Solving SAT with Antimatter in Membrane Computing

Daniel Díaz-Pernil[1], Artiom Alhazov[2], Rudolf Freund[3],
Miguel A. Gutiérrez-Naranjo[4]

[1] Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics - University of Sevilla, 41012, Spain
E-mail: sbdani@us.es
[2] Institute of Mathematics and Computer Science, Academy of Sciences of Moldova
Academiei 5, Chişinău, MD-2028, Republic of Moldova
E-mail: artiom@math.md
[3] Faculty of Informatics, Vienna University of Technology,
Favoritenstr. 9, 1040 Vienna, Austria
E-mail: rudi@emcc.at
[4] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence, University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: magutier@us.es

Summary. The set of **NP**-complete problems is split into *weakly* and *strongly* **NP**-complete ones. The difference consists in the influence of the encoding scheme of the input. In the case of weakly **NP**-complete problems, the intractability depends on the encoding scheme, whereas in the case of strongly **NP**-complete problems the problem is intractable even if all data are encoded in a unary way. The reference for *strongly* **NP**-complete problems is the Satisfiability Problem (the **SAT** problem). In this paper, we provide a uniform family of P systems with active membranes which solves **SAT** – without polarizations, without dissolution, with division for elementary membranes and with matter/antimatter annihilation. To the best of our knowledge, it is the first solution to a *strongly* **NP**-complete problem in this P system model.

1 Introduction

In [7], a solution of the Subset Sum problem in the polynomial complexity class of recognizer P systems with active membranes without polarizations, without dissolution and with division for elementary membranes endowed with antimatter and matter/antimatter annihilation rules was provided. In this way, antimatter was shown to be a frontier of tractability in Membrane Computing, since the P systems class without antimatter and matter/antimatter annihilation rules is exactly the complexity class P (see [10]).

The Subset Sum problem belongs to the so-called *weakly* **NP**-complete problems, since its intractability strongly depends on the fact that extremely large input numbers are allowed [8]. The reason for this *weakness* is based on the encoding scheme of the input, since every integer in the input denoting a weight w_i should be encoded by a string of length only $O(\log w_i)$.

On the other hand, *strongly* **NP**-complete problems are those which remain **NP**-complete even if the data are encoded in a unary way. The best-known one of these problems is the satisfiability problem (**SAT** for short). **SAT** was the first problem shown to be **NP**-complete, as proved by Stephen Cook at the University of Toronto in 1971 [5], and it has been widely used in Membrane Computing to prove the ability of a P system model to solve **NP**-problems (e.g. [9, 11, 12, 14, 16, 17]).

In this paper, we provide a solution to the **SAT** problem in the polynomial complexity class of recognizer P systems with active membranes without polarizations, without dissolution and with division for elementary membranes endowed with antimatter and matter/antimatter annihilation rules. To the best of our knowledge, this is the first time that a *strongly* **NP**-complete problem is solved in this P system model. The details of the implementation can provide new tools for a better understanding of the problem of searching new frontiers of tractability in Membrane Computing.

The paper is organized as follows. In Section 2, we present a general discussion about the relationship of model ingredients used in different solutions for solving computationally difficult problems by P systems with active membranes, and the emerging computational power. In Section 3, we recall the P systems model used in this paper. The main novelty is the use of antimatter and matter/antimatter annihilation rules as well as their semantics. In Section 4, some basics on recognizer P systems are recalled, and in Section 5 our solution for the **SAT** problem is provided. The paper finishes with some conclusions and hints for future work.

2 Computation Theory Remarks

A configuration consists of symbols (which, in the general sense, may include instances of objects, instances of membranes, or any other entities bearing information). A computation consists of transformations of symbols. Clearly, the computations without cooperation of symbols are quite limited in power (e.g., it is known that $E0L$-behavior with standard halting yields $PsREG$, and accepting P systems are considerably more degenerate).

In this sense, interaction of symbols is a fundamental part of Membrane Computing, or of Theoretical Computer Science in general. Various ways of interaction of symbols have been studied in membrane computing. For the models with active membranes, the most commonly studied ways are various rules changing polarizations (or even sometimes labels), and membrane dissolution rules. One object may engage such a rule, which would affect the *context* (polarization or label) of other objects in the same membrane, thus affecting the behavior of the latter, e.g., in

case of dissolution, such objects find themselves in the parent membrane, which usually has a different label.

In the literature on P systems with active membranes, normally only the rules with at most one object on the left side were studied. Since recently, the model with matter/antimatter annihilation rules, e.g. see [1] and [2], attracted the attention of researchers. Clearly, it provides a form of *direct* object-object interaction, albeit in a rather restricted way (i.e., by erasing a pair of objects that are in a bijective relation). Although it is known that non-cooperative P systems with antimatter are already universal, studying their efficiency turned out to be an interesting line of research. So how does matter/antimatter annihilation compare to other ways of organizing interaction of objects?

First, all known solutions of **NP**-complete (or more difficult) problems in membrane computing rely on the possibility of P systems to obtain *exponential space* in polynomial time (note that object replication alone does not count as building exponential space, since an exponential number can be written, e.g. in binary, in polynomial space). Such possibility is provided by either of membrane division rules, membrane separation rules, membrane creation rules (or string replication rules, but string-objects lie outside of the scope of the current paper); in tissue P systems, one could apply similar approach to cells instead of membranes.

Note that in case of cell-like P systems, membrane creation alone (unlike the other types of rules mentioned above) makes it also possible to construct a hierarchy of membranes, let us refer to it as *structured workspace*, which is used to solve **PSPACE**-complete problems. The structured workspace can be alternatively created by elementary membrane division plus non-elementary membrane division (plus membrane dissolution if we have no polarizations).

Besides creating workspace, to solve **NP**-complete problems, we need to be able to effectively use that workspace, by making objects interact. For instance, it is known that, even with membrane division, without polarizations and without dissolution only problems in **P** may be solved. However, already with two polarizations (the smallest non-degenerate value) P systems can solve **NP**-complete problems. What can be done without polarizations?

One solution is to use the power of switching the context by membrane dissolution. Coupled with non-elementary division, a suitable membrane structure can be constructed so that the needed interactions can be performed solving **NP**-complete or even **PSPACE**-complete problems, [4]. It is not difficult to realize that elementary and non-elementary division rules can be replaced by membrane creation rules, or elementary division rules can be replaced by separation rules.

Finally, an alternative way of interaction of objects considered in this paper following [7] is matter/antimatter annihilation. What are the strengths and the weaknesses of these three possibilities (the weaker is an ingredient, the stronger is the result, while sometimes a weaker ingredient does not let us do what a stronger one can)?

The power of matter/antimatter annihilation makes it possible to carry out multiple simultaneous interactions (for example, the checking phase is constant-

time instead of linear with respect to the number of clauses), and it is a direct object-object interaction.

The power of polarizations is the possibility of mass action (not critical for studying computational efficiency within **PSPACE** as all multiplicities are bounded with respect to the problem size) by changing context.

The power of non-elementary division lets us build structured workspace (probably necessary for **PSPACE** if membrane creation is not used instead of membrane division, unless $\mathbf{P}^{\mathbf{PP}}$=**PSPACE**), see [13], and change non-local context (e.g., the label of the parent membrane).

The power of dissolution provides mass action (not critical for studying computational efficiency within **PSPACE** as all multiplicities are bounded with respect to the problem size) by changing context.

3 The P System Model

In this paper, we use the common rules of evolution, communication and division of elementary membranes which are usual in P systems with active membranes. The main novelty in the model is the use of antimatter and matter/antimatter annihilation rules. The concept of antimatter was introduced in the framework of Membrane Computing as a control tool for the flow of spikes in spiking neural P systems [15, 18, 22, 23]. In this context, when one spike and one anti-spike appear in the same neuron, the annihilation occurs and both, spike and anti-spike, disappear. Antimatter and matter/antimatter annihilation rules later were adapted to other contexts in Membrane Computing, and currently this an active research area [1, 2, 7].

Inspired by physics, we consider the annihilation of two objects a and b from the alphabet Γ in a membrane with label h, with the annihilation rule for a and b written as $[\ ab \to \lambda\ ]_h$. The *meaning* of the rule follows the idea of annihilation: If a and b occur simultaneously in the same membrane, then both are consumed (disappear) and nothing is produced (denoted by the empty string λ). The object b is called the *antiparticle* of a and it is usually written $\overline{a}$ instead of b.

With respect to the semantics, let us recall that this rule must be applied as many times as possible in each membrane, according to the maximal parallelism. Following the intuition from physics, if a and $\overline{a}$ occur simultaneously in the same membrane h and the annihilation rule $[\ a\overline{a} \to \lambda\ ]_h$ is defined, then it has to be applied, regardless any other option. In this sense, any annihilation rule has priority over all rules of the other types of rules (see [7]).

A P system with active membranes without polarizations, without dissolution and with division of elementary membranes and with annihilation rules is a cell-like P system with rules of the following kinds (following [3], we use subscript 0 for the rule type to represent a restriction that such rule does not depend on polarization and is now allowed to change it; if all rules have this subscript, this is equivalent to saying that the P system is without polarizations):

(a_0) $[\ a \rightarrow u\ ]_h$ for $h \in H$, $a \in \Gamma$, $u \in \Gamma^*$. This is an object evolution rule, associated with a membrane labeled by h: an object $a \in \Gamma$ belonging to that membrane evolves to a string $u \in \Gamma^*$.

(b_0) $a[\ \ ]_h \rightarrow [\ b\ ]_h$ for $h \in H$, $a, b \in \Gamma$. An object from the region immediately outside a membrane labeled by h is taken into this membrane, possibly being transformed into another object.

(c_0) $[\ a\ ]_h \rightarrow b[\ \ ]_h$ for $h \in H$, $a, b \in \Gamma$. An object is sent out from a membrane labeled by h to the region immediately outside, possibly being transformed into another object.

(e_0) $[\ a\ ]_h \rightarrow [\ b\ ]_h[\ c\ ]_h$ for $h \in H$, $a, b, c \in \Gamma$. An elementary membrane can be divided into two membranes with the same label, possibly transforming one original object into a different one in each of the new membranes.

(g_0) $[\ a\overline{a} \rightarrow \lambda\ ]_h$ for $h \in H$, $a, \overline{a} \in O$. This is an annihilation rule, associated with a membrane labeled by h: the pair of objects $a, \overline{a} \in O$ belonging simultaneously to this membrane disappears.

Let us remark that dissolution rules - type (d_0) - and rules for non-elementary division - type (f_0) - are not considered in this model.

These rules are applied according to the following principles (with the special restrictions for annihilation rules specified above):

- All the rules are applied in parallel and in a maximal manner. In one step, one object of a membrane can be used by at most one rule (chosen in a non–deterministic way), and each membrane can be the subject of *at most one* rule of types (b_0), (c_0) and (e_0).
- If at the same time a membrane labeled with h is divided by a rule of type (e_0) triggered by some object a and there are other objects in this membrane to which rules of type (a_0) or (g_0) can be applied, then we suppose that first the rules of type (g_0) and only then those of type (a_0) are used, before finally the division is executed. Of course, this process in total takes only one step.
- The rules associated with membranes labeled by h are used for all copies of membranes with label h.

4 Recognizer P Systems

Recognizer P systems are a well-known model of P systems which are basic for the study of complexity aspects in Membrane Computing. Next, we briefly recall some basic ideas related to them. For a detailed description, for example, see [19, 20]. In recognizer P systems all computations halt; there are two distinguished objects traditionally called `yes` and `no` (used to signal the result of the computation), and exactly one of these objects is sent out to the environment (only) in the last computation step.

Let us recall that a decision problem X is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (the elements are called *instances*) and θ_X is a predicate

(a total Boolean function) over I_X. Let $X = (I_X, \theta_X)$ be a decision problem. A *polynomial encoding* of X is a pair (cod, s) of polynomial time computable functions over I_X such that for each instance $w \in I_X$, $s(w)$ is a natural number representing the *size* of the instance and $cod(w)$ is a multiset representing an encoding of the instance. Polynomial encodings are stable under polynomial time reductions.

Let $\mathcal{R}$ be a class of recognizer P systems with input membrane. A decision problem $X = (I_X, \theta_X)$ is solvable in a uniform way and polynomial time by a family $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ of P systems from $\mathcal{R}$ – we denote this by $X \in \mathbf{PMC}_{\mathcal{R}}$ – if the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, i.e., there exists a polynomial encoding (cod, s) from I_X to $\mathbf{\Pi}$ such that the family $\mathbf{\Pi}$ is polynomially bounded with regard to (X, cod, s); this means that there exists a polynomial function p such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps; the family $\mathbf{\Pi}$ is sound and complete with regard to (X, cod, s).

5 Solving SAT

Propositional Satisfiability is the problem of determining, for a formula of the propositional calculus, if there is an assignment of truth values to its variables for which that formula evaluates to true. By **SAT** we mean the problem of propositional satisfiability for formulas in conjunctive normal form (CNF). In this section we describe a family of P systems which solves it. As usual, we will address the resolution via a brute force algorithm, which consists of the following stages (some of the ideas for the design are taken from [6] and [21]):

- *Generation and Evaluation Stage:* All possible assignments associated with the formula are created and evaluated (in this paper we have subdivided this group into *Generation* and *Input processing* groups of rules, which take place in parallel).
- *Checking Stage:* In each membrane we check whether or not the formula evaluates to true for the assignment associated with it.
- *Output Stage:* The systems sends out the correct answer to the environment.

Let us consider the pair function $\langle\ ,\ \rangle$ defined by $\langle n, m\rangle = ((n + m)(n + m + 1)/2) + n$. This function is polynomial-time computable (it is primitive recursive and bijective from $\mathbb{N}^2$ onto $\mathbb{N}$). For any given formula in CNF, $\varphi = C_1 \wedge \cdots \wedge C_m$, with m clauses and n variables $Var(\varphi) = \{x_1, \ldots, x_n\}$ we construct a P system $\Pi(\langle n, m\rangle)$ solving it, where the multiset encoding of the problem to be the input of $\Pi(\langle n, m\rangle)$ (for the sake of simplicity, in the following we will omit m and n) is

$$cod(\varphi) = \{x_{i,j} :\ x_j \in C_i\} \cup \{y_{i,j} :\ \ \neg x_j \in C_i\}.$$

For solving SAT by a uniform family of deterministic recognizer P systems with active membranes, without polarizations, without non-elementary membrane

division and without dissolution, yet with matter/antimatter annihilation rules, we now construct the members of this family as follows:

$$\begin{aligned}
&\Pi = (O, \Sigma, H = \{0,1\}, \mu = [\ [\ ]_2\ ]_1, w_1, w_2, R, i_{in} = 2), \text{ where}\\
&\Sigma = \{x_{i,j}, y_{i,j} \mid 1 \le i \le m,\ 1 \le j \le n\},\\
&O = \{d, t, f, F, \overline{F}, T, \overline{no}_{n+5}, \overline{F}_{n+5}, \overline{yes}_{n+6}, yes_{n+6}, no_{n+6}, yes, no\}\\
&\quad \cup \{x_{i,j}, y_{i,j} \mid 1 \le i \le m,\ -1 \le j \le n\} \cup \{\overline{x}_{i,-1}, \overline{y}_{i,-1} \mid 1 \le i \le m\}\\
&\quad \cup \{c_i, \overline{c}_i \mid 1 \le i \le m\} \cup \{e_j \mid 1 \le j \le n+3\}\\
&\quad \cup \{yes_j, no_j, F_j \mid 1 \le j \le n+5\},\\
&w_1 = no_0\ yes_0\ F_0,\ w_2 = d^n\ e_1,
\end{aligned}$$

and the rules of set R are given below, presented in groups Generation, Input processing, Checking and Output, together with explanations how the rules in the groups work.

Generation

G1. $[\ d\ ]_2 \to [\ t\ ]_2 [\ f\ ]_2$;
G2. $[\ t \to \overline{y}_{1,-1} \cdots \overline{y}_{m,-1}\ ]_2$;
G3. $[\ f \to \overline{x}_{1,-1} \cdots \overline{x}_{m,-1}\ ]_2$;
G4. $[\ \overline{x}_{i,-1} \to \lambda\ ]_2$, $1 \le i \le m$;
G5. $[\ \overline{y}_{i,-1} \to \lambda\ ]_2$, $1 \le i \le m$.

In each step j, $1 \le j \le n$, every elementary membrane is divided, one child membrane corresponding with assigning *true* to variable j and the other one with assigning *false* to it. One step later, proper objects are produced to annihilate the input objects associated to variable j: in the *true* case, we introduce the antimatter object for the negated variable, i.e., it will annihilate the corresponding negated variable, and in the *false* case, we introduce the antimatter object for the variable itself, i.e., it will annihilate the corresponding variable. Remaining barred (antimatter) objects not having been annihilated with the input objects, are erased in the next step.

Input processing

I1. $[\ x_{i,j} \to x_{i,j-1}\ ]_2$, $1 \le i \le m$, $0 \le j \le n$;
I2. $[\ y_{i,j} \to y_{i,j-1}\ ]_2$, $1 \le i \le m$, $0 \le j \le n$;
I3. $[\ x_{i,-1}\ \overline{x}_{i,-1} \to \lambda\ ]_2$, $1 \le i \le m$;
I4. $[\ y_{i,-1}\ \overline{y}_{i,-1} \to \lambda\ ]_2$, $1 \le i \le m$;
I5. $[\ x_{i,-1} \to c_i\ ]_2$, $1 \le i \le m$;
I6. $[\ y_{i,-1} \to c_i\ ]_2$, $1 \le i \le m$.

Input objects associated with variable j decrement their second subscript during $j+1$ steps to -1. The variables not representing the desired truth value are eliminated by the corresponding antimatter object generated by the rules G2 and G3, whereas any of the input variables not annihilated then, shows that the associated clause i is satisfied, which situation is represented by the introduction of the object c_i.

Checking

C1. $[\ e_j \to e_{j+1}\ ]_2$, $1 \leq j \leq n+1$;
C2. $[\ e_{n+2} \to \overline{c}_1 \cdots \overline{c}_m e_{n+3}\ ]_2$;
C3. $[\ c_i\ \overline{c}_i \to \lambda\ ]_2$, $1 \leq i \leq m$;
C4. $[\ \overline{c}_i \to F\ ]_2$, $1 \leq i \leq m$;
C5. $[\ e_{n+3} \to \overline{F}\ ]_2$;
C6. $[\ F\ \overline{F} \to \lambda\ ]_2$, $1 \leq i \leq m$;
C7. $[\ \overline{F}\ ]_2 \to [\ \ ]_2 T$.

It took $n+2$ steps to produce objects c_i for every satisfied clause, possibly multiple times. Starting from object e_1, we have obtained the object e_{n+2} until then; from this object e_{n+2}, at step $n+2$ one anti-object is produced for each clause. Any of these clause anti-objects that is not annihilated, is transformed into F, showing that the chosen variable assignment did not satisfy the corresponding clause. It remains to notice that object T is sent to the skin (at step $n+4$) if and only if an object $\overline{F}$ did not get annihilated, i.e., no clause failed to be satisfied.

Output

O1. $[\ yes_j \to yes_{j+1}\ ]_1$, $1 \leq j \leq n+5$;
O2. $[\ no_j \to no_{j+1}\ ]_1$, $1 \leq j \leq n+5$;
O3. $[\ F_j \to F_{j+1}\ ]_1$, $1 \leq j \leq n+4$;
O4. $[\ T \to \overline{no}_{n+5} \overline{F}_{n+5}\ ]_1$;
O5. $[\ no_{n+5}\ \overline{no}_{n+5} \to \lambda\ ]_1$;
O6. $[\ no_{n+6}\ ]_1 \to [\ \ ]_1 no$;
O7. $[\ F_{n+5}\ \overline{F}_{n+5} \to \lambda\ ]_1$;
O8. $[\ F_{n+5} \to \overline{yes}_{n+6}\ ]_1$;
O9. $[\ yes_{n+6}\ \overline{yes}_{n+6} \to \lambda\ ]_1$;
O10. $[\ yes_{n+6}\ ]_1 \to [\ \ ]_1 yes$.

If no object T has been sent to the skin, then the initial *no*-object can count up to $n+6$ and then sends out the negative answer *no*, while the initial F-object counts up to $n+5$, generates the antimatter object for the *yes*-object at stage $n+6$ and annihilates with the corresponding *yes*-object at stage $n+6$. On the other hand, if (at least one) object T arrives in the skin, then the *no*-object is annihilated at stage $n+5$ before it would be sent out in the next step, and the F-object is annihilated before it could annihilate with the *yes*-object, so that the positive answer *yes* can be sent out in step $n+6$.

Finally, we notice that the solution is uniform, deterministic, and uses only rules of types (a_0), (c_0), (e_0) as well as matter/antimatter annihilation rules. The result is produced in $n+6$ steps.

6 Conclusions

Although the ability of the model for solving NP problems was proved in [7], to the best of our knowledge, this is the first solution to a strongly **NP** problem by using

annihilation rules in Membrane Computing. Let us remark the important role of the definition for recognizer P systems we have used in this paper. This definition is quite restrictive, since only one object *yes* or *no* is sent to the environment in any computation. In the literature one can find other definitions of recognizer P systems and therefore other definitions of what it means *to solve* a problem in the framework of Membrane Computing. The study of the complexity classes in Membrane Computing deserves a deep revision under these new definitions.

Acknowledgements

M.A. Gutiérrez-Naranjo acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

References

1. Artiom Alhazov, Bogdan Aman, and Rudolf Freund. P systems with anti-matter. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, Petr Sosík, and Claudio Zandron, editors, *Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, August 20-22, 2014, Revised Selected Papers*, volume 8961 of *Lecture Notes in Computer Science*, pages 66–85, Springer, 2014.
2. Artiom Alhazov, Bogdan Aman, Rudolf Freund, and Gheorghe Păun. Matter and anti-matter in membrane systems. In *DCFS 2014*, volume 8614 of *Lecture Notes in Computer Science*, pages 65–76, Springer, 2014.
3. Artiom Alhazov, Linqiang Pan, and Gheorghe Păun. Trading polarizations for labels in P systems with active membranes. *Acta Informatica*, 41(2-3): 111–144, 2004.
4. Artiom Alhazov and Mario J. Pérez-Jiménez. Uniform solution of QSAT using polarizationless active membranes. In *MCU 2007*, volume 4664 of *Lecture Notes in Computer Science*, pages 122–133, Springer, 2007.
5. Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, ACM, New York, NY, USA, 1971.
6. Andrés Cordón-Franco, Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez, and Fernando Sancho-Caparrini. A Prolog simulator for deterministic P systems with active membranes. *New Generation Computing*, 22(4): 349–363, 2004.
7. Daniel Díaz-Pernil, Francisco Peña-Cantillana, Artiom Alhazov, Rudolf Freund, and Miguel A. Gutiérrez-Naranjo. Antimatter as a frontier of tractability in membrane computing. *Fundamenta Informaticae*, 134: 83–96, 2014.
8. Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.
9. Zsolt Gazdag and Gábor Kolonits. A new approach for solving SAT by P systems with active membranes. In Erzsébet Csuhaj-Varjú, Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, and György Vaszil, editors, *International Conference on Membrane Computing*, volume 7762 of *Lecture Notes in Computer Science*, pages 195–207, Springer, 2012.

10. Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez, Agustin Riscos-Núñez, and Francisco José Romero-Campero. On the power of dissolution in P systems with active membranes. In Rudolf Freund, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Workshop on Membrane Computing*, volume 3850 of *Lecture Notes in Computer Science*, pages 224–240, Springer, 2005.
11. Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez, and Francisco José Romero-Campero. A uniform solution to SAT using membrane creation. *Theoretical Computer Science*, 371(1-2): 54–61, 2007.
12. Tseren-Onolt Ishdorj and Alberto Leporati. Uniform solutions to SAT and 3-SAT by spiking neural P systems with pre-computed resources. *Natural Computing*, 7(4): 519–534, 2008.
13. Alberto Leporati, Luca Manzoni, Giancarlo Mauri, Antonio E. Porreca, Claudio Zandron. Simulating elementary active membranes - with an application to the P conjecture. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, Petr Sosík, Claudio Zandron, editors, *Conference on Membrane Computing*, volume 8961 of *Lecture Notes in Computer Science*, pages 284–299, Springer, 2014.
14. Alberto Leporati, Giancarlo Mauri, Claudio Zandron, Gheorghe Păun, and Mario J. Pérez-Jiménez. Uniform solutions to SAT and subset sum by spiking neural P systems. *Natural Computing*, 8(4): 681–702, 2009.
15. Venkata Padmavati Metta, Kamala Krithivasan, and Deepak Garg. Computability of spiking neural P systems with anti-spikes. *New Mathematics and Natural Computation (NMNC)*, 08(03): 283–295, 2012.
16. Adam Obtulowicz. Deterministic P-systems for solving SAT-problem. *Romanian Journal of Information Science and Technology*, 4(1-2): 195–201, 2001.
17. Linqiang Pan and Artiom Alhazov. Solving HPP and SAT by P systems with active membranes and separation rules. *Acta Informatica*, 43(2): 131–145, 2006.
18. Linqiang Pan and Gheorghe Păun. Spiking neural P systems with anti-spikes. *International Journal of Computers, Communications & Control*, IV(3): 273–282, September 2009.
19. Mario J. Pérez-Jiménez. An approach to computational complexity in membrane computing. In Giancarlo Mauri, Gheorghe Păun, Mario J. Pérez-Jiménez, Grzegorz Rozenberg, and Arto Salomaa, editors, *Workshop on Membrane Computing*, volume 3365 of *Lecture Notes in Computer Science*, pages 85–109, Springer, 2004.
20. Mario J. Pérez-Jiménez, Agustin Riscos-Núñez, Álvaro Romero-Jiménez, and Damien Woods. Complexity - membrane division, membrane creation. In Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *The Oxford Handbook of Membrane Computing*, pages 302 – 336. Oxford University Press, Oxford, England, 2010.
21. Mario J. Pérez-Jiménez, Álvaro Romero-Jiménez, and Fernando Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3): 265–285, 2003.
22. Tao Song, Yun Jiang, Xiaolong Shi, and Xiangxiang Zeng. Small universal spiking neural P systems with anti-spikes. *Journal of Computational and Theoretical Nanoscience*, 10(4): 999–1006, 2013.
23. Gangjun Tan, Tao Song, Zhihua Chen, and Xiangxiang Zeng. Spiking neural P systems with anti-spikes and without annihilating priority working in a 'flip-flop' way. *International Journal of Computing Science and Mathematics*, 4(2): 152–162, July 2013.