International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 5 563 – 569

Performance Analysis of Selection Schemes in Genetic Algorithm for Solving Optimization Problem Using De Jong's Function1

Amit Wadhwa
Assistant Professor, Department of Computer Science & Engineering
Amity University Haryana, Gurgaon, India
awadhwa@ggn.amity.edu

Abstract—Genetic Algorithm (GA) is known to be a search algorithm based on idea of natural selection and survival of fittest [11]. The main idea behind the approach is, for a population of individuals adapting to some environment, they should behave naturally. Genetic Algorithm and other evolutionary algorithms are used over the years for finding solution to different emerging problems which could be better implemented and solved using approach of Genetic algorithm. Out of many such problems optimization problems are having their own importance and had a great scope for further research. There are different solutions given by different researchers for finding solutions to such optimization problems. This paper is about using De jong's function 1 adopted for finding a solution to an optimization problem and thereby comparing the performance of chosen selection schemes used in Genetic Algorithm

Keywords- *Genetic Algorithm, Evolutionary Algorithm, Selection, Crossover, Mutation Operators*

*****

I. INTRODUCTION

An GA belongs to family of evolutionary algorithms with genetic programming, evolution strategies, and evolutionary programming as its parts [31] [29]. Every algorithm mimicking its approach maintains a population of candidate solutions used to provide a solution to the problem at hand. Population then goes through an iterative process of with a set of operators usually consisting of encoding, selection, crossover, mutation and recombination. Over a span of time there are many algorithmic approaches being used for finding solution to number of different problems. Out of which many are based on or solved using approach of Genetic Algorithm.

Genetic algorithms depict an optimization problem as the area where the viable solutions are individuals living in that environment. The degree to which an individual adapts to its environment is equivalent to evaluated fitness function on a problem at hand. Further to it a set of feasible solutions take place of a population of organisms. An individual is treated as set of symbols drawn from a finite set or a string of binary digits each of which is encoded. And encoded individual in the population may be viewed as representation of a prospective solution to the problem.

II. OPTIMIZATION

In field of mathematics and computer science, optimization refers to choosing the best thing from some set of available viable options. The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics [16] [9]. More generally, it means finding best available values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

Optimization Problem Representation

Given: a function f: B→F from some set B of some elements to the real numbers [29].

Sought: an element i0 in B such that f(i0) <= f(i) for all i in B ("minimization") or such that f(i0) >= f(i) for all i in B ("maximization").

This type of formulated problem is called an optimization problem or a mathematical programming problem. The basic approach or idea behind optimization is the efficient and effective allocation of available pool of resources [29]. Optimization can be applied to any scientific or engineering discipline. The aim of optimization is to find an algorithmic solution, which can be adopted for a given set of problems. There exist no generalized methods which could be adopted to solves all the optimization problems.

Optimization Search Techniques

Different optimization search techniques available are as follows:

- Stochastic Hill Climbing
  Stochastic Hill Climbing is the simplest approach where every iteration consists of choosing a solution randomly in neighborhood of current solution and retains this new solution only if it improves the chosen fitness function. It converges towards the optimal solution if the fitness function of problem is continuous and has only one peak (i.e. a unimodal function) [29]. For multimodal functions with many peaks the algorithm is likely to stop on the first peak it finds even if it is not the highest one.

- Random Search
  This is quite an unintelligent strategy, and is rarely used by itself. It doesn't take much effort to implement it, and an important number of evaluations can be done quite quickly as it only explores the search space by randomly selecting solutions and evaluates their fitness [33].

- Simulated Annealing
  The iteration of the simulated annealing consists of randomly choosing a new solution in neighborhood of the actual solution. It behaves much like a hill climbing

method but with the possibility of going downhill to avoid being trapped at local optima [29].

Apart from these there are other optimization techniques which are said as computational optimization methods.

Computational Optimization Techniques

Method of computational optimization techniques are divided based on number of variables used as SVO (Single-variable optimization) and MVO (Multi-variable optimization). There are two types of problems which are associated with it as "Deterministic Problem" and "Non-Deterministic Problem" [34]. "Deterministic Problems" have search space area that is very small and one can easily find the feasible solution, so that these problems can be solved by simple simplex method differentiations. Whereas "Non-Deterministic Problems" have search space area that is very large so these are very difficult to analyze. These problems can be solved by using other different optimization techniques and some of the techniques are:

- Ant Colony Optimization
- Beam Search
- Differential Evolution
- Evolution Strategy
- Harmony Search
- Genetic Algorithm
- Stochastic Tunneling and
- Tabu Search

III. BASIC GENETIC ALGORITHM IMPLEMENTATION

Here a basic genetic algorithm is being depicted and discussed using a flow diagram as shown belowin Fig 1:

Fig 1: Basic Genetic Algorithm

Steps for applying basic genetic algorithm are:

Step 1: Choosing an Encoding scheme
Step 2: Choosing Fitness function
Step 3: Choosing Operators
Step 4: Choosing Parameters
Step 5: Choosing an Initialization method and Stopping criteria.

Step 1: Selecting Encoding Scheme - The first step for solving a problem in GA is to encode the problem at work into a state which can be solved using GA. The application of a genetic algorithm to a problem starts with the encoding. The encoding specifies a mapping that transforms a possible solution to the problem into a structure containing a collection of decision variables that are important to the problem worked upon.
Various types of encoding schemes are:

- Binary encoding
- Permutation encoding
- Value/Real encoding
- Tree encoding
- Octal encoding &
- Hexadecimal encoding

For our problem at hand value encoding is most appropriate.
Real/Value encoding:

Every chromosome is a string of values and the values can be anything related to the problem. This encoding produces best results for some special problems[35]. On the other hand, it is often necessary to develop new genetic operator specific to the problem. In value encoding, every chromosome is a string of some values.

Various examples of encoded chromosomes relating to our problem are:

[1.8643 1.4463 0.3677 -4.2028 4.4066]
[2.7808 -4.5313 0.7913 3.0527 -3.4004]
[-1.1779 4.8282 1.7078 4.8730 -2.3185]

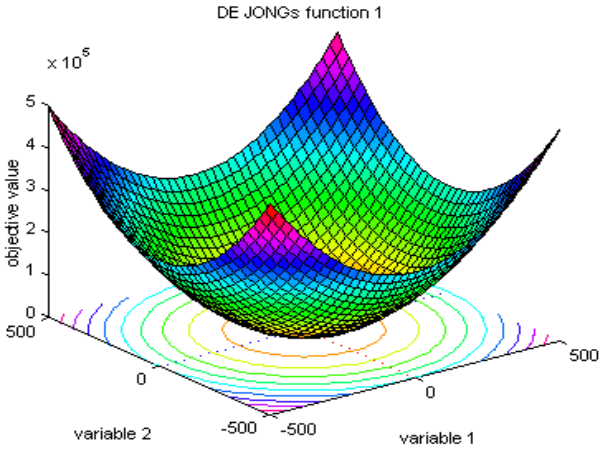
These are chosen considering the length of chromosome as 5. The next step after encoding is to select the fitness function.

Step2: Selecting Fitness function - The next step is to specify a function that can assign a score to any possible solution. The task of the GA is to discover solutions that have higher fitness values among the set of all possible solutions [36].
Here the fitness function chosen is from literature of benchmark functions commonly used to test optimization procedures dedicated for multidimensional, continuous optimization task [35]. The one which is used in this work is De Jong's function 1, also called as sphere model.
It's a simplest test function i.e. De Jong's function 1 shown here with multiple variables here in Fig 2. It is continuous, convex and unimodal. It has the following general definition:

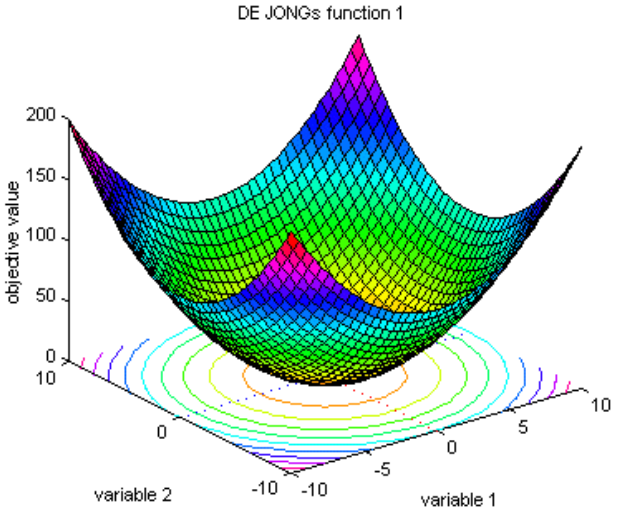
$F_1(x) = \sum_{i=1}^{n} x_i^2$, where $-5.12 <= x_i <= 5.12$

F1(x)=sum(x(i)^2), where, (i =1: n) and -5.12<=x(i)<=5.12
Global minimum is at: f (x)=0, x(i)=0, where, i =1:n

 ISSN: 2321-8169
563 – 569



(a) Surface plot of function in very large area from -500 to 500 for each of the variables [10].



(b) The function at a smaller area from -10 to 10, [10]

Fig 2: Visualization of De Jong's function 1 using different domains of the variables with changed scaling [10].

Step 3: Selecting Operators - Once the encoding and the fitness function are specified, next step is to choose selection and genetic crossover operators to evolve new solutions to the problem being solved[29]. The selection operator simulates the "survival-of-the-fittest". There are various mechanisms to implement this operator, and the idea is to give preference to better individuals. Selection replicates individuals with high fitness values and removes individuals with low fitness values[33].

There are many different Selection operators which one could work with for finding the solution to a problem like: Random Selection, Roulette wheel selection, Rank selection and Elitism etc.

For our case, the selection mechanisms used are:

- Random selection
- Roulette wheel selection and
- Elitism/Best Fit

Choosing Crossover Operators: Next comes is the crossover operation after the selection of parents for mating.

Crossover is the process of taking two parent chromosomes and producing from them number of offspring's [34]. After the selection (reproduction) process, the population is filled with better individuals. Crossover is a recombination operator that works in three steps:

- The reproduction operator selects at random a pair of two individual strings for the mating/crossover.
- A cross site or cross point is selected at random along the string length.
- Finally, the position values are swapped between the two strings following the cross site.

For our problem encoded with real/value encoding, arithmetic crossover is feasible.

Choosing Mutation Operator: After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum [36]. With an encoding, a fitness function, and operators in hand, the GA is ready to enter in action. But before doing that, the user should specify number of parameters such as population size, selection rate, and operator probabilities [36].

Step 4: Choosing Initialization Method and Stopping Criteria – For a function or algorithm to work properly there must an initialization method with specific values for required parameters and a stopping criteria should also be there to stop iterations [33].

For our Problem at hand:

- ✓ Population size: 100 approximately
- ✓ Selection Rate: Depends upon Various Selection Schemes employed
- ✓ Crossover probability: 100%. (to maintain the population diversity) &
- ✓ Mutation: after every 5 consecutive algorithm runs

Other Properties:

- ✓ Stopping Criteria: No. of Iterations
- ✓ Initialization method: Random Generation of Population

Other considerations: One is - Operator probabilities are chosen to maintain the population diversity and other is with every iteration/algorithm run crossover is performed always.

IV. PROPOSED ALGORITHMS

Our problem at hand is comparing 3 specified selection techniques used in GA. For our implementations Selection is considered as an operator which is being varied while applying GA, maximizing De Jong's function1 while keeping other operators of a basic Genetic algorithm constant.

START GA
Step 1. gen= 0, iter_num=gen [initialize generation variable and copy its value to iter_num variable].
Step 2. Initialize population randomly [for generation first only i.e. gen==0].
Step 3. Evaluate population [for initialized population] done= false.
Step 4. WHILE not Done DO
gen=gen+1
Step 5. Select P(gen) from P(gen -1) [Perform proper selection operation].
Step 6. Crossover P(gen) [Perform arithmetic crossover to population resulting in offspring's].
Step 7. If iter_num==5, Mutate P(gen) [perform mutation when iter_num=5] and set iter_num=0;
Step 8. Evaluate Fitness P(gen) [evaluate fitness for new population]
Step 9. Is done=Optimization criteria met?
Step 10. If (not) then,
iter_num=iter_num+1
[increase iteration number].
END WHILE
[If optimization criteria met, stop the algorithm].
Step 11. Output the best solution
END GA

Fig 3. Algorithm for proposed GA [29] [33]

Here in this algorithm shown in Fig 3, "gen" represents generation number and iter_num is used for counting the number of iterations. Then the population is initialized randomly for generation first.
Then the fitness of the population is evaluated using De Jong's function 1. Then after this the selection operation is performed [36] (choosing the selection mechanism out of the three techniques under consideration).
After selecting the parents for reproduction/mating, arithmetic crossover is performed resulting in generation of new offspring's.
Then if iter_num is equal to 5, then mutation is performed. Otherwise fitness is evaluated for new offspring's and next population is generated after replacement based on the fitness of offspring's[33].
Apart from this, other algorithms which are being used or called for the optimization process are Arithmetic crossover, Random selection/Roulette Wheel Selection/Elitism. Along with this mutation used is of special type, i.e. uniform mutation (for real encoded strings).

These all algorithms are explained as follows:

- ✓ Random Selection: This technique just randomly selects a pair of chromosomes from a given population.
- ✓ Roulette Wheel Selection: The roulette wheel algorithm depends upon the fitness's of members of the population. The basic procedure for the selection is as follows:

The roulette wheel algorithm depends upon the fitness's of members of the population. The basic procedure for the selection is shown in Fig 4 below [33]:

```
for all members of population
        sum += fitness of this individual
end for
for all members of population
 probability = sum of probabilities + (fitness/sum)
        sum of probabilities+= probability
end for
loop until new population is full
    do this twice
        number = Random between 0 and 1
     for all members of population
         if number > probability but less than
        next probability
           then you have been selected
      end for
    end
    create offspring
end loop
```

Fig 4. Algorithm for Roulette Wheel Selection

- ✓ Elitism: This algorithm involves calculating the fitness of all the chromosomes in population and then the individual or individuals with the high fitness value are copied directly to the new population and rest are selected in the traditional way.

- Calculate the fitness of all chromosomes.
- Choose a best or some best from them based on their fitness.
- Move the best chosen to new population.
- Then select the rest depending upon the general selection rule.

After selection, next operation to perform is crossover, and crossover operator used is Simple Arithmetic crossover as shown in Fig 5 [29].
This is performed when encoding scheme used is real/value encoding. As with our problem the chromosomes are encoded with real values.

Step 1. Let the two parents selected be:
$<x_1, x_2, \ldots, x_n>$ and
$<y_1, y_2, \ldots, y_n>$.
Step 2. Select a crossover point (k) at random.
Step 3. Select value for alpha (alp) at random.
Step 4.[perform arithmetic crossover as]
Child 1:
$<x_1, x_2, \ldots, x_k, alp*y_{k+1} + (1-alp)*x_{k+1}, \ldots, alp*y_n + (1-alp)*x_n>$
Child 2:
$<y_1, y_2, \ldots, y_k, alp*x_{k+1} + (1-alp)*y_{k+1}, \ldots, alp*x_n + (1-alp)*y_n>$

Fig 5: Algorithm for Arithmetic crossover

After crossover, next operation is to perform mutation. For our problem, mutation used is uniform mutation used for value encoded strings. Its Algorithm is shown in Fig 6 [29] below:

Step 1. Select a chromosome at random for mutation form a set of offspring's. (say) $<x_1,x_2, \dots ,x_n>$

Step 2. Select a mutation point (m) at random.

Step 3. Randomly generate a real valued number (r_{num}) for performing mutation.

Step 4. [perform mutation operation as]

$<x_1,x_2 ,\dots., x_m + r_{num} , x_{m+1} ,\dots, x_n>$

Fig 6: Algorithm for Uniform Mutation

After performing mutation operation, new population generated is evaluated and based on those Genetic algorithm further proceeds and completes based on the selected stopping criteria.

V. RESULTS ANALYSIS

After implementing the above stated algorithm with De Jong's function1 as fitness function and using three selected selection techniques, the results generated for our problem are illustrated below.

The values of different selection techniques in different number of iterations can be shown with the help of a table, having comparison of fitness values of function under consideration. The following table below will show the comparison of various selection techniques employed.

TABLE I. RESULTS OF VARIOUS SELECTION TECHNIQUES FOR APPROPRIATE NO. OF ITERATIONS

No. of Iterations	Fitness Values		
	Roulette Wheel Selection	Random Selection	Best Fit/Elitist Selection
10	66.5948	62.1955	60.6914
20	73.6391	67.2733	67.3982
30	75.6124	64.5586	65.8184
40	80.7086	70.2496	76.7383
50	86.7839	82.1488	84.9791
60	88.7983	85.1792	85.1064
70	93.2734	68.4312	87.3976
80	102.4238	83.7016	97.2417
90	112.4693	88.7104	90.9710
100	128.1761	90.1436	106.9876

Here in Table 1 above it's been shown that, how in different number of iterations the fitness values of different selection techniques changes and one of them finally reaches a maximum value, optimizing the function under processing [34] [33] [29]. Along with this one line graph is also shown below to display a comparison of these three techniques based on the fitness values.

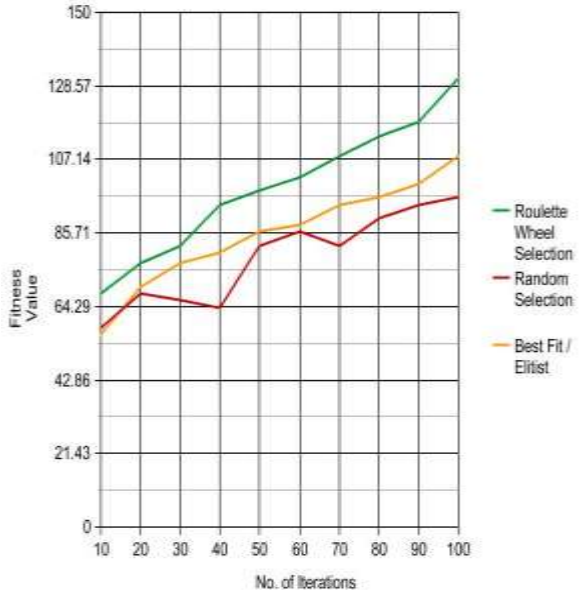


Fig 7: Line graph showing comparison of different selection techniques

The results can also be depicted using a bar graph showing the fitness values of various selection techniques, when applied for de Jong's function1 as shown in Fig 7 here. The results that came out are in favor of the roulette wheel selection technique. With this technique for implementing de Jong's function1 results in convergence towards the maximum attainable value within the specified number of iterations[20]. But with other functions the function converges to a less optimal value.

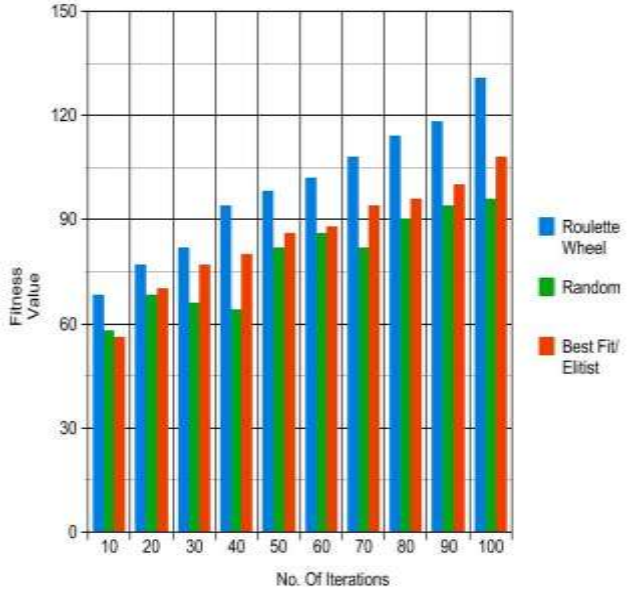


Fig 8: Results of various selection techniques for appropriate no. of iterations

The population initially chosen is as follows: (say, 4 chromosomes)

[1.8643	1.4463	0.3677	-4.2028	4.4066]
[2.7808	-4.5313	0.7913	3.0527	-3.4004]
[1.1779	4.8282	1.7078	4.8730	-2.3185]
[-1.8318	-2.9409	-0.0082	4.9956	-5.0898]

And after applying the algorithms presented here, to this population for a certain number of iterations, the results came out good as shown in Fig 8 here and roulette wheel selection comes out to be best of the other two techniques.

VI. CONCLUSION

Function optimization/maximization is the main point of discussion in this paper. This paper is based on implementing De Jong's function1 i.e. sphere model using different selection techniques used in Genetic algorithm and making a comparison of them based on the fitness values of function at different number of iterations. The main point around which all this work revolves is the different selection techniques employed for running this algorithm. All other parameters are kept constant, except the three selection techniques. Termination criteria chosen is the number of iterations and the function reaching its maximum value [29]. After all the experimentation and implementation, results that came out are like, out of all these three selection techniques, best one is the roulette wheel selection. Roulette wheel is a probability based selection technique. In future, further other different selection techniques can be employed for maximizing this function. While other prospect in future can be, changing other parts of the genetic algorithm keeping a particular selection technique fixed which might finally show even better results as compared to that came now. Further a larger size of chromosomes could be used for better results.

REFERENCES

[1] Back, T. (1996). "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms" Oxford University Press US. ISBN: 0-1950-9971-0, 978-0-19509-971-3 [Partly available online at:

http://books.google.de/books?id=EaN7kvl5coYC]

[2] Back, T., Fogel, David B. & Michalewicz, Z. (Eds.) (1997). "Handbook of Evolutionary Computation. Computational Intelligence" Library Oxford University Press in cooperation with the Institute of Physics Publishing / CRC Press, Bristol, New York, ring bound edition. ISBN: 0-7503-0392-1, 978-0-75030-392-7, 0-7503-0895-8, 978-0-75030-895-3. [Partly available online at:

http://books.google.de/books?id=n5nuiIZvmpAC]

[3] Bryant, Kylie (2000). "Genetic Algorithms and the Traveling Salesman Problem" in Proceedings of 1st GNT Regional Conference on Mathematics, Statistics and Applications

[4] Coello Coello, C. A. (2016, July). Constraint-handling techniques used with evolutionary algorithms. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (pp. 563-587). ACM.

[5] Coello, Carlos (2000). "An updated survey of GA-based multiobjective optimization techniques", ACM Computing Surveys, vol. 32, no. 2, pp. 109-143

[6] Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology, 10, 340-347.

[7] Davis, L. & Coombs, S. (1987). "Genetic algorithms and communication link speed design: theoretical considerations", In J.J. Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms, pp. 252-256

[8] Dehuri, S. et al. (2008). "Application of elitist multi-objective genetic algorithm for classification rule generation", Applied Soft Computing, pp. 477–487

[9] Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems. Expert Systems with Applications, 42(1), 193-202.

[10] http://www.geatbx.com/docu/fcnindex-01.html

[11] Goldberg, D. E. (2002). "The Design of Innovation: Lessons from and for Competent Genetic Algorithms". Norwell, MA: Kluwer

[12] Inazawa, H. & Kitakaze, K. (2006). "Locus-Shift Operator for Function Optimization in Genetic Algorithms", Complex Systems Publications, Inc

[13] Lal, N., Singh, A. P., Kumar, S., Mittal, S., & Singh, M. (2014). International Journal of Advanced Research in Computer Science and Software Engineering. International Journal, 4(12).

[14] Lau, T. L. & Tsang, E. P. K. (1998). Guided genetic algorithm and its application to the generalized assignment problem, Submitted to Computers and Operations Research

[15] Lima, C., Sastry, K., Goldberg, D. E., Lobo, F., "Combining competent crossover and mutation operators: A probabilistic model building approach". Proceedings of the 2005 Genetic and Evolutionary Computation Conference . 2005, 735—742

[16] Lin, C. D., Anderson‐Cook, C. M., Hamada, M. S., Moore, L. M., & Sitter, R. R. (2015). Using genetic algorithms to design experiments: a review. Quality and Reliability Engineering International, 31(2), 155-167.

[17] Lobo, Fernando Miguel (2000).The parameter-less genetic algorithm: rational and automated parameter selection for simplified genetic algorithm operation. Paper submitted in International Conference on Genetic Algorithms, in Lisboa

[18] Madureira, A., Ramos, C., & Silva, S. Carmo, (2002). "A Coordination Mechanism for Real World Scheduling Problems using Genetic Algorithms. Evolutionary Computation", in Proceedings of the 2002 CEC, vol. 1, pp. 175 –180

[19] Maleki, I., Ghaffari, A., & Masdari, M. (2014). A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization. International Journal of Innovation and Applied Studies, 5(1), 72.

[20] Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4), 1053-1073.

[21] Molga, M. & Smutnicki, C. (2005). "Test functions for optimization needs", in Proceedings of 4th Conference on Genetic Algorithms

[22] Onwubolu, G. C., & Babu, B. V. (2013). New optimization techniques in engineering (Vol. 141). Springer.

[23] Omar, M., Baharum, A., & Hasan, Y. Abu (2006). "A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm" ,in Proceedings of 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications, University Sains Malaysia, Penag

[24] Pakhira, M. K. & Rajat, K. De (2007). "Generational Pipelined Genetic Algorithm (PLGA) using Stochastic Selection", International journal of computer systems science and engineering vol. 1, no. 1, ISSN 1307-430X

[25] Pappa, G. L., Ochoa, G., Hyde, M. R., Freitas, A. A., Woodward, J., & Swan, J. (2014). Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. Genetic Programming and Evolvable Machines, 15(1), 3-35.

[26] Reed, M., Yiannakou, A., & Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. Applied Soft Computing, 15, 169-176

[27] Sandstrom, K. & Norstrom, C. (2002). Managing complex temporal requirements in real-time control systems, Engineering of Computer-Based Systems, 2002. Proceedings Ninth Annual IEEE International Conference and Workshop, pp. 103 –109

[28] Sharma, P., & Wadhwa, A. (2014). Analysis of Selection Schemes for Solving an Optimization Problem in Genetic Algorithm. International Journal of Computer Applications, 93(11)

[29] Sivanandam, S. N. & Deepa, S. N. (2008). "Introduction to Genetic Algorithms", Springer

[30] Thakur, M. (2014). A new genetic algorithm for global optimization of multimodal continuous functions. Journal of Computational Science, 5(2), 298-311.

[31] Tomassini, M. (1999). "Parallel and Distributed Evolutionary Algorithms: A Review". In Miettinen, K., Makela, M., & Periaux, J. (Eds.), Evolutionary Algorithms in Engineering and Computer Science (pp. 113 - 133). Chichester: J. Wiley and Sons

[32] von Lücken, C., Barán, B., & Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. Computational Optimization and Applications, 58(3), 707-756.

[33] Wadhwa, Amit. Analysis Of Selection Techniques In Genetic Algorithm. 1st ed. Germany: LAP Lambert Academic Publishing, 2016. Print

[34] Wadhwa, A., 2016 Comprehensive Analysis of Security Issues and Solutions While Migrating to Cloud Environment. International Journal of New Innovations in Engineering and Technology, 4(4)

[35] Wadhwa, A., & Garg, A. (2015). Studying and Analyzing Virtualization While Transition from Classical to Virtualized Data Center. International Journal of Computer Applications, 117(14), 10-14.

[36] Wadhwa, A., & Gupta, V. K. (2014). Framework for User Authenticity and Access Control Security over a Cloud. International Journal on Computer Science and Engineering, 6(4), 138

[37] Yalcinoz, T. & Altun, H. (2004). "A new genetic algorithm with arithmetic crossover to economic and environmental economic research dispatch". Submitted as a project work at Dept. of Electrical & Electronic Engg.. Nigde University, Turkey.