ISSN No. 0976-5697

Volume 4, No. 3, March 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Genetic Algorithm for optimization using MATLAB

Mr. Manish Saraswat
Research Scholar,
Banasthali University, Banasthali (Rajasthan), India
manishsaraswat24@gmail.com

Mr. Ajay Kumar Sharma
Research Scholar,
Mewar University, Chittorgarh (Rajasthan), India
profsharmaak@gmail.com

Abstract: As the applications of systems are increasing in various aspects of our daily life, it enhances the complexity of systems in Software design (Program response according to environment) and hardware components (caches, branch predicting pipelines).Within the past couple of years the Test Engineers have developed a new testing procedure for testing the correctness of systems: namely the evolutionary test. The test is interpreted as a problem of optimization, and employs evolutionary computation to find the test data with extreme execution times. Evolutionary testing denotes the use of evolutionary algorithms, e.g., Genetic Algorithms (GAs), to support various test automation tasks. Since evolutionary algorithms are heuristics, their performance and output efficiency can vary across multiple runs, there is strong need a environment that can be handle these complexities, Now a day's MATLAB is widely used for this purpose. This paper explore potential power of Genetic Algorithm for optimization by using new MATLAB based implementation of Rastrigin's function, throughout the paper we use this function as optimization problem to explain some key definitions of genetic transformation like selection crossover and mutation.

Keywords: Rastrigin's function, Evolutionary Testing, Genetic Algorithm (GA) , MatLab & Fitness.

I. INTRODUCTION

Genetic algorithms are an approach to optimization and learning based loosely on principles of biological evolution, these are simple to construct, and its implementation does not require a large amount of storage, making them a sufficient choice for an optimization problems. Optimal scheduling is a nonlinear problem that cannot be solved easily yet, a GA could serve to find a decent solution in a limited amount of time Genetic algorithms are inspired by the Darwin's theory about the evolution **"survival of fittest"**, it search the solution space of a function through the use of simulated evolution (survival of the fittest) strategy. Generally the fittest individuals of any population have greater chance to reproduce and survive, to the next generation thus it contribute to improving successive generations However inferior individuals can by chance survive and also reproduce, Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through the application of mutation, crossover and selection operations to individuals in the population.The development of new software technology and the new software environments (e.g. MATLAB) provide the platform to solving difficult problems in real time. It integrates numerical analysis, matrix computation and graphics in an easy to use environment.

MATLAB functions are simple text files of interpreted instructions Therefore; these functions can be re-implemented from one hardware architecture to another without even a recompilation step. MATLAB (**Mat**rix **Lab**oratory), a product of Mathworks, it is a scientific software package developed to provide an integrated environment for numeric computation and graphics visualization in high-level programming language. Originally it was written by Dr Cleve Moler, Chief scientist at MathWorks, Inc., to provide easy access to matrix software developed in the LINPACK and EISPACK projects [2]. MATLAB has a wide collection of functions useful to the genetic algorithm practitioner and those wishing to experiment with the genetic algorithm for the first time.

In MATLAB's high-level language, problems can be coded in **m-files** in a fraction of the time that it would take to create C or FORTRAN programs for the same purpose. It also provide advanced data analysis, visualization tools and special purpose application domain toolboxes. This paper is organized into three parts: Part I describes the usefulness of GA and features of new software MATLAB. Part II discusses the implementation issues of GA in various available languages, tools and software. Finally GAs is implemented using MATLAB for the **Rastrigin's function** as case study for optimization. The Part III concludes the objectives of paper.

II. OVERVIEW OF PROGRAMMING LANGUAGES USED TO IMPLEMENTATION OF GA

The implementation of genetic algorithm on high-performance computers is a difficult and time-consuming task. The implementing languages must be closely as possible to the mathematical description of the problem, simple and easy-to-use procedural language. The C/C++, FORTRAN are lower-level compiled programming languages (sometimes classified as a 3rd generation language) that is widely used in academia, industry, commerce and GA is also implemented by using these category of languages. The main advantage of compiled low-level languages is their execution speed and efficiency (for example embedded systems)but now a days the MATLAB is often employed in research and industry and it is an example of a high-level "scripting" or "4th generation"

"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

language. The most prominent difference between compiled languages and interpreted languages is that the interpreter program reads the source code and translates it into machine instructions on the fly, *i.e.* no compilation is required. This decreases the execution speed but it make the programmer free from memory management, allows dynamic typing and interactive sessions.

It is important to note that the programs written in scripting languages are usually significantly shorter [3] than equivalent programs written in compiled languages and also take significantly less time to code and debug. In short, there is a trade-off between the execution time (small for compiled languages) and the development time (small for interpreted languages). Another important feature of MATLAB (and other interpreted languages like Pythan) is the ability to have interactive sessions. The user can type one or several commands at the command prompt and after pressing return, these commands are executed immediately. By this it allows the programmer for interactive testing of small parts of the code (without any delay stemming from compilation) and encourages experimentation [9].

The MATLAB package comes with sophisticated libraries for matrix operations, general numeric methods and plotting of data, therefore MATLAB become first choice of programmer to implement scientific, graphical and mathematical applications and for the GA implementation MATLAB is come with special tool that is ***GA-tool or Optimtool***

III. THINGS TO CONSIDER FOR GENETIC ALGORITHMS IMPLEMENTATION IN MATLAB

The first thing must do in order to use a GA is to decide **if it is possible to automatically build solutions on problem**. For example, in the Traveling Salesman Problem, every route that passes through the cities in question is potentially a solution, although probably not the optimal one. It is must to do that because a GA requires an initial **population P** of solutions.

Then must decide **what "gene" representation will use** we have a few alternatives like *binary*, *integer*, *double*, *permutation*, etc. The *binary* and *double* being the most commonly used since they are the most flexible. After selecting the gene representation it must be decide:

The **method to select parents** from the population P (Cost Roulette Wheel, Stochastic Universal Sampling, Rank Roulette Wheel, Tournament Selection, etc.), the **way these parents will "mate"** to create descendants, the **mutation method** (optional but useful), the **method will use to populate the next generation** and the algorithm's **termination condition** (number of generations, time limit, acceptable quality threshold). Now second thing is **Processor** and **operating system** that must be capable of running the program the algorithm is coded in MATLAB. Matlab provides an optimization toolbox that includes a GA-based solver. The toolbox can be start by typing ***optimtool*** in the Matlab's command line and pressing enter. As soon as the optimization window appears, we can select the solver *ga – Genetic Algorithm* and now matlab are ready to go. The user should program (by writing m files) any extended functionality required.

We will implement *Rastrigin's Functions in the proper field and number of variable is 2* and population type is double vector (figure 1). The equation of this function and Matlab (m-file) code is given as below

$$Ras(x) = 20+x_1^2+x_2^2-10(\cos 2\pi x_1+\cos 2\pi x_2)$$

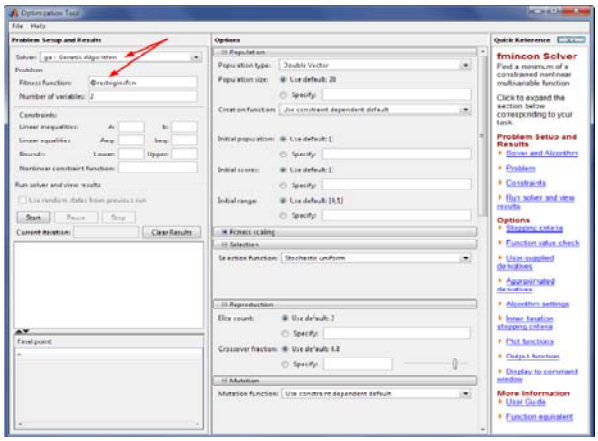

Figure:1 GAs in Matlab's Optimization Toolbox

MATLAB Code:

```
function y = rast(x)
% the default value of n = 2.
n = 2;
s = 0;
for j = 1:n
s = s+(x(j)^2-10*cos(2*pi*x(j)));
end
y = 10*n+s;
```

Now it is ready (the default settings in every-thing else is adequate). Press the **Start** button. The algorithm starts, the plots are pop-up and soon the results are displayed as in figure 2.

The best fitness function value (the smallest one since we minimize) and the termination condition met are printed, together with the solution (Final Point – it is very close to (0, 0)). Since the method is stochastic, don't expect to be able to reproduce any result found in a different run. Now check the two plots on the left. It is obvious that the population converges, since the average distance between individuals (solutions) in term of the fitness value is reduced, as the generations pass. This is a measure of the diversity of a population. It is hard to avoid convergence but keeping it low or postponing its appearance is better. Having diversity in the population allows the GA to search better in the solution space.

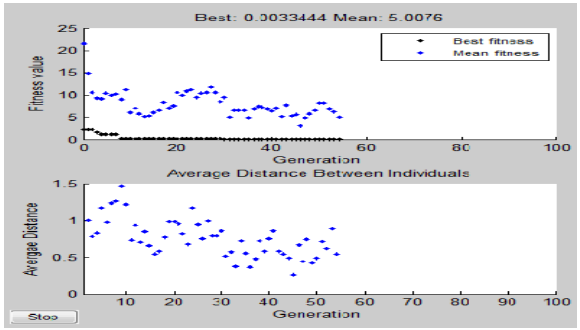

Figure 2: Rastrigin's function optimization with default setting

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

It is seen from figure-2 the fitness value as it gradually gets smaller. It is an indication that optimization takes place since not only the fitness value of the best individual was reduced, even the mean (average) fitness of the population was also reduced (that is, in terms of the fitness value, the whole population was improved we have better solutions in the population, at the end).

a. ***Population Diversity*– size – range, fitness scaling** The performance of a GA is affected by the diversity of the initial population. If the average distance between individuals is **large**, it is indication of **high** diversity; if the average distance is **small** its represent **low** diversity in the population**.**

If the diversity is too high or too low, the genetic algorithm might not perform well. We will explain this by the following: By default, the Optimization Tool creates a random initial population using a creation function. We can limit this by setting the *Initial range* field in *Population* options. Set it to (1; 1.1). By this we actually make it harder for the GA to search equally well in all the solutions space. Leave the rest settings as previous (figure: 1) except *Options-Stopping Criteria-Stall Generations* which should be set to 100. This will allow the algorithm run for 100 generation providing us with better results (and plots). Now click the Start button.The GA returns the best fitness function value of approximately 2 and displays the plots in as in figure 3.

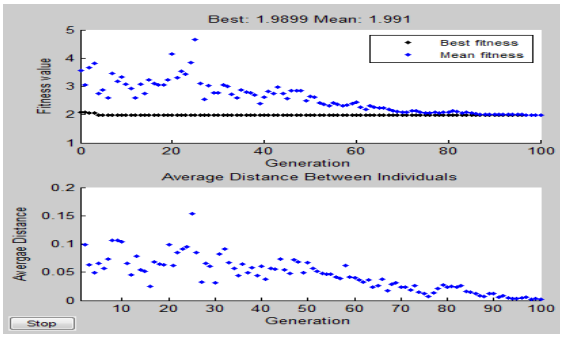

Figure 3: Rastrigin's function optimization with default setting, except Stopping Criteria-Stall Generations set 100 and initial range set [1; 1.1]

The upper plot, which displays the best fitness at each generation, show little progress in lowering the fitness value (black dots). The lower plot shows the average distance between individuals at each generation, which is a good measure of the diversity of a population. For this setting of initial range, there is too little diversity for the algorithm to make progress. The algorithm was trapped in a local minimum due to the initial range restriction.

Next, set Initial range to [1; 100] and run the algorithm again. The GA returns the best fitness value of approximately 3.3 and displays the following plots as in **figure: 4**. this time, the genetic algorithm makes progress, but because the average distance between individuals is so large, the best individuals are far from the optimal solution. Note though that if we let the GA to run for more generations (by setting *Generations* and *Stall*

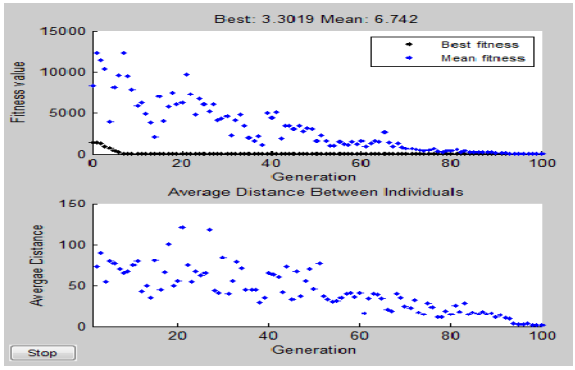

Figure 4: Rastrigin's function optimization with default setting, except Stopping Criteria-Stall Generations set 100 and initial range set [1; 100]

Generations in *Stopping Criteria* to 200) it will eventually find a better solution.

Set *Initial range* to [1; 2] and run the GA. This returns the best fitness value of approximately 0.012 and displays the plots that follow as in figure 5.

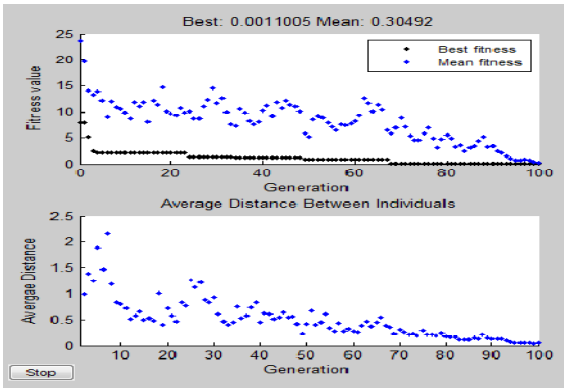

Fig 5: Rastrigin's function optimization with default setting, except Stopping Criteria-Stall Generations set 100 & initial range set [1; 2]

The diversity in this case is better suited to the problem, so the genetic algorithm returns a much better result than in the previous two cases.

In all the examples above, we had the *Population Size* (*Options-Population*) set to 20 (the default). This value determines the size of the population at each generation. Increasing the population size enables the genetic algorithm to **search more points and thereby obtain a better result.** However, the larger the population size, the longer the genetic algorithm takes to compute each generation.

It is important to note that Population *Size* to be at least the value of *Number of variables*, so that the individuals in each population span the space being searched.

Finally, another parameter that affects the diversity of the population is the ***Fitness Scaling***. If the fitness values vary too widely Figure: 6, the individuals with the lowest values (recall that we minimize) reproduce too rapidly, taking over the population pool too quickly and preventing the GA from searching other areas of the solution space. On the other hand, if the values vary only a little, all individuals have approximately the same chance of reproduction and the search will progress very slowly.

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

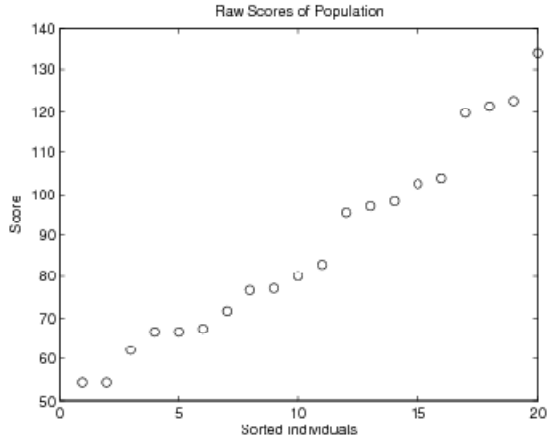

Figure 6: Raw fitness value lower

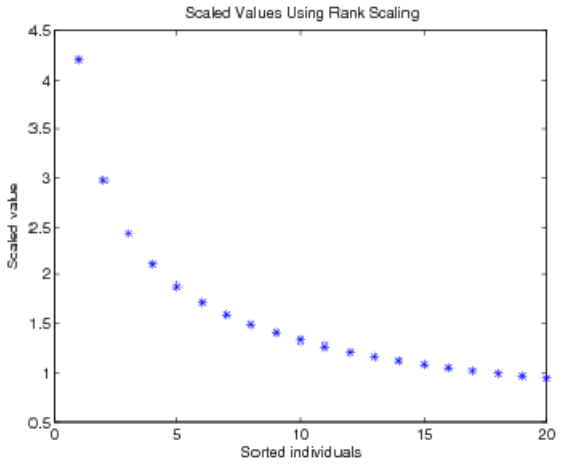

Figure 7: Raw fitness value Higher

It is clear from the Figure: 6 Raw fitness values (lower is better) vary too widely on the. Scaled values (figure: 7) do not alter the selection advantage of the good individuals (except that now bigger is better). They just reduce the diversity we have on the above. This prevents the GA from converging too early.

The *Fitness Scaling* adjusts the fitness values (scaled values) before the selection step of the GA. This is done without changing the ranking order, that is, the best individual based on the raw fitness value remains the best in the scaled rank, as well. Only the values are changed, and thus the probability of an individual to get selected for mating by the selection procedure. This prevents the GA from converging too fast which allows the algorithm to better search the solution space. We continue Rastrigin's function implantation in MATLAB, Use the following settings leaving every-thing else in its default value (*Fitness function*: @rastriginsfcn, *Number of Variables*: 2, *Initial Range*: [1; 20], *Plots*: Best Fitness, Distance).

The ***Selection*** panel in *Options* controls the ***Selection Function***, that is, how individuals are selected to become parents. Note that this mechanism works on the scaled values, as described previously.

Most well-known methods are presented (uniform, roulette and tournament). An individual can be selected more than once as a parent, in which case it contributes its genes to more than one child.

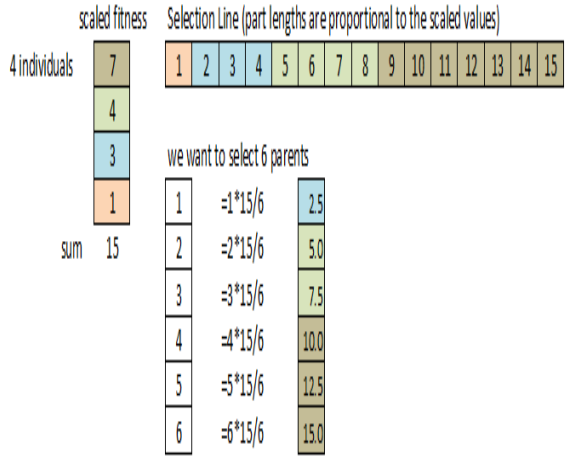

Figure 8: Stochastic uniform selection method. For 6 parents we step the selection line with steps equal to 15/6.

The default selection option, ***Stochastic Uniform***, lays out a line (Figure 8) in which each parent corresponds to a section of the line of length proportional to its scaled value. For example, assume a population of 4 individuals with scaled values 7, 4, 3 and 1. The individual with the scaled value of 7 is the best and should contribute its genes more than the rest. We create a line of length 1+3+4+7=15. Now, let's say that we need to select 6 individuals for parents. We step over this line in steps of 15/6 and select the individual for crossover.

The ***Reproduction*** panel in *Options* control how the GA creates the next generation. Here you specify the amount of **elitism** and the fraction of the population of the next generation that is generated through mating (the rest is generated by mutation). The options are: **Elite Count**: the number of individuals with the best fitness values in the current generation that are guaranteed to survive to the next generation. These individuals are called elite children. The default value of Elite count is 2. Try to solve the Rastrigin's problem by changing only this parameter. Try values of 10, 3 and 1. we will get results like those depicted in Figure 5. It is obvious that you should keep this value low. 1 or 2 (depending on the population size).

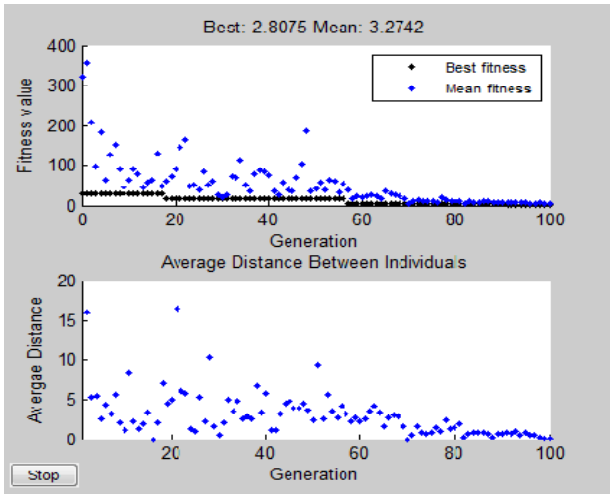

Figure 9: Elite count 10

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

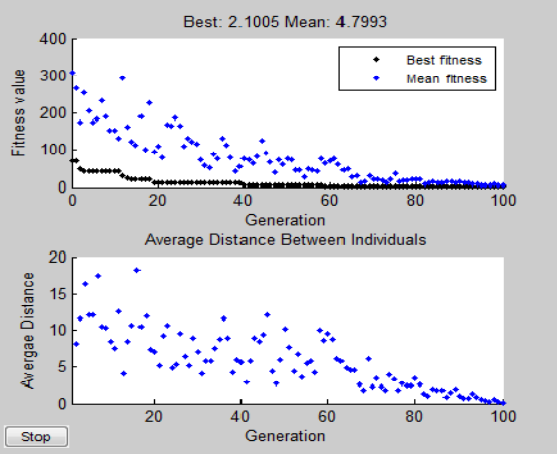

Figure 10: Elite count 3

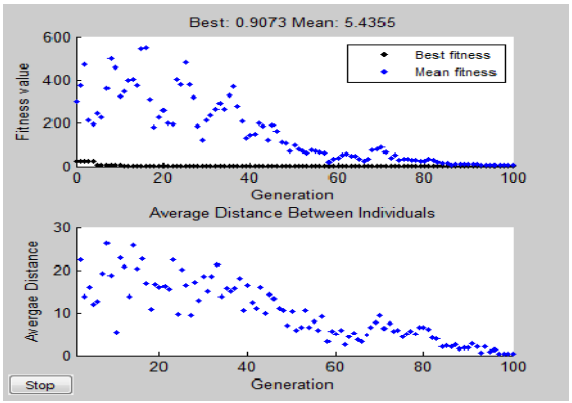

Figure 11: Elite count 1.

From the figure 9, 10, 11 it clear that too much elitism results in early convergence which can make the search less effective.

- ***a.*** ***Crossover Fraction:*** is the fraction of individuals in the next generation, other than elite children, that are created by crossover (remaining is generated by mutation). A crossover fraction of '1' indicates means that all children other than elite individuals are crossover children. A crossover fraction of '0' indicates that all children are mutation children.
- ***b.*** ***Two-point crossover-*** two crossover points are selected, binary string from the beginning of the chromosome to the first crossover point is copied from the first parent, the part from the first to the second crossover point is copied from the other parent and the rest is copied from the first parent again
- ***c.*** ***Mutation-*** It is the Random change one or more digits in the string representing an individual.

IV. CONCLUSION

The main objective in this paper is to illustrate that how the new technology of MATLAB can be used in order to implement a genetic algorithm in optimization problems. It uses the power of genetic algorithms to generate fast and efficient solutions in real time. The experimental results show that GATool can improve fitness value by providing quickly a set of near optimum solutions. Concerning the effect of different GA parameter configurations, it found that an increase in population size can improve performance of the system. The parameter of crossover rate does not affect seriously the quality of the solution. Genetic Algorithms are easy to apply to a wide range of optimization problems, like the traveling salesperson problem, inductive concept learning, scheduling, and layout problems. The result shows that the proposed GAs with the specification can find solutions with better quality in shorter time. The developer uses this information to search, locate, and debug the faults that caused the failures. While each of these areas for future consideration could be further investigated with respect to applicability for software testing, because it is also an optimization problem with the objective that the efforts consumed should be minimized and the number of faults detected should be maximized. Finally, it would be interesting for further research to test a series of different systems in order to see the correlation between genetic algorithm and system performances.

V. REFERENCES

[1]. D.E. Goldberg, Genetic Learning in optimization, search and machine learning. Addisson Wesley, 1994.

[2]. J.J. Grefenstette. Genetic algorithms for changing environments. In R. Manner abd B. Manderick, editor, Parallel Problem Solving from Nature 2, pages 465-501. Elsevier Science Publishers.

[3]. Mathworks, The: Matlab - UserGuide. Natick, Mass.: The Mathworks, Inc., 1994-1999. http://www.mathworks.com

[4]. Henriksson, D., Cervin, A., Arzen, K.E.: TrueTime: Real-time control system simulation with MATLAB/Simulink. In: Proceedings of the Nordic MATLAB Conference, Copenhagen, Denmark (2005)

[5]. A.J. Chipperfield, P. J. Fleming and H. Pohlheim, "A Genetic Algorithm Toolbox for MATLAB," Proc. International Conference on Systems Engineering, Coventq, UK, 6-8 Sept 1998.

[6]. Papadamou, S. and Stephanides, G., A New Matlab-Based Toolbox For Computer Aided Dynamic Technical Trading,

[7]. K. Lakhotia, M. Harman, and P. McMinn. A multi-objective approach to search-based test data generation. In Proc. 9th Annual Conf. on Genetic and Evolutionary Computation (GECCO'07), pages 1098–1105, ACM, 2007.

[8]. Pohlheim, H.: Genetic and Evolutionary Algorithm Toolbox for use with Matlab - Documentation. Technical www.geatbx.com.

[9]. A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering Hans Fangohr University of Southampton, Southampton SO17 1BJ, UK.

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India