STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 1, 2012

A MEMBRANE-IMMUNE ALGORITHM FOR SOLVING THE MULTIPLE 0/1 KNAPSACK PROBLEM

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

Abstract. In this paper a membrane-immune algorithm is proposed, which is inspired from the structure of living cells and the vertebrate immune system. The algorithm is used to solve one of the most famous combinatorial NP-complete problems, namely the Multiple Zero/One Knapsack Problem. Various heuristics, like genetic algorithms, have been devised to solve this class of combinatorial problems. The proposed algorithm is compared with two genetic based algorithms and overcame both of them. The algorithm is evaluated on nine benchmarks test problems and surpassed both of the genetic based algorithms in six problems, equaled with one of them in two problems and lost in one problem, which indicates that our algorithm surpasses in general genetic algorithms. We claim that the proposed algorithm is very useful in solving similar combinatorial NP-complete problems.

1. Introduction

Membrane computing is a branch of natural computing which investigates computing models abstracted from the structure and the functioning of living cells and from their interactions in tissues or higher order biological structures.

Membrane Computing was introduced by Gh. Paun in [6] under the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. The devices of this model are called P systems. A P system consists of a membrane structure, in the compartments of which one places multi-sets of objects which evolve according to given rules in a synchronous non-deterministic maximally parallel manner. The basic idea is to consider a distributed and parallel computing device structured in an arrangement of membranes which delimit compartments

Received by the editors: October 12, 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, Knapsack Problem.

where various chemicals (objects in our case) evolve according to local reaction rules. The objects can be eventually sent to the environment or to adjacent membranes under the control of specific rules. The reaction rules are applied in a parallel manner, with the objects to evolve by them and with the reactions themselves chosen in a non-deterministic manner. In this way, we can define transitions from one configuration to another configuration of our system and hence we can define computations [7]. A computation provides a result, for instance, in the form of the number of objects present in the halting configuration in a specified compartment, or in the form of a special object, yes or no, sent to the environment at the end of the computation, this is the way of answering a decision problem that the system had to solve. An abstraction of membrane structure is depicted in figure 1.

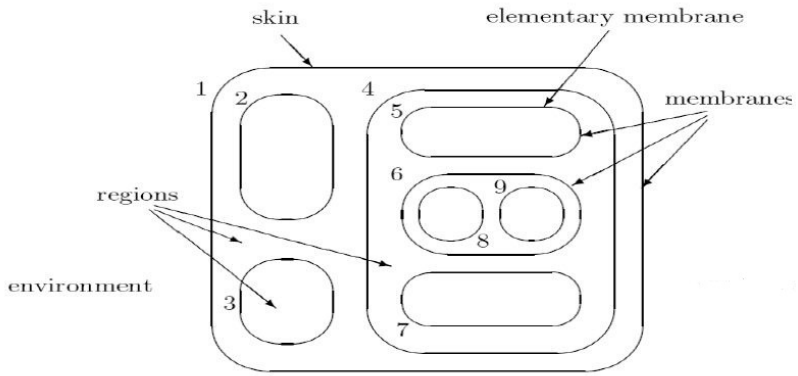

FIGURE 1. the membrane structure.

This paper suggests a membrane-immune algorithm which is an approximate algorithm for NP-complete optimization problems, the membrane-immune algorithm borrows nested membrane structures, rules in membrane separated regions, transporting mechanisms through membranes from P systems and the cloning selection principle which is imported from the vertebrate immune system and uses all these ingredients in solving NP-complete optimization problems approximately. We evaluated the proposed algorithm by applying it to one of the most famous NP-complete problems, the multiple 0/1 knapsack problem. The paper is organized as follows. Section 2 describes the multiple 0/1 knapsack problem, in section 3 an explanation of the clonal selection

principle is given, section 4 explains in detail the proposed algorithm and the experimental results, finally conclusions and some remarks are discussed in section 5.

2. The Multiple Zero/One Knapsack Problem

The problem we study here is a generalization of the 0/1 simple knapsack problem. In the simple version, we are given a knapsack of capacity c, and n objects. Each object has a weight w_i, and a profit p_i, $1 \leq i \leq n$, The objective is filling the knapsack with the objects that yield the maximum profit without firing the capacity constraint. The problem is also known as the single-line integer programming problem [3, 4].

The multiple 0/1 knapsack problem consists of m knapsacks of capacities c_1, c_2,...,c_m and n objects, each of which has a profit p_i, $1 \leq i \leq n$. Unlike the simple version in which the weights of the objects are fixed, the weight of the ith object in the multiple knapsack problem takes j values, $1 \leq j \leq m$. The ith object weighs w_{ij} when it is considered for possible inclusion in the jth knapsack of capacity c_j. The objective is finding a vector $\vec{x}$=(x_1, x_2,..., x_n) that guarantees that profit is maximized and in the same time takes into consideration that no knapsack is overfilled.

This problem is also known as the zero/one integer programming problem [12], and as the 0/1 linear Programming problem [14]. The multiple 0/1 knapsack problem can be thought as a resource allocation problem, where we have m resources (the knapsacks) and n objects. Each resource has its own budget (knapsack capacity), and w_{ij} represents the consumption of resource j by object i. Once more, we are interested in maximizing the profit, while working within a certain budget.

The popularity of knapsack problems stems from the fact that it has attracted theoreticians as well as practitioners [14]. Theoreticians enjoy the fact that these simple structured problems can be used as sub-problems to solve more complicated ones, practitioners on the other hand, enjoy the fact that these problems can model many industrial opportunities such as cutting stock, cargo loading, and the capital budget. Table 1 gives a formal definition of the multiple 0/1 knapsack problem.

2.1. **Related Work.**

P systems are parallel molecular computing models based on processing multisets of objects in cell-like membrane structures. In [9] a membrane algorithm used to solve the multidimensional 0-1 knapsack problem in linear time is given. The algorithm uses P system recognizers with input and with active membranes using two-divisions. This algorithm can also be modified to solve general 0-1 integer programming problem. In [11] the Knapsack problem is solved using a family of deterministic P systems with

active membranes using two-divisions. The number of steps of any computation is of linear order, but a polynomial time is required for pre-computing resources.

The works in [9, 11] use membrane computing to generate the whole exponential search space, then apply an exhaustive search to find the global optimal solution. Until now membrane algorithms have no real implementation and still in its infancy form. Researchers till the moment do not have a precise conception about implementation of p systems, an in vitro, in vivo or in silico implementation. Our algorithm differs from the two works previously mentioned that it produces an approximate accepted solutions and can be implemented in the current silicon based computers, and easily can run using parallel or grid computing.

TABLE 1. The multiple 0/1 knapsack problem formalization

knapsacks:	$1, 2, ..., m$
capacities:	$c_1, c_2, ..., c_m$
objects:	$1, 2, ..., n$
profits:	$P_1, P_2, ..., P_n$
Weights:	$KS_1 : w_{11}, w_{12}, ..., w_{1n}$ $KS_2 : w_{21}, w_{22}, ..., w_{2n}$ ..... $KS_m : w_{m1}, w_{m2}, ..., w_{mn}$
Feasible solution:	$\vec{x}$=(x_1,..., x_n), $x_i \in \{0, 1\}$, s.t. $\sum_{i=1}^{n} w_{ij}x_i \leq c_j, 1 \leq j \leq m$
Affinity function:	Maximize $\sum_{i=1}^{n} P_i x_i$
Optimal solution:	A feasible solution that gives the maximum profit

3. The Clonal selection principle

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. An abstraction of clonal selection principle [10] is described in figure 2.

The principle establishes the idea that only those cells that recognize the antigens are those with high ratio proliferate. 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, 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. The main features of the clonal selection theory [5] are:

- 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.

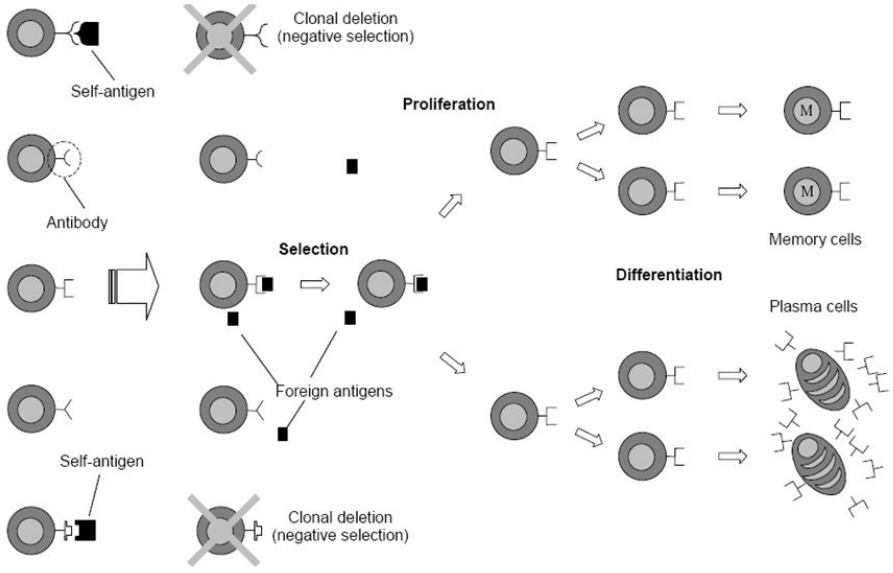

FIGURE 2. The clonal selection principle.

The fittest clones are the ones that best recognize antigens 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, 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 of an antigen [10].

4. The Membrane-Immune Algorithm and Experimental Results

The membrane-immune algorithm depicted in figure 3 is described below as Pseudocode.

```
1. Begin
2.   Initialize the initial repertoire R randomly;
3.   Identify the affinity measure;
4.   Validate repertoire;
5.   While (condition) do
6.     Begin
7.        For each P system in R.
8.        Begin
9.         a. Perform mutation over all solutions in each compartment;
10.        b. In every region, the best and worst solutions are sent
              to the adjacent inner and outer regions, respectively;
11.       End;
12.      Perform Clonal/Negative Selection according affinity;
13.      Metadymanics;
14. End;
15. End;
```

In the following subsections an explanation of how the algorithm is working in detail is given.

4.1. The Algorithm Structure.

The repertoire here is a set of P systems, each one consists of set of nested membranes, each region has a set of solutions and a simple mutation algorithm that mutates all solutions in this region, as depicted in figure 4. These solutions are initialized randomly.

In every region there are a few solutions of the optimization problem to be solved and a mutation algorithm works on them. After the initial settings all solutions are updated by the mutation algorithm placed in regions, simultaneously, in every region, the best and worst solutions, with respect to the optimization criterion, are sent to the adjacent inner and outer regions, respectively. The best solution exists in the innermost region of a P system.

The process of updating and transporting solutions is repeated until a termination condition is satisfied. In our implementation a fixed number of

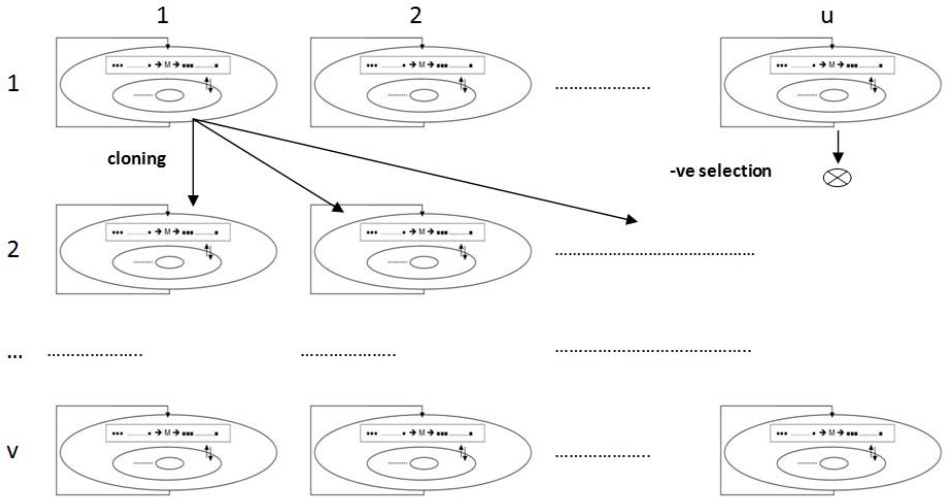

FIGURE 3. How the membrane-immune algorithm works, u represents the repertoire size; v represents how many times the cloning selection principle is applied.

iterations is used as a termination condition as explained in the algorithm setting, see subsection 4.8.

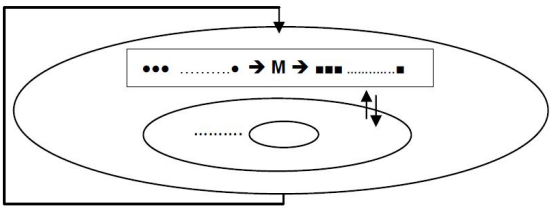

FIGURE 4. The Structure of repertoire's single p system.

4.2. **Repertoire Initialization.** All generated p systems' initial solutions are feasible. This means that all the generated solutions satisfy the knapsack constraints, and all subsequently infeasible individuals were made feasible by dropping some items randomly from the knapsacks until feasibility is obtained.

It might be allowed to let a p system to contain infeasible individuals (representing invalid solutions), to increase the variation of solutions. The infeasible individuals are often penalized so as to means lower affinity when

comparing to other feasible solution. According to [1] it does not indicate that it gives any benefit to allow infeasible solutions to be included, so we did not permit infeasibility of solutions in our implementation.

4.3. **The affinity measure.** The affinity measure for a solution is the same as described in table 1.

4.4. **Repertoire Validation.** As mentioned above infeasible solutions are not included in the repertoire, and thus no penalty function is used. Due to random initialization or mutation, infeasible solutions are produced, solutions' validation is performed by choosing a random bit contains value 1 to be flipped into 0 until this solution satisfies all knapsacks' constraints.

4.5. **Cloning Selection Mechanism.** Each p system in the repertoire is supposed to be enhanced continuously by time because of maturation performed by mutation. The higher affinity solutions will be found in the inner most regions and cloning will performed proportional to these solutions' affinity, i.e. a high membrane inner most solution affinity means a high cloning rate; a low membrane inner most solution affinity means a low cloning rate or a negative selection (clonal deletion).

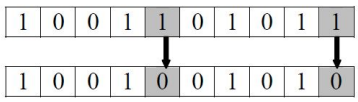

FIGURE 5. An example of mutation.

4.6. **Mutation.** The concept of mutation is essential to membrane-immune algorithm, as it is the only way for repertoire diversification and maturation. Without sufficient mutation the population will tend to converge towards a few good solutions, possibly representing local optima. On the other hand, with too much mutation the search will have problems of focusing on potentially good solutions; this is the tradeoff between exploration and exploitation. According to our experiments we found that 0.1 mutation rate is suitable.

4.7. **Meta-dynamics.** To keep the repertoire diversity, a number of randomly generated P systems are added to the repertoire, our experiments showed that there is no need to perform this step, but in other problems it could be useful.

4.8. **Parameters Setting.** The following setting is used in our experiments: Repertoire size = 30 P systems, maximum iterations number of each P system = 50, each P system in the repertoire has 100 nested membranes, proportional cloning is used according to the highest inner most solution affinity, clonal/negative selection iterations is performed 100 times. The proposed algorithm is tested on 9 benchmarks problem and every problem is solved 100 times. the 30 P systems that compose the population are independent at the iteration level, so they can work in a destribued system for effeciency measures.

Parameters setting in work mentioned in [2] are as follows, the population size is 100 individual, cross over rate=0.9, mutation rate = 0.01, No cloning is applied. The algorithm uses proportional selection. Total number of 100 runs is executed for each test problem. parameters setting in work mentioned in [13] are as follows, population size = 100, crossover rate = 0.6, mutation rate = 1/n, where n is the population size, No cloning is applied, The algorithm uses proportional selection. Total number of 100 runs is executed for each test problem.

Optimal solution in table 3 means the known optimum. The known optimum may be enhanced using the following suggestions, using a different stopping condition rather than fixed number of iterations, i.e. a heuristic termination condition, for example the algorithm stops when there no enhancement or the enhancement is less than certain value, also the mutation value could be degraded rather than fixed value, i.e. in the first iterations the mutation is high and degrades by incrementing iterations, this is under the assumption that the whole individuals are enhanced by time. Increasing the number of P systems allows the coverage of more search space and this could result in general enhancement to reach better optimal.

The membrane-immune algorithm is evaluated on the same nine test problems, for comparison reasons, which are previously solved by [2, 13], the abbreviations CT will be used to refer to the work [2] , and KBH for the work [13], while NBF will be used for referring to this paper. The problem sizes range from 15 to 105 objects and from 2 to 30 knapsacks. A collection of all of these problems is available from the OR-library by Beasley [8]. All information about each problem is explained in table 2, number of knapsacks, number of objects, optimal solution, also average value of solutions found by the membrane-immune algorithm in comparison with KBH and CT. Table 3 and figure 6 explains the degree of how much each algorithm finds optimal solution.

The membrane immune algorithm with the previously mentioned setting in subsection 4.8. is able to find the global optimum point exactly for all problems with a clear difference from KBH and CT, but the exception is given by the problem weing7-105. the previous setting worked badly, so we

changed the repertoire size to be 100, and the number of cloning iterations to be 400. Nevertheless most of the runs get stuck before finding the optimal solution, but still the average value of the found solutions is accepted.

The percent of optimum solutions found and the average affinity value over all runs turns out to results indicates that the membrane-immune algorithm can serve as a fast and robust approximation heuristic for combinatorial problems.

TABLE 2. Experimental results of KNH, CT and NBF that represent the average value of execution using the nine benchmarks problems solved by Khuri *et al.* in [13] and Cotta, Troya in [2].

Problem	Knapsacks	Objects	Optimal	Average			
				KBH	CT	NBF	Winner
Knap15	10	15	4015	4012.7	4015.0	4015.0	CT, NBF
Knap20	10	20	6120	6102.3	6119.4	6120.0	NBF
Knap28	10	28	12400	12374.7	12400.0	12400.0	CT, NBF
Knap39	10	39	10618	10546.9	10609.8	10618.0	NBF
Knap50	10	50	16537	16378.0	16512.0	16528.3	NBF
Sento1	30	60	7772	7626	7767.9	7772.0	NBF
Sento2	30	60	8722	8685	8716.5	8721.95	NBF
Weing7	2	105	1095445	1093897	1095386.0	1094775.0	CT
Weing8	2	105	624319	613383	622048.1	624319.0	NBF

TABLE 3. A Comparison between KNH, CT and NBF explains how much each algorithm finds the optimal solution

Problem	KBH	CT	NBF
Knap15	83%	100%	100%
Knap20	33%	94%	100%
Knap28	33%	100%	100%
Knap39	4%	60%	100%
Knap50	1%	46%	50%
Sento1	5%	75%	100%
Sento2	2%	39%	95%
Weing7	0%	40%	5%
Weing8	6%	29%	100%

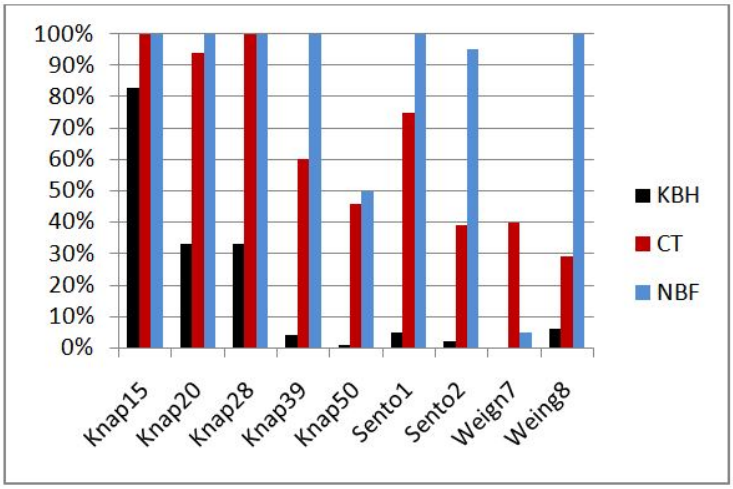

FIGURE 6. A Comparison of the three algorithms explains the percent of finding the optimal solution.

5. Conclusions and Remarks

We have proposed the membrane-immune algorithm which is inspired from the structure of living cells and the vertebrate immune system. The algorithm is evaluated by solving a famous NP-complete optimization problem namely the multiple 0/1 knapsack problem and overcame two genetic based algorithms.

Although there is a great success in the design of theoretical solutions to NP problems, these solutions have a fundamental drawback from a practical point of view. It is not clear yet what is the actual real implementation of Membrane Computing, an *in vitro, in vivo* or *in silico* implementation, any case, a membrane will have a space associated that could be a piece of memory in a computer, a pipe in a lab or the volume of a bacteria, only brute force algorithms will be able to implement little instances of such problems. If we take an *in vivo* implementation where each feasible solution will be encoded in an elementary membrane and such elementary membrane is implemented in a bacteria of mass $\cong 7 \times 10^{-16}$ kg., for example E. Coli, then, a brute force algorithm which solves an instance of an NP problem with an input size, for example 40 will need approximately a mass $\cong 6 \times 10^{24}$ kg., which equals approximately mass of the earth, so it is not practical for a P system to be implemented in solving relatively large NP-Complete problems, the proposed

membrane-immune algorithm can find approximate accepted solutions and still has the feasibility for a practical real implementation in terms of resources.

There are many possibilities for improving membrane-immune algorithm, for example different termination conditions could be used, i.e. one can terminate execution if the good solution is not changed during a predetermined number of steps, considering meta-dynamics in the earlier execution in the algorithm could enhance performance, meta-dynamics may be useful in some applications and in others not, also instead of performing cloning according to the affinity of the inner most solution, it could be performed according to the average value of all solutions for a P system, the first mechanism takes into consideration the affinity of one solution to perform cloning but the latter counts for all solutions' affinity in a p system. Because the repertoire has a number of independent P systems, the algorithm will be easily implemented in parallel, distributed, or grid computing systems which indicates a better performance. Each p system itself could be implemented in a parallel system as it contains a number of independent regions all of which has its solutions and maturation/mutation technique. Also different mutation techniques may be more suitable for different NP-complete problems like mutation per solution or Swap Mutation; the best mutation rate for repertoire is an open point, mutation may be high in early iteration and get smaller by time. The proposed structure could be involved in more complex structures as well.

Our positive results support the idea that membrane-immune algorithm is a suitable approach for tackling highly constrained NP-complete problems.

References

[1] A. Hoff, A. Lokketangen, I. Mittet, *Genetic Algorithms for 0/1 Multidimensional Knapsack Problems*, Proceedings of Norsk informatikk konferanse, NIK'96, pp. 291–301, 1996.

[2] C. Cotta, J. M. Troya, *A hybrid genetic algorithm for the 0-1 multiple knapsack problem*, In: G.D. Smith, N.C. Steele and R.F. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms* Vol. 3., Springer-Verlag, Wien New York, 1998, pp. 251–255.

[3] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization:Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[4] D. R. Stinson, *An Introduction to the Design and Analysis ofAlgorithms*, The Charles abbage ResearchCenter,Winnipeg, Manitoba, Canada, 2nd edition, 1987.

[5] F. M. Burnet, G. I. Bell, A. S. Perelson, Jr. G. H. Pimbley, *Clonal Selection and After*, In Theoretical Immunology,(Eds.), New York: Marcel Dekker Inc., 1978, pp. 63–85.

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

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

[8] J. E. Beasley, OR-Library: Distributing TestProblems by Electronic Mail, *Journal of Operational Research Society*, Vol. 41, No. 11,1990 ,pp. 1069–1072

[9] L. Pan, C. Martin-Vide, *Solving Multiset 0-1 Knapsack Problem by P Systems with Input and Active Membranes*. In Proceedings of the Second Brainstorming Week on Membrane

Computing, Sevilla, Spain, BWMC 2004, Report RGNC 01/04, University of Sevilla, 2004, pp. 342–353.
[10] L.N. De Castro, J.I. Timmis, *Artificial immune systems: A new computational intelligence approach*, Springer-Verlag, 2002.
[11] M. J. Perez-Jimenez , A. Riscos-Nunez, *A Linear-Time Solution to the Knapsack Problem Using P Systems with Active Membranes*. Membrane Computing, 2004, pp. 250–268.
[12] M. R. Garey, D. S. Johnson, *Computers and Intractability:A Guide to the Theory of NP-Completeness*, W. H. Freemanand Company, San Fransisco, 1979.
[13] S. Khuri, T. Back, J. Heitkotter, *The zero/one multiple knapsack problem and genetic algorithms*, Proceedings of the ACM Symposium on *Applied Computing*, SAC '92, ACM Press, pp.188–193, 1994.
[14] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester,West Sussex, England, 1990.

[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
E-mail address: i.farag@fci-cu.edu.eg