ISSN 1823-626X

Malaysian Journal of Fundamental and Applied Sciences

available online at http://mjfas.ibnusina.utm.my

Comparison between Genetic Algorithm and Prey-Predator Algorithm

Surafel Luleseged Tilahun[1]* and Hong Choon Ong[2]

[1]*Interdeciplinary Program, Addis Ababa Science and Technology University, 16417, Addis Ababa, Ethiopia*
[2]*School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia*

Received 16 February 2013, Revised 29 April 2013, Accepted 15 May 2013, Available online 20 May 2013

ABSTRACT

Metaheuristic algorithms are useful in solving complex optimization problems. Genetic algorithm (GA) is one of the well known and oldest metaheuristic algorithms. It was introduced in 1975 and has been used in many applications varying from engineering to management and many other fields as well. However, Prey-Predator algorithm (PPA) is one of recently introduced algorithm, in 2012, inspired by the interaction between preys and their predator. The motivation and the search mechanism for these two algorithms are different. In this paper the comparison of these two algorithms both from theoretical aspects and using simulation on selected benchmark problems is presented. According to the results, PPA performs better than GA in the selected test problems.

| Genetic algorithm (GA) | Prey-Predator algorithm (PPA) | Metaheuristic algorithms | Optimization |

® 2013 Ibnu Sina Institute. All rights reserved.
http://dx.doi.org/10.11113/mjfas.v9n4.104

1. INTRODUCTION

Metaheuristic algorithms are a non-deterministic solution method for optimization problems. Even though these solution methods don’t guarantee optimality of solutions they are used widely in many applications [1]. They have been used in transportation problems [2], engineering applications [3,4], decision making and many other applications [5-8]. The fact that they are not affected much by the behaviour of the problem makes them popular for being used in different disciplines.

Genetic algorithm (GA) is one of the first metaheuristic algorithms introduced in mid 1970’s [9]. It is inspired by Darwin’s theory of survival of the fittest. After the introduction of GA many algorithms have been developed and used. Even though the use of the word heuristic can be tracked back to Alan Turing during the second world war (1939-1945), the word metaheuristic was first used by Glover 1986 [10]. Most of these algorithms are inspired by a natural phenomenon. Currently there are more than 25 metaheuristic algorithms. Perhaps Prey-Predator Algorithm is the most recent algorithm [11]. It mimics the interaction between a predator animal and its preys.

In this paper genetic algorithm will be compared with Prey-Predator Algorithm. The comparison is done from the theoretical aspects as well as using simulation results. In the next section basic concepts will be discussed followed by a comparison section in section 3. The conclusion will be given in section 4.

2. BASIC CONCEPTS

2.1 Optimization Problems

An optimization problem is a problem to find a value for the variables of a function which maximizes or minimizes the function. One can switch between minimization and maximization problems by multiplying the function by negative one, hence without loss of generality we consider a maximization problem.

$$\max_{x\in S\subseteq \Re^n} f(x)$$

The function, $f(x)$, that needs to be maximized is called the objective function, the variables, x, are called decision variables and the set of possible values for the decision variables, S, is called feasible set.

A solution for a maximization problem is a value from the feasible set which maximizes the objective function.

2.2 Genetic Algorithm (GA)

Genetic algorithm (GA) is one of the oldest and widely used metaheuristic algorithms. The algorithm is inspired by Darwin's theory of survival of the fittest. It gives less probability for the unfit solutions to pass to the next generation. In the algorithm, randomly generated solutions will be decoded using 0’s and 1’s, as they are called chromosomes. Their fitness will be computed depending on their performances in the objective function. By giving high

*Corresponding author. E-mail: surafelaau@yahoo.com probability to the fittest solution members to be chosen crossover and mutation will be done. Crossover is one of the GA operators in which two parent solutions will switch part of their chromosomes whereas mutation is the other GA operator which changes a chromosome either from 0 to 1 or 1 to 0. The fittest solutions will pass to the next generation and this iteration will continue until a termination criterion is fulfilled. The main steps of the algorithm are discussed as follows:

1. Generate *k* random solutions
2. Compute the fitness of each solutions.
3. Choose parents for cross-over and mutation.
4. Perform cross-over and mutation.
5. Select the *k* fittest solutions from the parent and child solutions and construct a solution set.
6. If a termination criterion is met stop else go to step 2.

The algorithm has mainly three algorithm parameter, namely probability of cross-over, probability of mutation and chromosome size to represent the solution.

2.3 Prey-Predator Algorithm (PPA)

Prey-Predator algorithm (PPA) is a metaheuristic algorithm developed for optimization problems introduced by Tilahun and Ong [11]. It mimics how a predator runs after its preys and the preys run away to escape. In the algorithm randomly generated solutions will be assigned as a predator and preys depending on their performance in the objective function. A solution with least performance will be assigned as a predator and the others as preys. A prey with best performance in the objective function will be called best prey. After the assignment of predator and preys, the preys will run away from the predator and follow preys with better performance. The predator does the exploration by running randomly and chasing the prey with least performance. The best prey in the other hand does only a local search for exploitation purpose. The main steps of the algorithm are as follows:

1. Generate *k* random solutions
2. Assign predator, preys and the best prey; depending on measure of their performance on the objective function called survival value, *SV*.
3. Move the best prey, the prey with best *SV* in local search using the randomly generated *m* directions.
4. If probability of follow-up is met move the preys by following the better preys and also with some local search.
   If probability of follow-up is not met move the preys randomly away from the predator.
5. Move the predator towards the prey with least survival value and also randomly.
6. Update list of preys, best prey and predator.
7. If a termination criterion is met stop else go to step 3.

The parameters for PPA are $\lambda_{\max}$ and $\lambda_{\min}$ as a jump for exploration and exploitation or for local search, respectively. Hence $\lambda_{\max} > \lambda_{\min}$. The probability of follow-up, p_f , which guides the intensity of exploration and exploitation. Higher value of probability of follow-up results in more exploitation than exploration. The other algorithm parameter is number of random directions for local search, *m*.

3. COMPARISON OF GA AND PPA

3.1 Theoretical Comparison

In both algorithms, GA and PPA, there is a concept of survival of the fittest. Apart from this similarity the two algorithms have completely different search mechanisms. For the purpose of updating solutions, GA uses crossover and mutation operators which are different from the distance operator used in PPA. For this, in GA the solutions need to be encoded using 0's and 1's. However in PPA there is no need for encoding and decoding of solutions using 0's and 1's. PPA uses the concept of following and exploring by running away from the predator. In GA, the mutation operator is the operator which helps the solutions not to be trapped in local optimal solutions, whereas in PPA the predator force other solution members to explore the solution space to avoid being trapped in a local optimal solution. The motivation for GA is also quite different from PPA. Furthermore, the solution with worst performance in the objective function doesn't play any role in GA, which is not the case in PPA. Furthermore, unlike PPA, GA works on a grid based space which depends on the chromosome size and is unable to attain some solutions in the solution space.

3.2 Comparison Based On Simulation Results

A simulation comparison is done using two well known bench mark problems. For each problem the same 30 randomly generated feasible initial solutions are used for both algorithms. Probability of follow-up, $\lambda_{\max}$ and $\lambda_{\min}$ for PPA is taken to be 0.6, 4 and 1.2 with 20 number of random directions for local search. For GA the probability of cross-over and mutation is taken to be 0.8 and 0.4 whereas the chromosome size was set to be 20 for each variable. A statistical analysis used to compare the result of the two algorithms with a trial number of 100.

a) The first test problem

A two dimensional stochastic unimodal function is the objective function of the first test problem. It has a noise parameter which makes the function to give the probability that it will not attain the same value at a point [12]. The problem is given as follows:

$$\max\ f_1(x_1,x_2)=5e^{-(x_1-\pi)^2-(x_2-\pi)^2}+\sum_{j=1}^{2}\sum_{i=1}^{2}\varepsilon_{ij}e^{-(x_1-i)^2-(x_2-j)^2}$$
$$\text{s.t.}\quad 0\le x_1,x_2\le 10$$

where ε_{ij} is a random number from a uniform distribution between 0 and 1. The optimal value is between 5 and 5.0794 depending on the random number and is found at (π,π).

After running the PPA the average solution is found to be 5.0541 with a standard deviation of 0.0219, whereas the average best functional value and its standard deviation using GA is found to be 4.9675 and 0.0621, respectively.

b) *The second test problem*

The objective function of the second test problem is negative of Shubert's function [13]. It is given by:

$$\max\ f_2(x_1,x_2)=-[\sum_{i=1}^{5}i\cos((i+1)x_1+i)][\sum_{i=1}^{5}i\cos((i+1)x_2+i)]$$
$$\text{s.t.}\quad \text{-}10\le x_1,x_2\le 10$$

It is a multimodal benchmark problem with optimum functional value of 186.73067 found in 18 points. It also has 720 local solutions.

The mean final functional values after the 100 trials are found to be 186.5112 and 185.1899 with standard deviation of 0.0551 and 0.18 for PPA and GA respectively.

c) *The third test problem*

Negative of Michalewicz's function is the objective function of the third test problem [13]. The problem is a multimodal problem with k! local optimum solutions, for k number of dimension of the decision variable, here the dimension k is taken to be 5. Hence there are 120 solutions. And the optimal functional value is 4.687. The problem is given as in below:

$$\max\ f_3(x_1,x_2,x_3,x_4,x_5)=\sum_{i=1}^{5}\sin x_i(\sin\frac{ix_i^2}{\pi})^{20}$$
$$\text{s.t}\quad \text{-}5.12\le x_1,x_2,x_3,x_4,x_5\le 5.12$$

After simulation the mean performance in terms of the final objective functional value is found to be 4.6954 and 4.5984 with standard deviation of 0.0465 and 0.1328 for PPA and GA respectively.

The simulation results are summarized in table 1.

Table 1 Simulation results

Test problem	**Mean functional value**		**Standard deviation**		**Average CPU time**	
	GA	**PPA**	**GA**	**PPA**	**GA**	**PPA**
f₁	4.9675	5.0541	0.0621	0.0219	0.2496	0.2028
f₂	185.1899	186.5112	0.18	0.551	0.1105	0.0995
f₃	4.5984	4.6954	0.1328	0.0465	0.1564	0.1185

From the numerical results, PPA is efficient for solving the three test problems problem with smaller CPU time and also with better average functional values. Furthermore, GA has high variability than PPA as the standard deviation for GA is higher than PPA. This implies that at a random run of the algorithms PPA will achieve the average value with less deviation compared to GA.

4. CONCLUSION

In this paper a comparison is done between the well known genetic algorithm (GA) and the recently introduced Prey-Predator algorithm (PPA). The comparison is done both from theoretical aspects and using simulation results. From the theoretical comparison the main similarity and difference of the two algorithms are discussed. While in the simulation comparison three benchmark problems are used and a simulation is done 100 times. From the simulation results PPA is found to be efficient and fast with less variability compared to GA.

ACKNOWLEDGEMENT

This work was supported in part by U.S.M. Fundamental Research Grant Scheme (FRGS) No. 203/PMATHS/6711319.

REFERENCES

[1] S. Edelkamp and S. Schrodl, Heuristic Search: Theory and Applications. Morgan Kaufmann Publisher, MA, 2012

[2] S. L. Tilahun and H. C. Ong, Promet – Traffic & Transportation 24 (3)(2012) 183-191.

[3] A. A. Adewuya, New Methods in Genetic Search with Real-Valued Chromosomes. B.s. thesis, Department of Mechanical Engineering, Mississippi State University, USA, 1993.

[4] Y. Tenne, and S. W. Armfield, Evolutionary Computation in Dynamic and Uncertain Environments Studies in Computational Intelligence, 51 (2007) 389-415

[5] X. Luoa, Q.-Y. Wenc and G. Fieg., Computers & Chemical Engineering, 33 (6) (2009) 1169–1181

[6] X. Wang, J.-J. Ma, S. Wang and D.-W. Bi, Sensors, 7 (2007) 628-648

[7] E. Piazza, Applications of Evolutionary Computing Lecture Notes in Computer Science, 2037 (2001) 248-256

[8] E. W. Richards and E. A. Gunn, Canadian Journal of Forest Research, 33(6) (2003) 1126-1133

[9] M. Negnevitsky, “Artificial Intelligence: A Guide to Intelligent System”, Henry Ling, Harlow, (2005).

[10] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, Frome, UK, 2010

[11] S. L. Tilahun, Prey-Predator Algorithm: A new metaheuristic optimization approach, A PhD thesis submitted to School of Mathematical Sciences, Universiti Sains Malaysia, January 2013.

[12] X.-S.Yang, Engineering optimization: An Introduction with Metaheuristic Applications (2nd. Edition),. John Wiley and Sons, 2010.

[13] M. Molga and C. Smutnicki, Test functions for optimization needs, Online accessed 3rd Feb 2012, from: http://www.bioinformaticslaboratory.nl/ twikidata/ pub/Education/ NBICResearchSchool/Optimization/VanKampen/BackgroundInformation/TestFunctions-Optimization.pdf. (2005)