A Morse Membrane Boltzmann Machine Model With Applications in Cluster Analysis

XIYU LIU
Shandong Normal University
School of Management Science and Engineering
Jinan, Shandong 250014
CHINA
sdxyliu@163.com

JIE XUE
Shandong Normal University
School of Management Science and Engineering
Jinan, Shandong 250014
CHINA
xiaozhuzhu1113@163.com

Abstract: The purpose of this paper is to propose a new kind of P system on chain structure. We present the basic discrete Morse structure, membrane structures on complexes, objects with positive and negative charges and communication rules on chains. The computation completeness of Morse P system is proved by simulation of register machine. The process of Boltzmann Machines are implemented by Morse P system. A new clustering technique is described on Morse P system based Boltzmann Machines. Examples are given to show the effect of the algorithm.

Key–Words: Cluster analysis, membrane computing, discrete Morse theory, chain structure.

1 Introduction

Membrane computing is a new class of distributed and parallel computing devices which is initiated by Păun at the end of 1998, as an attempt to formulate models from the functioning of living cells [1]. One main advantage of the new computing models lies in its huge inherent parallelism which has drawn great attention from researchers. A number of successful applications have been reported in wide areas such as biology, bio-medicine, linguistics, computer graphics, economics, approximate optimization, cryptography, and so forth.

There are four main streams [1] in the research of membrane computing. They are design of new P systems, computational power, combination of P systems and biochemistry, and implementation of P systems. Up to now, there are many types of P systems such as the cell-like P systems, tissue-like P systems, spiking neural P systems [1][2][3]. Extensions of these P systems include numerical P systems adding integers to multisets [5], real-time extension of P systems [6]. Also an evolution communication P systems is proposed to measure the communication costs by means of quanta of energy [7]. Another kind of P systems with object processing rules and structure changing rules is investigated in [8].

Traditionally, Morse theory is a fundamental topic of differential topology and differential geometry which works on smooth manifolds. It is a great challenge to develop a discrete Morse structure when faced with discrete problems. By replacing manifold with simplices, Forman proposes a discrete Morse theory [12]. Recently, discrete Morse theory has attracted many researchers because it has been found applications in triangulations and graphics. In fact, simplicial complex, the basic data structure in discrete Morse theory, will prove to be an important extension of data structure other than trees and graphs.

The Boltzmann machine, introduced by Ackley D.H., Hinton G.E. and Sejnowski T.J. [13], is a traditional neural network model using a distributed knowledge representation and a massively parallel network of simple stochastic computing elements. Boltzmann machine is widely used in optimization applications [14]. Many empirical studies have been carried out in graph partitioning problems and the travelling Salesman problem, combinatorial optimization [9], pattern recognition and so on. Cluster analysis is a wide area of data analytics with both methodological value and applications such as document clustering [15][16]. However we find there are seldom joint study of membrane computing, Boltzmann machine with cluster analysis.

Inspired by the above research, this paper focuses on the joint study of discrete Morse theory with membrane computing. Our purpose is to propose a P system on chain structure. We propose a discrete Morse structure for a candidate of a class of new P systems. We also provide objects with positive and negative charges. We prove the computation completeness of Morse P system by simulation of register machine. Then we use membrane computing in Boltzmann Ma-

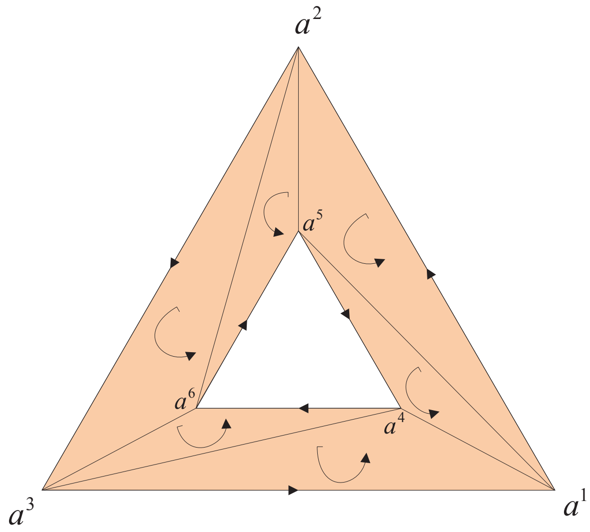

Figure 2.1: A simplicial complex with orientated faces in R^2.

chines, implementing the process of Boltzmann Machines by Morse P system with positive and negative objects. Finally, we present cluster analysis by Morse P system based Boltzmann Machines and give examples for detailed analyzing.

2 A Morse Membrane Structure

In this section we propose a class of membrane models based on discrete Morse structures. For simplicity we always assume that we are working in an Euclidean space R^n.

2.1 Simplices with Orientation

A $k-$simplex (cell) τ is the convex hull of $k+1$ affinely independent points denoted by $a_0, a_1, \cdots, a_k$. The integer k is called dimension of simplex σ, while $a_0, a_1, \cdots, a_k$ are called vertices. A simplex is uniquely indicated by its vertices and hence can be expressed by $\tau = [a_0, a_1, \cdots, a_k]$, or simply $\tau = a_0 a_1 \cdots a_k$. For a simplex $\tau = [a_0, a_1, \cdots, a_k]$, there are exactly two orientations induced by permutation of its vertices. So we will use $-\tau$ to denote a cell with the same set of vertices but opposite orientation to τ. A complex is a finite collection of non-empty simplices where each simplex is called its member and members are well placed (which will be described in detail later in this section). Figure Fig. 2.1 shows the orientation of a two dimensional complex.

A face τ_1 of a simplex τ is defined as a simplex generated by a nonempty subset of its vertices denoted by $\tau_1 < \tau$. A face τ_1 is called a *hyperface* of τ if $\dim \tau_1 = \dim \tau - 1$ and is denoted by $\tau_1 \prec \tau$. In this case, τ is called the parent of τ_1. Two cells τ_1 and τ_2 are called *incident* if $\tau_1, \tau_2 \prec \tau$, and τ is called the *coface* of τ_1 and τ_2. Two cells τ_1, τ_2 are called *neighbors* if they share a common hyperface.

A *simplicial complex* K is a collection of nonempty simplices for which $\tau \in K$ and $\tau_1 \prec \tau$ implies $\tau_1 \in K$; and $\tau_1, \tau_2 \in K$ implies that $\tau_1 \cap \tau_2$ is either empty or a face of both. Denote K_q as the subset of K containing simplices of dimension q. If τ is a $k-$simplex, then the collection of τ and all its faces form a simplicial complex K, which is called a simple complex, or simply, a complex.

For a $k-$dimensional simplicial complex K, K_q contains its $q-$dimensional simplices. We always suppose that each simplex in k_q is oriented. A $q-$chain is a collection of $q-$dimensional cells organized as follows

$$\sigma_q = \sum_i g_i \tau_i^q, \; g_i \in G, \tau_i^q \in K_q \tag{2.1}$$

Here G is an Abelian group. One widely used and typical group is the integer group J. Another common group is $J-2 = J \bmod 2$. In the following we will not specify the group G in general.

The set of $q-$dimensional chains form a group which is written by $C_q(K)$. Now we will use $[a_0, a_1, \cdots, \hat{a_i}, \cdots, a_k]$ to denote a face of τ where the vertex a_i is eliminated. For $\sigma_q = a_0 a_1 \cdots a_q$, the boundary operator is defined by:

$$\partial \sigma_q = \begin{cases} 0, \; q = 0 \\ \sum_{i=0}^{q} (-1)^i [a_0, \cdots, \hat{a_i}, \cdots, a_q], \\ \quad q > 0 \end{cases} \tag{2.2}$$

Boundary operator extends to chains in the natural way. An important property of the boundary operator is that $\partial \circ \partial = 0$ [17]. For a $(q-1)-$dimensional simplex τ^{q-1} and a $q-$dimensional simplex τ_q, define the *relevance operator* as follows

$$r(\tau^q, \tau^{q-1}) = \begin{cases} 0, \tau^{q-1} \not\prec \tau^q \\ 1, \tau^{q-1} \prec \tau^q \\ \text{ with same orientation} \\ -1, \tau^{q-1} \prec \tau^q \\ \text{ with opposite orientation.} \end{cases} \tag{2.3}$$

Clearly the boundary operator maps $C_q(K)$ into $C_{q-1}(K)$.

$$\partial = \partial_q : C_q(K) \to C_{q-1}(K) \tag{2.4}$$

For convenience of symbol consistency we denote $C_{-1}(K) = 0$. Write the kernel of ∂_q as $Z_q(K) =$

$\partial_q^{-1}(0)$ which is a subgroup of $C_q(K)$. Chains in $Z_q(K)$ are called closed chains. If σ_q is a boundary chain, i.e., the boundary of another chain, we write $\sigma_q \sim 0$. Define the boundary chain group $B_q(K)$ (which is a subgroup of $Z_q(K)$) as

$$B_q(K) = \{\sigma_q : \sigma_q \sim 0\} \tag{2.5}$$

Two chains σ'_q, σ''_q are homologic if $\sigma'_q - \sigma''_q \sim 0$ and are written by $\sigma'_q \sim \sigma''_q$. In case the dimension of K is k, we have $B_k(K) = 0$.

Definition 2.1 *The homology group of a simplicial complex K is the quotient group $H_q(K) = Z_q(K)/B_q(K)$.*

2.2 A Chain Membrane Model

Now we propose a membrane model based on chain structures. Suppose O is the alphabet in the traditional sense of membrane computing [19]. Now we define a extended alphabet Θ which is composed of symbols with positive and negative charges

$$\Theta = \{a_{+,-} : a \in O\} \tag{2.6}$$

If $a \in O$, we will identify a with a_+ that is positive charged for brevity. When two same symbols with positive charge and negative charge meet, they will annihilate, i.e., $a_+a_- = \lambda$. We will use x to represent a multiset on extended alphabet Θ. For a positive integer g, we define x^g as a new multiset with each symbol from x amplifying g copies. For $a \in O$, define $a^{-1} = a_-, a_-^{-1} = a_+$. Also we define $x^{-g} = (x^{-1})^g$. For two integers g_1, g_2, define $x^{g_1} \otimes x^{g_2} = x^{g_1+g_2}$.

Now suppose there exist multisets $x_i \in \tau_i$ where $\tau_i \in K_q$. In order to specify the host membrane of multisets, we will use a new symbol $[x_i : \tau_i]$. For a chain $\sigma_q = \sum_i g_i\tau_i^q \in C_q(K)$, a *chain membrane* is defined as a collection of cells with multiplicities and orientations. We will use the same symbol σ_q to represent a chain membrane.

Definition 2.2 *A chain multiset α_q of dimension q is a collection of charged multisets $\alpha_q = \bigoplus_i [x_i^{g_i} : \tau_i^q]$. The collection of $\{\alpha_q\} = \Gamma_q(C_q)$ is a group under the operator $\otimes$ with identity element λ.*

We can also define membrane structures corresponding to the group $Z_q(K), B_q(K)$ and obtain multiset groups $\Gamma_q(Z_q), \Gamma_q(B_q)$. Now we consider the homology group $H_q(K)$. Similarly we get

Definition 2.3 *The homology multiset group $\Gamma_q(H_q)$ is the quotient group $\Gamma_q(H_q) = \Gamma_q(Z_q(K))/\Gamma_q(B_q(K))$.*

Now we consider the example as shown in Fig 2.1. For simplicity, we use, for example, 314 to represent the two dimensional cell $a_3a_1a_4$. Hence the set of two dimensional cells are

$$\begin{bmatrix} 634 & 314 & 415 \\ 512 & 265 & 623 \end{bmatrix} \tag{2.7}$$

There are 12 edges and 6 vertices. By the face relationship they form a lattice. as shown in Fig 2.2.

3 A Morse P System Model

In this section, we propose a Morse P system model based on the membrane structure described in Section 2. We will assume that the graph G is the integer group Z. K is a simplicial complex with dimension k. $K = \{\tau_i^q : i = 1, \cdots, q = 0, \cdots, k\}$ is composed of m simplices each of which is oriented. We only assume that these simplices are oriented in some way. The set $\{\tau_i^q : i = 1, \cdots, q = 0, \cdots, k\}$ is called the base of K. Membrane corresponding to a base cell is called a base membrane.

First we define the concept of links between simplices with same dimension, say, in K_q. If $\tau_1, \tau_2 \in K_q$ are incident, we say there is an upper link (channel) between τ_1, τ_2. If τ_1, τ_2 are neighbors, we say there is an lower link between them. An upper link is denoted by $(\tau_1, \tau_2)^\tau$ where τ is their (common) parent, while lower link is written as $(\tau_1, \tau_2)_\tau$ where τ is their common hyperface. Upper link is also written by $(\tau_1, \tau_2)^U$, while lower link is denoted by $(\tau_1, \tau_2)_L$. Links have no directions. Thus (τ_1, τ_2) and (τ_2, τ_1) are identical.

3.1 A Morse P System Model

Suppose K is a simplicial complex of dimension k. For $0 \le q \le k$, we use $\mathscr{G}_p$ to represent one of the three groups $Z_p(K), B_q(K)$ and $H_q(K)$, and we omit the symbol K for simplicity. We use $\mathscr{G}$ to denote the collection of groups

$$\mathscr{G} = \{\mathscr{G}_0, \mathscr{G}_1, \cdots, \mathscr{G}_k\} \tag{3.1}$$

The multiset group $\Gamma_q = \Gamma_q(\mathscr{G}_q)$. And define Γ by

$$\Gamma = \{\Gamma_0(\mathscr{G}_0), \Gamma_1(\mathscr{G}_1), \cdots, \Gamma_k(\mathscr{G}_k)\} \tag{3.2}$$

Definition 3.1 *A Morse P system on a simplicial complex K, with chain rules is a construct*

$$\Pi = (O, \Theta, \mathscr{G}, \Gamma, \Omega, \mathfrak{R}, \mathscr{F}, \Upsilon, i_0) \tag{3.3}$$

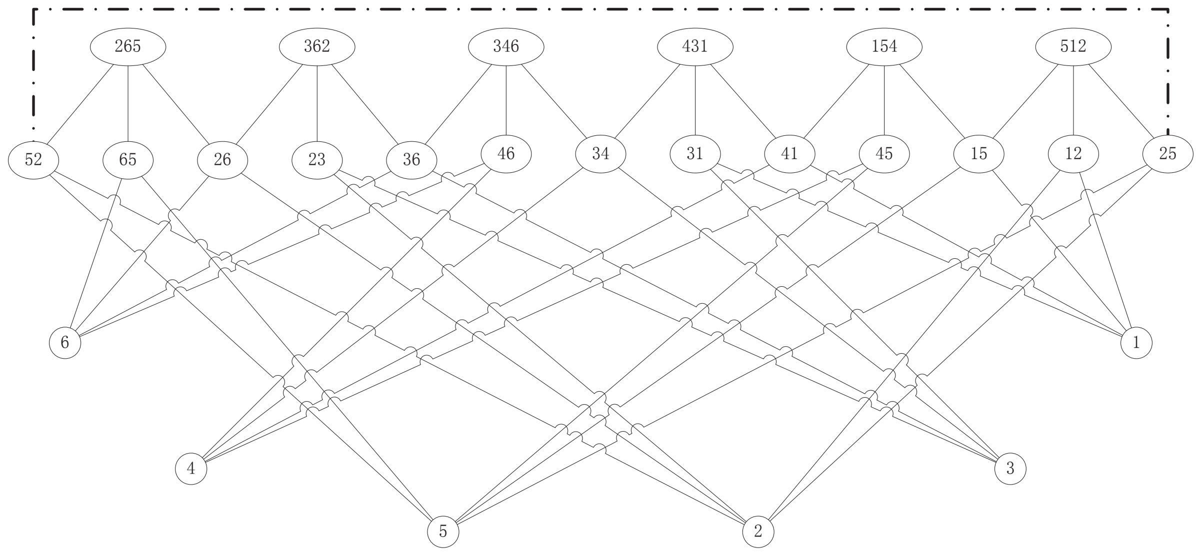

Figure 2.2: A membrane lattice structure.

Now we describe the construct Π in detail. Similar to traditional P system, O is the alphabet without charges, and Θ is the charged alphabet. $\mathscr{G}$ is the membrane structure and Γ is the multiset structure. i_0 is the output signal. The symbol Ω is the initial configuration of base membranes as follows:

$$\Omega = \{\omega_{q,i} \in \Theta^* : i = 1, \cdots, q = 0, \cdots, k\} \quad (3.4)$$

$\mathfrak{R} = \{R_1, \cdots, R_m\}$ is the set of unitary symport and antiport rules associated with each of the base membranes, where m is the total number of simplices. $\mathscr{F} = \{F(i,j) : (i,j) \in \Upsilon\}$ is the set of binary rules where Υ is the set of links. $\Upsilon \subseteq \{(i,j)^U, (i,j)_L : i,j \in \{1, \cdots, m\}\}$.

Now we consider the initial configuration of chain membranes. For a chain membrane $\sigma_q = \sum_i g_i \tau_i^q$ where τ_i^q are base membranes, the initial chain multiset is defined as

$$\alpha_q = \bigoplus_i [\omega_{q,i}^{g_i} : \tau_i^q] \quad (3.5)$$

The chain multiset α^q is called the induced chain by base multisets. Therefore we can always obtain an induced chain multiset whenever the base multisets are indicated.

3.2 Rules of Morse P Systems

Now we describe the communication rules of chain P systems. For our purpose in this paper, there are mainly four types of communication rules in a simplicial P system. Each type of rules has target one or more of the four operators $out, in, up, down$. It is important to note that rules are executed on base membranes.

First we consider the rule set $\mathfrak{R} = \{R_i\}$ where the rules R_i act on a base membrane τ_i^q. If the dimension of the current membrane is k, then there is no parent. In the case when $q < k$, there may exists none, unique, or multiple parents. Suppose one of the parents of τ_i^q is τ_i^{q+1}. Then symport rules on τ_i^q are (x, in) or (x, out). Antiport rules are in the form $(x, out; y, in)$.

To be specific, let $\tau_1 \prec \tau$. The antiport rule $[x, out; y, in | \tau_1 \prec \tau]$ in τ_1 means that exchanging multiset x inside membrane τ_1 with the multiset y outside it (in τ). The symport rule $[x, out | \tau_1 \prec \tau]$ sends the multiset x outside τ_1. The symport rule $[x, in | \tau_1 \prec \tau]$ works similarly.

$$\begin{array}{l} [x, out; y, in | \tau_1 \prec \tau], \\ [x, out | \tau_1 \prec \tau], \\ [y, in | \tau_1 \prec \tau] \end{array} \quad (3.6)$$

Next consider binary rule set $\mathscr{F}$. First suppose $(\tau_1, \tau_2)_\tau^U$ is an upper link where τ is the coface of τ_1, τ_2. Clearly this coface (parent) is unique. A rule like $[(x,y), up]^U$ means that the multiset x and y from τ_1 and τ_2 transform into z and go up to their parent τ. At the same time y in τ is transformed into u, v and sent to τ_1, τ_2 respectively. This upper link rule in $R^\tau(\tau_1, \tau_2)$ is shown as:

$$[(x,y), up; (u,v), in]^U \longrightarrow [y, in; \gamma, down] \quad (3.7)$$

An lower link rule in $R_\tau(\tau_1, \tau_2)$ may have the following forms

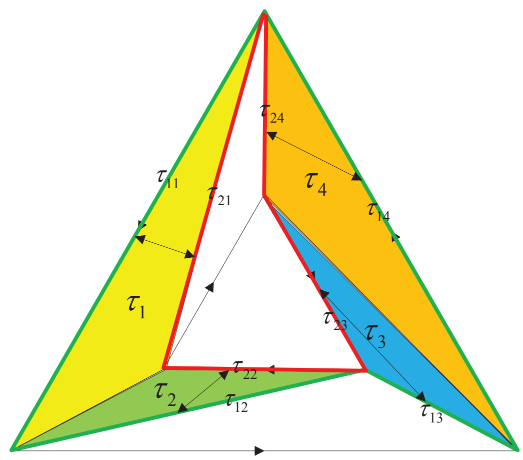

Figure 3.1: An example of upper link rules execution on chain multisets.

$$[(x, y), down; (u, v), in]_L \longrightarrow [z, in; y, up] \quad (3.8)$$

Now suppose there are two chain membranes $\sigma_1^q = \sum_i g_{1,i}\tau_{1,i}^q$ and $\sigma_2^q = \sum_i g_{2,i}\tau_{2,i}^q$. The corresponding chain multisets are $\alpha_1^q = \oplus_i[\omega_{q,i}^{g_{1,i}} : \tau_{1,i}^q]$ and $\alpha_2^q = \oplus_i[\omega_{q,i}^{g_{2,i}} : \tau_{2,i}^q]$ When upper link rules act on them, we must select incident pairs of the base membranes. Then the multisets change according to the rules associated to the base membranes. Fig 3.1 shows an example of rules acting on chains.

3.3 Configurations and Computations

Now we describe the configurations and computations of chain P systems. A *configuration* of a chain P system is the state of the system described by specifying the objects and rules associated to each base membrane. The initial state is called *initial configuration*. Therefore, the multisets represented by $\Omega = \{\omega_{q,i}, 1 \le i \le ...I_q, 0 \le q \le k\}$ in Π, constitute the initial configuration of the system.

The system evolves by applying rules in the base membranes and this evolution is called *computation*. The computation starts with the multisets specified by the base membranes. Chain membrane obtain chain multiset as configuration automatically. In each time unit, rules are used in a cell. If no rule is applicable for a cell, then no object changes in it. The system is synchronously evolving for all cells.

When the system has reached a configuration in which no rule is any longer applicable, we say that the computation *halts*. A configuration is *stable* if, even if some rules are still applicable, their application does not change the object content of the membranes. The computation is successful if and only if it halts, or it is stable. The result of a halting/stable computation is the number described by the multiplicity of objects present in the cell i_0 in the halting/stable configuration.

4 Computation Analysis of Morse P Systems

In this section, we will prove that the Morse P system is computational complete by means of simulation of register machines. Register machines which compute all sets of numbers are Turing computable. They characterize recursively enumerable language (RE). Therefore, register machines are computational complete. By a successful simulation of register machines we can show that the proposed Morse P system is computational complete as well.

A register machine is a construct $M = (m, H, l_0, l_h, I)$ where m is the number of registers, H is the set of instruction labels, l_0 is the start label, l_h is the halt label (assigned to instruction HALT), and I is the set of instructions. Each label from the label set H labels only one instruction of I [22]. Therefore a unique identification between H and I exists. An instruction from the instruction set I can have one of the following forms:

$l_i : [ADD(r), l_j, l_k]$ (add one to register r and then go to one of the instructions with labels l_j, l_k non-deterministically chosen);

$l_i : [SUB(r), l_j, l_k]$ (if register r is non-empty, then subtract one from r and go to the instruction with label l_j, otherwise go to the instruction with label l_k);

l_h: HALT (the halt instruction).

A register machine M generates a subset $N(M)$ of N in the following way. First M starts with all registers empty. Then we execute the instruction with label l_0 and proceed to apply instructions as indicated by the labels (and as made possible by the contents of registers). If M reaches the halting instruction while containing the number n in the first register, then n is an element of $N(M)$ [4]. Deterministic counter machines with ADD instructions of the form $l_i : [ADD(r), l_j]$ working in the accepting mode are known to be equivalent with Turing machines.

Consider an arbitrary deterministic register machine $M = (m, H, l_0, l_h, I)$. A Morse P system for simulating the register machine M is a construct:

$$\Pi = (O, \Theta, \mathscr{G}, \Gamma, \Omega, \mathfrak{R}, \mathscr{F}, \Upsilon, i_0) \quad (4.1)$$

An integer c in the register machine is denoted by a_+^c in the P system. We consider a Morse P system of degree 3 to simulate an add instruction. We put the initial object a_+ or a_- into the start label l_0 which contains an ADD or SUB instruction. The membrane structure is $([]_{l_i}, []_{l_j}) \prec []_r$, where $[]_{l_i}$ is the membrane of ADD instruction l_i. Membrane $[]_r$ represents the register machine r. Membrane $[]_{l_j}$ and $[]_{l_k}$ stand for instructions l_j and l_k.

The set of rules $\mathfrak{R}$ consists of the following.

$R_1 = [a_+, out; a_+, in | l_i \prec r]$
$R_2 = [(\lambda, \lambda), up; (\lambda, a_+), in]^U$
$\quad \to [\lambda, in; \lambda, down | (l_i, l_j) \prec r]$
$R_3 = [(\lambda, \lambda), up; (\lambda, a_-), in]^U$
$\quad \to [\lambda, in; \lambda, down | (l_i, l_j) \prec r]$

R_2 and R_3 represent instruction l_j with ADD and SUB instruction respectively. The initial instruction labeled with l_0 is an ADD instruction. Assume that we are in a step when we simulate an instruction $l_i : [ADD(r), l_j]$, with a_+ present in membrane l_i and no objects in any other membranes, except those objects associated with registers. Since we have a_+ inside, l_i gets rule R_1 triggered. Therefore one a_+ is added to the register r. Now we consider the case when l_j is an ADD instruction. Then the rule R_2 executes subsequently. The two membrane l_i and l_j will communicate with register r. Therefore the contents of l_j are $\{a_+\}$. Rules associated with instruction l_j will be initiated next. In this way, the number of copies of a_+ contained in the membrane $[]_r$ is identical to the counter number of register machine r. Similarly in the case when l_j is a SUB instruction, then the instruction R_3 will executes instead. And the contents of l_j are $\{a_-\}$, which will trigger SUB instruction l_j.

Next we use a Morse P system of degree 4 to simulate a sub instruction. Suppose the initial object is a_- in l_i. The membrane structure is designed as follows.

$$\begin{cases} ([]_{l_i}, []_{l_j}) \prec []_r \\ []_{l_i}, ([]_{l_i}, []_{l_k}) \prec []_r \end{cases} \tag{4.2}$$

In the equation we use $[]_{l_i}$ to denote the membrane of sub instruction l_i. Membrane $[]_r$ represents register machine r. Membrane $[]_{l_j}$ and $[]_{l_k}$ stand for instructions l_j and l_k.

The set of rules $\mathfrak{R}$ consists of the following.

$R_1 = [a_-, out; a_-, in | l_i \prec r]$
$R_2 = [(\lambda, \lambda), up; (\lambda, a_+), in]^U$
$\quad \to [a_+, in; a_+, down | (l_i, l_j) \prec r]$
$R_3 = [(\lambda, \lambda), up; (\lambda, a_+), in]^U$
$\quad \to [\lambda, in; \lambda, down | (l_i, l_j) \prec r]$
$R_4 = [(\lambda, \lambda), up; (\lambda, a_-), in]^U$
$\quad \to [a_+, in; a_+, down | (l_i, l_j) \prec r]$
$R_5 = [(\lambda, \lambda), up; (\lambda, a_-), in]^U$
$\quad \to [\lambda, in; \lambda, down | (l_i, l_j) \prec r]$
$R_6 = [(\lambda, \lambda), up; (\lambda, a_+), in]^U$
$\quad \to [\lambda, in; a_-, down | (l_i, l_k) \prec r]$
$R_7 = [(\lambda, \lambda), up; (\lambda, a_-), in]^U$
$\quad \to [\lambda, in; a_-, down | (l_i, l_k \prec r)]$

Priority: $R_2 \succ R_3, R_4 \succ R_5$

Now we simulate an instruction $l_i : [SUB(r), l_j, l_k]$, with a_- present in membrane l_i. Initially the simulation starts by R_1. With the execution of R_1 the membrane r loses one object. One object a_- appears in membrane r as $a_+ a_- \to \lambda$. If the number c stored in the register represented by membrane $[]_r$ is greater than one, then rule R_2 executes. In the case when $c = 1$, the rule R_3 will be activated instead. The priority rule $R_2 \succ R_3$ ensures that the stored number c will be always positive. The rules R_4 and R_5 have the same function as R_2 and R_3. However, they will be used for rules in l_j. R_2 and R_3 are executed in the condition that l_j is an ADD instruction, while R_4 and R_5 will be chosen when l_j is a SUB instruction. If $c = 0$ and l_k is an ADD instruction, R_6 will be triggered and rules in l_k will be executed subsequently. Otherwise, $c = 0$ and l_k is a SUB instruction, and R_7 is triggered.

5 Boltzmann Machines by Morse P Systems

A Boltzmann machine consists of a graph $K = (V, E)$ with N nodes $V = \{v_1, \cdots, v_N\}$ and the set of edges E [9]. The state of each node v_i at discrete time t is denoted by $x_i(t)$ which takes $0, 1$ as values. A connection strength w_{ij} which is a real number is set upon each edge $v_i v_j$. The graph may not be complete, and self loops are permitted. If we define $w_{ij} = 0$ when there exists no edge between v_i and v_j, then the graph K is extended to a complete graph. The matrix $W = [w_{ij}]_{N \times N}$ is then a symmetric connection strength matrix. Therefore we will assume K is complete in the following. For the purpose of this paper, we will assume that the members of W are integers and define N_s as

$$N_s = \sum_{j=1}^{N} w_{sj} - w_{ss} \tag{5.1}$$

The consensus function to be maximized is given by

$$C(t) = \sum_{1 \le i, j \le N} w_{ij} x_i(t) x_j(t) \tag{5.2}$$

Write $X(t) = [x_1(t), \cdots, x_N(t)]^T$ as the state variable. In asynchronous parallelism, when $X(t)$ moves to $X(t+1)$, only one vertex is changes. We denote by v_p the changing vertex. Then difference of consensus is given by

$$\Delta C(t) \;\; = \;\; C(t+1) - C(t)$$

$$= [1-2x_p(t)]\left(w_{pp}+\sum_{j\neq p} w_{pj}x_j(t)\right)$$

Now we define temperature $T(t)$ and acceptance probability in the case of $\Delta C(t) < 0$ with an initial probability P_0 (Often P_0 is chosen as $1/N$ and is a constant).

$$P(t) = \frac{P_0}{1+e^{-\Delta C(t)/T(t)}} \quad (5.3)$$

This can be implemented by a randomized value $\mathrm{rand}(0,1)$ and

$$\text{Accept } X(t+1) \begin{cases} \text{always, if } \Delta C(t) \geq 0 \\ \text{provided} \\ \qquad P(t) > \mathrm{rand}(0,1), \\ \qquad \text{if } \Delta C(t) < 0 \end{cases} \quad (5.4)$$

Let the initial temperature is T_0. Then the temperature $T(t)$ at time t is

$$T(t) = \frac{T_0}{1+\log(1+t)} \quad (5.5)$$

Now let r be a random variable in $(0,1)$. It is easy to see that Equation (**??**) is equivalent to the following if we choose $P_0 = 1/N$ (in the case of $\Delta C(t) < 0$):

$$|\Delta C(t)| < \max\{0, \frac{T_0 \log(\frac{1}{rN}-1)}{1+\log(1+t)}\} \quad (5.6)$$

Moreover, Equation (5.7) is equivalent (in all cases, $\Delta C(t) \geq 0$ and $\Delta C(t) < 0$):

$$\Delta C(t) > -\max\{0, \frac{T_0 \log(\frac{1}{rN}-1)}{1+\log(1+t)}\} \quad (5.7)$$

Now we use the same symbol K to denote the simplicial complex with the graph K. Then we construct the groups Z_q, B_q and H_q for $q = 0, 1, \cdots, N$ with a group $G = R$ where R is the additive group of real numbers. The base simplices are

$$\{\tau_1, \cdots, \tau_N\} \cup \{\tau_{ij} : 1 \leq i, j \leq N, i < j\} \quad (5.8)$$

Consider the P system

$$\Pi = (O, \Theta, \mathscr{G}, \Gamma, \Omega, \mathfrak{R}, \mathscr{F}, \Upsilon, i_0) \quad (5.9)$$

where the number of edges (links) are $m = \frac{1}{2}N(N+1)$, $\Omega = \{\omega_i, \omega_{ij}\}$ are initial configurations of base simplices of vertices and edges, $\mathfrak{R} = \{R_i\}$ are rules on node base membranes τ_i, $\mathscr{F} = \{F_{ij}\}$ are rules on edge membranes (links), Υ is the set of links, $i_0 = 1$ indicates the output node.

Now we define the alphabet as follows

$$\begin{aligned} O = &\{a_1, \cdots, a_N\} \cup \{a_+, a_-\} \\ &\cup\{a, b_+, b_-, c, u, v\} \\ &\cup\{\delta, \delta_s, \delta_p, \delta_+, \delta_-, \delta_0, \beta, \gamma, \gamma_+\} \\ &\cup\{\pi, \pi_0, \pi_c, \pi_h, \pi_\sigma\} \end{aligned} \quad (5.10)$$

The symbol a_i is a variable taking only two values, a_+ and a_-, indicating that the state of the corresponding vertex is on or off (value 1,0 respectively). As initialization we set vertex v_1 as the running vertex, and hence the initial configurations of node cells are

$$\begin{cases} \omega_1 = \{a_1 v\} \cup \{\delta_s, \delta^{N-1}, \delta_0^2, \gamma_+^{N-1}\} \\ \omega_i = \{a_i\} \cup \{\delta, \beta\},\ i > 1 \end{cases} \quad (5.11)$$

As initialization we set $a_i = a_+$ or a_- randomly. The initial configuration on edges are empty strings $\omega_{ij} = \{\lambda\}$.

5.1 Chain Rules of Boltzmann Machine

All the rules are categorized as several groups. The first group is annihilation rules acting on node cells

$$F_0 = \begin{cases} r_{00} : b_+^{-1} b_+ \to \lambda \\ r_{01} : [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (\delta_0\delta_+^2, \lambda) \to (\delta_0 b_+^2, \lambda) \\ \quad \succ (\delta_0\delta_+, \lambda) \to (b_+, c) \\ \quad \succ (\delta_0, \lambda) \to (\lambda, c) \\ r_{02} : [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (\delta_0\delta_+^{-2}, \lambda) \to (\delta_0 b_+^{-2}, \lambda) \\ \quad \succ (\delta_0\delta_+^{-1}, \lambda) \to (b_+^{-1}, c) \\ \quad \succ (\delta_0, \lambda) \to (\lambda, c) \\ r_{020} : \tau_1 \to \tau_1 | \delta_0\delta_+^{\pm 2} \to \delta_0 b_+^{\pm 2} \\ \quad \succ \delta_0\delta_+^{\pm 1} \to b_+^{\pm 1} c \succ \delta_0 \to c \\ r_{03} : \tau_s \to \tau_s | \beta\delta_+\delta_+ \to \beta\delta_+ \\ \quad \succ [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (\beta\delta_+, \lambda) \to (\lambda, c) \\ r_{04} : \tau_s \to \tau_s | \beta\delta_+^{-1}\delta_+^{-1} \to \beta\delta_+^{-1} \\ \quad \succ [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (\beta\delta_+^{-1}, \lambda) \to (\lambda, c) \\ r_{05} : \tau_1 \to \tau_1 | \beta\delta_+^{\pm 1} \to c \end{cases} \quad (5.12)$$

We also have a set of rules acting on node cells manipulating self loops. Suppose n is a positive integer, define an integer p as follows

$$p(n,t) = \max\{0, \left[\frac{T_0 \log(\frac{A}{nN} - 1)}{1 + \log(1+t)}\right]\} \qquad (5.13)$$

$$F_{01} = \begin{cases} r_{011} : [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (v\delta_s, \lambda) \to (\delta_p b_+^{(1-2a_s)w_{ss}} \gamma, cuv) \\ r_{012} : [\tau_s, \tau_1] \to [\tau_s, \tau_1]| \\ \quad (\delta_p, u^n v^t) \to (b_+^{p(n,t)}, cu^n v^t), \\ \quad n, t \text{ are maximal} \\ r_{013} : \tau_1 \to \tau_1 | u^A \to u \end{cases} \qquad (5.14)$$

The second group is on chain edge rules.

$$F_1 = \begin{cases} r_{11} : \tau_{ij} \to [\tau_i, \tau_j]| \\ a^{\pm w_{ij}} \to [\delta_+^{\pm w_{ij}}, \delta_+^{\pm w_{ij}}] \end{cases} \qquad (5.15)$$

The third group is rules on nodes by links where a chain $w_{ij}\tau_{ij}$ is used.

$$F_2 = \begin{cases} r_{21}^U : [\tau_s, \tau_i] \to [\tau_s, \tau_{si}, \tau_i]| \\ \quad (\delta\gamma_+ a_+, \delta) \to (\gamma, a^{(1-2a_s)w_{si}}, \lambda) \\ \quad (\delta\gamma_+ a_-, \delta) \to (\gamma, \lambda, \lambda) \\ r_{22}^U : [\tau_i, \tau_s] \to [\tau_i, \tau_{is}, \tau_s]| \\ \quad (\delta, \delta\gamma_+ a_+) \to (\lambda, a^{(1-2a_s)w_{is}}, \gamma) \\ \quad (\delta, \delta\gamma_+ a_-) \to (\lambda, \lambda, \gamma) \\ (i \neq s) \end{cases} \qquad (5.16)$$

Finally we consider controlling rules. The fourth group of rules control the active node of the neural network. These rules act on node cells.

$$F_3 = \begin{cases} r_{31}^{\rightarrow} : \\ \quad [\tau_1, \tau_2] \to [\tau_1, \tau_2]| \\ \quad (\pi_c \gamma^N, \lambda) \to (\delta\beta, \delta_s \delta^{N-1} \delta_0^2 \gamma_+^{N-1} v) \succ \\ \quad \tau_i \to \tau_i (i > 2) | a_i \to a_i \delta\beta \\ r_{32}^{\rightarrow} : \\ \quad [\tau_s, \tau_{s+1}] \to [\tau_s, \tau_{s+1}], 1 < s < N| \\ \quad (\gamma^N \pi_c, \lambda) \to (\delta\beta, \delta_s \delta^{N-1} \delta_0^2 \gamma_+^{N-1} v) \succ \\ \quad \tau_j \to \tau_j (j \neq s, s+1) | a_j \to a_j \delta\beta \end{cases} \qquad (5.17)$$

Now we discuss computation and halting conditions. The main idea is using a symbol π_0 to count the number of running nodes in one cycle and compute the number of stable nodes. When the number of stable nodes equals N, then the whole network is stable. We can send a symbol π_h to output node σ_1. Notice that rules are executed in an order.

$$F_4 = \begin{cases} r_{41}^{\rightarrow} : \tau_s \to \tau_s | \\ \quad \delta_0 \to \pi, \; b_+ b_+^{-1} \to \lambda \succ r_{42} \\ r_{42}^{\rightarrow} : \tau_s \to \tau_s | \\ \quad b_+ b_+ \to b_+, b_+^{-1} b_+^{-1} \to b_+^{-1} \succ r_{420} \\ r_{420} : \tau_s \to \tau_s | b_+ \pi a_s \to \hat{a_s} \succ r_{43} \\ r_{43}^{\rightarrow} : [\tau_s, \tau_1] \to [\tau_s, \tau_1] | (\pi, \lambda) \to (\lambda, \pi_0) \\ r_{44}^{\rightarrow} : \tau_1 \to \tau_1 | \pi_0^N c \to \pi_h (\text{halt}) \\ r_{45}^{\rightarrow} : \tau_1 \to \tau_1 | cc \to c \succ \\ \quad [\tau_1, \tau_s] \to [\tau_1, \tau_s] | (\text{continue}) \\ \quad (\pi_0^x c, \gamma^N) \to (\pi_\sigma, \gamma^N \pi_c), x < N \end{cases} \qquad (5.18)$$

Here the symbol $\hat{a_s}$ means that inverse a_s, i.e., $a_s = 1 - a_s$. Loop controlling rules are listed as F_5. If we found a symbol π_h in σ_1, then the computation is complete.

$$F_5 = \begin{cases} r_{51}^{\rightarrow} : [\tau_N, \tau_1] \to [\tau_N, \tau_1]| \\ \quad (\gamma^{N-1} \pi_c, \lambda) \to (\delta\beta, v\delta_s \delta^{N-1} \delta_0^2 \gamma_+^{N-1}), \\ \quad \tau_i \to \tau_i (i \neq 1, N) | a_i \to a_i \delta\beta \end{cases} \qquad (5.19)$$

The total number of loops is written in the symbol π_σ.

5.2 Configurations and Computations

Now we explain the process of computation in detail. We use $i_0 = 1$ to note the output cell. For initial configuration, the multisets of edges are $\{\lambda\}$ and the node cells have the following multisets.

$$\{a_1 v \delta_s \delta^{N-1} \delta_0^2 \gamma_0^{N-1}, a_2 \delta\beta, \cdots, \alpha_N \delta\beta\} \qquad (5.20)$$

The the running node of the neural network is $s = 1$. To illustrate the process more clearly, we assume that the current running node is $s = 2$ after the first round of running for $s = 1$. In the first node cell, the symbol π_σ notes the number of running rounds for the network. Hence the current configuration is

$$\{a_1 uv \delta\beta \pi_\sigma, a_2 \delta_s \delta^{N-1} \delta_0^2 \gamma_0^{N-1}, a_3 \delta\beta, \cdots, a_N \delta\beta\} \qquad (5.21)$$

Now for each $1 \leq i \leq N$ and $i \neq 2$ we run the rules in F_2 and the node cells are

$$\{a_1 \beta \pi_\sigma uv, a_2 \delta_s \delta_0^2 \gamma^{N-1}, a_3 \beta, \cdots, a_N \beta\} \qquad (5.22)$$

At the same time the edge cells σ_{2i} for $1 \leq i \leq N, i \neq 2$ are

$$\{a_2 a^{(1-2a_2)w_{21}}, \lambda, a_2 a^{(1-2a_2)w_{23}}, \cdots, a_2 a^{(1-2a_2)w_{2N}}\} \quad (5.23)$$

Next we run the rules in F_{01} and the edge cells remains the same while the node cells are

$$\{a_1\beta\pi_\sigma cu^2v^2, a_2\delta_0^2\gamma^N b_+^{(1-2a_2)w_{22}} b_+^p, a_3\beta, \cdots, \alpha_N\beta\} \quad (5.24)$$

The next step is to run F_1 on edges and the edge cells change into empty multisets and these states remain till the end of this round. At the same time the node cells are

$$\begin{array}{l}\{a_1\beta\pi_\sigma cu^2v^2\delta_+^{\pm w_{12}},\\ a_2\delta_0^2\gamma^N b_+^{\pm w_{22}}\delta_+^{\pm w_{12}}\delta_+^{\pm w_{23}}\cdots\delta_+^{\pm w_{2N}} b_+^p,\\ a_3\beta\delta_+^{\pm w_{23}}, \cdots, a_N\beta\delta_+^{\pm w_{2N}}\}\end{array} \quad (5.25)$$

Now we run rules $\{r_{01}, r_{02}, r_{020}, r_{03}, r_{04}, r_{05}, r_{00}\}$ in F_0 and count the number of c in τ_1.

$$\{a_1\pi_\sigma c^{N+1}u^2v^2, a_2\gamma^N\delta_0 b_+^{0;\pm 1}, a_3, \cdots, \alpha_N\} \quad (5.26)$$

Now we check if there exists one remaining b_+ and if the answer is yes, then we accept this step and let the status of a_2 be inverted:

$$\{a_1\pi_\sigma c^{N+1}\pi_0^x u^2v^2, a_2\gamma^N, a_3, \cdots, \alpha_N\} \quad (5.27)$$

Actually in one round of running we only accept one possible inverted node. In the case of none inverted node, we obtain a stable node which is noted by the symbol π_0. If after all the N nodes are processed we obtain π_0^N stable nodes, then the network is stable. We then send a symbol π_h to the output node the computation is complete. Otherwise we send a continuing symbol π_c to σ_1 and continue to the next step. The function of rules F_3 and F_5 is to control the running of current node to the next node.

Computation is listed in Table 5.1 as an algorithm.

The next figure (Fig 5.1) illustrate the whole computation process when $N = 4$ and $i = 2$.

6 Cluster Analysis by Morse Membrane Boltzmann Machines

In this section we will present an example of cluster analysis to show the effectiveness of the proposed Morse membrane Boltzmann machines. The data set to be clustered is chosen from the Euclidean space R^n.

Table 5.1: A chain membrane Boltzmann algorithm

Inputs: $O, \Theta, \mathscr{G}, \Gamma, \Omega, \mathscr{F}, \Upsilon, i_0$

Outputs: Stable status.

Begin

Set $t = 1, u = 1$, running node $s = 1$, a large integer A, initial temperature T_0.

while (condition **) do**

  for $s = 1$ **to** N

    if ($u > A$ **) set** $u = 1$

    end

    $r = u/A; u = u + 1; p = \max\{0, \left[\frac{T_0\log(\frac{1}{rN}-1)}{1+\log(1+t)}\right]\}$

    for $i = 1$ **to** N

      (1) Apply rules $F_2 = \{r_{21}^U, r_{22}^U\}$ on $[\tau_s, \tau_{si}, \tau_i]$ if $i \neq s$ to compute consensus. (2) Apply rules $F_{01} = \{r_{011}, r_{012}, r_{012}\}$ on $[\tau_s, \tau_1]$ for $i = s$ to compute consensus for self loop. Add comparison multiset by r_{012}. (3) Apply rule $F_1 = \{r_{11}\}$ on τ_{si} for $i \neq s$ to send computing result to node cells. (4) Apply rules $F_0 = \{r_{00}, r_{01}, r_{02}, r_{020}, r_{03}, r_{04}, r_{05}\}$ on τ_s to summarize result to computing node s. (5) Apply rules $F_4 = \{r_{41}, r_{42}, r_{420}, r_{43}\}$ on τ_s to count the number of stable nodes. (6) Apply controlling rules $F_4 = \{r_{44}, r_{45}\}$ on τ_1, τ_s. (7) Apply loop controlling rules $F_3 = \{r_{31}, r_{32}\}$ on τ_s, τ_{s+1}. (8) Apply loop controlling rule $F_5 = \{r_{51}\}$ to loop from $s = N$ to $s = 1$.

    end

  end

  $t = t + 1$

end

End

We will use the nearest-neighbor clustering algorithm which is a hierarchical clustering scheme. The idea of the algorithm is to find a shortest path as the clustering result from a weighted connected undirected graph $K(V, E)$. This graph is generated from the data set according to some criteria. This shortest path should be chosen as minimal in length while it connects all the vertices.

Whenever such a shortest path is found, this path can be broken into several pieces by removing certain edges. Each piece will correspond to one cluster. In this way, vertices belonging to the same cluster will connect more closely. The edge removing criteria is determined according to some predefined threshold value. By this criteria, edges with weights beyond the

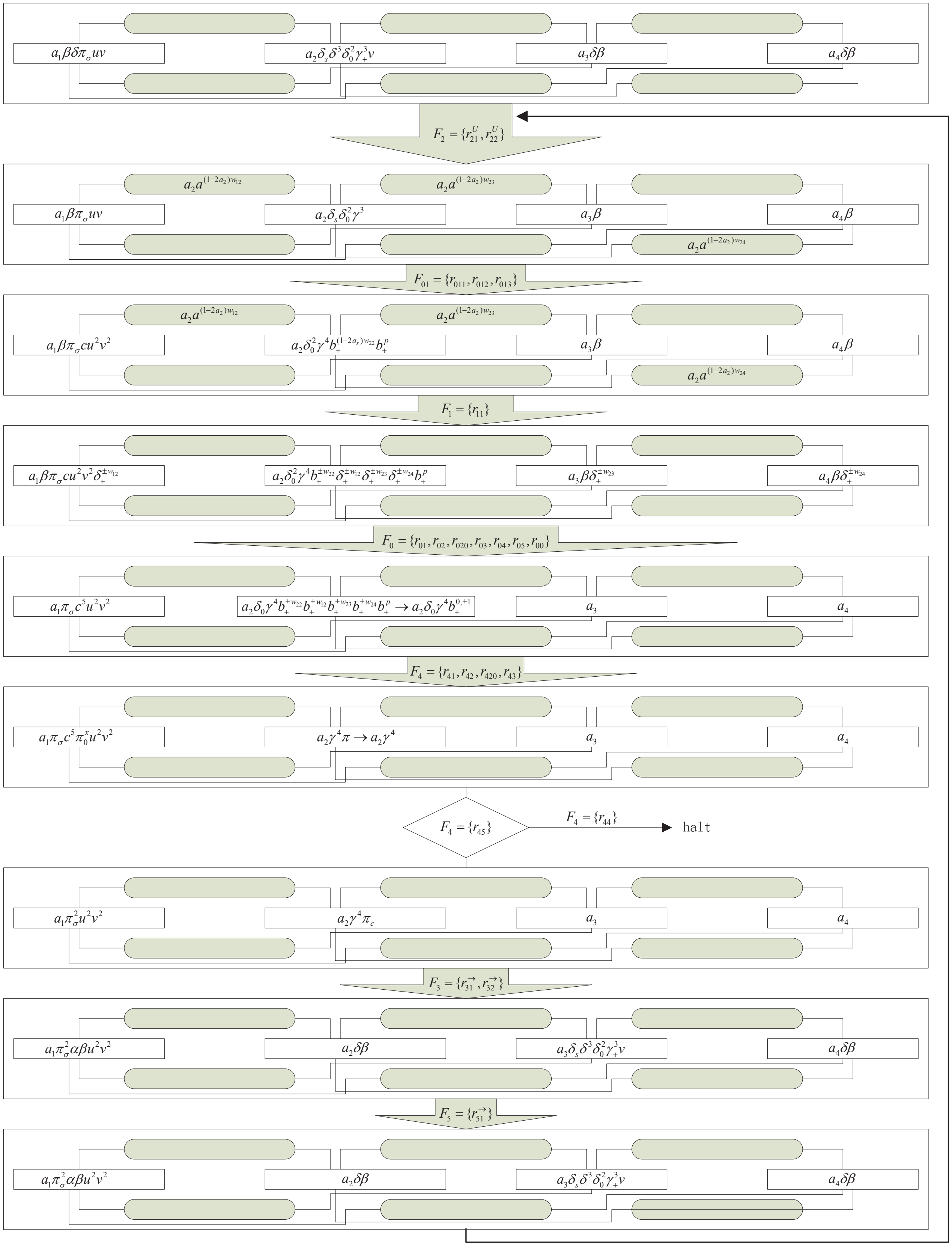

Figure 5.1: A flow chart of chain P system rule execution.

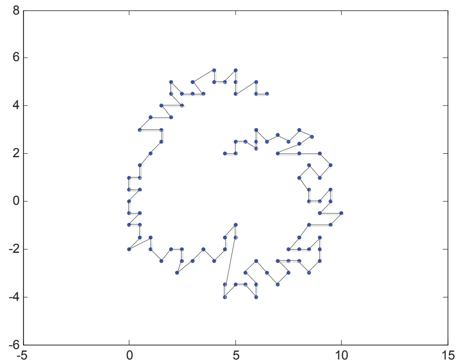

Figure 6.1: the shortest path of the 99 vertices

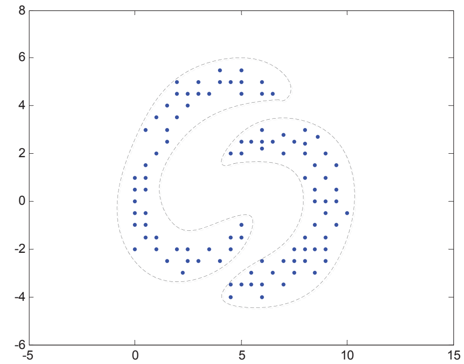

Figure 6.2: the clustering result of the data set

threshold will be cut off. In this way, the clustering problem is transformed into a path finding combinatorial problem.

Many well-known combinatorial optimization problems can be solved by Boltzmann machines. The clustering problem can also be directly mapped onto the structure of a Boltzmann machine by proper chosen of connections. Cost function can also be transformed into the consensus function with the help of specific strength definitions.

Now we proceed to construct a Boltzmann machine model to solve our clustering problem. In order to simplify the problem settings, we assume that the corresponding shortest path problem is a 0-1 integer problem. The state of a vertex v_i at time t is denoted by x_i. Let ω_{ij} be the connection strength of the edge (v_i, v_j).

The consensus function to be constructed must satisfy the condition that its maximal points will map to shortest paths of the graph. In order to do this, the consensus function will be designed as monotone with respect to the path length. Therefore the strength variables are set to be negative $-\omega_{ij}$. Finally the consensus function is defined as follows:

$$C_t = -\sum_{(v_i,v_j)\in E} \omega_{ij} x_i x_j \tag{6.1}$$

In the above equation, E is a disjoint set of edges, $E = \{(v_{i\mu}, v_{jv}) | (i \neq j) \wedge (\mu = (v+1) \bmod n)\}$.

Example one of the Morse P system based Boltzmann machine model for the clustering problem is carried out for a data set with 99 patterns (Fig. 6.1). Result for the clustering of a Morse P system based Boltzmann machine is shown in Fig.6.3. The corresponding results of data set is shown in Fig.6.2.

Next, we use the 5 and 6 dimensions of dataset seeds (Fig. 6.4) from the UC Irvine Machine Learning Repository to verify the effectiveness of our algorithm. The dataset is an examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each with 7 attributes (area A, perimeter P, compactness $C = 4 * pi * A/P^2$, length of kernel, width of kernel, asymmetry coefficient, length of kernel groove) randomly selected for the experiment. High quality visualization of the internal kernel structure was detected using a soft X-ray technique. Studies were conducted using combine harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin. The data set is used in clustering and classification frequently. In our algorithm, seeds is grouped into three clusters in Fig. 6.5. There are 21 patterns going to a wrong cluster and 15 patterns missing. The correct rate is 82.4%.

7 Conclusion

In this paper, a new kind of P system on chain structure is presented. We provide Morse membrane structures on complexes, objects with positive and negative charges and communication rules on chains. We simulate the register machine successfully by using 3 membranes, 3 rules on add instruction and 4 membranes, 7 rules on sub instruction. 5 parts of rules are provided in the implementation of Boltzmann Machines. Cluster analysis on Morse P system based Boltzmann Machines shows its effectiveness by two examples.

Acknowledgements: The research was supported by the Natural Science Foundation of China (No.61472231,61170038,71071090). Other grants:

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

1-9

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

10-19

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

20-29

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

30-39

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

40-49

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

50-59

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

60-69

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

70-79

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{matrix}$$

80-89

$$\begin{matrix}0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1\end{matrix}$$

90-99

Figure 6.3: Result for the clustering of a Morse P system based Boltzmann machine

201401202, 12YJA630152, 11CGLJ22.

References:

[1] Rozenberg G. Păun and A. Salomaa (eds.), *Handbook of Membrane Computing,* Oxford University Press, Cambridge, 2010.

[2] Xiangxiang Zeng, Tao Song, Xingyi Zhang and Linqiang Pan, Performing Four Basic Arithmetic Operations With Spiking Neural P Systems, *IEEE transactions on nanoscience*, 11:4, 2012, pp.366–374.

[3] Tao Song, Linqiang Pan and G. Păun, Asynchronous spiking neural P systems with local synchronization, *Information Sciences*, 219, 2012, pp.197-C207.

[4] Krithivasan Rama R. Kamala, *Introduction to Formal Languages, Automata Theory and Computation,* Pearson Education India, 2009.

[5] G. Păun and R. Păun, Membrane computing and economics: numerical P systems, *Fundamenta Informaticae*, 73(122), 2006, pp.213–227.

[6] Bogdan Aman and Gabriel Ciobanu, Behavioural Equivalences in Real-Time P Systems, *CMC14*, Chisinau, Moldova, 2013, pp.49–62.

[7] H. Adorna, Gh. Păun and M. Prez Jimnez, On Communication Complexity in Evolution-Communication P systems, *Romanian Journal of Information Science and Technology*, 13(2), 2010, pp.113–130.

[8] Ciprian Dragomir, Florentin Ipate, Savas Konur, Raluca Lefticaru and Laurentiu Mierla, Model Checking Kernel P Systems, *CMC14*, Chisinau, Moldova, 2013, pp.131–152.

[9] E.H.L. Aarts and Jan H.M. Korst, Boltzmann machines and their applications, *Lecture Notes in Computer Science*, Volume 258, 1987, pp.34–50.

[10] Cardona Mónica, M. Angels Colomer, Mario J. Pérez-Jiménez, and Alba Zaragoza, Hierarchical clustering with membrane computing, *Computing and Informatics*, vol.27(3+), 2008, pp.497–513.

[11] Jianwen Feng, Jingyi Wang, Chen Xu, and Francis Austin, Cluster Synchronization of Nonlinearly Coupled Complex Networks via Pinning Control, *Discrete Dynamics in Nature and Society*, Volume 2011, Article ID 262349, doi:10.1155/2011/262349.

[12] Forman Robin, Morse Theory for Cell Complexes, *Advances in Mathematics*, vol.134, 1998, pp.90–145.

[13] D.H. Ackley, G.E. Hinton and T.J. Sejnowski, A Learning Algorithm for Boltzmann Machines, *Cognitive Science*, 9, 1985, pp.147.

[14] D.H. Ackley, G.E. Hinton and T.J. Sejnowski, A learning algorithm for Boltzmann machine, *Cognitive Science*, vol. 9, 1985, pp.147–169.

[15] J. Han and M. Kamber, *Data Mining, Concepts and Techniques*, Higher Education Press, Morgan Kaufmann Publishers, Beijing, 2002.

[16] Yusuke Hosoi, Yuta Taniguchi and Daisuke Ikeda, Replacing Log-Based Profiles to Context Profiles and Its Application to Context-aware Document Clustering, *WSEAS Transactions on Information Science and Applications*, vol.11, 2014, pp.51–60.

[17] Z.H. Jiang, *Introduction to Topology*, Shanghai Science and Technology Press, Shanghai, 1978.

[18] Păun Gheorghe, A quick introduction to membrane computing, *The Journal of Logic and Algebraic Programming*, vol.79, 2010, pp.291–294.

[19] G. Păun, G. Rozenberg, and A. Salomaa, *Membrane Computing*, Oxford University Press, New York, 2010.

[20] J.H. Xiao, X.Y. Zhang and J. Xu, A membrane evolutionary algorithm for DNA sequence design in DNA computing, *Chinese Science Bulletin*, vol.57:6, 2012, pp.698–706.

[21] Liping Zhang and Haibo Jiang, Impulsive Cluster Anticonsensus of Discrete Multiagent Linear Dynamic Systems, *Discrete Dynamics in Nature and Society*, vol.2012, Article ID 857561, doi:10.1155/2012/857561.

[22] I. Korec, Small universal register machines, *Theor Comput Sci*, 168, 1996, pp.267–301.

This document is typeset by LaTeX

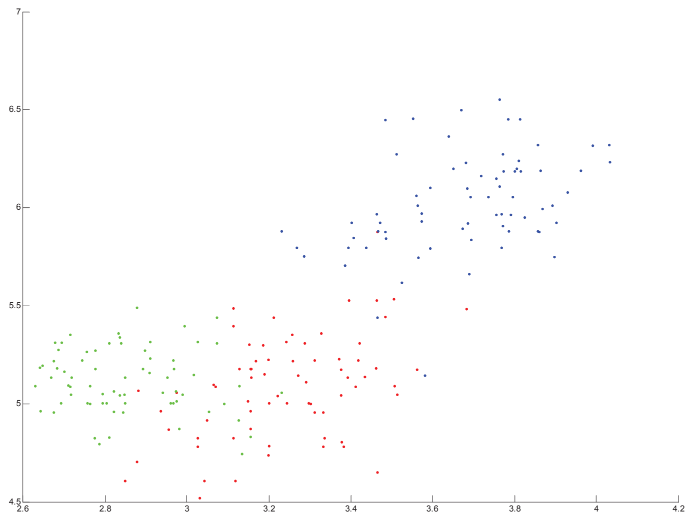

Figure 6.4: The 5 and 6 dimensions of data set seeds

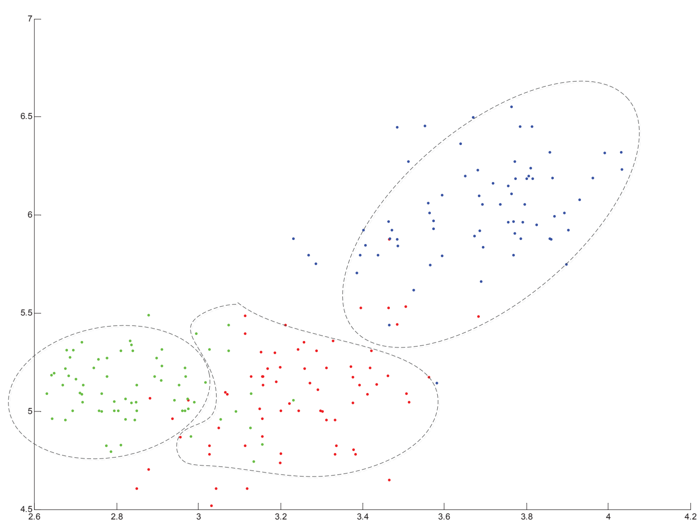

Figure 6.5: Clustering result of 5 and 6 dimensions of data set seeds