Hindawi
Discrete Dynamics in Nature and Society
Volume 2017, Article ID 6978146, 11 pages
https://doi.org/10.1155/2017/6978146

Research Article

An Improved Apriori Algorithm Based on an Evolution-Communication Tissue-Like P System with Promoters and Inhibitors

Xiyu Liu,[1] Yuzhen Zhao,[1] and Minghe Sun[2]

[1]*College of Management Science and Engineering, Shandong Normal University, Jinan, Shandong, China*
[2]*College of Business, The University of Texas at San Antonio, San Antonio, TX, USA*

Correspondence should be addressed to Yuzhen Zhao; 723567558@qq.com

Received 4 November 2016; Revised 6 January 2017; Accepted 30 January 2017; Published 19 February 2017

Academic Editor: Stefan Balint

Apriori algorithm, as a typical frequent itemsets mining method, can help researchers and practitioners discover implicit associations from large amounts of data. In this work, a fast Apriori algorithm, called ECTPPI-Apriori, for processing large datasets, is proposed, which is based on an evolution-communication tissue-like P system with promoters and inhibitors. The structure of the ECTPPI-Apriori algorithm is tissue-like and the evolution rules of the algorithm are object rewriting rules. The time complexity of ECTPPI-Apriori is substantially improved from that of the conventional Apriori algorithms. The results give some hints to improve conventional algorithms by using membrane computing models.

1. Introduction

Frequent itemsets mining, as a subfield of data mining, aims at discovering itemsets with high frequency from huge amounts of data. Interesting implicit associations between items then can be extracted from these data, which can help researchers and practitioners make informed decisions. One famous example is "beer and diapers" [1]. The supermarket management discovered a significant correlation between the purchases of beer and diapers which had nothing to do with each other ostensibly through frequent itemsets mining. Consequently, they put diapers next to beer. Through this layout adjustment, sales of both beer and diapers increased.

The Apriori algorithm is a typical frequent itemsets mining algorithm, which is suitable for the discovery of frequent itemsets in transactional databases [2]. To process large datasets, many parallel improvements have been made to improve the computational efficiency of the Apriori algorithm [3–6]. How to implement the Apriori algorithm in parallel to improve its computational efficiency is still an ongoing research topic. Given the extremity of the technology and theory of the silicon-based computing, new non-silicon-based computing devices P systems are used in this study.

P systems are new bioinspired computing models of membrane computing, which focus on abstracting computing ideas from the study of biological cells, particularly of cellular membranes [7, 8]. This study uses an evolution-communication tissue-like P system with promoters and inhibitors (ECPI tissue-like P systems) for computation. P systems are powerful distributed and parallel bioinspired computing devices, being able to do what Turing machines can do [9–11], and have been applied to many fields. The applications of P systems are based on two types of membrane algorithms, the coupled membrane algorithm and the direct membrane algorithm. The coupled membrane algorithm combines the traditional algorithm with some structural characters of P systems, such as dividing the whole system into several relatively independent computing units, where the computing units can communicate with each other, the computing units can be dynamically rebuilt, and rules can be executed in parallel [12–16]. The direct membrane algorithm designs the algorithm based on the structure, the objects, and the rules of P systems directly [17–21]. The final goal of membrane computing is to build biocomputers and the direct membrane algorithm can be transplanted to the biocomputers directly, which is more meaningful from this

perspective. However, the direct membrane algorithm needs to transform the whole traditional algorithm into P system, which is complex and difficult. Up to date, a few simple studies on the direct membrane algorithm focus on the arithmetic operations, the logic operations, the generation of graphic language, and clustering [17–21].

In this study, a novel improved Apriori algorithm based on an ECPI tissue-like P system (ECTPPI-Apriori) is proposed using the parallel mechanism in P systems. The information communication between different computing units in ECTPPI-Apriori is implemented through the exchange of materials between membranes. Specifically, all itemsets are searched in parallel, regulated by a set of promoters and inhibitors. For a database with t fields, $t + 2$ cells are used in the algorithm, where 1 cell is used to enter the data in the database into the system, t cells are used to detect the frequent itemsets, and one specific cell, called output cell, is used to store the results. The time complexity of ECTPPI-Apriori is compared with those of other parallel Apriori algorithms to show that the proposed algorithm is time saving.

The contributions of this study are twofold. From the viewpoint of data mining, new bioinspired techniques are introduced into frequent itemsets mining to improve the efficiency of the algorithms. P systems are natural distributed parallel computing devices which can improve the time efficiency in computation. Besides the hardware and software implementations, P systems can be implemented by biological methods. The computing resources needed are only several cells, which can decrease the computing resource requirements. From the viewpoint of P systems, the application areas of the new bioinspired devices P systems are extended to frequent itemset mining. The applications based on the direct membrane algorithms are limited. This study provides a new application of P systems in frequent itemsets mining, which expands the application areas of the direct membrane algorithms.

The paper is organized as follows. Section 2 introduces some preliminaries about the Apriori algorithm and about the ECPI tissue-like P systems. The ECTPPI-Apriori algorithm using the parallel mechanism of the ECPI tissue-like P system is developed in Section 3. In Section 4, one illustrative example is used to show how the proposed algorithm works. Computational experiments using two datasets to show the performance of the proposed algorithm in frequent itemsets mining are reported in Section 5. Conclusions are given in Section 6.

2. Preliminaries

In this section, some basic concepts and notions in Apriori algorithm [2] and ECPI tissue-like P system [7] are introduced.

2.1. The Apriori Algorithm. The Apriori algorithm is a typical frequent itemsets mining algorithm proposed by Agrawal and Srikant [2], which aims at discovering relationships between items in transactional databases.

Definitions

(i) *Item:* a field in a transactional database is called an item. If one record contains a certain item "1," otherwise "0," is placed in the corresponding field of the record in the transactional database.

(ii) *Itemset:* a set of items is called an itemset. For notational convenience, an itemset with n items $I_1, I_2, \ldots,$ and I_n is represented by $\{I_1, I_2, \ldots, I_n\}$.

(iii) *h-itemset:* a set containing h items is called a h-itemset.

(iv) *Transaction:* a record in a transactional database is called a transaction, and each transaction is a nonempty itemset.

(v) *Support count:* the number of transactions containing a certain itemset is called the support count of the itemset. Support count is also called the frequency or count of the itemset.

(vi) *Frequent itemset:* if the support count of an itemset is equal to or larger than the given minimum support count threshold k, this itemset is called a frequent itemset.

The general procedure of Apriori from Han et al. [1] is as follows.

Input. The database contains D transactions and the support count threshold k.

Step 1. Scan the database to compute the support count of each item, and obtain the frequent 1-itemsets L_1. Let $h = 2$.

Step 2. Obtain the candidate frequent h-itemsets C_h by joining two frequent $(h - 1)$-itemsets with only one item different.

Step 3. Prune those itemsets which have infrequent subset of length $(h - 1)$ from the candidate frequent h-itemsets.

Step 4. Scan the database to compute the support count of each candidate frequent h-itemset. Delete those itemsets which do not meet the support count threshold k and obtain the frequent h-itemsets L_h.

Step 5. Let $h = h + 1$. Repeat Steps 2 to 4 until no itemset meets the support count threshold k.

Output. The collection of all frequent itemsets is represented by L.

2.2. Evolution-Communication Tissue-Like P Systems with Promoters and Inhibitors. Membrane computing is a new branch of natural computing, which abstracts computing ideas from the construct and the functions of cells or tissues. In the nature, each organelle membrane or cell membrane works as a relatively independent computing unit. The amount and the types of materials in each organelle or cell change through chemical reactions. Materials can flow between different organelle or cell membranes to transport information.

Reactions in different organelles or cells take place in parallel, while reactions in the same organelle or cell take place also in parallel. These biological processes are abstracted as the computing processes of membrane computing. The internal parallel feature makes membrane computing a powerful computing method which has been proven to be equivalent to Turing machines [7–11].

The ECPI tissue-like P system, composed of a network of cells linked by synapses (channels), is a typical membrane computing model. The whole P system is divided into separate regions through these cells, each forming one region. Each cell has two main components, multisets of objects (materials) and rules, also called evolution rules (chemical reactions). Objects, as information carriers, are represented by characters.

Rules regulate the ways objects evolve to new objects and the ways objects in different cells communicate through synapses. Rules are executed in nondeterministic flat maximally parallel in each cell. That is, at any step, if more than one rule can be executed but the objects in the cell can only support some of them, then a maximal number of rules will be executed, and each rule can be executed for only once [22].

The computation halts if no rule can be executed in the whole system. The computational results are represented by the types and numbers of specified objects in a specified cell. Because objects in a P system evolve in flat maximally parallel, regulated by promoters and inhibitors, the systems compute very efficiently [10, 22]. Păun [7] provided more details about P systems.

A formal description of the ECPI tissue-like P system is as follows.

An ECPI tissue-like P system of degree m is of the form

$$\Pi = \left(O, \sigma_1, \sigma_2, \ldots, \sigma_m, \text{syn}, \rho, i_{\text{out}}\right), \quad (1)$$

where (1) O represents the alphabets including all objects of the system. (2) $\text{syn} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ represents all synapses between the cells. (3) ρ defines the partial ordering relationship of the rules; that is, rules with higher orders are executed with higher priority. (4) i_{out} represents the subscript of the output cell where the computation results are placed. (5) $\sigma_1, \ldots, \sigma_m$ represent the m cells. Each cell is of the form

$$\sigma_h = \left(w_{h,0}, R_h\right), \quad \text{for } 1 \le h \le m. \quad (2)$$

In (2), $w_{h,0}$ represents the initial objects in cell h. A $w_{h,0} = \lambda$ means that there is no object in cell h. If a represents an object, a^n represents the multiplicity of n copies of such objects. R_h in (2) represents a set of rules in cell h with the form of $w_z \rightarrow xy_{\text{go}}$, where w is the multiset of objects consumed by the rule, z in the subscript is the promoter or the inhibitor of the form $z = z'$ or $z = \neg z'$, and x and y are the multisets of objects generated by the rule. A rule can be executed only when all objects in the promoter appear and cannot be executed when any objects in the inhibitor appear. Multiset of objects x stay in the current cell, and multiset of objects y go to the cells which have synapses connected from the current cell. The qth subset of rules in cell h having similar functions is represented by r_{hq}, and the rules in the same subset are connected by $\cup$.

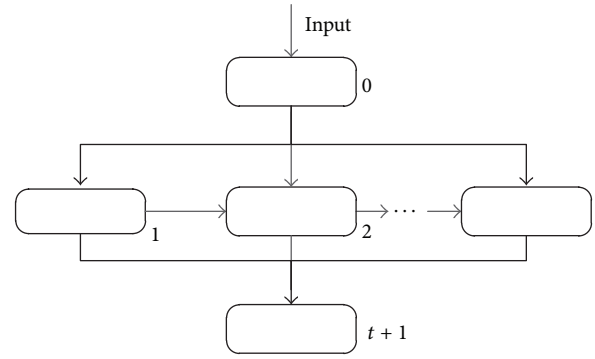

Figure 1: Cell structure for the ECTPPI-Apriori algorithm.

3. The ECTPPI-Apriori Algorithm

In this section, the structure of the P system used in the ECTPPI-Apriori algorithm is presented first, the computational processes in different cells are then discussed in detail, a pseudocode summarizing the operations is presented, and an analysis of the algorithm complexity is provided.

3.1. Algorithm and Rules. Assume a transactional database contains D records and t fields. An object a_{ij} is generated only if the ith transaction contains the jth item I_j (i.e., there is a 1 in the corresponding field in the transactional database). In this way, the database is transformed into objects, a form that the P system can recognize. The support count threshold is set to k. A cell structure with $t + 2$ cells, labeled by $0, 1, \ldots, t + 1$, as shown in Figure 1, is used as the framework for ECTPPI-Apriori. The evolution rules are not shown in this figure due to their length. Transactional databases are usually sparse. Therefore, the number of objects, represented by a_{ij}, to be processed in this algorithm is much smaller than Dt.

When computation begins, objects a_{ij} encoded from the transactional database and object θ^k representing the support count threshold k are entered into cell 0. Objects a_{ij} and θ^k are passed to cells $1, 2, \ldots, t$ in parallel, using a parallel evolution mechanism in tissue-like P systems. The auxiliary objects β_j^k are generated in cell 1. Next, the frequent 1-itemsets are produced and objects representing frequent 1-itemsets are generated in cell 1 by executing the evolution rules in parallel. The objects representing the frequent 1-itemsets are passed to cells 2 and $t + 1$. Cell $t + 1$ is used to store the computational results. The frequent 2-itemsets and the objects representing the frequent 2-itemsets are produced in cell 2 by executing the evolution rules in parallel. The objects representing the frequent 2-itemsets are passed to cells 3 and $t + 1$. This process continues until all frequent itemsets have been produced. As compared with that of the conventional Apriori algorithm, the computational time needed by ECTPPI-Apriori to generate the candidate frequent h-itemsets C_h and to compute the support count of each candidate frequent h-itemset can be substantially reduced.

The ECPI tissue-like P system for ECTPPI-Apriori is as follows.

$$\Pi_{\text{Apriori}} = \left(O, \sigma_0, \sigma_1, \ldots, \sigma_{t+1}, \text{syn}, \rho, i_{\text{out}}\right), \quad (3)$$

where (1) $O = \{a_{ij}, \beta_j, \beta_{j_1 j_2}, \ldots, \beta_{j_1 \cdots j_t}, \delta_j, \delta_{j_1 j_2}, \ldots, \delta_{j_1 \cdots j_t}, \alpha_j, \alpha_{j_1 j_2}, \ldots, \alpha_{j_1 \cdots j_t}\}$ for $1 \le i \le D$, $1 \le j, j_1, \ldots, j_t \le t$; (2) $\text{syn} = \{\{0,1\}, \{0,2\}, \ldots, \{0,t\}; \{1,t+1\}, \{2,t+1\}, \ldots, \{t,t+1\}; \{1,2\}, \{2,3\}, \ldots, \{t-1,t\}\}$; (3) $\rho = \{r_i > r_j \mid i < j\}$; (4) $\sigma_0 = (w_{0,0}, R_0)$, $\sigma_1 = (w_{1,0}, R_1)$, $\sigma_2 = (w_{2,0}, R_2), \ldots, \sigma_t = (w_{t,0}, R_t)$; (5) $i_{\text{out}} = t+1$.

In $\sigma_0 = (w_{0,0}, R_0)$, $w_{0,0} = \lambda$ and
R_0:
$r_{01} = \{a_{ij} \to a_{ij,\text{go}}\} \cup \{\theta^k \to \theta^k_{\text{go}}\}$
for $1 \le i \le D$ and $1 \le j \le t$.

In $\sigma_1 = (w_{1,0}, R_1)$, $w_{1,0} = \lambda$ and
R_1:
$r_{11} = \{\theta^k \to \beta^k_1 \cdots \beta^k_t\}$.
$r_{12} = \{\delta^p_j \beta^{k-p}_j \to \lambda\} \cup \{(\delta^k_j)_{\neg\beta_j} \to \alpha_{j,\text{go}}\}$
for $1 \le j \le t$ and $1 \le p \le k$.
$r_{13} = \{a_{ij}\beta_j \to \delta_j\}$
for $1 \le i \le D$ and $1 \le j \le t$.

In $\sigma_2 = (w_{2,0}, R_2)$, $w_{2,0} = \lambda$ and
R_2:
$r_{21} = \{()_{\alpha_{j_1}\alpha_{j_2}\theta^k\neg\beta^k_{j_1 j_2}} \to \beta^k_{j_1 j_2}\}$
for $1 \le j_1 < j_2 \le t$.
$r_{22} = \{\alpha_j \to \lambda\}$ $(1 \le j \le t)$.
$r_{23} = \{\delta^p_{j_1 j_2}\beta^{k-p}_{j_1 j_2} \to \lambda\} \cup \{(\delta^k_{j_1 j_2})_{\neg\beta_{j_1 j_2}} \to \alpha_{j_1 j_2,\text{go}}\}$
for $1 \le j_1 < j_2 \le t$ and $1 \le p \le k$.
$r_{24} = \{(\beta_{j_1 j_2})_{a_{ij_1}a_{ij_2}} \to \delta_{j_1 j_2}\}$
for $1 \le i \le D$ and $1 \le j_1 < j_2 \le t$.

$\vdots$

In $\sigma_h = (w_{h,0}, R_h)$, $w_{h,0} = \lambda$ and
R_h:
$r_{h1} = \{()_{\alpha_{j_1\cdots j_{h-2}j_{h_1}}\alpha_{j_1\cdots j_{h-2}j_{h_2}}\theta^k\neg\beta^k_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}}} \to \beta^k_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}}\}$
for $1 \le j_1 < \cdots < j_{h-2} \le t$ and $1 \le j_{h_1}, j_{h_2} \le t$.
$r_{h2} = \{\alpha_{j_1\cdots j_{h-1}} \to \lambda\}$
for $1 \le j_1 < \cdots < j_{h-1} \le t$.
$r_{h3} = \{\delta^p_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}}\beta^{k-p}_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}} \to \lambda\} \cup \{(\delta^k_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}})_{\neg\beta_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}}} \to \alpha_{j_1\cdots j_{h-2}j_{h_1}j_{h_2},\text{go}}\}$
for $1 \le j_1 < \cdots < j_{h-2} \le t$, $1 \le j_{h_1}, j_{h_2} \le t$ and $1 \le p \le k$.
$r_{h4} = \{(\beta_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}})_{a_{ij_1}\cdots a_{ij_{h-2}j_{h_1}j_{h_2}}} \to \delta_{j_1\cdots j_{h-2}j_{h_1}j_{h_2}}\}$
for $1 \le i \le D$, $1 \le j_1 < \cdots < j_{h-2} \le t$ and $1 \le j_{h_1}, j_{h_2} \le t$.

$\vdots$

In $\sigma_t = (w_{t,0}, R_t)$, $w_{t,0} = \lambda$ and
R_t:
$r_{t1} = \{()_{\alpha_{j_1\cdots j_{t-2}j_{t_1}}\alpha_{j_1\cdots j_{t-2}j_{t_2}}\theta^k\neg\beta^k_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}}} \to \beta^k_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}}\}$
for $1 \le j_1 < \cdots < j_{t-2} \le t$ and $1 \le j_{t_1}, j_{t_2} \le t$.
$r_{t2} = \{\alpha_{j_1\cdots j_{t-1}} \to \lambda\}$
for $1 \le j_1 < \cdots < j_{t-1} \le t$.
$r_{t3} = \{\delta^p_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}}\beta^{k-p}_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}} \to \lambda\} \cup \{(\delta^k_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}})_{\neg\beta_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}}} \to \alpha_{j_1\cdots j_{t-2}j_{t_1}j_{t_2},\text{go}}\}$
for $1 \le j_1 < \cdots < j_{t-2} \le t$, $1 \le j_{t_1}, j_{t_2} \le t$ and $1 \le p \le k$.
$r_{t4} = \{(\beta_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}})_{a_{ij_1}\cdots a_{ij_{t-2}j_{t_1}j_{t_2}}} \to \delta_{j_1\cdots j_{t-2}j_{t_1}j_{t_2}}\}$
for $1 \le i \le D$, $1 \le j_1 < \cdots < j_{t-2} \le t$ and $1 \le j_{t_1}, j_{t_2} \le t$.

In $\sigma_{t+1} = (w_{t+1,0}, R_{t+1})$, $w_{t+1,0} = \lambda$ and $R_{t+1} = \varnothing$.

The auxiliary objects β^k_j and δ_j, for $1 \le j \le t$, are used to detect the frequent 1-itemsets. The auxiliary objects β^k_j store the items in the candidate frequent 1-itemsets using their subscripts. For example, object β^k_1 means the itemset $\{I_1\}$ is a candidate frequent 1-itemset. The auxiliary objects δ_j are used to identify the frequent 1-itemsets. The k copies of β_j initially in cell 1 indicate that the jth item needs to appear in at least k records to make the itemset $\{I_j\}$ a frequent 1-itemset. One object β_j is removed from, and one object δ_j is generated in cell 1 when one more of the jth item is detected in a record. Therefore, if there is no object β_j left and k objects δ_j have been generated in cell 1, at least k records have been found to contain the jth item. The functions of $\beta^k_{j_1 j_2}$ in cells 2, $\beta^k_{j_1 j_2 j_3}$ in cell 3, ..., and $\beta^k_{j_1 \cdots j_t}$ in cell t are similar to that of β^k_j in cell 1. The functions of $\delta_{j_1 j_2}$ in cell 2, $\delta_{j_1 j_2 j_3}$ in cell 3, ..., and $\delta_{j_1 \cdots j_t}$ in cell t are similar to that of δ_j in cell 1. The objects α_j, for $1 \le j \le t$, are used to store the items in the frequent 1-itemsets using their subscript. For example, α_1 means the itemset $\{I_1\}$ is a frequent 1-itemset. The functions of $\alpha_{j_1 j_2}$ in cell 2, $\alpha_{j_1 j_2 j_3}$ in cell 3, ..., and $\alpha_{j_1 \cdots j_t}$ in cell t are similar to that of α_j in cell 1.

The evolution rules are object rewriting rules similar to chemical reactions. They take objects, transform them into other objects, and may transport them to other cells.

3.2. Computing Process

Input. Cell 0 is the input cell. The objects a_{ij} encoded from the transactional database and objects θ^k representing the

support count threshold k are entered into cell 0 to activate the computation process. Rule r_{01} is executed to put copies of a_{ij} and θ^k to cells $1, 2, \ldots, t$.

Frequent 1-Itemsets Generation. Frequent 1-itemsets are generated in cell 1. Rule r_{11} is executed to generate β_j^k for $1 \le j \le t$. Rule r_{13} is executed to detect all frequent 1-itemsets using the internal flat maximally parallel mechanism in the P system. Rule r_{12} cannot be executed because no object δ_j is in cell 1 at this time. The detection process of the candidate frequent 1-itemset $\{I_1\}$ is taken as an example. The detection processes of other candidate frequent 1-itemsets are performed in the same way. Rule r_{13} is actually composed of multiple subrules working on objects with different subscripts. If object a_{i1} is in cell 1 which means the ith record contains the first item, the subrule $\{a_{i1}\beta_1 \to \delta_1\}$ meets the execution condition and can be executed. If object a_{i1} is not in cell 1 which means the ith record does not contain the first item, the subrule $\{a_{i1}\beta_1 \to \delta_1\}$ does not meet the execution condition and cannot be executed. Initially, k copies of β_1 are in cell 1 indicating that the first item needs to appear in at least k records for the itemset $\{I_1\}$ to be a frequent 1-itemset. Each execution of a subrule consumes one β_1. Therefore, at most k subrules of the form $\{a_{i1}\beta_1 \to \delta_1\}$ can be executed in nondeterministic flat maximally parallel. The checking process continues until all objects a_{ij} have been checked or all of the k copies of β_1 have been consumed. If all of the k copies of β_1 have been consumed, the first item appeared in at least k records and the itemset $\{I_1\}$ is a frequent 1-itemset. If some copies of β_1 are still in this cell after all objects a_{ij} have been checked, the itemset $\{I_1\}$ is not a frequent 1-itemset.

Rule r_{12} is then executed to process the results obtained by rule r_{13}. The 1-itemset $\{I_1\}$ is again taken as an example. If p copies of β_1 have been consumed by rule r_{13}, and $k-p$ copies of β_1 are still in this cell, subrule $\{\delta_1^p \beta_1^{k-p} \to \lambda\}$ is executed to delete the objects δ_1^p and β_1^{k-p}. If all of the k copies of β_1 have been consumed, subrule $\{(\delta_1^k)_{\neg\beta_1} \to \alpha_{1,\mathrm{go}}\}$ is executed to put an object α_1 to cells 2 and $t+1$ to indicate that the itemset $\{I_1\}$ is a frequent 1-itemset and to activate the computation in cell 2. If no 1-itemset is a frequent 1-itemset, the computation halts.

Frequent 2-Itemsets Generation. The frequent 2-itemsets are generated in cell 2. Rule r_{21} is executed to obtain all candidate frequent 2-itemsets using the internal flat maximally parallel mechanism in the P system. The pair of empty parentheses in this subrule indicates that no objects are consumed when this rule is executed. The detection process of the candidate frequent 2-itemset $\{I_1, I_2\}$ is taken as an example. The detection processes of the other candidate frequent 2-itemsets are performed in the same way. Rule r_{21} is actually composed of multiple subrules working on objects with different subscripts. If objects α_1 and α_2 are in cell 2, which means itemsets $\{I_1\}$ and $\{I_2\}$ are frequent 1-itemsets, subrule $\{()_{\alpha_1\alpha_2\theta^k\neg\beta_{12}^k} \to \beta_{12}^k\}$ is executed to generate β_{12}^k. The presence of β_{12}^k means the 2-itemset $\{I_1, I_2\}$ is a candidate frequent 2-itemset.

Rule r_{22} is executed to delete the redundant objects α_j that were used by rule r_{21} but are not needed anymore.

Rule r_{24} is executed to detect all frequent 2-itemsets using the internal flat maximally parallel mechanism in the P system. Rule r_{23} cannot be executed because no object $\delta_{j_1 j_2}$ is in cell 2 at this time. The detection process of the frequent 2-itemset $\{I_1, I_2\}$ is taken as an example. Rule r_{24} is actually composed of multiple subrules working on objects with different subscripts. If objects a_{i1} and a_{i2} are in cell 2 which means the ith record contains the first and the second items, subrule $\{(\beta_{12})_{a_{i1}a_{i2}} \to \delta_{12}\}$ meets the execution condition and can be executed. If objects a_{i1} and a_{i2} are not both in cell 2 which means the ith record does not contain both the first and the second items, subrule $\{(\beta_{12})_{a_{i1}a_{i2}} \to \delta_{12}\}$ does not meet the execution condition and cannot be executed. Initially, k copies of β_{12} are in cell 2 indicating that both the first and the second items need to appear together in at least k records for the itemset $\{I_1, I_2\}$ to be a frequent 2-itemset. Each execution of these subrules consumes one β_{12}. Therefore, at most k subrules of the form $\{(\beta_{12})_{a_{i1}a_{i2}} \to \delta_{12}\}$ can be executed in nondeterministic flat maximally parallel. The checking process continues until all objects a_{ij} have been checked or all of the k copies of β_{12} have been consumed. If all of the k copies of β_{12} have been consumed, the first and the second items appeared together in at least k records and the itemset $\{I_1, I_2\}$ is a frequent 2-itemset. If some copies of β_{12} are still in this cell after rule r_{24} is executed, the itemset $\{I_1, I_2\}$ is not a frequent 2-itemset.

Rule r_{23} is executed to process the results obtained by rule r_{24}. The 2-itemset $\{I_1, I_2\}$ is again taken as an example. If p copies of β_{12} have been consumed by rule r_{24}, and $k-p$ copies of objects β_{12} are still in this cell, subrule $\{\delta_{12}^p \beta_{12}^{k-p} \to \lambda\}$ is executed to delete the objects δ_{12}^p and β_{12}^{k-p}. If all of the k copies of β_{12} have been consumed, subrule $\{(\delta_{12}^k)_{\neg\beta_{12}} \to \alpha_{12,\mathrm{go}}\}$ is executed to put an object α_{12} to cells 3 and $t+1$ to indicate that the itemset $\{I_1, I_2\}$ is a frequent 2-itemset and to activate the computation in cell 3. If no 2-itemset is a frequent 2-itemset, the computation halts.

Each cell j for $3 \le j \le t$ has 4 rules which are similar to those in cell 2. Each cell j performs similar functions as cell 2 does but for frequent j-itemsets.

After the computation halts, all the results, that is, objects representing the identified frequent itemsets, are stored in cell $t+1$.

3.3. Algorithm Flow. The conventional Apriori algorithm executes sequentially. ECTPPI-Apriori uses the parallel mechanism of the ECPI tissue-like P system to execute in parallel. A pseudocode of ECTPPI-Apriori is shown as in Algorithm 1.

3.4. Time Complexity. The time complexity of ECTPPI-Apriori in the worst case is analyzed. Initially, 1 computational step is needed to put copies of a_{ij} and θ^k to cells $1, 2, \ldots, t$.

Generating the frequent 1-itemsets needs 3 computational steps. Generating the candidate frequent 1-itemsets C_1 needs 1 computational step. Finding the support counts of the candidate frequent 1-itemsets needs 1 computational step. All

Input:
The objects a_{ij} encoded from the transactional database and objects θ^k representing the support count threshold k.
Rule r_{01}:
Copy all a_{ij} and θ^k to cells 1 to $t+1$.
Method:
{
Rule r_{11}:
Generate β_j^k for $1 \le j \le t$ to form the candidate frequent 1-itemsets C_1.
Rule r_{13}:
Scan each object a_{ij} in cell 1 to count the frequency of each item. If a_{ij} is in cell 1, consume one β_j and generate one δ_j. Continue until all k copies of β_j have been consumed or all objects a_{ij} have been scanned.
Rule r_{12}:
If all k copies of β_j have been consumed, generate an object α_j to add $\{I_j\}$ to L_1 as a frequent 1-itemset and pass α_j to cells 2 and $t+1$. Delete all remaining copies of β_j and delete all copies of δ_j.
For ($2 \le h \le t$ and $L_{h-1} \ne \varnothing$) do the following in cell h:
{
Rule r_{h1}:
Scan the objects $\alpha_{j_1 \cdots j_{h-1}}$ representing the frequent $(h-1)$-itemsets L_{h-1} to generate the objects $\beta^k_{j_1 \cdots j_h}$ representing the candidate frequent h-itemsets C_h.
Rule r_{h2}:
Delete all objects $\alpha_{j_1 \cdots j_{h-1}}$ after they have been used by rule r_{h1}.
Rule r_{h4}:
Scan the objects a_{ij} representing the database to count the frequency of each candidate frequent h-itemset C_h. If the objects $a_{i,j_1}, \ldots, a_{i,j_{h-1}}$ and a_{i,j_h} are all in cell h, consume one $\beta_{j_1 \cdots j_h}$ and generate one $\delta_{j_1 \cdots j_h}$. Continue until all copies of $\beta_{j_1 \cdots j_h}$ have been consumed or all objects a_{ij} have been scanned.
Rule r_{h3}:
If all k copies of $\beta_{j_1 \cdots j_h}$ have been consumed, generate $\alpha_{j_1 \cdots j_h}$ to add $\{I_{j_1}, \ldots, I_{j_h}\}$ to L_h as a frequent h-itemset and put $\alpha_{j_1 \cdots j_h}$ in cells $h+1$ and $t+1$. Delete all remaining copies of $\beta_{j_1 \cdots j_h}$ and delete all copies of $\delta_{j_1 \cdots j_h}$.
Let $h = h+1$.
}
Output:
The collection of all frequent itemsets L encoded by objects $\alpha_j, \alpha_{j_1 j_2}, \ldots, \alpha_{j_1 \cdots j_n}$.

Algorithm 1: ECTP-Apriori algorithm.

candidate frequent 1-itemsets in the database are checked in the flat maximally parallel. Passing the results of the frequent 1-itemsets to cells 2 and $t+1$ needs 1 computational step.

Generating the frequent h-itemsets ($1 < h \le t$) needs 4 computational steps. Generating the candidate frequent h-itemsets C_h needs 1 computational step. Cleaning the memory used by the objects needs 1 computational step. Finding the support counts of the candidate frequent h-itemsets needs 1 computational step. All candidate frequent h-itemsets in the database are checked in flat maximally parallel. Passing the results of the frequent h-itemsets to cells $h+1$ and $t+1$ needs 1 computational step.

Therefore, the time complexity of ECTPPI-Apriori is $1 + 3 + 4(t-1) = 4t$, which gives $O(t)$. Note that O is used traditionally to indicate the time complexity of an algorithm and the O used here has a different meaning from that used earlier when ECTPPI-Apriori is described.

Some comparison results between ECTPPI-Apriori and the original as well as some other improved parallel Apriori algorithms are shown in Table 1, where $|C_k|$ is the number of candidate frequent k-itemsets and $|L_{k-1}|$ is the number of frequent $k-1$-itemsets.

Table 1: Time complexities of some Apriori algorithms.

Algorithm	Time complexity
Apriori [2]	$O\left(Dt\left\|C_k\right\| + t\left\|L_{k-1}\right\|\left\|L_{k-1}\right\|\right)$
Apriori based on the boolean matrix and Hadoop [3]	$O(Dt)$
Multi-GPU-based Apriori [4]	$O(t\|L_{k-1}\|\|L_{k-1}\|)$
PApriori [5]	$O(Dt\|C_k\| + t\|L_{k-1}\|\|L_{k-1}\|)$
Parallel Apriori algorithm on Hadoop cluster [6]	$O(Dt\|C_k\| + t\|L_{k-1}\|\|L_{k-1}\|)$
ECTPPI-Apriori	$O(t)$

4. An Illustrative Example

An illustrative example is presented in this section to demonstrate how ECTPPI-Apriori works. Table 2 shows the

Table 2: The transactional database of the illustrative example.

TID	Items
T100	I_1, I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

transactional database of one branch office of All Electronics [1]. There are 9 transactions and 5 fields in this database; that is, $D = 9$ and $t = 5$. Suppose the support count threshold is $k = 2$. The computational processes are as follows.

Input. The database is transformed into objects a_{11}, a_{12}, a_{15}, a_{22}, a_{24}, a_{32}, a_{33}, a_{41}, a_{42}, a_{44}, a_{51}, a_{53}, a_{62}, a_{63}, a_{71}, a_{73}, a_{81}, a_{82}, a_{83}, a_{85}, a_{91}, a_{92}, and a_{93}, in a form that the P system can recognize. These objects and objects θ^2 representing the support count threshold $k = 2$ are entered into cell 0 to activate the computation process. Rule $r_{01} = \{a_{ij} \to a_{ij,\mathrm{go}}\} \cup \{\theta^k \to \theta^k_{\mathrm{go}}\}$ is executed to put copies of a_{11}, a_{12}, a_{15}, a_{22}, a_{24}, a_{32}, a_{33}, a_{41}, a_{42}, a_{44}, a_{51}, a_{53}, a_{62}, a_{63}, a_{71}, a_{73}, a_{81}, a_{82}, a_{83}, a_{85}, a_{91}, a_{92}, a_{93}, and θ^2 to cells $1 \cdots 5$.

Frequent 1-Itemsets Generation. Within cell 1, the auxiliary objects β^2_j, for $1 \le j \le 5$, are created by rule r_{11} to indicate that each item j needs to appear in at least $k = 2$ records for it to be a frequent 1-itemset. Rule r_{13} is executed to detect all frequent 1-itemsets in flat maximally parallel. The detection process of the candidate frequent 1-itemset $\{I_1\}$ is taken as an example. Objects a_{11}, a_{41}, a_{51}, a_{71}, a_{81}, and a_{91} are in cell 1 which means the first, the fourth, the fifth, the seventh, the eighth, and the ninth records contain I_1. The subrules $\{a_{11}\beta_1 \to \delta_1\}$, $\{a_{41}\beta_1 \to \delta_1\}$, $\{a_{51}\beta_1 \to \delta_1\}$, $\{a_{71}\beta_1 \to \delta_1\}$, $\{a_{81}\beta_1 \to \delta_1\}$, and $\{a_{91}\beta_1 \to \delta_1\}$ meet the execution condition and can be executed. Objects a_{21}, a_{31}, and a_{61} are not in cell 1, which means the second, the third, and the sixth records do not contain I_1. The subrules $\{a_{21}\beta_1 \to \delta_1\}$, $\{a_{31}\beta_1 \to \delta_1\}$, and $\{a_{61}\beta_1 \to \delta_1\}$ do not meet the execution condition and cannot be executed. Initially, 2 copies of β_1 are in cell 1 indicating that the first item needs to appear in at least 2 records to make the itemset $\{I_1\}$ a frequent 1-itemset. Each execution of a subrule consumes one β_1. Therefore, 2 of subrules among $\{a_{11}\beta_1 \to \delta_1\}$, $\{a_{41}\beta_1 \to \delta_1\}$, $\{a_{51}\beta_1 \to \delta_1\}$, $\{a_{71}\beta_1 \to \delta_1\}$, $\{a_{81}\beta_1 \to \delta_1\}$, and $\{a_{91}\beta_1 \to \delta_1\}$ can be executed in nondeterministic flat maximally parallel. Through the execution of 2 such subrules, both of the 2 copies of β_1 are consumed and 2 copies of δ_1 are generated. The detection processes of other candidate frequent 1-itemsets are performed in the same way. After the detection processes, β^2_1, β^2_2, β^2_3, β^2_4, and β^2_5 are consumed, and δ^2_1, δ^2_2, δ^2_3, δ^2_4, and δ^2_5 are generated.

Rule r_{12} is then executed to process the results obtained by rule r_{13}. The 1-itemset $\{I_1\}$ is again taken as an example. All of the 2 copies of β_1 have been consumed, subrule $\{(\delta^2_1)_{\neg\beta_1} \to \alpha_{1,\mathrm{go}}\}$ is executed to put an object α_1 to cells 2 and 6 to indicate that the itemset $\{I_1\}$ is a frequent 1-itemset and to activate the computation in cell 2. Subrules $\{(\delta^2_2)_{\neg\beta_3} \to \alpha_{2,\mathrm{go}}\}$, $\{(\delta^2_3)_{\neg\beta_3} \to \alpha_{3,\mathrm{go}}\}$, $\{(\delta^2_4)_{\neg\beta_4} \to \alpha_{4,\mathrm{go}}\}$, and $\{(\delta^2_5)_{\neg\beta_5} \to \alpha_{5,\mathrm{go}}\}$ are also executed to put objects α_2, α_3, α_4, and α_5 to cells 2 and 6 to indicate that the itemsets $\{I_2\}$, $\{I_3\}$, $\{I_4\}$, and $\{I_5\}$ are frequent 1-itemsets and to activate the computation in cell 2.

Frequent 2-Itemsets Generation. Within cell 2, rule r_{21} is executed to obtain all candidate frequent 2-itemsets. The detection process of the candidate frequent 2-itemset $\{I_1, I_2\}$ is taken as an example. Objects α_1 and α_2 are in cell 2, which means itemsets $\{I_1\}$ and $\{I_2\}$ are frequent 1-itemsets. Subrule $\{()_{\alpha_1\alpha_2\theta^2\neg\beta^2_{12}} \to \beta^2_{12}\}$ is executed to generate β^2_{12}. The presence of β^2_{12} means the 2-itemset $\{I_1, I_2\}$ is a candidate frequent 2-itemset, and both I_1 and I_2 need to appear together in at least 2 records for the itemset $\{I_1, I_2\}$ to be a frequent 2-itemset. The detection processes of the other candidate frequent 2-itemsets are performed in the same way. After the detection processes, objects β^2_{12}, β^2_{13}, β^2_{14}, β^2_{15}, β^2_{23}, β^2_{24}, β^2_{25}, β^2_{34}, β^2_{35}, and β^2_{45} are generated.

Rule r_{22} is executed to delete the objects $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, and α_5 that are not needed anymore.

Rule r_{24} is executed to detect all frequent 2-itemsets. Objects a_{11} and a_{12}, a_{41} and a_{42}, a_{81} and a_{82}, and a_{91} and a_{92} are in cell 2, which means the first, the fourth, the eighth, and the ninth records contain both I_1 and I_2. The subrules $\{(\beta_{12})_{a_{11}a_{12}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{41}a_{42}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{81}a_{82}} \to \delta_{12}\}$, and $\{(\beta_{12})_{a_{91}a_{92}} \to \delta_{12}\}$ meet the execution condition and can be executed. Objects a_{21} and a_{22}, a_{31} and a_{32}, a_{51} and a_{52}, a_{61} and a_{62}, or a_{71} and a_{72} are not both in cell 2, which means the second, the third, the fifth, the sixth, and the seventh records do not contain both I_1 and I_2. The subrules $\{(\beta_{12})_{a_{21}a_{22}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{31}a_{32}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{51}a_{52}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{61}a_{62}} \to \delta_{12}\}$, and $\{(\beta_{12})_{a_{71}a_{72}} \to \delta_{12}\}$ do not meet the execution condition and cannot be executed. Initially, 2 copies of β_{12} are in cell 2 indicating that both I_1 and I_2 need to appear together in at least 2 records for the itemset $\{I_1, I_2\}$ to be a frequent 2-itemset. Each execution of these subrules consumes one β_{12}. Therefore, 2 subrules among $\{(\beta_{12})_{a_{11}a_{12}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{41}a_{42}} \to \delta_{12}\}$, $\{(\beta_{12})_{a_{81}a_{82}} \to \delta_{12}\}$, and $\{(\beta_{12})_{a_{91}a_{92}} \to \delta_{12}\}$ can be executed. After the execution, 2 copies of β_{12} are consumed and 2 copies of δ_{12} are generated. The detection processes of other candidate frequent 2-itemsets are performed in the same way. After the detection processes, objects β^2_{12}, β^2_{13}, β_{14}, β^2_{15}, β^2_{23}, β^2_{24}, β^2_{25}, and β_{35} are consumed, and objects δ^2_{12}, δ^2_{13}, δ_{14}, δ^2_{15}, δ^2_{23}, δ^2_{24}, δ^2_{25}, and δ_{35} are generated.

Rule r_{23} is executed to process the results obtained by rule r_{24}. The 2-itemset $\{I_1, I_2\}$ is again taken as an example. All of the 2 copies of β_{12} have been consumed by rule r_{24}, subrule $\{(\delta^2_{12})_{\neg\beta_{12}} \to \alpha_{12,\mathrm{go}}\}$ is executed to put an object α_{12} to cells 3 and 6 to indicate that the itemset $\{I_1, I_2\}$ is a frequent 2-itemset and to activate the computation in cell 3. Subrules $\{(\delta^2_{13})_{\neg\beta_{13}} \to \alpha_{13,\mathrm{go}}\}$, $\{(\delta^2_{15})_{\neg\beta_{15}} \to \alpha_{15,\mathrm{go}}\}$, $\{(\delta^2_{23})_{\neg\beta_{23}} \to \alpha_{23,\mathrm{go}}\}$, $\{(\delta^2_{24})_{\neg\beta_{24}} \to \alpha_{24,\mathrm{go}}\}$, and $\{(\delta^2_{25})_{\neg\beta_{25}} \to \alpha_{25,\mathrm{go}}\}$ are also executed, which put objects α_{13}, α_{15}, α_{23}, α_{24}, and α_{25} to cells

Table 3: Generation of frequent 1-itemsets.

r_{ij}	Cell 0	Cell 1	Cell 6
0	$a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}$ $a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}$ $a_{83}a_{85}a_{91}a_{92}a_{93}\theta^2$		
1	(r_{01})	$a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}$ $a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}$ $a_{83}a_{85}a_{91}a_{92}a_{93}\theta^2$	
2		$a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}$ $a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}$ $a_{83}a_{85}a_{91}a_{92}a_{93}\beta_1^2\beta_2^2\beta_3^2\beta_4^2\beta_5^2(r_{11})$	
3		$a_{32}a_{42}a_{51}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}$ $a_{83}a_{91}a_{92}a_{93}\delta_1^2\delta_2^2\delta_3^2\delta_4^2\delta_5^2(r_{13})$	
4		$a_{32}a_{42}a_{51}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}$ $a_{83}a_{91}a_{92}a_{93}(r_{12})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$

Table 4: Generation of frequent 2-itemsets.

r_{ij}	Cell 2	Cell 6
4	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$
5	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\beta_{25}^2\beta_{34}^2\beta_{35}^2\beta_{45}^2\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}(r_{21})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$
6	$\beta_{12}^2\beta_{13}^2\beta_{14}^2\beta_{15}^2\beta_{23}^2\beta_{24}^2\beta_{25}^2\beta_{34}^2\beta_{35}^2\beta_{45}^2\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}(r_{22})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$
7	$\delta_{12}^2\delta_{13}^2\delta_{14}\beta_{14}\delta_{15}^2\delta_{23}^2\delta_{24}^2\delta_{25}^2\beta_{34}^2\delta_{35}\beta_{35}\beta_{45}^2\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}(r_{24})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$
8	θ^2 $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}(r_{23})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$

3 and 6 to indicate that the itemsets $\{I_1, I_3\}$, $\{I_1, I_5\}$, $\{I_2, I_3\}$, $\{I_2, I_4\}$, and $\{I_2, I_5\}$ are frequent 2-itemsets and to activate the computation in cell 3.

The 4 rules in each cell h for $3 \leq h \leq 5$ are executed in ways similar to those in cell 2. The rules in these cells detect the frequent h-itemsets for $3 \leq h \leq 5$. After the computation halts, the objects α_1, α_2, α_3, α_4, α_5, α_{12}, α_{13}, α_{15}, α_{23}, α_{24}, α_{25}, α_{123}, and α_{125} are stored in cell 6, which means $\{I_1\}$, $\{I_2\}$, $\{I_3\}$, $\{I_4\}$, $\{I_5\}$, $\{I_1, I_2\}$, $\{I_1, I_3\}$, $\{I_1, I_3\}$, $\{I_2, I_3\}$, $\{I_2, I_4\}$, $\{I_2, I_5\}$, $\{I_1, I_2, I_3\}$, and $\{I_1, I_2, I_5\}$ are all frequent itemsets in this database.

The change of objects in the computation processes is listed in Tables 3–6.

5. Computational Experiments

Two databases from the UCI Machine Learning Repository [23] are used to conduct computational experiments. Computational results on these two databases are reported in this section.

5.1. Results on the Congressional Voting Records Database. The Congressional Voting Records database [23] is used to test the performance of ECTPPI-Apriori. This database contains 435 records and 17 attributes (fields). The first attribute is the party that the voter voted for and the 2nd to the 17th attributes are sixteen characteristics of each voter identified by the Congressional Quarterly Almanac. The first attribute has two values, Democrats or Republican, and each of the 2nd to the 17th attributes has 3 values: yea, nay, and unknown disposition. The frequent itemsets of these attribute values need to be identified; that is, the problem is to find the attribute values which always appear together.

Initially, the database is preprocessed. Each attribute value is taken as a new attribute. In this way, each new attribute has only two values: yes or no. After preprocessing, each record in the database has $2 \times 1 + 3 \times (17 - 1) = 50$ attributes. ECTPPI-Apriori then can be used to discover the frequent itemsets. In this experiment, one itemset is called a frequent itemset if it appeared in more than 40% of all records; that is, the support count threshold is $k = 174$ ($435 \times 40\%$). The frequent itemsets obtained by ECTPPI-Apriori are listed in Table 7.

5.2. Results on the Mushroom Database. The Mushroom database [23] is also used to test ECTPPI-Apriori. This database contains 8124 records. The 8124 records are numbered orderly from 1 to 8124. Each record represents one

Table 5: Generation of frequent 3-itemsets.

r_{ij}	Cell 3	Cell 6
8	$\alpha_{12}\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$
9	$\alpha_{12}\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}\beta^2_{123}\beta^2_{125}\beta^2_{135}\beta^2_{234}\beta^2_{235}\beta^2_{245}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{31})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$
10	$\beta^2_{123}\beta^2_{125}\beta^2_{135}\beta^2_{234}\beta^2_{235}\beta^2_{245}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{32})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$
11	$\delta^2_{123}\delta^2_{125}\delta_{135}\beta_{135}\beta^2_{234}\delta_{235}\beta_{235}\beta^2_{245}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{34})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$
12	θ^2 $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{33})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$

Table 6: Generation of frequent 4-itemsets.

r_{ij}	Cell 4	Cell 6
12	$\alpha_{123}\alpha_{125}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$
13	$\alpha_{123}\alpha_{125}\beta^2_{1235}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{41})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$
14	$\beta^2_{1235}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{42})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$
15	$\delta_{1235}\beta_{1235}\theta^2$ $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{44})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$
16	θ^2 $a_{11}a_{12}a_{15}a_{22}a_{24}a_{32}a_{33}a_{41}a_{42}a_{44}a_{51}a_{53}a_{62}a_{63}a_{71}a_{73}a_{81}a_{82}a_{83}a_{85}a_{91}a_{92}a_{93}\,(r_{43})$	$\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_{12}$ $\alpha_{13}\alpha_{15}\alpha_{23}\alpha_{24}\alpha_{25}$ $\alpha_{123}\alpha_{125}$

Table 7: The frequent itemsets identified by ECTPPI-Apriori for the Congressional Voting Records database.

Size of frequent itemset	The corresponding frequent itemsets
1	{2}; {3}; {5}; {6}; {8}; {9}; {12}; {14}; {15}; {17}; {18}; {21}; {23}; {24}; {26}; {27}; {29}; {30}; {32}; {35}; {38}; {39}; {41}; {42}; {45}; {47}; {48}
2	{2, 9}; {2, 14}; {2, 17}; {2, 21}; {2, 24}; {2, 27}; {2, 38}; {2, 41}; {5, 18}; {9, 14}; {9, 17}; {9, 21}; {9, 24}; {9, 27}; {9, 38}; {14, 17}; {14, 21}; {14, 24}; {14, 27}; {14, 38}; {14, 41}; {15, 18}; {15, 29}; {15, 42}; {17, 21}; {17, 24}; {17, 27}; {17, 38}; {18, 29}; {18, 39}; {18, 42}; {18, 47}; {21, 24}; {21, 27}; {21, 38}; {24, 27}; {24, 38}; {24, 41}; {29, 42}; {39, 42}; {42, 47}
3	{2, 9, 14}; {2, 9, 17}; {2, 9, 21}; {2, 9, 24}; {2, 9, 38}; {2, 14, 17}; {2, 14, 21}; {2, 14, 24}; {2, 14, 27}; {2, 14, 38}; {2, 14, 41}; {2, 17, 21}; {2, 17, 24}; {2, 21, 24}; {2, 24, 27}; {2, 24, 38}; {9, 14, 17}; {9, 14, 21}; {9, 14, 24}; {9, 14, 38}; {9, 17, 24}; {9, 21, 24}; {9, 24, 38}; {14, 17, 21}; {14, 17, 24}; {14, 21, 24}; {14, 24, 27}; {14, 24, 38}; {15, 18, 29}; {15, 18, 42}; {17, 21, 24}; {17, 24, 27}; {17, 24, 38}
4	{2, 9, 14, 17}; {2, 9, 14, 21}; {2, 9, 14, 24}; {2, 9, 14, 38}; {2, 9, 17, 24}; {2, 9, 21, 24}; {2, 14, 17, 21}; {2, 14, 17, 24}; {2, 14, 21, 24}; {2, 14, 24, 38}; {2, 17, 21, 24}; {9, 14, 17, 24}; {9, 14, 21, 24}; {14, 17, 21, 24}; {2, 9, 14, 17, 24}; {2, 9, 14, 21, 24}
5	{2, 14, 17, 21, 24}
6	∅

Table 8: The frequent itemsets identified by ECTPPI-Apriori for the Mushroom database.

Size of frequent itemset	The corresponding frequent itemsets
1	{1}; {2}; {3}; {4}; {5}
2	{1, 2}; {1, 3}; {1, 4}; {1, 5}; {2, 3}; {2, 4}; {2, 5}; {3, 4}; {3, 5}; {4, 5}
3	{1, 2, 3}; {1, 2, 5}; {1, 3, 5}; {2, 3, 4}; {2, 3, 5}
4	{1, 2, 3, 5}
5	∅

mushroom and has 23 attributes (fields). The first attribute is the poisonousness of the mushroom and the 2nd to the 23rd attributes are 22 characteristics of the mushrooms. Each of the attributes has 2 to 12 values. The frequent itemsets of these attribute values need to be found; that is, the problem is to find the attribute values which always appear together.

Initially, the database is preprocessed. Each attribute value is taken as a new attribute. In this way, each new attribute has only two values, yes or no. After preprocessing, each record has 118 attributes. ECTPPI-Apriori then can be used to discover the frequent itemsets. In this experiment, one itemset is a frequent itemset if it appears in more than 40 percent of all records; that is, the support count threshold is $k = 3250$ ($8124 \times 40\%$). The frequent itemsets obtained by ECTPPI-Apriori are listed in Table 8.

6. Conclusions

An improved Apriori algorithm, called ECTPPI-Apriori, is proposed for frequent itemsets mining. The algorithm uses a parallel mechanism in the ECPI tissue-like P system. The time complexity of ECTPPI-Apriori is improved to $O(t)$ compared to other parallel Apriori algorithms. Experimental results, using the Congressional Voting Records database and the Mushroom database, show that ECTPPI-Apriori performs well in frequent itemsets mining. The results give some hints to improve conventional algorithms by using the parallel mechanism of membrane computing models.

For further research, it is of interests to use some other interesting neural-like membrane computing models, such as the spiking neural P systems (SN P systems) [8], to improve the Apriori algorithm. SN P systems are inspired by the mechanism of the neurons that communicate by transmitting spikes. The cells in SN P systems are neurons that have only one type of objects called spikes. Zhang et al. [24, 25], Song et al. [26], and Zeng et al. [27] provided good examples. Also, some other data mining algorithms can be improved by using parallel evolution mechanisms and graph membrane structures, such as spectral clustering, support vector machines, and genetic algorithms [1].

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Project is supported by National Natural Science Foundation of China (nos. 61472231, 61502283, 61640201, 61602282, and ZR2016AQ21).

References

[1] J. Han, M. Kambr, and J. Pei, *Data Mining: Concepts and Techniques*, Elsevier, Amsterdam, Netherlands, 2012.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the International Conference on Very Large Data Bases*, vol. 1, pp. 487–499, September 1994.

[3] H. Yu, J. Wen, H. Wang, and J. Li, "An improved Apriori algorithm based on the boolean matrix and Hadoop," *Procedia Engineering*, vol. 15, no. 1, pp. 1827–1831, 2011.

[4] J. Li, F. Sun, X. Hu, and W. Wei, "A multi-GPU implementation of apriori algorithm for mining association rules in medical data," *ICIC Express Letters*, vol. 9, no. 5, pp. 1303–1310, 2015.

[5] N. Li, L. Zeng, Q. He, and Z. Shi, "Parallel implementation of apriori algorithm based on MapReduce," in *Proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD '12)*, pp. 236–241, Kyoto, Japan, August 2012.

[6] A. Ezhilvathani and K. Raja, "Implementation of parallel Apriori algorithm on Hadoop cluster," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 4, pp. 513–516, 2013.

[7] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[8] Gh. Paun, G. Rozenberg, and A. Salomaa, *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, UK, 2010.

[9] L. Pan, G. Păun, and B. Song, "Flat maximal parallelism in P systems with promoters," *Theoretical Computer Science*, vol. 623, pp. 83–91, 2016.

[10] B. Song, L. Pan, and M. J. Pérez-Jiménez, "Tissue P systems with protein on cells," *Fundamenta Informaticae*, vol. 144, no. 1, pp. 77–107, 2016.

[11] X. Zhang, L. Pan, and A. Păun, "On the universality of axon P systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 11, pp. 2816–2829, 2015.

[12] J. Wang, P. Shi, and H. Peng, "Membrane computing model for IIR filter design," *Information Sciences*, vol. 329, pp. 164–176, 2016.

[13] G. Singh and K. Deep, "A new membrane algorithm using the rules of Particle Swarm Optimization incorporated within the framework of cell-like P-systems to solve Sudoku," *Applied Soft Computing Journal*, vol. 45, pp. 27–39, 2016.

[14] G. Zhang, H. Rong, J. Cheng, and Y. Qin, "A Population-membrane-system-inspired evolutionary algorithm for distribution network reconfiguration," *Chinese Journal of Electronics*, vol. 23, no. 3, pp. 437–441, 2014.

[15] H. Peng, J. Wang, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "An unsupervised learning algorithm for membrane computing," *Information Sciences*, vol. 304, pp. 80–91, 2015.

[16] X. Zeng, L. Xu, X. Liu, and L. Pan, "On languages generated by spiking neural P systems with weights," *Information Sciences*, vol. 278, pp. 423–433, 2014.

[17] X. Liu, Z. Li, J. Liu, L. Liu, and X. Zeng, "Implementation of arithmetic operations with time-free spiking neural P systems," *IEEE Transactions on Nanobioscience*, vol. 14, no. 6, pp. 617–624, 2015.

[18] T. Song, P. Zheng, M. L. Dennis Wong, and X. Wang, "Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control," *Information Sciences*, vol. 372, pp. 380–391, 2016.

[19] L. Pan and G. Păun, "On parallel array P systems automata," in *Universality, Computation*, vol. 12, pp. 171–181, Springer International, New York, NY, USA, 2015.

[20] T. Song, H. Zheng, and J. He, "Solving vertex cover problem by tissue P systems with cell division," *Applied Mathematics and Information Sciences*, vol. 8, no. 1, pp. 333–337, 2014.

[21] Y. Zhao, X. Liu, and W. Wang, "ROCK clustering algorithm based on the P system with active membranes," *WSEAS Transactions on Computers*, vol. 13, pp. 289–299, 2014.

[22] C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[23] M. Lichman, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2013, http://archive.ics.uci.edu/ml.

[24] X. Zhang, B. Wang, and L. Pan, "Spiking neural P systems with a generalized use of rules," *Neural Computation*, vol. 26, no. 12, pp. 2925–2943, 2014.

[25] X. Zhang, X. Zeng, B. Luo, and L. Pan, "On some classes of sequential spiking neural P systems," *Neural Computation*, vol. 26, no. 5, pp. 974–997, 2014.

[26] T. Song, L. Pan, and Gh. Paun, "Asynchronous spiking neural P systems with local synchronization," *Information Sciences*, vol. 219, pp. 197–207, 2012.

[27] X. Zeng, X. Zhang, T. Song, and L. Pan, "Spiking neural P systems with thresholds," *Neural Computation*, vol. 26, no. 7, pp. 1340–1361, 2014.