a$, because $L_U^1 = \text{a}, L_{U,1} = \varnothing$ and $L_{-,1} = \varnothing$, we have $\mathcal{FS}^*_{\subseteq L_{C1}} = \{\text{a}\}$. Next, we generate all the rules with the same left-hand side a (Line 5). We have $R_1^* = cfhi$ and $\mathcal{G}_{S_1^*}(S) = \{cf,\ ch\}$. Thus, $R_{\min} = \{cf,\ ch\}$. For cf, we have $R_U^1 = cf, R_{U,1} = \varnothing$ and $R_{\_,1} = cfhi\backslash(a + cf) = hi$. Next, $MFS\_RestrictMaxSC$ generates the right-hand sides of $cf + \varnothing + \varnothing,\ cf + \varnothing + i,\ cf + \varnothing + h$ and $cf + \varnothing + hi$ (in $\mathcal{FS}^*(S_{S_1^*}\backslash L')_{L',\subseteq R_1^*}$) without any duplicates. Therefore, we have four rules: $a \rightarrow cf,\ a \rightarrow cfi, a \rightarrow cfh, a \rightarrow cfhi$. For ch, we have $R_U^2 = cfh, R_{U,2} = f, R_{-,1} = cfhi\backslash(a + cfh) = i$. Then, $MFS\_RestrictMaxSC$ derives $ch{+}\varnothing{+}\varnothing,\ ch{+}\varnothing{+}i$. The right-hand sides of $ch + f + \varnothing,\ ch + f + i$ are not generated again as $cf \subset ch + f$. Finally, we have $\mathcal{AR}^*_{\subseteq L_1,\ \subseteq R_1}(L,\ S) = \{a \rightarrow cf,\ a \rightarrow cfi,\ a \rightarrow cfh,\ a \rightarrow cfhi,\ a \rightarrow ch,\ a \rightarrow chi\}$.

Example 3 implies that our algorithm solely derives the association rules satisfying the constraints of $\mathcal{AR}^*_{\subseteq L_1,\ \subseteq R_1}$ $(L,\ S)$ without generating duplicates and redundant candidates in $\mathcal{AR}(L,\ S)$.

3.3 Completely and distinctly deriving all the association rules with the constraints of $\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$

Theorem 2 follows from Theorem 1 and Proposition 3 as follows. Based on it, the MAR_MaxSC algorithm is proposed

```
$\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)$     $MAR\_MaxSC(s_0,s_1,c_0,c_1,L_1,R_1,\mathcal{LCG})$
$\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)=\varnothing$;  // check the necessary conditions for the nonemptiness of $\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)$:
  if ($s_0>s_1$ or $c_0>c_1$) then return $\varnothing$;
  $S_1^*=L_1\cup R_1$; $C_1=L_1$; $R_0^*=R_0$;
$s_0^*\equiv\max(s_0; c_0.\mathrm{supp}(C_1))$, $s_1^*\equiv s_1$;
$\mathcal{FCS}_{\subseteq S_1^*}(s_0^*,s_1^*)=MFCS\_FromLattice(\mathcal{LCG},S_1^*,s_0^*,s_1^*)$;
  for each ($<S_{S_1^*},\mathrm{supp}(S_{S_1^*}),\mathcal{G}_{S_1^*}(S)>\in\mathcal{FCS}_{\subseteq S_1^*}(s_0^*,s_1^*)$) do {
      $s_0'=\mathrm{supp}(S_{S_1^*})/c_1$;  $s_1'=\min(1;\mathrm{supp}(S_{S_1^*})/c_0)$;
      $\mathcal{FCS}_{\subseteq C_1}(s_0',s_1') = MFCS\_FromLattice(\mathcal{LCG}^S,C_1,s_0',s_1')$;
      for each ($<L_{C_1},\mathrm{supp}(L_{C_1}),\mathcal{G}_{C_1}(L)>\in\mathcal{FCS}_{\subseteq C_1}(s_0',s_1')$) do {
        $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S)= MAR\_MaxSC\_OneClass(L_{C_1},\mathcal{G}_{C_1}(\mathrm{L}),R_1,S_{S_1^*},\mathcal{G}_{S_1^*}(S),\mathcal{S})$;
        $\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)=\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)+\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L,S)$;
    }
 }
return $\mathcal{ARS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)$;
```

Fig. 5 The MAR_MaxSC algorithm

(see Fig. 5) to efficiently mine the set of all association rules with maximum single constraints $\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$ from lattice $\mathcal{LCG}$.

Theorem 2 *(Completely and distinctly deriving all the constrained association rules of $\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1)$). Suppose that (H_1) is satisfied. We have*

$$\mathcal{ARS}_{\subseteq L_1,\ \subseteq R_1}(s_0, s_1, c_0, c_1) = \sum\nolimits_{(L,S)\in\mathcal{NFCS}_{\subseteq L_1,\subseteq R_1}(s_0,s_1,c_0,c_1)} \mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S),$$

where $\mathcal{AR}^*_{\subseteq L_1,\subseteq R_1}(L, S) = \sum\nolimits_{{}'\in Suff\_FS^*_{\subseteq L_{C_1}}}$

$(S, R_1)\{r : L' \to R' : R' \in \mathcal{FS}^*(S_{S_1^*}\backslash L')_{L'}, \subseteq R_1^*\}$.

4 Experimental results

We compare the performance of three methods for mining association rules with constraints as follows. The first method, $PP\_MAR\_MaxSC\_1$, includes three phases: (a) using $dEclat$ to mine frequent itemsets, (b) integrating the constraints into the Gen_Rules [31] algorithm to generate rule candidates, and (c) post-processing to filter out the rules satisfying the constraints. The two remaining methods consist of first mining the lattice $\mathcal{LCG}$ of closed itemsets together with their generators with $Charm\_L$ and $MinimalGenerators$ and then executing the $PP\_MAR\_MaxSC\_2$ and MAR_MaxSC algorithms, respectively. The source code (in C^{++}) for $dEclat$, $Charm\_L$ and $MinimalGenerators$ can be downloaded from http://www.cs.rpi.edu/~zaki/wwwnew/pmwiki.php/Software/Software#patutils (converted to C$^{\#}$). The $PP\_MAR\_MaxSC\_2$ and MAR_MaxSC algorithms are also coded in C$^{\#}$. The experiments were carried out on an i5-2400 CPU 3.10 GHz @ 3.09 GHz PC with 3.16 GB of main memory. Four benchmark databases in the FIMDR (Frequent Itemset Mining Dataset Repository, http://fimi.cs.helsinki.fi/data/) were used in the experiments (Table 2).

Table 2 Database characteristics

Database	# Items	# Transactions	Average length
Connect	129	67,557	43
Mushroom	119	8124	23
Pumsb	7117	49,046	74
Chess	75	3196	37

We fixed the maximum support and confidence thresholds at 1 (as per tradition). For each database and given minimum support, we chose the set $\mathcal{A}^F$ of all frequent items. Ten pairs of maximum constraints (L_1, R_1), were randomly retrieved from $\mathcal{A}^F$ of sizes $|L_1| = p_1\% * |\mathcal{A}^F|$ and $|R_1| = p_1\% * |\mathcal{A}^F|$. We set $p_1 = 30\%$ and $p_2 = 70\%$ for Connect, Pumsb and Chess, and $p_1 = 8\%$, $p_2 = 58\%$ for Mushroom (we achieved similar results for different values of p_1, p_2). We executed the three methods on each database (DB) with two given minimum supports MS (%) and confidences MC (%) and noted the average running times of ten constraint pairs, $T\_MaxSC\_1(DB, MS, MC)$, $T\_MaxSC\_2(DB, MS, MC)$ and $T\_MaxSC(DB, MS, MC)$, called T_MaxSC_1, T_MaxSC_2 and T_MaxSC. All three methods finished their executions on Mushroom, Chess and Pumsb; however, after 12-h running on Connect, $PP\_MAR\_MaxSC\_1$ did not halt.

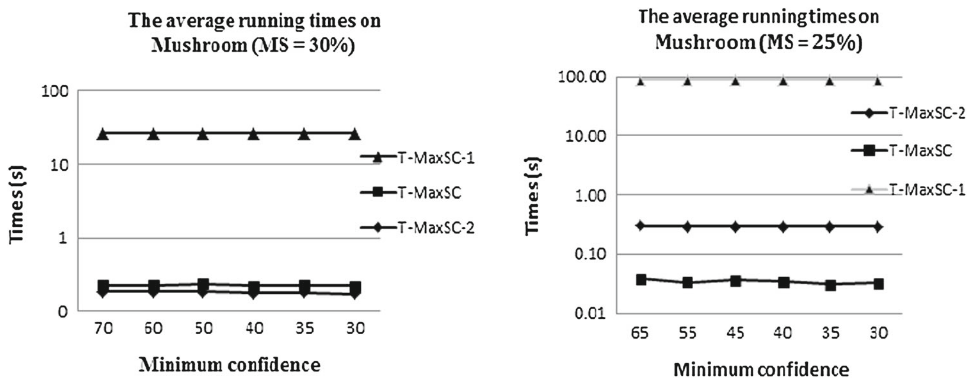


Fig. 6 Average running times of the three methods on Mushroom

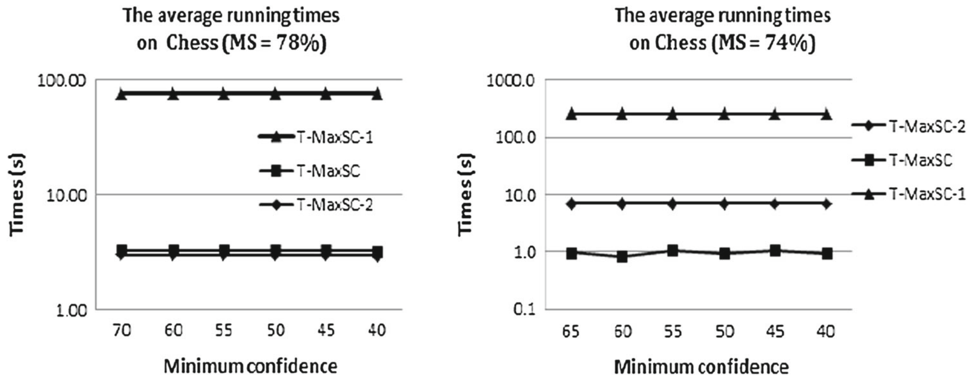


Fig. 7 Average running times of the three methods on Chess

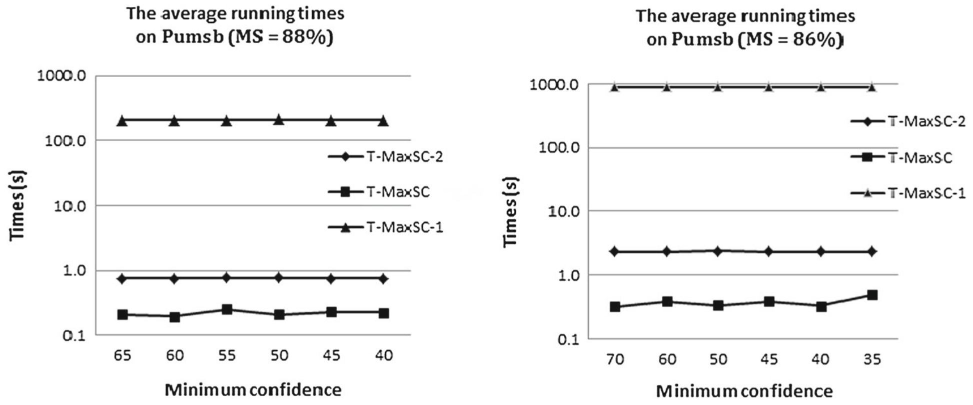


Fig. 8 Average running times of the three methods on Pumsb

Figures 6, 7, 8 and 9 show the average running times of the three methods on several characteristic experiments.

Table 3 shows the RT_1, RT_2 running time ratios (average on different minimum confidences) of two post-processing methods compared with our method for each (DB, MS) pair. More concretely, for (Chess, 78), we have $RT_1 = \sum_{MC\in\{70,65,60,55,50,45,40\}} \frac{T\_MaxSC\_1(Chess,78,MC)}{T\_MaxSC(Chess,78,MC)}/7{=}245.2$ and $RT_2 = \sum_{MC\in\{70,65,60,55,50,45,40\}} \frac{T\_MaxSC\_2(Chess,78,MC)}{T\_MaxSC(Chess,78,MC)}/7 = 10.2$. Thus, our MAR_MaxSC method is 245 and 10 faster than the post-processing methods using $PP\_MAR\_MaxSC\_1$ and $PP\_MAR\_MaxSC\_2$, respectively.

The reason is as follows. Two post-processing methods ($PP\_MAR\_MaxSC\_1$ and $PP\_MAR\_MaxSC\_2$) consume significant times to generate large amounts of rule candidates, however, most of them do not satisfy the maximum single constraints. Indeed, we find that the percent ratios of the numbers of redundant candidate rules to the total of all rules generated by $PP\_MAR\_MaxSC\_1$ and $PP\_MAR\_MaxSC\_2$ are both 99%, approximately, for all above experiments.

5 Conclusions and future work

Two serious problems encountered during the mining of association rules with maximum single constraints are that (1) their cardinality grows exponentially, and the known algorithms for mining them typically generate numerous

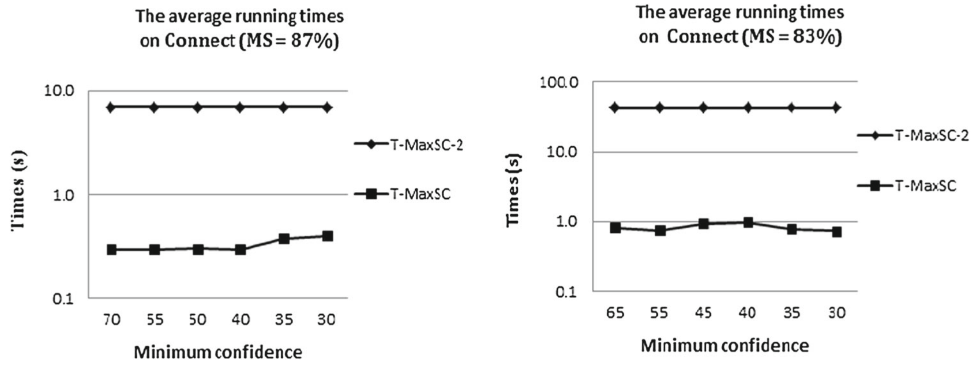


Fig. 9 Average running times of $PP\_MAR\_MaxSC\_2$ and MAR_MaxSC on Connect

Table 3 Running time reduction ratios

DB	MS (%)	Thresholds of MC (%)	RT_1	RT_2
Chess	78	70, 65, 60, 55, 50, 45, 40	245.2	10.2
	76		311.9	10.2
	74		273.1	7.4
	72		266.9	6.0
	70		286.2	5.7
Connect	89	70, 65, 60, 55, 50, 45, 40, 35, 30		7.7
	87			22.9
	85			38.4
	83			53.0
Pumsb	88	70, 65, 60, 55, 50, 45, 40, 35	969.8	3.5
	86		2470.9	6.4
Mushroom	30	70, 65, 60, 55, 50, 45, 40, 35, 30	572.3	4.0
	25		2738.1	9.1

redundancies and duplicates and (2) their constraints are frequently modified. We generate a solution to this problem with a mathematical approach. Starting with a lattice $\mathcal{LCG}$ of closed itemset and their generators, which are suitable for use with frequently modified constraints, we efficiently extract the corresponding frequent sub itemsets. Based on this lattice, a proposed suitable equivalence relation partitions the set of association rules with maximum single constraints into disjoint equivalence classes. We then use the closed itemsets and their generators (of generally low cardinality) to uniquely represent the rules in each class. After studying our results, we propose using the MAR_MaxSC algorithm to distinctly and completely produce all the constrained rules in each rule class.

Our approach can be adapted to process on big data because it can be exploited in parallel and distributed environments. In the future, we will get bigger data sets to test the approach. As an interesting extension, we plan to adapt a Hadoop Map/Reduce framework. In addition, it is important for us to apply our approach to mining problems with additional types of constraints.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499 (1994)
2. Agrawal, R., Imielinski, T., Swami, N.: Mining association rules between sets of items in large datasets. In: Proceedings of the 1993 ACM SIGMOID, pp. 207–216 (1993)
3. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, AAAI Press, pp. 307–328 (1996)
4. Anh, T., Hai, D., Tin, T., Bac, L.: Efficient algorithms for mining frequent itemsets with constraint. In: Proceedings of the 3rd International Conference on Knowledge and Systems Engineering, pp. 19–25 (2011)
5. Anh, T., Hai, D., Tin, T., Bac, L.: Mining frequent itemsets with dualistic constraints. In: PRICAI 2012, LNAI, vol. 7458, pp. 807–813. Springer (2012)
6. Anh, T., Tin, T., Bac, L., Hai, D.: Mining Association Rules Restricted on Constraint. In Proceedings of the IEEE-RIVF International Conference on Computing and Communication Technologies 2012, pp. 51–56 (2012)
7. Anh T., Tin T., Bac L.: Structures of association rule set. In: LNAI, vol. 7197, Part II, pp. 361–370. Springer (2012)
8. Anh T., Tin T., Bac L.: An approach for mining concurrently closed itemsets and generators. In: Advanced Computational Methods for Knowledge Engineering, SCI, vol. 479, pp. 355–366. Springer (2013)
9. Anh, T., Tin, T., Bac, L.: An approach for mining association rules intersected with constraint itemsets. Adv. Intell. Syst. Comput. **245**, 351–363 (2013b)
10. Bayardo, R.J. Jr.: Efficiently mining long patterns from databases. In: Proceedings of the ACM-SIGMOD 1998 International Conference on Management of Data, pp. 85–93 (1998)

11. Burdick, D., Calimlim, M., Gehrke, J.: MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proceedings of 2001 ICDE, pp. 443–452 (2001)
12. Bayardo, R.J., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. In: Data Mining and Knowledge Discovery, vol. 4, no. (2/3), pp. 217–240. Kluwer Academic Pub. (2000)
13. Cong, G., Liu, B.: Speed-up iterative frequent itemset mining with constraint changes. In: Proceedings of ICDM 2002, pp. 107–114 (2002)
14. Cristofor, L., Simovici, D.: Generating an informative cover for association rules. In: Proceedings of the IEEE International Conference on Data Mining 2002, pp. 597–600 (2002)
15. Das, A., Ng, W.-K., Woon, Y.-K.: Rapid association rule mining. In: Proceedings of 10th International conference on Information and knowledge management, pp. 474–481. ACM Press (2001)
16. Ganter, B., Wille, R., Franzke, C.: Formal concept analysis: mathematical foundations. Springer, New York (1997)
17. Grahne, G., Zhu, J.: High performance mining of maximal frequent itemsets. In: Proceedings of SIAM 2003 Workshop on High Performance Data Mining: Pervasive and Data Stream Mining (2003)
18. Goethals, B., Zaki, M.J.: Advances in frequent itemset mining implementations. In: Report on FIMI 2003, ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 109–117 (2004)
19. Gouda, K., Zaki, M.J.: Genmax: an effcient algorithm for mining maximal frequent itemsets. Data Min. Knowl. Discov. **11**(3), 223–242 (2005)
20. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. **8**(1), 53–87 (2004)
21. Hai, D., Tin, T., Bay, V.: An efficient method for mining frequent itemsets with double constraints. Int. J. Eng. Appl. Artif. Intell. **27**, 148–154 (2013)
22. Hai, D., Tin, T.: An efficient method for mining association rules based on minimum single constraints. Vietnam J. Comput. Sci. **2**(2), 67–83 (2015)
23. Ho, B.: An approach to concept formation based on formal concept analysis. IEICE Trans. Inf. Syst. **E78–D**(5), 553–579 (1995)
24. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Proceeding of the 3rd CIKM Conference, pp. 401–407 (1994)
25. Li, G., Hamilton, H.J.: Basic association rules. In: Proceedings of the 4th SIAM International Conference on Data Mining, pp. 166–177 (2004)
26. Lee, A.J., Lin, W.C., Wang, C.S.: Mining association rule with multi-dimensional constraints. J. Syst. Softw. **79**(1), 79–92 (2006)
27. Mannila, H., Toivonen, H., Verkamo, I.A.: Efficient algorithms for discovering association rules. In: Workshop on Knowledge Discovery in Databases 1994, pp. 181–192 (1994)
28. Nguyen, R.T., Lakshmanan, V.S., Han, J., Pang, A.: Exploratory Mining and Pruning Optimizations of Constrained Association Rules. In: Proceedings of the 1998 ACM-SIG-MOD International Conference on the Management of Data, pp. 13–24 (1998)
29. Oded, M., Lior, R.: Data mining and knowledge discovery Handbook. Springer, New York (2010)
30. Park, J.S., Chen, M.S., Yu, P.S.: An effective hash based algorithm for mining association rules. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 175–186 (1995)
31. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattice. Inf. Syst. **24**(1), 25–46 (1999)
32. Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. J. Intell. Inf. Syst. **24**(1), 29–60 (2005)
33. Pei, J., Han, J., Lakshmanan, V.S.: Pushing convertible constraints in frequent itemset mining. Data Min. Knowl. Discov. **8**(3), 227–252 (2004)
34. Szathmary, L., Valtchev, P., Napoli, A.: Efficient vertical mining of frequent closed itemsets and generators. In: Proceedigns of IDA 2009, pp. 393–404 (2009)
35. Srikant, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Proceedings of KDD 1997, pp. 67–73 (1997)
36. Tin, T., Anh, T.: Structure of set of association rules based on concept lattice. In: Advances in Intelligent Information and Database Systems, SCI, vol. 283, pp. 217–227. Springer (2010)
37. Tin, T., Anh, T., Thong, T.: Structure of association rule set based on min-min basic rules. In: Proceedings of the International Conference on Computing and Communication Technologies 2010, pp. 83–88 (2010)
38. Wille, R.: Concept lattices and conceptual knowledge systems. Comput. Math. Appl. **23**(6–9), 493–515 (1992)
39. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, pp. 283–286 (1997)
40. Zaki, M. J., Gouda, K.: Fast vertical mining using diffsets. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge discovery and Data Mining, pp. 326–335. ACM (2003)
41. Zaki, M.J.: Mining non-redundant association rules. Data Min. Knowl. Discov. **9**(3), 223–248 (2004)
42. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Trans. Knowl. Data Eng. **17**(4), 462–478 (2005)