Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 629–632
Progress in Natural Science
www.elsevier.com/locate/pnsc

Special report

Maximally parallel attribute on P Systems: Properties and applications

Chunyi Huang, Xiaoju Dong *

BASICS, Shanghai Jiao Tong University, Shanghai 200240, China

Received 21 December 2007; received in revised form 11 January 2008; accepted 17 January 2008

Abstract

In this paper, we investigate the maximally parallel attribute of P Systems. Some properties of P Systems are introduced, which are the filter property and the enumeration property. The two properties are applied to solving the sorting problem and the Hamilton cycle problem, respectively.

Keywords: Membrane computing; P Systems; Maximally parallel

1. Introduction

Membrane computing is a bio-inspired branch of natural computing introduced by Paŭn [1], abstracting computing models from the structure and functioning of living cells and from the organization of tissues or other higher order structure. The concrete computing devices of this model are called P Systems. The basic ingredient of the P Systems is the hierarchical membrane structure, in which lots of membranes are embedded hierarchically (Fig. 1). Each membrane forms a compartment, where multisets and evolution rules are placed. The multisets evolved step by step following the corresponding evolution rules in the same compartment. The evolution implementation stops until no rules can be used and we call that time point P Systems' halting. As a consequence, those evolution sequences are considered to be the computation of P Systems, while the multisets presented in the output membrane are considered to be the result of computation.

In this paper, we mainly investigate the maximally parallel attribute of the P Systems. Generally speaking, this attribute has two-folded meanings – rules maximally parallel and compartments maximally parallel. The rules maximally parallel refers to that any applicable rules must be used to the corresponding multisets during one-step implementation of P Systems; while the compartments maximally parallel refers to that the compartments are not mutually influenced by each other during the evolution of the P Systems – namely, they all evolved independently. In this study, we will show two properties of this attribute, and use these two properties to solve two corresponding problems, respectively.

2. Some general prerequisites

2.1. Rewriting P System [1,2]

A rewriting P System of degree n, $n \geqslant 1$, is a construct

$$\Pi = (V, M, W_1, \ldots, W_n, (R_1, P_1), \ldots, (R_n, P_n), i_0)$$

where V is an alphabet whose elements are called objects; M is a membrane structure of degree n, with the membranes and the regions labelled in a one-to-one manner with elements in a given set A; W_i, $1 \leqslant i \leqslant n$ are finite languages over V^*, representing multisets over V associated with the regions $1, 2, \ldots, n$ of M; R_i, $1 \leqslant i \leqslant n$, are finite sets of context-free evolution rules of the form $X \to v(tar)$, with $X \in V$, $v \in V^*$,

* Corresponding author. Tel.: +86 21 34205060.
E-mail address: dong-xj@cs.sjtu.edu.cn (X. Dong).

1002-0071/$ - see front matter © 2008 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2008.01.006

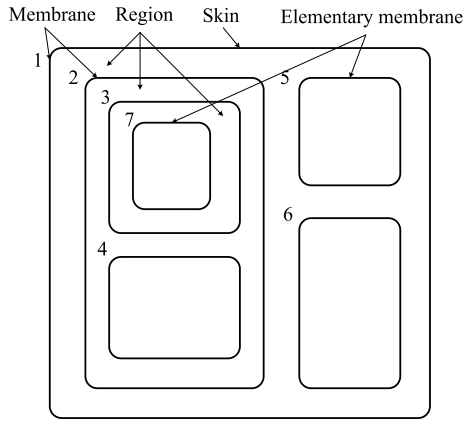

Fig. 1. P System structure.

$tar \in \{hear, out\} \cup \{in_j | 1 \leqslant j \leqslant n\}$, over V associated with the regions $1, 2, \ldots, n$ of M; P_i is a partial order relation over R_i, $1 \leqslant i \leqslant n$, specifying a priority relation among rules of R_i. The length of u is called the radius of the rule $u \to v$. i_0 is a number between 1 and n, which specifies the output membrane of Π. In this study, we will use rewriting P Systems to solve some hard problems and analyze its attribute.

2.2. Sorting problem

Sorting problem is a very common computing problem. We present it here for the purpose of emphasizing its time complexity. The input of sorting problem is a linear list in the form of $\langle A_1, A_2, \ldots, A_n \rangle$, while the output of it is an ordered list also in the form of $\langle B_1, B_2, \ldots, B_n \rangle$ after some transform steps implemented on the input. There existed a large number of sort algorithms for this problem such as quick-sort and bubble-sort. However, these algorithms are all based on comparison, which also had a lower-bound time complexity $O(n \log n)$.

2.3. Hamiltonian cycle problem [3]

This problem can be described as follows: given a graph G, decide if there is a Hamiltonian cycle. (A Hamiltonian cycle is a cycle which visits every vertex exactly once and returns to the start point.) This problem is an NP-Complete problem. It is in NP, and it is polynomial reducible from the SAT problem (which is proved as the first NP-Hard problem).

2.4. Time complexity in P Systems

As to the turing machine theory, the running time [4] is defined as follows:

Let $f : \{0,1\}^* \to \{0,1\}^*$ and let $T : N \to N$ be some functions. We say that a TM M computes function f if for every $x \in \{0,1\}^*$, if M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape. We say M computes f in $T(n)$-time if for all n and all inputs x of size n, the running time of M on that input is at most $T(n)$.

As we can see, common time complexity lies on the steps implemented by the turing machine. In this study, we give a presumption that time complexity in P Systems is based on the maximally parallel steps implemented by the membranes in the P Systems. We give this presumption for the reason that one maximally parallel implementation step is inseparable and the running time of each step is always the same [5].

3. Filter property

We summarize a filter property from this maximally parallel attribute of P Systems. This property can be explained as that data could be filtered by using a maximally parallel attribute to accomplish some corresponding work.

Property 1 (Filter property)**.** *The maximally parallel implementation in the P System can be regarded as a filter, used to filter computing data automatically and in parallel.*

The meaning of this property is that a bind of data go through the filter, filtering different kinds of data at a time. This process is in parallel and it is the same as our daily filtering staff. We will use this property to solve the sorting problem quickly [6], with a best condition of time complexity O(n) (Fig. 2).

This P System begins with the initial multisets in $A_1^{m_1} A_2^{m_2} A_3^{m_3} \ldots A_n^{m_n}$, which is the input of this P System, while the upper numbers $m_1, m_2, \ldots, m_n$ ($m_i \neq m_j$, $1 \leqslant i \neq j \leqslant n$) represents the sorting list. In the first step, the rule $A_1 A_2 A_3 \ldots A_n \to (B_1, out)$ fired, m_i number of $A_1 A_2 A_3 \ldots A_n$ will be exhausted while m_i is the smallest number in the sorting list $\langle m_1, m_2, \ldots, m_n \rangle$. At this time, the number of $A_1 A_2 A_3 \ldots A_n$ will be decreased to $(m_1 - m_i), (m_2 - m_i), \ldots, (m_n - m_i)$, the number of A_i will decrease to 0. Then, we analyze the rule $D_1 A_2 A_3 \ldots A_n \to (B_2, out)$, its usage is the same as $A_1 A_2 A_3 \ldots A_n \to (B_1, out)$, that is, to make the object A_j which has the smallest number disappears in one-step maximal implementation. Apparently, after $n - 1$ steps of these kinds

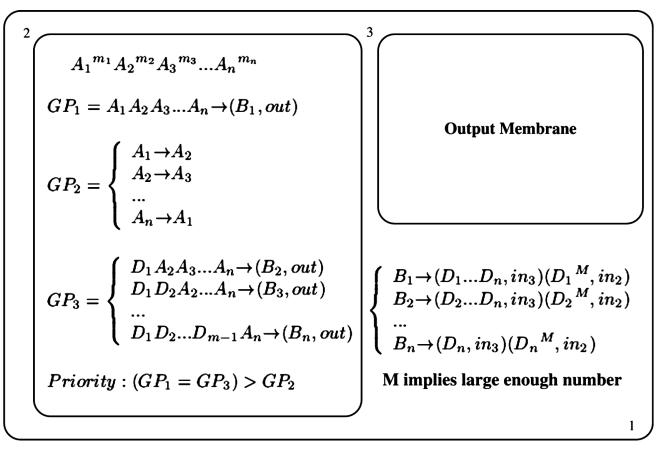

Fig. 2. P System that solves sorting problem.

of implementation, all the left $n-1$ numbers will be filtered out, and the sorting list will be sorted.

The rules $(A_1 \to A_2, A_2 \to A_3, \ldots, A_n \to A_1)$, which we also called cycle rules, are used to transform the multisets with the form $A_1A_2 \ldots A_{i-1}A_{i+1} \ldots A_n$ into the form $A_2A_3 \ldots A_n$, so that the rule in the form of $D_1A_2A_3 \ldots A_n \to (B_2, out)$ could be satisfied. These rules in membrane 1 are not difficult to understand since they are used to transit the smallest number sent out from membrane 2 into membrane 3 step by step. After n steps' implementation, membrane 3 will get the computing result in the form of $D_1^{h_1}D_2^{h_2}D_3^{h_3} \ldots D_n^{h_n}$. The n-tuple $\langle h_1, h_2, \ldots, h_n \rangle$ represents the sorted list.

3.1. Efficiency analysis and some improvement

In the above section, we have made a presumption that the performance of the P System lies mainly on the maximally parallel implementation steps it had processed in one computation. Therefore, we analyze this P System's performance by counting the maximally parallel implementation steps.

When the best input condition – that is, list with ascending sort – is presented, we can see that membrane 2 only needs to run n steps of maximally parallel implementation because the cycle rules never fire in the running process; besides, membrane 1 also run n steps of maximally parallel implementations. Therefore, totally $2n$ steps of maximally parallel implementations have been fired, and thereby the time complexity is $\Omega(n)$ exactly.

When the worst input condition – that is, list with descending sort – is presented, the cycle rules should be used. In membrane 2, besides the n steps filter maximally parallel implementation, $i-1$ number of extra steps should be added in the ith filter step, that is, $1+2+3+\cdots+(n-1) = (n-1)(n-2)/2$ extra steps. Apparently, the time complexity is $O(n^2)$ exactly.

As we know, the lower-bounded time complexity of sort algorithms based on comparison is $O(n \log n)$. Thus, we are interested in whether the average running steps of this P System could be decreased lower than $n \log n$. In fact, this task is not very hard to accomplish. As long as we use 2^n number of match evolution rules to replace the n cycle rules, let any i-steps cycle be one-step corresponding match, then any sorting list can be sorted in $2n$ steps. However, this method will sacrifice lots of space and seems to be not practical.

3.2. Usage of this property

Although this property can serve to solve the sorting problem quickly as mentioned above, there seems to be another important aspect which we have not covered. In fact, what we care mainly is how to use this property or what kinds of problems could be satisfied with this property. We make an informal conclusion from several experiments that any problem related to comparison could be solved quickly by using this property. This conclusion is some what not too hard to be proved, since any data comparison can be easily transformed to the corresponding parallel data filtering.

4. Enumeration property

This property shows mainly that the maximally parallel attribute can be assimilated to the role of an enumerator [7]. As we know, many problems such as problems in NP-complete are generally considered to be not being capable of giving a deterministic algorithm. Generally, we make a total search to solve problems in NP-complete, which spend exponential time. These solutions usually implement search operations step by step.

We introduce this property as follows, aiming to show how the maximally parallel attribute serve to improve the complete-search efficiency. In addition, we will present a solution to the Hamiltonian cycle problem using this property.

Property 2 (Enumeration property)**.** *The maximally parallel implementation in the P System can be regarded as an enumerator, providing efficient enumeration operators in solving search problems.*

The enumeration property of the P System is like a search tree, where one floor represents a parallel search step. Thus, the search efficiency would be increased in exponential speed. Fig. 3 shows how to use this property to solve the Hamiltonian cycle problem which is in NP as follows.

First, we make a presumption that the Hamilton cycle begins from vertex 1. Then we model the computation with $n+1$ sub-membranes placed in the skin. The membranes 1 to n correspond to the n vertices in the given graph. They represent the enumerations of the corresponding vertices. The membrane $n+1$ is the output membrane of the system. For example, if $(1,2), (1,3), (1,4)$ belong to the set $E(G)$, then membrane 1 will have enumeration operations and generate objects D_2, D_3 and D_4. These three objects will be sent out of membrane 1 and reach the skin. At last, they will be separately sent to the corresponding membranes

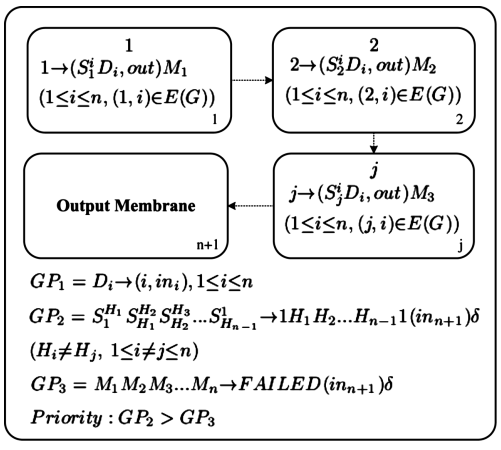

Fig. 3. P System that solves sorting problem.

($D_2 \to 2$, $D_3 \to 3$, $D_4 \to 4$). While D_2, D_3, D_4 arrive at the corresponding membranes, these membranes implement the same operations as membrane 1 did. Finally, n vertices membranes complete their enumerations. At this moment, there exist $M_1, M_2, \ldots, M_n$ in the skin. Note that the priority of the rule $M_1M_2M_3 \ldots M_n \to FAILED(in_{n+1})\delta$ is lower than the rule above it. In other words, if the above rule cannot match, the computation will send *FAILED* to membrane $n+1$. This means that there is no Hamilton cycle in the given graph. Thus we can simply find out that the rule $S_1^{H_1}S_{H_1}^{H_2}S_{H_2}^{H_3} \ldots S_{H_{n-1}}^{1} \to 1H_1H_2 \ldots H_{n-1}1(in_{n+1})$ is the key rule that used to be judge whether there is a Hamilton cycle in Graph G. Apparently, the multisets $S_1^{H_1}S_{H_1}^{H_2}S_{H_2}^{H_3} \ldots S_{H_{n-1}}^{1}$ mean that there is a path $1 \to H_1 \to H_2 \to \cdots \to H_n \to 1$ generated from the vertices membranes $1, 2, \ldots, n$, and then the path is sent to the output membrane as the form $1H_1H_2 \ldots H_{n-1}1$ to show that the computation is successful. The alphabet δ means that the skin membrane should dissolve into the environment no matter whether the computation is successful or not. We can see in the running process that the maximally parallel implementation is like an enumerator; at each step it enumerates the collected edge corresponding to one vertex, and this process is surely in parallel. Apparently, this system has a polynomial time complexity running time.

This property can also be regarded as a quick enumerator, which could decrease the general exponential to the polynomial time. Therefore, any hard problems that could only use the search method to solve are satisfied to this property. [Certainly, the solutions to different specifications would have some differences in detail.

5. Discussion to this attribute

As mentioned above, the maximally parallel attribute of the P system has a strong ability and efficiency in solving hard problems, and this model has a very efficient computation capacity. However, this attribute is too strong to be practical! We can imagine what result a computation model will lead to if it cannot be or hardly can be realized in the real world. Thus, we will discuss for the purpose of seeing if, there are some methods to improve this condition.

Discussion 1: The number of objects in one maximally parallel implementation is not limited.

This is an amazing property. Without being limited in the number of objects, the P System could generate any number of objects in just one step. Then we can easily transform an exponential generation in linear time by using only a simple rule. Think about the initial configuration with only one f in a membrane and the evolution rule with only $f \to ff$, the system just runs step by step as usual, we will find out that the number of f increases by $1, 2, 4, 8, \ldots$ namely, in exponential increasing speed, and we will also find out this generating process is in linear time, apparently.

Discussion 2: P System do not specify the running time of one maximally parallel implementation.

This is another amazing property in this attribute. The P System argued that each maximally parallel implementation has the same running time, which is also considered as an undividable running step. Therefore, we have the above presumption of time complexity in the P System. Generally speaking, the action in the living cell is indeed in a very highly parallel mode and even a very huge number of implementations are running at the same time, but on the other hand, the number is also indeed finite.

We believe this maximally parallel attribute can be weakened without any loss of computation universality, and further research is needed.

6. Conclusion

We have made some research on the analysis of the maximally parallel attribute of the P System. By using this attribute, we can solve some hard problems even in NP-complete very efficiently. However, although this attribute is so computational-efficient, it is not practical and nearly cannot be realized in our common life. Thus, to figure out whether this attribute would be weakened without loss of computational universality, further work is needed.

Acknowledgements

This work was supported by the National 973 Project (Grant No. 2003CB317005), National Natural Science Foundation of China (Grant Nos. 60473006 and 60573002), and The BoShiDian Research Fund (Grant No. 20010248033).

References

[1] Păun Gh. Computing with membrane – a variant. Int J Found Comput Sci 2000;11(1):167–82.
[2] Păun Gh. Membrane computing: basic ideas, results, applications. Workshop on theory and application of systems, Timisoara, Rumania, 2005.
[3] Alsuwaiyel MH. Algorithms-design techniques and analysis. London: World Scientific Publishing Company; 1998, pp. 14–31.
[4] Wegener I. Complexity theory. Berlin Heidelberg: Springer-Verlag; 2005, pp. 63–70.
[5] Păun Gh, Rozenberg G, Salomaa A. Membrane computing with external output. Fundam Inform 2000;41(3):259–66.
[6] Huang CY, Dong XJ, Long H. Using P Systems to solve sorting problem. J Shanghai Jiao Tong Univ 2008;42(2):206–8.
[7] Păun A. On P Systems with global rules. Lect Notes Comput Sci 2004:329–421.