Research in Random Parameters of Genetic Algorithm and Its Application on TSP and Optimization Problems

Mohammad Masoud JAVIDI[*], Roghiyeh Hoseinpour FARD and Mahdi JAMPOUR

Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran

([*]Corresponding author's e-mail: javidi@uk.ac.ir)

Received: 25 December 2012, Revised: 8 May 2013, Accepted: 29 June 2013

Abstract

In this paper, we consider a variety of random parameters of genetic algorithms based on some benchmark functions and traveling salesman problem (TSP). We have analyzed parameters of the genetic algorithm such as population size, crossover probability and mutation probability. The experiments have shown that we cannot propose a uniform model for the parameters of a genetic algorithm. However increasing of population size can reduce genetic algorithm iterations but crossover probability and also mutation probability strongly depend on benchmark functions.

Keywords: Genetic algorithm, Schaffer, Ackley, Rastrigin, traveling salesman problem

Introduction

The optimization problem is finding an alternative with the most cost effective or highest achievable performance under the given constraints, by maximizing desired factors and minimizing undesired ones [1]. The goal of all optimization procedures is to obtain the best results possible (again, in some defined sense) subject to restrictions or constraints that are imposed. The first step in modern optimization is to obtain a mathematical description of the process or the system to be optimized. A mathematical model of the process or system is then formed on the basis of this description. Depending on the application, the model complexity can range from very simple to extremely complex.

During the past years many methods have been proposed for tackling the problem of global optimization. These methods can be divided in 2 main categories namely deterministic and stochastic [2]. Methods belonging to the first category are more difficult to implement and they depend on priory information about the objective function. On the other hand, stochastic methods are implemented more easily and they do not require priory information about the objective function [2]. Among the stochastic methods for global optimization we refer to Random Line Search [3], Adaptive Random Search [4], Competitive Evolution [5], Controlled Random Search [6], Simulated Annealing [7-10], Genetic Algorithms [11,12], Differential Evolution [13,14], Tabu Search [15] etc.

Genetic algorithms were developed by Holland at the University of Michigan in the early 1970's [16]. Genetic algorithms are probabilistic, robust and heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic [17]. Genetic algorithms are applied in numerous application areas mainly for optimization. The most common example that can be cited is the traveling salesman problem, in which the best shortest route has to be determined by optimizing the path.

GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. The genetic algorithm is applied by evaluating the fitness of all of the individuals in the population. Once the fitness function is evaluated, a new population is created by performing operations such as crossover, fitness-proportionate reproduction, and mutation on the individuals. Every time the iteration is performed, the old population is rejected and the iteration continues using the new population. This process leads to the evolution of better populations than the previous populations [18].

In this paper we examine 3 different functions in order to test specific parameters of GAs. So, we want to analyze the effect of the variation of random parameters on application problems for instance the traveling salesman problem and to evaluate the general computational behavior of GAs. This paper is organized as follows. Section II gives the outline of the genetic algorithm and traveling salesman problem. In section III and IV we describe the genetic algorithm for different functions and traveling salesman problem. Experimental results are present in section IV.

Genetic algorithm overview

Genetic algorithms are a family of computational models inspired by evolution. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. Genetic algorithms are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad. In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0 and 1 s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations have been produced, or a satisfactory fitness level has been reached for the population.

Selection: During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a *fitness-based* process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. The purpose of selection is to emphasize fitter individuals in the population. There are several types of selection methods used in genetic algorithms, including:

- Rank-based fitness assignment
- Roulette wheel selection
- Stochastic universal sampling
- Local selection
- Truncation selection
- Tournament selection

Crossover: The next step is to perform crossover. This operator selects genes from parent chromosomes and creates a new offspring. The simplest way of doing this is to randomly choose some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent [19]. The crossover operator is applied to the mating pool with the hope that it creates a better offspring [20]. The various crossover techniques are single-point, two-point, multi-point and uniform crossover.

Mutation: After crossover, the strings are subjected to mutation. Mutation prevents the algorithm becoming trapped in a local minimum. Mutation plays the role of recovering the lost genetic material as well as for randomly disturbing genetic information [20].

Elitism: The first best chromosome or the few best chromosomes are copied to the new population. The rest is done in a classical way. Such individuals can be lost if they are not selected to reproduce or if crossover or mutation destroys them. This significantly improves the GA's performance [20]. The structure of the GA is shown in **Figure 1**.

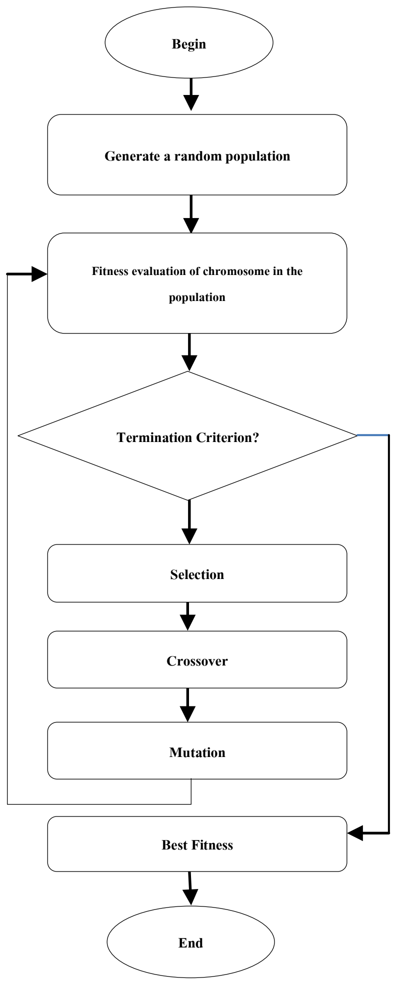
Begin
Generate a random population
Fitness evaluation of chromosome in the population
Termination Criterion?
Selection
Crossover
Mutation
Best Fitness
End

Figure 1 Structure of a traditional genetic algorithm.

Traveling salesman problem overview

The traveling salesman problem (TSP) is a well-known problem in combinatorial optimization. The aim of TSP is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized. Each possible tour is a permutation of 1, 2, 3, . . . n, where n is the number of cities, so therefore the number of tours is n!. The search path in the TSP problem can grow exponentially with the city number N, so it is a NP-complete problem, and to accurately determine the optimal solution is impossible [21]. There are several practical uses for this problem, such as vehicle routing (with the additional constraints of vehicle's route, such as the vehicles' capacity) and drilling problems [22]. Therefore, the mathematical model for TSP is as follows:

$$\begin{aligned}
&min \sum d_{ij}\, x_{ij}\\
&s.t. \quad \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1,2,\ldots,n\\
&\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1,2,\ldots,n\\
&\sum_{i,j\in s} x_{ij} \le |s| - 1 \;\; 2 \le |s| \le n-2 \quad s \in \{1,2,\ldots\; n\}\\
&x_{ij} = \begin{cases} 1 & if\ route(i,j)\ is\ selected\ . \\ 0 & othetwise \end{cases}
\end{aligned} \tag{1}$$

The TSP has received considerable attention over the last 2 decades and various approaches are proposed to solve the problem, such as branch-and-bound, cutting planes, 2-opt , simulated annealing, neural network, and tabu search [23]. Some of these methods are exact algorithms, while the others are near-optimal or approximate algorithms. The exact algorithms include the integer linear programming approaches with additional linear constraints to eliminate infeasible subtours [23]. On the other hand, network models yield appropriate methods that are flexible enough to include the precedence constraints [23]. More recently, genetic algorithm (GA) approaches have been successfully implemented to the TSP [24]. At present, there are many web sites discussing the traveling salesman problem, and have the benchmark in the standard TSPLIB format [25], such as eil51, st70, where the number behind the name represents the number of cities to be studied.

Table 1 Test functions.

Name	**Function**	**Search space**	**Optimal**
Ackley	$f(x,y) = 20 + e - 20\exp(-0.2\surd(1/2(x\hat{}2 + y\hat{}2))) - \exp(1/2(cos2\pi x + cos2\pi y)$	(-32,32)	22.2397
Schaffer	$f(x,y) = 0.5 + \frac{(sin\sqrt{x^2+y^2})^2 \;\; - 0.5}{(1+0.01(x^2+y^2)^2)^2}$	(-100,100)	0.9957
Rastrigin	$f(x,y) = 20 + (x^2+y^2) - 10cos2\pi(x+y)$	(-5.12,5.12)	80.7065

Table 2 Result of different crossover probability.

Pc%	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
Schaffer	441.5	372.3	422.7	657.6	558.4	625.2	622.4	712.1	835.1
Ackley	1418.4	1513.8	1576.3	1493.1	1607	1580.7	1632.7	1625.1	1713.3
Rastrigin	1534.8	1464.5	1443.2	1503.4	1583.6	1524	1377.9	1515.9	1500.6

Experimenting with simulation parameters

In this section, we use several functions in order to test specific parameters of the GAs and, then consider the effect the parameters have over TSP and evaluate the general computational behavior of GAs. These functions selected from several articles are listed in **Table 1**. It's noted that the functions are to be maximized. We have analyzed parameters of genetic algorithm such as population size, crossover probability and mutation probability.

Function test

In the evaluation, the length of the chromosomes is set to chrom=14; the number of generations is Max-gen = 2000 and the type of crossover is two-point crossover. In the experiment, we ran the GA 100 times and calculated the mean generation of the 100 run. The results of the experiments are given in **Tables 2 - 4**.

Crossover: The basic parameter in the crossover technique is the crossover probability (Pc). The crossover probability is a parameter to describe how often crossover will be performed [19]. The crossover in our experiment is applied with nine probability values $Pc \in [0.1,0.9]$ with a step of 0.1, and its application or non-application is defined by a random number generator. When it is not applied, the offspring is considered as an exact copy of one of the parents with equal probability [19]. **Figure 2** shows the average number of generations taken by the GA with different crossover probability.

Mutation: After a crossover is performed, mutation takes place. This is to prevent all solution populations falling into a local optimum of solving problem or local convergence [19]. Mutation is viewed as a background operator to maintain genetic diversity in the population. The mutation in our experiments is applied with 6 probability values (0.001, 0.01, 0.02, 0.03, 0.04, 0.05 and 0.06) and its application or non-application is defined by a random number generator. **Figure 3** shows the average number of generations taken by the GA with different mutation probability.

Population size: For each and every problem, the population size will depend on the complexity of the problem. The population size affects both the overall performance and the efficiency of GAs [19]. The larger the population is, the easier it is to explore the search space. But it has been established that the time required by a GA to converge is O (nlogn) function evaluations where n is the population size [19]. The population size in our experiments is with 10 values $Pop\_size \in [20,120]$ with step 20. **Figure 4** shows the average number of generations taken by the GA with different population size.

Table 3 Result of different crossover probability.

Pc%	10	20	30	40	50	60	70	80	90
Schaffer	441.5	372.3	422.7	657.6	558.4	625.2	622.4	712.1	835.1
Ackley	1418.4	1513.8	1576.3	1493.1	1607	1580.7	1632.7	1625.1	1713.3
Rastrigin	1534.8	1464.5	1443.2	1503.4	1583.6	1524	1377.9	1515.9	1500.6

Table 4 Result of different mutation probability.

Pm%	0.001	0.01	0.02	0.03	0.04	0.05	0.06
Schaffer	1075.5	810.03	832.3	831.9	565.3	625.2	698.2
Ackley	1996.7	1914.8	1806	1813.5	1726.7	1698.8	1576.7
Rastrigin	1986.6	1809.8	1811.8	1695.7	1562.1	1624.3	1412.2

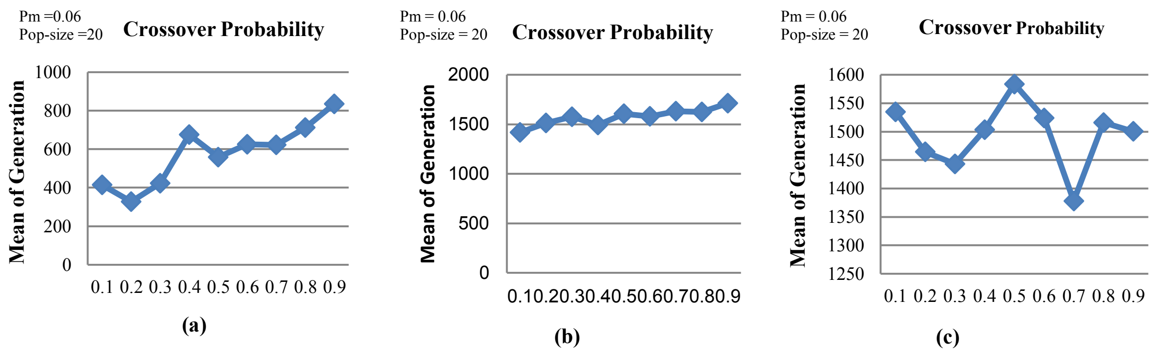

Figure 2 Mean of the generation of 3 test functions with different crossover probability. (a) Schaffer (b) Ackley and (c) Rastrigin.

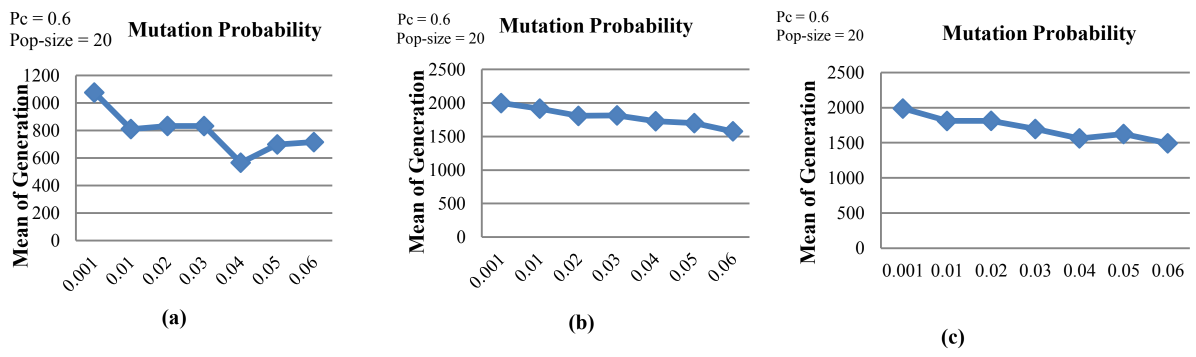

Figure 3 Mean of the generation of 3 test functions with different mutation probability. (a) Schaffer (b) Ackley and (c) Rastrigin.

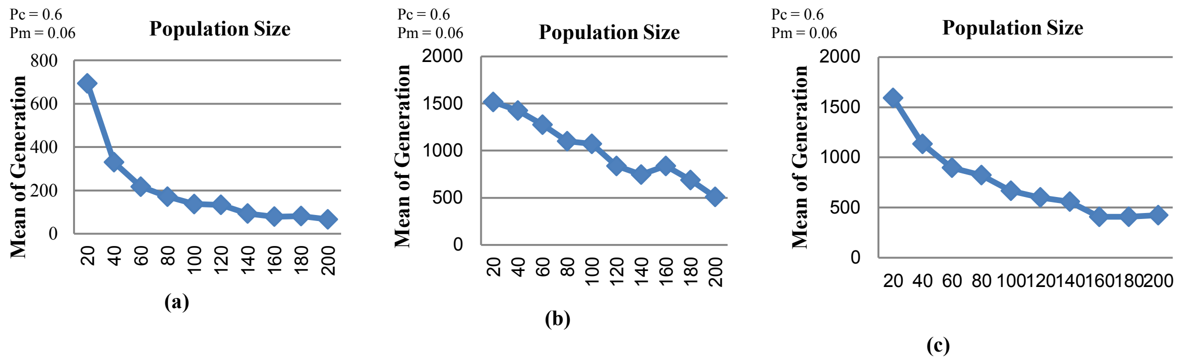

Figure 4 Mean of the generation of 3 test functions with different population size. (a) Schaffer (b) Ackley and (c) Rastrigin.

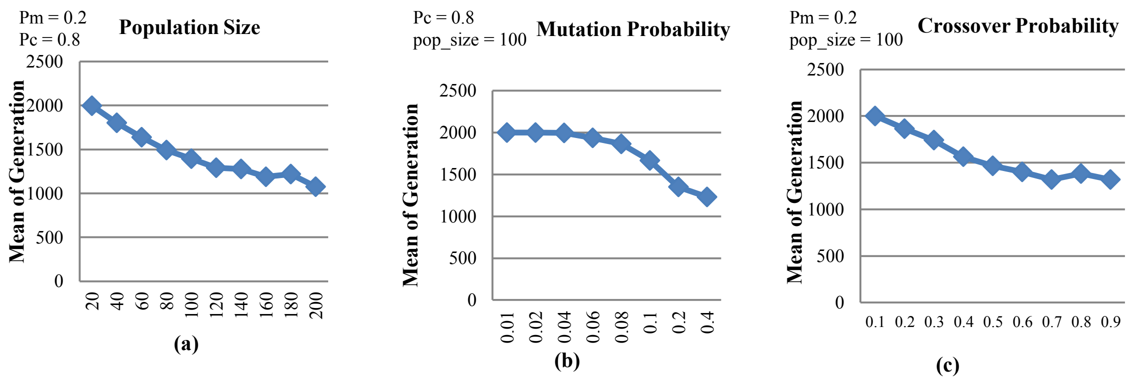

Figure 5 Mean of the generation of TSP with different probability (a) Population size (b) Mutation (c) Crossover.

Traveling salesman problem

The algorithm parameters of GA are that the number of generations is max-gen = 2000, the type of crossover is order-crossover and the type of mutation is 2-opt mutation. We used the TSP problem as a test with eil51, which has 51 cities. The optimal solution to this problem is known to be 426 m. In the experiment, we run the GA 100 times and calculate mean of generation of the 100 run. **Figure 5** shows the average number of generation taken by the GA with different population size, crossover probability and mutation probability.

Conclusions

In this study, the effect of variation of random parameters on the performance of the GA was compared. We used 3 different functions and a combinatorial optimization problem in this case the traveling salesman problem (TSP) in order to test specific parameters of GAs. The results show that the variation of random parameters of the genetic algorithm depends on the objective function and we cannot propose a constant pattern for random parameters of GA. Our results show that increasing, crossover rate leads to poor results for test functions but with increasing population size the value improves our results and also we could not find any robust rules with a variation of mutation rate. By attention to the results achieved from the variation of random parameters on TSP, we can say that the presence of even a small crossover rate improves our results. A large population requires more evaluations per generation, and consequently increases the computational time.

References

[1] CS Danalakshmi and GM Kumar. Optimization of supply chain network using genetic algorithm. *J. Manuf. Eng.* 2010; **3**, 30-5.
[2] IG Tsoulos and A Stavrakoudis. Enhancing PSO methods for global optimization. *J. App. Math. Comput*. 2010; **216**, 2988-3001.
[3] X Liu. The barrier attribute of filled functions. *J. Appl. Math. Comput*. 2004, **149**, 641-9.
[4] M Gaviano. *Some General Results on the Convergence of Random Search Algorithms in Minimization Problems*. *In*: LCW Dixon and GP Szegö (eds.). Towards Global Optimization, 1975, p. 149-57.
[5] HA Bremermann. A method for unconstrained global optimization. *Math. Biosci.* 1970; **9**, 1-15.
[6] RA Jarvis. Adaptive global search by the process of competitive evolution. *IEEE Trans. Syst. Man Cybern.* 1975; **75**, 297-311.
[7] WL Price. Global optimization by controlled random search. *J. Comput*. 1977; **20**, 367-70.

[8] S Kirkpatrick, CD Gelatt and MP Vecchi. Optimization by simulated annealing. *J. Stor*. 1983; **220**, 671-80.
[9] PJM van Laarhoven and EHL Aarts. *Simulated Annealing: Theory and Applications*. Springer, Boston, 1987, p. 1-186.
[10] A Corana, M Marchesi, C Martini and S Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Software* 1987; **13**, 262-80.
[11] WL Goffe, GD Ferrier and J Rogers. Global optimization of statistical functions with simulated annealing. *J. Econometrics* 1994; **60** ,65-100.
[12] D Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
[13] Z Michaelewizc. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.
[14] R Storn and K Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* 1997; **11**, 341-59.
[15] MM Ali and LP Fatti. A differential free point generation scheme in the differential evolution algorithm. *J. Global Optim.* 2006; **35**, 551-72.
[16] JH Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.
[17] TP Patalia and GR Kulkarni. Behavioral analysis of genetic algorithm for function optimization. *In*: Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research, Tamil Nadu, India, 2011, p. 1-5
[18] E Eiben, R Hinterding, Z Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* 1999; **3**, 124-41.
[19] SN Sivanandam and SN Deepa. *Introduction to Genetic Algorithm*. Springer, 2008, p. 1-442.
[20] P Bajpai and M Kumar. Genetic algorithm: An approach to solve global optimization problems. *J. Comput. Sci. Eng.* 2010; **1**, 199-206.
[21] D Hua-Fa, L Xiao-Lu and L Xue. An Improved Genetic Algorithm for Combinatorial Optimization. *In*: Proceedings of the IEEE International Conference on Computer Science and Automation Engineering, Shanghai, China, 2011, p. 58-61.
[22] G Laporte. The vehicle routing problem: an overview of exact and approximate algorithms. *Eur. J. Oper. Res.* 1992, **59**, 345-58.
[23] A Otman and A Jaafar. A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *J. Comput. Appl.* 2011; **31**, 49-57.
[24] G Carpaneto and P Toth. Some new branching and bounding criteria for the asymmetric traveling salesman problem. *Manag. Sci.* 1980; **26**, 736-43.
[25] TSPLIB, Institute of Information, University Heidelberg, Available at: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95, accessed February 2012.