Shallow Non-Confluent P Systems*

Alberto Leporati, Luca Manzoni, Giancarlo Mauri,
Antonio E. Porreca, and Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
{leporati, luca.manzoni, mauri, porreca, zandron}@disco.unimib.it

Abstract. We prove that non-confluent (i.e., strongly nondeterministic) P systems with active membranes working in polynomial time are able to simulate polynomial-space nondeterministic Turing machines, and thus to solve all **PSPACE** problems. Unlike the confluent case, this result holds for shallow P systems. In particular, depth 1 (i.e., only one membrane nesting level and using elementary membrane division only) already suffices, and neither dissolution nor send-in communication rules are needed.

1 Introduction

Families of *confluent* recogniser P systems with active membranes [9] are known to characterise **PSPACE** in polynomial time [13,14]. In this kind of P systems the computations can be locally nondeterministic, but the final result (acceptance or rejection) must be consistent across all computations. This result seems to require that the membrane nesting depth of each P system in the family depends on the length of the input (the published results show that a *linear* depth suffices [13]); furthermore, all known algorithms employ non-elementary membrane division (i.e., division of membranes containing further membranes, resulting in the replication of whole subtrees of the membrane structure).

More recently, it has been proved [2] that when only elementary division (i.e., division for membranes not containing further membranes) is available, the power of P systems decreases to $\mathbf{P}^{\#\mathbf{P}}$, the class of problems solved in polynomial time by deterministic Turing machines with an oracle for a counting problem [8]; this class is conjectured to be strictly smaller than **PSPACE**. More specifically, P systems of depth 1 (consisting of an outermost membrane containing only elementary membranes) already characterise $\mathbf{P}^{\#\mathbf{P}}$ [3], and thus increasing the depth without also allowing non-elementary division does not increase the computing power. P systems of depth 0, where there exists only one membrane and division is unavailable, are known to characterise **P** [15,3].

* This work was partially supported by Fondo d'Ateneo (FA) 2015 of Università degli Studi di Milano-Bicocca: "Complessità computazionale e applicazioni crittografiche di modelli di calcolo bioispirati".

Fewer results are known for non-confluent P systems, where the computations need not agree on the result, and the overall behaviour is accepting if and only if there exists an accepting computation (i.e., a strongly nondeterministic behaviour analogous to Turing machines). Clearly, all lower bounds of confluent P systems hold for non-confluent ones. The only other published result concerning non-confluent recogniser P systems with active membranes, to the authors' best knowledge, is a characterisation of **NP** by means of polynomial-time non-confluent P systems with active membranes without any kind of membrane division [11].

Membrane division in confluent P systems is commonly used to simulate the effect of nondeterminism, by exploring in parallel all possible nondeterministic choices and combining the results by disjunction [15], threshold or majority [3], or alternation of conjunctions and disjunctions [13], depending on which rules are available and the depth of the membrane structure. It is then natural to ask whether these results can be somehow improved by employing actual nondeterminism, i.e., by exploiting non-confluence in addition to membrane division.

In this paper we prove that this is indeed the case, since the lower bound **PSPACE** can actually be reached by "shallow" (small-depth) polynomial-time non-confluent P systems: specifically, depth 1, and thus division only for elementary membranes, already suffice for reaching **PSPACE**. Furthermore, the P systems employed can be *monodirectional* [1,4], i.e., without using send-in communication rules. Monodirectionality is known to decrease the power of confluent P systems; for instance, polynomial-time monodirectional confluent P systems of depth 1 characterise $\mathbf{P}_{\parallel}^{\mathbf{NP}}$ (the class of problems solved in polynomial time with parallel queries to an **NP** oracle, conjectured to be smaller than $\mathbf{P}^{\#\mathbf{P}}$), and $\mathbf{P}^{\mathbf{NP}}$ (the class where the oracle queries are not restricted to be parallel, which is probably smaller than **PSPACE**) if the depth is unbounded [4].

2 Basic Notions

In this paper we use P systems with active membranes [9] using only object evolution rules $[a \to w]_h^\alpha$, send-out communication rules $[a]_h^\alpha \to [\ ]_h^\beta\ b$ and elementary membrane division rules $[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma$.

The *depth* of a P system is defined as the depth of its membrane structure when considered as a rooted tree, i.e., as the length of the longest path from the outermost membrane to an elementary membrane.

In particular, we are dealing with *recogniser P systems* Π [10], whose alphabet includes the distinguished result objects yes and no; exactly one result object must be sent out from the outermost membrane to signal acceptance or rejection, and only at the last computation step. If all possible computations of Π agree on the result, the P system is said to be *confluent*. In this paper, however, we only deal with the more general *non-confluent* recogniser P systems, where different computations need not agree on the result, and the final result is acceptance if and only if *at least one* computation is accepting.

A decision problem, or language $L \subseteq \Sigma^\star$, is solved by a *family* of P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$, where Π_x accepts if and only if $x \in L$. In that case, we

say that $L(\boldsymbol{\Pi}) = L$. As usual, we require a uniformity condition [6] on families of P systems:

Definition 1. *A family of P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is* (polynomial-time) uniform *if the mapping $x \mapsto \Pi_x$ can be computed by two polynomial-time deterministic Turing machines E and F as follows:*

- *$F(1^n) = \Pi_n$, where n is the length of the input x and Π_n is a common P system for all inputs of length n, with a distinguished input membrane.*
- *$E(x) = w_x$, where w_x is a multiset encoding the specific input x.*
- *Finally, Π_x is simply Π_n with w_x added to its input membrane.*

The family $\boldsymbol{\Pi}$ is said to be (polynomial-time) semi-uniform if there exists a single deterministic polynomial-time Turing machine H such that $H(x) = \Pi_x$ for each $x \in \Sigma^\star$.

The class of decision problems solved by uniform families of non-confluent P systems with active membranes working in polynomial time is denoted by the symbol $\mathbf{NPMC}_{\mathcal{AM}}$. The corresponding class for families of P systems with depth-1 membrane structures using only object evolution, send-out communication and elementary membrane division rules is denoted by $\mathbf{NPMC}_{\mathcal{AM}(\text{depth-1},-\text{i},-\text{d},-\text{ne})}$, where $-$i, $-$d, and $-$ne denote the lack of send-in, dissolution, and non-elementary division rules, respectively. For the complexity classes defined in terms of Turing machines, such as **NP**, $\mathbf{P}_{\|}^{\mathbf{NP}}$, $\mathbf{P}^{\mathbf{NP}}$, $\mathbf{P}^{\#\mathbf{P}}$, and **PSPACE** we refer the reader to Papadimitriou's book [8].

3 Simulating Nondeterministic Turing Machines

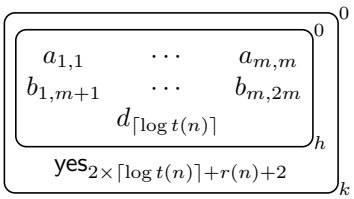
Let N be a nondeterministic Turing machine working in polynomial space $p(n)$. Let Σ be the tape alphabet of N, and let Q be its set of states. Without loss of generality, we assume that N has a unique accepting configuration, consisting of a unique accepting state, an entirely blank tape, and the tape head located on the leftmost position. We can assume that all computations of N halt within exponential time $t(n) = |\Sigma|^{p(n)} \times |Q| \times p(n)$.

Suppose that string x is an input for N, and let $m = p(|x|) + 3$. A configuration $\mathcal{C}$ of N can be encoded as a delimited string $\$a_1 \cdots a_{k-1} q a_k \cdots a_{p(n)}\$$ of length m over the alphabet $\Sigma \cup Q \cup \{\$\}$. This denotes that the tape of N contains the string $a_1 \cdots a_{k-1} a_k \cdots a_{p(n)}$, that the machine is in state q, and that the tape head is located on the k-th tape cell.

Deciding whether N accepts an input x is equivalent to deciding whether the unique final accepting configuration $\mathcal{C}'$ is reachable from the initial configuration $\mathcal{C}$ on input x within $t(|x|)$ steps. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be configurations of N, let $\mathcal{C}_1 \to \mathcal{C}_2$ denote that $\mathcal{C}_2$ is reachable from $\mathcal{C}_1$ via a single computation step, and let $\mathcal{C}_1 \to^t \mathcal{C}_2$ denote reachability within t steps. Then, we have

$$\begin{aligned}
&\mathcal{C}_1 \to^1 \mathcal{C}_2 && \text{iff} && \mathcal{C}_1 = \mathcal{C}_2 \text{ or } \mathcal{C}_1 \to \mathcal{C}_2 \\
&\mathcal{C}_1 \to^t \mathcal{C}_2 \text{ with } t > 1 && \text{iff} && \text{there exists } \mathcal{C} \text{ such that } \mathcal{C}_1 \to^{\lceil t/2 \rceil} \mathcal{C} \to^{\lfloor t/2 \rfloor} \mathcal{C}_2
\end{aligned}$$

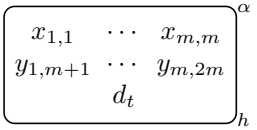
The Turing machine N on input x of length n is simulated by a P system Π_x, whose initial configuration is



(1)

Here the string $a_1 \cdots a_m$ encodes the initial configuration of N on input x, that is, $a_1 \cdots a_m = \$q_0 x \sqcup^{p(n)-n} \$$ with $\sqcup$ denoting a blank cell and q_0 the initial state of N; these symbols need a further subscript in Π_x in order to keep track of their position within the string. Analogously, the string $b_1 \cdots b_m$ encodes the unique accepting configuration of N, that is, $b_1 \cdots b_m = \$q_{\mathsf{yes}} \sqcup^{p(n)} \$$, where q_{yes} is the accepting state. In this case, the symbols are subscripted with $m+1, \ldots, 2m$ as if the P system stored a single string $a_1 \cdots a_m b_1 \cdots b_m$; this will prove useful in a later phase of the simulation. The other objects do not encode information about N, but play an auxiliary role in the simulation; in particular, the function $r(n)$, appearing in the subscript of the object yes, will be defined later.

For the whole first phase of the computation of Π_x (including the initial configuration) the membranes with label h maintain, as an invariant, a configuration of the form



(2)

where $1 \leq t \leq \lceil \log t(n) \rceil$, the charge α is either 0 or $+$, and $x_{1,1} \cdots x_{m,m}$ and $y_{1,m+1} \cdots y_{m,2m}$ are multisets respectively encoding the strings $x_1 \cdots x_m$ and $y_1 \cdots y_m$, which in turn encode two configurations $\mathcal{C}_1$ and $\mathcal{C}_2$ of N as described above. This invariant is restored every two steps of the first phase of the computation of Π_x.

The purpose of this membrane, for $t > 1$, is to guess a computation path of length at most 2^t from $\mathcal{C}_1$ to $\mathcal{C}_2$; computation paths from $\mathcal{C}_1 \to \cdots \to \mathcal{C}_2$ shorter than 2^t are padded to length 2^t by repeating some intermediate configurations (recall that $\mathcal{C} \to^1 \mathcal{C}$ for all $\mathcal{C}$). If $t = 0$, the membrane checks whether the configuration $\mathcal{C}_2$ is reachable by N in one step from $\mathcal{C}_1$; if it is the case, then it outputs an object yes, and otherwise an object no after exactly $2t + r(n) + 1$ computation steps.

Let us describe recursively the behaviour of the membrane. If $t > 0$, then the problem of guessing a computation $\mathcal{C}_1 \to^{2^t} \mathcal{C}_2$ is divided into the two subproblems of guessing a computation $\mathcal{C}_1 \to^{2^{t-1}} \mathcal{C}$ and a computation $\mathcal{C} \to^{2^{t-1}} \mathcal{C}_2$, where $\mathcal{C}$ is a nondeterministically guessed mid-point. This mid-point is guessed by rewriting each object σ_i of the multiset $x_{1,1} \cdots x_{m,m}$ into a primed version σ'_i of itself,

together with an object τ_i'', where τ is a nondeterministically chosen symbol of the alphabet of the configurations of N:

$$[\sigma_i \to \sigma_i' \, \tau_i'']_h^\alpha \qquad \text{for } \alpha \in \{0,+\},\ \sigma,\tau \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m \tag{3}$$

Notice that the P system guesses an arbitrary string as a configuration $\mathcal{C}$; the string might even be an invalid encoding (e.g., with multiple symbols denoting the state of N); the validity of the configuration will be checked later. Simultaneously, the objects of the target configuration $y_{1,m+1} \cdots y_{m,2m}$ are primed:

$$[\sigma_i \to \sigma_i']_h^\alpha \qquad \text{for } \alpha \in \{0,+\},\ \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } m+1 \le i \le 2m \tag{4}$$

While the objects encoding the configurations of N are thus rewritten, the membrane is divided by d_t into two membranes differing only in their charge:

$$[d_t]_h^\alpha \to [d_t']_h^+ \, [d_t']_h^0 \qquad \text{for } \alpha \in \{0,+\} \text{ and } 1 \le t \le \lceil \log t(n) \rceil$$

Hence, the original membrane h leads to the following configuration:

$$\underset{h}{\boxed{\begin{array}{ccc} x'_{1,1} & \cdots & x'_{m,m} \\ z''_{1,1} & \cdots & z''_{m,m} \\ y'_{1,m+1} & \cdots & y'_{m,2m} \\ & d'_t & \end{array}}}^{+} \qquad \underset{h}{\boxed{\begin{array}{ccc} x'_{1,1} & \cdots & x'_{m,m} \\ z''_{1,1} & \cdots & z''_{m,m} \\ y'_{1,m+1} & \cdots & y'_{m,2m} \\ & d'_t & \end{array}}}^{0}$$

where the objects $z''_{i,i}$ represent the mid-point configuration $\mathcal{C}$ guessed by the membrane. Now configuration $\mathcal{C}$ becomes the target configuration in the left membrane, having charge $+$. This requires eliminating all primes, deleting the objects $y'_{1,m+1} \cdots y'_{m,2m}$ and adjusting the subscripts of $z''_{1,1} \cdots z''_{m,m}$; this is performed by the following rules:

$$\begin{array}{ll} [\sigma_i' \to \sigma_i]_h^+ & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m \\ [\sigma_i' \to \epsilon]_h^+ & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } m+1 \le i \le 2m \\ [\sigma_i'' \to \sigma_{i+m}]_h^+ & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m \end{array}$$

On the other hand, the configuration $\mathcal{C}$ becomes the source configuration in the right membrane (with charge 0), where the objects $x'_{1,1} \cdots x'_{m,m}$ must be deleted:

$$\begin{array}{ll} [\sigma_i' \to \epsilon]_h^0 & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m \\ [\sigma_i' \to \sigma_i]_h^0 & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } m+1 \le i \le 2m \\ [\sigma_i'' \to \sigma_i]_h^0 & \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m \end{array}$$

Finally, the object d_t' is rewritten into d_{t-1} inside both membranes:

$$[d_t' \to d_{t-1}]_h^\alpha \qquad \text{for } \alpha \in \{0,+\} \text{ and } 1 \le t \le \lceil \log t(n) \rceil$$

Hence, the P system reaches the configuration

$$\left[\begin{array}{ccc} x_{1,1} & \cdots & x_{m,m} \\ z_{1,m+1} & \cdots & z_{m,2m} \\ & d_{t-1} & \end{array}\right]_h^+ \qquad \left[\begin{array}{ccc} z_{1,1} & \cdots & z_{m,m} \\ y_{1,m+1} & \cdots & y_{m,2m} \\ & d_{t-1} & \end{array}\right]_h^0$$

with two membranes having a configuration of the form (2). This restores the associated invariant.

After $2 \times \lceil \log t(n) \rceil$ steps (twice the initial subscript of the object d_t), all membranes with label h simultaneously reach a configuration of the form

$$\left[\begin{array}{ccc} x_{1,1} & \cdots & x_{m,m} \\ y_{1,m+1} & \cdots & y_{m,2m} \\ & d_0 & \end{array}\right]_h^\alpha$$

for some $\alpha \in \{0, +\}$ and $x_1, \ldots, x_m, y_1, \ldots, y_m \in \Sigma \cup Q \cup \{\$\}$. The last phase of the simulation is triggered by objects d_0 being sent out and changing the charge of the membranes to $-$:

$$[d_0]_h^\alpha \to [\,]_h^- \#\qquad\qquad \text{for } \alpha \in \{0, +\} \tag{5}$$

While rule (5) is applied, the charge of the membrane is still 0 or $+$, and the rules of type (3) and (4) are still enabled; thus, each membrane h reaches a configuration of the form

$$\left[\begin{array}{ccc} x'_{1,1} & \cdots & x'_{m,m} \\ z''_{1,1} & \cdots & z''_{m,m} \\ y'_{1,m+1} & \cdots & y'_{m,2m} \end{array}\right]_h^-$$

When the charge of a membrane with label h is $-$, the objects of the form τ''_i (which have been just produced, but are actually not needed in this phase) are deleted:

$$[\tau''_i \to \epsilon]_h^- \qquad\qquad \text{for } \tau \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le m$$

The remaining objects σ'_i are rewritten into a "tilded" version. This allows us to re-use the charges 0 and $+$ in the next phase of the computation (which will make use of a simulation from [3]) without creating conflicts with previous rules.

$$[\sigma'_i \to \tilde{\sigma}_i]_h^- \qquad\qquad \text{for } \sigma \in \Sigma \cup Q \cup \{\$\} \text{ and } 1 \le i \le 2m$$

This leads to the configuration

$$\left[\begin{array}{ccc} \tilde{x}_{1,1} & \cdots & \tilde{x}_{m,m} \\ \tilde{y}_{1,m+1} & \cdots & \tilde{y}_{m,2m} \end{array}\right]_h^- \tag{6}$$

Each membrane h now contains what can be considered as a string of length $2m$, consisting of the concatenation of two (possibly malformed) encodings of configurations of N.

From [3] we know that a single membrane is able to efficiently simulate a polynomial-time *deterministic* Turing machine (as long as there is no communication with adjacent membranes) if the tape is encoded as in configuration (6). The idea is to simulate a larger number of charges (referred to as *extended* charges) by encoding them in the subscripts of each object; the subscripts are kept synchronised by multiple sequential updates of the actual charges $\{+, 0, -\}$. The extended charges are employed in order to store pairs (q, i) of state and tape head position of the simulated Turing machine. A transition such as $\delta(q, a) = (q', a', +1)$ is then implemented as a rule of the form $[a_i]_h^{(q,i)} \to [a'_i]_h^{(q',i+1)}$, which is actually carried out in multiple steps using standard charges, object evolution and send-out communication rules.

In our particular case, each membrane h can simulate a Turing machine that checks whether the content of such membrane consists of two valid consecutive configurations of N, or two identical valid configurations of N. We can assume, without loss of generality, that such Turing machine halts exactly in polynomial time $r(n)$ for all strings of length n; the result is given by outputting yes or no from the membrane.

After $2 \times \lceil \log t(n) \rceil + r(n)$ steps, a total of $2^{\lceil \log t(n) \rceil}$ instances of yes and no reach the outermost membrane k. If there is at least one instance of no, then there existed an instance of membrane h containing either an invalid configuration, or two non-consecutive configurations; this means that the P system Π_x guessed a malformed computation of N. In that case, it sends out any of the objects no as the final result of the computation:

$$[\mathsf{no}]_k^0 \to [\ ]_k^- \ \mathsf{no}$$

If there is no instance of no inside k, then all membranes h contained valid consecutive configurations (or pairs of identical valid configurations). Since the initial membrane h contained the initial and final configurations of N on input x, this means that Π_x guessed a legitimate accepting computation of N. In that case, all objects yes sent out from the elementary membranes h are ignored, and instead the timed object yes_t, which already appears in the initial configuration (1), produces the output of the P system. This object counts down for the entire duration of the simulation:

$$[\mathsf{yes}_t \to \mathsf{yes}_{t-1}]_k^0 \qquad \text{for } 1 \leq t \leq 2 \times \lceil \log t(n) \rceil + r(n) + 2$$

If at time $2 \times \lceil \log t(n) \rceil + r(n) + 2$ membrane k still has charge 0, then the P system has not rejected, and it can send out yes_0 as yes, as the result:

$$[\mathsf{yes}_0]_k^0 \to [\ ]_k^- \ \mathsf{yes}$$

In both cases Π_x halts by sending out a result at the last computation step.

The P system Π_x has thus an accepting computation if and only if there exists a computation path from the initial configuration of N on input x to its accepting computation, that is, if and only if N accepts.

Notice that the mapping $x \mapsto \Pi_x$ is uniform, since all rules of Π_x only depend on the *length* of the input, and not on the input itself. Furthermore, the initial membrane structure is the same for all Π_x. The only portion of the initial configuration that depends on the actual input x is the content of membrane h, which is chosen as the input of the P system. The mapping $x \mapsto \Pi_x$ can also be computed in polynomial time: the encoding of the input simply consists in adding subscripts to the input symbols of x, and the rules are easy to compute, since they all range over sets independent of the input, or over sets of natural numbers depending on the input length, and never require sophisticated computation.

Theorem 1. *Let N be a nondeterministic Turing machine working in polynomial space. Then, there exists a uniform family $\boldsymbol{\Pi}$ of non-confluent P systems of depth 1, using only object evolution, send-out and elementary division rules, and working in polynomial time such that $L(N) = L(\boldsymbol{\Pi})$.* □

Since arbitrary polynomial-space Turing machines can be simulated, the whole class they characterise is solved by shallow non-confluent P systems with a limited range of rules:

Corollary 1. $\mathbf{PSPACE} \subseteq \mathbf{NPMC}_{\mathcal{AM}(\text{depth-1},-\text{i},-\text{d},-\text{ne})}$. □

4 Conclusions

The results obtained in this paper show that, even with depth-1 membrane structures and monodirectional communication, non-confluent P systems with active membranes are already able to solve **PSPACE**-complete problems in polynomial time, and are thus conjecturally stronger than confluent ones with the same restrictions (and even those with only one of such restrictions).

This result is a first step towards a characterisation of the power of polynomial-time non-confluent P systems with active membranes. If **PSPACE** turned out to also be an upper bound, although it has been conjectured that it might not be so [14], this would show that non-confluence subsumes both nesting depth beyond 1 and bidirectionality. In that case, it would be interesting to find other parameters (such as unusual combinations of admissible rules) which can be "tuned" in order to obtain complexity classes between **NP** and **PSPACE**.

We also conjecture that an algorithm similar to the one provided here for P systems with active membranes can also be implemented for tissue-like P systems using either cell division [12] or cell separation rules [7], since they seem to share several features and limitations of cell-like P systems of depth 1 [5].

References

1. Alhazov, A., Freund, R.: On the efficiency of P systems with active membranes and two polarizations. In: Mauri, G., Păun, Gh., Pérez-Jiménez, M.J., Rozenberg,

G., Salomaa, A. (eds.) Membrane Computing, 5th International Workshop, WMC 2004. Lecture Notes in Computer Science, vol. 3365, pp. 146–160. Springer (2005)
2. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Simulating elementary active membranes, with an application to the P conjecture. In: Gheorghe, M., Rozenberg, G., Sosík, P., Zandron, C. (eds.) Membrane Computing, 15th International Conference, CMC 2014, Lecture Notes in Computer Science, vol. 8961, pp. 284–299. Springer (2014)
3. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Membrane division, oracles, and the counting hierarchy. Fundamenta Informaticae 138(1–2), 97–111 (2015)
4. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Monodirectional P systems. In: Macìas-Ramos, L.F., Păun, Gh., Riscos-Núñez, A., Valencia-Cabrera, L. (eds.) Proceedings of the 13th Brainstorming Week on Membrane Computing. pp. 207–226. Fénix Editora (2015)
5. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Tissue P systems can be simulated efficiently with counting oracles. In: Rozenberg, G., Salomaa, A., Sempere, J.M., Zandron, C. (eds.) Membrane Computing, 16th International Conference, CMC 2015. Lecture Notes in Computer Science, vol. 9504, pp. 251–261 (2015)
6. Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions. Natural Computing 10(1), 613–632 (2011)
7. Pan, L., Pérez-Jiménez, M.J.: Computational complexity of tissue-like P systems. Journal of Complexity 26(3), 296–315 (2010)
8. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1993)
9. Păun, Gh.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics 6(1), 75–90 (2001)
10. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing 2(3), 265–284 (2003)
11. Porreca, A.E., Mauri, G., Zandron, C.: Non-confluence in divisionless P systems with active membranes. Theoretical Computer Science 411(6), 878–887 (2010)
12. Păun, Gh., Pérez-Jiménez, M.J., Riscos Núñez, A.: Tissue P systems with cell division. International Journal of Computers, Communications & Control 3(3), 295–303 (2008)
13. Sosík, P.: The computational power of cell division in P systems: Beating down parallel computers? Natural Computing 2(3), 287–298 (2003)
14. Sosík, P., Rodríguez-Patón, A.: Membrane computing and complexity theory: A characterization of PSPACE. Journal of Computer and System Sciences 73(1), 137–152 (2007)
15. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference, pp. 289–301. Springer (2001)