Solving the N-Queens Puzzle with P Systems

Miguel A. Gutiérrez-Naranjo, Miguel A. Martínez-del-Amor,
Ignacio Pérez-Hurtado, Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
magutier@us.es, mdelamor@us.es, perezh@us.es, marper@us.es

Summary. The N-queens puzzle consists on placing N queens on an $N \times N$grid in such way that no two queens are on the same row, column or diagonal line. In this paper we present a family of P systems with active membranes (one P system for each value of N) that provides all the possible solutions to the puzzle.

1 Introduction

The N-queens puzzle is very popular among computer scientists. It is a generalization of a classic puzzle known as the 8-queens puzzle. The original one is attributed to the chess player Max Bezzel and it consists on putting eight queens on an 8×8 chessboard in such way that none of them is able to capture any other using the standard movement of the queens in chess, i.e., only one queen can be placed on each row, column and diagonal line.

The 8-queens puzzle was later generalized to the N-queens puzzle, with the same rules but placing N queens on a $N \times N$ board. The problem is computationally very expensive, since there exists $64!/(56! \times 8!) \sim 4.4 \times 10^9$ possible arrangements of 8 queens in a 8×8 chessboard and there are only 92 solutions. If two solutions are considered the same when one of them can be obtained from the other one via a rotation or a symmetry, then there are only 12 different solutions.

For this reason, the brute force algorithm is not useful with current computers. In fact, this simple puzzle is usually presented in Computer Science as an standard of use of heuristics which allows us discard options and deal with a little number of candidate solutions.

In this paper, we present a first solution to the N-queens puzzle in Membrane Computing. For that purpose, we propose a family of deterministic P systems with active membranes (the N-th element of the family solves the N-queens puzzle) such that the halting configuration encodes *all* the solutions of the puzzle. As usual, we use the massive parallelism to check all the feasible solutions at the same time and obtain the solution in a reduced number of steps.

The paper is organized as follows: In Section 2 we show how an instance of the N-queens puzzle can be expressed as a formula in conjunctive normal form. In Section 3, we briefly recall the P systems with active membranes and in Section 4, we present our family of P systems that solve SAT. The difference with other solutions is that in this case the P system does not only send `Yes` or `No` to the environment by showing the existence or not of a solution to the problem, but it keeps all the truth values that satisfy the formula. In section 5 we build the family of P systems that solve the N-queens problem by choosing the appropriate P systems from the previous family. Section 6 shows several experimental results obtained by running these solutions in an updated version of the P-lingua [1] simulator. Finally, some conclusions and new open research lines are presented.

2 Changing the Representation

The key idea of our solution is that an instance of the N-queens puzzle for a fixed N can be represented as a formula in conjunctive normal form (CNF) in such way that one truth assignment of the formula can be considered as a solution of the puzzle. In Section 5 we will show a family of P systems with active membranes associated to the N-queens puzzle which encodes the truth assignments of the associate formula in the halting configuration.

The N-queens puzzle can be represented by a formula in CNF with N^2 propositional variables s_{ij}, where s_{ij} stands for the cell (i, j) of the $N \times N$ chessboard. The variable s_{ij} is assigned true if and only if a queen is assigned to the cell (i, j). The different constraints of the puzzle can be expressed with this representation in the following way:

- There is at most one queen in each column.
  $$\psi_1 \equiv \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ij} \vee \neg s_{ik})$$
- There is at most one queen in each row.
  $$\psi_2 \equiv \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ji} \vee \neg s_{ki})$$

Next we deal with the restriction of diagonal lines. Let us call D_1 the set of diagonal lines parallel to the bottom-left to up-right diagonal and D_2 the set of diagonal lines parallel to the bottom-right to up-left diagonal. It is easy to see that any line of D_1 is characterized by a number from $\{-(n-1), -(n-2), \ldots, -1, 0, 1, \ldots, n-2, n-1\}$ which represents the difference $i-j$ in the cell (i, j). In order to fix ideas, let us consider an 8×8 and the diagonal line $\langle (1,3), (2,4), (3,5), (4,6), (5,7), (6,8) \rangle$. Cells (i, j) in this diagonal line are characterized by number -2, since in all of them $i - j = -2$.

On the other hand, any line of D_2 is characterized by a number from $\{2, 3, \ldots, 2n-1, 2n\}$ which represents the sum $i + j$. For example, in a 8×8

chessboard diagonal line $\langle (5,8),(6,7),(7,6),(8,5)\rangle$ can be characterized by number 13 since in all of them $i+j=13$.

Firstly, we consider the diagonal lines of D_1 corresponding to the bottom semi-square. Each of these lines is characterized by a number d in $\{0,\ldots,n-2\}$, and each line is compounded by the cells (i,j) such that $i-j=d$. Notice that $d=n-1$ is not considered, since such a diagonal line has only one cell. The formula that codifies that there must occur one queen at most in these lines is

$$\psi_3 \equiv \bigwedge_{d=0}^{n-2} \bigwedge_{j=1}^{n-d} \bigwedge_{k=j+1}^{n-2} (\neg s_{d+j\,j} \vee \neg s_{d+k\,k})$$

The remaining diagonal lines from D_1 correspond to the values d from the set $\{-(n-2),\ldots,-1\}$ and they are codified by the formula

$$\psi_4 \equiv \bigwedge_{d=-(n-2)}^{-1} \bigwedge_{j=1}^{n+d} \bigwedge_{k=j+1}^{n+d} (\neg s_{j\,j-d} \vee \neg s_{k\,k-d})$$

We also split the set D_2 into two subsets. The first of them corresponds to the lines associated with numbers d in $\{3,\ldots,n+1\}$ which represents the bottom semi-square. Notice that the line with only one cell (the corner) is removed. The formula that codifies that there must appear one queen at most in these lines is

$$\psi_5 \equiv \bigwedge_{d=3}^{n+1} \bigwedge_{j=1}^{d-1} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\,d-j} \vee \neg s_{k\,d-k})$$

Analogously, the upper semi-square is associated with numbers d in $\{n+2,\ldots,2n-1\}$. The formula associated to these lines is

$$\psi_6 \equiv \bigwedge_{d=n+2}^{2n-1} \bigwedge_{j=d-n}^{n} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\,d-j} \vee \neg s_{k\,d-k})$$

The conjunction of the previous formula says that in each column, row and diagonal line, there must be at most one queen. These conditions are satisfied by the empty board or by a board with only one queen. In order to fulfill the conditions of the N-queens puzzle we need to impose N queens to be placed. Since ψ_1 encodes that *There is at most one queen in each column*, it suffices to add the restriction *There is at least one queen in each column* in order to get that *There is exactly one queen in each column*. Since there are N columns, this leads us to place exactly N queens.

- There is at least one queen in each column.

  $$\psi_7 \equiv \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} s_{ij}$$

The conjunction of these seven formulae

$$\Phi \equiv \psi_1 \bigwedge \psi_2 \bigwedge \psi_3 \bigwedge \psi_4 \bigwedge \psi_5 \bigwedge \psi_6 \bigwedge \psi_7$$

is a formula in conjunctive normal form and each truth assignment which makes it true represents a solution to the N-queens puzzle.

3 The P System Model

P systems with active membranes is one of the most studied models on Membrane Computing and it is very well-known by the P system community. It is one of the first models presented by Gh. Păun in [4]. Here we provide a brief recall of its features.

A *P system with active membranes* is a construct:

$$(V, H, \mu, w_1, \ldots, w_m, R)$$

where:

1. $m \geq 1$, is the initial *degree* of the system;
2. V is the alphabet of *symbol-objects*;
3. H is a finite set of *labels* for membranes;
4. μ is a *membrane structure*, of m membranes, bijectively labeled with elements of H;
5. $w_1, \ldots, w_m$ are strings over V, describing the initial *multisets* of objects placed in the m regions of μ;
6. R is a finite set of *evolution rules*, of the following forms:
   a) $[\, x \rightarrow y \,]_h^{\alpha}$, for $h \in H$, $\alpha \in \{+, -, 0\}$, $x \in V$, $y \in V^*$. This is an object evolution rule, associated with a membrane labeled with h and depending on the polarity of that membrane. The empty string is represented by $\lambda \in V^*$.
   b) $x\, [\,]_h^{\alpha_1} \rightarrow [\, y \,]_h^{\alpha_2}$, for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $x, y \in V$. An object from the region immediately outside a membrane labeled with h is introduced in this membrane, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.
   c) $[\, x \,]_h^{\alpha_1} \rightarrow y\, [\,]_h^{\alpha_2}$, for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $x, y \in V$. An object is sent out from membrane labeled with h to the region immediately outside, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.
   d) $[\, x \,]_h^{\alpha} \rightarrow y$, for $h \in H$, $\alpha \in \{+, -, 0\}$, $x, y \in V$. A membrane labeled with h is dissolved in reaction with an object. The skin is never dissolved.
   e) $[\, x \,]_h^{\alpha_1} \rightarrow [\, y \,]_h^{\alpha_2} [\, z \,]_h^{\alpha_3}$, for $h \in H$, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $x, y, z \in V$. A membrane can be divided into two membranes with the same label, possibly transforming some objects and their polarities.

These rules are applied according to the following principles:

- All the rules are applied in parallel and in a maximal manner. At one step, one object of a membrane can be used by only one rule (chosen in a non deterministic way), but any object being able to evolve by one rule of any form, should evolve.
- If a membrane is dissolved, its content (multiset and internal membranes) is left free in the surrounding region.

- All objects and membranes not specified in a rule and which do not evolve remain unchanged to the next step.
- At the same time, if a membrane h is divided by a rule of type (e) or dissolved by a rule of type (d) and there are objects in this membrane which evolve by means of rules of type (a), then we suppose that evolution rules of type (a) are used first and then, the division is produced. This process takes of course only one step.
- The rules associated with membranes labeled with h are used for all copies of this membrane. At one step, a membrane labeled with h can be the subject of *only* one rule of types (b)-(e).

4 A New Solution for the SAT Problem

Propositional Satisfiability is the problem to determine, 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.

According to Section 2, in order to solve the N-queens puzzle we need to find a truth assignment such that it makes true a formula in CNF. This problem is exactly SAT. In [3], we can find a uniform solution to the problem SAT. This design takes SAT as a decision problem and each P system of the family sends a `Yes` or `No` answer to the environment at the last step of computation specifying whether the solution exists or not. In this paper, we are not interested in the existence or not of a solution for the N-queens puzzle, but finding (and storing) the truth assignment in an effective way.

In this section we present a uniform family of deterministic recognizer P systems[1] which solves SAT as a decision problem (i.e., the P system sends a `Yes` or `No` answer to the environment at the last computation step) but it also stores truth assignments that make the formula true. We can find all the solutions to the N-queens puzzle encoded in the elementary membranes of the halting configuration.

Let us suppose that $\varphi = C_1 \wedge \cdots \wedge C_m$ in a formula in CNF, and $Var(\varphi) = \{x_1, \ldots, x_n\}$ is the set of variables in φ. Formula φ will be provided to the P systems as the following *input multiset*

$$cod(\varphi) = \{x_{ji} \,|\, x_i \in C_j\} \cup \{y_{ji} \,|\, \neg\, x_i \in C_j\}$$

Such initial multiset will be placed in the membrane with *input label* at the initial configuration. For each $(m, n) \in \mathbb{N}^2$ we consider the recognizer P system

$$(\Pi(\langle m, n\rangle), \Sigma(m, n), i(m, n))$$

where the *input alphabet* is

[1] A detailed description of recognizer P systems can be found in [3].

$$\Sigma(m,n) = \{x_{ij}, y_{ij} : \ 1 \leq i \leq m, \ 1 \leq j \leq n\}$$

the *input label* is $i(m,n) = 2$ and the P system

$$\Pi(\langle n,m \rangle) = (\Gamma(m,n), \{1,2\}, [\ [\ ]_2\ ]_1, w_1, w_2, R)$$

is defined as follows:
$\Gamma(m,n) = \Sigma(m,n) \ \cup \{d_k : 1 \leq k \leq 43n + 2m + 1\} \cup \{s_k : 1 \leq k \leq n\} \cup$
$\{t_j, f_j : 1 \leq j \leq n\} \cup \{z_{ij}, h_{ij} : 1 \leq i \leq m \, 2 \leq j \leq n,\} \cup$
$\{r_{ij}, \ 1 \leq i \leq m, \, 1 \leq j \leq 2n,\} \cup \{c_k : 1 \leq k \leq m+1\} \cup$
$\{v_k : 1 \leq k \leq 6n + 2m - 1\} \cup \{e, r, \texttt{Yes}, \texttt{No}\}$
The initial content of each membrane is $w_1 = \emptyset$ y $w_2 = \{d_0, v_0\}$. As usual, the initial polarization is 0. The set of rules, R, is given by:

$(a.1)$ $[\, d_j \,]_2^0 \rightarrow [\, s_{j+1} \,]_2^+ \, [\, s_{j+1} \,]_2^-$ for all $j \in \{0, \ldots, n-1\}$.
$(a.2)$ $[\, d_j \,]_2^+ \rightarrow d_j \, [\,]_2^0 \quad [\, d_j \,]_2^- \rightarrow d_j \, [\,]_2^0$ for all $j \in \{1, \ldots, n\}$.
$(a.3)$ $d_j \, [\,]_2^0 \rightarrow [\, d_j \,]_2^0$ for all $j \in \{1, \ldots, n-1\}$.
$(a.4)$ $[\, d_i \rightarrow d_{i+1}]_1^0$ for all $i \in \{n, \ldots, 3n-4\} \cup \{3n-2, \ldots, 3n+2m\}$.
$(a.5)$ $[\, d_{3n-3} \rightarrow d_{3n-2} e]_1^0$.
$(a.6)$ $[\, d_{3n+2m+1} \,]_1^0 \rightarrow \texttt{No} \, [\,]_1^+$.

By using these rules, a membrane with label 2 is divided into two membranes with the same label, but with different polarizations. These rules allow us to duplicate, in one step, the total number of internal membranes. When object d_n is reached, the counter changes its function. From d_n to $d_{4n+2m-3}$ the sequence of objects d_i is just a counter. If object $d_{4n+2m-3}$ is reached in the membrane labeled with 1 with polarization 0, then the answer `No` is sent to the environment.

(b) $\quad [\, s_j \rightarrow t_j d_j \,]_2^+ \quad [\, s_j \rightarrow f_j d_j \,]_2^-$ for all $j \in \{1, \ldots, n\}$.

Instead of producing an object d_{j+1}, objects d_j $(0 \leq j \leq n-1)$ produce an object s_{j+1} (see the set $(a.1)$). With this new set of rules (b) we get such d_j from s_j. We also obtain markers t_j (true) and f_j (false) depending on the polarization of the membranes.

$(c.1)$ $\left\{ \begin{array}{ll} [\, x_{i1} \rightarrow r_{i1} \,]_2^+ & [\, y_{i1} \rightarrow \lambda \,]_2^+ \\ [\, x_{i1} \rightarrow \lambda \,]_2^- & [\, y_{i1} \rightarrow r_{i1} \,]_2^- \end{array} \right\}$ for all $i \in \{1, \ldots, m\}$.

$(c.2)$ $\left\{ \begin{array}{ll} [\, x_{ij} \rightarrow z_{ij} \,]_2^+ & [\, y_{ij} \rightarrow h_{ij} \,]_2^+ \\ [\, x_{ij} \rightarrow z_{ij} \,]_2^- & [\, y_{ij} \rightarrow h_{ij} \,]_2^- \end{array} \right\}$ for all $i \in \{1, \ldots, m\}$ and $j \in \{2, \ldots, n\}$.

The rules of $(c.1)$ implement a process allowing internal membranes to encode the *assignment* of a variable and, simultaneously, to check the value of all clauses by this assignment, in such a way that if the clause is true then an object $r_{i,1}$ will appear in the membrane. In other cases, the object encoding the variable will disappear. Rules from set $(c.2)$ perform a technical renaming.

$$(d)\ \left\{ \begin{array}{ll} [\, z_{ij} \rightarrow x_{ij-1} \,]_2^+ & [\, h_{ij} \rightarrow y_{ij-1} \,]_2^+ \\ [\, z_{ij} \rightarrow x_{ij-1} \,]_2^- & [\, h_{ij} \rightarrow y_{ij-1} \,]_2^- \end{array} \right\} \text{ for all } i \in \{1, \ldots, m\} \text{ and } j \in \{2, \ldots, n\}.$$

The checking process previously described is always carried out with respect to the *first* variable appearing in the internal membrane. Hence, the rules of (d) take charge of making a cyclic path through all the variables to get that, initially, the first variable is x_1, then x_2, and so on.

$(e.1)$ $[\, r_{ij} \rightarrow r_{ij+1} \,]_2^0$ for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, 2n-1\}$.
$(e.2)$ $[\, r_{1\,2n} \,]_2^+ \rightarrow r_{1\,2n}[\,]_2^-$.
$(e.3)$ $[\, r_{1\,2n} \rightarrow \lambda \,]_2^-$.
$(e.4)$ $[\, r_{j\,2n} \rightarrow r_{j-1\,2n} \,]_2^-$ for all $j \in \{2, \ldots, m\}$.
$(e.5)$ $r_{1\,2n}\,[\,]_2^- \rightarrow [\, r \,]_2^+$.

In objects r_{jk}, index j represents a clause. Index i evolves in all the membranes until reaching $r_{j\,2n}$ for each $j \in \{1, \ldots, m\}$. These objects $r_{j\,2n}$ play their role at the checking stage.

$(f)\, e\,[\,]_2^0 \rightarrow [\, c_1 \,]_2^+$.

Objects e are created in membrane 1 by objects d_{3n-3}. They send objects c_1 into the elementary membranes and start the checking stage.

$(g.1)$ $[\, v_i \rightarrow v_{i+1} \,]_2^0 \quad [\, v_i \rightarrow v_{i+1} \,]_2^+ \quad [\, v_i \rightarrow v_{i+1} \,]_2^-$, for all $i \in \{0, \ldots, 6n+2m-2\}$.
$(g.2)$ $[\, v_{6n+2m-1} \rightarrow \lambda]_2^-$.
$(g.3)$ $[\, v_{6n+2m-1}]_2^+ \rightarrow r$.

The sequence of objects v_i is merely a counter. If object $v_{6n+2m-1}$ appears in an elementary membrane with negative polarization, it just disappears. Otherwise, if the polarization is positive, it dissolves the membrane. The importance of this counter is crucial since all the membranes which do not encode a solution are dissolved.

$(h.1)$ $[\, c_j \rightarrow c_{j+1} \,]_2^-$ for all $j \in \{1, \ldots, m\}$.
$(h.2)$ $[\, c_{m+1} \,]_2^+ \rightarrow c_{m+1}\,[\,]_2^-$.
$(h.3)$ $[\, c_{m+1} \,]_1^0 \rightarrow \texttt{Yes}\,[\,]_1^+$.

Evolution from c_1 to c_{m+1} is completed only in the elementary membranes that represents truth assignments that make the whole formula true. If an object c_{m+1} reaches the skin, then it sends out an object `Yes`.

(i) $[\, r \rightarrow \lambda]_2^+$.

Just a cleaning rule.

4.1 Some notes on the computation

All the P systems of the family are deterministic. The first stage of the computation finishes with configuration C_{4n-1}. In such configuration, there are 2^n elementary membranes, one for each possible truth assignment of the set of variables $\{x_1, \ldots, x_n\}$. We also have 2^n copies of the object d_n in the membrane labeled by 1. The checking stage starts with the configuration C_{6n-2}. An object c_1 appears in every elementary membrane in such configuration. If the answer is `Yes`, then the halting configuration is C_{6n+2m}, otherwise, if the answer is `No`, then the halting configuration is $C_{6n+2m+1}$.

If the answer is `No`, all the elementary membranes have been dissolved and the unique membrane in the halting configuration is the skin. If the answer is `Yes`, at least one elementary membrane has not been dissolved. Each elementary membrane in the halting configuration represents a truth assignment that makes the formula true. The encoding is quite easy: for each $i \in \{1, \ldots, n\}$, either object t_i or object f_i belongs to the elementary membrane. Objects t_i means that in this assignment, variable x_i takes the value *true*, and f_i means that such variable is false.

It is easy to check that if we have a formula with n variables and m clauses we need $10mn + 26n + 5m + 6$ rules.

5 A Family of P Systems

In Section 2, we have seen that an instance of the N-queens puzzle can be represented as a formula in CNF and in Section 4 we have shown that there exists one P system with active membranes which is able to solve any instance of the problem SAT with m clauses and n variables.

In this section, we will select one P system of the family for each N. Since there exist one P system associated to each pair (m, n) where m is the number of clauses and n the number of variables, it only remains to know how many variables and how many clauses there are in the CNF formula associated to each instance on the N-queens puzzle. Both amounts are fixed by the following theorem.

Theorem 1. *Given an integer $N \geq 3$, the formula Φ in conjunctive normal form that encodes the N-queens puzzle according to the previous description has N^2 variables and $\frac{1}{3}(5N^3 - 6N^2 + 4N)$ clauses.*

Proof. It is trivial to check that the number of variables is N^2, since each variable represents a cell in a $N \times N$ chessboard. In order to obtain the number of clauses, we will sum the number of clauses in clauses $\psi_1, \ldots, \psi_7$, since the global formula Φ is the conjunction of these seven formulae in CNF.

- Clause 1: $\psi_1 \equiv \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ij} \vee \neg s_{ik})$

For each column $i \in \{1, \ldots, n\}$ we compare the cells pairwise, so, in the formula $\wedge_{j=1}^{n} \wedge_{k=j+1}^{n} (\neg s_{ij} \vee \neg s_{ik})$ there are $1 + 2 + \cdots + N - 1 = \frac{N(N-1)}{2}$ clauses and in ψ_1 the number of clauses is

$$M_1 = \frac{N^2(N-1)}{2}$$

- Clause 2: $\psi_2 \equiv \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ji} \vee \neg s_{ki})$

The situation is symmetric to the previous one, but considering rows instead of columns, so the number of clauses is the same

$$M_2 = \frac{N^2(N-1)}{2}$$

- Clause 3: $\psi_3 \equiv \bigwedge_{d=0}^{n-2} \bigwedge_{j=1}^{n-d} \bigwedge_{k=j+1}^{n-2} (\neg s_{d+j\,j} \vee \neg s_{d+k\,k})$

In this set of diagonal lines, the first one corresponds to $d = 0$. In this line there are n cells. The number of pairwise comparisons between cells of this line is $S_{n-1} = 1 + 2 + \cdots + n - 1$. The following line corresponds to $d = 1$. In this line there are $n - 1$ cells and the number of pairwise comparisons is $S_{n-2} = 1 + 2 + \cdots + n - 2$. The whole number of comparisons is the sum $M = S_1 + S_2 + \cdots + S_{n-1}$ where

$$S_k = 1 + \cdots + k = \frac{k(k+1)}{2} = \frac{1}{2}k^2 + \frac{1}{2}k \quad \text{for all } k \in \{1, \ldots, n-1\}$$

Bearing in mind that

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

we have that

$$M_3 = \sum_{k=1}^{n-1} S_k = \frac{1}{2}\left(\sum_{k=1}^{n-1} k^2 + \sum_{k=1}^{n-1} k\right) = \frac{1}{2}\left(\frac{(n-1)\,n\,(2n-1)}{6} + \frac{n(n-1)}{2}\right)$$

$$= \frac{1}{6}\,n\,(n+1)\,(n-1)$$

- Clause 4: $\psi_4 \equiv \bigwedge_{d=-(n-2)}^{-1} \bigwedge_{j=1}^{n+d} \bigwedge_{k=j+1}^{n+d} (\neg s_{j\,j-d} \vee \neg s_{k\,k-d})$

The reasoning for ψ_3 is also valid in this case. The difference is that we sum from S_1 to S_{n-2}, so the number of clauses in this case is

$$M_4 = \frac{1}{6}\,n\,(n-1)\,(n-2)$$

- Clause 5: $\psi_5 \equiv \bigwedge_{d=3}^{n+1} \bigwedge_{j=1}^{d-1} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\,d-j} \vee \neg s_{k\,d-k})$
- Clause 6: $\psi_6 \equiv \bigwedge_{d=n+2}^{2n-1} \bigwedge_{j=d-n}^{n} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\,d-j} \vee \neg s_{k\,d-k})$

  Due to the symmetry, the number of clauses in these formulae are the same than in ψ_3 and ψ_4, which are

$$M_5 = \frac{1}{6}\, n\,(n+1)\,(n-1) \quad \text{and} \quad M_6 = \frac{1}{6}\, n\,(n-1)\,(n-2)$$

- Clause 7: $\psi_7 \equiv \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} s_{ij}$

  Trivially ψ_7 has n clauses, $M_7 = n$.

Finally, a simple calculus show that the whole number of clauses is

$$M_1 + M_2 + \cdots + M_7 = \frac{1}{3}(5n^3 - 6n^2 + 4n)$$

From the previous theorem we have the set of all solutions of the N-queens puzzle are encoded in the elementary membranes of the halting configuration of the P system

$$\Pi(\langle \frac{1}{3}(5N^3 - 6N^2 + 4N), N^2 \rangle)$$

with input membrane $i(\langle \frac{1}{3}(5N^3 - 6N^2 + 4N), N^2 \rangle) = 2$ and input the appropriate multiset on $\Sigma(\langle \frac{1}{3}(5N^3 - 6N^2 + 4N), N^2 \rangle) = 2$ encoding the formula Φ.

6 Experimental Results

In this section, we show a couple of experimental results obtained by running the corresponding P systems with an updated version of the P-lingua simulator [1]. The experiments were performed on a one-processor Intel core2 Quad (with 4 cores at 2,83Ghz), 8GB of RAM and using a C++ simulator over the operating system Ubuntu Server 8.04.

The 3-queens puzzle. In this case the problem consists on putting three queens on a 3×3 chessboard. According to our representation, the puzzle can be expressed by a formula in CNF with 9 variables and 31 clauses. This means that we can use the P system $\Pi(\langle 31, 9 \rangle)$ from the family that solves the SAT problem to obtain the solution. The *input multiset* has 65 elements and the P system has 3185 rules.

Along the computation, $2^9 = 512$ elementary membranes need to be considered in parallel. Since the simulation is carried out in a one-processor computer, in the simulation, these membranes are evaluated sequentially. It takes 7 seconds to reach the halting configuration. It is the 117-th configuration and in this configuration

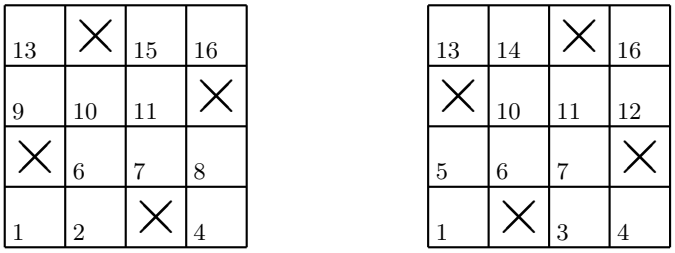

Fig. 1. Solutions to the 4-queens puzzle

one object `No` appears in the environment. As expected, this means that we cannot place three queens on a 3×3 chessboard satisfying the restriction of puzzle.

The 4-queens puzzle. In this case, we try to place four queens on a 4×4 chessboard. According to our representation, the puzzle can be expressed by a formula in CNF with 16 variables and 80 clauses. This means that we can use the P system $\Pi(\langle 80, 16\rangle)$ from the family that solves the SAT problem to obtain the solution. The *input multiset* has 168 elements.

Along the computation, $2^{16} = 65536$ elementary membranes need to be considered in parallel and the P system has 13622 rules.

The simulation takes 20583 seconds ($>$ 5 hours) to reach the halting configuration. It is the 256-th configuration and in this configuration one object `Yes` appears in the environment. This means that there exists at least one solution to the problem. In order to know such solutions, we check the multiset of the elementary membranes. In this case there are two elementary membranes in the halting configuration with the following multisets:

$$w_1 = \{f_1, f_2, t_3, f_4, t_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, t_{12}, f_{13}, t_{14}, f_{15}, f_{16}\}$$

$$w_2 = \{f_1, t_2, f_3, f_4, f_5, f_6, f_7, t_8, t_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, t_{15}, f_{16}\}$$

Such multisets encode the solution showed in the Figure 1

7 Conclusions and Future Work

In this paper we have presented a first solution to the N-queens puzzle based on Membrane Computing. The necessary resources and the number of computational steps for obtaining all the solutions of the puzzle are polynomial in N. Nonetheless, the simulation in one-processor computer needs an exponential amount of time.

Looking for solutions to toy puzzles as this one is not viable in the current conditions. This leads to us to three reflections: The first one is the necessity of giving the first steps for a wet implementation of P systems. The second aim, in the short-term, is to explore the possibilities of the most recent hardware, able to implement in parallel a big amount of simple rules as a realistic implementation

of P systems [2]. A third research line is to follow the same path than other computation models: To avoid brute force algorithms and start to go deeply in the study of heuristics in the design of cellular solutions.

Acknowledgements

The authors acknowledge the support of the project TIN2006-13425 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

1. D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos–Núñez: A P-Lingua Programming Environment for Membrane Computing. LNCS 5391, Springer, 2009, 187–203.
2. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, J.M. Cecilia, G. Guerrero, J.M. García: Simulation of Recognizer P Systems by Using Manycore GPUs. In these proceedings.
3. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrinini: A polynomial complexity class in P systems using membrane division. In E. Csuhaj-Varjú, C. Kintala, D. Wotschke, G. Vaszil, eds., *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems*, DCFS 2003, Computer and Automaton Research Institute of the Hungarian Academy of Sciences, 2003, 284–294.
4. Gh. Păun: *Membrane Computing. An Introduction.* Springer-Verlag, Berlín, 2002.