The Convergence Speed of Single- And Multi-Objective Immune Algorithm Based Optimization Problems

Mohammed Abo-Zahhad zahhad@yahoo.com
Faculty of Engineering,
Electrical and Electronics Engineering Department,
Assiut University,
Assiut, 71516, Egypt.

Sabah M. Ahmed sabahma@yahoo.com
Faculty of Engineering,
Electrical and Electronics Engineering Department,
Assiut University,
Assiut, 71516, Egypt.

Nabil Sabor nabil_sabor@yahoo.com
Faculty of Engineering,
Electrical and Electronics Engineering Department,
Assiut University,
Assiut, 71516, Egypt.

Ahmad F. Al-Ajlouni alajlouna@hotmail.com
Hijjawi Faculty for Engineering Technology,
Communication Engineering Department,
Yarmouk University,
Irbid, 21163, Jordan.

Abstract

Despite the considerable amount of research related to immune algorithms and it applications in numerical optimization, digital filters design, and data mining, there is still little work related to issues as important as sensitivity analysis, [1]-[4]. Other aspects, such as convergence speed and parameters adaptation, have been practically disregarded in the current specialized literature [7]-[8]. The convergence speed of the immune algorithm heavily depends on its main control parameters: population size, replication rate, mutation rate, clonal rate and hyper-mutation rate. In this paper we investigate the effect of control parameters variation on the convergence speed for single- and multi-objective optimization problems. Three examples are devoted for this purpose; namely the design of 2-D recursive digital filter, minimization of simple function, and banana function. The effect of each parameter on the convergence speed of the IA is studied considering the other parameters with fixed values and taking the average of 100 times independent runs. Then, the concluded rules are applied on some examples introduced in [2] and [3]. Computational results show how to select the immune algorithm parameters to speedup the algorithm convergence and to obtain the optimal solution.

Keywords: Immune Algorithm, Convergence, Mutation, Hypermutation, Population Size, Clonal Selection.

1. INTRODUCTION

The parameters of the immune algorithm have a large effect on the convergence speed. These parameters are the population size (p_s) which estimates the number of individuals (antibodies) for each generation, the mutation rate (p_m) which increases the diversity in population, and the replication rate (p_r) which estimates the number of antibodies chosen from the antibody population pool to join the algorithm operations. Other parameters such as the clonal rate (p_c) which estimates the number of individuals chosen from the antibody population pool to join the clonal proliferation (selection), as well as the hypermutation rate (p_h) which improves the capabilities of exploration and exploitation in population, have also great effect on the speed of convergence. In spite of the research carried out up to date, there are no general rules on how these parameters can be selected. In literature [1]-[2] and [13], the immune parameters are selected by certain values (e.g. p_s =200, p_r =0.8, p_m =0.1, p_c =0.06, p_h =0.8) without stating the reason for this selection.

In this paper we investigate the effect of parameters variation on the convergence speed of the immune algorithms developed for three different illustrative examples: 2-D recursive digital filter design (multi-objective problem), minimization of simple function (single-objective problem), and finding the global minimum of banana function. The obtained results can be used for selecting the values of these parameters for other problems to speed up the convergence. The paper is organized as follows. Section 2 describes the immune algorithm behavior. In Section 3 three illustrative examples are given to investigate the effect of parameters variation on the convergence speed of the immune algorithm. Section 4 discusses the selection criteria of these parameters to guarantee the convergence speed. In section 5, some examples introduced in [3] and [12] are considered to demonstrate the effectiveness of the selection of immune algorithm control parameters. And finally, Section 6 offers some conclusions.

2. IMMUNE ALGORITHMS BEHAVIOR

Immune algorithms are randomized algorithms inspired by immune functions and principles observed in nature [10]. Such algorithms begin by generating population pool (chromosome) using real coding representation and evaluating the objective values. Then, the population pool undergoes the algorithm operations which will be described in this section. The operations are repeated at each generation (gen) until the termination condition is satisfied [1]-[2]. Table (1) illustrates the main steps of the immune algorithm [16].

2.1 Generation of Antibody Population

The antibody population is generated either by using binary coding representation or real coding representation. In the binary coding representation, each variable is encoded as a binary string and the resulting strings are concatenated to form single chromosome (antibody) [11]. However, in the real coding representation, each antibody is encoded as a vector of floating point numbers, with the same length as the vector of decision variables. This representation is accurate and efficient because it is closest to the real design space, and the string length represents the number of design variables.

2.2 Selection for Reproduction

The roulette wheel selection is employed in immune bases algorithms for chromosomes reproduction. Its basic idea is to determine the selection probability for each solution in proportion with the fitness value. For solution j with fitness f_j, its probability p_j is defined as:

$$p_j = \frac{f_j}{\sum_{j=1}^{p_s} f_j} \qquad , j = 1, 2, ..., \rho_s \tag{1}$$

And the cumulative probability q_j for each solution is calculated as:

$$q_j = \sum_{i=1}^{j} p_i \qquad , j = 1, 2, ..., \rho_s \tag{2}$$

Where, the fitness f_j is relation to the objective function value of the j[th] chromosome.

```
Gen=1;                                          % The first generation
Chrom=Initial_pop();                            % Construct the initial population pool
While (termination_condition)
      Evaluuate (Chrom);                        % Objective function evaluation
      Chrom_sel=RWS_Selection(Chrom);           % Roulette wheel selection
      Chrom_rep=replication(Chrom_sel);         % Selection of better antibodies using
Replication
      Chrom_clon=Cloning(Chrom_rep);            % Clonal operation
      Chrom_hyper=Hypermutation(Chrom_clon);    % Hypermutation operation
      Chrom_tot=[ Chrom_rep, Chrom_hyper];
      Chrom_child=Mutation(Chrom_tot);          % Mutation Operation
      Evaluuate (Chrom_child);                  % Objective function evaluation
      Chrom=Better_selection(Chrom, Chrom_child);   % Selection of better antibodies for next
generation
      gen=gen+1;                                % Increment the number of generations
end
```

TABLE (1): The Immune Algorithm

2.3 Replication Operation

The replication operation is used to select better antibodies, which have low objective values to undergo algorithm operations. This is termed by clonal proliferation within hypermutation and mutation operations.

2.4 Clonal Proliferation within Hypermutation

Based on the biological immune principles, the selection of a certain antibody from the antibody population pool to join the clonal proliferation depends on the clonal selection rate (p_c). Each gene, in a single antibody, depending on the hypermutation rate (p_h), executes the hypermutation of convex combination. The hypermutation rate (p_h) has an extremely high rate than the mutation rate to increase the antibody diversity. For a given antibody $X = (X_1, X_2, ..., X_i, X_j, X_k, ..., X_\rho)$, if the gene X_i is determined to execute the hypermutation and another gene X_k is randomly selected to join in, the resulting offspring antibody becomes $X' = (X_1, X_2, ..., X_i', X_j, X_k, ..., X_\rho)$, where the new gene X_i' is $X_i' = (1-\beta)X_i + \beta X_k$, and β ∈ [0, 1] is a random value.

2.5 Mutation Operation

Similar to the hypermutation mechanism, the mutation operation is also derived from the convex set theory [9], where each gene, in a single antibody, depending on the mutation rate (p_m), executes the mutation of convex combination. Two genes in a single solution are randomly chosen to execute the mutation of convex combination [15]. For a given antibody $X = (X_1, X_2, ..., X_i, X_j, X_k, ..., X_\rho)$, if the genes X_i and X_k are randomly selected for

mutation depend on the mutation rate (p_m), the resulting offspring is $X^{'} = \left(X_1, X_2,..., X_i^{'}, X_j, X_k^{'},..., X_\rho\right)$. The resulting two genes $X_i^{'}$ and $X_k^{'}$ are calculated as:

$$X_i^{'} = (1-\beta)X_i + \beta X_k \text{ and } X_k^{'} = \beta X_i + (1-\beta)X_k \quad (3)$$

where, β is selected randomly in the range [0, 1].

2.6 Selection Operation

The selection operation is generally used to select the better p_s antibodies which have low objective values as the new antibody population of the next generation.

3. ILLUSTRATIVE EXAMPLES

In this section three different examples are considered to investigate the effect of parameters variation on the convergence speed of the immune algorithm. The first example simulates the multi-objective function problem that has an infinite set of possible solutions difficult to find [7]. The second example is a single-objective function problem and it is less difficult and the third example represents the family of problems with slow convergence to the global minimum [6].

Example 1:

This example considers the design of a second order 2-D narrow-band recursive LPF with magnitude and group delay specifications. The specified magnitude $M_d(\omega_1, \omega_2)$ is shown in Figure (1) [1], [5]. Namely, it is given by Equation (4) with the additional constant group delay $\tau_{d_1}=\tau_{d_2}= 5$ over the passband $\sqrt{\omega_1^2 + \omega_2^2} \le 0.1\pi$ and the design space is [-3 3]. To solve this problem, the frequency samples are taken at $|\omega_i / \pi| = 0,\ 0.02, 0.04, \ldots,\ 0.2,\ 0.4, \ldots, 1$ in the ranges $-\pi \le \omega_1 \le \pi$, and $-\pi \le \omega_2 \le \pi$.

$$M_d(\omega_1, \omega_2) = \begin{cases} 1.0, & for \sqrt{\omega_1^2 + \omega_2^2} \le 0.08\pi \\ 0.5, & for\ 0.08\pi < \sqrt{\omega_1^2 + \omega_2^2} \le 0.12\pi \\ 0.0, & for \sqrt{\omega_1^2 + \omega_2^2} > 0.12\pi \end{cases} \quad (4)$$

Example 2:

This example considers the optimization of the exponential function shown in Figure (2) and described by the following equation:

$$y(x) = \sum_{i=0}^{9} a_i x^i \quad (5)$$

With the following desired specified values $Y_d(x)$ at x= [0, 1, 2, 3, ……., 20].

$$\begin{aligned} Y_d(x) = [&0.01 \quad -0.01 \quad -3.83 \quad -4.79 \quad 758.33 \quad 9.0021\times10^3 \quad 5.7237\times10^4 \quad 5.7237\times10^4 \\ &9.2998\times10^5 \quad 2.8368\times10^6 \quad 7.6281\times10^6 \quad 1.8563\times10^7 \quad 4.165\times10^7 \quad 8.7358\times10^7 \\ &1.7309\times10^8 \quad 3.2667\times10^8 \quad 5.9104\times10^8 \quad 1.0306\times10^9 \quad 1.7397\times10^9 \quad 2.8528\times10^9 \\ &4.5587\times10^9] \end{aligned}$$

Example 3:

This example considers a Rosenbrock banana function that described by the following equation [6]. This function is often used to test the performance of most optimization algorithms [6]. The

global minimum is inside a long, narrow, parabolic shaped flat valley as shown in Figure (3). In fact find the valley is trivial, however the convergence to the global minimum is difficult.

$$f(x,y) = (1-x)^2 + 100\left(y - x^2\right)^2 \tag{6}$$

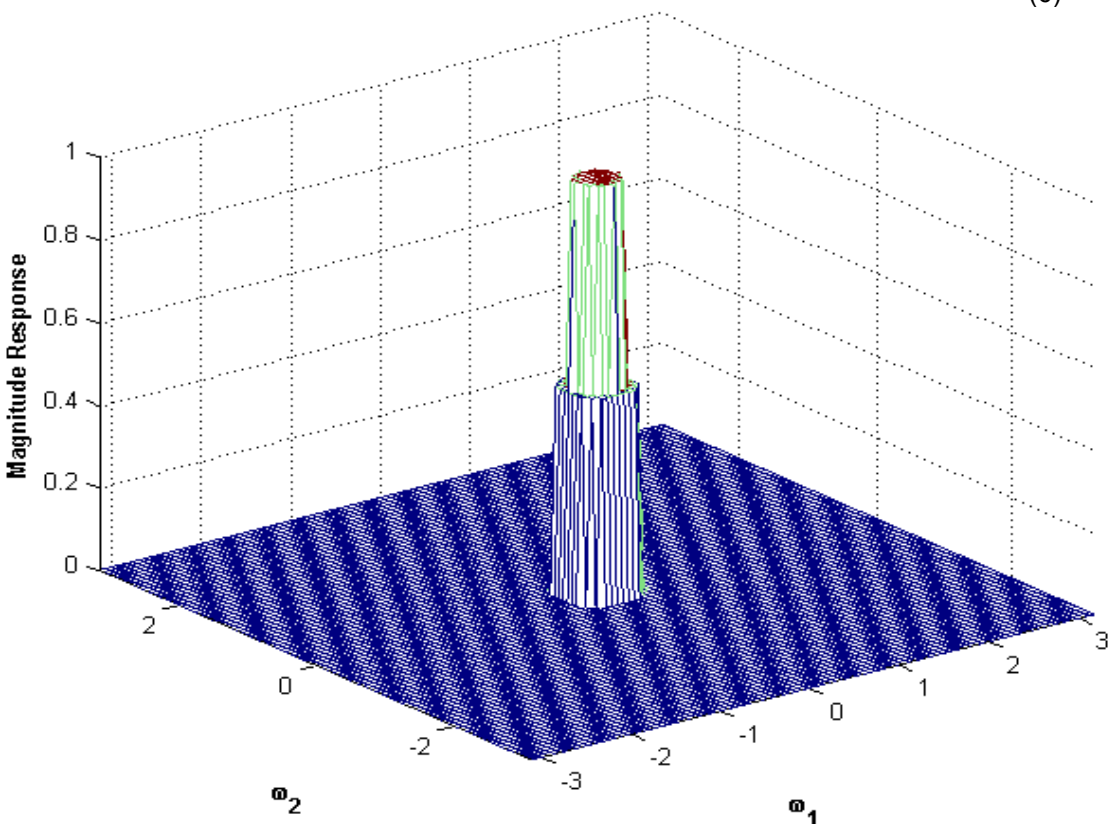

FIGURE 1: Desired Amplitude Response $\left|M_d(\omega_1, \omega_2)\right|$ Of The 2-D Narrow-Band LPF (Example 1)

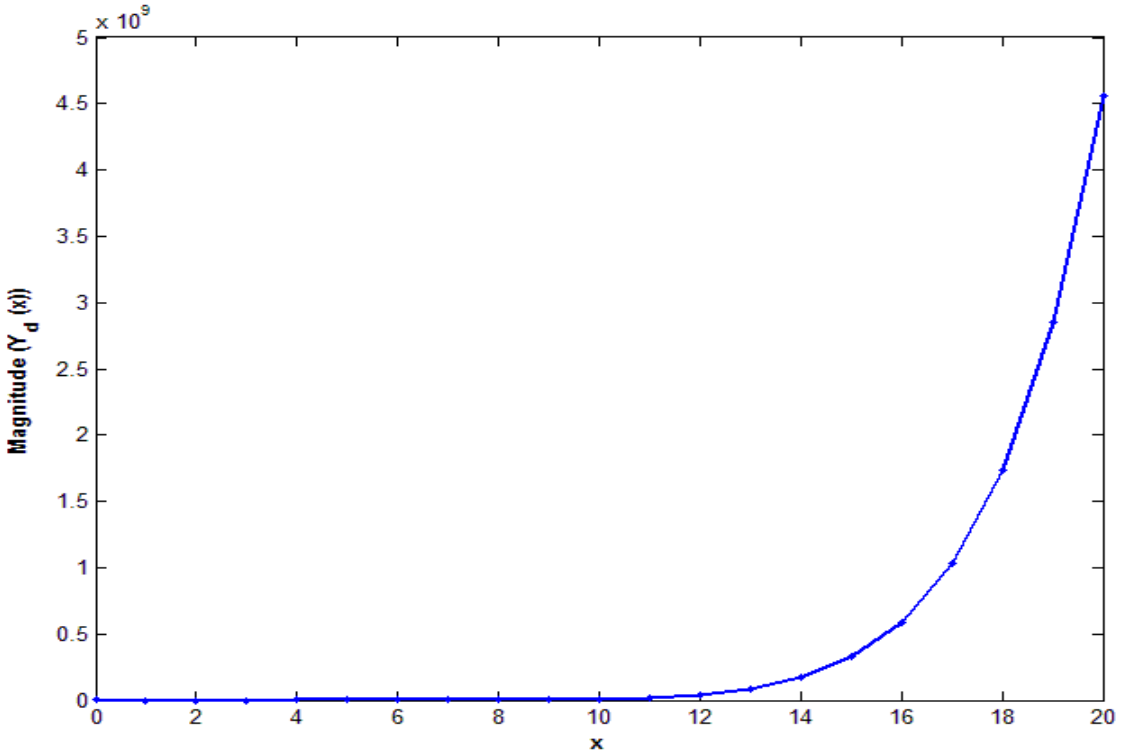

FIGURE 2: Desired Specifications of the Function $y(x)$ (Example 2)

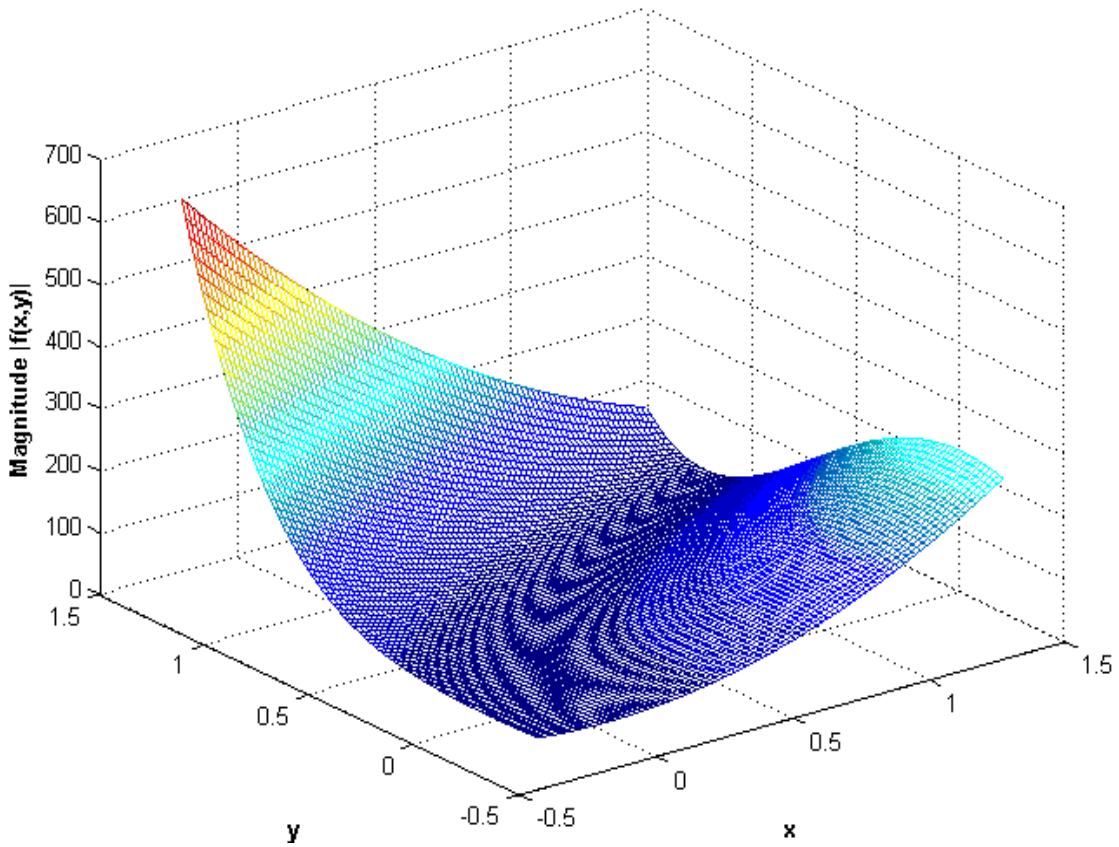

FIGURE 3: Rosenbrock Banana Function (Example 3)

4. SENSITIVITY ANALYSIS

In this section, we examine the effect of parameters variations on the convergence speed of the immune algorithm for the three examples described in section 3. The number of genes (the encoding length L) for each example is defined by the number of unknown coefficients. For the filter design problem, the filter transfer function is expressed by:

$$H(z_1,z_2)=H_0\frac{a_{00}+a_{01}z_2+a_{02}z_2^2+a_{10}z_1+a_{11}z_1z_2+a_{12}z_1z_2^2+a_{20}z_1^2+a_{21}z_1^2z_2+a_{22}z_1^2z_2^2}{(1+b_1z_1+c_1z_2+d_1z_1z_2)(1+b_2z_1+c_2z_2+d_2z_1z_2)},\ a_{00}=1 \tag{7}$$

So, 15 genes can be adjusted to approximate the specified magnitude and group delay. For the simple function and banana function problems, the number of genes considered are 10 and 2 respectively.

4.1 Effect of the population size (p_s)

The population size (p_s) is defined as the number of antibodies used in each generation. The variations in p_s can have substantial effect on the convergence speed of immune algorithm. If the p_s is too small, the IA cannot reach to optimal solution. However, if it is too large, the IA wastes computational time effort on extra objective values evaluations. Here, the effect of p_s on the convergence speed of the algorithm is studied by taking the average of 100 times independent runs at each p_s value. The value of p_s was varied from 10 to 400 with the other parameters fixed at p_r =0.8, p_h =0.8, p_m =0.1, and p_c =0.06. The effect of population size variations on number of generations required to get the solution for filter design problem, simple function and banana function are shown in Figures (4-6), respectively.

The results illustrated in Figures (4-6) show that, the speed of convergence can be measured by the number of generations required to reach to the optimal chromosome (global solution). Moreover, it can be noticed that the speed of convergence depends not only on the p_s but also on the number of genes. Here, the p_s after which optimal chromosome is obtained is denoted by p_s*. Increasing the p_s above p_s* has insignificant effect on speeding up the convergence.

4.2 Effect of the Replication Rate (p_r)

The replication rate (p_r) estimates the number of antibodies chosen from the antibody population pool to join the algorithm operations. The effect of p_r on the speed of convergence of the IA is studied by taking the average of 100 times independent runs at each p_r value. The value of p_r was varied from 0.1 to 1 with the other parameters fixed at p_s =100 p_h =0.8, p_m =0.1, and p_c =0.06. The effect of p_r variation on the number of generations required to produce the solution for filter design problem, simple function and banana function are shown in Figures (7-9), respectively.

These figures show that, the high values of replication rate have a significant effect on speeding up the convergence, but the computational time increases as the p_r increases. It is also noticed that the values of p_r greater than p_r* have no further effect on speeding up the convergence.

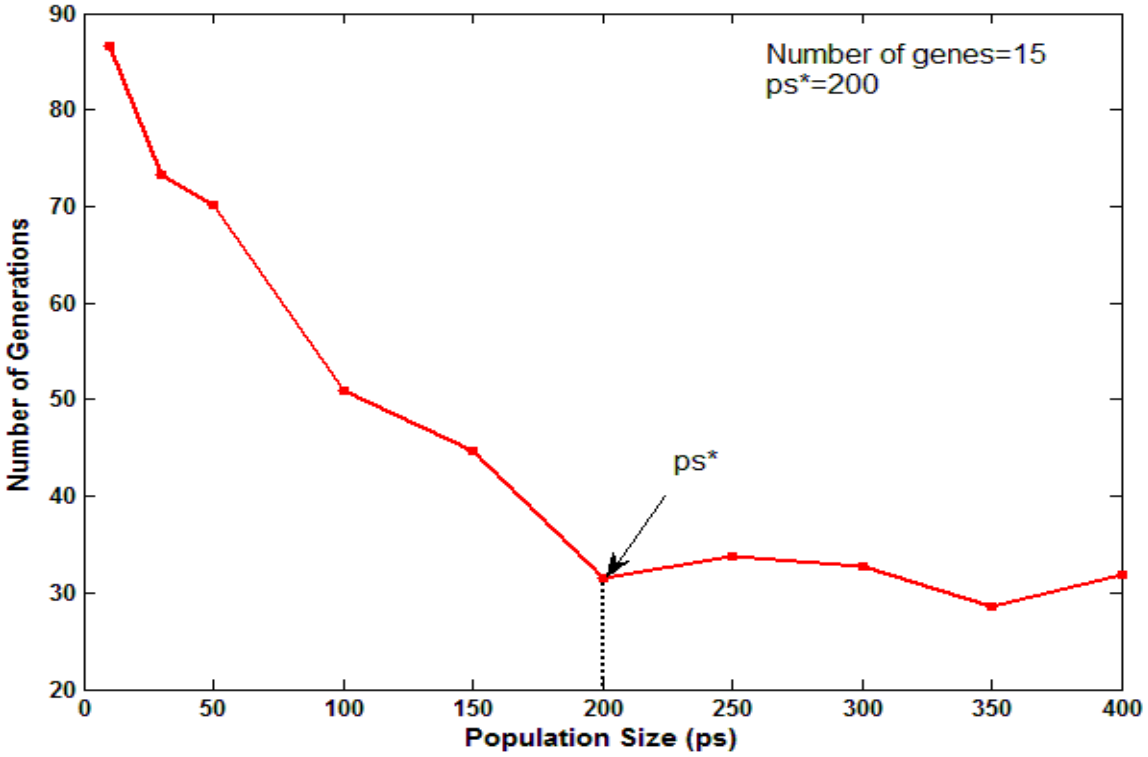

FIGURE 4: The Effect of Population Size on the Speed of Convergence of the Filter Design Problem.

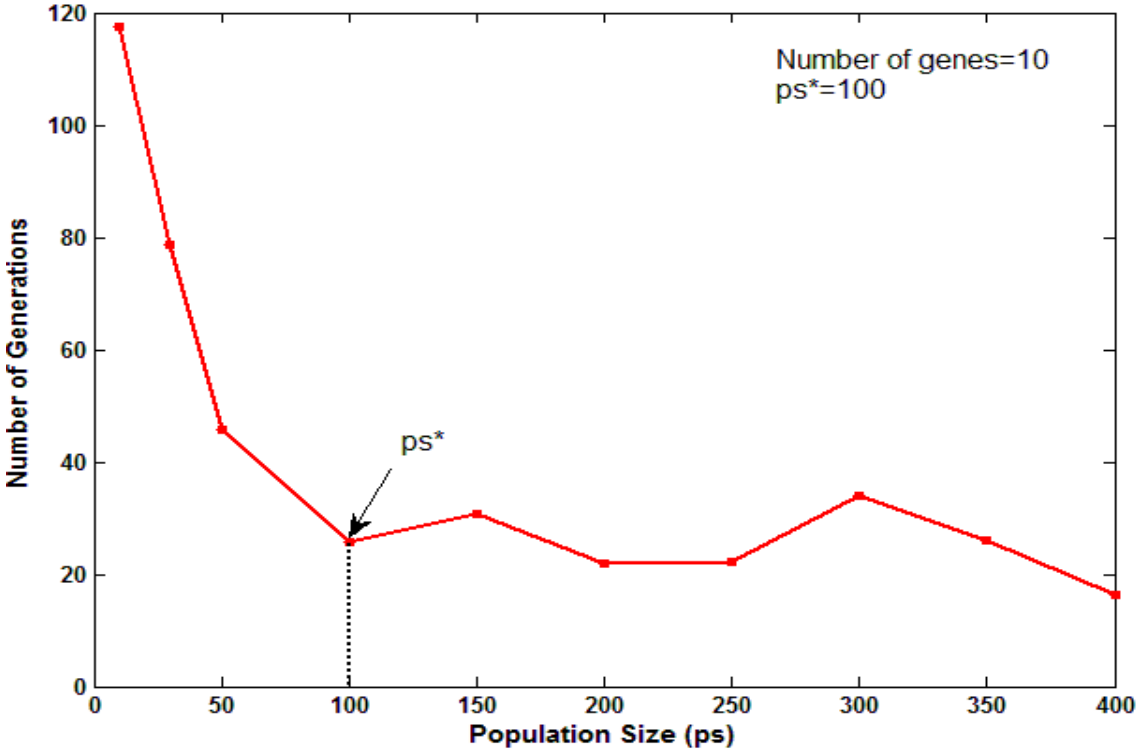

FIGURE 5: The Effect of Population Size on the Speed of Convergence for Simple Function Minimization

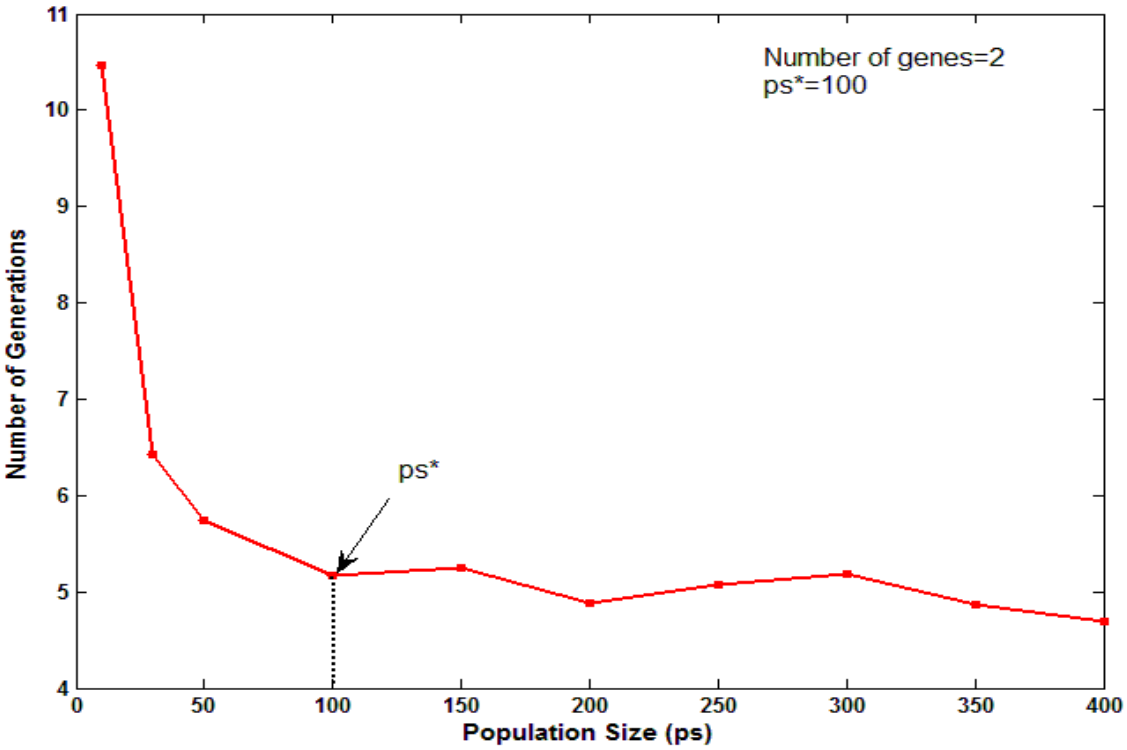

Figure 6: The Effect Of Population Size On The Speed Of Convergence For Finding The Global Minimum Of Banana Function.

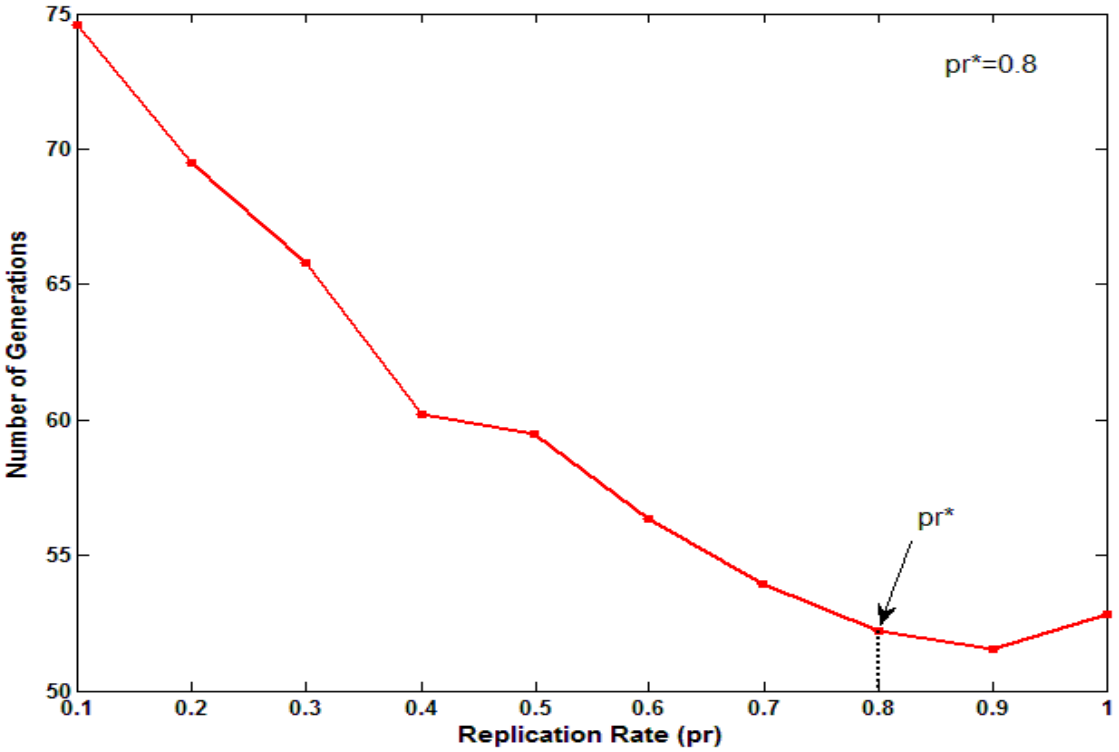

FIGURE 7: The Effect of Replication Rate on the Speed of Convergence for Filter Design Problem.

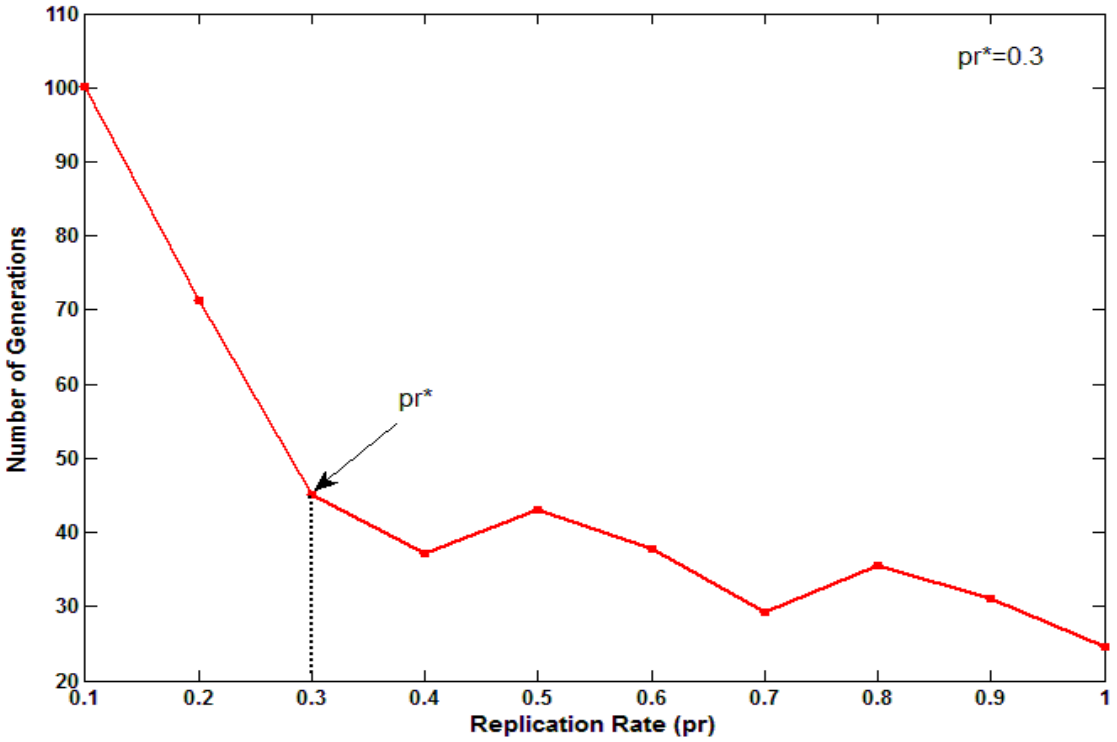

FIGURE 8: The Effect of Pr on the Speed of Convergence for Simple Function Minimization.

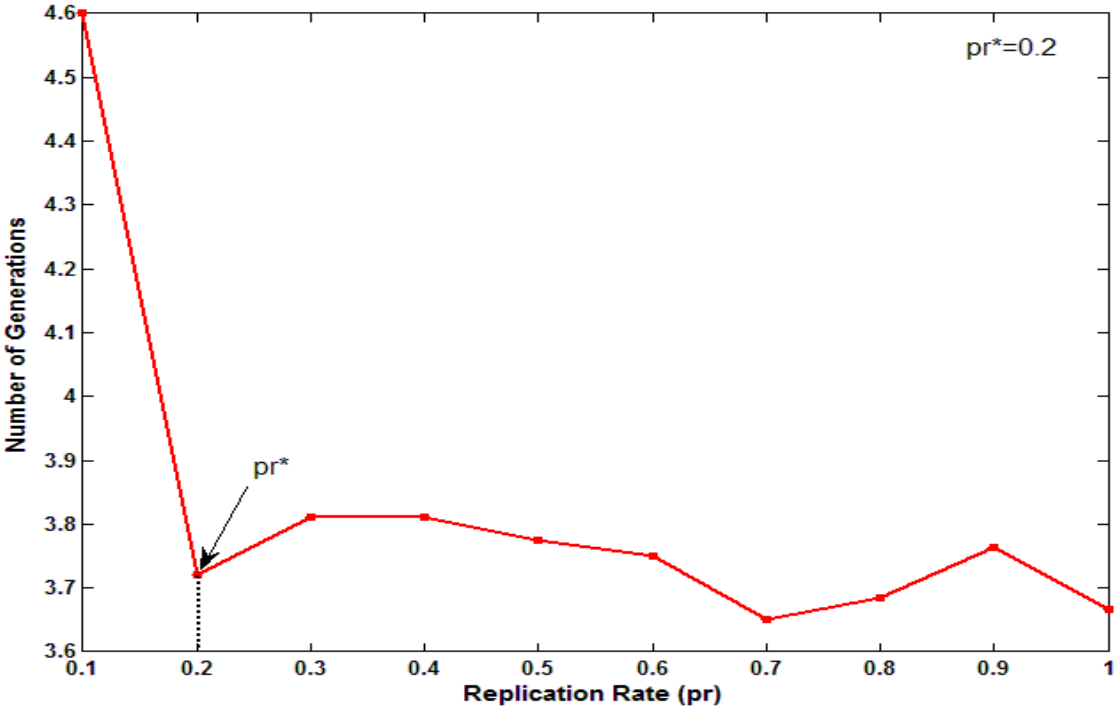

FIGURE 9: The Effect of Pr on the Speed of Convergence for Finding the Global Minimum of Banana Function.

4.3 Effect of the Clonal Selection Rate (p_c)

The clonal selection rate (p_c) estimates the number of antibodies that can be chosen from the antibody population pool to join the clonal proliferation. The effect of p_c on the speed of convergence of the IA is studied by taking the average of 100 times independent runs at each p_c value. The value of p_c was varied from 0.01 to 1 with the other parameters fixed at p_s =100, p_r =0.8, p_h =0.8, and p_m =0.1. The effect of p_c variation on the number of generations required to produce the optimal solution for filter design problem, simple function and banana function are shown in Figures (10-12), respectively.

From these figures, we can conclude that low values of p_c ($0.05 \leq p_c < 0.1$) have significant effect on speeding up the convergence. It is also noticed that the use of high values of p_c ($p_c \geq p_c^*$) have an effect of slowing down the convergence. This is mainly due to the infeasible selected individuals which joined to the clonal proliferation.

4.4 Effect of the Hypermutation Rate (p_h)

The hypermutation rate (p_h) is used to improve the capabilities of exploration and exploitation in population. The effect of p_h on the convergence speed of the IA is evaluated by taking the average of 100 times independent runs at each p_h value. The value of p_h was varied from 0.01 to 1 with the other parameters fixed at p_s =100, p_r =0.8, p_c =0.06, and p_m =0.1. The effect of hypermutation variation on the number of generations required to produce the solution for filter design problem, simple function and banana function are shown in Figures (13-15), respectively.

The results given in Figures (13-15) show that, the value of p_h depends on the problem domain. The values of p_h for the three illustrative examples are 0.5, 0.5, and 0.7, respectively. The p_h should be in the range ($0.5 \leq p_h < 1$) to speed up the convergence of small number of genes problems (example 3) and it is about 0.5 for other ones.

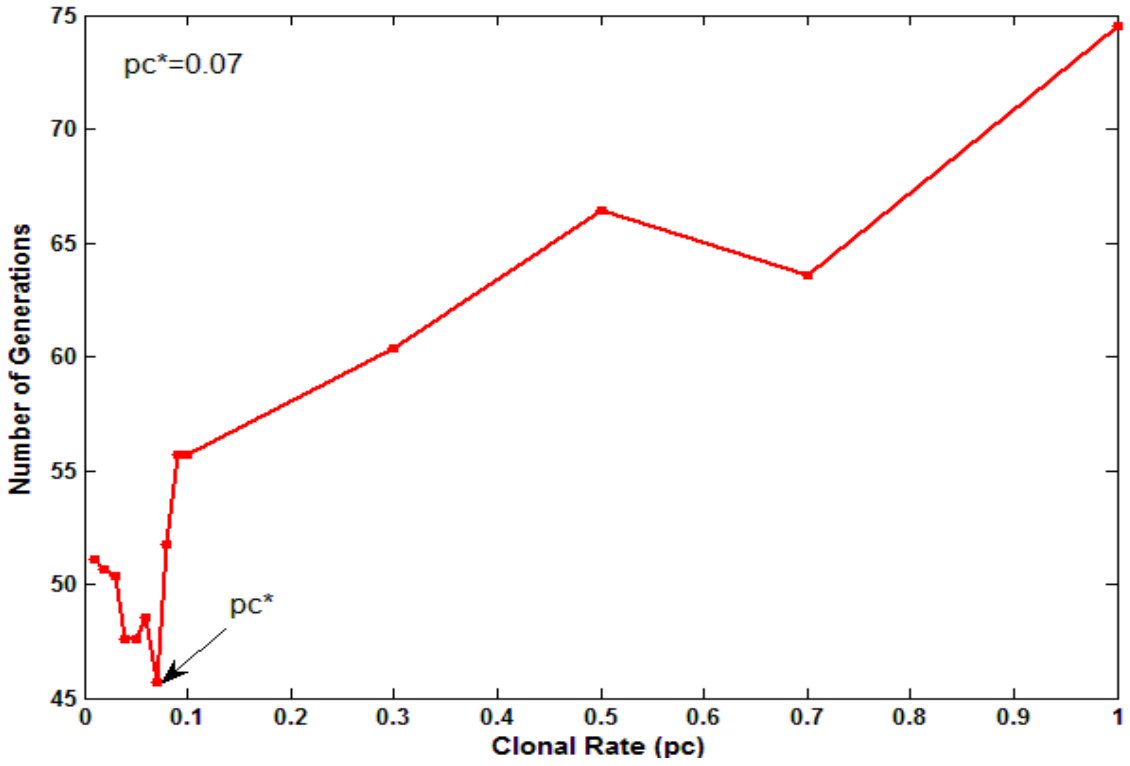

FIGURE 10: The Effect of Clonal Rate on the Speed of Convergence for Filter Design Problem.

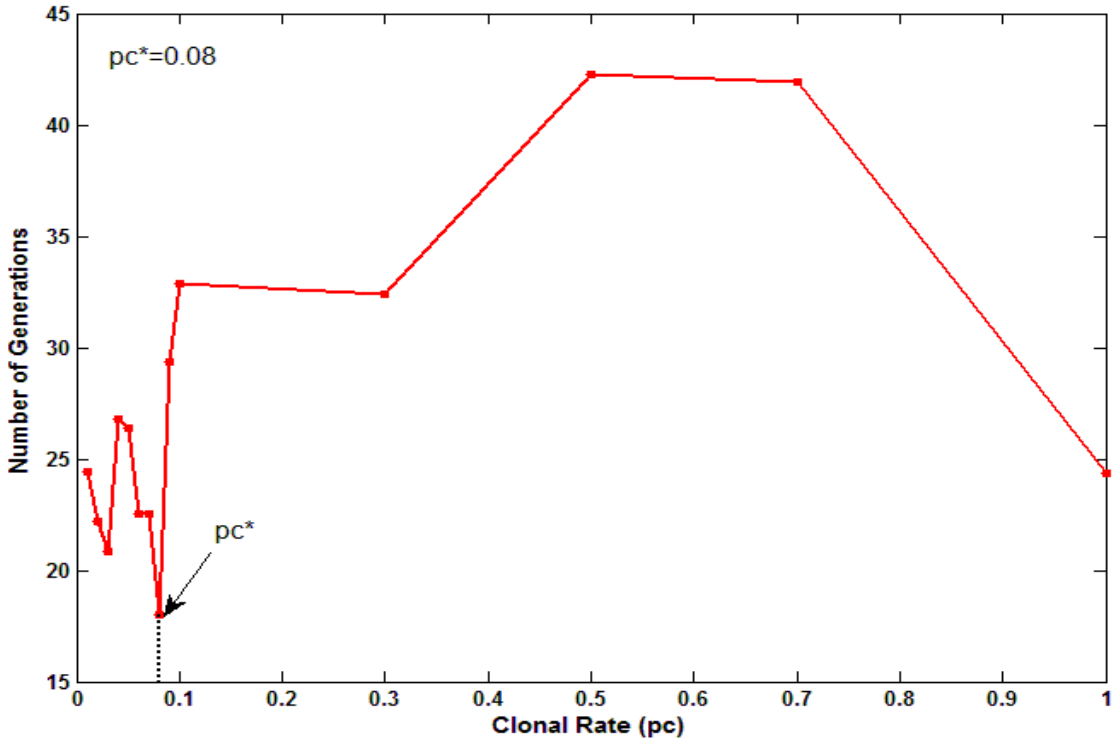

FIGURE 11: The Effect of Clonal Rate on the Speed of Convergence for Simple Function Minimization.

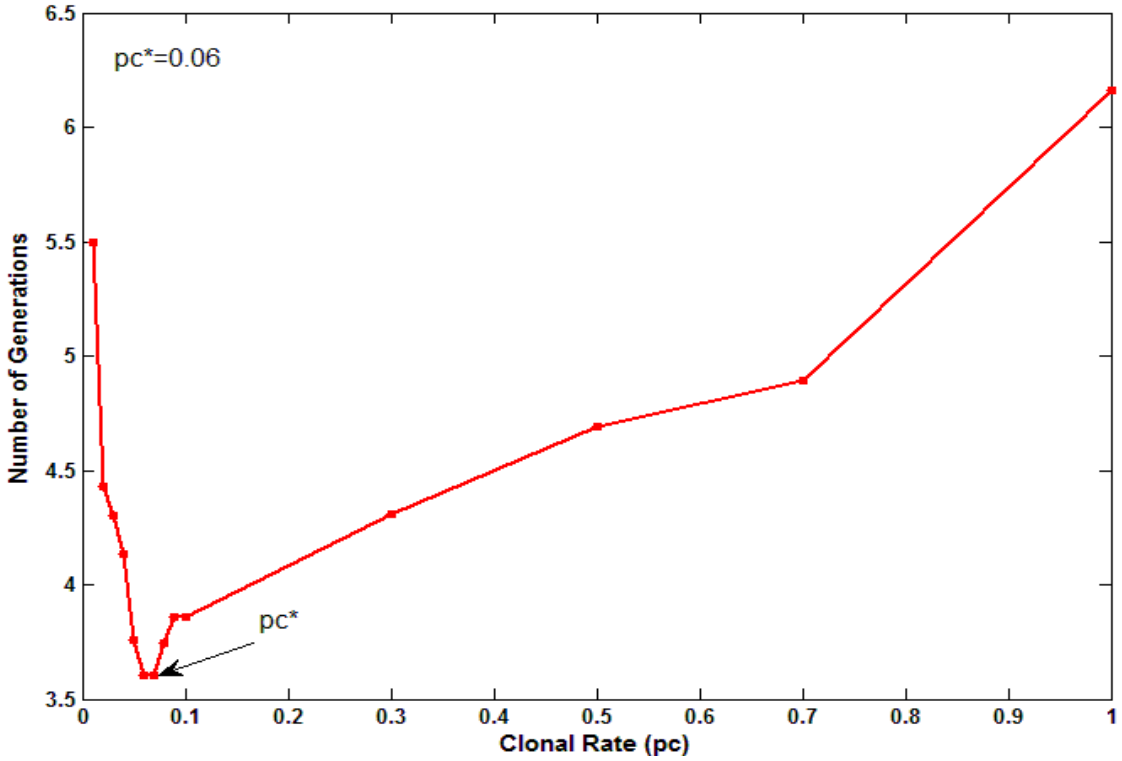

FIGURE 12: The Effect of Clonal Rate on the Speed of Convergence for Finding the Global Minimum of Banana Function.

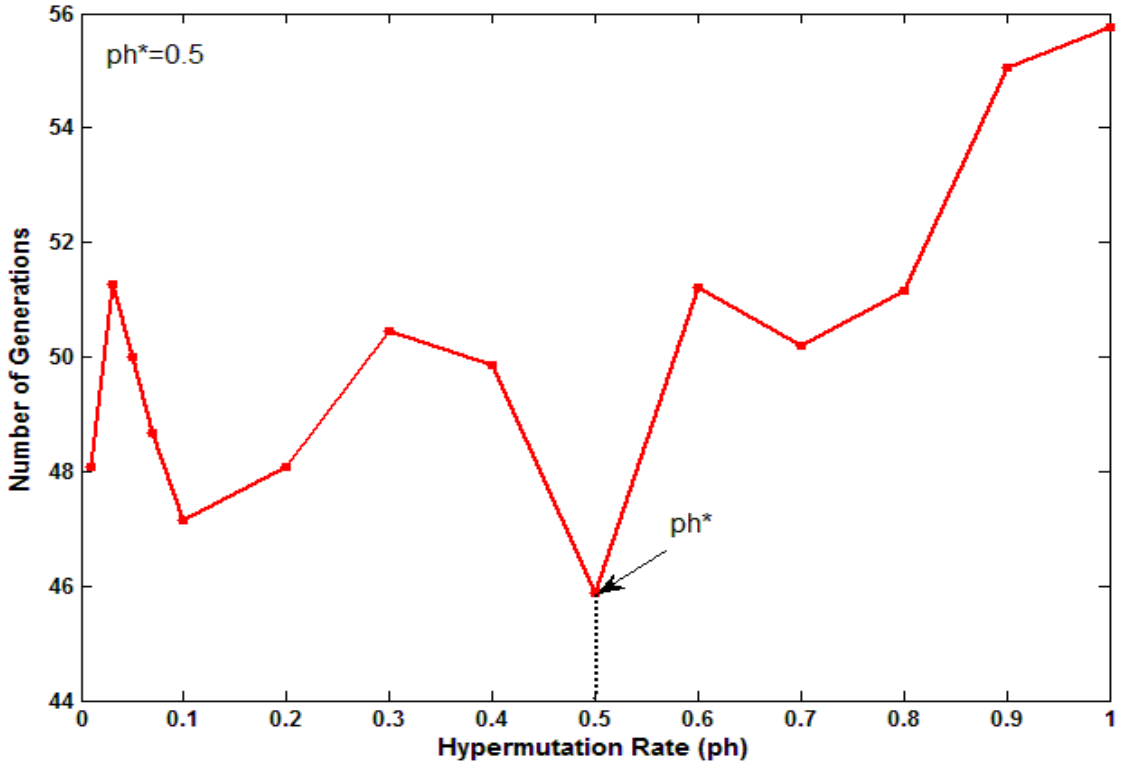

FIGURE 13: The Effect of Hypermutation Rate on the Speed of Convergence for Filter Design Problem.

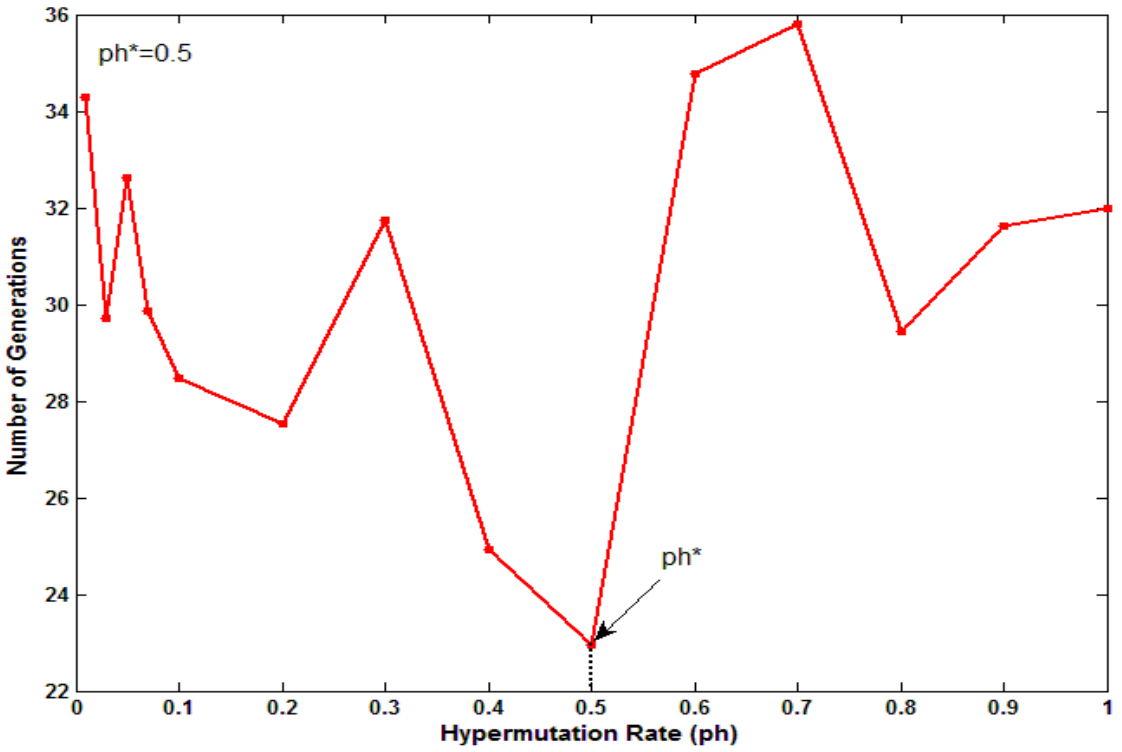

FIGURE 14: The Effect of Hypermutation Rate on the Speed of Convergence for Simple Function Minimization.

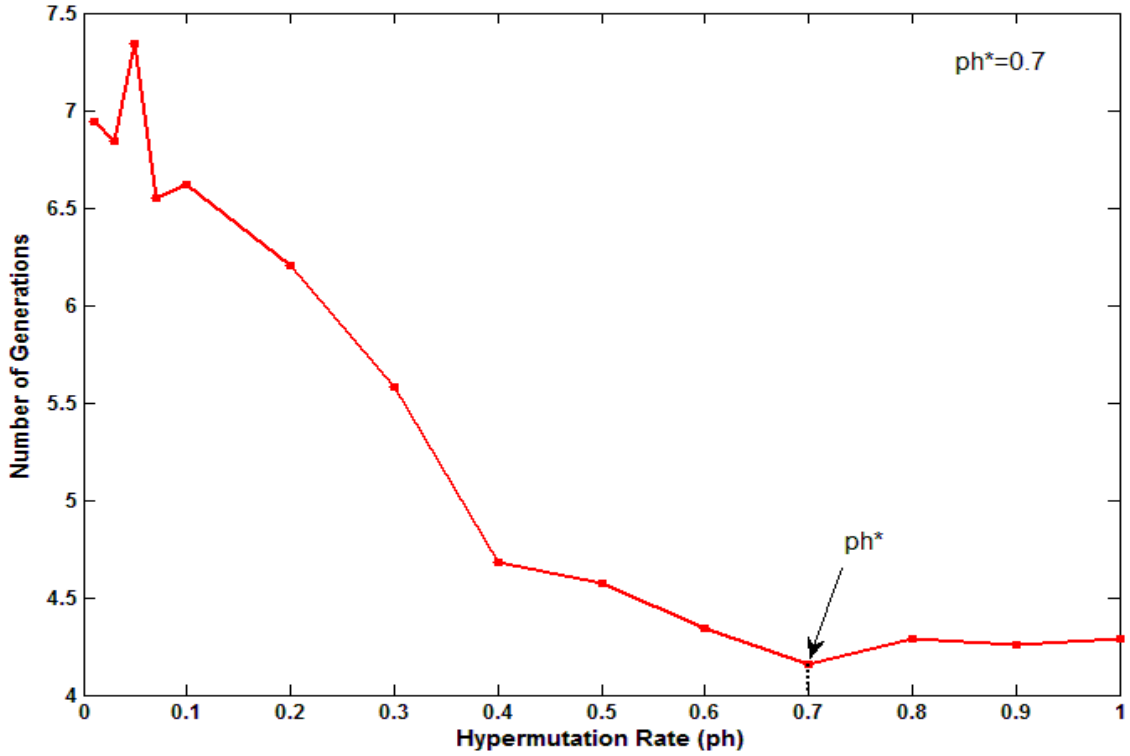

FIGURE 15: The Effect of Hypermutation Rate on the Speed of Convergence for Finding the Global Minimum of Banana Function.

4.5 Effect of the Mutation Rate (p_m)

The mutation rate (p_m) is one of the most sensitive immune algorithm parameters, since it increases the diversity in population. The choice of mutation rate is essentially a tradeoff between conservatism and exploration [14]. The effect of p_m on the convergence speed of IA is studied by taking the average of 100 times independent runs at each p_m value. The value of p_m was varied from 0.01 to 1 with the other parameters fixed at p_s =100, p_r =0.8, p_c =0.06, and p_h =0.8. The effect of mutation rate variation on the number of generations required to produce the solution for filter design problem, simple function and banana function are shown in Figures (16-18), respectively.

From these figures, we can conclude that the low values of mutation rate ($p_m \leq p_m^*$) have significant effect on speeding up the convergence. Also, it is noticed that to guarantee the convergence speed, the p_m should be between 1/ p_s and 1/L, where p_s is the population size and L is the encoding string length.

From above studying, we can conclude that the general heuristics on IA parameters to guarantee the convergence speed are: 1) the population size should be greater than 100; 2) the replication rate should be higher than 0.2; 3) the clonal rate should be small in the range ($0.05 \leq p_c < 0.1$); 4) the hypermutation rate should be high in the range ($0.5 \leq p_h < 1$); and 5) the mutation rate should be between 1/ p_s and 1/L.

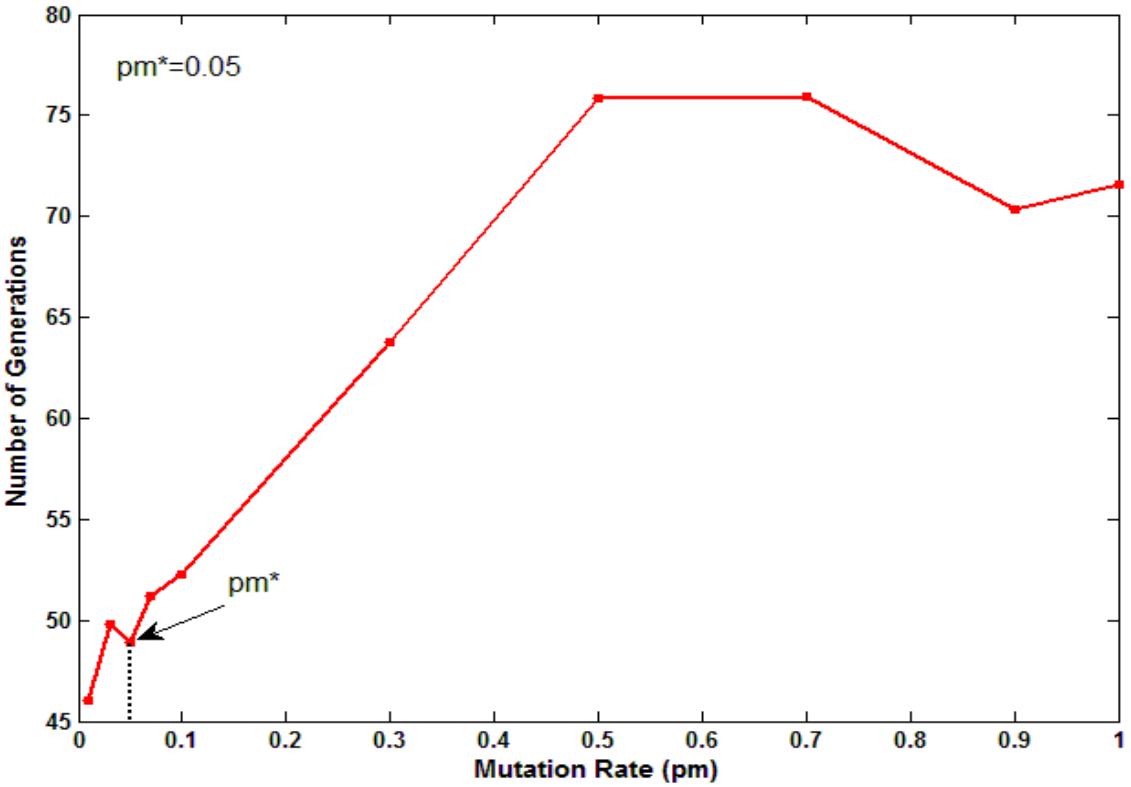

FIGURE 16: The Effect of Mutation Rate on the Speed of Convergence for Filter Design Problem (Ps=100 and L=15).

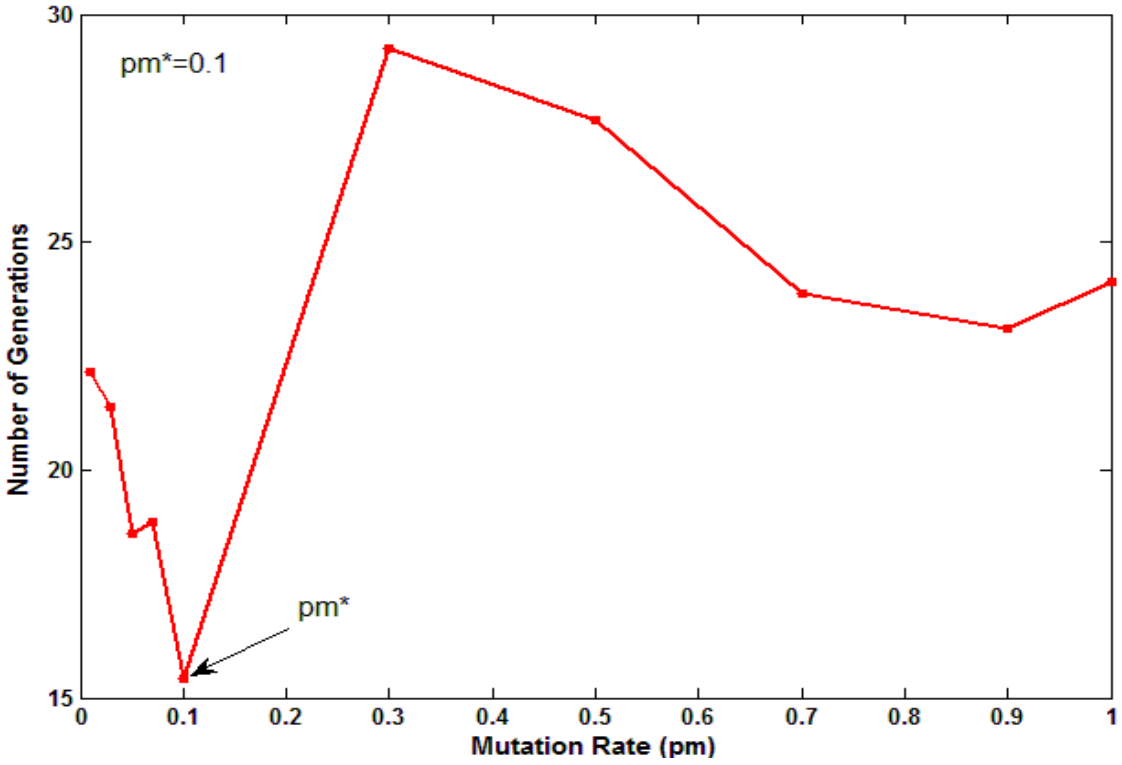

FIGURE 17: The Effect of Mutation Rate on Speed of Convergence for Simple Function Minimization (Ps=100 and L=10).

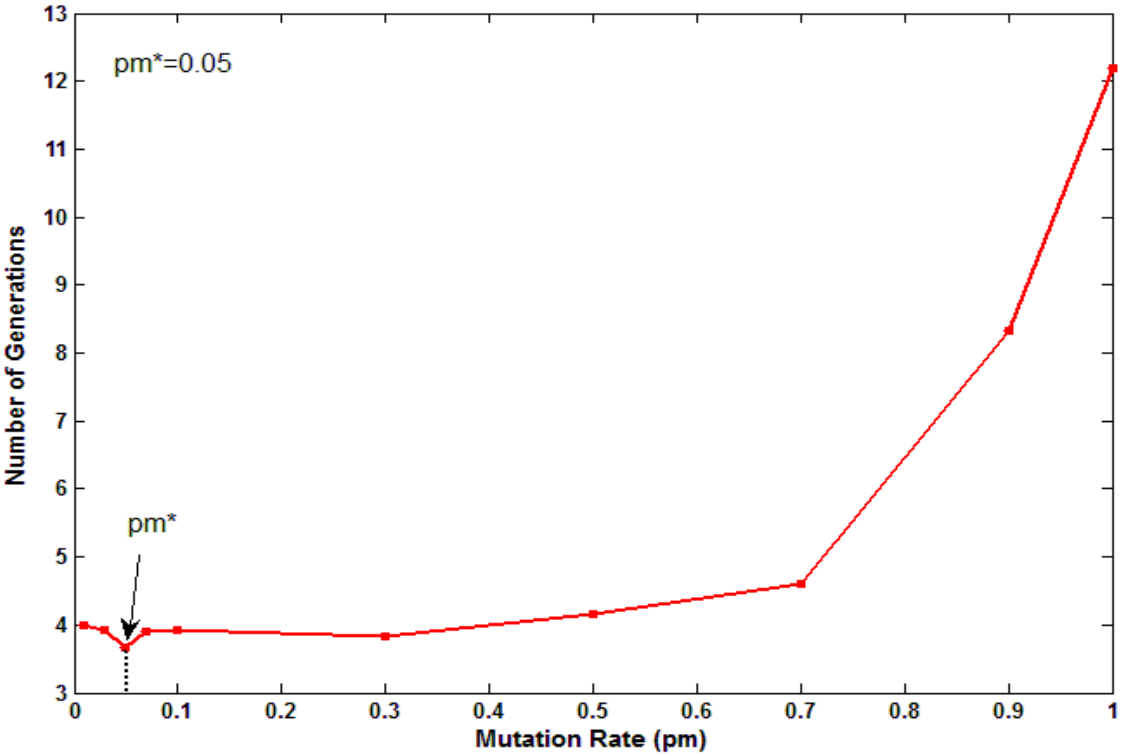

FIGURE 18: The Effect Of Mutation Rate On Speed Of Convergence For Finding The Global Minimum Of Banana Function (Ps=100 And L=2).

5 RESULTS AND DISCUSSION

In this section, some examples introduced in [3] and [12] are considered to illustrate the effect of immune algorithm parameters on the convergence speed.

Example 4:

This example is considered in [3] for solving system identification problem. It is repeated here to demonstrate the effectiveness of the selection of immune algorithm control parameters. In this example, it is required to approximate second-order system by first-order IIR filter. The second-order system and the filter are described respectively by the following transfer functions [3]:

$$H_p\left(z^{-1}\right)=\frac{0.05-0.4z^{-1}}{1-1.1314z^{-1}+0.25z^{-2}} \quad \text{and} \quad H_f\left(z^{-1}\right)=\frac{a_0}{1-b_1z^{-1}} \tag{8}$$

In Table (2), the control parameters selected based on the study described in previous section and that used in [3] are given. Table (3) illustrates the transfer function, the number of function evolution and NMSE of the resulting IIR filter and that is described in [3]. The NMSE is calculated using the following equation:

$$NMSE=\sqrt{\sum_{k=1}^{N}\left(\left|M\left(k\right)\right|-M_d\left(k\right)\right)^2}\Bigg/\sqrt{\sum_{k=1}^{N}\left(M_d\left(k\right)\right)^2} \tag{9}$$

Where, $M_d\left(k\right)$ and $M\left(k\right)$ are the magnitude responses of the 2nd order system and that of the designed filter respectively calculated at N=2000 sampling points.

IA Parameters	The selected parameters based on the above study	The selected parameters in [3]
Population size	100	50
Replication rate	0.85	0.80
Mutation rate	0.2	0.015
Clone rate	0.05	Not used in this method
Hypermutation rate	0.8	Not used in this method

TABLE 2: The IA Control Parameters Of Examples 1 And 2

	IIR filter obtained using proposed parameters values	IIR filter obtained using parameters values stated in [3]
Transfer Function	$H_f\left(z^{-1}\right)=\frac{-0.4153}{1-0.8645z^{-1}}$	$H_f\left(z^{-1}\right)=\frac{-0.311}{1-0.906z^{-1}}$
NMSE	0.0796	0.2277
Number of function evaluations to find the global optimal solution	1056	1230

TABLE 3: The Transfer Function, Number Of Function Evolutions And NMSE Of Both Resulting IIR Filter And IIR Filter Described In [3].

Figure (19) shows the magnitude responses of the second-order system, the resulting IIR filter and IIR filter described in [3]. From Figure (19) and Table (3), noticed that the resulting IIR filter

converge to the second-order system after smaller number of objective function evaluations with smaller NMSE compared to that given in [3]. So, the good selection of the IA control parameters speeds up the algorithm convergence.

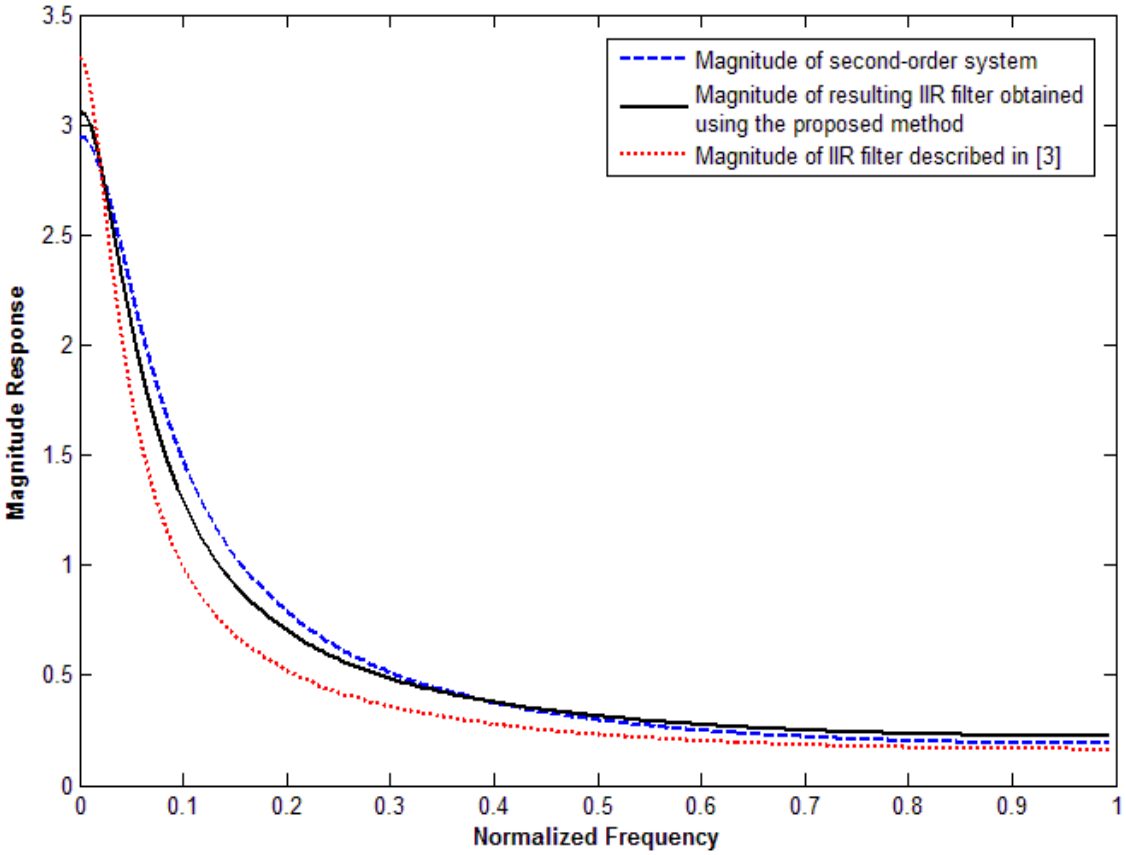

FIGURE 19: The magnitude responses of second-order system and IIR filter

Example 5:

This example is also considered in [3] for solving system identification problem. It is required to approximate a second order system by IIR filter with the same order. The system and the filter are described respectively by the following transfer functions [3]:

$$H_p\left(z^{-1}\right)=\frac{1}{1-1.2z^{-1}+0.6z^{-2}} \quad \text{and} \quad H_f\left(z^{-1}\right)=\frac{1}{1-b_1z^{-1}-b_2z^{-2}} \tag{10}$$

Using the same control parameters of example 1, the optimal solution (b_1= -1.1966, b_2= -0.59522) is obtained after 1503 objective function evaluations with MSE=0.393x10-3. However, the solution in [3] is obtained after 3000 objective function evaluations with MSE=0.5x10-3.

Example 6:

This example is considered in [12], for finding the global solution of the following test function:

$$f_4=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \tag{11}$$

The proposed IA is used to solve this function with 30 dimensions (i.e. N=30) in solution space [-600, 600]. In Table (4), the control parameters selected based on the study described in previous section and that used in [12] are given.

IA Parameters	The selected parameters based on the above study	The selected parameters in [12]
Population size	200	200
Replication rate	0.2	0.1
Mutation rate	0.02	0.02
Clone rate	0.06	0.01
Hypermutation rate	0.8	0.01

Table 4: The IA Control Parameters Of Example 3

Using the proposed IA, the solution is obtained after 13120 function evaluations; however in [12] is reached after 15743 function evaluations. So, the IA control parameters are having significant effect on the convergence speed.

6 CONCLUSIONS

In this paper, general rules on speeding up the convergence of the IA are discussed. The convergence speed of the IA is important issues and heavily depends on its main control parameters. In spite of the research carried out up to date, there are no general rules on how the control parameters of the IA can be selected. In literature [12]-[13], the choice of these parameters is still left to the user to be determined statically prior to the execution of the IA. Here, we investigate the effect of the parameters variation on the convergence speed by adopting three different objective optimization examples (2-D recursive filter design, minimization of simple function, and banana function). From the studied examples, the following general heuristics on immune algorithm parameters that guarantee the convergence speed are concluded: 1) the population size should be greater than 100; 2) the replication rate should be higher than 0.2; 3) the clonal rate should be small in the range ($0.05 \leq p_c < 0.1$); 4) the hypermutation rate should be high in the range ($0.5 \leq p_h < 1$); and 5) the mutation rate should be between $1/p_s$ and 1/L. These heuristics are applied to study cases solved in [3] and [12] to show effect of control parameter selection on the IA performance. Numerical results show that the good selection of the control parameters of the IA have significant effect on the convergence speed of the algorithm.

7 REFERENCES

1. J. T. Tsai, W. H. Ho, J. H. Chou. "*Design of Two-Dimensional Recursive Filters by Using Taguchi Immune Algorithm*". IET signal process, 2(2):110-117, March 2008
2. J. T. Tsai, J. H. Chou. "*Design of Optimal Digital IIR Filters by Using an Improved Immune Algorithm*". IEEE Trans. signal processing, 54(12): 4582–4596, December 2006
3. A. Kalinli, N. Karaboga. "*Artificial immune algorithm for IIR filters design*". Engineering Applications of Artificial Intelligance, 18(8): 919-929, December 2005
4. Alex A. Freitas, Jon Timmis. "*Revisiting the Foundations of Artificial Immune Systems for Data Mining*". IEEE Trans. on Evolutionary Compuation, 11(4): 521 - 540, August 2007
5. A. H. Aly, M. M. Fahmy. "*Design of Two Dimensional Recursive Digital Filters with Specified Magnitude and Group Delay Characteristics*". IEEE Trans. on Circuits and Systems, 25(11): 908-916. November 1978
6. Roy Danchick. "*Accurate numerical partials with applications to optimization*", Applied mathematics and computation, 183(1): 551-558, December 2006

7. M. Villalobos-Arias, C. A. Coello, O. Hernandez-Lerma. "*Asymptotic convergence of some metaheuristics used for multiobjective optimization*". LNCS, Springer, 3469: 95-111, 2005

8. M. Villalobos-Arias, C. A. Coello, O. Hernandez-Lerma. "*Convergence Analysis of a Multiobjective Artificial Immune System Algorithm*". In Nicosia et al. (eds) Proc. Int. Conf. Artificial Immune Systems (ICARIS 2004), LNCS, Springer, 3239: 226-235, 2004

9. M. Bazaraa, J. Jarvis, H. Sherali. "*Linear Programming and Network Flows*". John Wiley & Sons, New York (1990)

10. V. Cutello, G. Nicosia, M. Romeo, P.S. Oliveto. "*On the convergence of immune algorithm*". IEEE Symposium on Foundations of Computational Intelligence: 409 - 415, April 2007

11. Z. Michalewiz*. "Genetic Algorithm and Data Structure".* Springer-Verlag Berlin Heidelberg, 3rd ed. (1996)

12. J. T. Tsai ,W. Ho ,T.K. Liu, J. H. Chou. *"Improved immune algorithm for global numerical optimization and job-shop scheduling problems ".* Applied Mathematics and Computation, 194(2): 406-424, December 2007

13. G. Zilong, W. Sun'an, Z. Jian*. "A novel Immune Evolutionary Algorithm Incorporating Chaos Optimization".* Pattern Recognition Letter, 27(1): 2:8, January 2006

14. F. Vafaee, P.C. Nelson. "*A Genetic Algorithm that Incorporates an Adaptive Mutation Based on an Evolutionary Model*", International Conference on Machine Learning and Applications, Miami Beach, FL, December 2009.

15. K. Kaur, A. Chhabra, G. Singh. "*Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System*". International Journal of Computer Science and Security, 4(2): 183-198, May 2010.

16. M. Abo-Zahhad, S. M. Ahmed, N. Sabor and A. F. Al-Ajlouni, "*Design of Two-Dimensional Recursive Digital Filters with Specified Magnitude and Group Delay Characteristics using Taguchi-based Immune Algorithm*", Int. J. of Signal and Imaging Systems Engineering, vol. 3, no. 3, 2010.