Journal of Universal Computer Science, vol. 10, no. 5 (2004), 567-580
submitted: 1/4/04, accepted: 10/4/04, appeared: 28/5/04

Finding the Maximum Element Using P Systems

Federico Fontana, Giuditta Franco
(University of Verona, Italy
{fontana,franco}@sci.univr.it)

Abstract: A nondeterministic, maximally parallel methodology for finding the maximum element in a set of numerical values is presented, suitable for being implemented on P systems. Several algorithms of maximum search are then developed for different types of such systems, namely using priorities, nested membranes and linked transport, and their performances are evaluated accordingly. The proposed solutions are expected to find application inside membrane models devoted to compute algorithmic procedures in which the greatest element in a data set must be found. Dynamic algorithms for DNA sequence alignment are an example of such procedures.

Key Words: Maximum value, P systems, Natural computing, Membrane computing

Category: F.2.1, F.1.1

1 Introduction

The capability of P systems [Păun and Rozenberg 2002, Păun 2002] to solve in polynomial time problems that (provided that we do not have **P** = **NP**) need exponential execution time if coded over traditional computing resources, makes these systems especially eligible for dealing with **NP**-complete problems. Conversely, the application of P systems to problems that have already found efficient solutions on traditional computing architectures has, apparently, a minor relevance. A rearrangement of these solutions on membrane models in fact raises several implementation problems that, in most cases, are not experienced otherwise. From this point of view it does not make sense to rewrite well-known algorithms with an aim to implement them on a P system: the drawbacks overwhelm the possible advantages in this operation.

On the other hand, there is at least one strong point that justifies such an operation. It is likely (or at least strongly hoped) that, sooner or later, P systems will take their place in the set of computing resources *available* to the scientist, either running on traditional hardware in the form of simulations, or being realized inside whatever type of device. In any case they will be in charge of solving specific and often complex problems and tasks and, reasonably, these tasks in their turn will rely on the solution of simpler subtasks whenever possible.

In particular, it is expected that P systems will find a relevant field of application in the solution of bio-molecular problems [Fontana et al. 2004]. In this perspective we have started an investigation aimed at understanding if a relocation in the framework of P systems of the *sequence alignment problem*, well known in bioinformatics [Lipman and Pearson 1985, Altshul et al. 1990], can gain insight on the mechanisms ruling the alignment of the DNA, particularly those causing the mismatchings that determine changes in the DNA structure during a species' evolution. A first, unavoidable step in this investigation consists in reproducing some well-known algorithms of sequence alignment inside P systems.

Interestingly, the exponential complexity of these algorithms becomes (pseudo) polynomial if a dynamic approach is chosen instead of a naive one. In the case of sequence alignment, dynamic programming asks for finding the maximum element in a set of relative scores.

The literature on P systems touches this issue only partially, i.e., in a computer-graphical problem of rectangular pictures tiling [Ceterchi et al. 2003] and in the modeling of sorting algorithms [Alhazov and Sburlan 2003]. This paper, then, focuses on the problem of finding the maximum element using membrane systems—coming out with some interesting points of discussion—and leaves the modeling of the rest of the DNA sequence alignment procedure to a forthcoming research.

2 Nondeterministic, Maximally Parallel Search

It is known that the maximum between N numbers can be easily searched in linear time on a serial machine, by comparing number pairs one after the other [Cormen et al. 1990]. If m (parallel) computing resources come into play, then at every computation step we can compare m pairs simultaneously—though, in principle our processing unit might be able to evaluate not only pairs, but also triples or even K-uples. Whatever the number m of processing units and their parallel evaluation capability (i.e., the value of K), at the end of every computation step we will have several relative maxima at hand. At this point the maximum must be searched again on a reduced set of numbers, until reducing this set to one single value. We will call this kind of approach to the problem *horizontal*.

Alternatively, we should be able to count at least up to the maximum: during this counting up, we must evaluate N data (possibly all of them at the same time) just until all of them become smaller or equal than the value counted out. This value is the maximum, and must finally be sent to the output. We will call this second approach *vertical*.

We can cast the above considerations in the framework of a P system containing N distinct symbol-objects, each with its own number of occurrences. This P system selects the symbol having the largest number of occurrences, and it will send out this value (thus solving a problem of *maximum search*) possibly along with the symbol itself (thus solving, in addition, a problem of *maximum element search*).

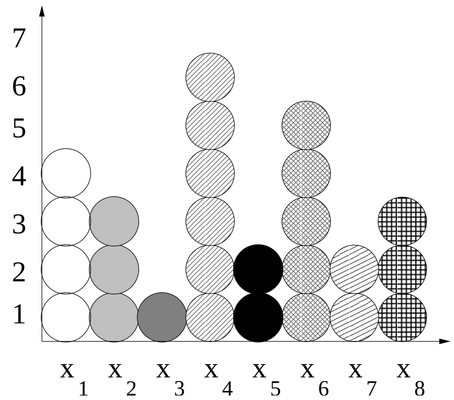

Figure 1: "Horizontal" and "vertical" approach to the maximum search.

The horizontal approach calls for a procedure that works on symbols, whereas the vertical approach calls for a procedure working on their occurrences (refer to figure 1, in which 8 symbols $x_1, \ldots, x_8$ are compared or, alternatively, a counter stops as soon as no further occurrences of any two objects exist—that is, when it counts up to 6—implementing the horizontal and the vertical approach, respectively). We will model both approaches and discuss the efficiency and complexity of each implementation in several types of membrane systems, namely using multiple priorities on the rules, multiple membranes, and linked transport. In the following, we will indicate the number of occurrences of x with $|x|$.

3 Search With Priorities on the Rules

In the most simple case we can find the maximum M between two symbols, x_1 and x_2, by considering the following construct (for ease of notation, here and in the following of the paper we will denote subsets of rules having the same priority as $r_i : R_1, R_2, \ldots, R_s;$, and as usual specify on the symbols r_i a partial order relation on the priorities among subsets):

$$\begin{aligned} &V = \{x_1, x_2, x'\}, \quad T = \{x'\}, \quad w = x_1^{k_1} x_2^{k_2}, \\ &R = \{r_1 : x_1 x_2 \to x'_{out};\ r_2 : x_1 \to x'_{out}, x_2 \to x'_{out}\}, \quad \rho = \{r_1 > r_2\} \end{aligned} \tag{1}$$

First, either all symbols x_1 or x_2 are consumed and an identical number of symbols x' are sent out by means of the rule $r_1 : x_1 x_2 \to x'_{out}$. Then, either the remaining occurrences of x_1 or x_2 are sent out in form of occurrences of x', in a way that at the end of the computation it is clear that $M = |x'| = \max\{k_1, k_2\}$. In the meantime, the information on the maximum element is lost.

The information on the maximum element is preserved by changing the rules having priority r_2 in $x_1 \to x'_{1\,out}$, $x_2 \to x'_{2\,out}$, after inclusion of x'_1, x'_2 in V and T. With this change the maximum is figured out as the number of primed symbols sent out to the environment, and the presence of x'_1 (x'_2) in the environment signals that the maximum element is x_1 (x_2).

In general, we can extend to N symbols the reasoning leading to (1), by repeatedly comparing pairs until one goes out of the skin membrane. For the sake of simplicity we can consider N to be a power of 2—it is anyway possible to add in the membrane system one occurrence of as many "ghost" symbols as needed to achieve this condition, the maximum being unaffected by this operation.

The corresponding construct is the following one:

$$V = \{x_1, x_2, \ldots, x_N,\ \ x'_1, x'_2, \ldots, x'_{N/2},\ \ x''_1, x''_2, \ldots, x''_{N/4},\ \ \ldots\ \ ,\ \ x^{(\log N)}\},$$
$$T = \{x^{(\log N)}\}, \quad w = x_1^{k_1} x_2^{k_2} \ldots x_N^{k_N},$$
$$R = \left\{ \begin{array}{llll} r_1 : x_1x_2 \to x'_1, & x_3x_4 \to x'_2, \ldots & & x_{N-1}x_N \to x'_{N/2}; \\ r_2 : \quad x_1 \to x'_1, & x_3 \to x'_2, \ldots & & x_{N-1} \to x'_{N/2}, \\ \qquad\ x_2 \to x'_1, & x_4 \to x'_2, \ldots & & x_N \to x'_{N/2}; \\ & & & \\ r'_1 : x'_1x'_2 \to x''_1, & x'_3x'_4 \to x''_2, \ldots & & x'_{N/2-1}x'_{N/2} \to x''_{N/4}; \\ r'_2 : \quad x'_1 \to x''_1, & x'_3 \to x''_2, \ldots & & x'_{N/2-1} \to x''_{N/4}, \\ \qquad\ x'_2 \to x''_1, & x'_4 \to x''_2, \ldots & & x'_{N/2} \to x''_{N/4}; \\ & & & \\ & \ldots & & \\ & & & \\ r_1^{(\log N-1)} : & x_1^{(\log N-1)} x_2^{(\log N-1)} \to x_{out}^{(\log N)}; & & \\ r_2^{(\log N-1)} : & x_1^{(\log N-1)} \to x_{out}^{(\log N)}, & & \\ & x_2^{(\log N-1)} \to x_{out}^{(\log N)} & & \end{array} \right\}, \tag{2}$$
$$\rho = \{r_1 > r_2 > r'_1 > r'_2 > \ldots > r_1^{(\log N-1)} > r_2^{(\log N-1)}\},$$

where log is the logarithm on base 2.

The system behaves as follows: rules having priority r_1 consume all the symbol pairs x_1x_2, x_3x_4 and so on, that are initially present in the membrane, producing symbols $x'_1, x'_2, \ldots, x'_{N/2}$. At the following system configuration such rules cannot be applied, and rules with priority r_2 turn all the remaining initial symbols (actually, those which cannot take part in a couple) into symbols of type x'. At the third transition step rules having priority r_1 and r_2 cannot be applied any longer (all the initial symbols have disappeared), so the rules with priority r'_1 are applied to the new pairs $x'_1x'_2$, $x'_3x'_4$ and so on. Rules having priority r'_2 are applied at the fourth step to the symbols of type x' that cannot be paired, and so on until the last symbol is sent out of the skin in a way that

$$|x^{(\log N)}| = \max\{k_1, k_2, \ldots, k_N\}.$$

A construct which keeps the information on the maximum element can be derived, too. Though, at the end of this section we will see that it results in an exponential proliferation of the rules.

The number of symbols in the multiset is halved after every two transitions, by means of the application of the rules with priority $r_1^{(i)}$ and $r_2^{(i)}$. Then, the proposed system computes the maximum in $2 \log N$ steps making use of

$$N + N/2 + N/4 + \ldots + 2 + 1 = 2N - 1$$

symbols,

$$(N/2 + N) + (N/4 + N/2) + \ldots + (1 + 2) = (N - 1) + (2N - 2) = 3N - 3$$

rules, and $2 \log N$ priorities. That is, the maximum is found in logarithmic discrete time using one membrane with priorities. This approach requires a linear number of symbols and rules.

We can save resources if we do not make use of symbols other than the initial ones. In this case we can maintain the correct system evolution by producing a different symbol from each pair. The new rules are contained in the following set, that relies on the same priority set ρ seen in (2):

$$R = \left\{ \begin{array}{lll} r_1 : & x_1 x_2 \to x_1,\ x_3 x_4 \to x_3, \ldots & x_{N-1} x_N \to x_{N-1}; \\ r_2 : & x_2 \to x_1, \quad x_4 \to x_3, \ldots & x_N \to x_{N-1}; \\ & & \\ r'_1 : & x_1 x_3 \to x_1,\ x_5 x_7 \to x_5, \ldots & x_{N-3} x_{N-1} \to x_{N-3}; \\ r'_2 : & x_3 \to x_1, \quad x_7 \to x_5, \ldots & x_{N-1} \to x_{N-3}; \\ & \ldots & \\ r_1^{(i)} : & x_1 x_{2^i+1} \to x_1, \ldots & x_{N-2^{i+1}+1} x_{N-2^i+1} \to x_{N-2^{i+1}+1}; \\ r_2^{(i)} : & x_{2^i+1} \to x_1, \ldots & x_{N-2^i+1} \to x_{N-2^{i+1}+1}; \\ & \ldots & \\ r_1^{(\log N-1)} : & x_1 x_{N/2+1} \to x_{1\ out}; & \\ r_2^{(\log N-1)} : & x_{N/2+1} \to x_{1\ out} & \end{array} \right\} \qquad (3)$$

Now we make use of N symbols and $2N - 1$ rules, again with $2 \log N$ priorities (and consequent transition steps).

We now wonder whether increasing the parallelism in the comparisons improves the system performance or not. The answer is that it does *not*, as long as we have to compare every combination of elements during the production of symbols. To have an idea of it, consider the case of three elements. In that case we can define the following construct:

$$
\begin{aligned}
&V = \{x_1, x_2, x_3, x'\}, \quad T = \{x'\}, \quad w = x_1^{k_1} x_2^{k_2} x_3^{k_3}, \\
&R = \left\{ \begin{array}{l} r_1 : x_1x_2x_3 \to x'_{out}; \\ r_2 : x_1x_2 \to x'_{out},\ x_2x_3 \to x'_{out},\ x_1x_3 \to x'_{out}; \\ r_3 : x_1 \to x'_{out},\ x_2 \to x'_{out},\ x_3 \to x'_{out} \end{array} \right\}, \\
&\rho = \{r_1 > r_2 > r_3\}
\end{aligned} \tag{4}
$$

From this it follows that if we have N symbols, with N being a power of 3, then $3 \log_3 N$ steps are needed to find the maximum. More general, we will need $K \log_K N$ steps to maximize between N symbols using K-element parallel comparisons (again providing N to be a power of K, possibly adding "ghost" symbols).

The function $K \log_K N$ has an absolute minimum for $K = e$. In practice, simple and efficient implementations of the algorithm can be in principle realized by setting $K = 2$ or $K = 3$. It is interesting to calculate the figures of complexity in the case when one N-element comparison is performed by the system at once. This means that we must consider all possible combinations of $N, N-1, \ldots, 2, 1$ symbols. In this case the maximum is searched in $N \log_N N = N$ steps, using $N + 1$ symbols (or N if we prefer to send out one of the initial symbols at our choice, hence avoiding the use of x'), provided the existence of a number of rules equal to

$$\sum_{i=1}^{N} \binom{N}{i} = 2^N - 1. \tag{5}$$

On the other hand, N-element comparisons allow for a straightforward implementation of a maximum element search. In fact it is sufficient to substitute the lowest priority rules with the following ones:

$$r_N : \qquad x_1 \to x_{1\,out},\ x_2 \to x_{2\,out},\ \ldots,\ x_N \to x_{N\,out}$$

in a way that the symbol x_i outside the skin membrane can be associated to the corresponding element, and the maximum is equal to $|x'| + |x_i|$. If we want the system to work in the general case, in which many maxima can be present simultaneously, then we must implement a more elaborate strategy; for example, we can send out along with x', at every transition step, additional symbols related to the tuples being processed during that transition.

4 Search with Nested Membranes

Similarly to what we have done using priorities on the rules, we can define a construct analogous to (2) using the productions expressed by (3), this time using $\log_K N$ nested membranes when a K-element parallel comparison is implemented. In the case $K = 2$ we have the following construct (in an attempt to simplify the notation we have labeled the membranes using a reversed order, instead of the usual one [Păun and Rozenberg 2002]):

$$
\begin{aligned}
V &= \{x_1, x_2, \ldots, x_N\}, \quad T = \{x_1\}, \\
\mu &= [_{\log N}[_{\log N-1} \cdots [_2[_1 \quad]_1]_2 \cdots]_{\log N-1}]_{\log N}, \\
w_1 &= x_1^{k_1} x_2^{k_2} \ldots x_N^{k_N}, \quad w_i = \lambda \quad , \quad i = 2, \ldots, \log N, \\
R_i &= \left\{ \begin{array}{lll} r_1 : x_1 x_{2^{i-1}+1} \to x_1, & \cdots & x_{N-2^i+1} x_{N-2^{i-1}+1} \to x_{N-2^i+1}; \\ r_2 : x_1 \to x_1 \delta, & \cdots & x_{N-2^i+1} \to x_{N-2^i+1}\delta, \\ \quad\; x_{2^{i-1}+1} \to x_1 \delta, & \cdots & x_{N-2^{i-1}+1} \to x_{N-2^i+1}\delta \end{array} \right\}, \quad (6) \\
& \quad\; i = 1, 2, \ldots, \log N - 1, \\
R_{\log N} &= \left\{ \begin{array}{l} r_1 : x_1 x_{N/2+1} \to x_{1\,out}; \\ r_2 : x_1 \to x_{1\,out},\, x_{N/2+1} \to x_{1\,out} \end{array} \right\}, \\
\rho_i &= \{r_1 > r_2\}, \quad i = 1, 2, \ldots, \log N
\end{aligned}
$$

The evolution mechanism using this construct is the following:

– Pairs $x_1x_2, x_3x_4, \ldots$ are consumed by the rules with priority r_1 in membrane 1. After the application, at the next step, of the rules having priority r_2, membrane 1 dissolves setting the symbols $x_1, x_3, \ldots$ free to float in membrane 2. Each of those symbols is present in a number of occurrences equal to the relative maximum related to the pair the symbol itself comes from.

– The same operations happen in membrane 2, this time applied to the pairs $x_1x_3, x_3x_5, \ldots$ Half rules are needed to perform the same kind of transition as the previous one, to select new relative maxima from the existing pairs.

– Identical transitions happen in membranes 3, 4, and so on. Finally, in the skin membrane (labeled $\log N$) the absolute maximum is computed, and sent out as the number of occurrences of x_1.

Compared to the construct (2), the implementation using multiple membranes again needs $3N - 3$ rules with N symbols, to compute the result in $2 \log N$ steps. Implementing a single N-element parallel comparison reduces the number of nested membranes to one, hence leading to a P system that is identical to the one seen in the case of N-element parallel comparison using multiple priorities on the rules.

5 Search Using Linked Transport

Symport/antiport rules can be used to find the maximum, provided that a sufficient amount of substances are present in the environment to ensure the necessary exchange of symbols via antiport rules. Apart from this technical aspect, the algorithmic mechanism of symbol selection is the same as those seen in the previous sections.

In the easiest case the idea is to select symbols from pairs, by exchanging molecules with the environment. Once the symbols that are present in fewer copies have been sent out, the same selection is performed over new pairs until one symbol is left in the membrane.

$$
\begin{aligned}
V = \{ & x_1, \ldots, x_N, \;\; x'_1, \ldots, x'_{N/2}, \;\; x''_1, \ldots, x''_{N/4}, \;\; \ldots \;\; , \;\; x^{(\log N)}, \\
& y_1, \ldots, y_N, \;\; y'_1, \ldots, y'_{N/2}, \;\; y''_1, \ldots, y''_{N/4}, \;\; \ldots \;\; , \;\; y_1^{(\log N-1)}, y_2^{(\log N-1)}, \\
& a'_1, \ldots, a'_{N/2}, a''_1, \ldots, a''_{N/4}, \ldots, a_1^{(\log N)}, \\
& b_1, \ldots, b_{N/2}, b'_1, \ldots, b'_{N/4}, \ldots, b_1^{(\log N-1)}, \\
& c_1, \ldots, c_{N/2}, c'_1, \ldots, c'_{N/4}, \ldots, c_1^{(\log N-1)}, \\
& d_1, \ldots, d_{N/2}, d'_1, \ldots, d'_{N/4}, \ldots, d_1^{(\log N-1)}\}, \\
T = \{ & x^{(\log N)}\}, \quad w = x_1^{k_1} x_2^{k_2} \ldots x_N^{k_N}, \\
R = & \text{(see Figure 2)}
\end{aligned}
\tag{7}
$$

Despite the number of rules, the system evolution is quite simple:

- The initial symbols x_i are exchanged with identical numbers of occurrences of symbols y_i, respectively. Meanwhile, the "synchronization" symbols b_1, $\ldots$, $b_{N/2}$ are imported from the environment (note that $|b_i| = |x_i| + |x_{i+1}|$ must be larger than $|x_i|$ and $|x_{i+1}|$).
- The pairs $y_1 y_2, \ldots, y_{N-1} y_N$ are exchanged with the symbols $x'_1, \ldots, x'_{N/2}$. In parallel, symbols b_i are exchanged with new control symbols c_i and d_i, respectively.
- For each $i = 1, \ldots, N/2$, the symbols y_i, y_{i+1} which were not paired are carried out by c_i and exchanged with additional occurrences of x'_i. Now, x'_i contains the information on the relative maximum computed from the pair $y_i y_{i+1}$. In parallel, the synchronization symbol d_i is exchanged with a'_i, which enables the processing of a new set of pairs (i.e., $x'_1 x'_2, \ldots, x'_{N/2-1} x'_{N/2}$).

The horizontal lines drawn between the braces of R in (7) help in identifying the rules that are activated at each new symbol selection taking place in the system. In the final step the symbols $x^{(\log N)}$ are sent out of the skin membrane, and the maximum can thus be read as $|x^{(\log N)}|$.

The number of steps needed to figure out the result is $3 \log N$ plus the last (symport) transport. The symport/antiport rules involved in the process are $7(N/2 + N/4 + \ldots + 1) + 1 = 7N - 6$. The system needs $(2N-1) + (2N-2) =$

$$
R = \left\{
\begin{array}{llll}
(x_1, \mathrm{out}; b_1 y_1, \mathrm{in}) & (x_3, \mathrm{out}; b_2 y_3, \mathrm{in}) & \ldots & (x_{N-1}, \mathrm{out}; b_{N/2} y_{N-1}, \mathrm{in}), \\
(x_2, \mathrm{out}; b_1 y_2, \mathrm{in}) & (x_4, \mathrm{out}; b_2 y_4, \mathrm{in}) & \ldots & (x_N, \mathrm{out}; b_{N/2} y_N, \mathrm{in}), \\
\\
(y_1 y_2, \mathrm{out}; x'_1, \mathrm{in}) & (y_3 y_4, \mathrm{out}; x'_2, \mathrm{in}) & \ldots & (y_{N-1} y_N, \mathrm{out}; x'_{N/2}, \mathrm{in}), \\
(b_1, \mathrm{out}; c_1 d_1, \mathrm{in}) & (b_2, \mathrm{out}; c_2 d_2, \mathrm{in}) & \ldots & (b_{N/2}, \mathrm{out}; c_{N/2} d_{N/2}, \mathrm{in}), \\
\\
(c_1 y_1, \mathrm{out}; x'_1, \mathrm{in}) & (c_2 y_3, \mathrm{out}; x'_2, \mathrm{in}) & \ldots & (c_{N/2} y_{N-1}, \mathrm{out}; x'_{N/2}, \mathrm{in}), \\
(c_1 y_2, \mathrm{out}; x'_1, \mathrm{in}) & (c_2 y_4, \mathrm{out}; x'_2, \mathrm{in}) & \ldots & (c_{N/2} y_N, \mathrm{out}; x'_{N/2}, \mathrm{in}), \\
(d_1, \mathrm{out}; a'_1, \mathrm{in}) & (d_2, \mathrm{out}; a'_2, \mathrm{in}) & \ldots & (d_{N/2}, \mathrm{out}; a'_{N/2}, \mathrm{in}), \\
\hline
(a'_1 x'_1, \mathrm{out}; b'_1 y'_1, \mathrm{in}) & (a'_3 x'_3, \mathrm{out}; b'_2 y'_3, \mathrm{in}) & \ldots & (a'_{N/2-1} x'_{N/2-1}, \mathrm{out}; b'_{N/4} y'_{N/2-1}, \mathrm{in}), \\
(a'_2 x'_2, \mathrm{out}; b'_1 y'_2, \mathrm{in}) & (a'_4 x'_4, \mathrm{out}; b'_2 y'_4, \mathrm{in}) & \ldots & (a'_{N/2} x'_{N/2}, \mathrm{out}; b'_{N/4} y'_{N/2}, \mathrm{in}), \\
\\
(y'_1 y'_2, \mathrm{out}; x''_1, \mathrm{in}) & (y'_3 y'_4, \mathrm{out}; x''_2, \mathrm{in}) & \ldots & (y'_{N/2-1} y'_{N/2}, \mathrm{out}; x''_{N/4}, \mathrm{in}), \\
(b'_1, \mathrm{out}; c'_1 d'_1, \mathrm{in}) & (b'_2, \mathrm{out}; c'_2 d'_2, \mathrm{in}) & \ldots & (b'_{N/4}, \mathrm{out}; c'_{N/4} d'_{N/4}, \mathrm{in}), \\
\\
(c'_1 y'_1, \mathrm{out}; x''_1, \mathrm{in}) & (c'_2 y'_3, \mathrm{out}; x''_2, \mathrm{in}) & \ldots & (c'_{N/4} y'_{N/2-1}, \mathrm{out}; x''_{N/4}, \mathrm{in}), \\
(c'_1 y'_2, \mathrm{out}; x''_1, \mathrm{in}) & (c'_2 y_4, \mathrm{out}; x'_2, \mathrm{in}) & \ldots & (c'_{N/4} y'_{N/2}, \mathrm{out}; x''_{N/4}, \mathrm{in}), \\
(d'_1, \mathrm{out}; a''_1, \mathrm{in}) & (d'_2, \mathrm{out}; a''_2, \mathrm{in}) & \ldots & (d'_{N/4}, \mathrm{out}; a''_{N/4}, \mathrm{in}), \\
\hline
\multicolumn{4}{c}{\ldots} \\
\hline
\multicolumn{4}{l}{(a_1^{(\log N-1)} x_1^{(\log N-1)}, \mathrm{out}; b_1^{(\log N-1)} y_1^{(\log N-1)}, \mathrm{in})} \\
\multicolumn{4}{l}{(a_2^{(\log N-1)} x_2^{(\log N-1)}, \mathrm{out}; b_1^{(\log N-1)} y_2^{(\log N-1)}, \mathrm{in})} \\
\\
\multicolumn{4}{l}{(y_1^{(\log N-1)} y_2^{(\log N-1)}, \mathrm{out}; x^{(\log N)}, \mathrm{in})} \\
\multicolumn{4}{l}{(b_1^{(\log N-1)}, \mathrm{out}; c_1^{(\log N-1)} d_1^{(\log N-1)}, \mathrm{in})} \\
\\
\multicolumn{4}{l}{(c_1^{(\log N-1)} y_1^{(\log N-1)}, \mathrm{out}; x^{(\log N)}, \mathrm{in})} \\
\multicolumn{4}{l}{(c_1^{(\log N-1)} y_2^{(\log N-1)}, \mathrm{out}; x^{(\log N)}, \mathrm{in})} \\
\multicolumn{4}{l}{(d_1^{(\log N-1)}, \mathrm{out}; a_1^{(\log N)}, \mathrm{in})} \\
\\
\multicolumn{4}{l}{(a_1^{(\log N)} x^{(\log N)}, \mathrm{out})}
\end{array}
\right\}
$$

Figure 2: Formulation of R in (7)

$4N-3$ "informative" symbols and $4(N/2+N/4+\ldots)=4N-4$ carrier and "synchronization" symbols.

Note that the process ends leaving a certain amount of "garbage", in the form of symbols of type a and c. In fact, the number of copies of such symbols left inside the membrane does not equal the number of copies of the initial symbols. Strategies to get rid of this garbage can be implemented anyway [Păun 2004].

Once again, higher-degree parallel comparisons can be chosen [Păun 2004]. In particular, we give the construct performing N-element parallel comparison (again, horizontal lines separate different symbol selections. Furthermore, we shorten each rule of the type $(\alpha, \text{out}; \beta, \text{in})$ in $(\alpha;\beta)$ to save horizontal space).

$$
\begin{aligned}
&V=\{x,\ x_1,\ldots,x_N,\ y_1,\ldots,y_N,\ a_1,\ldots,a_{N+1},\ b_0,\ldots,b_N,\ A_1,\ldots,A_N\},\\
&T=\{x\},\quad w=x_1^{k_1}x_2^{k_2}\ldots x_N^{k_N},\\
&R=\left\{\begin{array}{c}
(x_1;b_0A_1y_1)\ (x_2;b_0A_2y_2)\ \ldots\ (x_N;b_0A_Ny_N),\\
\hline
(y_1y_2\ldots y_N;x),\\
(b_0;a_1b_1),\\
\hline
(a_1y_2\ldots y_N;x)\ (a_1y_1y_3\ldots y_N;x)\ \ldots\ (a_1y_1\ldots y_{N-1};x),\\
(b_1;a_2b_2),\\
\hline
\cdots\\
\left(\binom{N}{i}\text{combinations of } y_i \text{ carried out by } a_i;x,\text{in}\right)\\
(b_i;a_{i+1}b_{i+1}),\\
\cdots\\
\hline
(a_NA_1y_1;x)\ (a_NA_2y_2;x)\ \ldots\ (a_NA_Ny_N;x),\\
(b_N;a_{N+1}),\\
\hline
(a_{N+1}x,\text{out})
\end{array}\right\}
\end{aligned}
\tag{8}
$$

The system first exchanges x_i with an equal number of occurrences of y_i, furthermore it acquires the synchronizing symbol b_0. Next it exchanges as many N-uples of symbols y_i as possible with x, meanwhile substituting b_0 with b_1 along with acquiring the carrier symbol a_1. All the remaining steps consist in exchanging (via the carrier) combinations of symbols y_i with x, meanwhile updating the synchronization and carrier symbols through rules of the type $(b_i, \text{out}; a_{i+1}b_{i+1}, \text{in})$. In the last transition all the occurrences of x are sent out via a symport rule in a way that the maximum, $|x|$, can be read in the environment. As before, garbage is left in the membrane.

Again, the N-element comparison allows to search the maximum value in a straightforward way. In the above implementation we have in fact added the symbols $A_1,\ldots,A_N$ that enter the membrane in the beginning of the computation, and turn out to be useful in the end: since at most one of the rules of type $(a_NA_iy_i, \text{out}; x, \text{in})$ will be activated, then A_i will signal that x_i corresponds to the maximum element. As briefly discussed in Section 3, the search for the maximum element can be extended to the multiple case, by sending out additional symbols in correspondence to every rule of type $(\text{something}, \text{out}; x, \text{in})$.

The system using linked transport and implementing N-element comparison computes the result in N steps, plus two additional steps needed to exchange molecules respectively in the beginning and the end of the computation. $2N+1$ informative symbols and $2N+2$ auxiliary symbols suffice, though—refer to (5)—$(2^N-1)+N+N+1 = 2^N+2N = O(2^N)$ rules are present in the system. Additional N symbols are needed to compute also the maximum element (if unique).

6 "Vertical" Search

Introduced in Section 2, the vertical approach consists in a parallel counting up of all symbols with the final exclusion of those that sum up a number of occurrences below the (absolute) maximum. Only the object occurrences forming the maximum are sent out of the skin membrane. In this way, we also come up with a straightforward method computing the maximum element.

We consider the following P system:

$$
\begin{aligned}
&V = \{a_1,\ldots,a_N,a'_1,\ldots,a'_N,b,d_1,\ldots,d_N\}, \quad T=\{a_1,\ldots,a_N\},\\
&\mu = [_s\,[_1\,]_1\ \ [_2\,]_2\ \cdots\ [_N\,]_N\ \ ]_s, \quad w_i = a'_i a_i^{k_i}\,,\ i=1,\ldots,N,\\
&R_i = \{r_1 : \ a'_i a_i \to a'_i a_{i\,out};\ r_2 : \ a'_i \to a'_{i\,out}\}\,,\\
&\rho_i = \{r_1 > r_2\}\,,\ i=1,\ldots,N,\\
&R_s = \left\{\begin{array}{rl}
a'_2a'_3\ldots a'_{N-1}a'_N & \to d_1,\\
a'_1a'_3\ldots a'_{N-1}a'_N & \to d_2,\\
& \vdots\\
a'_1a'_2\ldots a'_{N-2}a'_N & \to d_{N-1},\\
a'_1a'_2\ldots a'_{N-2}a'_{N-1} & \to d_N,\\
d_1a_1 & \to d_1a_{1\,out},\\
d_2a_2 & \to d_2a_{2\,out},\\
& \vdots\\
d_Na_N & \to d_Na_{N\,out},\\
a'_1a'_2\ldots a'_{N-1}a'_N & \to b_{out}
\end{array}\right\}
\end{aligned}
\tag{9}
$$

We start with N membranes labeled $1,2,\ldots,N$, internal to the skin region, containing $k_1,k_2,\ldots,k_N$ occurrences of the symbol $a_1,a_2,\ldots,a_N$ plus one copy of $a'_1,a'_2,\ldots,a'_N$, respectively.

The system evolution is described by the following phases:

- In the first phase every rule having priority r_1 is applied inside the corresponding membrane. This causes the i-th membrane to expel one occurrence of the symbol a_i at any transition, for each $i = 1, 2, \ldots, N$. As long as the i-th membrane has no more symbols a_i left inside it, then the corresponding rule having priority r_2 is applied in a way that a'_i is sent to the skin membrane. This phase lasts until $N-1$ primed symbols are sent to the skin region, that is, until $N-1$ membranes have been emptied;

– At this point one of the first N rules of R_s in (9)—the one including all the $N-1$ primed symbols floating inside the skin membrane—is applied. In consequence of that, that combination of primed symbols is transformed in a corresponding symbol of type d. Finally, this symbol cooperates to send out of the skin all occurrences of the symbols of type a having its same index.

– The process ends when there are no rules left to apply; in this system the final configuration is $\mu = [_s a_1^{k_1} \ldots a_{i-1}^{k_{i-1}} a_{i+1}^{k_{i+1}} \ldots a_N^{k_N} a'_i d_i [_1\ ]_1 [_2\ ]_2 \cdots [_N\ ]_N]_s$. If M is the maximum and a_i is the maximum element, then all the M occurrences of a_i have been sent to the environment.

With this approach the number of computational steps depends on the maximum M and on the number of occurrences of the element that has the second greatest number of occurrences, say, $\tilde{M}$. In fact, the first phase is carried out in $\tilde{M}$ steps. Then in one step the production of d is completed and, finally, in further M steps the M occurrences of the maximum element are sent out. Hence, the total number of transitions is $\tilde{M} + 1 + M = O(M)$.

This number can be reduced by increasing the production of occurrences of the carrier d. This can be done by changing the first N rules in the skin region according to

$$R_s = \left\{ \begin{array}{r} a'_2 \ldots a'_{N-2} a'_{N-1} a'_N \to d_1^h, \\ a'_1 a'_3 \ldots a'_{N-1} a'_N \to d_2^h, \\ \vdots \\ a'_1 a'_2 \ldots a'_{N-2} a'_N \to d_{N-1}^h, \\ a'_1 a'_2 \ldots a'_{N-2} a'_{N-1} \to d_N^h, \\ \vdots \end{array} \right\} \tag{10}$$

where h is a number large enough. For instance, if h is greater than or equal to $\tilde{M} + 1$, then the system sends out $\tilde{M} + 1$ copies of the maximum symbol in one step, and the total number of computation steps becomes exactly M.

The vertical approach is especially effective when we have a lot of elements with low multiplicity. Despite this, the proposed system cannot compute the maximum if more relative maxima exist, that is, if two or more symbols having the same (maximum) number of occurrences are present in the system. To account for this, one special rule exists in the skin that transforms all primed symbols into b and sends this symbol out. The role of b is signaling to the

system configuration	computation steps	number of symbols	number of membranes	number of rules	number of priorities
p/2-spc	$O(\log_2 N)$	$O(N)$	$O(1)$	$O(N)$	$O(\log_2 N)$
p/N-spc	$O(N)$	$O(N)$	$O(1)$	$O(2^N)$	$O(N)$
nm/2-spc	$O(\log_2 N)$	$O(N)$	$O(\log_2 N)$	$O(N)$	$O(1)$
nm/N-spc	$O(N)$	$O(N)$	$O(1)$	$O(2^N)$	$O(N)$
lt/2-spc	$O(\log_2 N)$	$O(N)$	$O(1)$	$O(N)$	—
lt/N-spc	$O(N)$	$O(N)$	$O(1)$	$O(2^N)$	—
p/v	$O(M)$	$O(N)$	$O(N)$	$O(N)$	$O(1)$

Table 1: Figures of performance for maximum search under different membrane systems (N: number of elements; M: maximum value. p: priorities; nm: nested membranes; lt: linked transport; 2-spc: 2-elements parallel comparison; N-spc: N-elements parallel comparison; v: vertical approach).

environment that two or more primed symbols have been sent to the skin simultaneously, resulting in the total number of primed symbols present inside the skin membrane to have become equal to N without ever been equal to $N-1$: this happens only if more than one membrane contains a maximum number of symbols.

On the other hand, if an absolute maximum exists, then the proposed approach finds the maximum element without the exponential proliferation of rules seen in the previous sections.

7 Summary and Ongoing Research

In an aim to understand the meaning of DNA sequence alignment made using P systems we had to investigate a specific point of the dynamic alignment algorithms, that is, the search for the maximum element in a set. The investigation we have conducted is nevertheless valid *per se*, given the wide range of contexts where the maximum search problem must be dealt with. In particular, we have evaluated the performance of several algorithms based on two orthogonal approaches to the problem, running on different P systems. These performances are summarized in table 1 in terms of computation time and resources needed to figure out the solution.

The investigation on DNA alignment might reveal the existence of links between membrane systems and DNA computing paradigms, due to the analogies existing between the dynamic (i.e., step-by-step) algorithms of sequence alignment and the mechanisms of DNA replication. The work to do now deals with a strategy to model the growing of a DNA sequence using P systems, and it is actually in progress.

Acknowledgements

The authors want to acknowledge Agustín Riscos-Núñez, Rodica Ceterchi and Gheorghe Păun for the fruitful discussions taken at the Second Brainstorming Week on Membrane Computing held on February 2-7 in Seville, Spain. Vincenzo Manca has given many suggestions for the improving of the paper, moreover he has been a constant source of inspiration during this research.

References

[Alhazov and Sburlan 2003] Alhazov A., Sburlan D.: "Static sorting algorithms for P systems"; "Pre-proceedings of Workshop on Membrane Computing, WMC2003" (A. Alhazov, C. Martín-Vide, Gh. Păun, eds.), Tarragona, Spain (July 2003), 17–40

[Altshul et al. 1990] Altschul S., Gish W., Miller W., Myers E., Lipman D.: "Basic local alignment search tool"; J. Molecular Biology, 215 (1990), 403–410

[Ceterchi et al. 2003] Ceterchi R., Gramatovici R., Jonoska N.: "P systems for tiling rectangular pictures"; "Pre-proceedings of Workshop on Membrane Computing, WMC2003" (A. Alhazov, C. Martín-Vide, Gh. Păun, eds.), Tarragona, Spain (July 2003), 133–144

[Cormen et al. 1990] Cormen T.H., Leiserson C.E., Rivest R.L.: "Introduction to Algorithms"; McGraw-Hill, New York, NY (1990)

[Fontana et al. 2004] Fontana F., Franco G., Bianco L., Manca V.: "P Systems in Bio Systems"; "Applications of Membrane Computing" (G. Ciobanu, Gh. Păun, M.J. Peréz-Jiménez, eds.), Springer, Berlin. In press.

[Lipman and Pearson 1985] Lipman D.J., Pearson W.R.: "Rapid and sensitive protein similarity searches"; Science, 227 (1985), 1435–1441

[Păun 2002] Păun Gh.: "Membrane Computing. An Introduction"; Springer, Berlin (2002)

[Păun 2004] Păun Gh.: Personal communication (Feb 2004)

[Păun and Rozenberg 2002] Păun Gh., Rozenberg G.: "A guide to membrane computing"; Theoretical Computer Science, 287 (2002), 73–100