A chain membrane model with application in cluster analysis

Yuzhen Zhao, Xiyu Liu* and Wenxing Sun

College of Business,
Shandong Normal University,
Jinan, 250014, China
Email: zhaoyuzhen_happy@126.com
Email: sdxyliu@163.com
Email: 373253360@qq.com
*Corresponding author

Abstract: Membrane computing is a kind of bio-inspired parallel distributed computing paradigm which can reduce computational complexity by the strategy of a space-time tradeoff. Traditionally, there are three kinds of membrane computing models (P systems) based on the tree and the graph topological structures. In this paper, a new P system with chain topological structure is proposed which is called the chain P systems. In the chain P systems, membranes, objects and rules are all in the form of chains which can store more information and therefore further improve the computational efficiency. The computational power and efficiency of the chain P systems are analysed. The graph clustering and the ROCK clustering algorithms based on the chain P systems are given as applications.

Keywords: membrane computing; membrane model; chain P system; computational power; computational efficiency; ROCK clustering; graph clustering.

Reference to this paper should be made as follows: Zhao, Y., Liu, X. and Sun, W. (2019) 'A chain membrane model with application in cluster analysis', *Int. J. Adaptive and Innovative Systems*, Vol. 2, No. 4, pp.324–348.

Biographical notes: Yuzhen Zhao received her PhD in Management Science from Shandong Normal University, China. She is a Lecturer, and the Master Supervisor of Business School, Shandong Normal University. Her research interests include membrane computing and data mining.

Xiyu Liu received his PhD in Mathematical Sciences from Shandong University. He is a Professor, the Doctorial Supervisor, and the President of Business School, Shandong Normal University. His research interests include membrane computing and data mining.

Wenxing Sun received his BS in Physics from Shandong Normal University, China. He is an Associate Professor of Demission-Retirement Affair Office, Shandong Normal University. His research interests include membrane computing and data mining.

1 Introduction

Biological systems, such as cells, tissues, and human brains, have deep computational intelligences. Biologically inspired computing, or bio-inspired computing in short, focuses on abstracting computing ideas from biological systems to construct computing models and algorithms. Membrane computing is a lately initiated research area of bio-inspired computing in 2002, which seeks to discover new computational facility from the dynamics of cells, particularly of the cellular membranes (Păun et al., 2010). The new models are distributed and parallel bio-inspired computing facilities, usually called P systems. There are three mainly investigated P systems, cell-like P systems, tissue P systems, and neural-like P systems (and their variants, see, e.g., Cabarle et al., 2017; Zeng et al., 2014; Song and Wang, 2015; Song et al., 2016; Song and Pan, 2015; Zeng et al., 2009; Zhang et al., 2014a, 2014b; Peng et al., 2017; Zhao et al., 2016; Liu et al., 2018; Song et al., 2019a, 2017). It has been proved that many P systems are universal, that is, they are able to do what a Turing machine can do efficiently (Song et al., 2019b; Wang et al., 2016; Zeng et al., 2014; Zhang et al., 2017). The parallel evolution mechanism of variants of P systems has been found to perform well in doing computation, even solving computational hard problems. Therefore, P systems have been introduced to many fields gradually (Wang et al., 2019a; Ju et al., 2016; Liu et al., 2015a; Liu et al., 2017; Liu and Xue, 2017; Wang et al., 2019b).

Researchers pay close attention to the computational efficiency of P systems, especially the judgment whether NP-complete problems have solutions or not in feasible time (Song et al., 2014a, 2014b; Pan et al., 2011; Wang et al., 2011). In previous studies, if a NP-complete problem has a solution, a specific object is output to show that; otherwise, another object or nothing is output to show that. However, the solutions need to be found out in many situations. For instance, the register allocation problem is an application of SAT problem. This problem aims to build a mapping relationship between the virtual registers and the physical registers, and realises the rational utilisation of physical register resources. In this case, we need to judge whether a good solution exists, while searching the solution by distributing the physical register resources according to the solution. In applications, many problems can be transformed into graph colouring problems, which is equivalent to SAT problems. To solve these problems, exact solutions are also essential.

For this purpose, the chain thought in DNA computing is introduced into the P systems and a new variant of P systems called the chain P system is proposed in this paper. In the chain P systems, the concepts of membranes, objects and rules are expended to chains, and the operations of crossover, mutation and so forth are transplanted. The chain P systems can record more information and realise the same function with less computing resources. Each chain object represents one solution, and the redundant objects can be removed from the system which can be used to remove the wrong results. Objects which represent all possible results are output. Chain P systems which give uniform solutions to SAT problem and Hamilton Path problem (HPP), which work in a deterministic way, not using the membrane division rules, are constructed as examples in this paper.

The application of P systems is another research hotspot in the field. Although P systems have been used in many fields, the intensive coupling between membrane computing and the optimisation is still an open problem. For this purpose, two

applications in clustering: the graph clustering and the ROCK clustering based on the chain P systems are presented. These two applications also show the advantage of the chain P systems.

The contributions of this paper focus on two folds: for membrane computing, a new variant of P systems is proposed which can decrease the computing resources; for clustering analysis, the new algorithms combined with the chain P systems are presented which can reduce the time complexity of data processing and satisfy the requirement of improving the processing speed of the big data.

The paper is organised as follows. The chain P system is proposed in Section 2. Section 3 analyses the computational power and efficiency of the chain P systems. Section 4 and Section 5 give the graph clustering algorithm and the ROCK algorithm based on the chain P systems. Conclusions are given in Section 6.

2 The chain P systems

Several concepts are defined firstly. If no other membranes are in a membrane, this membrane is called an elementary membrane. The ordered chain consisting of several linked membranes is called a chain membrane. Each membrane in the chain membrane is called a cell membrane. Each symbol in the alphabet is called an elementary object. The ordered object consisting of several linked symbols is called a chain object.

The formal description of the chain P system is as follows.

$$\Pi = \left(O, \mu, w_1, w_2, \ldots, w_m, R, \rho, i_{in}, i_{out}\right)$$

- O is the alphabet which includes all elementary objects of the system.
- μ is the membrane structure.
- $w_j (1 \le j \le m)$ is the initial chain objects in membrane j, and object λ shows no object is in the current membrane.
- R is the set of the chain rules. A chain rule is composed of n sub-rules with the form of $r_j = \{r_{j.1}, r_{j.2}, \ldots, r_{j.n}\}$ which is executed from left to right. If a certain sub-rule can not be executed, the execution of this chain rule is end, and the remaining sub-rules are no longer executed.
- ρ defines the partial ordering relationship of the rules, i.e., rules with higher orders are executed with higher priority.
- i_{in} is the label of the membrane where the objects are put into.
- i_{out} is the label of the membrane where the computational result is placed.

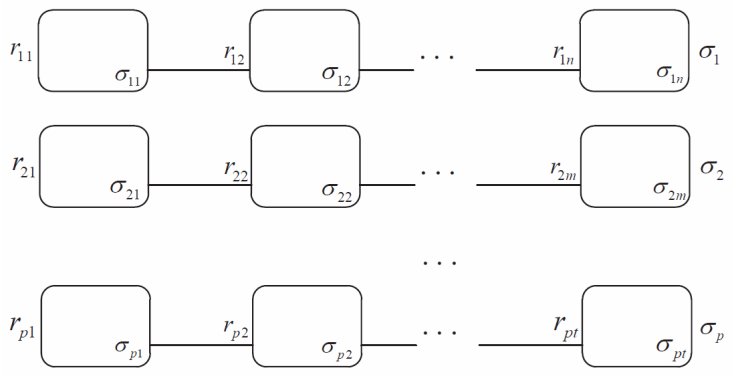
  If $i_{out} = 0$, the computational result is reserved in the environment.

2.1 The chain membranes

The membranes exist in the form of chain $\sigma_i = r_{i1} * \sigma_{i1} \otimes r_{i2} * \sigma_{i2} \otimes \ldots \otimes r_{in} * \sigma_{in}$ (the * can be omitted if there is no ambiguity). Where, $\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{in}$ represent the cell membranes in the chain membrane σ_i, $r_{i1}, r_{i2}, \ldots, r_{in}$ are integers, $r_{ij} * \sigma_{ij}$ represents this chain membrane σ_i contains $|r_{ij}|$ copies of σ_{ij}. If $r_{ij} > 0$, σ_{ij} carries positive charge;

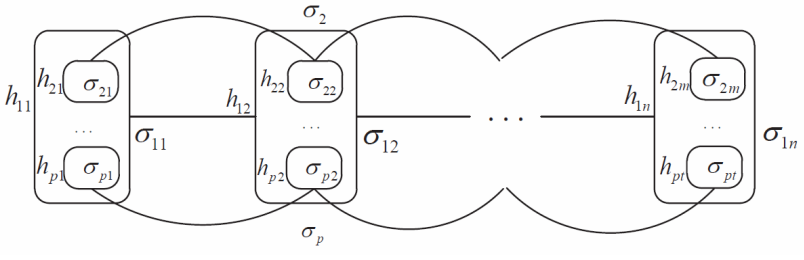
otherwise, σ_{ij} carries negative charge. Figure 1 shows an example of the chain membranes.

Figure 1 p chain membranes



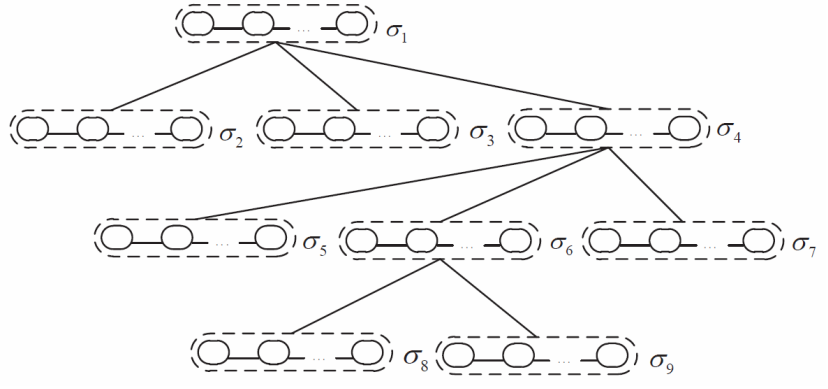
Notes: Where the chain membrane σ_1 contains n cell membranes, the chain membrane σ_2 contains m cell membranes, …, and the chain membrane σ_p contains t cell membranes.

Figure 2 The chain membrane σ_1 contains n cell membranes σ_{11}, σ_{12}, …, σ_{1n}



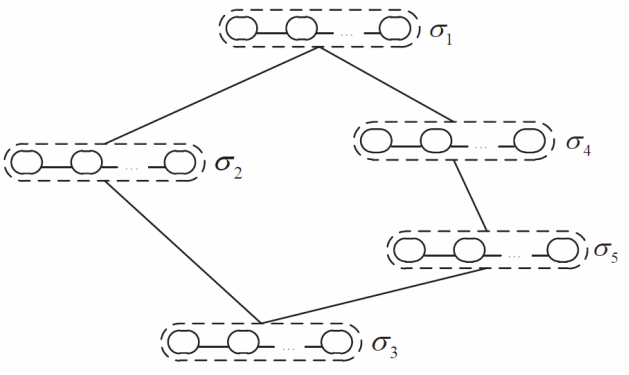
Notes: σ_1 contains $p - 1$ child chain membranes σ_2, σ_3, …, σ_p, then each child chain membrane $\sigma_i (2 \le i \le p)$ contains n cell membranes σ_{i1}, σ_{i2}, …, σ_{in}, and the objects in σ_{1j} are the union of all objects in cell membranes σ_{2j}, σ_{3j}, …, σ_{pj}.

Figure 3 The whole structure of the tree topological P systems



The whole system has two topological structures: tree and graph. In the tree topological P systems, the relationship among the chain membranes is inclusion, i.e., the cell membrane of the parent chain membrane contains the corresponding cell membranes of the child chain membranes. Figure 2 and Figure 3 show examples of the tree topological P systems structure. Figure 4 gives an example of the graph topological P systems structure.

Figure 4 The whole structure of the graph topological P systems



2.2 The chain objects

The objects exist in the form of chain $a = r_1 * x_1 \otimes r_2 * x_2 \otimes \ldots \otimes r_n \otimes x_n$ (the * can be omitted if there is no ambiguity). Where, $r_j * x_j$ represents this chain object a contains $|r_j|$ copies of x_j. If $r_j > 0$, x_j is a positive object; otherwise, x_j is a negative object.

The structured objects can store a large amount of information. For instance, to calculate the value of 201 + 12, two chain objects $a_1 = 2x_3 \otimes x_1$ and $a_2 = x_2 \otimes 2x_1$ are constructed. Each object represents a number. By two rules $a_3 = a_1 + a_2$ and $10x_i \rightarrow x_{i+1}$, $i \geq 1$, the new object $a_3 = 2x_3 \otimes 1x_2 \otimes 3x_1$ is obtained which means the result is 213. If the traditional unstructured objects are used, 213 objects are needed. The structured objects can improve the computational efficiency and reduce the space complexity.

2.3 The chain rules

There are two types of rules: rules on the chain objects and rules on the chain membranes. The traditional rules can also be used in the chain P systems, and several new types of rules are designed.

2.3.1 Rules on objects

For arbitrary two chain objects $a_1 = r_1 * x_1 \otimes r_2 * x_2 \otimes \ldots \otimes r_n * x_n$ and $a_2 = h_1 * x_1 \otimes h_2 * x_2 \otimes \ldots \otimes h_n * x_n$:

- *Object addition rule* $a_3 = a_1 + a_2$:

  The sum of a_1 and a_2 is $a_1 + a_2 = (r_1 + h_1)x_1 \otimes (r_2 + h_2)x_2 \otimes (r_n + h_n)x_n$, a_3 is also a chain object.

- *Object subtraction rule* $a_3 = a_1 - a_2$:

  The difference of a1 and a_{i2} is $a_1 - a_2 = (r_1 - h_1) * x_1 \otimes (r_2 - h_2) * x_2 \otimes \ldots \otimes (r_n - h_n) * x_n$, a_3 is also a chain object.

- *Object crossover rule* $(a_1\$_t a_2)$:

  Given a cross point t, cross $a_1 = r_1 * x_1 \otimes \ldots \otimes r_t * x_t \otimes r_{t+1} * x_{t+1} \otimes \ldots \otimes r_n * x_n$ and $a_2 = h_1 * x_1 \otimes \ldots \otimes h_t * x_t \otimes h_{t+1} * x_{t+1} \otimes \ldots \otimes h_n * x_n$, the obtained objects are $a_1 = r_1 * x_1 \otimes \ldots \otimes r_t * x_t \otimes h_{t+1} * x_{t+1} \otimes \ldots \otimes h_n * x_n$ and $a_2 = h_1 * x_1 \otimes \ldots \otimes h_t * x_t \otimes r_{t+1} * x_{t+1} \otimes \ldots \otimes r_n * x_n$.

- *Object variation rule* a: $r_t * x_t \rightarrow r'_t * x'_t$:

  Given a variation point t, a varies to $r_1 * x_1 \otimes r_2 * x_2 \otimes \ldots \otimes r'_t * x'_t \otimes \ldots \otimes r_n * x_n$. Note that x_t and x'_t can be the same one.

- *Extended object variation rule* $a : r_{t_1} * x_{t_1} \otimes r_{t_2} * x_{t_2} \otimes \ldots \otimes r_{t_m} * x_{t_m} \rightarrow r'_{t_1} * x'_{t_1} \otimes r'_{t_2} * x'_{t_2} \otimes \ldots \otimes r'_{t_m} * x'_{t_m}$:

  Given $m (m \geq 1)$ variation points $t_1, t_2, \ldots, t_m$, a varies to $r_t * x_t \otimes \ldots \otimes r'_{t_1} * x'_{t_1} \otimes \ldots \otimes r'_{t_2} * x'_{t_2} \otimes \ldots \otimes r'_{t_m} * x'_{t_m} \otimes \ldots \otimes r_n * x_n$.

2.3.2 Rules on membranes

For chain membrane $\sigma_{q+1} = r_{(q+1)1} * \sigma_{(q+1)1} \otimes r_{(q+1)2} * \sigma_{(q+1)2} \otimes \ldots \otimes r_{(q+1)n} * \sigma_{(q+1)n}$ and its child membrane $\sigma_q = r_{q1} * \sigma_{q1} \otimes r_{q2} * \sigma_{q2} \otimes \ldots \otimes r_{qn} * \sigma_{qn}$:

- *Parent-child communication rule* $[\sigma_q, \sigma_{q+1}]$: $(a, up); (b', in) \rightarrow (b, down); (a', in)$ or $[\sigma_{q+1}, \sigma_q]$: $(b, down); (a', in) \rightarrow (a, up); (b', in)$, $a, a', b, b' \in O^*$:

  Object a in σq evolves to a' and enters its parent membrane σ_{q+1}, at the same time, b in σ_{q+1} evolves to b' and enters its child membrane σ_q. Note that for the skin membrane, its parent membrane is the environment.

- *Extended parent-child communication rule* $[\sigma_{q1}, \ldots, \sigma_{qn}, \sigma_{q+1}]$: $(a_1, \ldots, a_n, up); (b_1, \ldots, b_n, in) \rightarrow (b, down); (a'_1, \ldots, a'_n, in)$ or $[\sigma_{q+1}, \sigma_{q1}, \ldots, \sigma_{qn}]$: $(b, down); (a'_1, \ldots, a'_n, in) \rightarrow (a_1, \ldots, a_n, up); (b_1, \ldots, b_n, in), a_1, \ldots, a_n, a'_1, \ldots, a'_n, b, b_1, \ldots, b_n \in O$:

  Objects $a_1, \ldots,$ an in $\sigma_{q1}, \ldots, \sigma_{qn}$ evolve to $a'_1, \ldots, a'_n$ and enter their parent membrane σ_{q+1}, at the same time, b in σ_{q+1} evolves to $b_1, \ldots, b_n$ and enters its child membranes $\sigma_{q1}, \ldots, \sigma_{qn}$.

2.4 The system computational process

Rules are executed in non-deterministic maximally parallel manner in each membrane, i.e., at any step, if more than one rule can be executed but the objects in the membrane can only support some of them, a maximal number of rules will be executed. Each P system contains a global clock as the timer, and the execution time of one rule is set to a time unit. 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 membrane. Because objects in a P system evolve in maximally parallel, the system computes very efficiently.

3 Computational power and efficiency analysis

3.1 Computational power analysis

The computational power of the chain P systems is analysed by simulating the register machine.

It has been proved that a register machine with three registers can generate the set of the length of the recursively enumerable language. Therefore, a register machine with three registers $M = (m, H, l_0, l_h, I)$ is considered. The generated number stores in register 1 which number will not decrease. Registers 2 and 3 are empty when the register machine halts. The following chain P system is constructed to simulate M.

$$\Pi_r = \left(O, \mu, w_1, w_2, w_3, w_4, R, \rho, i_{out}\right)$$

where

- $O = \{a\} \cup \{l \in H\}$
- $\mu = [[\ ]_1[\ ]_2[\ ]_3]_4$
- $w_i = \lambda,\ 1 \le i \le 3,\ w_4 = l_0$
- $\rho = \{r_1 = r_2, r_3 > r_4\}$
- $i_{out} = 1$
- R.

In Π_r, membranes 1, 2 and 3 are corresponding to registers 1, 2 and 3, and the number of objects a in a membrane represents the value of the corresponding register. The instructions in the register machine are simulated by rules.

For each add instruction l_i: $(ADD(r), l_j, l_k)$, r = 1, 2, 3, the following rules are introduced:

$$r_1 : \left[\sigma_4, \sigma_r\right] : \left(l_i, down\right)\left(l_j, in\right) \rightarrow (\lambda, up)(a, in)$$

$$r_2 : \left[\sigma_4, \sigma_r\right] : \left(l_i, down\right)\left(l_k, in\right) \rightarrow (\lambda, up)(a, in)$$

The two rules can simulate the add instruction. At one step, r_1 or r_2 is chosen non-deterministically to execute. If r_1 is chosen, l_i evolves to a and enters its child membrane σ_r, and l_j enters σ_4 at the same step. Through this rule, the number of a in membrane r increases by 1, and the next instruction changes to l_j. Similarly, if r_2 is chosen, the number of a in membrane r increases by 1, and the next instruction changes to l_k.

For each sub instruction l_i: $(SUB(r), l_j, l_k)$, r = 1, 2, 3, the following rules are introduced:

$$r_3 : \left[\sigma_4, \sigma_r\right] : \left(l_i, down\right)\left(l_j, in\right) \rightarrow (a, up)(\lambda, in)$$

$$r_4 : [\sigma_4, \sigma_r] : (l_i, down)(l_k, in) \to (\lambda, up)(\lambda, in)$$

The two rules can simulate the sub instruction. At one step, if the number of a in membrane r is not 0, r_3 executes. Object l_i is dissolved, and a evolves to l_j and enters σ_4 at the same step. Through this rule, the number of a decreases by 1, and the next instruction changes to l_j. If the number of a in membrane r is 0, r_3 can not execute, and r_4 obtains the chance to execute. Object l_i is dissolved, and l_k enters σ_4 at the same step. The number of a in membrane r is still 0, and the next instruction changes to l_k.

The halt instruction is simulated when l_h appears in membrane 4.

The chain P system Π_r simulates the register machine M, therefore, $N(M) = N(\Pi_r)$.

3.2 Computational efficiency analysis

Uniform solutions to two NP problems (SAT problem and HPP) working in a deterministic way are used to show the computational efficiency of the chain P systems without the expansion of membranes.

3.2.1 SAT problem

SAT (the satisfiability of conjunctive normal form expression) problem is one of the most typical NP-complete problems. For a Boolean variable set $X = \{x_1, x_2, \ldots, x_n\}$, a literal l_i is x_i or $\neg x_i$ for $1 \le i \le n$. A clause C_i is a disjunction of literals $C_i = l_{n_1} \vee l_{n_2} \vee \ldots \vee l_{n_r}$, $1 \le r \le n$. A conjunctive normal form (CNF, for short) is a conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_m$. An assignment is a mapping $X \to \{0, 1\}$ from each variable x_i to its value (value 1 represents true and value 0 represents false.). For example, $X = \{x_1, x_2, x_3\}$, the conjunctive normal form is $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_3)$. The $x_1 \vee \neg x_2$ and $x_1 \vee x_3$ are the two clauses. The first clause contains two literals x_1 and $\neg x_2$, and the second clause contains two literals x_1 and x_3. If an assignment of $x_1, x_2, \ldots, x_n$ can be found, which makes at least one literal true in each clause and then makes all m clauses true, this SAT problem is satisfiable. Otherwise, this SAT problem is unsatisfiable. In the above example, let $x_1 = x_2 = x_3 = 1$, the value of the conjunctive normal form is $(1 \vee 0) \wedge (1 \vee 0) = 1 \wedge 1 = 1$. Therefore, the SAT problem is satisfiable.

The formal definition of SAT problem is as follows:

- Problem 1 – NAME: SAT.

  Instance: A set of clauses $C = \{C_1, C_2, \ldots, C_m\}$, which is built on a Boolean variable set $X = \{x_1, x_2, \ldots, x_n\}$.

  Question: Is there an assignment of Boolean variables $x_1, x_2, \ldots, x_n$ that can make the values of all clauses true?

$SAT(n, m)$ denotes the set of all instances of the SAT problem having n variables and m clauses. In this section, a uniform solution working in a deterministic way is constructed by the chain P systems, which can solve all $SAT(n, m)$ problems in linear time.

The instance parameters need to enter a chain P system, therefore the CNF needs to be encoded as object $a = d_{11} * a_{11} \otimes d_{12} * a_{12} \otimes \ldots \otimes d_{1n} * a_{1n} * d_{21} * a_{21} \otimes d_{22} * a_{22} \otimes \ldots \otimes d_{2n} * a_{2n} \otimes \ldots \otimes d_{m1} * a_{m1} \otimes d_{m2} * a_{m2} \otimes \ldots \otimes d_{mn} * a_{mn}$. The coefficient d_{ij} of a_{ij} has three values: 0, 1 and –1. The value 0 represents the i^{th} clause does not contain the literal

x_j or $\neg x_j$, and the value 1 (resp. –1) represents the i^{th} clause contains the literal x_j (resp. $\neg x_j$).

The formal definition of the chain P system for $SAT(n, m)$ problems is as follows.

$$\Pi_{SAT} = (O, \mu, w, R, \rho, i_{in}, i_{out})$$

where

- $O = \{a_{ji}, v_i, s_j, \varphi\}, 1 \le i \le n, 1 \le j \le m$
- $\mu = [_1]_1$
- $w = \lambda$
- $\rho = \{r_1 > r_2\}$
- $i_{in} = 1$
- $i_{out} = 1$
- R.

Other objects used in Π_{SAT} are explained. Object $v = h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$ shows the assignment of all Boolean variables. The coefficient h_i of v_i has two values: 1 and –1. The value 1 (resp. –1) represents the value of the variable x_i is set to 1 (resp. 0). Object $s = h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \otimes d_{11} * a_{11} \otimes d_{12} * a_{12} \otimes \ldots \otimes d_{1n} * a_{1n} \otimes d_{21} * a_{21} \otimes d_{22} * a_{22} \otimes \ldots \otimes d_{2n} * a_{2n} \otimes \ldots \otimes d_{m1} * a_{m1} \otimes d_{m2} * a_{m2} \otimes \ldots \otimes d_{mn} * a_{mn} \otimes t_1 * s_1 \otimes t_2 * s_2 \otimes \ldots \otimes t_m * s_m$ uses to obtain the satisfiable assignment. Object φ has only one elementary object which is used to control the computational process.

$$r_1 : \{r_{1.1}, r_{1.2}, r_{1.3}, r_{1.4}\}$$

$r_{1.1.}$:

$$(h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n)_{a\varphi} \to h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$$
$$\otimes (h_1 + d_{11}) * a_{11} \otimes (h_2 + d_{12}) * a_{12} \otimes \ldots \otimes (h_n + d_{1n}) * a_{1n} \otimes (h_1 + d_{21}) * a_{21}$$
$$\otimes (h_2 + d_{22}) * a_{22} \otimes \ldots \otimes (h_n + d_{2n}) * a_{2n} \otimes \ldots \otimes (h_1 + d_{m1}) * a_{m1} \otimes (h_2 + d_{m2})$$
$$* a_{m2} \otimes \ldots \otimes (h_n + d_{nn}) * a_{mn} \otimes 0 * s_1 \otimes 0 * s_2 \otimes \ldots \otimes 0 * s_m$$

$r_{1.2}$:

$$s : d * a_{ij} \otimes 0 * a_{ij} \otimes 1 * s_1 (d = \pm 2)$$

$r_{1.3}$:

$$h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \otimes d_{11} * a_{11} \otimes d_{12} * a_{12} \otimes \ldots \otimes d_{1n} * a_{1n}$$
$$\otimes d_{21} * a_{21} \otimes d_{22} * a_{22} \otimes \ldots \otimes d_{2n} * a_{2n} \otimes \ldots \otimes d_{m1} * a_{m1} \otimes d_{m2} * a_{m2} \otimes \ldots$$
$$\otimes d_{mn} * a_{mn} \otimes t_1 * s_1 \otimes t_2 * s_2 \otimes \ldots \otimes t_m * s_m \to \lambda (t_1 * \cdots * t_m = 0)$$

$r_{1.4}$:

$$h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \otimes d_{11} * a_{11} \otimes d_{12} * a_{12} \otimes \ldots \otimes d_{1n} * a_{1n}$$
$$\otimes d_{21} * a_{21} \otimes d_{22} * a_{22} \otimes \ldots \otimes d_{2n} * a_{2n} \otimes \ldots \otimes d_{m1} * a_{m1} \otimes d_{m2} * a_{m2} \otimes \ldots$$
$$\otimes d_{mn} * a_{mn} \otimes t_1 * s_1 \otimes t_2 * s_2 \otimes \ldots \otimes t_m * s_m \to h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$$

$r_2 : \{r_{2.1}, r_{2.2}, \ldots, r_{2.n}\}$

$r_{2.1}$:
$h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \rightarrow h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n, -h_1 * v_1$
$\otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \left(h_i = \pm 1, 1 \le i \le n\right)$

$r_{2.2}$:
$h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \rightarrow h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n, h_1 * v_1 - h_2$
$* v_2 \otimes \ldots \otimes h_n * v_n \left(h_i = \pm 1, 1 \le i \le n\right)$
…

$r_{2.n}$:
$h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n \rightarrow h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n, h_1 * v_1$
$\otimes h_2 * v_2 \otimes \ldots - h_n * v_n, \varphi\left(h_i = \pm 1, 1 \le i \le n\right)$

Computation begins when $v = v_1 \otimes v_2 \otimes \ldots \otimes v_n$ and a enter membrane 1. Object $v = v_1 \otimes v_2 \otimes \ldots \otimes v_n$ means the values of all x_i are set to 1 in the given assignment.

Generating stage

The priority of r_1 is higher than that of r_2, however, the promoter φ is not in the membrane in the beginning, therefore, r_2 executes firstly. The chain rule r_2 contains n sub-rules. At step 1, $r_{2.1}$ divides the initial v into $v_1 = v_1 v_2 \otimes \ldots \otimes v_n$ and $v_2 = -v_1 \otimes v_2 \otimes \ldots \otimes v_n$ showing that the value of x_1 can be 1 or –1 in an assignment. Sub-rule $r_{2.2}$ divides v_1 into $v_{11} = v_1 \otimes v_2 \otimes \ldots \otimes v_n$ and $v_{12} = v_1 \otimes -v_2 \otimes \ldots \otimes v_n$, and divides v_2 into $v_2 = -v_1 \otimes v_2 \otimes \ldots \otimes v_n$ and $v_2 = -v_1 \otimes -v_2 \otimes \ldots \otimes v_n$ at step 2. This shows the value of x_2 can be 1 or –1 in an assignment. And so on, until $r_{2.n}$ divides each $h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$ into $h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$ and $h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes -h_n * v_n$ showing that the value of x_n can be 1 or –1 in an assignment. Rule r_2 generates 2^n chain objects v showing all combinations of the values of x_i. An object φ is generated by $r_{2.n}$ showing the generation stage is over.

Checking stage

With φ, r_1 begins to execute. Each v evolves to s by $r_{1.1}$. In the generated s, $h_1 * v_1 \otimes h_2 * v_2 \otimes \ldots \otimes h_n * v_n$ shows the assignment of x_i, $(h_1 + d_{21}) * a_{21} \otimes (h_2 + d_{22}) * a_{22} \otimes \ldots \otimes (h_n + d_{2n}) * a_{2n} \otimes \ldots \otimes (h_1 + d_{m1}) * a_{m1} \otimes (h_2 + d_{m2}) * a_{m2} \otimes \ldots \otimes (h_n + d_{nn}) * a_{mn}$ shows the relationship between the assignment and the CNF, and $0 * s_1 \otimes 0 * s_2 \otimes \ldots \otimes 0 * s_m$ shows the value of each clause in the initial. The coefficient of v_i has two values: 1 and –1, and the coefficient of a_{ij} has three values: 0, 1 and –1. Therefore, the relationship between the values of the two coefficients and the satisfiability of the corresponding literal is as follows:

- $h_j = 1$, $d_{ij} = 1$, $h_j + d_{ij} = 2$, the value of literal x_j in clause C_i is true
- $h_j = 1$, $d_{ij} = 0$, $h_j + d_{ij} = 1$, the clause C_i dose not contain the literal x_j
- $h_j = 1$, $d_{ij} = -1$, $h_j + d_{ij} = 0$, the value of literal x_j in clause C_i is false

- $h_j = -1$, $d_{ij} = 1$, $h_j + d_{ij} = 0$, the value of literal x_j in clause C_i is false
- $h_j = -1$, $d_{ij} = 0$, $h_j + d_{ij} = -1$, the clause C_i dose not contain the literal x_j
- $h_j = -1$, $d_{ij} = -1$, $h_j + d_{ij} = -2$, the value of literal x_j in clause C_i is true.

Therefore, the certain literal x_j in clause C_i is true when the coefficient value of a_{ij} is 2 or –2; false otherwise.

Sub-rule $r_{1.2}$ changes the coefficient value of s_i from 0 to 1 if the coefficient value of a_{ij} is 2 or –2 to show that C_i is true. A SAT problem is satisfiable when all m clauses are true, i.e., s with the coefficient values of all s_i equal to 1 needs to be found. Sub-rule $r_{1.3}$ dissolves s with the coefficient values of several s_i equal to 0, and $r_{1.4}$ evolves s to v abstracting the assignment.

Computational resources and complexity

- the initial chain objects number: $2 \in \Theta(1)$
- the membranes number: $1 \in \Theta(1)$
- the number of rules: $2 \in \Theta(1)$
- the number of sub-rules: $n + 4 \in \Theta(n)$
- the computational steps: $n + 4 \in \Theta(n)$.

The chain P system solves the SAT problem in linear time with linear computational resources, and all solutions can be obtained through it. The traditional P systems solve the SAT problem by membrane division rules, while the chain P system realises the computational process without the expansion of membranes (Song et al., 2015).

3.2.2 Hamilton path problem

HPP is one of the most typical NP-complete problems. The formal definition of HPP is as follows.

- Problem 2 – NAME: HPP.

  Instance: A graph $\gamma = (N, E)$, where $N = \{a_1, a_2, \cdots, a_n\}$ is the set of nodes and $E = \{e'_{ij}, 1 \le i, j \le n\}$ is the set of edges.

  Question: Is there a Hamilton path in γ which length is n visiting each node from γ exactly once?

HPP(n) denotes the set of all instances of the HPP having n nodes. In this section, a uniform solution working in a deterministic way is constructed by the chain P systems, which can solve all *HPP*(n) in linear time.

The instance parameters need to enter a chain P system, therefore the edges in γ need to be encoded as object e'_{j1j2} which represents there is an edge between a_{j1} and a_{j2}.

The formal definition of the chain P system for *HPP*(n) problems is as follows:

$$\Pi_{HPP} = \left(O, \mu, w, R, i_{in}, i_{out}\right)$$

- $O = \{\beta_q, V_{ij}, \alpha_{it}, e_{j1j2}, 1 \le q \le n + 1, 1 \le i, j \le n, 0 \le t \le n\}$
- $\mu = [_1[_2]_2]_1$

- $w_1 = w_2 = \lambda$
- $i_{in} = 1$
- $i_{out} = 2$
- R.

Other objects used in Π_{HPP} are explained. Object $v = \beta_j \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{nt_n}$ is constructed by $2n + 1$ components, where β_j represents the j^{th} point in a path is going to be chosen, V_{ji_j} represents the j^{th} point in a path is a_{ij}, α_{jt_j} represents the occurrence number of a_j in a path is t_j. Object $e_{j_1 j_2}$ represents that there is not an edge between a_{j_1} and a_{j_2}. Object $s = V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n}$ represents the final Hamilton path, where V_{ji_j} represents the j^{th} point in a path is a_{i_j}.

$$R_1 : r_1 : \{r_{1.1}, r_{1.2}, \ldots, r_{1.5}\}$$

$$r_{1.1} :$$
$$\beta_j \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ji_j} \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{nt_n} \to \beta_{j+1}$$
$$\otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes B_{j1} \ldots \otimes V_{ni_n} \otimes \alpha_{1(t_1+1)} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{nt_n}, \beta_{j+1}$$
$$\otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes B_{j2} \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2(t_2+1)} \otimes \ldots \otimes \alpha_{nt_n},$$
$$\ldots$$
$$\beta_{j+1} \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{jn} \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{n(t_n+1)}$$

$$r_{1.2} :$$
$$()_{\neg e'_{j1j2}} \to e_{j1j2}$$

$$r_{1.3} :$$
$$\left(\beta_{n+1} \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{nt_n}\right)_{e_{ji_j i_{j+1}}} \to \lambda$$

$$r_{1.4} :$$
$$\beta_{n+1} \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{i0} \ldots \otimes \alpha_{nt_n} \to \lambda$$

$$r_{1.5} :$$
$$[\sigma_1, \sigma_2] : (\beta_{n+1} \otimes V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes \alpha_{1t_1} \otimes \alpha_{2t_2} \otimes \ldots \otimes \alpha_{nt_n}, down); (\lambda, in)$$
$$\to (\lambda, in); (V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n}, in)$$
$$(1 \le i_1, i_2, \ldots, i_n, j \le n, 0 \le t_1, t_2, \ldots, t_n \le n)$$

$$R_2 :$$
$$\lambda$$

Computation begins when $v = \beta_1 \otimes V_{10} \otimes V_{20} \otimes \ldots \otimes V_{n0} \otimes \alpha_{10} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n0}$ and $e_{j_i j_2}$ enter membrane 1. Object $v = \beta_1 \otimes V_{10} \otimes V_{20} \otimes \ldots \otimes V_{n0} \otimes \alpha_{10} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n0}$ means each point in a path is not chosen and the occurrence number of each point is zero in the initial. The points in a path begin to be chosen from the first point.

Generating stage

The chain rule r_1 contains four sub-rules. Sub-rule $r_{1.1}$ executes n times to generate all candidate paths. At step 1, the initial v is divided into $v_1 = \beta_2 \otimes V_{11} \otimes V_{20} \otimes \ldots \otimes V_{n0} \otimes \alpha_{11} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n0}$, $v_2 = \beta_2 \otimes V_{12} \otimes V_{20} \otimes \ldots \otimes V_{n0} \otimes \alpha_{10} \otimes \alpha_{21} \otimes \ldots \otimes \alpha_{n0}$, $\cdots$, $v_n = \beta_2 \otimes V_{1n} \otimes V_{20} \otimes \ldots \otimes V_{n0} \otimes \alpha_{10} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n1}$ showing that the first point in a path can be $a_1, a_2, \cdots, a_n$. At step 2, v_1 is divided into $v_{11} = \beta_3 \otimes V_{11} \otimes V_{21} \otimes \ldots \otimes V_{n0} \otimes \alpha_{12} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n0}$, $v_{12} = \beta_3 \otimes V_{11} \otimes V_{22} \otimes \ldots \otimes V_{n0} \otimes \alpha_{11} \otimes \alpha_{21} \otimes \ldots \otimes \alpha_{n0}$, …, $v_{1n} = \beta_3 \otimes V_{11} \otimes V_{2n} \otimes \ldots \otimes V_{n0} \otimes \alpha_{11} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n1}$ showing that the second point in a path can be $a_1, a_2, \ldots, a_n$. Similarly, v_i is divided into $v_{i1} = \beta_3 \otimes V_{11} \otimes \ldots \otimes V_{i1} \otimes \ldots \otimes V_{n0} \otimes \alpha_{12} \otimes \ldots \otimes \alpha_{n0}$, $v_{i2} = \beta_3 \otimes V_{11} \otimes \ldots \otimes V_{i2} \otimes \ldots \otimes V_{n0} \otimes \alpha_{11} \otimes \alpha_{21} \otimes \ldots \otimes \alpha_{n0}$, $\cdots$, $v_{in} = \beta_3 \otimes V_{11} \otimes \ldots \otimes V_{in} \otimes \ldots \otimes V_{n0} \otimes \alpha_{11} \otimes \alpha_{20} \otimes \ldots \otimes \alpha_{n1}$. And so on, until the n^{th} point in a path is chosen. Sub-rule $r_{1.1}$ generates n^n chain objects v showing all combinations of paths.

Checking stage

Sub-rule $r_{1.2}$ generates $e_{j_1 j_2}$ which shows the edges not belonging to the graph. Sub-rule $r_{1.3}$ dissolves v which contains the edges not belonging to the graph. Sub-rule $r_{1.4}$ dissolves v which does not contain all nodes. In v, α_{it_i} shows the occurrence number of a_i in this path is t_i. If a certain t_i is 0, a_i dose not in this path and the corresponding v is dissolved.

Output stage

Sub-rule $r_{1.5}$ deals with the remaining v which contains the final solutions. The corresponding s enters membrane 2 which stores the final results.

Computational resources and complexity

- the initial objects number: $1 + m \in \Theta(m)$
- the membranes number: $2 \in \Theta(1)$
- the number of rules: $1 \in \Theta(1)$
- the number of sub-rules: $5 \in \Theta(1)$
- the computational steps: $n + 3 \in \Theta(n)$.

where m is number of the edges in the graph. The chain P system solves the HPP in linear time with linear computational resources, and all solutions can be obtained through this system. The traditional P systems solve the HPP by membrane division rules, while the chain P system realises the computation process without the expansion of membranes (Liu et al., 2015b).

4 The graph clustering based on the chain P systems

In this section, a general clustering problem that database $X = \{a_1, a_2, \cdots, a_n\}$ is clustered into k clusters is considered. Data points are transformed into nodes and dissimilarities

among the data points are transformed into the edge weights of complete undirected graph. The smaller the weight is, the more similar the two data points are. The clustering problem is then transformed into a graph theory problem of finding the shortest path which length is n visiting each node from the graph exactly once.

The process of the algorithm is as follows: A shortest path that connects all nodes is found; the edges with the $k-1$ biggest weights are selected then; the path is divided into k parts from the edges selected above finally. The k parts are the k clusters.

The definition of the dissimilarity matrix is given informally firstly. Matrix D'_{nn} between any two data points is:

$$D'_{nn} = \begin{pmatrix} f'_{11} f'_{12} \ldots f'_{1n} \\ f'_{21} f'_{22} \ldots f'_{2n} \\ \ldots \\ f'_{n1} f'_{n2} \ldots f'_{nn} \end{pmatrix}, \tag{1}$$

where, f'_{ij} is the dissimilarity between a_i and a_j. Specific calculational method is selected depending on the data type.

The matrix elements f'_{ij} are changed to integer f_{ij} by expanding 100 times and rounding for membrane computing. By this, the dissimilarity matrix D_{nn} is obtained:

$$D_{nn} = \begin{pmatrix} f_{11} f_{12} \ldots f_{1n} \\ f_{21} f_{22} \ldots f_{2n} \\ \ldots \\ f_{n1} f_{n2} \ldots f_{nn} \end{pmatrix}. \tag{2}$$

The instance parameters need to enter a chain P system, therefore the weight among the nodes needs to be encoded as object $f = d_{11} * f_{11} \otimes d_{12} * f_{12} \otimes \ldots \otimes d_1 n * f_{1n} \otimes d_{21} * f_{21} \otimes d_{22} * f_{22} \otimes \ldots d_{2n} * f_{2n} \otimes \ldots \otimes d_{n1} * f_{n1} \otimes d_{n2} * f_{n2} \otimes \ldots \otimes d_{nn} * f_{nn}$, where $d_{ij} * f_{ij}$ represents the weight between a_i and a_j is d_{ij}.

4.1 The chain P system for improving the graph clustering

The formal definition of the chain P system for the graph clustering is as follows.

$$\Pi_{graph} = (O, \mu, w_1, w_2, w_3, R_1, R_2, R_3, \rho, i_{in}, i_{out})$$

where

- $O = \{V_{ij}, S, \alpha_i, p, f_{ij}, U_{ij}, c_{ij}, 1 \le i, j \le n\}$
- $\mu = [_1[_3]_3[_2]_2]_1$
- $w_1 = w_2 = \lambda$, $w_3 = \{x = V_{11} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes 0 * S \otimes 0 * \alpha_1 \otimes 0 * \alpha_2 \otimes \ldots \otimes 0 * \alpha_n \otimes (n-k) * p, f\}$
- $\rho = \{r_{i,j} > r_{i,t}\ i = 1, 3, j < t\}$
- $i_{in} = 3$
- $i_{out} = 2$

- R.

Other objects used in Π_{graph} are explained. Object $x = V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes d * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes h * p$ is constructed by $2n + 2$ components, where V_{ij} represents the i^{th} point in a path is a_j, $d * S$ represents the weight sum of a path is d, $h_i * \alpha_i$ represents the occurrence frequency of a_i is h_i, $h * p$ represents h times mergence of the clusters are needed in a path. Object $y = h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_2i_3} * U_{i_2i_3} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes h * p$ is constructed by $n + 1$ components, where $h_{ij} * U_{ij}$ represents the weight between a_i and a_j is h_{ij}, the meaning of $h * p$ is the same with that above. Object $c_{i_ji_{j+1}}$ only constructs by one component, which represents that a_{i_j} and $a_{i_{j+1}}$ belong to the same cluster.

R_1 :

$r_{1.1}$:

$$h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_1i_3} * U_{i_2i_3} \otimes \ldots -1 * U_{i_ji_{j+1}} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes g * p \rightarrow h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_2i_3} * U_{i_2i_3} \otimes \ldots -2 * U_{i_ji_{j+1}} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes (g-1) * p, c_{i_ji_{j+1}}$$

R_1 :

$r_{1.1}$:

$$h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_1i_3} * U_{i_2i_3} \otimes \ldots -1 * U_{i_ji_{j+1}} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes g * p \rightarrow h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_2i_3} * U_{i_2i_3} \otimes \ldots -2 * U_{i_ji_{j+1}} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes (g-1) * p, c_{i_ji_{j+1}}$$

$r_{1.2}$:

$$h_{i_1i_2} * U_{i_1i_2} \otimes h_{i_1i_3} * U_{i_2i_3} \otimes \ldots \otimes h_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes g * p \rightarrow (h_{i_1i_2} - 1) * U_{i_1i_2} \otimes (h_{i_2i_3} - 1) * U_{i_1i_2} \otimes (h_{i_2i_3} - 1) * U_{i_2i_3} \otimes \ldots \otimes (h_{i_{n-1}i_n} - 1) * U_{i_{n-1}i_n} \otimes g * p, g \geq 1$$

$r_{1.3}$:

$$[\sigma_1, \sigma_2](d_{i_1i_2} * U_{i_1i_2} \otimes d_{i_2i_3} * U_{i_2i_3} \otimes \ldots \otimes d_{i_{n-1}i_n} * U_{i_{n-1}i_n} \otimes 0 * p, c_{i_ji_{j+1}}, down)(\lambda, in) \rightarrow (c_{i_ji_{j+1}}, in)(\lambda, up)$$

R_2 :

λ

R_3 :

$r_{3.1}$: $\{r_{3.1.1}, r_{3.1.2}, \ldots, r_{3.1.(n+1)}\}$

$r_{3.1.1}$:

$$V_{11} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes 0 * S \otimes 0 * \alpha_1 \otimes 0 * \alpha_2 \otimes \ldots \otimes 0 * \alpha_n \otimes (n-k) * p \rightarrow$$
$$V_{11} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes 0 * S \otimes 1 * \alpha_1 \otimes 0 * \alpha_2 \otimes \ldots \otimes 0 * \alpha_n \otimes (n-k) * p,$$
$$V_{12} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes 0 * S \otimes 0 * \alpha_1 \otimes 1 * \alpha_2 \otimes \ldots \otimes 0 * \alpha_n \otimes (n-k) * p$$
$$\ldots$$
$$V_{1n} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes 0 * S \otimes 0 * \alpha_1 \otimes 0 * \alpha_2 \otimes \ldots \otimes 1 * \alpha_n \otimes (n-k) * p$$

$r_{3.1.2}$:
$\left(V_{1i} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes d * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes (n-k) * p\right)_f$
$\rightarrow V_{1i} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes \left(d + d_{i1}\right) * S \otimes \left(h_1 + 1\right) * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n$
$\otimes (n-k) * p, V_{1i} \otimes V_{22} \otimes \ldots \otimes V_{n1} \otimes \left(d + d_{i2}\right) * S \otimes h_1 * \alpha_1 \otimes \left(h_2 + 1\right) * \alpha_2$
$\otimes \ldots \otimes h_n * \alpha_n \otimes (n-k) * p,$
…
$V_{1i} \otimes V_{21} \otimes \ldots \otimes V_{n1} \otimes \left(d + d_{in}\right) * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n$
$\otimes (n-k) * p$
…

$r_{3.1.n}$:
$\left(V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{(n-1)i_{n-1}} \otimes V_{n1} \otimes d * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n\right.$
$\left.\otimes (n-k) * p\right)_f \rightarrow V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{(n-1)i_{n-1}} \otimes V_{n1} \otimes \left(d + d_{i_{n-1}1}\right) * S \otimes \left(h_1 + 1\right)$
$* \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes (n-k) * p, V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{(n-1)i_{n-1}} \otimes V_{n2}$
$\otimes \left(d + d_{i_{n-1}2}\right) * S \otimes h_1 * \alpha_1 \otimes \left(h_2 + 1\right) * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes (n-k) * p,$
…
$V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{(n-1)i_{n-1}} \otimes V_{nn} \otimes \left(d + d_{i_{n-1}n}\right) * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots$
$\otimes \left(h_n + 1\right) * \alpha_n \otimes (n-k) * p$

$r_{3.2} : h_n$
$V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes d * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes (n-k)$
$* p \rightarrow \lambda, h_1 * h_2 * \ldots * h_n = 0$

$r_{3.3} : \left\{r_{3.3.1}, r_{3.3.2}\right\}$

$r_{3.3.1}$:
$\left[\sigma_1, \sigma_2\right]\left(\left(V_{1i_1} \otimes V_{2i_2} \otimes \ldots \otimes V_{ni_n} \otimes 0 * S \otimes h_1 * \alpha_1 \otimes h_2 * \alpha_2 \otimes \ldots \otimes h_n * \alpha_n \otimes\right.\right.$
$\left.\left.(n-k) * p\right) f, up\right)(\lambda, in) \rightarrow (\lambda, down)\left(d_{i_1 i_2} * U_{i_1 i_2} \otimes d_{i_2 i_3} * U_{i_2 i_3} \otimes \ldots \otimes d_{i_{n-1} i_n}\right.$
$\left.* U_{i_{n-1} i_n} \otimes (n-k) * p, in\right)$

$r_{3.3.2}$:
$x \rightarrow \lambda$

$r_{3.4}$:
$x : d * S \rightarrow (d-1) * S, d > 0$

Computation begins when f enters membrane 3.

Generating stage

In the initial, $x = V_{11} \otimes V_{21} \otimes \ldots \otimes V_{n10} \otimes S_0 * \alpha_1 \otimes 0 * \alpha_2 \otimes \ldots \otimes 0 * \alpha_n \otimes (n-k) * p$ is in membrane 3. $V_{i,1}$ represents that all points in a path are set to a_1, $0 * S$ represents the

weight sum is set to 0 because no path is formed in the initial, 0 * α_i represents that the occurrence frequency of each data point in the dataset is set to 0, and $(n - k) * p$ represents $n - k$ times mergence of the clusters are needed in a path. Rule $r_{3.1}$ generates all solution space. Sub-rule $r_{3.1.1}$ generates n different f. Each f carries V_{11}, V_{12}, …, V_{1n} respectively, which means the first point in a path has n choices: a_1, a_2, ⋯, a_n. At the same time, the corresponding coefficient of α_i adds one. Sub-rule $r_{3.1.2}$ is executed then. Each f generates n different f again. Each f carries V_{21}, V_{22}, …, V_{2n} respectively, which means the second point in a path has n choices: a_1, a_2, ⋯, a_n. At the same time, the corresponding coefficient of α_i adds one. And so on, until the n^{th} point in a path is chosen. Rule $r_{3.1}$ generates n^n objects f showing all combinations of paths.

Shortest path search stage

The path with the minimum weight sum is found which is called the shortest path. The weight sum of each path is recorded by the coefficient of S. When all weights of a path are add up, the coefficient of S in all f decreases simultaneously by $r_{3.4}$ until 0 * S appearing. The f with 0 * S is changed to $y = d_{i_1 i_2} * U_{i_1 i_2} \otimes d_{i_2 i_3} * U_{i_2 i_3} \otimes \ldots \otimes d_{i_{n-1} i_n} * U_{i_{n-1} i_n} \otimes (n-k) * p$ and enters σ_1 by $r_{3.3.1}$, other f are dissolved by $r_{3.3.2}$.

Division stage

The n points in a path are divided into k parts from the edges with the $k - 1$ biggest weights according to the preset number of clusters k. This can make the points in the same cluster closer, and the points in different clusters more distant. All coefficients of $U_{i_j i_{j+1}}$ reduce at the same time by $r_{1.2}$. When the coefficient of a certain $U_{i_j i_{j+1}}$ equals to –1, an object $c_{i_j i_{j+1}}$ is produced by $r_{1.1}$. The coefficient of $U_{i_j i_{j+1}}$ reduces until $y = d_{i_1 i_2} * U_{i_1 i_2} \otimes d_{i_2 i_3} * U_{i_2 i_3} + \ldots + d_{i_6 i_7} * U_{i_6 i_7} + 5 * p,$ are produced. These $c_{i_j i_{j+1}}$ show that the weight between a_{i_j} and $a_{i_{j+1}}$ is one of the $n - k$ shortest weights among all. Therefore, a_{i_j} and $a_{i_{j+1}}$ belong to the same cluster. All $c_{i_j i_{j+1}}$ enter membrane 2 by $r_{1.3}$.

4.2 Time complexity analysis

The time complexity of this algorithm is $n + 1 + s_{\min} + 1 + d + 1 = O(n + s_{\min} + d)$, where, $s_{\min}$ is the value of the minimum weight sum in all paths, and d is the $(n - k)^{\text{th}}$ minimum weights in the shortest path.

The chain P system solves the graph clustering in linear time.

4.3 Test and analysis

To illustrate how the P system runs specifically, the following simple example is considered: cluster 7 integral points (1, 1), (2, 1), (2, 2), (3, 4), (4, 2), (4, 3), (5, 4) into two clusters. Obviously, $n = 7$, $k = 2$.

The dissimilarity matrix D_{77} is constructed firstly. In this example, the Euclidean distance is used as the dissimilarity.

$$D_{77} = \begin{pmatrix} 0 & 1 & 2 & 13 & 10 & 13 & 25 \\ 1 & 0 & 1 & 10 & 5 & 8 & 18 \\ 2 & 1 & 0 & 5 & 4 & 5 & 13 \\ 13 & 10 & 5 & 0 & 5 & 2 & 4 \\ 10 & 5 & 4 & 5 & 0 & 1 & 5 \\ 13 & 8 & 5 & 2 & 1 & 0 & 2 \\ 25 & 18 & 13 & 4 & 5 & 2 & 0 \end{pmatrix}, \tag{3}$$

Rules $r_{3.1}$ and $r_{3.2}$ are executed firstly generating 7! objects f which means all combinations of paths.

Rules $r_{3.3}$ and $r_{3.4}$ find the f representing the shortest path. This f is changed to $y = d_{i_1 i_2} * U_{i_1 i_2} \otimes d_{i_2 i_3} * U_{i_2 i_3} + \ldots + d_{i_6 i_7} * U_{i_6 i_7} + 5 * p$, and enters σ_1, other f are dissolved. In this example, one of the four shortest paths is obtained randomly:

1 7-4-6-5-3-2-1

2 4-7-6-5-3-2-1

3 1-2-3-5-6-4-7

4 1-2-3-5-6-7-4.

Rules $r_{1.1}$, $r_{1.2}$ and $r_{1.3}$ generate $c_{i_j i_{j+1}}$ which means a_{i_j} and $a_{i_{j+1}}$ are in one cluster, and put them into membrane 2. If the first path is chosen, the two clusters can be {4, 6, 7}, {1, 2, 3, 5} or {4, 5, 6, 7}, {1, 2, 3} because the weights between 6, 5 and 5, 3 are the same. Similarly, if the second path is chosen, the two clusters can be {4, 6, 7}, {1, 2, 3, 5} or {4, 5, 6, 7}, {1, 2, 3} because the weights between 6, 5 and 5, 3 are the same. If the third path is chosen, the two clusters are {4, 6, 7}, {1, 2, 3, 5}. If the fourth path is chosen, the two clusters are {4, 6, 7}, {1, 2, 3, 5}.

5 The ROCK clustering based on the chain P systems

Boolean or categorical type database consists of transactions. Each transaction consists of some items and each item has a number of fixed values. In this section, a clustering problem for Boolean or categorical type database is considered: database $P = \{p_1, p_2, \cdots, p_n\}$ with each transaction has m items is clustered into k clusters.

Three notions are introduced firstly (Guha et al., 2000).

- *Neighbours:* The neighbours of a data point are the data points which are similar to it. In this paper, the similarity between two transactions is set as the number of the same value items. If the number is larger than or equal to a certain threshold, the two transactions are called neighbours.
- *Links: Links*(p_i, p_j) is the number of the common neighbours between transactions p_i and p_j. The larger the *Links*(p_i, p_j), the more common neighbours they have.
- *Goodness measure:* Goodness measure evaluates the similarity between clusters. link [C_i, C_j] defines the number of links between clusters C_i and C_j, i.e., $link\left[C_i, C_j\right]$

$= \sum_{p_q \in C_i, p_r \in C_j} links(p_q, p_r)$. In this paper, the goodness measure for merging clusters C_i and C_j is defined as follow: $g(C_i, C_j) = \frac{link[C_i, C_j]}{n_i + n_j}$. This goodness measure is equivalent to the average number of links in the two clusters. The pair of clusters with the maximum goodness measure is the best pair of clusters to be merged.

The process of the algorithm is as follows: Each transaction is seen as a cluster initially. The two clusters with the largest goodness measure are merged iteratively until k clusters are left.

The instance parameters need to enter a chain P system, therefore the database needs to be encoded as object $p = h_1 * a_1 \otimes h_2 * a_2 \otimes \ldots \otimes h_m * a_m$, $h_i \in \{0, 1\}$ representing the database, where $1 * a_i$ (resp. $0 * a_i$) represents the i^{th} item is contained (resp. not contained) in a transaction; and the number of clusters needs to be encoded as object ξ showing the number of clusters needs to be obtained by its number.

5.1 The chain P system for improving the ROCK clustering

The formal definition of the chain P system for the ROCK clustering is as follows.

$$\Pi_{ROCK} = (O, \mu, w, R, \rho, i_{in}, i_{out})$$

- $O = \{a_i, \xi, t_j, \gamma_j, \delta_{i_1 i_2}, \delta'_{i_1 i_2}, c_{i_1 i_2}, c_{i_1 i_2 i_3}, d_{i_1 i_2}, \gamma, \gamma', C_{i_1 i_2}, 1 \le i \le m, 1 \le i \le n\}$
- $\mu = [_1]_1$
- $w = \{1 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_1, 0 * t_1 \otimes 1 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_2, \ldots, 0 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 1 * t_n \otimes 1 * \gamma_n, 0 * c_{ij1} \otimes 0 * c_{ij2} \otimes \ldots \otimes 0 * c_{ijn} \otimes 0 * \delta_{ij} \otimes 2 * \gamma, C_{00}\}$
- $\rho = \{r_i > r_j, i < j\}$
- $i_{in} = 1$
- $i_{out} = 1$
- R.

Other objects used in Π_{ROCK} are explained. Object $X = f_1 * t_1 \otimes f_2 * t_2 \otimes \ldots \otimes f_n * t_n \otimes * q \otimes \gamma_i$ represents the order number and the total number of the transactions in C_i. Object $T = h_1 * a_1 \otimes h2 * a_2 \otimes \ldots \otimes h_m * a_m \otimes d * \delta_{ij}$ is an auxiliary variable, where $h_f * a_f$ represents the total number of the f^{th} item in transactions p_i and p_j is h_f, and $d * \delta_{ij}$ represents the similarity between p_i and p_j is d. Object c_{ij} represents p_i and p_j are neighbours. Object $S = h_1 * c_{ij1} \otimes h_2 * c_{ij2} \otimes \ldots \otimes h_n * c_{ijn} \otimes d * \delta_{ij} \otimes q * \gamma$ is an auxiliary variable, where the first n components represent p_i and p_j contain/do not contain the common neighbour: p_t, $d * \delta_{ij}$ represents the number of the common neighbours between p_i and p_j is d, and $q * \gamma$ represents the total number of the considered transactions is q. Object $D = q * \delta'_{ij} \otimes d * \gamma'$ represents the number of the common neighbours between C_i and C_j is q and the total transactions number in C_i and C_j is d. Object $D' = \mathrm{q} * \delta'_{ij} \otimes d * \gamma' \otimes h * d_{ij}$ represents the number of the common neighbours between C_i and C_j is q, the

total transactions number in C_i and C_j is d, and the goodness is h. Object C_{ij} shows the goodness between C_i and C_j is the biggest one.

$r_1 : \{r_{1.1}, r_{1.2}, r_{1.3}, r_{1.4}, r_{1.5}, r_{1.6}, r_{1.7}\}$

$r_{1.1}$:
$()_{t_i t_j \neg T_{ij}} \to (h_1 + g_1) * a_1 \otimes (h_2 + g_2) * a_2 \otimes \ldots \otimes (h_m + g_m) * a_m \otimes 0 * \delta_{ij}$
$i, j \in \{1, 2, \ldots, n\}, i \neq j$

$r_{1.2}$:
$p \to \lambda$

$r_{1.3}$:
$T_{ij} : 2 * a_f \otimes d * \delta_{ij} \to 0 * a_f \otimes (d+2) * \delta_{ij} \quad f, i, j \in \{1, 2, \ldots, n\}, i \neq j$

$r_{1.4}$:
$h_1 * a_1 \otimes h_2 * a_2 \otimes \ldots \otimes h_m * a_m \otimes d * \delta_{ij} \to \lambda \quad d < \theta$

$r_{1.5}$:
$h_1 * a_1 \otimes h_2 * a_2 \otimes \ldots \otimes h_m * a_m \otimes d * \delta_{ij} \to c_{ij}$

$r_{1.6}$:
$S_{ij} : (0 * c_{ijf} \otimes d * \delta_{ij})_{c_i f c_j f} \to 1 * c_{ijf} \otimes (d+1) * \delta_{ij} \quad i, j \in \{1, 2, \ldots, n\}, i \neq j$

$r_{1.7}$:
$h_1 * c_{ij1} \otimes h_2 * c_{ij2} \otimes \ldots \otimes h_n * c_{ijn} \otimes d * \delta_{ij} \otimes q * \gamma \to d * \delta_{ij} \otimes q * \gamma, d * \delta'_{ij} \otimes q$
$* \gamma' \otimes 0 * d_{ij} \quad i, j \in \{1, 2, \ldots, n\}, i \neq j$

r_2 :
$D' : (d+q) * \delta'_{ij} \otimes q * \gamma' \otimes h * d_{ij} \to d * \delta'_{ij} \otimes q * \gamma' \otimes (h+1) * d_{ij} \quad d \geq 0$

r_3 :
$(d * \delta'_{ij} \otimes q * \gamma' \otimes 0 * d_{ij}, C_{pq})_{\neg D' : h * d_{ij}} \to d * \delta'_{ij} \otimes g * \gamma' \otimes -1 * d_{ij}, C_{ij} \quad h > 0$

r_4 :
$D' : h * d_{ij} \to (h-1) * d_{ij} \quad h \geq 0$

$r_5 : \{r_{5.1}, r_{5.2}, r_{5.3}, r_{5.4}\}$

$r_{5.1}$:
$(\xi, h_1 * t_1 \otimes h_2 * t_2 \otimes \ldots \otimes h_n * t_n \otimes q * \gamma_i h'_1 * t_1 + h'_2 * t_2 \otimes \ldots \otimes h'_n * t_n$
$\otimes q' * \gamma_j)_{C_{ij}} \to (h_1 + h'_1) * t_1 \otimes (h_2 + h'_2) * t_2 \otimes \ldots \otimes (h_n + h'_n) * t_n \otimes (q + q') * \gamma_i$

$r_{5.2}$:
$q * \delta'_{ij} \otimes g * \gamma' \otimes h * d_{ij} \to \lambda$

$r_{5.3}$:

$$\left(q * \delta_{it} \otimes g * \gamma, q' * \delta_{jt} \otimes g' * \gamma\right)_{C_{ij}} \rightarrow \left(q+q'\right) * \delta_{it} \otimes \left(g+g'\right) * \gamma$$

$$\cup\left(q * \delta_{it} \otimes g * \gamma, q' * \delta_{tj} \otimes g' * \gamma\right)_{C_{ij}} \rightarrow \left(q+q'\right) * \delta_{ti} \otimes \left(g+g'\right) * \gamma$$

$r_{5.4}$:

$$\left(d * \delta_{ij} \otimes g * \gamma\right)_{\neg d * \delta'_{ij} \otimes g * \gamma' \otimes 0 * d_{ij}} \rightarrow d * \delta_{ij} \otimes g * \gamma, d * \delta'_{ij} \otimes g * \gamma' \otimes 0$$

$$* d_{ij} \quad i, j \in \{1, 2, \ldots, n\}, i \neq j$$

In the initial, $1 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_1$, $0 * t_1 \otimes 1 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_2$, …, $0 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 1 * t_n \otimes 1 * \gamma_n$, $0 * c_{ij} \otimes 1 * 0 \otimes c_{ij} * 2 \otimes \ldots \otimes 0 * c_{ijn} \otimes 0 * \delta_{ij} \otimes 2 * \gamma$ and C_{00} are in membrane 1. Objects $1 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_1$, $0 * t_1 \otimes 1 * t_2 \otimes \ldots \otimes 0 * t_n \otimes 1 * \gamma_2$, …, $0 * t_1 \otimes 0 * t_2 \otimes \ldots \otimes 1 * t_n \otimes 1 * \gamma_n$ show each transaction is seen as one cluster in the initial, $0 * c_{ij1} \otimes 0 * c_{ij2} \otimes \ldots \otimes 0 * c_{ijn} \otimes 0 * \delta_{ij} \otimes 2 * \gamma$ shows the number of neighbours has not been obtained, and C_{00} shows the two clusters need to be merged have not been obtained. Computation begins when p and ζ^{n-k} enter membrane 1.

Neighbours judging stage

Sub-rule $r_{1.1}$ generates T_{ij} where the coefficient of a_f in T_{ij} is the sum of the coefficient of a_f in t_i and t_j. The f^{th} item is the common item of p_i and p_j if the coefficient of a_f is 2. Sub-rule $r_{1.3}$ counts the number of the common items of p_i and p_j by the coefficient of δ_{ij}. If the number of the common items of p_i and p_j is less than the previously set threshold, T_{ij} is dissolved by $r_{1.4}$. Otherwise, c_{ij} is generated by $r_{1.5}$ to record that p_i and p_j are neighbours.

Links counting stage

If c_{if} and c_{jf} exist at the same time, transaction p_f is the common neighbour of p_i and p_j, the coefficient of δ_{ij} adds 1 to record this by $r_{1.6}$, i.e., the coefficient of δ_{ij} shows the number of links between p_i and p_j.

Maximum goodness finding stage

Sub-rule $r_{1.7}$ generates D and D'. In D', the coefficient of d_{ij} is the goodness of clusters C_i and C_j by r_2. The coefficients of all d_{ij} are decreased at the same time by r_4, and the C_{ij} representing the biggest goodness is obtained by r_3.

Clustering merging stage

Rule r_5 merges clusters C_i and C_j. Sub-rule $r_{5.1}$ obtains the new cluster C_i which contains the transactions in previous C_i and C_j. Sub-rule $r_{5.3}$ obtains the new number of common neighbours between the new C_i and other clusters. Sub-rule $r_{5.4}$ generates the new D' which is used to measure the goodness of the new clusters.

Above process continues until only k clusters are remained.

Table 1 The seven clusters obtained by the ROCK algorithm

Cluster	*The serial number of data points belonging to the corresponding cluster*
1	1, 4, 9, 14, 18-20, 22, 26, 32, 38, 44, 54, 55, 79, 82, 115, 121, 123, 136, 139, 181, 186, 206, 222, 229, 232, 244, 252, 262, 270, 272, 281, 300, 312, 316, 328, 331, 358, 381, 386, 400, 403, 415, 418, 423, 492, 493, 506, 524, 533, 535, 543, 557, 566, 569, 580, 591, 594, 596, 599, 600, 614, 654, 655, 663, 664, 695, 698, 699, 701, 726, 733, 734, 738, 749, 777, 786, 789, 795, 796, 799, 803, 813, 814, 815, 836, 838, 842, 860, 907, 909, 910, 928, 933, 943, 950, 951, 957, 967, 985, 1000
2	2, 3, 6-8, 10-13, 21, 23-25, 27, 28, 31, 33-35, 40-42, 45-53, 56, 58-60, 62-65, 67-69, 72, 74, 75, 78, 80, 85, 87-89, 92-94, 96-100, 102-111, 114, 116, 118, 122, 125, 127, 129-132, 134, 138, 140-142, 144, 145, 148-151, 153, 156-162, 164-170, 172, 174-180, 184, 185, 187, 189, 190, 193, 195, 197-199, 201-203, 205, 207-212, 215-219, 221, 223, 225-228, 230, 233, 235, 237-243, 246, 248, 249, 251, 254-256, 258-264, 268, 269, 271, 273-278, 280, 282, 284-289, 292-297, 302, 304, 305, 307-309, 311, 313, 315, 318, 321, 323-327, 329, 330, 332-335, 337, 338, 340-343, 345, 348, 352-355, 357, 359, 360, 364, 365, 367, 369, 371-373, 375, 376, 378-380, 383, 384, 387, 389-393, 395, 397, 398, 401, 402, 404-411, 413, 414, 416, 419, 421, 422, 424, 425, 427, 428, 431, 432-434, 436, 438-449, 451-453, 455, 457-464, 466-472, 478, 480-482, 486, 487, 489, 490, 494-496, 498-505, 512, 517-523, 526-528, 530-532, 534, 537, 539, 541, 544, 546, 547, 550, 553, 555, 556, 559-561, 565, 567, 570, 571, 573-576, 578, 579, 581, 582-586, 590, 593, 595, 597, 598, 602, 604-606, 608, 609, 615, 616, 618, 619, 621, 622, 624, 626, 629, 630, 632, 634-638, 640, 641-645, 647-649, 651-653, 656, 658, 659, 661, 662, 665-671, 675, 678-680, 682, 684, 686, 688-693, 696, 697, 703-707, 709, 710, 711, 713-719, 721, 722, 725, 727, 729-731, 735, 736, 739, 742-748, 751, 752, 754-757, 759-763, 765, 769-773, 776, 778, 779-781, 783-785, 788, 790-794, 797, 800-802, 804-809, 811, 812, 816-818, 820, 821, 823-826, 829, 833-835, 837, 839, 840, 843, 845, 847-849, 851, 852, 856, 858, 859, 861-866, 868, 870-873, 875-880, 883-885, 887-890, 892-901, 905, 911, 913, 916, 917, 923, 930, 934, 937, 941, 948, 952, 956, 960, 977, 979, 982, 990, 991, 997, 999
3	5, 15, 17, 57, 66, 81, 84, 86, 95, 101, 124, 126, 128, 146, 147, 152, 191, 204, 250, 267, 279, 298, 301, 310, 346, 349, 351, 361-363, 370, 382, 388, 396, 412, 420, 429, 456, 497, 507, 508, 516, 536, 542, 545, 552, 564, 568, 592, 601, 603, 607, 611-613, 617, 631, 639, 657, 673, 694, 700, 702, 712, 724, 728, 732, 758, 766, 775, 798, 846, 854, 882, 902-904, 906, 908, 912, 914, 915, 919-922, 925-927, 931, 932, 936, 938, 939, 942, 944, 946, 947, 949, 953-955, 961, 962, 965, 966, 969-971, 974, 976, 980, 983, 984, 986, 987-989, 992, 993, 995, 996
4	16, 29, 37, 43, 61, 70, 83, 90, 91, 112, 117, 120, 143, 154, 163, 171, 182, 183, 188, 196, 214, 220, 231, 234, 253, 257, 266, 283, 290, 291, 339, 350, 356, 366, 374, 377, 394, 399, 430, 465, 474, 476, 477, 484, 488, 510, 511, 514, 515, 525, 529, 538, 548, 551, 577, 587-589, 610, 620, 625, 628, 660, 672, 676, 677, 687, 708, 720, 740, 741, 764, 768, 782, 787, 822, 831, 844, 853, 855, 867, 869, 929, 935, 994
5	30, 36, 39, 71, 73, 76, 77, 113, 119, 133, 135, 137, 155, 173, 192, 194, 200, 213, 224, 236, 245, 247, 265, 299, 303, 314, 317, 319, 320, 322, 344, 347, 368, 417, 426, 435, 437, 454, 473, 475, 479, 483, 485, 491, 509, 513, 540, 549, 554, 558, 562, 563, 572, 623, 627, 646, 650, 674, 681, 683, 723, 737, 750, 753, 767, 774, 810, 819, 827, 828, 830, 832, 841, 850, 857, 881, 886, 924, 959, 998
6	306
7	336, 385, 450, 633, 685, 874, 891, 918, 940, 945, 958, 963, 964, 968, 972, 973, 975, 978, 981

5.2 Time complexity analysis

The time complexity of this algorithm is $(1 + 1 + 1 + 1 + 1 + 1 + 1) + (1 + 1 + g_{max} + (1 + 1 + 1 + 1)) * (n - k) = O(g_{max} * n)$, while the time complexity of the conventional ROCK algorithm is $O(n^2 + nm_{max}m_a + n^2 \log n)$ (Guha et al., 2000). Where, g_{max} is the value of the maximum goodness, m_{max} is the maximum number of the neighbours, and m_a is the average number of the neighbours.

The chain P systems solve the ROCK algorithm in linear time.

5.3 Test and analysis

Mushroom database of UCI dataset contains 8,124 data points. Each data point contains 22 properties. All data points are divided into two classes: edible and poisonous. In this paper, the first 1,000 data points are used. The 1,000 data points are numbered from 1 to 1000 following the order. Two data points with 18 or more same property values are seen as neighbours. The seven clusters obtained are listed in Table 1.

As shown in the table, the sizes of the first cluster, the third cluster, the fourth cluster and the fifth cluster are almost the same (there sizes are 102, 122, 85 and 80 respectively). The second cluster is the largest cluster with the size of 591 and the sixth cluster is the smallest cluster with the size of 1. There is a big difference between cluster sizes. All clusters obtained are pure clusters, i.e., in any cluster, all data points are either all poisonous or all edible. The error rate is 0%.

6 Conclusions

In this paper, a new chain P system is proposed. The chain membranes, objects and rules are presented. Computational power and efficiency are analysed. It is proved that the chain P systems are universal, and they can solve the NP problems without the expansion of membranes. The graph clustering algorithm and the ROCK algorithm based on the chain P system are given as applications. Results show that improvements of conventional algorithms by chain P systems are non-trivial. For the future research, it is worth investigating other variants of P systems according to the structure and function of the biological membrane. It is also worth improving other data mining algorithms by using P systems, such as spectral clustering, support vector machines, and genetic algorithms, etc. (Han et al., 2012).

Acknowledgements

Project supported by National Natural Science Foundation of China (No. 61806114, 61876101, 61802234, 61472231, 61502283, 61640201, 61602282), China Postdoctoral Science Foundation (No. 2018M642695, 2019T120607), Natural Science Foundation of Shandong Province ZR2016AQ21) and Funding for Study Abroad Program by the Government of Shandong Province.

References

Cabarle, F., Adorna, H., Jiang, M. and Zeng, X. (2017) 'Spiking neural P systems with scheduled synapses', *IEEE Transactions on Nanobioscience*, DOI: 10.1109/TNB.2017.2762580.

Guha, S., Rastogi, R. and Shim, K. (2000) 'Rock: a robust clustering algorithm for categorical attributes', *Information Systems*, Vol. 25, No. 5, pp.345–366.

Han, J., Kambr, M. and Pei, J. (2012) *Data Mining: Concepts and Techniques*, Elsevier Inc., Amsterdam, Netherlands.

Ju, Y., Zhang, S., Ding, N., Zeng, X. and Zhang, X. (2016) 'Complex network clustering by a multiobjective evolutionary algorithm based on decomposition and membrane structure', *Scientific Reports*, DOI: 10.1038/srep33870.

Liu, X. and Xue, J. (2017) 'A cluster splitting technique by Hopfield networks and P systems on simplices', *Neural Processing Letters*, Vol. 46, pp.1–24.

Liu, X., Li, Z., Liu, J., Liu, L. and Zeng, X. (2015a) 'Implementation of arithmetic operations with time-free spiking neural P systems', *IEEE Transactions on Nanobioscience*, Vol. 14, No. 6, pp.617–624.

Liu, X., Suo, J., Leung, S., Liu, J. and Zeng, X. (2015b) 'The power of time-free tissue P systems: attacking NP-complete problems', *Neurocomputing*, Vol. 159, No. 1, pp.151–156.

Liu, X., Zhao, Y. and Sun, M. (2017) 'An improved apriori algorithm based on an evolutioncommunication tissue-like P system with promoters and inhibitors', *Discrete Dynamics in Nature and Society*, Vol. 2017, No. 1, pp.1–11.

Liu, X., Zhao, Y. and Sun, W. (2018) 'Tissue P systems with cooperating rules', *Chinese Journal of Electronics*, Vol. 27, No. 2, pp.324–333.

Pan, L., Păun, G., Pérez-Jiménez, M.J. (2011) 'Spiking neural P systems with neuron division and budding', *Science China Information Sciences*, Vol. 54, No. 8, pp.1596–1607.

Păun, G., Rozenberg, G. and Salomaa, A. (2010) *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, UK.

Peng, H., Yang, J., Wang, J., Wang, T., Sun, Z., Song, X., Luo, X. and Huang, X. (2017) 'Spiking neural P systems with multiple channels', *Neural Networks the Official Journal of the International Neural Network Society*, Vol. 95, No. 66, pp.66–71.

Song, B., Song, T. and Pan, L. (2015) 'Time-free solution to SAT problem by P systems with active membranes and standard cell division rules', *Natural Computing*, Vol. 14, No. 4, pp.673–681.

Song, T. and Pan, L. (2015) 'Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy', *IEEE Transactions on Nanobioscience*, Vol. 14, No. 1, pp.38–44.

Song, T. and Wang, X. (2015) 'Homogenous spiking neural P systems with inhibitory synapses', *Neural Processing Letters*, Vol. 42, No. 1, pp.199–214.

Song, T., Gong, F., Liu, X., Zhao, Y. and Zhang, X. (2016) 'Spiking neural P systems with white hole neurons', *IEEE Transactions on NanoBioscience*, Vol. 15, No. 7, pp.666–673.

Song, T., Luo, L., He, J., Chen, Z. and Zhang, K. (2014a) 'Solving subset sum problems by timefree spiking neural P systems', *Applied Mathematics and Information Sciences*, Vol. 8, No. 1, pp.327–332.

Song, T., Zheng, H. and He, J. (2014b) 'Solving vertex cover problem by tissue P systems with cell division', *Applied Mathematics and Information Science*, Vol. 8, No. 1, pp.333–337.

Song, T., Rodrguez-patn, A., Zheng, P. and Zeng, X. (2017) 'Spiking neural P systems with colored spikes', *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 10, No. 4, pp.1106–1115.

Song, T., Pan, L., Wu, T., Zheng, P., Wong, M. and Rodrguez-patn, A. (2019a) 'Spiking neural P systems with learning functions', *IEEE Transactions on NanoBioscience*, Vol. 18, No. 2, pp.176–190.

Song, T., Zheng, P., Wong, M., Jiang, M. and Zeng, X. (2019b) 'On the computational power of asynchronous axon membrane systems', *IEEE Transactions on Emerging Topics in Computational Intelligence*, 29 April, No. 1, pp.1–9, DOI: 10.1109/TETCI.2019.2907724.

Wang, J., Hoogeboom, H.J. and Pan, L. (2011) 'Spiking neural P systems with neuron division', *Membrane Computing*, Vol. 6501, pp.361–376.

Wang, T., Wei, X., Huang, T., Wang, J., Peng, H., Prez-Jimnez, M. and Valencia-Cabrera, L. (2019a) 'Modeling fault propagation paths in power systems: a new framework based on event SNP systems with neurotransmitter concentration', *IEEE Access*, Vol. 7, pp.12798–12808.

Wang, J., Peng, H., Yu, W., Ming, J., Prez-Jimnez, M., Tao, C. and Huang, X. (2019b) 'Interval-valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks', *Engineering Applications of Artificial Intelligence*, Vol. 82, pp.102–109.

Wang, X., Song, T., Gong, F. and Zheng, P. (2016) 'On the computational power of spiking neural P systems with self-organization', *Scientific Reports*, DOI: 10.1038/srep27624.

Zeng, X., Xu, L., Liu, X. and Pan, L. (2014) 'On languages generated by spiking neural P systems with weights', *Information Sciences*, Vol. 278, No. 10, pp.423–433.

Zeng, X., Zhang, X. and Pan, L. (2009) 'Homogeneous spiking neural P systems', *Fundamenta Informaticae*, Vol. 97, No. 1, pp.275–294.

Zeng, X., Zhang, X., Song, T. and Pan, L. (2014) 'Spiking neural P systems with thresholds', *Neural Computation*, Vol. 26, No. 7, pp.1340–1361.

Zhang, X., Pan, L. and Paun, A. (2017) 'On the universality of axon P systems', *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 11, pp.2816–2829.

Zhang, X., Wang, B. and Pan, L. (2014a) 'Spiking neural P systems with a generalized use of rules', *Neural Computation*, Vol. 26, No. 12, pp.2925–2943.

Zhang, X., Zeng, X., Luo, B. and Pan, L. (2014b) 'On some classes of sequential spiking neural P systems', *Neural Computation*, Vol. 26, No. 5, pp.974–997.

Zhao, Y., Liu, X. and Wang, W. (2016) 'Spiking neural P systems with neuron division and dissolution', *PLOS One*, DOI: 10.1371/journal.pone.0162882.