Knowledge-Based Systems 41 (2013) 77–88

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Interrelation analysis of celestial spectra data using constrained frequent pattern trees

Jifu Zhang [a,*], Xujun Zhao [a], Sulan Zhang [a], Shu Yin [b], Xiao Qin [b], Senior Member, IEEE

[a] School of Computer Science and Technology, Department of Computer, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, PR China
[b] Department of Science and Software Engineering, Auburn University, Auburn, AL 36849-5347, USA

ARTICLE INFO

Article history:
Received 24 March 2012
Received in revised form 25 November 2012
Accepted 28 December 2012
Available online 10 January 2013

Keywords:
Celestial spectra data
Interrelation analysis
Performance evaluation
I/O performance
Association rule
Constrained frequent pattern trees

ABSTRACT

Association rule mining, in which generating frequent patterns is a key step, is an effective way of identifying inherent and unknown interrelationships between characteristics of celestial spectra data and its physicochemical properties. In this study, we first make use of the first-order predicate logic to represent knowledge derived from celestial spectra data. Next, we propose a concept of constrained frequent pattern trees (CFP) along with an algorithm used to construct CFPs, aiming to improve the efficiency and pertinence of association rule mining. Finally, we quantitatively evaluate the CPU and I/O performance of our novel interrelation analysis method using a variety of real-world data sets. Our experimental results show that it is practical to study the laws of celestial bodies using our new interrelation analysis method to discover correlations between celestial spectra data characteristics and the physicochemical properties.

1. Introduction

In year 2008, a large sky area multi-object fiber spectroscopic telescope (LAMOST) was built by the National Observatory of China [1]. Compared with other telescopes in the world, LAMOST is able to acquire celestial spectra data with the highest rate. LAMOST – enabling large-scale optical spectroscopic observations – is a breakthrough in combining large aperture with wide field of view in optical telescope. The amount of spectra data acquired by LAMOST is as large as approximately 4 TB [2]. It is challenging to analyze the mass amount of data collected by LAMOST; therefore, there is a pressing need to develop novel data analysis tools to be integrated into data processing systems located in observatory headquarters.

Existing approaches to data analysis and recognition of celestial spectra data include cross correlation analysis and principal component analysis (PCA), artificial neural networks, wavelet transformation, Bayesian statistics, and the like. Cheeseman et al. investigated an automatic classification scheme using Bayesian statistics [3]. Their classification method allows researchers to discover some unnoticed spectrum types and spectral lines in the unique classification results. Gulati et al. adopted the two layers-BP (Back Propagation) neural network to classify star spectrum quadratic forms [4]. Making use of neural networks, Weaver and Torres-Dodgen [5], Singh et al. [6], Vieira and Ponz [7], and Bailer-Jones et al. [8] successfully studied the MK (Morgan–Keenan) classification and recognition system for stars. The aforementioned strategies are suitable for medium–low resolution (i.e., 0.1–1.5 nm) ultraviolet and spectra in the optical band. The classification precision of the above methods can reach the second type of the star spectrum. Although these schemes differ in network structures and the selection of neurons, learning time reduction is a common issue addressed in these studies. However, the existing solutions are inadequate for star recognition problems in the context of special spectrum types.

When it comes to recognition of clusters and galaxy systems, astronomers widely rely on the well-known Hubble classification of galaxies. For example, since redshifts are observed in almost all the spectra of extragalactic celestial bodies, Connolly et al. [9], Gaspar et al. [10], and Zaritsky et al. [11] respectively adopted the principal component analysis to implement automatic recognition tools for the galaxies spectrum in which redshift values have been discovered. Folkes et al. [12] applied the three principal components to classify the spectra of the 2dF redshift survey into five broad-spectrum categories. Qin et al. studied two efficient stellar spectra classification methods using automatic pattern recognitions [2,13]. Li [14], Zhao [15] and Liu et al. [16] investigated automated recognitions of active galactic nuclei, quasars, starburst galaxies, and normal galaxies by employing machine learning algorithms.Zhang et al. introduced a new term – concept lattice, which is a core component in Zhang's novel approach to recognizing celestial spectra outliers [17]. In addition, Zhang and Cai designed and implemented a celestial spectra outlier mining system for

* Corresponding author.
E-mail address: jifuzh@sina.com (J. Zhang).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.12.013

LAMOST [18]. Because of the limited cognition of the universe in the astronomical field, one of the main goals of the LAMOST survey project is to discover new special celestial bodies. There is a pressing need to develop new data mining systems to extract special unknown celestial bodies and to discover the laws of celestial bodies from the huge amount of celestial spectra data collected by LAMOST.

In the data mining field, association rules can be discovered having domain knowledge specified as a minimum support threshold [31]. The accuracy of a process in setting up this minimum support threshold directly affects the number and the quality of discovered association rules. It is not uncommon that the number of association rules may miss a few interesting rules and the rules' quality may need further analysis. Therefore, using these rules to make any decision may lead to risky actions [31]. Concise representations of frequent item sets generated using minimum support thresholds – sacrifice readability and direct interpretability by a data analyst of concise patterns extracted [32]. Efficient algorithms for mining frequent itemsets are crucial for mining association rules as well as for many other data mining tasks [33]. Therefore, it is inconvenient and sometimes difficult for users to extract useful association rules in which the users are interested.

Mining association rules among sets of items play an important role in a wide range of data mining systems (see [19] for details on association rules). Association rules effectively describe interrelations among the attributes of data sets that satisfy the minimum support thresholds and the minimum confidence thresholds specified by users. Therefore, it is feasible and valuable to make use of association rules to describe existing and unknown correlations between the characteristics of celestial spectra data and its physicochemical properties. In doing so, researchers are able to discover new laws of celestial bodies from the celestial spectra data.

In this study, we make the following contributions:

- We are among the first to propose a concept of constrained frequent pattern trees to substantially improve the efficiency and quality of interrelation analysis of celestial spectra data.
- We develop a new algorithm to construct constrained frequent pattern trees or FP trees. In this algorithm, the celestial spectra knowledge is represented by first-order predicate logic. The constrained FP trees can be used by interrelation analysis methods to provide an effective way of searching the unknown laws of celestial bodies.
- We evaluate the performance of the algorithm used to construct constrained FP trees. We pay particular attention on the I/O performance of the data analysis system.
- We develop a celestial spectra data analysis system to validate the idea of constrained FP trees in the context of interrelation analysis of spectra data sets. We demonstratively show that our system can efficiently discover interrelations among the celestial spectra data characteristics and identify correlations among the physicochemical properties.

The rest of the paper is organized as follows. Basic concepts on association rules are described in Section 2. Section 3 presents the process of celestial spectra data and the acquisition of celestial spectra knowledge. Section 4 presents a novel interrelation analysis method for celestial spectra data. To evaluate performance of our proposed data analysis approach, in Section 5 we analyse experimental results. We pay particular attention on the I/O performance of the data analysis module. Finally, Section 6 concludes the paper with future research directions.

2. Association rules

Let DB be a database of transactions, $I = \{I_1, I_2, \ldots, I_m\}$ be a set of m transaction item sets in DB. Each transaction T in DB is represented as a binary vector, with $t[j] = 1$ if T bought the item I_j in I, and $t[j] = 0$ otherwise.

Definition 1. We refer pattern P as a subset of I. Thus, we have $P = I_1 \infty I_2 \infty, \ldots, \infty I_k$ where ∞ is the union operator of patterns or items, $I_i \in I$ $(i = 1, 2, \ldots, k)$, and the length of pattern P is the number of items in P. For example, the length of the above pattern P is k, because there are k items in P.

A transaction t satisfies pattern P if for all item I_j in pattern P, $t[j] = 1$.

Definition 2. Let $|DB|$ be the total number of transactions in database DB. Pattern P has a support $0 \leqslant \sigma \leqslant 1$ with respect to database DB if and only if at least $\sigma \times |DB|$ number of transactions in DB satisfy pattern P.

Let $|t \to P|$ be the number of transactions that satisfies pattern P. We define *support* σ as Eq. (1), where $\sigma(P/DB)$ is a fraction of transactions in DB that satisfies the given pattern. Thus, we have:

$$\sigma(P/DB) = |t \to P|/|DB|. \tag{1}$$

We say two patterns U and V are disjoint patterns if and only if their intersection is an empty set, i.e., $\{U_1, U_2, \ldots, U_k\} \cap \{V_1, V_2, \ldots, V_m\} = \emptyset$, where $U = U_1 \infty U_2 \infty \cdots \infty U_k$, $V = V_1 \infty V_2 \infty \cdots \infty V_m$.

Definition 3. Let U and V be two disjoint patterns, and $U \Rightarrow V$ be an association rule in DB, we define the confidence of the association rule as below:

$$\psi(U \Rightarrow V/DB) = \sigma(U \infty V/DB)/\sigma(U/DB), \tag{2}$$

where $\sigma(U \infty V/DB)$ and $\sigma(U/DB)$ can be derived from the two support factors defined in Eq. (1).

Definition 4. Let $\sigma_{\min}$ be the threshold of the minimum support factor, then the k-frequent pattern set L_k in database DB is defined as:

$$L_k = \{A_1 \infty A_2 \cdots \infty A_k | A_i \in I, \sigma(A_1 \infty A_2 \cdots \infty A_k/DB) \geqslant \sigma_{\min}\} \tag{3}$$

To extract association rules from DB, one has to specify the minimum support threshold $\sigma_{\min}$ and the minimum confidence threshold $\psi_{\min}$. The goal of mining association rules is to search all the association rules whose support factor is greater than $\sigma_{\min}$ and confidence factor is greater than $\psi_{\min}$. Thus, we need to search any association rule (e.g., $A \Rightarrow B$) that satisfies the following two conditions:

$$\sigma(A \infty B/DB) \geqslant \sigma_{\min} \quad \text{and} \quad \psi(A \Rightarrow B/DB) \geqslant \psi_{\min}. \tag{4}$$

The value of $\psi(A \Rightarrow B/DB)$ in the above condition can be derived from two support factors computed by $\sigma(A \infty B/DB)$ and $\sigma(A/DB)$, and mining association rules (e.g., $A \Rightarrow B$) lies in the generation of the frequent-pattern sets formally defined in Eq. (3). The efficiency of mining association rules heavily depends on ways of generating frequent-pattern sets. In the past decade, much attention has been paid to approaches to improving performance of generating frequent-pattern sets (see, for example, [19–22]).

Frequent-pattern generating algorithms can be categorized into two groups-Apriori [19] and Frequent-Pattern tree (FP-tree) [20–22]. The main drawback of the Apriori algorithms is twofold. First, generating enormous number of candidate items increases the amount of data traversing in databases during the process of generating frequent patterns. Second, increasing the number of candidate items inevitably enlarges I/O processing time spent traversing the databases. Such a drawback makes it difficult for the Apriori algorithm to deal with massive amount of high-dimensional data. Unlike the Apriori algorithms, the Frequent-Pattern tree

algorithms or FP-tree proposed by Han et al. have no need to generate candidate items [20]. A salient feature of the FP-tree algorithms is to compress and organize frequent sets into a tree called frequent-pattern tree, from which frequent patterns can be efficiently extracted. There are two major advantages of applying FP-tree to generate frequent-pattern sets. First, the FP-tree algorithms only have to construct and recursively traverse FP trees as well as conditional FP trees, meaning that no candidate item set should be generated and maintained. Second, the FP-tree algorithms only traverse a transaction database twice. Thus, in the first traversing over the database, frequent l-item sets are automatically generated; in the second traversing a frequent-pattern tree is constructed. Traversing databases only twice helps reducing the time spent on accessing database. A disadvantage of the FP-tree algorithms is that the algorithms require a large main memory capacity to construct and store frequent-pattern trees.

Definition 5 (*FP-tree [20]*). A frequent-pattern tree (or FP-tree in short) is a tree structure defined below.

1. It consists of one root labeled as "null", a set of item-prefix sub-trees as the children of the root, and a frequent-item-header table.
2. Each node in the item-prefix sub-tree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.
3. Each entry in the frequent-item-header table consists of two fields, (1) item-name and (2) head of node-link (a pointer pointing to the first node in the FP-tree carrying the item-name).

Lemma 1. *Let DB be a transaction database and σ_{min} be the minimum support threshold. The frequent-pattern sets extracted from the FP tree necessarily are the frequent pattern sets of transaction in DB. (See [20] for the proof of Lemma 1.)*

3. Preprocess of celestial spectra data and celestial spectra knowledge

3.1. Normalized and discrete preprocess

Celestial spectra is an ordered arrangement on the electromagnetic radiation of celestial bodies arranged in wavelengths, containing important physical information of celestial bodies. Such physical information includes chemical composition, surface temperature, diameter, quality, luminosity and the like.

Through analyzing the information of celestial spectra, astronomers and scientists of celestial bodies not only can study distribution characteristics of the universe, but also can investigate intriguing scientific problems such as the forming of celestial bodies and evolvement with the time. Therefore, the celestial-spectra data process and analysis play an important role in astronomy, especially when it comes to the physics of celestial bodies.

Spectra data are series of continuous data made up of (1) different flux to which each wavelength is corresponding and (2) physicochemical properties of this spectrum. Because the wide value range of the flux (e.g., the flux varies anywhere from 10^{-19} to 10^{-3}) affects the efficiency and precision of computing, the continuous spectra are required to be normalized to eliminate the discrepancy in orders of magnitude. Furthermore, the flux values have to be discrete values, and the reason is twofold. First, the raw flux values are inadequate to describe the characteristics of spectrum straightforwardly. Second, unprocessed flux values cannot meet the needs of data-mining systems. Fig. 1 shows star spectra curves drawn using the classification standard of spectrum type.

The polynomial approximation scheme is a common approach to extracting continuous spectra. In addition to the polynomial approximation scheme, other ways of extracting spectra data include morphological filters, wavelet transform, median filters, etc. Morphological filters are nonlinear filters that can transform geometry features of signals. However, morphological filters are normally complicated when it comes to the implementation of the filters. The time complexity of wavelet transform is high; therefore, it is not efficient to process mass amount of spectra data. Median filters (see [23] for details on the median filters) are a typical nonlinear digital signal process technique widely used in the normalization process of spectra data. Therefore, in our system described in this paper, spectra data are normalized using the median filter. In the median filter, one has to configure a window, within which all specimens are first traversed. Then, the center point value in the window is replaced with the median value of the original values in the window. In doing so, high quality spectra images can be created.

Let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ be a set of celestial spectra data from which association rules are extracted. Let v be the radius of median-filter windows, y_i be the median of the values in the flux window in which flux x_i is residing. Thus, y_i can be expressed as follows, where $(x_{i-v}, \ldots, x_i, \ldots, x_{i+v})$ is the flux window containing flux x_i:

$$y_i = Median(x_{i-v}, \ldots, x_i, \ldots, x_{i+v}).$$

The discretization of celestial spectra data includes discretizations of flux and physicochemical properties. In the discretization process of star spectrum flux, we consider wave intensities (i.e., the flux intensities) and peak widths of wavelengths. We rely on two characteristic variables, I and W, to process celestial spectra data. Variable I represents peak intensity and W represents peak width of a certain wavelength respectively. Thus, each wavelength in the star spectra data can be converted into the characteristic data denoted by variable I and W.

Let us denote wavelengths contained in a certain set of star spectrum images as $A = [A_1, A_2, \ldots, A_n]$, where A_i is the number of wavelengths extracted from the ith spectrum, and n represents the total number of wavelengths. A_i in A can be modeled as $A_i = \langle I_i, W_i \rangle$, where I_i and W_i are the characteristic data of A_i, $i = 1, 2, 3, \ldots, n$.

Recent studies show that wave intensities are divided into five groups: strong, median-strong, weak, median-weak, and null. Peak widths are categorized into three camps: narrow, wide, and super wide. During the course of discretization, two-dimensional characteristic vectors are converted and stored as one-dimensional data. Combining five peak-intensity levels and three width levels, we

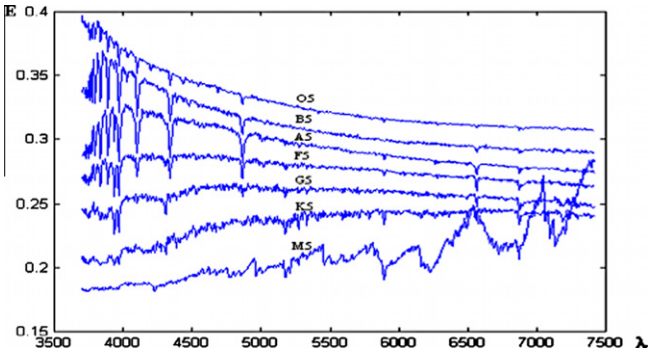

Fig. 1. Star spectra curves drawn using the classification standard of spectrum type.

Table 1
Discretization parameters of star spectra data.

Attribute	Parameters Ranges (discretization values)
Peak width	<5 (narrow), 5–10 (wide), >10 (super wide)
Peak intensity	<0.5 (weak), 0.5–1 (general), 1–5 (stronger), >5 (very strong)
Temperature	2000–2500 (1), 2500–3000 (2), 3000–3500 (3), 3500–4000 (4), 4000–4500 (5) ...
Luminosity	−1 to 0 (1), 0–1.1 (2), 1.1–2.6 (3), 2.6–3.6 (4), 3.6–4.1(5)
Chemical abundance	−5 to −3 (1), −3 to −0.5 (2), −0.5 to −0.1 (3), −0.1 to 0.3 (4), 0.3–0.7 (5)
Micro-turbulence	<1(1), ⩾1(2)

have the following thirteen cases: strong-narrow, strong-wide, strong-super wide, median-strong-narrow, median-strong-wide, median-strong-super wide, median-weak-narrow, median-weak - wide, median-weak-super wide, weak-narrow, weak-wide, weak-super wide, and null. We use thirteen single characters (i.e., 1, 2, ..., 9, *A*, *B*, *C*, and 0) to denote the corresponding flux values.

The discretization of physicochemical properties mainly includes temperature, luminosity, chemical abundance, and micro-turbulence. Let S denote a set of all possible physicochemical properties. Thus, $S = \{S_j\}$, $j = 1, 2, 3, \ldots, t$, where S_j specifies the physicochemical property of the jth star spectrum. Each physicochemical property can be represented as an array of conditions as well as single characters summarized in Table 1 below. Please note that attributes in the left column of Table 1 are defined by the conditions given in the right column of Table 1.

3.2. First-order predicate logic and celestial spectra knowledge

The generation of frequent pattern sets is not only an important step toward mining association rules, but also a critical process of improving mining efficiency. This argument is especially true when it comes to the massive amount of high dimensional celestial spectra data. By the virtue of experience accumulated with long time studies, astronomers have deep understanding of spectrum characteristics that are important to the analysis and recognition of spectra data. More specifically, it is fundamental to enhance the mining efficiency and the pertinence of mining results when the celestial spectra knowledge is incorporated into the generating process of FP trees. Taking the celestial spectra knowledge into account, one can substantially reduce the scale of FP trees, because only intriguing frequent patterns are included in the FP trees.

In what follows, we describe the fact that tradeoffs between data mining accuracy and performance can be made by adding different types of constraints:

- It takes a great deal of time and main memory space to construct FP-Trees. The performance of a module that constructs FP-trees can sharply decrease when applications scale up, because big FP-Trees include a large number of frequent patterns and most of these patterns have no applicable value from users' perspective.
- Importantly, in many real-world applications, users not only have good understandings of data sets, but also have certain data-mining preferences. Hence, it is feasible to take the understandings and preferences as a priori information or background knowledge, which can be merged into the FP-Tree constructing process. Background knowledge offered by users can help in effectively improving the performance of data mining applications by virtue of satisfying data-mining requirements raised by users.
- A few interesting frequent patterns could be missed, if background knowledge is not complete for described data sets. It is worth noting that the overhead is going up when we increase different types of constraints to offer background knowledge. Thus, the high accuracy of data mining processes can be achieved at the cost of performance due to high overhead introduced by background knowledge.

First-order predicate logic is a formal language system, which applies the logic method to study the law of reasoning, and it is adequate for representing the states of objects, attributes, and factual knowledge of concepts. In addition, the first-order predicate logic can be used to express certain consequence relations and rules among objects. A set of celestial spectra data contains characteristics and physicochemical properties of spectrum. After the pretreatment process of normalized and discretization, each wavelength flux value and its physicochemical properties are converted into special discretization values. Generally speaking, the knowledge of celestial spectra is represented as constrained relationships among objects and attributes, attributes and attributes of celestial spectra datasets. Therefore, it is practical to make use of the first-order predicate logic to express the celestial spectra knowledge.

Let us consider a celestial spectra database $DB = \{I_1, I_2, \ldots, I_m\}$, which is a set of m spectra datasets. The ith spectra dataset I_i in DB is a group of subsets in I, i.e., $I_i \subseteq I$, where $I = \{A_1, A_2, \ldots, A_n, S_1, S_2, \ldots, S_t\}$. A_i is the discretization characteristic attributes at the ith wavelength, and S_j is the discretization attributes of the jth physicochemical property.

We can express celestial spectra knowledge by making use of the first-order predicate logic as the following two steps. First, we define the predicates of celestial spectra knowledge, meaning that each predicate must be well specified. Second, these predicates are combined with logic operation symbols to form a predicate formula representing a piece of complete celestial spectra knowledge.

The predicates of celestial spectra knowledge can be defined using the following three predicates:

- Interesting ($f(r)$),
- Support ($f(r), \sigma_{\min}$) and,
- Interested ($f(r)$).

where r is an individual variable in the relation table name of DB; f is a function name specifying the mapping from a relation table to attributes; and $f(r)$ is an attribute set of the relation table r, which is the item subsets of the discretization celestial spectra. The above three types of predicates are detailed as follows:

1. Interesting (f (r)) is a concluding predicate indicating the fact that users are interested in spectra patterns that include item sets $f(r)$. If $f(r)$ is composed of the discretization characteristic attributes at the spectrum wavelength, then users are concerned with patterns in which the antecedent parts in association rules contain $f(r)$. In the case that $f(r)$ outlines the discretization attributes of physicochemical properties, users are concerned with patterns in which the consequent parts in association rules contain $f(r)$.
2. The predicate Support ($f(r), \sigma_{\min}$) indicates that the support degree of spectrum item subsets $f(r)$ is greater than the given minimum support threshold $\sigma_{\min}$.
3. Suppose that spectrum item subsets $f(r)$ is the frequent pattern subsets of the historical mining, then the predicate Interested ($f(r)$) indicates that $f(r)$ is an interesting pattern subsets in the next mining phase.

The FP-tree algorithms only have to construct and recursively traverse FP trees as well as conditional FP trees, meaning that no candidate item set should be generated and maintained. The FP-tree algorithms simply traverse a transaction database twice. Thus, in the first traversing over the database, frequent l-item sets are automatically generated; in the second traversing, a frequent-pattern tree is constructed. Traversing databases twice helps in reducing data base accessing time. A disadvantage of the FP-tree algorithms lies in their requirement of large main memory capacity to construct and store frequent-pattern trees.

Historical mining is meant a mining result generated by traversing database twice to calculate both CFP tree and frequent patterns. The next mining phase – the second phase that traverses database – is in charge of generating CFP tree from 1-frequent patterns.

Definition 6. Let r be a variable in the relation table name of a celestial spectra database, f be a function specifying a mapping from the relation table to attributes, and $\sigma_{\min}$ be the minimum support threshold ($0 \leqslant \sigma_{\min} \leqslant 1$). Celestial spectra knowledge G can be defined as the well formed formula that contains the following predicate formulas through the combination with logic operation symbols.

(1) Interesting ($f(r)$)
(2) support ($f(r), \sigma_{\min}$) → Interesting ($f(r)$)
(3) Interested ($f(r)$) → Interesting ($f(r)$)

The above three predicate formulas are explained as follows:

1. The first predicate formula represents the celestial spectra item sets that the users are interested in. If $f(r)$ describes the characteristics sets of the spectrum wave, then users are interested in spectrum patterns that the antecedent parts in association rules contain $f(r)$. When $f(r)$ describes the physicochemical property sets, then users are interested in the spectrum patterns that the consequent parts in association rules contain $f(r)$.
2. The second predicate formula signifies that in a historical mining, if the support degree of celestial spectra item sets $f(r)$ is greater than $\sigma_{\min}$, then in the new mining $f(r)$ is also the interesting spectrum item sets in which users are interested. Note that $f(r)$ may be either the spectrum characteristic attributes sets of the discretization flux or the discretization attributes sets of physicochemical properties.
3. The third predicate formula shows that if he spectrum item subsets $f(r)$ is the frequent pattern subsets of the historical mining, then the predicate Interested ($f(r)$) implies that $f(r)$ is an interesting pattern subsets in the next mining.

In accordance to the resolution-deduction principle (see [24] for details), a predicate formula is often converted to its equivalent clause set in the sense of an unsatisfiable condition. Since an atomic predicate formula depicts one pattern of association rules (see Definition 6), those clause sets converted from predicate formula – describing background knowledge – can effectively model a group of interesting patterns. In the process of constructing FP trees, it is easier to deal with clause sets than predicate formula. Also, it is straightforward to merged clause sets into the process of constructing FP trees. Since the background knowledge is generalized from the understandings and preferences for data sets, predicate formula describing background knowledge is generally believed to be logically true. Therefore, the unsatisfiable condition is satisfied.

Therefore, celestial spectra knowledge G can be simplified as a conjunctive normal form denoted by a clause set S.

Theorem 1. *Let S be a clause set representing celestial spectra knowledge G. For any clause in S (i.e., $s \in S$), s is a disjunctive form of finite atomic predicates, which is expressed as a combination of predicates like Interesting ($f(r)$), support ($f(r)$, σ_{min}), and Interested ($f(r)$). Given $f1(r1)$, $f2(r1)$, $f3(r1)$, and σ_{min}, Interesting ($f1(r1)$) $\wedge$ support ($f2(r1)$,σ_{min}) $\wedge$ Interested ($f3(r1)$) implies that $f1(r1) \cap f2(r1) \cap f3(r1)$ is a celestial spectra pattern in which users are interested. Thus, we have*

$$\text{Interesting}\ (f1(r1)) \wedge \text{support}(f2(r1), \sigma_{\min}) \wedge \text{Interested}(f3(r1)) \Rightarrow \text{Interesting}\ (f1(r1) \cap f2(r1) \cap f3(r1))$$

Proof. Recall that celestial spectra knowledge G is a well formed formula composed of predicates like Interesting ($f(r)$), support ($f(r)$, $\sigma_{\min}$), and Interested ($f(r)$) through the combination with the logic operation symbols (see Definition 6). Therefore, the characters in the clause are expressed with the limited number of predicates of Interesting ($f(r)$), support ($f(r)$,$\sigma_{\min}$) and Interested ($f(r)$). We can prove that any one of the clauses is some limited characters of disjunctive normal form (see details in [24]).

From Definitions 1 and 6, we have Interesting ($f1(r1)$) $\wedge$ support ($f2(r1)$,$\sigma_{\min}$) $\wedge$ Interested ($f3(r1)$) $\Rightarrow$ Interesting ($f1(r1)$) $\wedge$ Interesting ($f2(r1)$) $\wedge$ Interesting ($f3(r1)$) $\Rightarrow$ Interesting ($f1(r1) \cap f2(r1) \cap f3(r1)$). Thus, $f1(r1) \cap f2(r1) \cap f3(r1)$ is a celestial spectra pattern in which users are interested. □

Theorem 2. *Let S be a clause set denoting celestial spectra background knowledge G, and s be any clause in S (i.e., $s \in S$), then s is a celestial spectra pattern in which users are interested.*

Proof. A clause is some limited characters of disjunctive normal form [24]. Therefore, any clause denoting celestial spectra knowledge G is made up of some limited characters of disjunctive normal forms containing predicates like Interesting ($f(r)$), support ($f(r)$, $\sigma_{\min}$), and Interested ($f(r)$). From Theorem 1, we prove that s is a celestial spectra pattern in which users are interested. □

4. Interrelation analysis of celestial spectra data

4.1. Constrained frequent-pattern tree construction

Extracting association rules is a critical issue in a wide variety of data mining applications. Association rule extraction not only discovers interrelations among different data attributes, but also searches dependent relations of multiple attributes according to any given support threshold and confidence threshold. The extraction of association rules describes association relations among the attributes of data in *DB*.

When it comes to a large amount of data in *DB*, improving efficiency is a challenging problem in the extraction of association rules from *DB*. Mining association rules mainly involves two steps. The first step is generating all important frequent pattern sets; the second step is extracting association rules from the frequent pattern sets. The second step, an uncomplicated process, can be easily and inexpensively implemented. Therefore, the mining efficiency of association rules lies mainly on the performance of the first step – frequent pattern mining. The FP-tree algorithms [20] proposed by Han et al. are typical approaches to generating frequent patterns.

In recent years, many researchers have extensively investigated association rule mining methods using FP trees. For example, combining the FP-tree algorithms with the theory of maximum clique, Chen et al. studied a method of generating frequent 2-item sets with adjacency matrix [22]. Pei et al. proposed the CLOSET

algorithms, in which FP trees were used to denote pattern support sets [25]. Aiming at the problem of unavailable minimum support threshold given by users, Wang et al. proposed a frequent closed pattern mining TFP algorithm in 2005 [26]. Zaki et al. investigated a closed pattern mining CHARM algorithm using transaction-marking sets of vertical format to declare pattern support sets[27]. Pei et al. studied a method of mining frequent closed partial order from strings [21] by introducing the corresponding pruning technology. However, Pei's method had a certain limitation, because it was only adequate for mining string databases. Gudes et al. developed an algorithm of discovering frequent graph patterns using disjoint paths [28]. Gudes's method was implemented based on the Apriori algorithms, thereby unavoidably scanning databases many times. *Song et al.* designed a transaction-mapping algorithm to discover frequent item sets [29]. Lucchese et al. proposed an algorithm of partition visiting data space along with the corresponding pruning method [30]. Lucchese's approaches do not need to save all pattern sets into the main memory, thereby reducing the storage demand. Unfortunately, the efficiency of Lucchese's algorithm was low when the size of datasets was huge.

Using users' long accumulated experience, interest, and the deep understanding of specific domains of data as apriori information (a.k.a., background knowledge) to guide the building of FP trees and to generate frequent patterns, one can improve the pertinence of association rules' mining, reduce the complexity of FP tree, and effectively solve the data storage bottleneck of the FP-tree algorithms.

Let G be celestial spectra knowledge, we can define a constrained frequent pattern tree with respect to knowledge G as follows:

Definition 7. Given background knowledge G, a FP tree, and a frequent pattern P described using the path from the root node of the tree to a leaf node, if P can satisfy G represented as $G(P) = True$, then FP-Tree is a constrained frequent pattern tree or CFP tree for short.

Let us consider celestial spectra database D, the minimum support threshold $\sigma_{\min}$, and celestial spectra knowledge G. Because any constrained frequent pattern P is satisfied with $G(P) = True$, only a FP tree built from any piece of celestial spectral data T in D satisfying G can contain interesting constrained frequent patterns of users' preferences. Hence, the CFP tree of celestial spectra data can be built through traversing database D twice according to the following algorithm.

CFP-Construct: An algorithm to build a CFP tree:

1. Scan through celestial spectra database D for the first time, gathering frequent length-1 patterns of sets and their supports, sort frequent length-1 pattern in descending order of the supports and generate frequent item table L;
2. create the root node of the celestial spectra data and mark with "null";
3. for each transaction T in D, if there is no interesting pattern in T (note: an interesting pattern is the one about which users are concerned), then skip to the next transaction. Otherwise, turn to step 4;
4. sort frequent items of T in order of L and generate a new list of frequent items named T', then update the CFP tree according to the followings three steps:
   (4.1) search for a path that is the longest prefix matching with T' in the CFP tree;
   (4.2) the count of the node that are in the matching path is increased by 1;
   (4.3) search for the mismatching suffix in T', and determine the node to which the last frequent item in the longest matching prefix is corresponding as the root node, then create child nodes successively in the CFP tree and set the count value to 1.

We show in Theorem 3 below that given a constraint frequent pattern tree or CFP tree, each frequent pattern P extracted from the CFP tree necessarily satisfies constrained frequent patterns,

Theorem 3. *Let D be a celestial spectra database, σ_{min} be the minimum support threshold, G be celestial spectra knowledge, and P be a constrained frequent pattern. If a created FP tree is a CFP tree, then any frequent pattern P extracted from the CFP tree necessarily satisfies constrained frequent patterns, i.e., $G(P) = true$.*

Proof. We can prove the correctness of Theorem 3 by two steps. First, we need to prove that a pattern P extracted from the CFP tree is a frequent pattern. Second, we have to show that pattern P satisfies the constrained condition in celestial spectra knowledge G.

According to the above CFP-Construct algorithm, each path of the CFP tree necessarily satisfies the constrained condition (i.e., G). Thus, frequent pattern P extracted from the CFP tree necessarily satisfies G. Recall that a CFP tree conforms to the building process of FP trees; a CFP tree is a sub-tree of FP tree. Based on Lemma 1, we can show that frequent pattern P extracted from the CFP tree necessarily satisfies the frequent patterns. □

Next, we prove in Theorem 4 that if P satisfies constrained frequent patterns in e G (i.e., $G(P) = true$), then P is a frequent pattern in a CFP tree built by the CFP-Construct algorithm.

Theorem 4. *Let D be a celestial spectra database, σ_{min} be the minimum support threshold, and G be celestial spectra knowledge. Let CFP be a constrained PF tree, P be a frequent pattern. If P satisfies constrained frequent patterns (i.e., $G(P) = true$), then P is necessarily a frequent pattern of the CFP tree.*

Proof. Let us consider a frequent pattern $P(s_1, s_2; \ldots, s_i, \ldots, s_n)$ in which users are interested. We have $s \subseteq G$; for any transaction W, we have $P \subseteq W$. Based on the CFP-Construct algorithm constructing CFP trees, we can prove the following facts:

(1) CFP-Construct scans through transaction database once, and gather frequent length-1 pattern sets and their supports. Suppose the support of s sups $\geqslant \sigma_{\min}$, sort frequent length-1 pattern in the descending order of the supports, and generate frequent item table L;
(2) CFP-Construct creates the root node of the CFP tree and mark with "null". For the transaction W in database D, CFP-Construct selects frequent items $(s_1, s_2, \ldots, s_i, \ldots, s_n)$ in W, where s is a pattern that users are interested in. The sequence is sorted in the order of L;
(3) CFP-Construct judges s_1: if there exists child node N in tree T, where *N.item-name* equals to *p.item-name*, then the counter of N is increased by 1; otherwise CFP-Construct creates a new node N whose counter is set to 1, then the node is linked to its parent node T and the other nodes with the same item-name through the structure of linker. Note that step (3) is repeatedly applied to $s_2, \ldots, s_i, \ldots, s_n$.

The aforementioned three facts show that all the transactions containing P are transformed to the paths of the CFP tree; therefore, P must be a frequent pattern in the CFP tree. This concludes the proof of Theorem 4. □

Now we are in a position to prove that a CFP tree is a subset of a FP tree.

Theorem 5. *A CFP tree is a subset of a FP tree.*

Proof. Let D be a celestial spectra database, $\sigma_{\min}$ be the minimum support threshold, and G be celestial spectra knowledge. Applying knowledge G, one can divide database D into two disjoint sets D_1 and D_2 (i.e., $D = D_1 \cup D_2$), where records in set D_1 satisfy the background knowledge G, and records in set D_2 do not satisfy the background knowledge G. Thus, for any record T, if $T \in D_1$, then $G(T)$ = True. Otherwise $T \in D_2$, then $G(T)$ = False. Based on Definitions 5 and 7, a FP tree is built from records in D, and a CFP tree is built from records in D_1. As such, a CFP tree is subset of its FP tree if and only if $D_2 = \varnothing$, $D = D_1$, and FP tree = CFP tree. □

Theorem 6 below shows the monotonicity constrained frequent-pattern trees.

Theorem 6. *Let D be a celestial spectra database, G_1 and G_2 be celestial spectra knowledge, $G_1 \subseteq G_2$. If T_1 and T_2 are two CFP trees constructed respectively from G_1 and G_2, then $T_1 \supseteq T_2$. (the monotonicity of constrained FP trees)*

Proof. Suppose transactions $k_1, k_2, \ldots, k_n$ in D contain knowledge G_1, and transactions $s_1, s_2, \ldots, s_m$ contain knowledge G_2. Because $G_1 \subseteq G_2$, we have $s_1, s_2, \ldots, s_m \subseteq k_1, k_2, \ldots, k_n$ and $m \leqslant n$. Let us suppose $s_1, s_2, \ldots, s_m = k_1, k_2, \ldots, k_m$, then transactions $k_{m+1}, \ldots, k_n$ only contain knowledge G_1, excluding G_2. According to the construction process of CFP tree (see the CFP-Construct algorithm), we show that T_2 is firstly built and the path of T_2 is made up of transactions $s_1, s_2, \ldots, s_m$, whose width is not greater than m. Then, T_1 is built and the path of T_1 is comprised of transactions $k_1, k_2, \ldots, k_m$ and $k_{m+1}, \ldots, k_n$,which are denoted by T_1' and T_1''. Because $s_1,s_2, \ldots ,s_m = k_1, k_2, \ldots, k_m$, we prove that $T_1' = T_2$. When m equals to n, we have $k_{m+1}, \ldots, k_n = \varnothing, T_1'' = \varnothing$ and $T_1 = T_2$; when m is smaller than $n, k_{m+1}, \ldots, k_n \neq \varnothing$ and $T_1'' \neq q\varnothing$. Thus, T_1 contains some paths that are not included in T_2, meaning $T_1 \supset T_2$, Therefore, we prove that $T_1 \supseteq T_2$. □

Corollaries 1 and 2 can be obtained using the above theorem.

Corollary 1. *Let D be a celestial spectra database, G_1 and G_2 be celestial spectra knowledge. Suppose T_1 and T_2 are two CFP trees constructed respectively from G_1 and G_2. If the two CFP trees are identical (i.e., $T_1 = T_2$), then G_1 and G_2 are the same, i.e., $G_1 = G_2$.*

Corollary 1 states that if two constrained frequent-pattern tree are the same, then the knowledge sets used to construct these two trees must be identical.

Corollary 2. *Let D be a celestial spectra database, G_1 be celestial spectra knowledge. Suppose T_{CFP} is a CFP tree constructed using knowledge G_1 and T_{FP} is a FP tree in database D. If knowledge G_1 is an empty set (i.e., $G_1 = \varnothing$), then constrained frequent-pattern tree T_{CFP} equals to frequent pattern tree T_{FP} (i.e., $T_{CFP} = T_{FP}$).*

Corollary 2 shows that a CFP tree is degraded to a FP tree if celestial spectra knowledge is null.

Applying background knowledge G, database D can be partitioned into two disjoint sets D1 and D2, where transactions in set D1 satisfy the background knowledge G, and transactions in set D2 do not satisfy the background knowledge G. The theorem below shows that a constrained frequent- pattern tree built from a celestial spectra database D is the same as a frequent-pattern tree constructed from the sub-database D1 of D.

Theorem 7. *Let us consider a constrained frequent-pattern tree denoted as CFP-tree. Let D be a celestial spectra database, σ_{min} be the minimum support threshold, and G be celestial spectra knowledge. Applying knowledge G, one can divide database D into two disjoint sets, D_1 and D_2 (i.e., $D = D_1 \cup D_2$), where transactions in set D_1 satisfy the background knowledge G (i.e., $\forall T \in D_1$: $G(T)$ = True), and transactions in set D_2 do not satisfy the background knowledge G (i.e., $\forall T \in D_2$: $G(T)$ = False). If FP-Tree$_1$ is a frequent-pattern tree constructed from sub-database D_1 using the traditional method, then CFP-tree and FP-Tree$_1$ is the same tree.*

Proof. Suppose FP-Tree$_2$ is a frequent-pattern tree built from sub-database D_2 by applying the traditional FP-tree construction method. Since $\forall T \in D_1$: $G(T)$ = True, we have $G(P)$ = True for any frequent pattern P described in the path from the root node to the leaf node of FP-tree$_1$. Moreover, because $\forall T \in D_2$: $G(T)$ = False, we have $G(P)$ = False for any frequent pattern P described in the path from the root node to the leaf node of FP-Tree$_2$, as $D = D_1 \cup D_2$, FP-tree$_1$ can represent all the constrained frequent patterns of FP-tree. Thus, we prove that CFP-Tree is the same tree as FP-Tree$_1$. □

4.2. Frequent pattern extraction

Traversing a CFP tree of celestial spectra data, one can extract constrained frequent patterns of the celestial spectra data. To efficiently traverse a CFP tree, we create an item-head table in which each item has a node link pointing to its location in the CFP tree. The extraction process starts by generating frequent length-1 patterns, which can be constructed using condition pattern bases. Note that a condition pattern base is a sub-database composed of the prefix path sets appearing in both the CFP tree and the suffix pattern. Next, a condition CFP tree is built. Association rule mining process is performed recursively on the CFP tree. Finally, the pattern growth is realized through the combination of frequent pattern generated by the suffix and the condition CFP tree [20].

Suppose $\psi_{\min}$ is the minimum confidence threshold, and L is the constrained frequent pattern sets extracted from a CFP tree. For any frequent pattern P in the pattern set L (i.e., $P \in L$), we have $P = P_1 \cap P_2$, where P_1 is the nonempty sub-pattern of the celestial spectra data characteristics, whereas P_2 is the nonempty sub-pattern of physicochemical properties, and $\sigma = \sigma(P/DB)$. If confidence ψ is larger than or equal to the minimum confidence threshold (i.e., $\psi = \sigma(P_1 \cap P_2/DB)/\sigma(P_1/DB) \geqslant \psi_{\min}$), then association rule "$P_1 \Rightarrow P_2\,(\sigma,\psi)$" is created, where σ and ψ are the important degree and confidence degree describing the association rule.

5. Algorithm and I/O performance

5.1. The interrelation analysis algorithm

After analyzing the process of constructing constrained frequent pattern trees, we present in Fig. 2 the pseudocode for the CFP-Construct algorithm used to build CFP trees of celestial spectra data.

The goal of CFP-Construct algorithm is to create the root node denoted as T of a constrained frequent pattern tree for celestial spectra data. Initially, the root node T is marked as "null". Before processing transactions in database D, the algorithm (1) obtains celestial spectra frequent item set F and (2) calculate the support of each frequent item. This step (see Step 1 in Fig. 2) is performed by scanning the entire celestial spectra database D. Then, the algorithm generates a new list L by sorting frequent items in F, and is sorted in the decreasing order of their supports (see Step 2 in Fig. 2). Next, the CFP-Construct algorithm converts the celestial spectra knowledge G into the celestial spectra pattern set S (see Step 3 in Fig. 2), which is sorted in the decreasing order of the support of frequent length-1 pattern (see Steps 4 and 5 in Fig. 2).

Algorithm CFP-Construct for celestial spectra data:
Input: celestial spectra database D, a minimum support threshold σ_{min} and celestial spectra knowledge $G(g_1, g_2,... g_k)$.
Output: a CFP tree of celestial spectra data
1: scan the entire celestial spectra database D, obtaining set F of celestial spectra frequent items coupled with the support of each frequent item;
2: sort items in F in the decreasing order of their supports; the new list is L;
3: convert the celestial spectra knowledge G into the celestial spectra pattern set S;
4: sort set S in the decreasing order of the support of frequent length-1 pattern;
5: $\forall s \in S$: $S=S-\{s\}$,
if s contains only frequent length-1 patterns **then**
6: sort s in the decreasing order of the supports;
7: $S = S \cup \{s\}$; **endif**
//Trans is a transaction in database D
8: **while** (Trans != ∅) **do**
9: Sort spectra data frequent items in Trans based on the order of L, obtaining new list (L_1, L_2,..., L_n);
10: **if** (L_1, L_2,..., L_n) contain no frequent length-1 pattern of S) **then** Continue;
11: Insert_tree(L_i, T);
12: **end while**
13: **end CFP-Construct.**

Fig. 2. Algorithm CFP-construct: construct constrained frequent pattern trees for celestial spectra data.

Now, the algorithm starts processing each transaction *Trans* in the celestial spectra database with the following operations:

- the celestial spectra frequent items in *Trans* are chosen and sorted using the order of L;
- the ordered frequent item lists are created for *Trans* as $[p|P]$, where p is the first item in the list, and P the remainder of the list;
- invoke function insert_tree ($[p|P]$,T) to insert the list $[p|P]$ into the constrained frequent pattern tree T.

Inserting a list of patterns in a constrained frequent pattern tree is a critical process in the CFP-Construct algorithm. This insertion operation (see Step 11 in Fig. 2) is implemented by function insert_tree ($[p|P]$,T) outlined in Fig. 3.

If there is a child node N in T, and if N.item-name is the same as p.item-name, then the count value of N is increased by 1 (see Steps 1 and 2 in Fig. 3). Otherwise, the insert_tree function creates a new node N, whose count value is set to1 (see Step 5 in Fig. 3), and the parent node of the new node is set to T (see Step 7 in Fig. 3). In addition to creating a new node, the node_link and item_name items of the new node are set to N and P, respectively (see Steps 6 and 8 in Fig. 3). If P is not an empty set, then insert_tree (P,N) will be recursively called (see Steps 11 in Fig. 3).

The extraction of celestial spectra data frequent patterns is implemented by calling the function FP_growth (details on the FP_grwoth function can be found in [20]).

Insert_tree(P,T)
1: **if** (there is a child node N in CFP tree T && N.item_name == p) **then** //* p is first element in P
2: N.item_count++;
3: **else**
4: create new node N;
5: N.item_count=1;
6: N.item_name= p;
7: N.node_parent=T;
8: N.node_link=N;
9: **end if**
10: P=P-{ p };
11: if (P != ∅) then insert_tree(P, N);
12: **end Insert_tree.**

Fig. 3. Function Insert_tree (P,T) inserts pattern P into constrained frequent pattern tree T. The Insert_tree function is invoked by Step 11 in the CFP-construct algorithm described in Fig. 2.

5.2. I/O performance

The efficiency of the data-mining tool can be measured in terms of the mining efficiency of association rules, which is significantly affected by the performance of generating frequent patterns. Recall that frequent-pattern generating algorithms fall into two camps: Apriori [19] and Frequent-Pattern tree (i.e., FP-tree) [20–22]. The Apriori algorithms, relying on recursive statistics, generate frequent item sets by pruning. One limitation of the Apriori algorithms lies in the large number of generated candidate items, which forces a data mining tool to traverse a database excessive number of times in order to produce frequent patterns. Thus, I/O overhead of traversing the database is very high for the Apriori algorithms, making it time consuming to deal with massive amount of high-dimensional data sets.

The main benefits of FP-tree algorithms are:

- There is no need to generate candidate item sets, because frequent patterns are obtained by accessing a FP tree, which can be quickly constructed.
- Any database should only be traversed twice. In the first-round traversing, length-1 frequent item sets are yielded. In the second traversing, a FP tree is constructed. In doing so, I/O accessing time on the database is substantially reduced.

The downside of the FP-tree algorithms is that large main memory capacity is required for FP trees. Recent studies focus on either constructing FP trees [20] or reducing computation time of existing FP-tree construction algorithms [21,22,25–30]. However, less attention has been paid to analyzing and improving I/O performance of FP-tree construction algorithms.

The interrelation analysis of massive amount of celestial spectra data sets leads to an excessive number of I/O operations issued to access spectra databases in the building process of FP trees from huge amounts of high-dimensional data. Therefore, boosting the I/O performance of FP-tree construction algorithms is a challenging issue. The I/O performance analysis of the construction process for frequent pattern trees is the first step toward improving the overall performance of FP-tree construction algorithms.

To measure the I/O performance of our FP-tree construction algorithm discussed in Section 5, we introduce a metric called I/O factor denoted as γ, which is computed as a ratio between I/O processing time and the total executing time of the FP-tree construction algorithm. Thus, I/O factor γ is defined as:

$$\gamma = \frac{T_{IO}}{T_{CFP-Build}} = \frac{T_{IO}}{T_{IO} + T_{CPU}}, \quad (5)$$

whereas T_{IO} is the I/O time spent in access the celestial spectra database, $T_{CFP-Construct}$ is the execution time of the FP-tree construction algorithm. Note that $T_{CFP-Construct}$ is the sum of the CPU time (i.e., T_{CPU}) and I/O time (i.e., T_{IO}) of the FP-tree construction algorithm.

6. Performance evaluation

6.1. Experimental setup

We implement a data-mining tool for the interrelation analysis of celestial spectra data sets. The CFP-Construct algorithm

described in Sections 4 and 5 is incorporated in our interrelation analysis module for celestial spectra data.

The experiments are performed on a PC with Intel Pentium IV 3.0 GHz processor and 512 MB main memory, and the operating system is Windows XP professional. We have fully implemented our interrelation analysis module on top of Oracle 9i – a representative commercial DBMS. Our celestial-spectra data-mining tool and the interrelation analysis module, in which the CFP-Construct algorithm is implemented, are developed with Visual C++ 6.0.

The data sets used in our experiments is the latest and largest celestial-spectra data sets provided by the National Observatory of China. To evaluate data size on system performance, we synthetically partition the largest data set into six data subsets. The sizes of these six data subsets are 50 MB, 100 MB, 200 MB, 300 MB, 400 MB, and 500 MB, respectively. For each spectra data set, the discretized values of 44 (forty-four) characteristic lines of the spectra data, are selected as attribute values of the celestial spectra data (see Table 1 for the parameters used in the discretization process of the star spectra data). In other words, each spectra data record has 44 attributes, each of which contains 13 values.

I/O processing time measured in our experiments is defined as the sum of (1) the time spent loading data from the database and (2) the time spent accessing constrained frequent pattern trees. In the CFP-Construct algorithm, the first step and the fourth step (see Fig. 2) comprise I/O operations of loading data; the seventh step comprises I/O operations of accessing constrained frequent pattern trees (i.e., to insert new nodes and update nodes of FP trees).

In the CFP-Construct algorithm, CPU time is mainly spent on processing the *Insert_tree*$(p|P,T)$ function. The first step of algorithms is to sort frequent item sets in descending order of support; Steps 2, 3, 5, and 6 aim to compute support S_i of the background knowledge. We apply the MFC standard function long__cdecl time (long $*$) in Visual C++ 6.0 to accumulate I/O and CPU processing time spent in the algorithm. The total execution time of the interrelation analysis module is derived as the sum of CPU time and I/O time.

6.2. I/O performance analysis

We vary the size of the celestial spectra data from 100 MB to 500 MB with an increment of 100 MB. A small data size of 50 MB is also tested. Fig. 4 shows the I/O factor γ (see Eq. (5)) of the interrelation analysis module when the input data size increases from 50 to 500 MB. Table 2 below details I/O time, CPU time, and the number of created nodes for six datasets with different data sizes. We execute the same interrelation analysis module multiple times with the six different datasets. The default minimum support threshold is set to 3%. Four constrained conditions evaluated in our experiments include: cross patterns, single constrained patterns, union patterns and null constrained patterns.

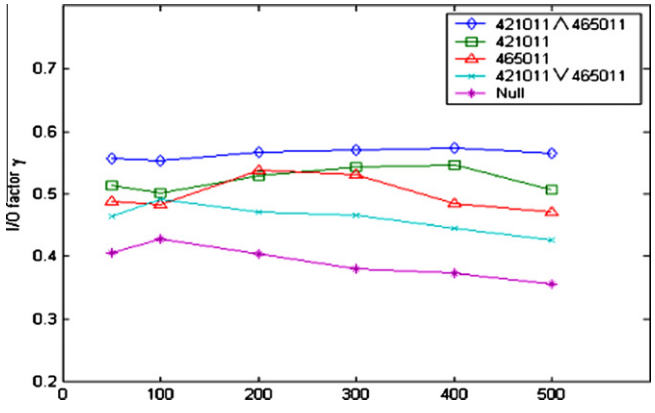

Fig. 4. I/O factor with the different data size measured in terms of MB.

Given the same celestial spectra data set, we study the impact of constraint degree on the I/O processing time of the interrelation analysis module. Fig. 4 illustrates that when constrained condition becomes weaker, I/O factor slightly increases. The main reason for such an I/O time increase is that building constrained FP trees contains two parts: (1) loading data from the database and (2) accessing constrained FP trees. During the course of building constrained FP trees, the database is accessed by the first step of the CFP-Construct algorithm and frequent length-1 pattern is generated. Since this operation is independent of the constrained conditions, I/O time spent loading data from database is independent of the constrained conditions.

With the constraint degree weakened, an increasing number of records can satisfy the weak constraints. Thus, more nodes of a FP tree will be created, making the size of the constrained FP tree increase. Therefore, weakening the constraint degree leads to an increased I/O time spent accessing constrained frequent pattern trees.

Keeping the constrained condition unchanged, we observe from Table 2 that when the size of the input celestial spectra dataset increases, the I/O time becomes significantly longer. We contribute this I/O performance trend to the fact that increasing data size leads to an increased time scanning the celestial spectra database; therefore, I/O time is proportional to the size of processed datasets. With the increasing data size, a growing number of records can satisfy constraints specified by users, meaning that an increased number of nodes will be created to enlarge the size of the frequent pattern trees. Given a fixed record set, a weak constrained degree leads to an increased CPU processing time spent on (1) sorting attributes of each record that meet the specified constraints condition and (2) inserting these attributes into the constrained frequent pattern tree.

Note that the CPU time of the interrelation analysis module is greatly affected by the number of nodes in a frequent pattern tree. In other words, a large number of nodes in a frequent pattern tree can give rise to a huge number of comparison/insertion operations, which increases CPU processing time.

Under the same constraints, the CPU time of building a constrained frequent pattern tree largely depends on the node count of frequent patterns. When the input record set becomes large, the number of nodes of frequent patterns is greatly increased, thereby increasing the number of operations for inserting nodes into the constrained frequent pattern tree. Thus, the CPU time of processing the large record set is increased sharply.

Table 3 shows the impact of minimum support threshold $\sigma_{\min}$ on I/O performance of the interrelation analysis module. For comparison purpose, we also show the CPU time in Table 3. In this set of experiments, we fix the record set to the number of 209,999 pieces and the data size to 300 MB while setting the minimum support threshold $\sigma_{\min}$ to 8%, 5%, 3%, 1%, and 0.5%, respectively. Again, five constraint conditions studied are cross pattern, single constrained pattern, union pattern, and null constraint pattern.

Fig. 5 shows the impact of the minimum support threshold on I/O factor γ. When the constraint condition is fixed, varying the minimum support threshold does not affect time spent loading data items from the celestial spectra database (see Steps 1 and 3 in the CFP-Construct algorithm), because accessing the database in Steps 1 and 3 of the algorithm is independent of the minimum support threshold.

Fig. 5 illustrates that regardless of constraint conditions, a large minimum support threshold gives rise to a high I/O factor γ. For example, when the constraint is set to 421011, the I/O factor for the minimum support threshold of 8% is 11% higher than that

Table 2
Impact of data size on I/O performance (minimum support threshold σ_{min} = 3%).

Record set		Constrained conditions				
		421,011 ∧ 465,011	421,011	465,011	421,011 ∨ 465,011	Null
(50 M) 34,117 pieces of records	I/O(s)	197	204	214	216	255
	CPU(s)	157	193	224	250	373
	The number of nodes	91,440	143,770	192,599	244,577	437,424
(100 M) 72,999 pieces of records	I/O(s)	444	448	523	578	590
	CPU(s)	359	444	560	597	789
	The number of nodes	197,500	303,579	407,179	512,982	904,415
(200 M) 139,999 pieces of records	I/O(s)	852	880	940	961	1165
	CPU(s)	650	784	811	1078	1711
	The number of nodes	349,754	536,349	730,622	916,940	1,622,165
(300 M) 209,999 pieces of records	I/O(s)	1346	1371	1460	1518	1662
	CPU(s)	1004	1155	1292	1735	2708
	The number of nodes	500,134	770,811	1,058,029	1,328,424	2,371,635
(400 M) 272,999 pieces of records	I/O(s)	1746	1784	1831	1903	2104
	CPU(s)	1318	1484	1941	2372	3541
	The number of nodes	630,661	975,400	1,341,779	1,686,265	3,020,189
(500 M) 334,668 pieces of records	I/O(s)	2007	2091	2262	2528	2557
	CPU(s)	1549	2030	2536	3400	4604
	The number of nodes	755,599	1,168,750	1,621,000	2,033,912	3,652,095

Table 3
Impact of support threshold on I/O performance (recorder Set = 209,999 pieces, data size = 300 M).

Support threshold σ_{min}		Constrained conditions minimum				
		421,011 ∧ 465,011	421,011	465,011	421,011 ∨ 465,011	Null
8%	I/O(s)	1164	1224	1302	1416	1576
	CPU(s)	868	832	956	1263	2166
	The number of nodes	397,997	618,954	822,889	1,043,564	1,900,831
5%	I/O(s)	1203	1251	1324	1436	1582
	CPU(s)	932	979	1265	1640	2580
	The number of nodes	457,078	708,083	966,477	1,217,200	2,193,503
3%	I/O(s)	1346	1371	1460	1518	1662
	CPU(s)	1004	1155	1292	1735	2708
	The number of nodes	500,134	770,811	1,058,029	1,328,424	2,371,635
1%	I/O(s)	1370	1435	1586	1684	1743
	CPU(s)	1135	1220	1456	1767	3462
	The number of nodes	523,604	809,502	1,113,491	1,399,107	2,481,794
0.5%	I/O(s)	1480	1445	1611	1689	1769
	CPU(s)	1253	1282	1650	1826	3667
	The number of nodes	525,478	812,093	1,119,813	1,406,146	2,493,244

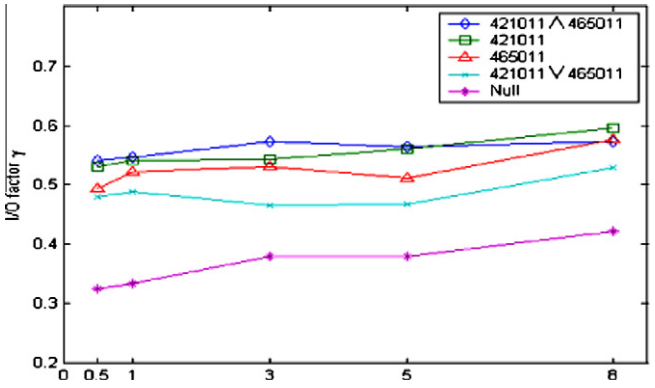

Fig. 5. I/O factor with the different minimum support (%).

for the minimum support threshold of 0.5%. High minimum support threshold noticeably affect I/O factor, and the reason is twofold. First, minimum support thresholds have impacts on both CPU and I/O time of the interrelation analysis. Second, compared to I/O time, CPU time is more sensitive to the minimum support threshold. For example, let us consider the case where the constraint is set to 421001. Table 3 shows that when we increase the minimum support threshold from 0.5% to 8%, CPU time decreases from 1282 s to 832 s (i.e., reduced by 35.1%) whereas I/O time is reduced only from 1445 s to 1224 s (i.e., only reduced by 15.3%). The evidence shows that a large number of nodes in a frequent pattern tree results in a long CPU processing time due to the fact that function *insert_tree*($p|P,T$) needs to be recursively invoked when a new node is created. Therefore, compared with the minimum support threshold of 0.5%, minimum support threshold of 8% has a high I/O factor.

Table 3 shows that a small minimum support threshold leads to a large number of nodes created in frequent pattern trees (i.e., large-scale FP trees). Therefore, time spent accessing the large constrained frequent pattern trees is increased. For example, if the constraint is set to 465011, then the numbers of nodes created in the frequent pattern tree are 618954 and 812,093 when the minimum support threshold is 8% and 0.5%, respectively. Thus, we conclude that under the same constraint condition, I/O processing time in the interrelation analysis increases with the decreasing of the minimum support threshold.

When the minimum support threshold is fixed, constraint conditions have a great impact on the number of nodes created in frequent pattern trees. For example, let us consider a case where the minimum support threshold is set to 3%. Table 3 shows that the number of nodes in the created frequent pattern tree is 500,134 and 2,371,635 for a strong constraint (i.e., $421001 \wedge 465011$) and a weak constraint (i.e., NULL).

In the process of building constrained frequent pattern trees, a significant portion of the time is spent on I/O processing, which is greatly affected by the constraint conditions, the minimum support threshold, and the size of celestial spectra dataset. The experimental results indicate that improving the I/O performance of the CFP-Construct algorithm is a critical step towards boosting the performance of the interrelation analysis module for a huge amount of celestial spectra data. With the increasing size of the celestial spectra data, the I/O processing time in building constrained frequent pattern trees almost linearly grows. Please note that in our experiments, we only considered I/O time spent on accessing the database in a single machine. In a client/server computing platform where multiple clients concurrently access the database, the I/O performance is expected to reduce sharply. We can improve the I/O performance for the interrelation analysis of celestial spectra data using two effective approaches.

- Parallel disks allow multiple disks to load celestial data simultaneously by the virtue of parallel I/O.
- Solid-state disks can be used to reduce I/O accessing times for constrained frequent pattern trees.

6.3. A celestial-spectra data mining tool

Using the SDSS star spectra data provided by the National Observatory of China, we select 200 pieces of wavelength and five physicochemical properties as attributes to represent star spectra data. All the experimental datasets are preprocessed and discretized in accordance to the method described in Section 3.1. The preprocessing module and the interrelation analysis module are integrated into our celestial-spectra data mining tool. Tables 4 and 5 show the user interface of the data-mining tool.

Table 4 shows the user interface used to display mining results in terms of constrained frequent patterns. In this set of experiments, the background knowledge are the characteristics of spectrum patterns containing wide peak-intensity flux at the 3870 wavelength, wide peak-weaker flux at the 3950 wavelength, wide peak-intensity flux at the 4010 wavelength or narrow peak-intensity flux at the 5510 wavelength, and the minimum support threshold $\sigma_{\min} = 1\%$. A corresponding table of discretized celestial spectra data is selected, and a constrained frequent pattern tree is created using the CFP-Construct algorithm. The frequent patterns satisfying the constraint conditions are extracted by traversing the constrained frequent pattern tree. In Table 4, the last frequent pattern (i.e., 3870_1, 3950_1, 4010_1, 4330_3, 4450_1, 4850_3, 4990_5 and 5850_1 (1.000%)) indicates that the discretization values at the spectrum wavelength of 3870, 3950, 4010, 4450, and 5850 equal to 1; the discretization values at the spectrum wavelength of 4330 and 4850 equal to 3; the discretization value at the spectrum wavelength of 4990 is 5; and the support of this frequent pattern in the discretization table s8000ls is 1%.

Table 4
Constrained frequent patterns generated by the celestial-spectra data mining tool.

Constrained frequent pattern (item sets)	Support (%)
3870_1, 3950_1, 4010_1, 4090_3, 4450_1, 4850_3, 6550_3	1.250
3870_1, 3950_1, 4010_1, 4330_3, 4450_1, 4850_3, 4990_3	1.200
3870_1, 3950_1, 4010_1, 4330_3, 4450_1, 4850_3, 6550_3	1.550
3870_1, 3950_1, 4090_3, 4330_3, 4450_1, 4990_5, 5850_1	2.400
3870_1, 3950_1, 4090_3, 4330_3, 4690_1, 4850_3, 5850_1	1.400
3870_1, 3950_1, 4090_3, 4330_3, 4850_1, 4990_3, 5850_1	2.150
3870_1, 3950_1, 4090_3, 4330_3, 4850_1, 5850_5, 6550_B	2.200
3870_1, 3950_1, 4330_3, 4450_1, 4850_3, 5850_1, 6550_3	1.050
3870_1, 4010_1, 4090_3, 4330_3, 4450_1, 4850_3, 6550_3	1.450
3870_1, 3950_1, 4010_1, 4330_3, 4450_1, 4850_3, 4990_5, 5850_1	1.000
...	...

Table 5
User interface to display association rules generated by the celestial-spectra data mining tool.

Association rule	Support (%)	Confidence (%)
3970_weak wide, 4010_strong-wide ⇒ temperature_3, chemistry_2, others_1	1.900	73.07
4010_strong-wide, 4050_strong-wide ⇒ temperature_3, others_1	2.100	100.0
4010_strong-wide, 4050_strong-wide ⇒ temperature_3, microturbulence_1others_1	2.100	100.0
4010_strong-wide, 5190_weak-narrow ⇒ temperature_3, others_1	1.400	70.00
4010_strong-wide, 5190_weak-narrow ⇒ temperature_1, microturbulence_1, others_1	1.400	70.00
4870_weak-median, 5510_weak-narrow ⇒ temperature_1, others_1	2.100	100.0
4870_weak-median, 5510_weak-narrow ⇒ temperature_1, chemistry_2, others_1	2.100	100.0
4870_weak-median, 5510_weak-narrow ⇒ temperature_1, microturbulence_1, others_1	2.100	100.0
3910_weak wide, 3950_median-weak-wide, 4390_weak-median ⇒ temperature_5, microturbulence_2, others_2	2.000	83.33
3870_strong-wide, 4090_median-weak-wide, 4850_median-weak-wide, 5250_strong-wide ⇒ temperature_D, chemistry_ 2, microturbulence_ 2, luminosity_2	1.800	78.26
...	...	...

Table 5 shows the results of the interrelation analysis of celestial spectra data. In this experiment, the minimum support threshold $\sigma_{\min}$ is set to 1% and the minimum confidence threshold $\psi_{\min}$ is set to 70%. Interesting conclusions can be drawn from the association rules.

For example, the last association rule in Table 5 is 3870_strong wide, 4090_weaker wide, 4850_weaker wide, 5250_strong wide ⇒ temperature_D, chemical_2, microturbulence_2, luminosity_2 (1.800%, 78.26%). This association rule can be explained as follows. (1) There exists very strong and very wide peaks at the wavelength of 3870; (2) there exists weaker and very wide peaks at the wavelength of 4090; (3) the peak at the wavelength of 4850 is weaker and very wide; (4) if the peak at the wavelength of 5250 is very strong and very wide, then the temperature range of this spectrum is from 7500 to 8300, the chemical abundance range is from −3 to −0.5, the microturbulence value is 2, and the luminosity range is from 0 to 1.1. The support of the association rule (i.e., the important degree of this rule) is 1.8%, and the confidence (i.e., the confidence degree of this rule) is 78.26%. Comparing the rules generated by our celestial-spectra data-mining tool with the relations among the wave characteristics and physicochemical properties acquired by the experience of spectra data, one can conclude that it is basically similar to the characteristics of type A. Thus, it is practical and valuable to make use of our interrelation analysis module to extract correlations among the characteristics and physicochemical properties of the celestial spectra data.

7. Conclusion

LAMOST can acquire huge amount of celestial spectra data with the highest rate. To support the interrelation analysis of celestial spectra data, we not only study constrained frequent pattern trees, but also develop an algorithm called CFP-Construct to build constrained frequent pattern trees efficiently using the first-order predicate logic to represent background knowledge. We implement a data-mining tool in which the CFP-Construct algorithm was incorporated in an interrelation analysis module for celestial spectra data. We evaluated the performance of the data-mining tool, focusing on the I/O performance of the interrelation analysis module. The experimental results on the SDSS star spectra data show that it is practical to rely on the interrelation analysis method to discover correlations among the celestial spectra data characteristics and the physicochemical properties. One can leverage our data-mining tool as an effective new approach to exploring the unknown laws of celestial bodies.

The interrelation analysis method presented in this paper not only can be applied to process celestial spectra data, but also can be extended to deal with interrelation analysis in other application domains. Thus, a particular interesting topic that we plan to pursue in the future is to investigate how to apply our analysis approach to other domains. We will show in the future study that if users provide good understandings of their applications as well as data-mining preferences in terms of the predicate formula for domain data, our scheme can be employed to perform correlation analysis of data in a wide variety of application domains.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of PR China (Grant No. 61073145), the Natural Science Foundation of Shanxi Province, PR China (Grant No. 2010011021-2) and the Returning Students and Scholars Research Project of Shanxi Province, PR China (Grant No. 2009-77). Xiao Qin's work was made possible thanks to NSF awards CCF-0845257 (CAREER), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (Cyber-Trust), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS), and an Intel gift (Number 2005-04-070) as well as an Auburn University startup grant.

References

[1] Richard Stone, China's LAMOST observatory prepares for the ultimate test, Science 320 (5872) (2008) 34–35.
[2] Qin Dongmei, Studies on Automated Spectral Recognition of Celestial Object, [Ph.D. Dissertation],Institute of Automation, Chinese academy of science, May, 2003.
[3] P. Cheeseman, J. Sutz, Bayesian classification (Autoclass): theory and results, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, The AAAI Press, Menlo Park, 1995.
[4] R.K. Gulati, R. Gupta, N.K. Rao, A comparison of synthetic and observed spectra for G-K dwarfs using artificial neural networks, Astronomy and Astrophysics 322 (1997) 933–937.
[5] B. Weaver, Torres-Dodgen, Accurate two-dimensional classification of stellar spectra with artificial neural networks, Astrophysical Journal 487 (1997) 847–857.
[6] H.P. Singh, R.K. Gulati, R. Gupta, Stellar spectral classification using principal component analysis and artificial neural networks, Monthly Notes of the Royal Astronomical Society 295 (2) (1998) 312–318.
[7] E.F. Vieira, J.D. Ponz, Automated classification of IUE low-dispersion spectra. I. Normal stars, Astronomy and Astrophysics Supplement 111 (393) (1995).
[8] C. Bailer-Jones, M. Irwin, G. Gilmore, et al., Physical parameterization of stellar spectra – the neural network approach, Monthly Notices of the Royal Astronomical Society(MNRAS) 292 (1997) 157–166.
[9] A.J. Connolly, A.S. Szalay, M.A. Bershady, et al., Spectral classification of galaxies: an orthogonal approach, Astronomical Journal 110 (3) (1995) 1071–1082.
[10] Gaspar Galaz, Valerie de Lapparent, The ESO-sculptor survey: spectral classification of galaxies with $z \leqslant 0.5$, Astronomy and Astrophysics 332 (4) (1998) 459–478.
[11] D. Zaritsky, A.I. Zabludoff, A.W. Jeffrey, Spectral classification of galaxies along the hubble sequence, Astronomical Journal 110 (4) (1995) 1602–1614.
[12] S. Folkes et al., The 2dF galaxy redshift survey: spectral types and luminosity functions, Monthly Notices of the Royal Astronomical Society (MNRAS) 308 (1999) 459–476.
[13] D.-M. Qin, Z.-Y. Hu, Y.-H. Zhao, A PCA based efficient stellar spectra classification method, Spectroscopy and Spectral Analysis 23 (1) (2003) 182–186.
[14] Xiangru Li, Study on Several learning algorithms and its Application in Galaxy Classification [Ph.D. Dissertation], Institute of Automation, Chinese Academy of Science, June 2006.
[15] M.-F. Zhao, Automated spectral recognition and classification of galaxies [Ph.D. Dissertation], Institute of Automation Chinese Academy of Science, June 2006.
[16] R. Liu, F.-Q. Duan, et al., Spectral classification of galaxy based on wavelet feature, Acta Electronica Sinica 33 (11) (2005) 2059–2062.
[17] J.-F. Zhang, Y.-Y. Jiang, L.-H. Hu, et al., A concept lattice based recognition method of celestial spectra outliers, Acta Automatica Sinica 34 (9) (2008) 1060–1066.
[18] J.-F. Zhang, J.-H. Cai, A study on the outlier mining system for LAMOST spectra, Spectroscopy and Spectral Analysis 27 (3) (2007) 606–609.
[19] R. Agrawal, T. Imielinski, A. Swami, Mining association rule between sets of items in large databases, in: Proceedings of 1st International Conference on Management of Data, 1993, pp. 207–216.
[20] J.-W. Han, J. Pei, Y.-W. Yin, R.-Y. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Journal of Data Mining and Knowledge Discovery 8 (1) (2004) 53–87.
[21] J. Pei, H.-X. Wang, J. Liu, et al., Discovering frequent closed partial orders from strings, IEEE Transactions on Knowledge and Data Engineering 18 (11) (2006) 1467–1481.
[22] Chen, C.-J. Tang, H.-C. Tao, et al., An improved algorithm based on maximum clique and FP-tree for mining association rules, Journal of Software 15 (8) (2004) 1198–1207.
[23] T.S. Huang, Two-Dimensional Digital Signal Processing, Science Press, Chinese Beijing, 1985.
[24] Y.-Q. Wang, Principles and Methods of Artificial Intelligence, Xi'an Jiao Tong University Press, 2003.
[25] J. Pei, J. Han, R. Mao, CLOSET: an efficient algorithm for mining frequent closed item sets, in: Proceedings of ACM-SIGMOD Workshop Data Mining and Knowledge Discovery (DMKD'00), Dallas, TX, May 2000.
[26] J.-Y. Wang, J.-W. Han, TFP: an efficient algorithm for mining top-K frequent closed item sets, IEEE Transactions on Knowledge and Data Engineering 17 (5) (2005) 652–664.
[27] Mohammed J. Zaki, Efficient algorithms for mining closed item sets and their lattice structure, IEEE Transactions on Knowledge and Data Engineering 17 (4) (2005) 462–478.
[28] E. Gudes, S.E. Shimony, N. Vanetik, Discovering frequent graph patterns using disjoint paths, IEEE Transactions on Knowledge and Data Engineering 18 (11) (2006) 1441–1456.
[29] Mingjun Song, Sanguthevar Rajasekaran, A transaction mapping algorithm for frequent item sets mining, IEEE Transactions on Knowledge and Data Engineering 18 (4) (2006) 472–481.
[30] C. Lucchese, S. Orlando, R. Perego, Fast and memory efficient mining of frequent closed item sets, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 21–36.
[31] A.T.H. Sim, M. Indrawan, S. Zutshi, Logic-based pattern discovery, IEEE Transactions on Knowledge and Data Engineering 22 (6) (2010) 798–811.
[32] S. Ruggieri, Frequent regular itemset mining, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 25–28, 2010, Washington, DC, USA, pp. 263–272.
[33] Wei Song, Bingru Yang, Zhangyan Xu, Index-BitTableFI: an improved algorithm for mining frequent itemsets, Knowledge-Based Systems 21 (6) (2008) 507–513.