International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 4, August 2017, pp. 2071~2084
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i4.pp2071-2084

The Evaluated Measurement of a Combined Genetic Algorithm and Artificial Immune System

Pongsarun Boonyopakorn[1], Phayung Meesad[2]
[1]Departement of Information Technology, King Mongkut's University of Technology North Bangkok, Thailand
[2]Departement of Information Technology Management, King Mongkut's University of Technology North Bangkok, Thailand

Article Info

Article history:

Received Jan 26, 2017
Revised May 30, 2017
Accepted Jun 14, 2017

Keyword:

Artificial immune system
Genetic algorithm
Hybrid algorithm
Immune genetic algorithm
Optimization mathematical

ABSTRACT

This paper demonstrates a hybrid between two optimization methods which are the Artificial Immune System (AIS) and Genetic Algorithm (GA). The novel algorithm called the immune genetic algorithm (IGA), provides improvement to the results that enable GA and AIS to work separately which is the main objective of this hybrid. Negative selection which is one of the techniques in the AIS, was employed to determine the input variables (populations) of the system. In order to illustrate the effectiveness of the IGA, the comparison with a steady-state GA, AIS, and PSO were also investigated. The testing of the performance was conducted by mathematical testing, problems were divided into single and multiple objectives. The five single objectives were then used to test the modified algorithm, the results showed that IGA performed better than all of the other methods. The DTLZ multi-objective testing functions were then used. The result also illustrated that the modified approach still had the best performance.

Corresponding Author:

Pongsarun Boonyopakorn,
Departement of Information Technology,
King Mongkut's University of Technology North Bangkok,
1518 Pracharat 1 Road, Wongsawang, Bangsue, Bangkok 10800, Thailand.
Email: pongsarun.b@it.kmutnb.ac.th

1. INTRODUCTION

Optimization search research is an operation that refers to the procedure of finding the best solution to objective functions. In general, optimization searching considers the solution into local and global searches. It can be group into two categories which are single and population based. In recent years, the most popular technique used to solve the solutions was the Genetic Algorithm (GA) which is a population based search system. The GA is a metheuristic method pioneered by Darwin's [1] which is based on the principle of natural genetics and natural biological selection. The process of GA's is iteratively the initial population of candidate solutions until the criteria have been met. The GA operation begin with the initial the number of population which are related to the solution thats needed to be solve. The selection is a method that uses selection of parents from each individual in each iteration for regeneration. The crossover and mutation are operators used for regeneration to the next iterator. Finally, the evaluation of candidate solution will calculate then terminate the solusions once reached or iteration terminated.

The GA has been successfully applied to many research areas such as optimized tools, engineering, science, and management. In recent years, GAs have been proposed as hybrids by combining to virous approaches such as PSO, Ant Colony, and Artificial Immune System which are effective for local and global searches aimed at improving the solution quality. Various problems have been solved by the hybrid GA which includes finding optimal traffic networks, job scheduling, stock markets, and data mining. Tarek A. [2], proposed a hybridization between Ant-based Algorithm and Genetic Algorithm. In their research, an Ant

Colony was used to monitor the behavior of a genetic-local hybrid algorithm and dynamically adjusted its control parameters to optimize the exploitation exploration balance according to the fitness landscape. Jyoti [3], presented a hybrid combining the Particle Swarm Optimization algorithm (PSO) based on testing five functions. The idea behind the hybrid algorithm is that the total iterations have to be distributed between the genetic algorithm and particle swarm optimization algorithm. The proposed hybrid algorithm is proven to be more efficient than GA and PSO. In [4] Zhao, proposed a hybrid genetic algorithm for Bayesian network optimization. Their work used the Simulated Annealing technology to select children and used self-adaptive probabilities of crossover and mutation to conduct the local search. Finally, the Hill-climbing algorithm was employed to optimize the results. In [5], Wu and Lu studied the effects of hybrid optimization strategies by incorporating the metropolis acceptance criterion of Simulated Annealing (SA) into the crossover operator of GA. The algorithm was used to simultaneously optimize the input feature subset selection, the type of kernel function and the kernel parameter setting of SVR, namely GASA–SVR. In summary of the above, the study of hybrid Genetic Algorithms has yielded several successful approaches.

The Artificial Immune System (AIS), has been studied deeply in recent years which is a class of biologically inspired computation paradigm [6]. AIS approaches are used in various optimization applications and most of them show better efficiency in comparison with other population based algorithms. Various AIS models such as clonal selection, immune networks, and negative selection are also used in several applications such as optimization, clustering, pattern recognition and anomaly detection. In general, GA and AIS have been adopted as optimizers in the binary base which is categorized as NP-hard. Zhu [7], investigated two theories of AIS which are clonal selection and immune network theory, and integrated them with PSO to solve the job scheduling problem. In his research, the clonal selection theory is used to set up the framework which contains the processes of selection, cloning, hypermutation and receptor editing, while the immune network theory is applied to increase the diversity of the potential solution repertoire. Barani [8], proposed an approach based on the genetic algorithm (GA) and artificial immune system (AIS), called GAAIS, for dynamic intrusion detection in AODV-based MANETs. His approach was able to adapt itself to network topology changes using two updating methods: partial and total. Each normal feature vector extracted from network traffic was represented by a hypersphere with fix radius. Ali et al [9], improved the results of performance in the hybrid AIS and GA. The hybrid included two processes; firstly, AIS enables it to develop local searching ability and efficiency although the convergence rate for AIS is preferably not precise compared to the GA. Secondly, a Genetic Algorithm is typically initializing population randomly. The last generation of AIS will be the input to the next process of the hybrid which is the GA in this hybrid AIS-GA. A hybrid can ensure that a GA enters the stage of standard solutions more rapidly and accurately compared to GA initialized population at random.

As mention above, the hybrid AIS and GA have been applied to difference optimization application areas in recent years. The object of this paper is to describe the modified Genetic algorithm (GA) which is a combination of an Artificial Immune System (AIS) to form an Immune Genetic Algorithm (IGA) to reduce the search space and achieve efficient searches. Performances of the IGA and two other techniques will be compared. This paper is divided as follows: Section 2 presents the research method of the evolutionary algorithm. Section 3 covers the results and analysis. Finally, the conclusion will be presented in session 4.

2. RESEARCH METHOD

This section discusses and analyzes the aim of the hybrid immune genetic algorithm concepts to utilize the locally characteristic information to seek out the ways and means of discovering the optimal solution when dealing with difficult problems. One must first generate a random detector, and then the initial population. Next perform selection, crossover, and mutation upon the population for a number of generations, until termination criterion is met.

2.1. Negative Selection

Negative selection inspired from the T cell maturation process has been developed for self-nonself detection in computer systems. In this technique, the first information is represented in a suitable form such as string form, real valued vector form, and hybrid form are considered as self-data. Then additional data are created in the same form as the self-data, in such a way that any of the newly created data does not match the self-data. The matching is done according to a matching rule which is selected depending on suitability. These newly created data which are used to distinguish between self-data and nonself-data are called detectors. If any of the detectors matches the data, then that data is considered nonself-data. Whereas, if no detector matches the data then that data is considered self-data. The detectors are created in such a way that they do not match any of the self-data. In negative selection, the T cell is presented to the self-body cells. If

the T cell recognizes any of the self-body cells, then the cell is rejected. Remaining T cells are considered matured T cells and are used for the self-nonself detection [10].

Pseudocode for detector generation

```
1:  Input: SelfData
2:  Output: Repertoire
3:  Repertoire ←Φ
4:  While (¬StopCondition())
5:    Detectors ← GenerateRandomDetectors()
6:    For (Detector_i ∈ Repertoire)
7:      If (NotMatches(Detector_i, SelfData))
8:        Repertoire ← Detector_i
9:      End
10:   End
11: End
12: Return (Repertoire)
```

Figure 1. Pseudocode for detector generation

Figure 1 describes the major steps in such an algorithm. In the generation stage, the detectors are generated by a few random process and censored by trying to match self samples. Those candidates that match are eliminated and the rest are kept as detectors. In the detection stage, the collection of detectors (or detector set) are used to check whether an incoming data instance is self or nonself. If it matches any detector (refered to Figure 2), it is claimed as nonself or an anomaly. This description is limited to a few extents, but conveys the essential idea.

Pseudocode for detector application

```
1:  Input: InputSamples, Repertoire
2:  For (Inputi_class □ InputSamples)
3:      Inputi_class ← "non-self"
4:    For (Detector_i □ Repertoire)
5:      If (Matches(Input_i, Detector_i))
6:          Inputi_class ← "self"
7:        Break
8:      End
9:    End
10: End
```

Figure 2. Pseudocode for detector application

2.2. Matching Rules

Matching rule is an important part in detector generation. There are different matching rules such as Hamming distance, Binary distance, Edit distance, and Value difference metric to match strings. In this paper, focus is on the R-Contiguous Bits (RCB) matching rule and R-Chunk matching rule [11]. The RCB matching rule is defined as follows: If x and y is equal-length strings defined over a finite alphabet, match (x, y) is true if x and y agree in at least r contiguous locations. As in the RCB matching rule, a detector is specified by a binary string c and parameter r.

2.3. Detector Generation

The detector generation technique can be divided into two parts. *i)* The value of the length of the chunk is taken from the user. Let the chunk length be x then from the first bit of a self-string x, none of the continuous bits are taken to form a chunk. Then to form the second bit x, none of the continuous bits are taken form another chunk and this goes on as long as x has none of the continuous bits taken to form a chunk. So, if the length of self-string is y, then y-x+1 none of the chunks are formed from each self-string. *ii)* Each self-chunk set is taken one by one to the detector sets separately. As chunks are already created from self-strings two strings are considered the same only if all the bits of the two strings exactly match each other. Next, detectors are to be created such that newly created detectors do not match previously generated detectors or the self-chunk strings even-though the randomness of the detector generation process are maintained.

2.4. Chromosome Representation

The chromosome representation depends on the nature of the problem variables. The value of a bit string can be an integer number or binary number. For example, the representation choice of timetabling schedules for a few objects. A possible number of 15-bit strings can be used to represent a possible solution to a problem. In this case bits or subsets of bits might represent a choice of a few features: subject, section, instructor, time, and room. Figure 3 shows the chromosome Representation

1	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**

Figure 3. Chromosome Representation

where bit 1-3 represents subject, 4-6 represents course section, 7-9 represents instructor or professor, 10-12 represents times, and 13-15 represents room.

2.5. Initial Population

The chromosome's fitness value is assessed during the initial population process. Each individual contains its own fitness value. One possible way to assign a fitness value to individuals is by the following formula

$$fitness_i = \sum_{i=0}^{n} \alpha_i \quad (1)$$

where α is an element of bit-string and i is a number of bit-string that's contained in each individual.

2.6. Selection

The selection process chooses the next generation of the best individual. It stochastically allocates a higher number of copies in the following generation to highly fitting strings in the present generation. Four common methods for selection are Roulette Wheel selection, Stochastic Universal sampling, Normalized Geometric selection, and Tournament selection. For example, Figure 4 shows Tournament selection which provides a chance to all individuals to be selected and thus it preserves diversity, although keeping diversity may degrade the convergence speed. In tournament selection, n individuals are selected randomly from the larger population, and the selected individuals compete against each other. The individual with the highest fitness wins and will be included as one of the next generation population. The number of individuals competing in each tournament is referred to as tournament size, commonly set to 2 (also called binary tournament).

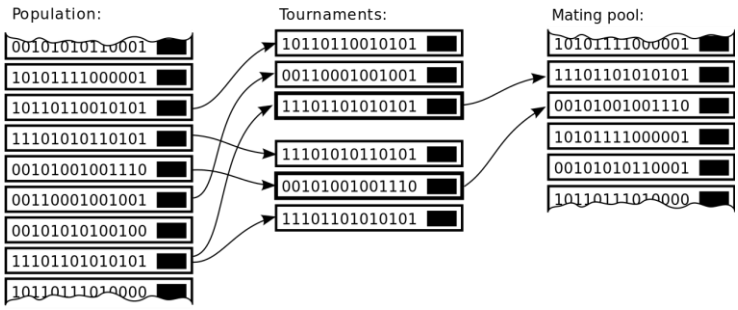


Figure 4. Illustration of Tournament Selection in Size of 2

2.7. Crossover

The crossover process produces better chromosomes, two of the strongest are picked to produce a new chromosome of offspring. Figure 5 shows the example of single point crossover. Three types of

crossover are applied in this process including Single point crossover, double point crossover, and Uniform crossover.

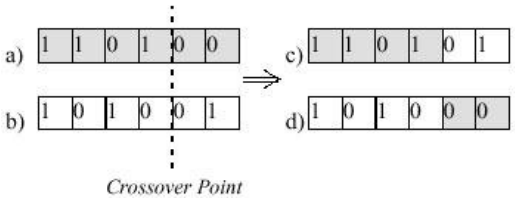


Figure 5. Illustration of Sing Poing Crossover

2.8. Mutation

Mutation is the occasional random alteration of a value of a string position. The purpose of mutation in GAs are to preserve and introduce diversity. For different genome types, different mutation types are Bit string mutation, Flip Bit, Boundary, Uniform, and Gaussian. A randomly selected element of the string is altered or mutated when a string is chosen for mutation. Normally, mutation ranges around 0.1% - 0.2%. Figure 6 shows an example of bit string mutation.

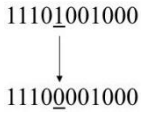


Figure 6. Illustration of Bit String Mutation

2.9. The Pseudocode of the Algorithm

The proposed algorithm begins with initialize detector *D*, each of which fails to be a random value. The next step is to calculate the fitness of each cell in the population and rank them. In this case, the best candidate will be chosen to be detector *D*. Next, initialize a population *P* of gene, each set will have a random value then perform negative selection in any *P* which matches *D*. Calculate fitness of each chromosome in *P* and rank them and perform crossover and mutation. Loop if termination condition is not met then stop. The psudocode for the Immune Genetic Algorithm is shown in Figure 7.

Pseudocode for immune genetic algorithm

```
d ← 0;
InitDetector[D(d)];    {Initializes the detector}
EvalDetector[D(d)];    {Evaluates the detector}
t ← 0;
InitPopulation[P(t)];  {Initializes the population}
EvalPopulation[P(t)]; {Evaluates the population}
Matches[P(t), D(d)]; {Matches between the population and detector}
while (not terminatation) do
   P'(t) ← Variation[P(t)];        {Creation of new solutions}
   EvalPopulation[P(t)];           {Evaluates the new solutions}
   P(t+1) ← ApplyGeneticOperators[P'(t) □ Q];        {Next generation pop.}
   t ← t+1;
end while
```

Figure 7. Pseudocode for immune genetic algorithm

2.10. Mathematical Functions

In order to compare and evaluate different algorithms, researchers have been looking for various benchmark functions with various properties. Five test functions are used in this paper comprising of Ackley function, Bohachevsky functions, Sphere function, Rastrigin function, and Fifth function of De Jong to compare between GA, AIS, IGA, and PSO.

2.10.1. Single Objective Test Functions

In order to compare and evaluate different algorithms, various benchmark functions with various properties have been suggested. Five single objective test functions are used in this thesis to compare between GA, AIS, IGA, and PSO. The following are the test functions.

2.10.1.1. Ackley Function

$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \tag{3-6}$$

subject to $-35 \le x_i \le 35$.
The global minima is located at origin $x = (0, \cdots, 0)$, $f(x) = 0$
where $a = 20$, $b = 0.2$ and $c = 2\pi$

2.10.1.2. Bohachevsky Functions

$$f_1(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \tag{3-7a}$$

subject to $-100 \le x_i \le 100$.
The global minimum is located at $x = f(0, 0)$, $f(x) = 0$

$$f_2(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3 \tag{3-7b}$$

subject to $-100 \le x_i \le 100$.
The global minimum is located at $x = f(0, 0)$, $f(x) = 0$

$$f_3(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3 \tag{3-7c}$$

subject to $-100 \le x_i \le 100$.
The global minimum is located at $x = f(0, 0)$, $f(x) = 0$

2.10.1.3. Sphere Function

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{3-8}$$

subject to $0 \le x_i \le 10$.
The global minima is located $x = f(0, \cdots, 0)$, $f(x) = 0$

2.10.1.4. Rastrigin Function

$$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)] \tag{3-9}$$

2.10.1.5. Fifth Function of De Jong

$$f(x) = \left(0.002 + \sum_{i=1}^{25}\frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6}\right)^{-1} \tag{3-10}$$

where a_1 and a_2 = -32 to 32

2.10.2. DTLZ Many Objectives Test Functions

The DTLZ suite of benchmark problems, created by Deb et al., is unlike the majority of multi-objective test problems in that the problems are scalable to any number of objectives. This is an important characteristic that has facilitated several recent investigations into what are commonly called "many" objective problems. DTLZ1-DTLZ5 are scalable with respect to the number of distance parameters but have a fixed number of *M*-1 position parameters, where *M* is the number of objectives. Note also that the objective functions of DTLZ1–DTLZ4 have multiple global optima since terms such as $\cos(y_i\pi/2)$ can evaluate to zero, thereby allowing flexibility in the selection of other parameter values. Technically speaking, these objectives are non-separable, as attempting to optimize them one parameter at a time (in only one pass) will not identify all global optima. As this is a minor point, one can classify the objectives of DTLZ1–DTLZ4 as being separable irrespective, as attempting to optimize them one parameter at a time will identify at least one global optima. Incidentally, there being multiple global optima is why many of the DTLZ problems are Pareto many-to-one.

DTLZ5 is claimed to be problems with degenerate Pareto optimal fronts; the Pareto optimal fronts are meant to be an arc embedded in *M*-objective space. However, it has been found that this is untrue for instances with four or more objectives. The problem arises from the expectation that the minimization of (when $g = 0$) results in a Pareto optimal solution. Table 1 shows five of the DTLZ Many Objective Problems.

Table 1. Five of the DTLZ Many Objective Problems

Name	Problem	Parameter Domains
DTLZ1	$f_1 = (1+g)0.5\prod_{i=1}^{M-1} y_i$ $f_{m=2:M-1} = (1+g)0.5(\prod_{i=1}^{M-m} y_i)(1-y_{M-m+1})$ $f_M = (1+g)0.5(1-y_1)$ $g = 100\left[k+\sum_{i=1}^{k}\left((z_i-0.2)^2-\cos(20\pi(z_i-0.5))\right)\right]$	[0, 1]
DTLZ2	$f_1 = (1+g)\prod_{i=1}^{M-1}\cos(y_i\pi/2)$ $f_{m=2:M-1} = (1+g)\left(\prod_{i=1}^{M=m}\cos(y_i\pi/2)\right)\sin(y_{M-m+1}\pi/2)$ $f_M = (1+g)\sin(y_1\pi/2)$ $g = \sum_{i=1}^{k}(z_i-0.5)^2$	[0, 1]
DTLZ3	As DTLZ2, except the equation for g is replaced by the one from DTLZ1.	[0, 1]
DTLZ4	As DTLZ2, except all $y_i \in y$ are replaced by y_i^{α} ,where $\alpha > 0$.	[0, 1]
DTLZ5	As DTLZ2, except all $y_2,\ldots,y_{M-1} \in y$ are replaced by $\frac{1+2_{gy_i}}{2(1+g)}$.	[0, 1]

3. RESULTS AND ANALYSIS

The evaluation of the developed technique was grounded on the operation simulation. The estimation of the running time and the minimum fitness values were composed of five mathematical test functions.

3.1. The Experiment Setup

In order to test the effectiveness of IGA, AIS, and GA when solving timetabling problems, a comparison with the PSO algorithm was performed to investigate trends of performance. All coding was written in MATLAB, and the test case focused on the four above algorithms. All tests were executed on a 3.30 Ghz Intel core i5 processor with 16 GB of ram. The convergence graph for IGA, AIS, GA, and PSO below shows progress until a valid solution for each of the algorithms were discovered. The specifications of the problem are shown in Table 2.

Table 2. General data used by the algorithm

No	Operator	Quantity/Type
1	Number of individuals	1000
2	Crossover probability	0.7
3	Mutation probability	0.2
4	Number of generations	1000
5	Selection mechanism	Tournament selection
6	Crossover type	Two point crossover

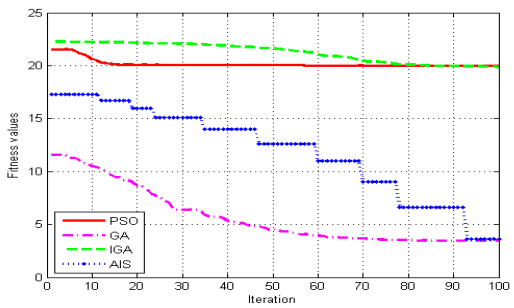
Table 3. Comparison of run time with small size problems

Generation	GA	AIS	IGA	PSO
100	11 (sec)	10 (sec)	8 (sec)	9 (sec)
200	23 (sec)	22 (sec)	20 (sec)	23 (sec)
400	46 (sec)	39 (sec)	37 (sec)	39 (sec)
800	95 (sec)	80 (sec)	78 (sec)	82 (sec)
1000	121 (sec)	92 (sec)	85 (sec)	93 (sec)

In the PSO calculation, the practice set size of population for GA, AIS, and IGA with $c_1 = 2$, $c_2 = 2$, was $w = 1 / (2 \times \log_2)$. In the GA and IGA calculation, mutation rate = 0.2 and crossover = 0.7. By observing the simulation results and the graphs, it is inferred that, for most of the populations, there was no great difference between the execution times and fitness values for the GA, AIS, IGA and PSO. Until the population reached 500, the IGA had an increase of time whereas PSO remained linear. However, the GA and IGA fitness values rose more than PSO when the population increased. In AIS, it still has less fitness values at the beginning but then rose higher when the population increased. This might stem from the fact that the parameters of GA, AIS and IGA were different from PSO such as velocity, global and local search space, and so on. Over all, when comparing between IGA and GA, it is cleared to see that IGA had less fitness than GA but IGA consumed less time to reach the optimal solution than GA. From Table 3, it is clear to see that IGA greatly reduced the running time. Off course, the search space could be less and also the generation of run time more effective. The results illustrated that the highest fitness value occurred when crossover probability was in the range of 0.7.

3.2. The Single Objective Mathematical Result of Experiment

In order to compare and evaluate different algorithms, various benchmark functions with various properties were used. Five test functions were used in this research as follows, Ackley, Bohachevsky, Sphere, Rastrigin, and the fifth function of De Jong were compared between GA, AIS, IGA, and PSO. The following are the five functions as presented in section 2.



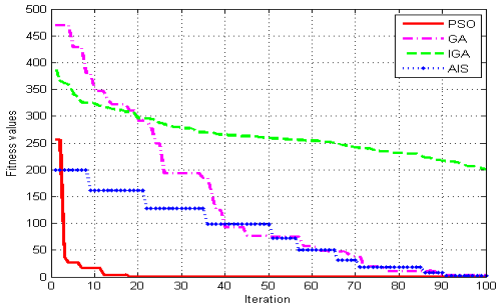


Figure 8. The comparison between fitness Ackley function (left) and Bohachevsky functions (right)

Figure 8 shows the results between the hybrid algorithms compared to other algorithms with various mathematical functions as presented in section 2. It is cleared to see that the hybrid algorithm performed better than other techniques. Surprisingly, the Ackley function performed closely to the hybrid algorithm. Figure 9 to Figure 10 shows that PSO reached the optimal result very quickly because this algorithm works as a local search which makes a narrow space for the search of a solution, rather than other algorithms which work as a global search space.

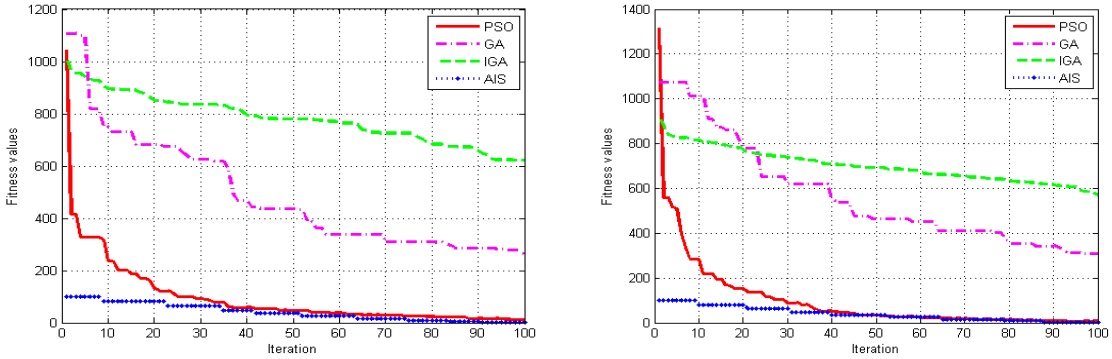


Figure 9. The comparison between fitness Sphere function (left) and Rastrigin function (right)

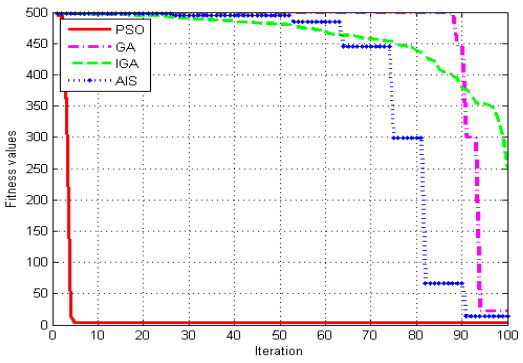


Figure 10 The comparison between fitness fifth function of De Jong

3.3. The Multi Objective Mathematical Result of Experiment

For the next investigation, the DTLZ1 – DTLZ5 multi-objectives mathematical functions described in section 2, will be used to study the behavior of the algorithms. The test function focused on two objective functions to compute the best individual fitness values from each iteration, the algorithm parameters are shown in Table 1. The DTLZ1 test function results are shown in Figure 11 to Figure12 show the testing revealed that IGA performed better than others followed by AIS and GA, the worse was PSO.

The results from the DTLZ1 testing function are shown in Figures 11 to 12. Figure 11 shows that the GA solutions were found to be very close when the algorithm started iterations and once nearly finished, whereas the IGA found the solutions at the beginning of the iterations. Figure 12 (left) depicts the solutions were more widely spread because of AIS function of searching all of the dimension search space. Although shown in Figure 12 (right), the PSO algorithm reached solutions very quickly by using a narrow search because this algorithm was locked at a local space. To summarize the DTLZ1 testing function, it is clear to see that IGA performed the highest fitness value while GA and AIS took longer times to complete.

Figure 13 to Figure 14 depicts the results from the DTLZ2 testing function. In this test suit, the IGA found a continual solution as shown in Figure 13 (right). The AIS and PSO had a similar result in this test which was widely spread. GA, in Figure 13 (left) achieved an inferior result. A few mutations could not find

the solution. To summarize the DTLZ2 testing function, all algorithms performed the same fitness values but IGA reached a higher number of solutions than the others.

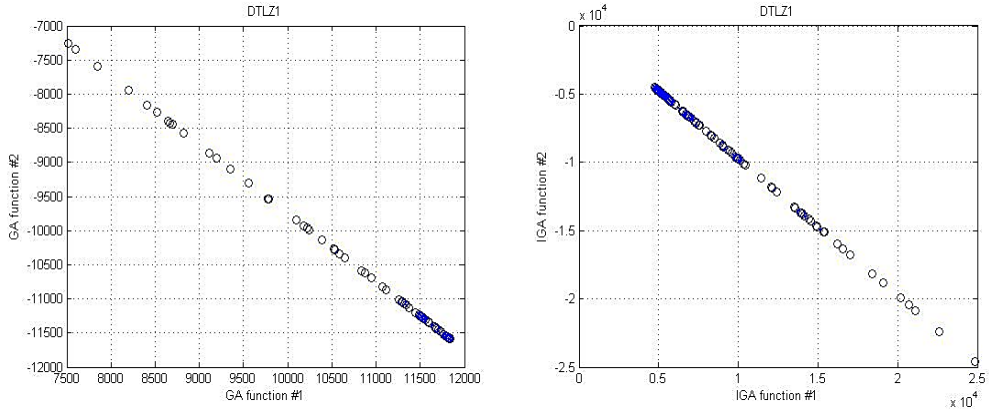


Figure 11. The DTLZ1 test function coded by GA algorithm (left) and IGA algorithm (right)

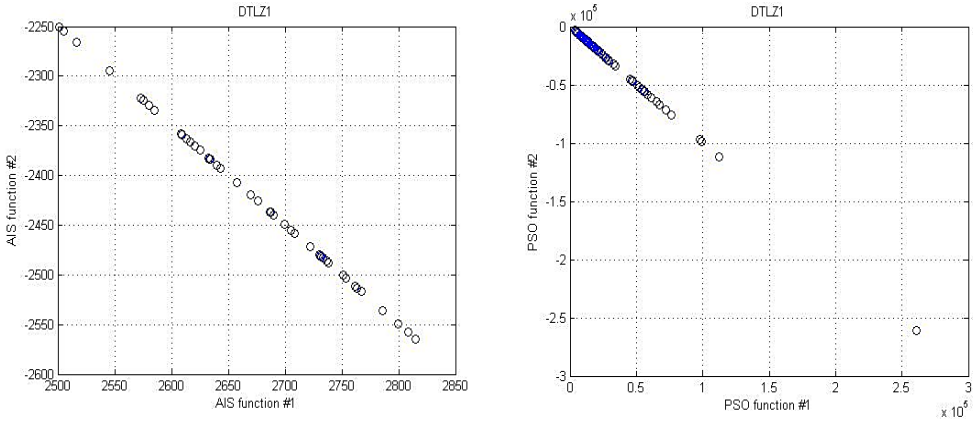


Figure 12. The DTLZ1 test function coded by AIS algorithm (left) and PSO algorithm (right)

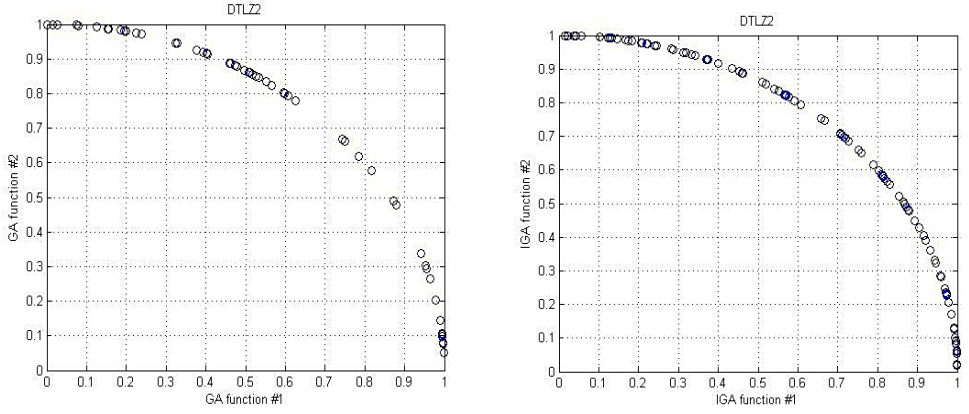


Figure 13. The DTLZ2 test function coded by GA algorithm (left) and IGA algorithm (right)

Figure 15 to Figure 16 depicts the results from the DTLZ3 testing function. Figure 15 shows that the GA solutions were found continually, similar to PSO (shown in Figure 16). In Figure 16, the AIS did not find a high number of solutions at the first iteration but reached an optimal number of solutions when iterations

were close to the finish. Finally, IGA in Figure 15 shows the solutions were found continually and widely spread throughout the search space. It is clear to see that in the DTLZ3, the IGA still outperformed the other algorithms.

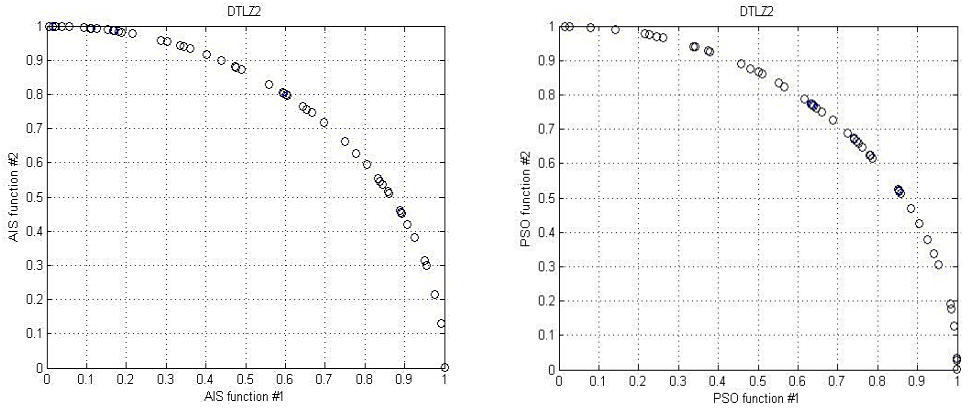


Figure 14. The DTLZ2 test function coded by AIS algorithm (left) and PSO algorithm (right)

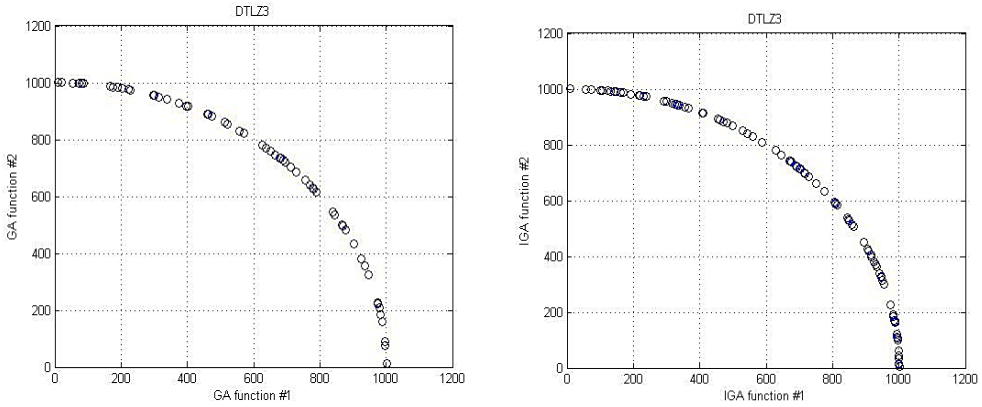


Figure 15. The DTLZ3 test function coded by GA algorithm (left) and IGA algorithm (right)

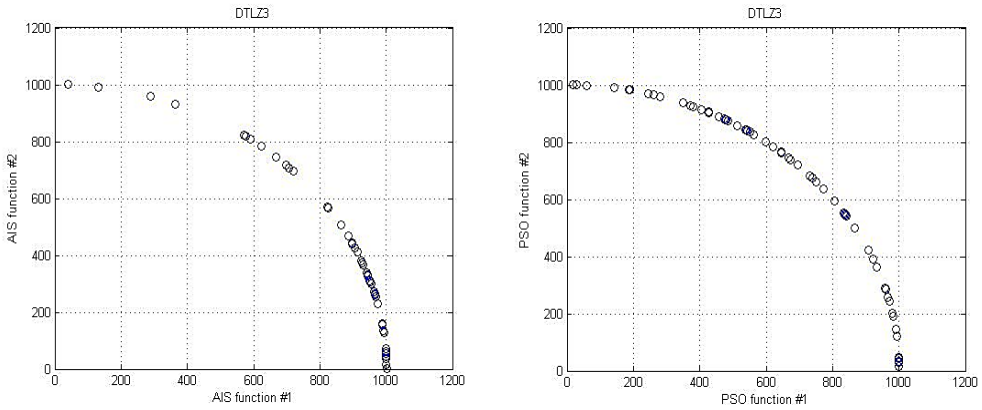


Figure 16. The DTLZ3 test function coded by AIS algorithm (left) and PSO algorithm (right)

Figure 17 shows the DTLZ4 test function coded by GA algorithm (left) and IGA algorithm (right)

Figure 18 shows the DTLZ4 test function coded by AIS algorithm (left) and PSO algorithm (right)

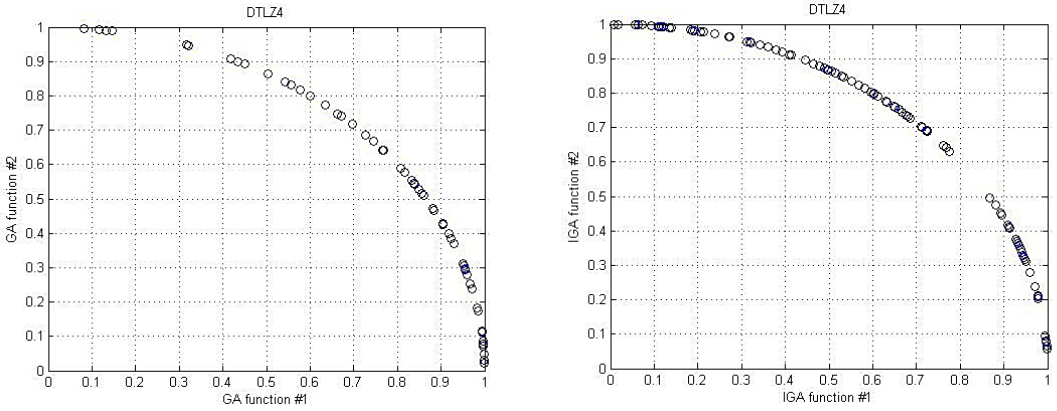


Figure 17. The DTLZ4 test function coded by GA algorithm (left) and IGA algorithm (right)

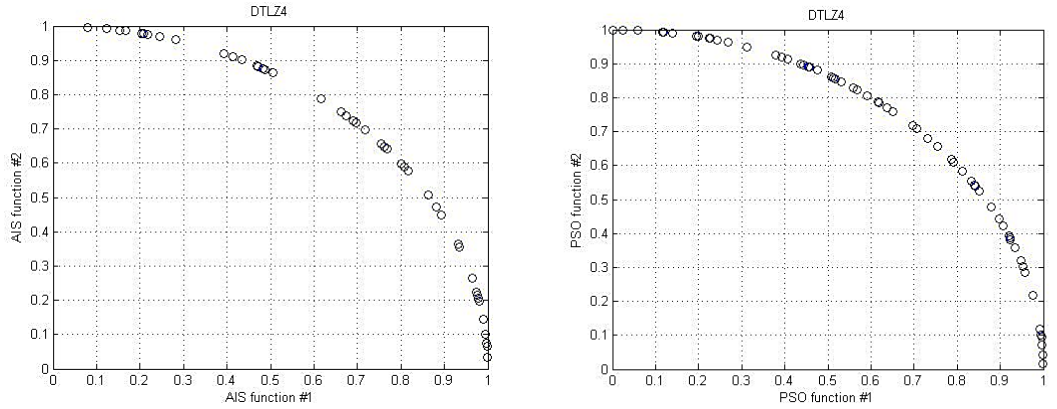


Figure 18. The DTLZ4 test function coded by AIS algorithm (left) and PSO algorithm (right)

DTLZ5 clearly demonstrated the suitability of IGA which is proven in Figures 19 to 20. The results show that the IGA indeed has the ability to maintain different mathematical testing functions. The highest frequency of the solutions was found by IGA, while AIS spread the optimal points from the search space. The PSO performed similar to the GA, even though PSO was tested with a local search and the GA within a global search.

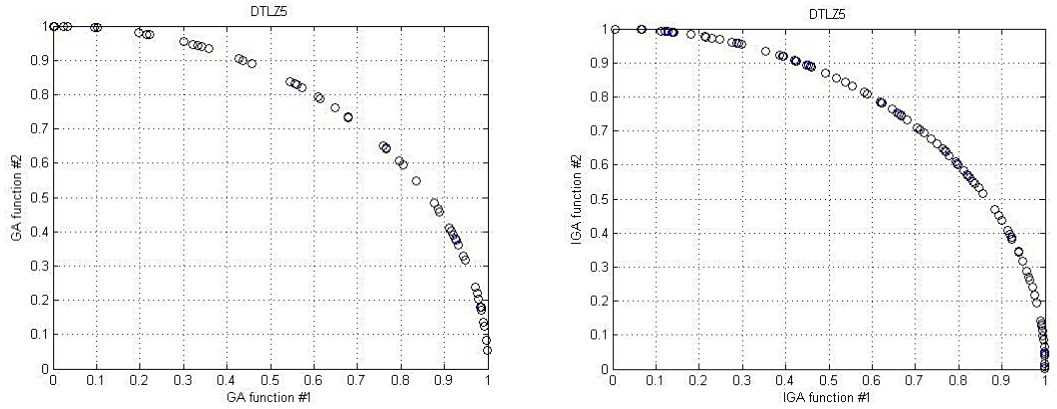


Figure 19. The DTLZ5 test function coded by GA algorithm (left) and IGA algorithm (right)

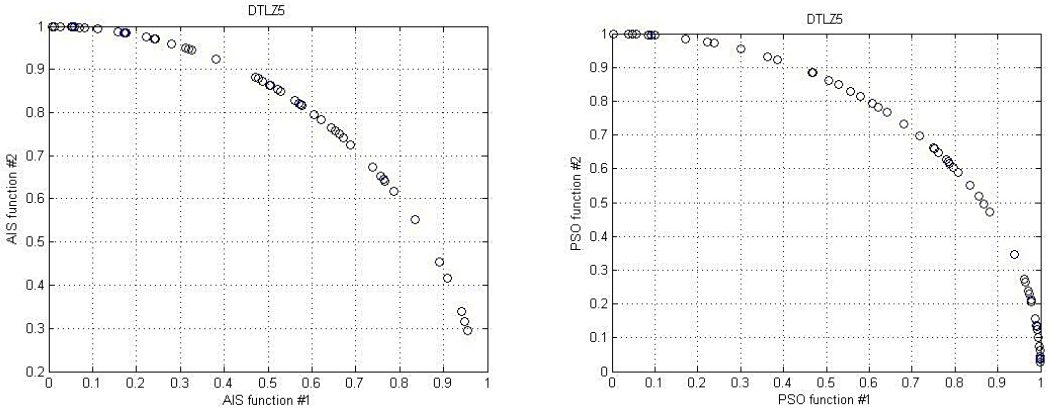


Figure 20. The DTLZ5 test function coded by AIS algorithm (left) and PSO algorithm (right)

4. CONCLUSION

This paper presented a modified method from a Genetic Algorithm (GA) and the Artificial Immune System (AIS) called the Immune Genetic Algorithm (IGA). Although Genetic Algorithms can rapidly locate the region in which the global optimum exists, they take a relatively long time to locate the optimum in the region of convergence. In practice, the population size is finite, which influences the sampling ability of a genetic algorithm and as a result affects its performance. Incorporating a negative selection method within a genetic algorithm can help to overcome most of the obstacles that arise as a result of finite population sizes. Due to the GA limited population size, a Genetic Algorithm may also sample bad representatives of good search regions and good representatives of bad regions. A negative selection method can ensure fair representation of the different search areas by sampling their self and nonself antigen which in turn can reduce the possibility of population size. The new modified algorithm could be used to improve the quality of initial solutions which are generated randomly.

Negative selection which is one of techniques in the Artificial Immune System, was employed to determine the input variables (populations) of the system. Basic concepts of negative selection theory, especially, the concept of attribute reduction were also used to define the chromosome populations. In order to illustrate the effectiveness of the Immune Genetic Algorithm, the comparison with a steady-state genetic algorithm, artificial immune system, and particle swarm optimization were also investigated.

The testing of the performance was conducted in mathematical testing, problems were divided into single and multiple objectives. The five single objectives were then used to test the modified algorithm, the results showed that IGA performed better than all of the other methods. The DTLZ multi-objective testing functions were then used. The result also illustrated that the modified approach still had the best performance. Thus, suggested optimal attribute crossover is 0.7 and mutation rate is 0.2. The selection mechanism would be tournament selection with a value set of 0.2, this would find the optimal fitness values efficiently.

REFERENCES

[1] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] T. A. El-Mihoub, A. Hopgood, I. A. Aref, *"Self-Adaptive Hybrid Genetic Algorithm using an Ant-based Algorithm"*, in 2014 IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), 2014, pp. 166-171.

[3] J. Sharma, R. S. Singhal, *"Comparative Research on Genetic Algorithm, Particle Swarm Optimization and Hybrid GA-PSO"*, in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 110-114.

[4] J. Zhao, H. Xu, W. Li, *"A Hybrid Genetic Algorithm for Bayesian Network Optimization"*, in The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014), 2014, pp. 906-910.

[5] J. Wu, Z. Lu, *"A Novel Hybrid Genetic Algorithm and Simulated annealing for Feature Selection and Kernel Optimization in Support Vector Regression"*, in 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), 2012, pp. 999-1003.

[6] L. N. D. Castro, J. Timmis, *"Artificial Immune Systems: A Novel Paradigm to Pattern Recognition"*, in University of Paisley, 2002, pp. 67-84.

[7] Y. Zhu, X. Qiu, *"A Hybrid AIS-based Algorithm for Solving Job Shop Scheduling Problem"*, in 2012 International Conference on Communication Systems and Network Technologies, 2012, pp. 498-502.
[8] F. Barani, *"A Hybrid Approach for Dynamic Intrusion Detection in ad hoc Networks using Genetic Algorithm and Artificial Immune System"*, in 2014 Iranian Conference on Intelligent Systems (ICIS), 2014, pp. 1-6.
[9] M. O. Ali, S. P. Koh, K. H. Chong, D. F. W. Yap, *"Hybrid Artificial Immune System-Genetic Algorithm Optimization based on Mathematical Test Functions"*, in 2010 IEEE Student Conference on Research and Development (SCOReD), 2010, pp. 256-261.
[10] Z. Ji D, Dasgupta, "Revisiting Negative Selection Algorithms," *Evol. Comput.*, vol. 15, no. 2, pp. 223–251, Jun. 2007.
[11] T. Stibor, K. M. Bayarou, C. Eckert, "An Investigation of R-Chunk Detector Generation on Higher Alphabets," in *Genetic and Evolutionary Computation-GECCO 2004*, 2004, pp. 299-307.