STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 1, 2011

A P SYSTEM DESIGN USING CLONAL SELECTION ALGORITHM

EMAD NABIL[1], AMR BADR[2], AND IBRAHIM FARAG[2]

Abstract. Membrane Computing is an emergent and promising branch of Natural Computing. Designing P systems is heavy constitutes a difficult problem. The candidate has often had an idea about the problem solution form. On the other hand, finding the exact and precise configurations and rules is a hard task, especially if there is no tool used to help in the designing process. The clonal selection algorithm, which is inspired from the vertebrate immune system, is introduced here to help in designing a P system that performs a specific task. This paper illustrates the use of the clonal selection algorithm with adaptive mutation in P systems design and compares it with genetic algorithms previously used to achieve the same purpose. Experimental results show that clonal selection algorithm surpasses genetic algorithms with a great difference.

1. Introduction

Artificial intelligence can be seen as a combination of several research disciplines such as computer science, physiology, philosophy, sociology, biology, physics and chemistry. Enormous successes have been achieved through modeling of biological and natural intelligence [4] resulting in what is called natural computing.

The natural computing can be classified into the three following branches.

- Bio-inspired approaches
- Artificial Life
- Computing With Natural Means

The above branches - depicted in figure 1 with their fields - together with logic, deductive reasoning, expert systems, case-based reasoning and symbolic

Received by the editors: March 20, 2011.

2010 *Mathematics Subject Classification.* 68T20, 92D25, 92F05.

1998 *CR Categories and Descriptors.* I.2.8 [**Computing Methodologies**]: artificial intelligence – *Problem Solving, Control Methods, and Search*; I.1.2 [**Computing Methodologies**]: Symbolic and algebraic manipulation – *Algorithms*.

Key words and phrases. membrane computing, P systems, artificial immune system, clonal selection algorithm.

machine learning systems form a big part of Artificial Intelligence (AI). A Brief preview of each category is described below.

Bio-inspired approaches are inspired from nature for the development of novel problem-solving techniques. Bio-inspired approaches include the following fields: Artificial Neural Networks inspired by the functioning of mammalian brain [22, 36], Evolutionary Algorithms motivated by evolutionary biology [21, 34], Simulated Annealing which borrows ideas from the annealing of metals and glasses [7, 30], Swarm Intelligence which is based on the collective behavior of social organisms[8, 20], Artificial Immune Systems inspired by the vertebrate immune system [6, 9, 23, 24, 25] and Growth and Developmental Models which are based upon the growth and development processes of living organisms[31].

The synthesis of natural phenomena using computers is the second branch of natural computing that provides new tools for synthesizing and studying of natural phenomena which can be used to test biological theories that cannot be tested via traditional experimental and analytic techniques. There are basically two main approaches to the simulation and emulation of nature in computers: using tools for studying the fractal geometry of nature and using artificial life techniques[23].

There are a number of techniques for modeling fractal patterns and structures; these techniques include cellular automata [1, 33], L-systems [2], iterated function systems [19, 27, 28], particle systems [37] and Brownian motion [11, 29].

Computing with Natural Means (molecular computing) is the third branch of natural computing that employs natural materials (e.g., molecules) for computing. Computing with natural means is the approach that brings the most radical change in paradigm. The question that led to the thinking and creation of this approach was: "What are the other means or media which can be used to perform computation in place of silicon?" Motivated by the need to identify alternative media for computing, researchers are now trying to design new computers based on molecules, such as: Membrane Computing [13, 14, 15, 17, 18], DNA Computing [16] and Quantum Computing [3].

All forms of molecular computing are currently in their infancy. But in the long run they are likely to replace traditional silicon computers which face barriers in reaching higher levels of performance. This paper will present how clonal selection algorithm, which is inspired from the human body immunity, can be used to help in designing P systems. This paper is organized as follows: section 2 represents a background about P systems. Section 3 deals with the clonal selection algorithm which will be used as a helping tool for designing P systems. Section 4 tackles the problem of designing P systems and the

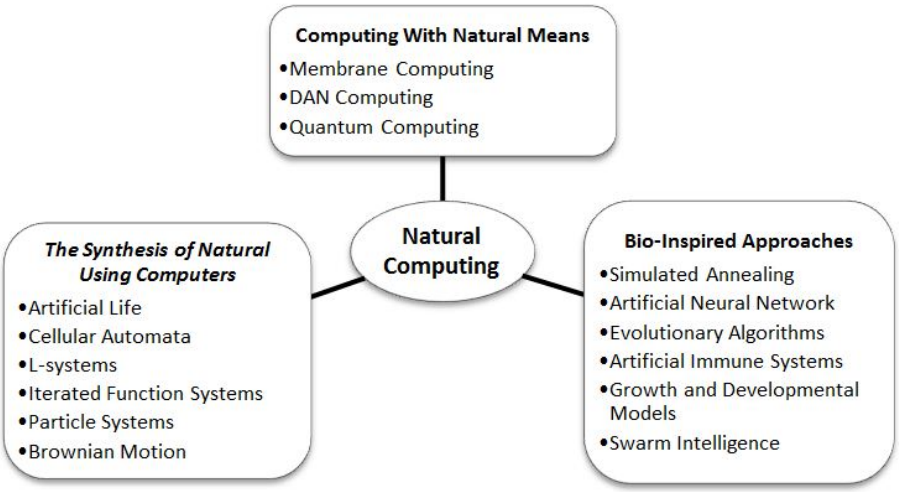

FIGURE 1. illustration of natural computing branches

experimental results of applying the clonal selection algorithm. Finally section 5 gives some conclusions and future work remarks.

2. MEMBRANE COMPUTING

Membrane computing (MC) or P systems, is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures [23]. Membrane computing (P systems) is the branch of Molecular Computing initiated by Gheorghe Paun discussed in a paper entitled Computing With Membranes [15] and this is the reason for calling it P systems.

A P system is a computing model which abstracts the way alive cells process chemical compounds in their compartmental structure. In short, in the regions defined by a membrane structure, one finds objects which evolve according to given rules. The objects can be described by symbols or strings of symbols. By using the rules in a nondeterministic and maximally parallel manner, one gets transitions between system configurations. A sequence of transitions is a computation. With a halting computation, one can associate a result, in the form of objects present in a given membrane in the halting configuration or expelled from the system during computation.

Various ways of controlling the transfer of objects from a region to another and applying the rules, as well as possibilities to dissolve, divide, create, or move membranes, were considered. Moreover, tissue P systems, neural P systems, and population P systems were investigated. Many of these variants lead to computationally universal systems, while several variants with an enhanced parallelism are able to "solve" NP-complete problems in polynomial (often, linear) time, by making use of an exponential space. A series of applications, in biology, linguistics, computer science, management, and many other areas were reported [13].

Formally, a P system with active membranes is a construct of the form below:

$$\prod = (O, H, \mu, \omega_1, ..., \omega_m, R)$$

where:

(1) $m \leftarrow 1$(the initial degree of the system);
(2) O is the alphabet of objects;
(3) H is a finite set of labels for membranes;
(4) μ is a membrane structure, consisting of m membranes initially having neutral polarizations, labeled (not necessarily in a one-to-one manner) with elements of H;
(5) $\omega_1, ..., \omega_m$ are strings over O, describing the multisets of objects placed in the m regions of μ;
(6) R is a finite set of developmental rules, of the following forms:

a: $[a \rightarrow v]_h^e$, for $h \in H, e \in \{+, -, 0\}, a \in O, v \in O^*$
Object evolution rules, associated with membranes and depending on the label and the charge of the membranes. Hint: only for simplicity, the label is written only one time outside the brackets and the internal label is omitted.

b: $a[v]_h^{e_1} \rightarrow [b]_h^{e_2}$, for $h \in H, e \in \{+, -, 0\}, a, b \in O$
Communication rules: an object is introduced in the membrane, and possibly modified during this process; the polarization of membrane can also be modified, but its label may not.

c: $[a]_h^{e_1} \rightarrow []_h^{e_2}$, for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
Out-communication rules; an object is sent out of the membrane, and possibly modified during this process; the polarization of the membrane can also be modified.

d: $[a]_h^e \to b$, for $h \in H, e \in \{+, -, 0\}, a, b \in O$
Dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified.

e: $[a]_h^{e_1} \to [b]_h^{e_2}[c]_h^{e_3}$, for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$
Division rules for elementary membranes: in reaction to an object, the membrane is divided into two membranes with the same label, and possibly of different polarizations. The object specified in the rule is replaced in the two new membranes possibly by new objects; the remaining objects are duplicated and may evolve in the same step by rules of type (a). It is possible to allow the change of membrane labels. For instance, a division rule can take the more general form below.

$$[a]_{h_1}^{e_1} \to [b]_{h_2}^{e_2}[c]_{h_3}^{e_3}, \text{ for } h_1, h_2, h_3 \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$$

The change of labels can also be considered for rules of types (b) and (c). The possibility of dividing membranes into more than two copies or even of dividing non-elementary membranes can be considered. In such case, all inner membranes are duplicated in the new copies of the membrane.

It is important to note that in case of P systems with active membranes, the membrane structure evolves during computation by decreasing the number of membranes, due to dissolution operations and increasing the number of membranes by division. The increase can be exponential in a linear number of steps: using a division rule successively, due to the maximal parallelism, 2^n copies of the same membrane can be obtained. This is one of the most frequently investigated ways of obtaining an exponential working space in order to trade time for space and solve computationally hard problems, i.e. NP-complete problems, in polynomial or even linear time [13].

3. The Clonal Selection Algorithm

The clonal selection principle is an algorithm used by the immune system to describe the basic features of an *immune response* to an antigenic stimulus. The clonal selection principle is depicted in figure 2. The principle establishes the idea that only those cells that recognize the antigens proliferate, thus being selected against those which do not. Clonal selection operates on both T cells and B cells. The *immune response* occurs inside the lymph nodes .When an animal is exposed to an antigen, some subpopulation of its bone marrow's derived cells (B lymphocytes) respond by producing antibodies. Each cell secretes only one kind of antibody, which is relatively specific for the antigen. By binding to these immune receptors, with a second signal from accessory

cells, such as the T-helper cell, an antigen stimulates the B cell to proliferate (divide) and mature into terminal (non-dividing) antibody secreting cells, called *plasma cells*. While plasma cells are the most active antibody secretors, large B lymphocytes, which divide rapidly, also secrete Ab, albeit at a lower rate. While B cells secrete Ab, T cells do not secrete antibodies, but play a central role in the regulation of the B cell response and are core in cell mediated immune responses. Lymphocytes, in addition to proliferating or differentiating into plasma cells, can differentiate into long-lived B *memory cells*. Memory cells circulate through the blood, lymph and tissues, probably not manufacturing antibodies [32], but when exposed to a second antigenic stimulus commence differentiating into large lymphocytes capable of producing high affinity antibody, preselected for the specific antigen that had stimulated the primary response. Figure 2 depicts the clonal selection principle [10]. The main features of the clonal selection theory are described below:

- the new cells are copies of their parents (clone) subjected to a mutation mechanism with high rates (somatic hypermutation);
- elimination of newly differentiated lymphocytes carrying self-reactive receptors;
- proliferation and differentiation on contact of mature cells with antigens
- The persistence of forbidden clones, resistant to early elimination by self-antigens, as the basis of autoimmune diseases.

The analogy with natural selection [24] should be obvious, the fittest clones being the ones that best recognize antigen or, more precisely, the ones that are triggered best. For this algorithm to work, the receptor population or repertoire has to be diverse enough to recognize any foreign shape. A mammalian immune system contains a heterogeneous repertoire of approximately 10^{12} lymphocytes in human [32], and a resting (unstimulated) B cell may display around $10^5 - 10^7$ identical antibody-like receptors. The repertoire is believed to be complete, which means that it can recognize any shape.

In our case the repertoire contains P systems; each p system will represent an antibody. The best antibody achieves the smallest difference from our target (solution). The smallest difference is zero in our case. Look at subsection 4.1. for more details about the affinity measure.

4. The Problem and Experimental Results

Designing a P system evaluation rules in order to perform a specific task is a hard job. In many cases, the designer has an idea about membrane structure, initial multi-sets and approximately the set of rules necessary to describe the

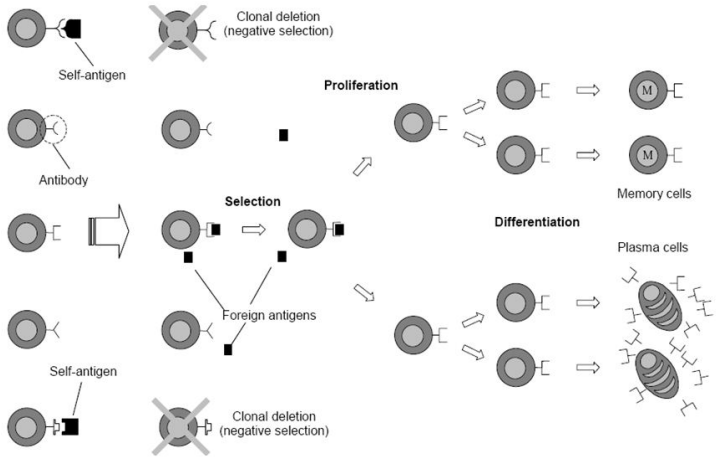

FIGURE 2. The clonal selection principle, small resting B cells created in the bone marrow, each carry a different receptor type. Those cells carrying receptors specific for the antigen, proliferate and differentiate into plasma and memory cells.

P system, but a small mistake in the description of the initial configuration or in the set of rules can lead to undesired consequences [12].

In this paper, an example of designing a P system using the clonal selection algorithms is presented. In order to do this, some information about the problem should be known, namely: membrane structure, initial multi-sets and approximately the set of rules necessary to describe the P system.

The clonal selection is applied on designing a simple p system for computing a simple mathematical operation, namely, square of 4. Clonal selection results will be compared with the previous work that solved the same problem using genetic algorithms; we used the same set of rules for repertoire initialization and the same affinity measure used in [12].

An initial repertoire of rules will be generated. Such repertoire will be evolved according to the clonal selection algorithm using cloning and mutation in order to reach the desired P system, of course by the help of a predefined affinity measure.

Only rules of type (a) and (d), namely evolution and dissolution rules respectively are used in our experiments. The initial repertoire is considered to have the same configuration, the initial configuration goes as follows.

$\prod = (O, H, \mu, \omega_1, ..., \omega_m, R_i)$

- The alphabet $O = x, y, z, m, n, u, v$

TABLE 1. The set of rules R

$r_1 : [x \to xy]_a$	$r_7 : [n \to m]_a$	$r_{13} : [x \to \lambda]_b$
$r_2 : [y \to yc]_a$	$r_8 : [u \to v]_a$	$r_{14} : [y \to \lambda]_b$
$r_3 : [c \to y_2]_a$	$r_9 : [m]_a \to y$	$r_{15} : [y \to c]_a$
$r_4 : [x \to yc]_a$	$r_{10} : [n]_a \to x$	$r_{16} : [c \to \lambda]_a$
$r_5 : [m \to n]_a$	$r_{11} : [u]_a \to c$	$r_{17} : [v \to m]_a$
$r_6 : [n \to u]_a$	$r_{12} : [v]_a \to x$	$r_{18} : [v]_a \to y$

- The set of labels $H = a, b$
- The membrane structure $\mu = [[]_a]_b$
- The initial multisets $w_a = x^2ym, w_b = \Phi$
- R is the set of rules such that $R_i \in R, R$ is explained in table 1.

4.1. **The Clonal Selection Algorithm.** The clonal selection algorithm with the properties described in section 3 is depicted below as Pseudocode.

```
Begin
 t=0;
 Initialize initial repertoire p(t);
 Identify the affinity function;
 Validate repertoire;
 Evaluate p(t);
   While (condition) do
   Begin
      1.t= t+1;
      2.Select C from p(t-1);
      3.Clone C to form C';
      4.mutate C' to form C";
      5.validate C";
      6.Select individuals from C"(t) and P(t-1) to create P(t);
      7.Metadymanics;
   End
End
```

A brief explanation of each function in the clonal selection algorithm is depicted below.

- Initial repertoire Initialization
  The initial repertoire is initialized with a randomly selected subset of rules, which are depicted in table 1. The maximum number of rules in each individual is 14 rules.

- Repertoire Validation
  The validation is made by ensuring that: for each P system and each membrane. No two rules are triggered by the same object, i.e. if a P system $\prod_1$ contains the following rules:
  $r_1 : [n]_a \to x$
  $r_2 : [n]_a \to c$
  $r_3 : [n]_a \to x$
  $r_4 : [x \to xyz]_b$
  $r_5 : [x \to \lambda]_b$

  Two rules from r_1, r_2 and r_3 are selected randomly to be removed from $\prod_1$; also one rule is selected randomly from r_4 and r_5 to be deleted.
- Repertoire evaluation and affinity measure
  The affinity measure is defined to select the highest affinity individuals. According to our example square of 4, the affinity measure is the absolute difference between the count of object z in membrane b in the halting configuration of the P system and the expected number of such objects in the ideal state, i.e., 16 objects of z. In order to prevent non-ending computations, the number of computations is limited to 20 steps.
- The selection mechanism
  The repertoire is evaluated, then the highest affinity members are selected in order to be cloned. The mutation operation is performed on the cloned members. The new population's individuals are selected from the old one and from the mutated cloned members. This ensures that our population is continuously enhanced by time. Results show that selecting the best individuals is always better than replacing the old generation by a new one.
- Mutation
  Mutation is applied as follows: Given a rule $[x \to y]_h$ such that $x \in O$, and $y \in O^*$, the mutation operator changes the object x by one from O other than x, or the object w, such that $w \in y$, by one object from $O - \{w\}$ or by λ.

  For a dissolution rule $[x]_h \to y$, the mutation operator changes the object x by a different one from O, and y by a different one from $O \cup \{\lambda\}$.
- Metadymanics
  To keep repertoire diversity, and enhance the exploration of solutions in space, a number of randomly generated individuals from the set of rules R are added in each iteration to the repertoire.

- Algorithm setting
  The following setting is used in our experiments; this setting is the same setting mentioned in [12] to enable us compare both algorithms: Repertoire size = 30 individuals, maximum number of generations= 30.

4.2. **Experimental Results.** Table 2 explains a comparison between 4 experiments using genetic algorithm depicted from [12] and three experiments using clonal selection with a new mutation mechanism. Each experiment consists of 30 runs. Genetic algorithm uses a fixed mutation rate in each experiment while we use clonal selection with mutation ranges which are applied as follows: The first experiment mutation range was from 0.1 to 0.4. The second experiment mutation range was from 0.5 to 1.0. The third one was from 0.1 to 1.0. It is clear from observing clonal selection results that low mutation rates give good results. On the other hand, it is too difficult for high mutation rates to find a solution. This situation is reversed in genetic algorithm cases where high rates find a solution with great difficulty, and low rates find no solutions.

TABLE 2. A comparison between clonal selection algorithm and genetic algorithm

experiment	Clonal Selection		Genetic Algorithm		
	Mutation	successful runs	Crossover	Mutation	successful runs
1	0.1 to 0.4	17/30	0 0.5	0.5 0.5	0/30 0/30
2	0.5 to 1.0	0/30	0.8	0.8	1/30
3	0.1 to 1.0	15/30	1.0	0.8	1/30

Instead of the usual mutation method we used an adaptive mutation mechanism, i.e. mutation value is proportional to the individual affinity, high affinity individuals have low mutation value and low affinity individuals have high rate of mutation. This proposed adaptive mutation takes into consideration that good solutions don't distorted too much, on the other hand low affinity solutions needs more changes, so we assign it higher mutation rate.

One can also observe that there is no significant difference in results between the first mutation's range (0.1 to 0.4) and the third one (0.1 to 1.0). This is because the range (0.1 to 0.4) is applied in the two cases, and the best individuals are chosen to be included in the new repertoire.

Cloning makes the repertoire almost full with good solutions, which means that the algorithm can fall in a local optimum solution. However this is prevented by using meta-dynamics mechanism which adds randomly initializes

TABLE 3. The best P systems rules generated by the clonal selection algorithm

$[z \to z, z]_a$	$[y \to m]_a$	$[v]_a \to y$
$[n \to u]_a$	$[u]_a \to n$	$[x \to z, z]_a$
	$[m \to n]_a$	

individuals to keep repertoire diversity. This maintains the balance between exploration and exploitation.

It is clear that clonal selection algorithm surpasses genetic algorithm, at a time when genetic algorithm finds one solution from 30 runs, clonal selection algorithm find 17 solutions from 30 runs. This is because of cloning and adaptive mutation which differentiates between individuals according to an affinity value. Furthermore, clonal selection algorithm finds more than one solution in the same run. This is a very important advantage, where one can choose the most appropriate initial configuration and structure when one uses this algorithm in designing a P system for more complex problems. It also finds solutions before reaching half of the maximum generation's number in most runs. This means that the algorithm converges are very fast.

Plingua simulator [16] is used for calculating the affinity of each P system in the repertoire. One of the best P system's rules that achieved affinity measure is depicted in table 3. Table 3 rules are used for generating a *plingua* code illustrated below. This code is executed and the generated output determines the affinity of these rules. A graphical representation that explains the *plingua* code execution is depicted in figure 3.

```
@model<membrane_creation>
def numOfZs()
{
  @mu = [[]'a]'b;
  @ms(a) = x,x,y,m;
  @ms(b)=#;

  [z-->z,z]'a;  [y-->m]'a;  [v]'a-->y;
  [n-->u]'a;    [u]'a-->n;  [x-->z,z]'a;
  [m-->n]'a;
}
def main()
{
  call numOfZs();
}
```

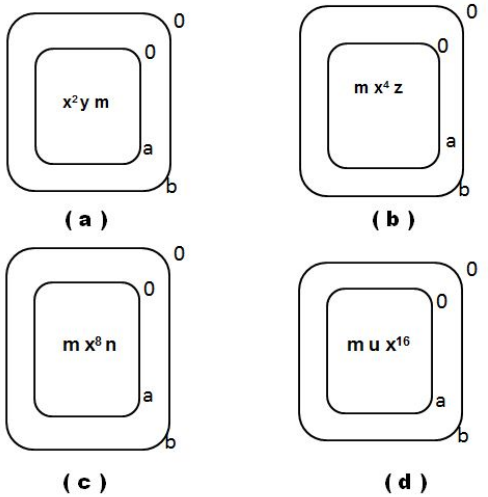

FIGURE 3. . Representation of the membranes generated by the p system rules depicted in table 3, (a) represents the initial membrane and (d) represents the final one.

5. Conclusion and Future Work

Membrane computing is an interesting area of research. Designing a P system to solve complex real world problems is not easy, and the researcher has no alternatives to do this hard task by hand. Moreover, Membrane Computing solutions to real-life problems need to be quite precise in the design in order to find a sharp simulation of the processes [35]. For all these reasons, the candidate explored the use of Clonal Selection Algorithm as an aid for designing P systems and as an alternative of genetic algorithms. Results show that clonal selection can find 17 solutions out of 30 runs while genetic algorithm finds one solution out of 30 runs.

Many questions arise regarding the automation of P systems design using Clonal selection algorithm. One of them, the affinity measure, being the most complex factor, discusses how the researcher can determine that a P system is better than another. In the candidate's example, a simple problem is used, but in more complex problems, it needs more work. The second question is what about the mutation which is responsible for the repertoire maturity. In this paper, mutation is applied to only objects, but it could be applied to membrane structures, polarizations, activation and inactivation of rules.

Another issue is what the appropriate selection method is, which is more adequate for such types of applications. Meta-dynamics also needs a deeper

view, and it could be implemented by applying very high mutation rates to a number of selected members besides adding some new randomly generated ones.

In this paper, the use of clonal selection algorithm is just illustrated as an aid in designing a simple P system, but the target is to use clonal selection in more complex real world problems. Hybridization between bio-inspired approaches [9], depicted on figure 1, may be used for the automation of P systems design, and may result in benefits better than clonal selection only, so this is another open point of research.

References

[1] A. Ilachinski, *Cellular automata: A discrete universe*, World Scientific, Singapore, 2001.

[2] A. Lindenmayer, *Mathematical models for cellular interaction in development, Parts I and II*, Journal of Theoretical Biology, 18(1968), No. 3, pp. 280-315.

[3] A.O. Pittenger, *An introduction to quantum computing algorithms*, Birkhauser, 2000.

[4] A.P. Engelbrecht, *Computational Intelligence: An Introduction*, Second Edition, John Wiley and Sons. 2007.

[5] C.S. Calude, G. Paun, Computing with cells and atoms. Taylor & Francis. 2001.

[6] D. Dasgupta, *Artificial immune systems and their applications*, Springer-Verlag, 1999.

[7] E. Aarts, and J. Korst, *Simulated Annealing and Boltzman machines - A stochastic approach to combinatorial optimization and neural computing*, New York, USA :John Wiley & Sons, Inc., 1989.

[8] E. Bonabeau, M. Dorigo and T. Theraulaz, *Swarm intelligence: From natural to artificial systems*, New York: Oxford University Press, 1999.

[9] E. Nabil, A. Badr and I. Farag, *An immuno-hybrid genetic algorithm*, International Journal of Computers, Communications & Control, 4(2009), No. 4, pp. 374-385.

[10] F. M. Burnet, *Clonal Selection and After*, in Theoretical Immunology, G. I. Bell , A. S. perelson,G. H. Jr. Pimbley, ed., Marcel Dekker Inc., 1978, pp. 63-85 .

[11] Fournier, D. Fussell, L. Carpenter, *Computer rendering of stochastic models*, Communications of the ACM, 25(1982), no. 6, pp. 371-384.

[12] G. Escuela, M.A. Gutirrez-Naranjo, *An application of Genetic Algorithms to Membrane Computing*, In M.A. Martinez-del-Amor, Gh. Paun, I. Prez-Hurtado, A. Riscos-Nunez (Eds.), Eighth Brainstorming Week on Membrane Computing (BWMC 2010). Fnix Editora, 2010.

[13] G. Paun, *Applications of membrane computing*, Springer-Verlag, Berlin, 2002.

[14] G. Paun, *Computing with Membranes*, Journal of Computer and System Sciences, 61(2000), No. 1, pp. 108-143.

[15] G. Paun, *Computing with membranes*,TUCS Report 208, Turku Center for Computer Science, 1998.

[16] G. Paun, G. Rozenberg, A. Saloma, *DNA computing*, Springer-Verlag, 1998.

[17] G. Paun, *Membrane Computing: An Introduction*, Springer-Verlag, Berlin, 2002.

[18] http://psystems.disco.unimib.it/.

[19] J. Hutchinson, *Fractals and self-similarity*, Indiana Journal of Mathematics, 30(1981), No. 5, 713-747.

[20] J. Kennedy, R. Eberhart, Y. Shi, *Swarm intelligence*, Morgan Kaufmann Publishers, 2002.

[21] J.J. Holland, *Adaptation in natural and artificial systems*, MIT Press, 1975.
[22] L. Fausett, *Fundamentals of neural networks: architectures, algorithms, and applications*, Prentice-Hall, 1994.
[23] L.N. De Castro, F.J. Von Zuben, *Recent Developments in Biologically Inspired Computing*, Idea Group Publishing, 2005.
[24] L.N. De Castro ,F.J. Von Zuben, *Artificial Immune Systems: Part I - Basic Theory and Applications*, Technical Report - RT DCA, 1999.
[25] L.N. De Castro, J.I. Timmis, *Artificial immune systems: A new computational intelligence approach*, Springer-Verlag, 2002.
[26] M. Garca-Quismondo, R. Gutierrez-Escudero, I. Perez-Hurtado,M.J. Perez-Jimenez, A. Riscos-Nunez, *An overview of P-Lingua 2.0.*,in Membrane Computing,Paun, Gheorghe and Prez-Jimnez, Mario and Riscos-Nunez, Agustin and Rozenberg, Grzegorz and Salomaa, Arto,ed. ,Springer Berlin, 5957(2010), pp.264-288.
[27] M.F. Barnsley, S. Demko, *Iterated function systems and the global construction of fractals*, Proceedings of the Royal Society of London, 339 (1985), No. 1817, pp. 243-275.
[28] M.F. Barnsley, *Fractals everywhere*, Academic Press, 1988.
[29] R.F. Voss, *Random fractals forgeries.* In RA. Earnshaw, Ed., Fundamental Algorithms for Computer Graphics, Berlin: Springer-Verlag,1985, pp. 805-835.
[30] S. Kirkpatrick, C.D. Gerlatt, M.P. Vecchi, *Optimization by simulated annealing*, Science, 220(1983), No. 4598, pp. 671-680.
[31] [31] S. Kumar, P.J. Bentley, *On growth, form and computers*, Academic Press, 2003.
[32] S. Perelson, M. Mirmirani, G.F. Oster, *Optimal Strategies in Immunology I. B-Cell Differentiation and Proliferation*, Journal of mathematical biology, 3(1978),No. 3, pp. 325-67.
[33] S. Wolfram, *Cellular automata and complexity*, Perseus Books, 1994.
[34] T. Back, DB. Fogel, Z. Michalewicz, *Evolutionary computation 2: advanced algorithms and operators*, Bristol and Philadelphia: Institute of Physics Publishing (IOP), 2000.
[35] T. Hinze, T. Lenser, G. Escuela, I. Heiland, S. Schuster, *Modeling Signaling Net- works with Incomplete Information about Protein Activation States: A P System Framework of the KaiABC Oscillator*, Lecture Notes in Computer Science, 5957(2010), pp. 316-334.
[36] W. McCulloch, W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 52(1990), No. 1, pp. 99-155.
[37] W.T. Reeves, *Particle systems - A technique for modeling a class of fuzzy objects*, ACM Transactions on Graphics, 2(1983), no. 2, pp. 91-108.

[1]Department of Computer Science, Faculty of Information Technology Misr University for Science and Technology, Al-Motamayez District, Postal code: 15525, 6th of October City, Egypt

[2] Department of Computer Science, Faculty of Computers and Information Cairo University, 5 Dr. Ahmed Zewail Street, Postal Code: 12613,Orman, Giza, Egypt

E-mail address: emadnabilcs@gmail.com
E-mail address: a.badr.fci@gmail.com, i.farag@fci-cu.edu.eg