An Improved Version of opt-aiNet Algorithm (I-opt-aiNet) for Function Optimization

Hamdy N. Agiza[†], **Ahmed E. Hassan**[††], **Ahmed M. Salah**[†††]
agizah@mans.edu.eg arwaahmed1@gmail.com a_salah@mans.edu.eg

† Mathematics Department, Faculty of Science, Mansoura University, Egypt
†† Electrical Engineering Department, Faculty of Engineering, Mansoura University, Egypt
††† Mathematics Department, Statistics and Computer Science Branch, Faculty of Science, Mansoura University, Egypt

Abstract
This paper presents an improved version of opt-aiNet, which is an algorithm for multimodal function optimization based on the natural immune system metaphor. The proposed algorithm has some major and minor changes on the way the clonal selection principle is applied within the original opt-aiNet algorithm which allows for fast localization of the optima. The output of the proposed algorithm is tested on the same data as the original opt-aiNet and the results show the validity of the new improved one.

Key words:
Artificial Immune Systems, Clonal Selection, opt-aiNet, Function Optimization

1. Introduction

Artificial Immune Systems (AIS) are computational systems inspired by theoretical immunology, observed immune functions, principles and mechanisms in order to solve problems [1]. The fundamental features of the natural immune system, like distribution, adaptability, learning from experience, complexity, and coordination have decided that immune algorithms have been applied to a wide variety of tasks, including optimization.
The task of natural immune system is to identify and destroy foreign invaders or antigens. The basic elements of natural immune system are immune cells (such as B cells, T cells, and other lymphocytes), B-cells produce antibodies, which bind to the invading antigens and help destroy them. Each B-cell produces only one kind of antigenic receptors. When an antigen enters the body, it activates only the lymphocytes whose receptors can bind to it. Activated by an antigen and with a second signal from accessory cells, such as the T-cells, the B-cells proliferates (divides) producing large number of clones. In the final stage these clones can mutate in order to produce antibodies with very high affinity to a specific antigen [2].
This process is explained by the clonal selection principle [3], according to which only those cells that recognize the antigens are selected to proliferate. The selected cells are subject to affinity maturation, which improves their affinity to the antigens.

Manuscript received March 5, 2011
Manuscript revised March 20, 2011

This paper reviews work done in optimization using AIS and briefly introduce immune optimization algorithm (opt-aiNet). An improved version of opt-aiNet is presented with detailed description of the modifications made together with a theoretical comparison to original opt-aiNet algorithm followed by the results of experimental comparison of the proposed and the original one. The paper ends with conclusion of the proposed work.

2. Optimization using Artificial Immune Systems

There is a natural parallel between the immune system and optimization. Whilst the immune system is not specifically an optimizer, the process of the production of antibodies in response to an antigen is evolutionary in nature; hence the comparison with optimization, the location of better solutions. The process of clonal selection (a theory widely held by many immunologists [4]) describes how the production of antibodies occurs in response to an antigen, and also explains how a memory of past infections is maintained. This process of clonal selection has proved to be a source of inspiration of many people in AIS and there have been a number of algorithms developed for optimization inspired by this process [5].

Opt-aiNet, proposed in [1], is an optimization version of aiNet which is a discrete immune network algorithm that was developed for data compression and clustering [6]. AiNet and has subsequently been developed further and applied to areas such as bioinformatics [7] and even modeling of simple immune responses [8]. Opt-aiNet evolves a population that consists of a network of antibodies (considered as candidate solutions to the function being optimized). These undergo a process of evaluation against the objective function, clonal expansion, mutation, selection and interaction between themselves. Opt-aiNet creates a memory set of antibodies that represent (over time) the best candidate solutions to the objective function. Opt-aiNet is capable of either unimodal or multimodal optimization and it has defined stopping criteria.

In order to keep track of local optima as well as global optima, opt-aiNet has to select all cells for cloning. However, according to the clonal selection theory of acquired immunity presented in [3], only a set of cells are selected to be cloned.

Another thing to note about opt-aiNet is the number of clones per cell in the cloning process. Opt-aiNet gives all cells the same number of clones for each iteration as published in [1]. That's because opt-aiNet aims to preserve local optima's so it needs to explore the whole space. Doing so contradicts again with the clonal selection theory of acquired immunity. As [5, 9] stated that the proliferation rate of each immune cell is proportional to its affinity with the selective antigen (higher the relative affinity, the more progeny). It is obvious that opt-aiNet had to sacrifice many of clonal selection theory ideas for the benefit of maintaining local optima's.

Also, the stopping criterion adopted for opt-aiNet is based upon the size of the memory population. After network suppression, a fixed number of cells remain. If this number does not vary from one suppression to another, then the network is said to have stabilized and the remaining cells are memory cells corresponding to the solutions of the problem [1]. This stopping criterion does not necessarily correspond to algorithm convergence since it is prone to bottlenecks.

To overcome all the previous shortages of opt-aiNet algorithm, an improved version of opt-aiNet algorithm is proposed and demonstrated through the rest of the paper.

3. The proposed algorithm (I-opt-aiNet)

As discussed earlier, opt-aiNet never pretends to be fast in terms of number of function evaluations [2]. Its strategy concentrates on finding as well as maintaining all the global and local optima and if applicable determining their number.

An improved version of opt-aiNet algorithm (I-opt-aiNet) is introduced to modify some shortages of opt-aiNet algorithm in order to achieve better performance. The modifications made includes the number of selected cells each iteration and how to select them, the number of cells that could be removed and number of cells that could be added, which cells may be cloned and how many clones would be created, optimizing evaluation processes, and replacing one stopping condition with a more effective one.

The most general view of the I-opt-aiNet while searching for function minimum is shown in Figure 1.

```
1. Initialization: Generate a set N of candidate solutions.
2. While a stopping criterion is not met do
     a. Evaluation (phase one): Evaluate all unevaluated
        cells and mark them as "EVALUATED".
     b. Selection (phase one): Select all cells with affinity
        higher than Best Fitness Average and remover the
        rest. The Best Fitness Average is the best population
        fitness average over all iterations including the
        current.
     c. Cloning: Create a number of clones to each cell
        proportional to the cell affinity
     d. Mutation: Mutate each clone inversely proportional
        to its fitness
     e. Evaluation (phase two): Evaluate all mutated clones
        and mark them as "EVALUATED"
     f. Selection (phase two): Select the best clones with
        respect to Best Fitness Average and add them to the
        population
     g. Network interaction: For each antibody in the
        population determine the similarity with other
        antibodies in the population. From any pair of
        antibodies whose affinity is less than a specified
        threshold, eliminate the worse element
     h. Diversity: Add new cells to the population as much
        as removed cells from step 2.c
3. End While
```

Figure 1: Pseudo code of the proposed algorithm I-opt-aiNet

A complementing diagrammatic representation of the tasks and workflow of the I-opt-aiNet algorithm is also provided in Figure 2.

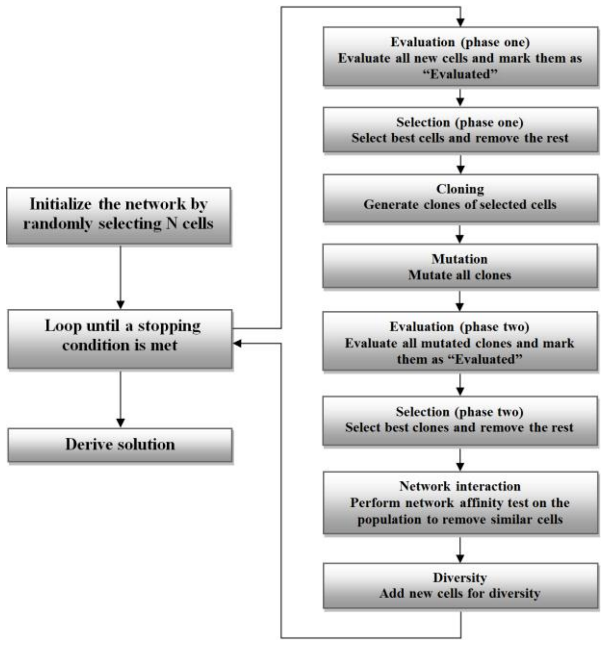
Initialize the network by randomly selecting N cells
Loop until a stopping condition is met
Derive solution
Evaluation (phase one) Evaluate all new cells and mark them as "Evaluated"
Selection (phase one) Select best cells and remove the rest
Cloning Generate clones of selected cells
Mutation Mutate all clones
Evaluation (phase two) Evaluate all mutated clones and mark them as "Evaluated"
Selection (phase two) Select best clones and remove the rest
Network interaction Perform network affinity test on the population to remove similar cells
Diversity Add new cells for diversity

Figure 2: Flowchart of I-opt-aiNet

The behavior of the new algorithm can be explained in detailed steps as follows next.

- Initialization

A population is initialized randomly. The number of generated cells is a parameter of the algorithm, chosen experimentally to help achieving high quality solutions in less time. For simple objective functions we can use small values for this parameter yet solution quality is still maintained. For more complex problems there will be a need to raise the value of the parameter a bit to ensure enough space for the algorithm to converge.

- The main loop

After initializing the population, the algorithm starts the main loop in which cells are going to undergo clonal selection until a stopping condition is satisfied. I-opt-aiNet has two stopping conditions defined to control the algorithm flow. The first is the maximum number of iterations to be used. This parameter is inherited from opt-aiNet without a change. The second stopping condition is a new one introduced to aid the algorithm to converge in less time. This stopping condition terminates the run if a cell remains the best solution for a specified number of iterations, a new parameter, claiming this cell to be the optimum solution to the proposed objective function. For this we chose to remove one of opt-aiNet stopping conditions that depend on comparing the average of two consecutive iterations until there's no significant difference.

- Evaluation (Phase one)

At the beginning of each iteration, all new cells are exposed to the objective function. I-opt-aiNet adds further step here; that all exposed cells are marked as "EVALUATED". The marks on the cells prevents the algorithm from evaluating them anymore in their life time which means that the total number of evaluation would be lesser and the algorithm converges faster.

- Selection (Phase one)

A number of cells are then selected to be cloned. I-opt-aiNet uses a unique approach for selecting cells. We introduce a new term called Best Fitness Average (BFA). This term, as it sounds, represents the best fitness average over all iterations preceding the current one. BFA has a dynamic nature by design. At the first iteration, BFA is initialized with the fitness average of all the cells in the population. For all iterations afterwards, the current population fitness average is calculated and if it was less than BFA, then BFA will be updated. Otherwise, it keeps its value without any change.

After calculating the BFA, every cell with fitness less than or equal to the BFA are selected for the next phase. The remaining cells are then killed and removed from the population but we keep track of their count.
It is worth to note about I-opt-aiNet that the number of cells removed each iteration is dynamic and depends indirectly on the value of BFA. While the original opt-aiNet uses a fixed parameter to remove a percentage from the population [1].

- Cloning

In our implementation, the number of clones generated for any selected cell is proportional to the cell's fitness value. And there's a new parameter introduced to set the maximum number of clones to any cell. Here's how the process goes, the number of clones that should be created for a candidate cell c_i is given by: $N_{c_i} = f_i^*.\mu$ Where's:

  - μ is the maximum number of clones that any cell could have (fixed parameter).
  - f_i^* is the fitness norm to cell number i, and is calculated by : $f_i^* = \frac{f_{max}-f_i}{f_{max}-f_{min}}$ which is obviously a number in the range $[0,1]$

For example, let us have a minimization problem with 5 cells that have the finesses: $f_1 = 0, f_2 = 12, f_3 = 23, f_4 = 37$ and $f_5 = 50$

Hence, $f_{min} = 0$ and $f_{max} = 50$ and this leads to $f_1^* = 1, f_2^* = 0.76, f_3^* = 0.54, f_4^* = 0.26$ and $f_5^* = 0$.

Now for different values of μ we find that:

$\boldsymbol{\mu}$	Number of clones for cells				
10	$N_{c_1} = 10$	$N_{c_2} = 8$	$N_{c_3} = 6$	$N_{c_4} = 3$	$N_{c_5} = 0$
20	$N_{c_1} = 20$	$N_{c_2} = 15,$	$N_{c_3} = 11$	$N_{c_4} = 5$	$N_{c_5} = 0$

It's clear that I-opt-aiNet is biased towards cell's fitness's values at the cloning process. Because despite that the first cell has the least fitness, it still gets much more clones than the other cells.

- Mutation

All clones then undergo somatic mutation so that they become variations of their parent. The affinity proportional mutation works in the same way as in [1]. It is performed according to the following expression:

$$c' = c + \alpha \cdot N(0,1),$$
$$\alpha = (1/\beta)e^{-f^*},$$

Where c' is a mutated cell c, $N(0,1)$ is a Gaussian random variable of zero mean and standard deviation $\sigma = 1$, β is a parameter that controls the decay of the inverse exponential function, and f^* is the fitness of an individual normalized in the interval $[0,1]$. A mutation is only accepted if the mutated cell c' is within its range of domain.

- Evaluation (Phase two)

All newly created mutated clones needs to be evaluated. The evaluation process goes the same way as the first evaluation phase. Those clones get evaluated against the objective function and are then marked as "EVALUATED".

- Selection (Phase two)

A subset of the mutated clones is selected to be added to the population and the rest is removed from the cloning pool. The selection is made using the same method discussed in the first selection phase with the help of BFA. That is, all mutated clones with fitness less than BFA of the current iteration are kept in the pool, while the rest are removed.

This selection phase is necessary to ensure that no bad cells exist in the population so the algorithm converges faster.

- Network interaction

Next the algorithm determines the affinity of all cells in the network. Suppress all but the highest fitness of those cells whose affinities are less than the suppression threshold (a parameter) and determine the number of network cells, named memory cells, after suppression.

- Diversity

At the end of any iteration, new cells are generated randomly to be added to the population in the next iteration. The number of cells to be generated is the same number of cells removed in parent cell's selection stage.
It should be noticed that the number of removed cells (or added cells) in I-opt-aiNet is not fixed and varies dynamically throughout the iterations.

4. Validating the performance of I-opt-aiNet

The new variant of opt-aiNet proposed in Section 4, I-opt-aiNet, was implemented by modifying the opt-aiNet code kindly obtained from [10] which formed part of [11]. The implementation had been experimented against the opt-aiNet with two metrics were tested. These were the number of evaluations taken to obtain a solution and the quality of the solution obtained, which is a standard benchmark criteria. Table 1 contains the details of suitable number of functions (ranging in dimensions) that were selected from an established literature on which to perform these tests. In addition, the parameters for I-opt-aiNet have been selected experimentally and the values presented for the original opt-aiNet were empirically the best performing parameters for these functions and all the parameters values can be reviewed in Table 2. The reader is referred to [1] for further analysis of the sensitivity of original opt-aiNet parameters.

Table 1: Functions Employed For Experimentation

Name	Function
F1	$f(x) = 2(x-0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125$ Where, $0 \le x \le 1$
F3	$f(x) = -\sum_{j=1}^{5} j \sin[(j+1)x + j]$ Where, $-10 \le x \le 10$
Branin	$f(x,y) = a(y - bx^2 + cx - d)^2 + h(1-f)\cos(x) + h$ Where, $a = 1, b = 5.1/4\pi^2, c = 5/\pi,$ $d = 6, h = 10, f = 1/8\pi$ $-5 \le x \le 10, and\ 0 \le y \le 15$
Pshubert 1	$f(x,y) = \{\sum_{i=1}^{5} i\cos[(i+1)x + i]\} \times \{\sum_{i=1}^{5} i\cos[(i+1)y + i]\} + \beta[(x+1.42513)^2 + (y+0.80032)^2]$ Where, $\beta = 0.5, -10 \le x \le 10,$ $and\ -10 \le y \le 10$
Pshubert 2	$f(x,y) = \{\sum_{i=1}^{5} i\cos[(i+1)x + i]\} \times \{\sum_{i=1}^{5} i\cos[(i+1)y + i]\} + \beta[(x+1.42513)^2 + (y+0.80032)^2]$ Where, $\beta = 1, -10 \le x \le 10,$ $and\ -10 \le y \le 10$
Quartic	$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}$ Where, $-10 \le x \le 10\ and -10 \le y \le 10$
Shubert	$f(x,y) = \{\sum_{i=1}^{5} i\cos[(i+1)x + i]\} \times \{\sum_{i=1}^{5} i\cos[(i+1)y + i]\}$ Where, $-10 \le x \le 10\ and\ -10 \le y \le 10$

Table 2: Algorithm Parameters

Algorithm	Parameter	Value
I-opt-aiNet	Initial population size	In {10, 100}
	Maximum number of clones	10
	Stable state limit	100
	Suppression threshold	0.2
	Scale of affinity proportion selection	100
Opt-aiNet	Initial population size	20
	Suppression threshold	0.2
	Number of clones generated	10
	Percentage of random new cells each	40%
	Scale of affinity proportion selection	100

The new algorithm was then applied to the selected functions to compare it with the results obtained for applying the original opt-aiNet on the same functions published in [12]. Tables 3.a and 3.b provides a set of results averaged over 50 runs for the functions being optimized. I-opt-aiNet was executed until either the minimum value was found, or 500 iterations had passed just as in [12]. All results reported presented with standard deviations where greater than zero. The columns in Table 3.a represent (in order from left to right): the objective function, the minimum possible value for that function, the next two columns show the value located by each algorithm for that function. Table 3.b shows the average and the standard deviation of the number of evaluations taken to achieve that minimum result. The number of evaluations is the cumulative value for the number of times an individual in the population called the evaluate function method.

Table 3.a: Optimum value averaged over 50 runs for the functions being optimized

Function	Actual optimum	opt-aiNet	I-opt-aiNet
F1	-1.12	-1.12	-1.12
F3	-12.03	-12.03	-12.03
Branin	0.40	0.39	0.39
Pshubert 1	-186.73	-180.83	-186.72
Pshubert 2	-186.73	-173.16	-186.71
Quartic	-0.35	-0.26	-0.35
Shubert	-186.73	-186.73	-186.73

Table 3.b: Average and standard deviations of the number of evaluations over 50 runs for the functions being optimized

Function	opt-aiNet	I-opt-aiNet
F1	6717 ±.538	388.7 ± 106.5
F3	41419 ± 25594	973.74 ± 282.82
Branin	6346 ± 4656	2275.6 ± 422.61
Pshubert 1	363528 ± 248161	25331.3 ± 9264
Pshubert 2	346330 ± 255980	25762.3 ± 4574.1
Quartic	54703 ± 29701	2658.6 ± 451.2
Shubert	50875 ± 45530	3270 ± 887.9

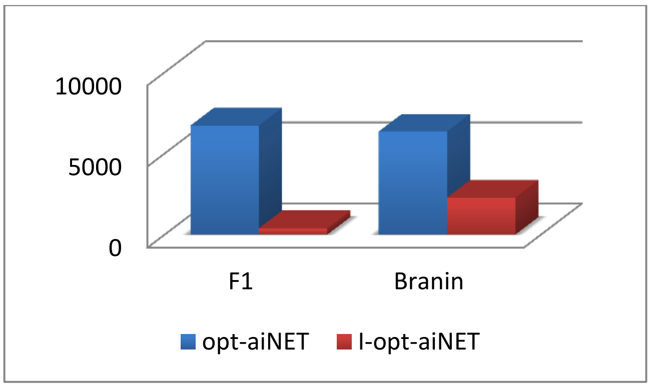

Figure 3: Average number of evaluations for functions F1 and Branin

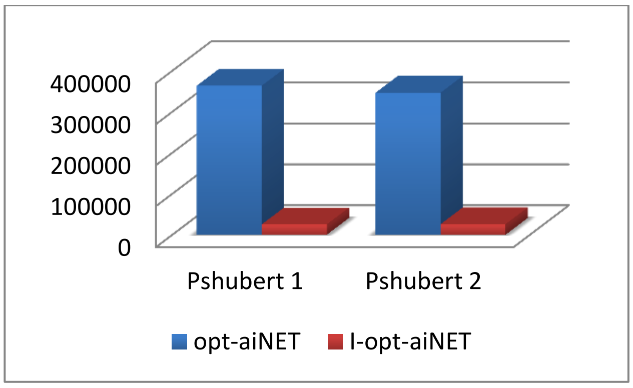

Figure 4: Average number of evaluations for functions Pshubert 1 and Pshubert 2

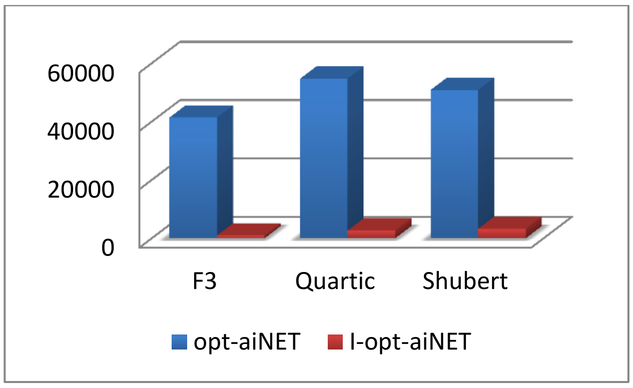

Figure 5: Average number of evaluations for functions F3, Quartic, and Shubert

Looking at the data provided in Tables 3.a, 3.b and Figures 3, 4, and 5 above; both the algorithms performed well at finding optimal solutions for all of functions presented. Therefore, in terms of the metric for quality of solutions there seems little to distinguish the two algorithms. However, when the number of evaluations is taken into account, there is a significant difference in the number of evaluations taken to obtain the solution. That significant difference is a strong evident on the less time needed by I-opt-aiNet to converge to global optimum. Also, the standard deviation of the number of evaluations was very small for I-opt-aiNet in comparing with opt-aiNet. This is a good indicator on how stable is the solutions in case of using I-opt-aiNet. Using these statistics we can see that I-opt-aiNet outperformed the original algorithm noticeably.

Conclusion

This paper has presented I-opt-aiNet algorithm, which is an improved version of the well-known immune algorithm opt-aiNet for solving optimization problem for multimodal functions. The improved version of opt-aiNet has introduced new method for selecting cells, cloning cells, adding cells for diversity and replaced a stopping condition for a smarter one. It was theoretically and experimentally compared with opt-aiNet. The performance was illustrated for seven functions. The algorithm demonstrated to be capable of finding global optima with much smaller number of function evaluations, comparing to the original algorithm. The promising results of this version of the algorithm obtained for multimodal function optimization encourage using the algorithm for other optimization or even dynamic optimization problems.

References

[1] De Castro, L.N and Timmis, J. (2002). An Artificial Immune Network for Multimodal Function Optimization. Proc. Of IEEE World Congress on Evolutionary Computation. Pp. 669-674.

[2] Malgorzata Lucinska and Slawomir T. Wierzchon. Hybrid Immune Algorithm for Multimodal Function Optimization. Recent Advances in Intelligent Information Systems ISBN 978-83-60434-59-8, pages 301-313.

[3] Burnet, F. M. (1959). The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge.

[4] D. R. FORSDYKE (1995). The Origins of the Clonal Selection Theory of Immunity. FASEB. Journal 1995, VOL 9, 164-166.

[5] Jason Brownlee (2007). Clonal Selection Algorithms, Technical Report. Victoria, Australia: Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology; Technical Report ID: 070209A.

[6] De Castro, L.N., Von Zuben, F.J. (2001). aiNet: An Artificial Immune Network for Data Analysis, (full version, pre-print), Book Chapter in Data Mining: A Heuristic Approach, H. A. Abbass, R. A. Sarker, and C. S. Newton (eds.), Idea Group Publishing (2001), USA, Chapter XII, pp. 231-259.

[7] A. Secker, M.N. Davies, A.A. Freitas, J. Timmis, E. Clark, D.R. Flower (2008). An Artificial Immune System for Evolving Amino Acid Clusters Tailored to Protein Function Prediction. Lecture Notes in Computer Science, Volume 5132/2008, 242-253, DOI: 10.1007/978-3-540-85072-4_22.

[8] Jerne, N (1975). Towards a Network theory for the Immune System. Annals of Immunology, Inst. Pasture.

[9] Leandro N. de Castro and Jon Timmis. Artificial Immune Systems: A new computational intelligence approach, Great Britain: Springer-Verlag

[10] Andrews, P. (2007). Opt-aiNet source code in Java, last modified October 2005.

[11] Andrews, P. S., & Timmis, J. (2005). On Diversity and Artificial Immune Systems: Incorporating a Diversity Operator into aiNet. International Workshop on Natural and Artificial Immune Systems (NAIS), Vietri sul Mare, Salerno, Italy. Lecture Notes in Computer Science 391. pp. 293-306.

[12] Jon Timmis, J., Edmonds, C., Kelsey, J. (2004). Assessing the performance of two immune inspired algorithms and a hybrid genetic algorithm for Function Optimization. In: Evolutionary Computation, CEC2004.

Hamdy N. Agiza is a professor of applied mathematics at Mansoura University, Egypt. He obtained his B.Sc. and M.Sc. both in Pure Mathematics from Mansoura University, Faculty of Science, Egypt in 1976 and 1980 respectively. He obtained his Ph.D. degree in applied Mathematics from Heriot-Watt University, UK, in 1987. He was a member of International Center of Theoretical Physics (ICTP) 1999-2005 as Senior Associate Group and a member of Egyptian Mathematical Society since 1991 till now. Prof. Agiza is currently the manager of Quality Assurance unit in faculty of science, Mansoura University.

Ahmed E. Hassan is an associate professor of computer engineering at Mansoura University, Egypt. He received a PhD in computer engineering from West Virginia University, West Virginia USA. He received a Master in computer engineering from Stevens Institute of Technology NJ, USA. Also he got Master of Artificial Intelligence applications from Mansoura University, Egypt. Ahmed E. Hassan is IEEE Member since 1998, IEEE Computer Society since 1999, IEEE Education society since 2002, IEEE Computational Intelligence Society since 2002, ACM Member since 2002 and ACM Education Society since 2002. He is a consultant manager of EgyTronic Co. www.egytronic.com and he is also the consultant manager of the "CadTronic" Co. "ww.cadtronic.com". From 2006 until now he is the manager of the technical office of Mansoura University Quality Assurance Center.

Ahmed M. Salah is an assistant lecturer of computer science at Mansoura University, Egypt. He has obtained the B.Sc. in Statistics and Computer Science from Mansoura University at Faculty of Science, Egypt in 2005.