Journal of Computer Science, 9 (2): 264-270, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.264.270 Published Online 9 (2) 2013 (http://www.thescipub.com/jcs.toc)

Membrane Computing Inspired Genetic Algorithm on Multi-Core Processors

Ali Maroosi and Ravie Chandren Muniyandi

Center for Software Technology and Management,
Faculty of Information Science and Technology,
University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

Received 2012-09-27, Revised 2013-04-04; Accepted 2013-04-12

ABSTRACT

Membrane computing is a branch of natural computing. Several studies have recently attempted to utilize the structure of membrane computing to improve intelligent algorithms. These studies have applied communication rules in membrane models to facilitate information exchange between membranes, thereby improving the performance of those algorithms. However, parallel membrane computing has not yet been considered. This study proposes a membrane computing-inspired genetic algorithm. Similar to previous studies, the algorithm also uses communication rules to facilitate information exchange. In this study, an appropriate membrane computing-inspired genetic algorithm is defined, in which each membrane can be executed over different cores in a parallel manner. The proposed algorithm can be executed over different cores and uses multi-core processing to implement parallel membrane computation. Simulation with a Colville minimization problem shows that the membrane computing inspired genetic algorithm has improved performance, with a mean error of the solution 61.9 times better than genetic algorithm.

Keywords: Membrane Computing, Tissue P Systems, Genetic Algorithms, Multi-Core Processing, Colville Function

1. INTRODUCTION

Membrane computing (also known as P systems) (Paun *et al.*, 2010) is a branch of natural computing. Since its inception, membrane computing has been used to solve various problems. Specifically, membrane computing has been used to solve biological problems, such as molecular interactions (Twycross *et al.*, 2010; Muniyandi and Abdullah, 2012), bearded vulture evolution prediction (Cardona *et al.*, 2009) and predator and prey relationship modeling. Membrane computing techniques have also been utilized to solve other problems, such as computation of the threshold of two-dimensional images (Christinal *et al.*, 2010), segmentation of images (Christinal *et al.*, 2011) and robot controlling (Buiu *et al.*, 2012). Difficult optimization problems, such as N-queens problem (Gutierrez-Naranjo and Perez-Jimenez, 2011), three-coloring problems (Adrian and Florentin, 2012) and satisfiability problems (Ishdorj *et al.*, 2010), have also been solved by membrane computing models.

The main components of membrane computing are as follows: (i) the membrane structure and the delimiting compartments, in which (ii) multi-sets of objects evolve according to (iii) (reaction) biochemically inspired rules. The rules can process both objects and membranes (Paun *et al.*, 2010). Thus, membrane computing can be defined as a framework for devising cell-like, tissue-like, or spiking-like computing models (Paun *et al.*, 2010). This study applies the tissue-like membrane computing model.

Membrane computing models have recently been used to improve intelligent algorithms (Cheng *et al.*, 2011; Zhang *et al.*, 2011; 2012a; 2012b). Quantum-inspired and differential evolution intelligent algorithms are used in previous studies based on membrane computing to solve difficult non-deterministic, polynomial-time problems.

Corresponding Author: Ali Maroosi, Center for Software Technology and Management, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

However, these studies merely use communication rules to aid information exchange between membranes.

This study introduces a Membrane Computing-Inspired Genetic Algorithm (MCIGA). MCIGA uses a parallel structure aside from communication rules to assist in information exchange between membranes. In this study, membrane computing is defined with appropriate communication rules and membranes such that each membrane has the ability to run parallel on different processors.

2. MATERIALS AND METHODS

2.1. Tissue-like P Systems

Tissue-like P systems (Pena-Cantillana *et al*., 2011) have two biological inspirations: inter-cellular communication and cooperation between neurons. The common mathematical model of these two mechanisms is a network of processors dealing with symbols and then communicating these symbols along the channels specified in advance. Tissue-like P systems contain many cells within a common environment. Two cells can communicate with each other through channels between them and all cells can communicate among each other through the environment. A tissue-like P system with degree $m \geq 1$ is expressed as follows:

$$\pi = (O,E,w_1,\ldots,w_m,R,i_{out})$$

Where:

m = The number of cells in the system
O = finite non-empty alphabet of objects
$E \subseteq O$ = the set of objects present in the environment
$w_1,\ldots w_2$ = Strings over O, representing multi-sets of objects associated with m cells at the initial state of the computation
R = A finite set of communication and transformation rules of the following form

- Transformation rules $x \rightarrow y$ allow cell $i \in O$ to consume a multi-set x to produce a new multi-set *y* inside the cell i.
- Communication rules (i,u/v,j) for $i,j \in \{0, 1, 2,\ldots,m\}$, $i \neq j$ and $u, v \in O$

$i_{out} \in \{0,1,2,\ldots,m\}$ = The output cell

A tissue-like P system of degree m is as a set of m cells (each one consisting of an elementary membrane) labeled by 1,2,…,m. Here, 0 refers to the label of the environment and i_{out} denotes the output region, which can be the region inside a cell or the environment. The strings $w_1,\ldots,w_m$ describe the multi-sets of objects placed in the m cells of the P system and $E \subseteq O$ is the set of objects placed in the environment. Each set is available in arbitrary large amount of copies.

The communication rule (I,u/v,j) can be applied over two cells labeled by i and j such that u is contained in cell i and v is contained in cell j. The communication rule denotes that the objects of the multi-sets represented by u and v are interchanged between the two cells. When either i = 0 or j = 0, the objects are interchanged between the cell and the environment. Rules are used in the framework of membrane computing, that is, in a maximally parallel way (a universal clock is considered). Each object in a membrane can only be used in one rule, which is non-deterministically chosen when there are several possibilities, but any object should participate in a rule of any form, that is, in each step, a maximal set of rules should be applied.

2.2. Genetic Algorithms

Evolutionary Algorithms (EAs) (Affenzeller *et al*., 2009) are generic population-based meta-heuristics inspired by biological evolution to address combinatorial optimization problems. Four main EAs have been applied to different types of problem domains: Genetic Algorithms (GAs), genetic programming, evolutionary strategies and evolutionary programming. GAs were introduced to study self-adaptation in biological processes and to solve optimization problems (Affenzeller *et al*., 2009).

GAs are a class of probabilistic algorithms that start with a population of randomly generated candidates. Moreover, GAs are iterative procedures that operate on a population where individuals are evaluated according to a certain fitness value. Individuals are selected according to this value. The selected individuals produce offspring candidates, thus forming the next generation. Two operators, namely, crossover and mutation, are used to produce new individuals. Crossover takes two individuals called parents and then produces one or two new individuals called offspring. In its simplest form, crossover works by swapping pieces of information from the parents. The second operator is called mutation, which is applied by modifying an information unit in one individual according to a mutation rate (Affenzeller *et al*., 2009). A simple pseudo code for GAs is as follows:

Step 1: Randomly generate an initial population.
Step 2: Select pairs of individuals based on the fitness function.
Step 3: Produce next generations from the selected pairs by applying crossover and mutation.

Step 4: Replace previous generation with the new generation if the new generation is better or identical to the previous generation.
Step 5: The algorithm is finished when the coverage reaches a solution or a pre-determined number of iterations. Otherwise, increase the number of iterations and go back to Step 2.

2.3. Proposed Method

The structure of the proposed MCIGA is as follows:

$$\pi_{MCIGA} = (O, E, w_1, ..., w_m, R, i_{out})$$

where, m = NumCores where NumCore is the number of cores that exist on a computer. The parallelism structure of membrane computing is utilized, aside from communication rules. The number of cells in a membrane is equal to the number of cores in a computer (NumCores). Thus, each cell can be executed on different cores in a parallel manner:

- O includes all individuals in the cells and index of cells, that is, {0,1,…,m} (0 means environment)
- E = ϕ means no object in the initial state exists in the environment
- $w_p = \{I_1^p, I_2^p, ..., I_{n_p}^p\}$ where $p = 1,...,m$ in the initial state, I_q^p is the q^{th} individual in the p^{th} cell and each cell labeled with p contains n_p individuals as $I_q^p; q = 1,...,n_p$; p =1,..,m. These individuals evolve according to the operations of GA, that is, crossover and mutation in each membrane
- R is a finite set of communication and transformation rules and expressed as follows

The transformation rules x→y are identical to the concepts of mutation and crossover operation in GAs that evolve individuals. Transformation rules (including mutation and crossover) are executed on different cores for each membrane for determined times that set with user and named as maximum iteration of transformation rules (Max_Iter_Tran). Individuals resulting from transformation rules are transferred to the master core so that the communication rules will be applied on them and information will be exchanged between membranes.

The communication rules are (i,u/v,j) for i,j∈{1,2,…,m}, i≠j that is, different cells can exchange their individuals using these rules. u can be chosen from n_i individuals inside cell i (i.e., $u \in \{I_1^i, I_2^i, ..., I_{n_i}^i\}$) and exchange with v from cell j (i.e., $v \in \{I_1^j, I_2^j, ..., I_{n_j}^j\}$). **Figure 1** shows the communication direction for four membranes.

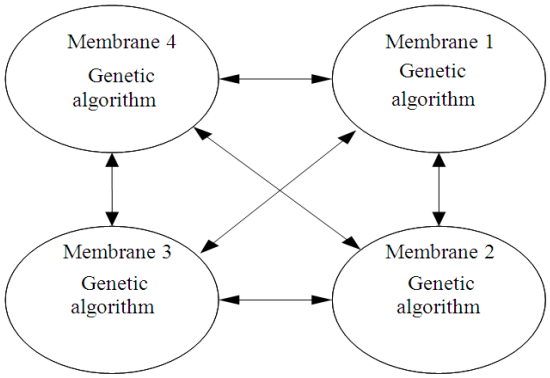

Fig. 1. Communication between four different membranes

For example, in rule $(1, I_7^1 / I_5^3, 3)$, the seventh individual in Cell 1 is exchanged with the fifth individual in Cell 3. By exchanging information between membranes, the diversity of individuals in each membrane increases and the trapping of the algorithm in a local minimum solution are avoided. After each separate execution of transformation rules in each membrane Max_Iter_Tran times, communication rules are executed on the master core, after which membranes exchange individuals. Afterwards, the transformation rules execute on a separate core for Max_Iter_Tran times. This process is repeated until the termination condition is reached.

i_{out} = 0 (0 indicates the environment of the output cell) at the end of computation. All membranes send their best individuals to the environment. The best individual in the environment is chosen as the solution.

The steps of the introduced membrane computing inspired genetic algorithm are as follows. Except for the communication step that runs on the master core in other steps, each membrane is executed on different cores in a parallel way.

Step1. Initialization

In this step, a membrane structure with m cells is created. m is equal to the number of existing cores in a computer, that is, m = NumCroses, because each membrane should be executed on different cores. n_p individuals are randomly generated as $\{I_1^p, I_2^p, ..., I_{n_p}^p\}$ in cell P, that is, I_q^p is the q^{th} individual in the P^{th} cell and p =1,..,m. This process is applied over different cores. Master cores pass the number of individuals n_p in cell p to the core p where p =1,…,m. Each core generates n_p individuals.

Step2. Evaluation

A fitness function is calculated for each individual in this step. Fitness functions differ based on the problem.

This step shows how near a solution is to the optimal solution and is executed on different cores for each cell.

Step3. Transformation

In each membrane, pairs of individuals are chosen based on their fitness and crossover rate. Individuals with better fitness have higher chances to crossover. If the crossover rate is equal to one, all individuals are chosen for crossover. Crossover takes two randomly chosen individuals (parents) as input and combines them to generate two children. The combination is performed by choosing a crossing point in the strings of the two parents and then exchanging the values. After crossover, mutation operation is used for individuals resulting from the crossover stage to prevent convergence of the algorithm with the local solution. The mutation operator chooses an individual and then changes the value of a randomly chosen gene in this individual. Different kinds of mutations and crossover operators can be defined based on the problem (Raad, 2011). This stage is performed on different cores for each membrane to exploit parallel structures of membrane computing and utilize multi-cores. The distribution of data over cores and gathering data from cores to master cores is a time-consuming process. Thus, this step is repeated several times on each core because the expected overhead time for sending data to the cores and receiving data from the cores with respect to the times data are processed on each core should be very low. This way, the advantages of multi cores are used. Results of each membrane in distinct cores are sorted according to fitness values. The results are then sent to the master core for communication information between membranes resulting from different cores.

Step 4. Communication

In this step, all of the individuals in the membranes are gathered from cores to the master core. According to the rules of the tissue-like P system, individuals are exchanged between membranes based on (i,u/v,j) for i, $j\in\{1,2,\ldots,m\}$, $i\neq j$ rules. This study defines $u = I^i_{l\times m+(j-i)+1}$ and $v = I^j_{l\times m+(m-(j-i))+1}$. We assume that the numbers of individuals (n_i) in all m membranes are the same, that is, ($n_1 = \ldots = n_i = \ldots = n_m$), j>i, $l = 0,1,\ldots,\frac{n_i}{m}$ and the numbers of individuals (n_i) are multiples of the number of membranes (m). The membranes exchange information by applying these rules. In this study, the information being exchanged are individuals containing the solution to the problem. After applying these rules, individuals inside membrane i change from $\{I^i_1, I^i_2, \ldots, I^i_{n_i}\}$ to the following individuals when i>1:

$$\begin{aligned}
&\{I^i_{0\times m+1}, I^{i+1}_{0\times m+2}, I^{i+2}_{0\times m+3}, \ldots, I^m_{0\times m+(m-i)+1},\\
&I^1_{0\times m+(m-i)+2}, I^2_{0\times m+(m-i)+3}, \ldots, I^{i-1}_{0\times m+m},\\
&\quad I^i_{1\times m+1}, I^{i+1}_{1\times m+2}, I^{i+2}_{1\times m+3}, \ldots, I^m_{1\times m+(m-i)+1},\\
&I^1_{1\times m+(m-i)+2}, I^2_{1\times m+(m-i)+3}, \ldots, I^{i-1}_{1\times m+m},\\
&\quad \vdots\\
&\quad I^i_{(\frac{n_i}{m}-1)\times m+1}, I^{i+1}_{(\frac{n_i}{m}-1)\times m+2}, I^{i+2}_{(\frac{n_i}{m}-1)\times m+3}, \ldots,\\
&I^m_{(\frac{n_i}{m}-1)\times m+(m-i)+1}, I^1_{(\frac{n_i}{m}-1)\times m+(m-i)+2},\\
&I^2_{(\frac{n_i}{m}-1)\times m+(m-i)+3}, \ldots, I^{i-1}_{(\frac{n_i}{m}-1)\times m+m}\}
\end{aligned}$$

When i =1, the individuals inside membrane i change to the following individuals:

$$\begin{aligned}
&\{I^i_{0\times m+1}, I^{i+1}_{0\times m+2}, I^{i+2}_{0\times m+3}, \ldots, I^m_{0\times m+(m-i)+1},\\
&\quad I^i_{1\times m+1}, I^{i+1}_{1\times m+2}, I^{i+2}_{1\times m+3}, \ldots, I^m_{1\times m+(m-i)+1}\\
&\quad \vdots\\
&\quad I^i_{(\frac{n_i}{m}-1)\times m+1}, I^{i+1}_{(\frac{n_i}{m}-1)\times m+2}, I^{i+2}_{(\frac{n_i}{m}-1)\times m+3}, \ldots, I^m_{(\frac{n_i}{m}-1)\times m+(m-i)+1}\}
\end{aligned}$$

For example, individuals for three membranes with six individuals inside each membrane is $\{I^i_1, I^i_2, \ldots, I^i_6\}; i = 1,2,3$ after applying the rules, as shown in **Fig. 2a**.

The results of the rules for each membrane are similar to that at the start of membrane i. The first individual of this membrane is selected as the new first individual in membrane i. afterwards, the membrane is changed according to **Fig. 2b** and a second individual of the next membrane is chosen as the new second individual in membrane i. Afterwards, the membrane is again changed and a third individual is chosen as a new third individual in membrane i. This process is repeated until all n_i new individuals are chosen for membrane i. This process is repeated for other membranes. As mentioned in Step 3, the materials are sorted according to their fitness values in each membrane before being transferred to the master core. Thus, after individuals are exchanged between membranes, each membrane includes individuals with good fitness and low fitness values from itself and other membranes, thereby increasing the diversity of individuals in each membrane and increasing the performance of the algorithm.

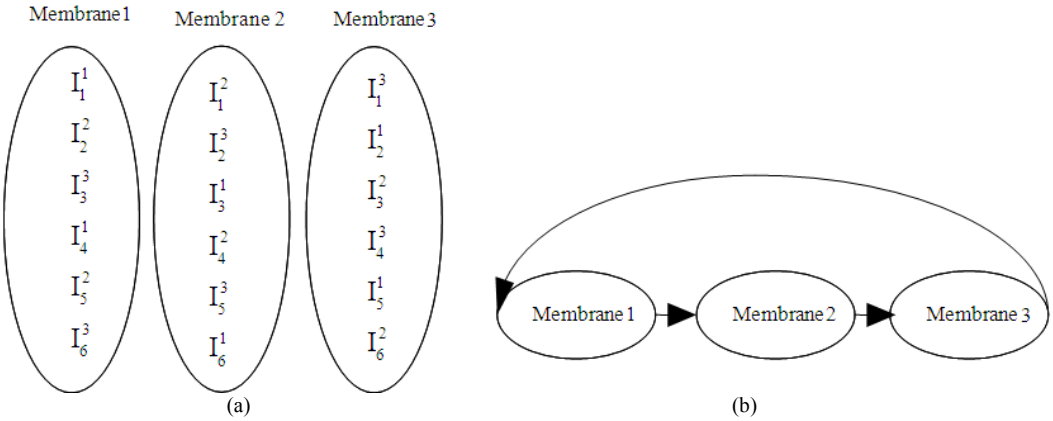

Fig. 2. Exchange of individuals: (a) exchange of individuals between three membranes; (b) order of choosing individuals from membranes

Table 1. Comparison of errors of solutions for GA and MCIGA in 500 runs

	GA	Proposed MCIGA	Speed up
Max error	4.49E-02	6.20E-03	7.20
Min error	1.04E-16	1.09E-16	-
Mean error	1.30E-03	2.10E-05	61.90
Standard deviation	4.30E-03	6.60E-04	6.50
Average time	1.6	1.09	1.46

2.4. Case Study

MCIGA is tested on familiar Colville minimization problem, such as benchmark. The Colville problem (Pires *et al*., 2010) is expressed as follows:

$$f(\bar{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 + 90(x_3^2 - x_4)^2 + (1 - x_3)^2 + 10.1((1 - x_2)^2 + (1 - x_4)^2) + 19.8(x_2 - 1)(x_4 - 1)$$

where, $-18 \le x_i \le 10$; i = 1, 2, 3, 4 with global solution (1,1,1,1) when f(1,1,1,1)=0.

2.5. Simulation Method

Simulations were implemented with Visual C++ software in a computer with Intel core i5 2.5 GHz (two cores) and 4 GB random access memory. In the simulations, blended crossover (Raad, 2011) is used as $[X_{min}+a(X_{max}- X_{min}),X_{max}-a (X_{max}-X_{min})]$, where a is a user-defined parameter; $X_{min} = \min(x_i^1, x_i^2)$; $X_{max} = \max(x_i^1, x_i^2)$ and x_i^1 and x_i^2 are the i^{th} gen of the first and second individuals chosen for the crossover, respectively. Mutation for this simulation is based on (Chen and Wang, 2011) where gen x_i is expressed as follows:

$$g_i^{new} = \begin{cases} x_i + \gamma(U_i - x_i) \text{ if } \rho = 0 \\ x_i - \gamma(x_i - L_i) \text{ if } \rho = 1 \end{cases}$$

where, ρ randomly choosen with a value of 1 or 0 with similar probabilities and U_i and L_i are the upper and lower boundaries of gene x_i, respectively. γ decreases when $\gamma = 1\text{-}b^{(1\text{-Iter/Iter}_{max})C}$, where $iter_{max}$ is the total number of iterations, iter is the iteration for this step, b is a random variable from [0, 1] and C is the user defined variable. In the following simulation, the constant for crossover is a = 0.5, the crossover rate is 1, the mutation rate is 0.7, the constant C for mutation is 3, the maximum iteration number $iter_{max}$ = 1000, the number of membrane for the MCIGA is equal to the number of cores on our computer (i.e., 2) and the numbers of individuals for the GA and MCIGA are both 300. Each membrane has 150 individuals. These parameters are obtained via experimental trials.

3. RESULTS

Results of simulation for proposed algorithms and GA have been shown in **Table 1 and Fig. 3**. In **Table 1** times in seconds and maximum, minimum, mean and deviation of error from global solution have been illustrated. These amount have been obtained from 500 simulation runs and the maximum iteration number for each run is $iter_{max}$ = 1000. In **Fig. 3** the error of solution in each run for 500 runs has been shown for both proposed algorithm and GA.

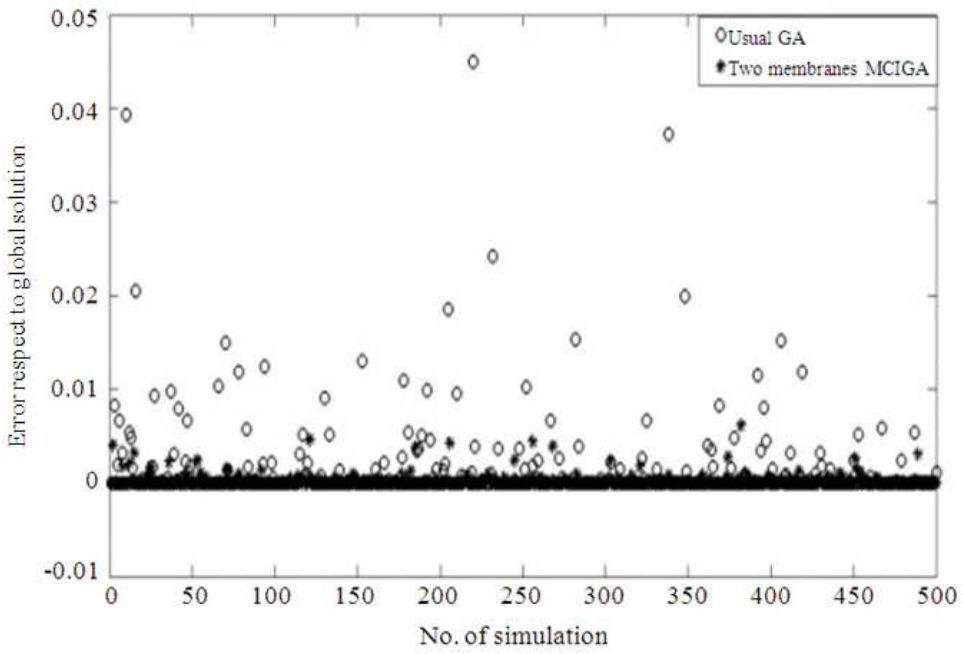

Fig. 3. Comparison of errors of solutions between GA and MCIGA for 500 runs

4. DISCUSSION

The results demonstrate the advantages of the proposed approach of membrane computing inspired inteligent algorithm. This illustrates that the MCIGA has enough diversity to avoid local solutions and has enough parallelism to run on different cores. As illustrated in **Table 1**, the elapsed time of MCIGA is less than GA because MCIGA can execute on different cores in a parallel manner. Furthermore, the standard deviation of errors of solutions for 500 runs using MCIGA is 6.5 times lower and the mean error of the solution 61.9 times better than that obtained using GA. These enhancement caused by exchanging information between membranes. As shown in **Fig. 3**, in some runs, the error of the solutions for GA is higher than that of MCIGA. Because in the algorithm of MCIGA there are information (individuls) exchanged among membranes, thus, the diversity of the proposed algorithm increases and prevents the proposed algorithm to trap in local solutions as errors decreases.

5. CONCLUSION

A new membrane computing-inspired algorithm was introduced for solving optimization problems. This algorithm outperformed a previous algorithm because, unlike the latter, the former used parallel structures. Previous studies used communication rules in membrane computing to improve their algorithms. Aside from using communication rules, this study uses parallel structures in membrane computing by defining appropriate membrane-inspired algorithm and using multi-cores. The speed of the algorithm increases by executing each membrane on different cores. The diversity of the algorithm increases and the algorithm avoids being trapped in the local minimum answer by using communication rules to exchange information between membranes. Thus, the execution times and mean error are decreased.

The idea of using parallel structures in membrane computing for GA is introduced for the first time in this study. The proposed algorithm can also parallelize other intelligent algorithms. For our future work, we will extend the application of this proposed algorithm to manufacturing optimization problems and then compare it with several other algorithms.

6. ACKNOWLEDGMENT

This study supported by the Young Researcher Grant of the National University of Malaysia (Grant code: GGPM-2011-051).

7. REFERENCES

Adrian, T. and I. Florentin, 2012. Computational properties of two p systems solving the 3-colouring problem. Proceedings of the 14th International Symposium, Symbolic and Numeric Algorithms for Scientific Computing, Sept. 26-29, IEEE Xplore Press, Timisoara, Romania, pp: 62-69. DOI: 10.1109/SYNASC.2012.61

Affenzeller, M., S. Winkler, S. Wagner and A. Beham, 2009. Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. 1st Edn., Taylor and Francis, Boca Raton, ISBN-10: 1420011324, pp: 379.

Buiu, C., C.I. Vasile and O. Arsene, 2012. Development of membrane controllers for mobile Robots. Inform. Sci., 187: 22-51. DOI: 10.1016/j.ins.2011.10.007

Cardona, M., M.A. Colomer, M.J. Perez-Jimenez, D. Sanuy and A. Margalida, 2009. Modeling ecosystems using p systems: The bearded vulture, a case study. Comput. Sci., 5391: 95-116. DOI: 10.1007/978-3-540-95885-7_11

Chen, Z.Q. and R.L. Wang, 2011. Two efficient real-coded genetic algorithms for real parameter optimization. Int. J. Innov. Comput. Inform. Control, 7: 4871-4884.

Cheng, J.X., G.X. Zhang and X.X. Zeng, 2011. A novel membrane algorithm based on differential evolution for numerical optimization. Int. J. Unconventional Comput., 7: 159-183.

Christinal, H.A., D. Diaz-Pernil and P. Real, 2011. Region-based segmentation of 2D and 3D images with tissue-like P systems. Patt. Recogn. Lett., 32: 2206-2212. DOI: 10.1016/j.patrec.2011.05.004

Christinal, H.A., D. Diaz-Pernil, M.A. Gutierrez-Naranjo and M.J. Perez-Jimenez, 2010. Thresholding of 2D images with cell-like P systems. Romanian J. Inform. Sci. Technol., 13: 131-140.

Gutierrez-Naranjo, M.A. and M.J. Perez-Jimenez, 2011. Local search with p systems: A case study. Int. J. Natural Comput. Res., 2: 47-55. DOI: 10.4018/jncr.2011040104

Ishdorj, T., A. Leporati, L. Pan, X. Zeng and X. Zhang, 2010. Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. Theor. Comput. Sci., 25: 2345-2358. DOI: 10.1016/j.tcs.2010.01.019

Muniyandi, R. and M.Z. Abdullah, 2012. Modeling hormone-induced calcium oscillations in liver cell with membrane computing. Romanian J. Inform. Sci. Technol., 15: 63-76.

Paun, G., G. Rozenberg and A. Salomaa, 2010. Membrane Computing. 1st Edn., Oxford University Press, Oxford, ISBN-10: 0199556679, pp: 672.

Pena-Cantillana, F., D. Diaz-Pernil, A. Berciano and M. A. Gutierrez-Naranjo, 2011. A parallel implementation of the thresholding problem by using tissue-like P systems. Comput. Anal. Images Patt., 6855: 277-284. DOI: 10.1007/978-3-642-23678-5_32

Pires, E.S., J.T. Machado, P.B.D.M. Oliveira, J.B. Cunha and L. Mendes, 2010. Particle swarm optimization with fractional-order velocity. Nonlinear Dynamics, 61: 295-301. DOI: 10.1007/s11071-009-9649-y

Raad, D.N., 2011. Multi-objective optimisation of water distribution systems design using metaheuristics. PhD Thesis, University of Stellenbosch, Stellenbosch.

Twycross, J., L.R. Band, M.J. Bennett, J.R. King and N. Krasnogor, 2010. Stochastic and deterministic multiscale models for systems biology: An auxin-transport case study. BMC Syst. Biol., 4: 34-45. DOI: 10.1186/1752-0509-4-34

Zhang, G., F. Zhou, X. Huang, J. Cheng and M. Gheorghe *et al*., 2012b. A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems. J. Universal Comput. Sci., 18: 1821-1841. DOI: 10.3217/jucs-018-13-1821

Zhang, G., J. Cheng and M. Gheorghe, 2011. A membrane-inspired approximate algorithm for traveling salesman problems. Romanian J. Inform. Sci. Technol., 14: 3-19.

Zhang, G., J. Cheng, M. Gheorghe and Q. Meng, 2012a. A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. J. Applied Soft Comput., 13: 1528-1542. DOI: 10.1016/j.asoc.2012.05.032