Maximum Search Using P Systems

Federico FONTANA, Giuditta FRANCO

University of Verona
Department of Computer Science
strada Le Grazie, 14
37134 Verona, Italy
E-mail: {fontana,franco}@sci.univr.it

Abstract. Several algorithms of maximum search are investigated and evaluated in different types of P systems, namely using priorities, multiple nested membranes and linked transport. The proposed solutions are expected to find application in a wide range of problems. In particular, the authors are currently working on modeling an algorithm for DNA sequence alignment using P systems.

1 Introduction

The capability of P systems to execute in polynomial time programs that, otherwise, need exponential time when executed using traditional computing resources, makes these systems especially eligible for dealing with NP-complete problems [10, 9].

This is the case of the algorithms for aligning sequences of symbols. Such algorithms must deal with the *pair* and *multiple sequence alignment problem* [4]. At an abstract level they can be seen as procedures for finding the optimal alignment between a pair or a multiple set of strings [7]. Though, at an application level, they have found major diffusion among the most popular biological database search programs [8, 2].

In principle it is easy to figure out the best alignment: it is sufficient to construct and then score all possible gapped alignments between two (or more) DNA sequences. This corresponds to generate the entire tree of the alignments, then to label every leaf with the related score: the higher the score, the better the alignment [5].

The procedure of sequence alignment becomes much more parsimonious, if the optimality is *dynamically* checked out step by step. Instead of generating all possible gapped alignments between two or more sequences and, finally, determining the best one, the dynamic approach asks to evaluate at every time step some (partial) highest scoring alignments, possibly leading to the (final) optimal score: this approach, corresponding to pruning the tree of the alignments, dramatically reduces the number of computations needed to achieve the final, optimal gapped alignment.

P systems can be in principle accommodated for generating the whole tree of the alignments in polynomial time. Though, exploiting the parsimony coming from the dynamic programming strategy results in a more efficient use of the resources provided by the membranes. The key point of this strategy consists in computing the maximum score between several partial alignments competing to form the final optimal one: only the

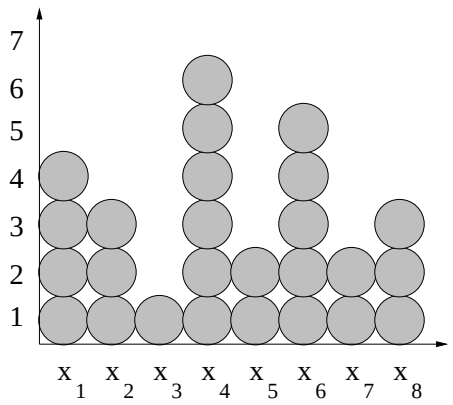

Figure 1: "Horizontal" and "vertical" approach to the maximum search.

highest-scoring of them will be eligible to be included in the final alignment. A similar problem has been dealt with in the context of rectangular pictures tiling using P systems [3].

From the point of view of membrane computing, we can model this stage of the procedure as a search for the greatest multiplicity of N symbols. Literature concerned with a similar issue includes the modeling of sorting algorithms [1]. This paper, then, focuses on the problem of maximum search using membrane systems, and leaves the modeling of the rest of the DNA sequence alignment procedure to a forthcoming research.

2 Non Deterministic, Maximally Parallel Maximum Search

It is known that the maximum search between N numbers can be easily carried out in linear time on a serial machine, by performing successive comparisons of value pairs [4]. If multiple (parallel) computing resources come into play, then at every computation step we can compare more than one pair simultaneously–though, in principle our processing unit might be able to test triples or n-uples. Whatever the number of processing units and their capability to perform multiple tests simultaneously, 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 a single value. We 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 should evaluate the input values (possibly all of them at the same time) just until they become smaller or equal than the value counted out. This value (i.e., the maximum) should finally be sent to the output. We call this second kind of approach to the problem *vertical*.

We can cast the above considerations in the framework of a P system containing N symbols, that selects the symbol having the largest number of occurrences and sends out this number (thus solving a problem of *maximum search*) possibly along with the symbol itself (thus solving, in addition, a problem of *maximum value search*).

The horizontal approach calls for a procedure that works on the symbols, whereas the vertical approach calls for a procedure working on the symbol occurrences (refer to figure 1). We will model both the horizontal and vertical maximum search, and discuss the efficiency and complexity of the procedure under several types of membrane systems,

namely using multiple priorities on rules, multiple membranes, and linked transport. In the following, we will indicate with $|x|$ the number of copies of x.

3 Maximum Search with Priority

In the most simple case we can determine the maximum between two symbols x_1 and x_2, each one present in the skin membrane with its own number of copies, by considering the following construct [11]:

$$V = \{x_1, x_2, x'\}, \quad T = \{x'\}, \quad w = x_1 x_2,$$
$$R = \{r_1 : x_1 x_2 \to x'_{out},\ r_2 : x_1 \to x'_{out}, x_2 \to x'_{out}\}, \quad \rho = \{r_1 > r_2\}$$

It is straightforward to see that the number of copies of symbols x' coming out from this simple membrane system is the maximum between $|x_1|$ and $|x_2|$, but the information on the maximum value is lost.

To preserve the information on the maximum value it is sufficient to change the rules having priority r_2 in $x_1 \to x_1'_{out}, x_2 \to x_2'_{out}$, after substitution of x' with x'_1, x'_2 in V and T. After this change the maximum is figured out as $|x'_1| + |x'_2|$ and the presence of x'_2 in the environment signals that it is $|x'_2| > |x'_1|$.

In general, we can extend the same reasoning to N symbols by repeatedly comparing pairs of symbols until the final 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 "ghost" symbols in one copy in the membrane to achieve this condition, the maximum being unaffected by this operation.

The corresponding construct is the following one:

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

where log is the logarithm with basis 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 time step 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'_i. At the third time 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 time step to the symbols of type x'_i that cannot be paired, and so on until the last symbol is sent out of the skin in a way that

$$|x^{(\log N)}| = \max\{x_1, x_2, \ldots, x_N\}$$

The number of symbols in the multiset is halved every two time steps, after the application of the rules with priority $r_1^{(i)}$ and $r_2^{(i)}$. Then, the proposed system computes the maximum in $2\log N$ time 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 search is computed in logarithmic time using one membrane with priorities. This system then 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 evolution in the system by producing a different symbol from each pair. The new rules are contained in the following set:

$$R = \left\{ \begin{array}{lrlll} r_1: & x_1x_2 \to x_1, & x_3x_4 \to x_3, & \ldots & x_{N-1}x_N \to x_{N-1}, \\ r_2: & x_2 \to x_1, & x_4 \to x_3, & \ldots & x_N \to x_{N-1}, \\ & & & & \\ r'_1: & x_1x_3 \to x_1, & x_5x_7 \to x_5, & \ldots & x_{N-3}x_{N-1} \to x_{N-3}, \\ r'_2: & x_3 \to x_1, & x_7 \to x_5, & \ldots & x_{N-1} \to x_{N-3}, \\ & & & & \\ & & \ldots & & \\ & & & & \\ r_1^{(i)}: & x_1x_{2^i+1} \to x_1, & \cdots & x_{N-2^{i+1}+1}x_{N-2^i+1} \to x_{N-2^{i+1}+1}, \\ r_2^{(i)}: & x_{2^i+1} \to x_1, & \cdots & x_{N-2^i+1} \to x_{N-2^{i+1}+1}, \\ & & & \\ & & \ldots & \\ & & & \\ r_1^{(\log N-1)}: & & x_1x_{N/2+1} \to x_{1\,out}, & \\ r_2^{(\log N-1)}: & & x_{N/2+1} \to x_{1\,out} & \end{array} \right\} \quad (2)$$

using the same priority set ρ seen in (1). Now we make use of N symbols and $2N - 1$ rules, again with $2\log N$ priorities (and consequent computation 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 possible combination of symbols during the production of new symbols. To have an

idea of it, consider the case of three symbols to compare. In that case we can define the following construct:

$$
\begin{aligned}
&V = \{x_1, x_2, x_3, x'\}, \quad T = \{x'\}, \quad w = x_1 x_2 x_3, \\
&R = \left\{ \begin{array}{ll} r_1: & x_1 x_2 x_3 \to x'_{out}, \\ r_2: & x_1 x_2 \to x'_{out},\ x_2 x_3 \to x'_{out},\ x_1 x_3 \to x'_{out}, \\ r_3: & x_1 \to x'_{out},\ x_2 \to x'_{out},\ x_3 \to x'_{out} \end{array} \right\}, \\
&\rho = \{r_1 > r_2 > r_3\}
\end{aligned}
\tag{3}
$$

From this it descends 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 in general, we will need $K \log_K N$ time steps to maximize between N symbols using K-symbol parallel comparisons (again providing N to be a power of K, eventually adding "ghost" symbols).

The function $K \log_K N$ has an absolute minimum in correspondence of $K = e$. 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-symbol comparison is performed by the system at once. This means that we have to 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

$$
\sum_{i=1}^{N} \binom{N}{i} = 2^N - 1 \tag{4}
$$

rules.

On the other hand, N-symbol comparisons allow for a straightforward implementation of a maximum value search algorithm. In this case 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 presence of the symbol x_i outside the skin membrane signals that it has maximal numeracy, and the maximum is equal to $|x'| + |x_i|$. If we want to account for the more general case of multiple maximum values then a more elaborate strategy must be implemented, for example by sending out at every computation, along with x', additional symbols related to the tuples being processed during that computation.

4 Maximum Search with Multiple Membranes

Similarly to what we have done with priorities we can define a construct analogous to (1) using the rules expressed by (2), this time using $\log_K N$ nested membranes when K-symbol 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 reverse order instead of increasing their numerical value while moving from the outside toward

the inside):

$$
\begin{aligned}
V &= \{x_1, x_2, \ldots, x_N\}, \qquad T = \{x_1\},\\
\mu &= [_{\log N}[_{\log N-1} \cdots [_2[_1 \quad]_1]_2 \cdots]_{\log N-1}]_{\log N},\\
w_1 &= x_1 x_2 \ldots x_N, \qquad w_i = \lambda \quad , \quad i = 2, \ldots \log N,\\
R_i &= \left\{ \begin{array}{lllll} r_1: & x_1 x_{2^{i-1}+1} \to x_1 & \cdots & x_{N-2^i+1} x_{N-2^{i-1}+1} \to x_{N-2^i+1} \\ r_2: & x_1 \to x_1\delta, & \cdots & x_{N-2^i+1} \to x_{N-2^i+1}\delta \\ & x_{2^{i-1}+1} \to x_1\delta, & \cdots & x_{N-2^{i-1}+1} \to x_{N-2^i+1}\delta \end{array} \right\}, \qquad (5)\\
& i = 1, 2, \ldots, \log N - 1\\
R_{\log N} &= \left\{ \begin{array}{ll} r_1: & x_1 x_{N/2+1} \to x_{1\,out} \\ r_2: & x_1 \to x_{1\,out} \\ & x_{N/2+1} \to x_{1\,out} \end{array} \right\},\\
\rho_i &= \{r_1 > r_2\}, \quad i = 1, 2, \ldots, \log N
\end{aligned}
$$

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 time 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 copies 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 of the previous rules are needed to perform the same kind of processing occurred in membrane 1, to select new relative maxima from the existing pairs;
- an identical processing happens in membrane 3, 4, and so on. Finally, in the skin membrane (labeled $\log N$) the absolute maximum is computed, and sent out as $|x_1|$.

Compared to the construct (1), the implementation using multiple membranes again needs $3N - 3$ rules with N symbols, to compute the result in $2 \log N$ time steps. Implementing a single N-symbol 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-symbol parallel comparison using multiple priorities on the rules.

5 Maximum Search with Linked Transport

Linked transport can realize the maximum search, provided that a sufficient amount of substances are present in the environment in order to enable the needed exchange of molecules (symbols) via antiport rules. Apart from this technical aspect, the algorithmic mechanism of "vertical" symbol selection is the same as the one 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.

$$
\begin{aligned}
V &= \{x_1, x_2, \ldots, x_N,\ x'_1, x'_2, \ldots, x'_{N/2},\ x''_1, x''_2, \ldots, x''_{N/4},\ \ldots\ ,\ x^{(\log N)},\\
&\quad y_1, y_2, \ldots, y_N,\ y'_1, y'_2, \ldots, y'_{N/2},\ y''_1, y''_2, \ldots, y''_{N/4},\ \ldots\ ,\ y_1^{(\log N-1)}, y_2^{(\log N-1)},\\
&\quad b_1, \ldots, b_{N/2}, b'_1, \ldots, b'_{N/4}, \ldots, b_1^{(\log N-1)},\ c_1, \ldots, c_{N/2}, c'_1, \ldots, c'_{N/4}, \ldots, c_1^{(\log N-1)},\\
&\quad d_1, \ldots, d_{N/2}, d'_1, \ldots, d'_{N/4}, \ldots, d_1^{(\log N-1)},\ a'_1, \ldots, a'_{N/2}, a''_1, \ldots, a''_{N/4}, \ldots, a_1^{(\log N)}\}\\
T &= \{x^{(\log N)}\}, \qquad w = x_1 x_2 \ldots x_N,
\end{aligned}
\tag{6}
$$

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

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

- the initial symbols x_i are exchanged with identical numbers of copies of symbols y_i,

respectively. Meanwhile, the "synchronization" symbols $b_1, \ldots, b_{N/2}$ are imported from the environment (note that $|b_i| = |x_i| + |x_{i+1}| > \max\{|x_i|, |x_{i+1}|\}$);

- the pairs $x_1x_2, \ldots, x_{N-1}x_N$ 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 could not be paired are carried out by c_i and exchanged with additional x'_i. Now, x'_i contains the information on the relative maximum related to the pair (y_i, y_{i+1}). In parallel, the synchronization symbol d_i is exchanged with a'_i, which enables the processing of the new level of pairs (i.e., $x'_1x'_2, \ldots, x'_{N/2-1}x'_{N/2}$) to start.

The horizontal lines drawn between the braces of R separate the computation steps in groups, each related to the parallel processing of a level of pairs. In the final step the symbol $x^{(\log N)}$ is 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 $(2N - 1) + (2N - 2) = 4N - 3$ 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 anyway implemented [**?**].

Once again, higher-degree parallel comparisons can be chosen [**?**]. In particular, we give the construct performing N-symbol parallel comparison (again, horizontal lines in R separate successive computation steps).

$$
\begin{array}{rcl}
V & = & \{x,\ x_1, \ldots, x_N,\ y_1, \ldots, y_N,\ a_1, \ldots, a_{N+1},\ b_0, \ldots, b_N,\ A_1, \ldots, A_N\}, \\
T & = & \{x\}, \qquad\qquad w = x_1x_2 \ldots x_N, \\
R & = & \left\{ \begin{array}{c}
(x_1, \text{out}; b_0A_1y_1, \text{in})\ (x_2, \text{out}; b_0A_2y_2, \text{in})\ \ldots\ (x_N, \text{out}; b_0A_Ny_N, \text{in}), \\
\hline
(y_1y_2 \ldots y_N, \text{out}; x, \text{in}), \\
(b_0, \text{out}; a_1b_1, \text{in}), \\
\hline
(a_1y_2 \ldots y_N, \text{out}; x, \text{in})\ (a_1y_1y_3 \ldots y_N, \text{out}; x, \text{in})\ \ldots\ (a_1y_1 \ldots y_{N-1}, \text{out}; x, \text{in}), \\
(b_1, \text{out}; a_2b_2, \text{in}), \\
\hline
\ldots \\
\left(\binom{N}{i} \text{combinations of } y_i \text{ carried out by } a_i; x, \text{in}\right) \\
(b_i, \text{out}; a_{i+1}b_{i+1}, \text{in}), \\
\ldots \\
\hline
(a_NA_1y_1, \text{out}; x, \text{in})\ (a_NA_2y_2, \text{out}; x, \text{in})\ \ldots\ (a_NA_Ny_N, \text{out}; x, \text{in}), \\
(b_N, \text{out}; a_{N+1}, \text{in}), \\
\hline
(a_{N+1}x, \text{out})
\end{array} \right\}
\end{array}
\tag{7}
$$

The system first exchanges x_i with y_i respectively preserving the number of copies, furthermore acquires the synchronizing symbol b_0. Next it exchanges with x as many N-uples of symbols y_i as possible, 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 sync and carrier symbols through rules of the type $(b_i, \text{out}; a_{i+1}b_{i+1}, \text{in})$. During the last step, all x's 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-symbol comparison allows to search the maximum value straightforwardly. In the above implementation we have in fact added the symbols $A_1, \ldots, A_N$ that enter the membrane during the first computation step and turn out to be useful in the end of the computation: since at most one of the rules of the type $(a_N A_i y_i, \text{out}; x, \text{in})$ will be activated in practice, then A_i will signal that x_i has the largest number of copies as long as it is sent out of the membrane. As briefly discussed in section 3, the search for the maximum value can be extended to the multiple case, by sending out additional symbols in correspondence to each rule of the type $(\text{something}, \text{out}; x, \text{in})$.

The system implementing maximum search using linked transport with N-symbol comparison computes the result in N steps, plus two additional steps needed to exchange molecules respectively in the beginning and the end of the process. $2N+1$ informative symbols and $2N+2$ auxiliary symbols suffice, though—refer to (4)—$(2^N-1)+N+N+1 = 2^N+2N = O(2^N)$ rules are present in the system. Additional N symbols are needed to compute also the maximum value (if unique).

6 "Vertical" Approach to the Maximum Search

The vertical approach consists in a parallel counting of symbols with a sort of elimination of those having a lower number of copies. The iteration of this process finally leaves a membrane which is related to the symbol with maximum number of copies, and all these copies are produced by the system and sent to the environment as output of the algorithm for maximum search.

In order to compute the maximum number of copies among N given elements $a_1, \ldots, a_N$ we consider the following P system, in which standard evolution rules and few priorities are used.

$$V = \{a_1, \ldots, a_N, a'_1, \ldots, a'_N, d_1, \ldots, d_N\}, \quad T = \{a_1, \ldots, a_N\}$$
$$\mu = [_s\, [_1\,]_1 \quad [_2\,]_2 \ \ldots\ [_N\,]_N \quad]_s, \quad w_i = a'_i a_i^{k_i}, \quad k_i = |a_i|, \quad i = 1, \ldots, N$$
$$R_i = \left\{ r_1 :\ a'_i a'_i \to a'_i d_{i\,out}, \quad r_2 :\ a'_i a_i \to a'_i a_{i\,out}, \quad r_3 :\ a'_i \to a'_i \delta \right\}$$
$$\rho_i = \{r_1 > r_2 > r_3\}, \qquad i = 1, \ldots, N$$

$$R_s = \left\{ \begin{array}{rcl} a'_1 a'_2 \ldots a'_{N-2} a'_{N-1} & \to & a'_{N\,in_N}, \\ a'_1 a'_2 \ldots a'_{N-2} a'_N & \to & a'_{N-1\,in_{N-1}} \\ & \vdots & \\ a'_1 a'_3 \ldots a'_{N-1} a'_N & \to & a'_{2\,in_2} \\ a'_2 a'_3 \ldots a'_{N-1} a'_N & \to & a'_{1\,in_1} \\ d_1 a_1 & \to & d_1 a_{1\,out} \\ d_2 a_2 & \to & d_2 a_{2\,out} \\ & \vdots & \\ d_N a_N & \to & d_N a_{N\,out} \end{array} \right\} \tag{8}$$

The system evolution can be described by the following steps:

- Initially, we have N membranes floating in the skin region; each of them, labeled i, with $i = 1, \ldots, N$, contains all copies of symbol a_i and one copy of a'_i.
- In the first time step only r_2 rules are applied (with maximum parallelism) inside each membrane, until membrane j with the minimum number of elements contains only its primed symbol. Since no rules with higher priority can be applied, then r_3 (with lower priority) is applied and membrane j is dissolved. Consequently, a'_j will pass in the skin region.
- The previous step is performed until $N-1$ different primed symbols are present in the skin region, that is, until $N-1$ membranes (containing at every 'dissolution step' the minimum number of elements) have been dissolved. In fact, in this case one of the first N rules of R_s—see (8)—is applied, and the presence of two copies of a primed symbol in a membrane blocks the application of the second rule (just for one step), because the first one with major priority has to be applied.
- If a_i is the symbol with maximum number of copies, say m, at this point a process of 'exit' from the skin membrane of the copies a_i is triggered. In fact, the $(N+i)$th rule of R_s is applied while also r_2 is applied inside the membrane i. When r_2 is no longer applicable (that is, all copies of a_i are out of the membrane), the rule r_3 provides to eliminate the last internal membrane.
- As usual, the process halts when there are no rules left to apply; in this system the final configuration is $\mu = [_s a_1^{k_1} \ldots a_{i-1}^{k_{i-1}} a_{i+1}^{k_{i+1}} \ldots a_N^{k_N} d_i \,]_s$, and all and only the m copies of a_i are sent by the system to the environment.

Differently from the other approaches the number of computational steps here depends on the initial values of the multiplicities (and in particular on the maximum). In fact, initially a number of steps equal to the second greater multiplicity, say m_1, is performed, and then, in further m steps, the m copies of 'maximum-symbol' are produced externally.

In general the total number of computational steps is thus $O(m)$, but this complexity can be reduced by increasing the number of primed symbols entering the membrane containing the maximum and, hence, the occurrences of the carrier d. This can be done by changing the rules in the skin region into:

$$R_s = \left\{ \begin{array}{rcl} a'_1 a'_2 \ldots a'_{N-2} a'_{N-1} & \rightarrow & a'^{h}_{N\, in_N}, \\ a'_1 a_2{}' \ldots a'_{N-2} a'_N & \rightarrow & a'^{h}_{N\, in_N} \\ & \vdots & \\ a'_1 a'_3 \ldots a'_{N-1} a'_N & \rightarrow & a'^{h}_{2\, in_2} \\ a'_2 a'_3 \ldots a'_{N-1} a'_N & \rightarrow & a'^{h}_{1\, in_1} \\ d_1 a_1 & \rightarrow & d_1 a_{1\, out} \\ d_2 a_2 & \rightarrow & d_2 a_{2\, out} \\ & \vdots & \\ d_N a_N & \rightarrow & d_N a_{N\, out} \end{array} \right\}$$

If h is equal to $2m_1$ then, by means of the carrier, the system sends out m_1 copies of the maximum symbol in the first step, and the total number of computation steps becomes exactly m.

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	$O(max)$	$O(N)$	$O(N)$	$O(N)$	$O(1)$

Table 1: Figures of performance for maximum search under different membrane systems (P: priorities; NM: nested membranes; LT: linked transport; 2-SPC: 2-symbol parallel comparison; N-SPC: N-symbol parallel comparison), V: vertical approach.

The vertical approach is effective when we have a lot of objects with low multiplicity. Likewise, strategies for searching the maximum value can be set up avoiding the exponential explosion of the resources as it happens with the horizontal approach.

7 Summary

Starting from the idea of modeling a dynamical sequence alignment algorithm using P systems, we have finally focused on a specific point of it, that is, the maximum search between elements 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 search algorithms based on two orthogonal approaches to the problem, and different P-systems. This performance is summarized in table 1 in terms of computational time and resources needed to figure out the solution.

The overall investigation on sequence alignment might reveal unexpected links between membrane systems and DNA computing paradigms, due to the close relationship existing between dynamical sequence alignment and several DNA processes. 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.

The authors want to acknowledge in particular 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.

Acknowledgements. The authors want to acknowledge in particular 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 Sevilla, Spain.

References

[1] A. Alhazov and D. Sburlan, Static Sorting Algorithms for P Systems, *Proc. Workshop on Membrane Computing* (A. Alhazov, C. Martín-Vide, and G. Păun, eds.),

Tarragona, July 2003, 17–40.

[2] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, Basic Local Alignment Search Tool, *J. Molecular Biology*, 215 (1990), 403–410.

[3] R. Ceterchi, R. Gramatovici, and N. Jonoska, P Systems for Tiling Rectangular Pictures, *Proc. Workshop on Membrane Computing* (A. Alhazov, C. Martín-Vide, and G. Păun, eds.), Tarragona, July 2003, 133–144.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

[5] R.A. Dwyer, *Genomic Perl*, Cambridge University Press, New York, 2003.

[6] F. Fontana, G. Franco, and V. Manca, P Systems in Bio Systems, in *Applications on Membrane Systems* (G. Ciobanu, G. Păun, and M.J. Pérez-Jiménez, eds.), Springer, 2004 (in progress).

[7] D. Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, New York, 1997.

[8] D.J. Lipman and W.R. Pearson, Rapid and Sensitive Protein Similarity Searches, *Science*, 227 (1985), 1435–1441.

[9] C. Martín-Vide, V. Mitrana, and G. Păun, On the Power of P Systems with Valuations, *Computacion Sistemas*, 5 (2001), 120–127.

[10] G. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[11] G. Păun and G. Rozenberg, A Guide to Membrane Computing, *Theoretical Computer Science*, 287 (2002), 73–100.