International Journal of Computer Applications (0975 – 8887)
Volume 31– No.6, October 2011

An Empirical Study of the Role of Control Parameters of Genetic Algorithms in Function Optimization Problems

V.Kapoor
Institute of Engineering & Technology
Devi Ahilya University,
Indore, India

S.Dey
Indian Institute of Management
Indore, India

A.P.Khurana
School of Computer Science
Devi Ahilya University
Indore, India

ABSTRACT

Genetic algorithms (GAs) are multi-dimensional, blind heuristic search methods that involve complex interactions among parameters (such as population size, number of generations, GA operators and operator probabilities). The question whether the quality of results obtained by GAs depend upon the values given to these parameters, is a matter of research interest. This work studies the problem of how changes in four GA parameters (population size, number of generations, crossover and mutation probabilities) affect GA performance from a practical stand point. To examine the robustness of GA to these parameters, we have tested three groups of parameters and the interactions in each group (a) Crossover and mutation separately (b) Crossover combined with mutation together (c) Population size and number of generations. The results show that for simple problems mutation plays a momentous role, and for complex problems crossover is the key search operator. Based on our study we conclude that, complementary crossover and mutation probabilities combined with a reasonable population size is a reliable approach.

Keywords: Genetic algorithm, control parameters, crossover, mutation, population sizing.

1. INTRODUCTION

Genetic algorithms are a class of flexible optimization procedures inspired by evolution and natural selection, first applied to artificial systems by J H Holland. They are based on Darwinian evolutionary processes and naturally occurring genetic operations on chromosomes proposed by Koza. Though they are highly parallelizable mathematical algorithms, the implementation are often serialized and executed on serial machines. In essence, GAs transform population of individuals (representing solutions to the problem coded into binary strings), each with an associated fitness value, into new population of individuals (i.e next generation) using operators modeled on Darwinian principles of reproduction and survival of fittest.

GAs begin by selecting a random sample of potential solutions to the problem to be solved – represented by the initial population. Initially, a set of starting solutions to the problem have to be coded into binary-string chromosome representation. In the second step, fitness value of every string is calculated according to the objective function defined. In the third step, a selection operator is applied to the initial set of potential solutions, whereby mostly individuals with higher fitness values are selected. In the fourth step, crossover and mutation operators are applied where binary bits of chromosomes are exchanged and mutated to generate a new population (set of solutions). Thus the life-cycle of one generation completes. After these iterative steps from second to fourth are repeated for a fixed number of times or until population converges (known as the number of generations). There are two essential events in the GA process: (i) Creation of new solutions or concepts to solve the problem through crossover (recombination) and mutation. (ii) Elimination of bad solutions by selection operator. The crossover operator serves as an accelerator and is expected to propagate existing ‘good’ building blocks to the next generation i.e its role is of exploitation. Mutation is expected to add random diversity to the population at the expense of disrupting building blocks [1].

GAs are designed to search for global optima but cannot guarantee that the best solution will be found, sometimes the solutions converge to local rather than global optima. This problem can be avoided to a large extent by making use of appropriate control parameters choices. GA parameters (such as population size, GA operators employed, operator probabilities etc) interact in complex ways. Given a finite computational effort to obtain a solution to a problem, it is better to know the GA parameter settings that would lead to a good solution. Since overall computational effort required to run a GA is proportional to the number of function evaluations needed, any advance knowledge of interaction among GA parameters will lead to better global solution in lesser time & making GA more robust. It is seen that lack of robustness in the design choices always lead to local optima and lower levels of performance.

There are a number of studies that explore the interaction among different GA parameters in different application contexts (discussed in Section II). In this paper we investigate how changes in GA parameters affect the GA performance in the context of optimization of functions. We have carried out a large number of tests to evaluate a range of GA parameters and their combinations. Based on these tests we conclude that the choice of GA parameters affect the solutions obtained when GAs are restricted to finite computational resources.

There are three conditions that are relevant to GA design: (i) Encoding (ii) Operators, and (iii) control parameters. Our study is limited to the third of these i.e control parameters. For encoding we use the most commonly used binary coding. For operators we have chosen roulette wheel selection, one point crossover and bit level representation mutation. Since there are a number of other selection schemes such as Bolzman selection, Tournament Selection, Rank Selection, steady state selection etc., we do not have any reason to justify the choice of roulette wheel selection. We have chosen roulette wheel selection due to its simplicity and its wide mention in GA literature. Our interest is only centered on the analysis of control parameters: Crossover and mutation probabilities, number of generations, and size of the population.

To examine the robustness of the GA to control parameters we test three groups of parameters and the interactions inside each group: (a) Crossover and mutation applied separately (b) Crossover combined with mutation (c) Population size and number of generations.

In our study we evaluate the changes in the performance of GA with respect to the changes in the control parameters. This paper is organized as follows: Section 2 presents a literature survey, Section 3 describes the experimental details, Section 4 reports the results obtained, and the analysis thereof, while conclusions and future work appear in Section 5 and Section 6 respectively.

2. LITERATURE SURVEY

Analysis of various selection schemes used in modern GA such as Roulette wheel, rank, tournament and steady state has been done [1] and [10]. Schemes are compared and verified according to convergence time and growth ratio. A parameter less Genetic Algorithm which is one step closer in the direction of making GA more robust [2]. In case of dominant set of decision variable the crossover does not have a significant effect on the performance measures, whereas high mutation rates are more suitable [3]. The problem of finding optimal parameters have been studied by many. Optimization of control parameters of GA is often time consuming. An approach of having meta level GA for control parameter optimization is a good approach [4]. To study dynamics of these interactions more sophisticated stochastic models using Markov chain have also been developed and analyzed [5].

It is found that parameter values adjusted during evaluation gives better results than if set in advance [6]. This has potential of adjusting the algorithm to the problem while solving the problem. Crossover operator is largely dependent on the coding used to represent the decision variables [7]. The success of Genetic algorithm depends on how well the crossover operator respects the underlying coding of the problem [8]. The effect of crossover and mutation can be interchanged by using a suitable coding transformation [9]. It does not help in terms of deciding to which operator we should give importance. Crossover is useful in problems where preservation of building block is necessary. Mutation may destroy already obtained good information [10]-[12]. With this in mind it is suggested that GAs will work well with high crossover and low mutation probability [13]. A crossover hill climbing algorithm is presented illustrates the power of mechanics of crossover [14]. Comparison between normal GA and a GA that uses random crossover has been made. The merits of crossover for genetic research has been questioned [15]. Exploratory power of crossover depends on the differences between its parents. Recent work has extended the theoretical analysis of n-point and uniform crossover with respect to random sampling distributions [16]. An adapting mechanism for controlling the use of crossover in a Evolutionary algorithm is a better approach [17]. An adaptive genetic algorithm that describes the optimal crossover probability as it runs has been proposed [18].

It has been suggested that optimal mutation rate is proportional to the length of chromosomes [19]. For deceptive functions an evolutionary algorithm with a good hill climbing strategy and reasonable mutation rate performs the best. It has been shown that optimal mutation probability is dependent on the representation being used [20]. Adaptive crossover and mutation probabilities help in locating global optimum in a multimodal landscape [21]. It has been shown that mutation can be an independent operator [22]. On implementing on an infinite population model it is found that mutation is a poorer operator than had been recognized [23]. A model of GA has been proposed that applies varying mutations parallel to crossover and background mutation by using extinctive selection to enhance the effectiveness of GA by [24]. It has also been seen that complementary crossover and mutation probabilities are a reliable approach [26]. This paper studies the problem of how changes in the four GA parameters (population size, number of generations, crossover and mutation probabilities) in isolation or in combination have an effect on GA performance in the context of function optimization problems.

3. EXPERIMENTAL DETAILS

The overall computational effort required to run a GA is proportional to the number of function evaluations needed. Number of function evaluations (S) that is to be assigned for an application is product of number of generations (T) and the population size (N) i.e. S = T × N. The minimum number of function evaluations required for an application depends on the nature of the function being optimized. It is understood that if function is 'difficult' than number of function evaluations required would be higher. Following are the major complexities that may be present in an arbitrary problem: 1. Multi Modality 2. Deception 3. Isolation 4. Collateral noise. In this research we have chosen a one-variable uni-modal function, a two-variable function from the De Jong test function bed, a two-variable uni-modal function and a two-variable four-peaked function. Each test function is described below.

A. Test Functions

One-variable Uni-modal function:

This function has only one optimum solution. We evaluate the function: $f_1(x) = \frac{x}{Coeff.}$. The actual value of *coeff.* is chosen to normalize the *x* variable for a bit string of length 30 bits. Thus, *Coeff.* = $2^{30} - 1$ which is equal to 1073741823.

Two-variable De Jong Function:

Our second function: $f_2(x_1,x_2)$ is a two variable function from De Jong five function test bed. The function has convex characteristics. It is a two variable uni-modal function:

$$f_2(x_1,x_2) = 100(x_1^2 + x_2)^2 + (1 - x_1)^2$$

When the search space is restricted to the range of $-2 \le x_1, x_2 \le 2$, it has a single maximum point at (−2, −2) with a function value equal to 3609. Variables x_1 and x_2 are represented as 10 bit binary strings. Hence total search space is 1024 × 1024.

Two-variable Uni-modal function:

Our third function: $f_3(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ is a two-variable uni-modal function (Himmelblau function) often used in optimization literature [25]. The search space is of range $0 \le x_1,x_2 \le 6$ in which the above function has a single minimum point a (3,2), with a function value equal to zero. Here we converted minimization problem into maximization problem by using the formula:

$$f_3'(x_1,x_2) = \frac{1}{(1+f_3(x_1,x_2))}$$

x_1,x_2 are represented as 10 bit binary strings and the total search space is 1024 ×1024.

Four-peaked function:
This function is same as the previous one, but the ranges for x_1,x_2 are extended to $-6\leq x_1,x_2 \leq 6$. The function has a total of four minima, one in each quadrant. All minima have function values equal to zero. In order to make one of them the global minimum, we add a term to the above function.

$$f_3(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 0.1(x_1 - 3)^2 (x_2 - 2)^2$$

As x_1, x_2 are represented as 10 bit binary strings; the total search space is 1024 × 1024.

All the above functions have been tested for various crossover and mutation probabilities for fixed function evaluations and population sizes in our earlier work [26]. The results obtained can be summarized as follows:

- Mutation alone approaches perform better for only simple problems.
- Crossover alone based GA performs better than mutation alone based GA for all the function tested and for all parameters.
- GA with all three operators (Crossover, Mutation and Selection) performs better compared to crossover alone and mutation alone based GAs.

In this study we have limited the computational resources employed by choosing the total number of function evaluations S = 5000. In order to reduce bias in the initial population, we run each experiment with 50 different initial populations. Four program in C++ were implemented for this study (one for each function) for varying population sizes and number of generations.

4. EXPERIMENTAL RESULTS AND ANALYSIS

Our earlier research [26] assessed the functioning of the GA for different combinations of crossover and mutation parameters. However it did not provide a clear answer to the question of the effect of varying population sizes keeping number of function evaluations constant, on the results obtained by GA.

In continuation of our previous work in this paper, we present a formal statistical analysis of the results of our previous work, and the effect of varying population sizes on the robustness of GA.

A. Mutation Alone

Mutation plays a secondary role in the operation of GA. Mutation adds random diversity in the population, which helps in avoiding the possibility of getting trapped in a local optima. Here each bit is selected probabilistically and then flipped (assuming a bit level representation). Mutation reverses a 0 to a 1 and vice versa. The mutation operator considered due to its high explorative power.

- Statistical data presented in Table I shows that there is an increase in the mean values as mutation probability increases from 0.01 to 0.2. Mean values in function f_1 increases from 0.9746 to 0.9830. This is due to the fact that high mutation rate sometimes deletes low order schemata and introduces new desirable high order schemata.
- To gauge reliability of mean and to compare the series with regard to stability, we have calculated the standard deviation. It was seen that dispersion for function f1 increases to 0.017 as the mutation rate is increases. This is due to the fact that diversity in the population increases due to high mutation rate.
- There is a considerable decrease in solution quality or performance of GA for function f_2, f_3, f_4 as shown in Table 1. Mean values for the functions f_2, f_3, f_4 for all mutation parameters tested are comparatively low as compared with function f_1. Thus, it is clear that effect of mutation alone based GA is deleterious for complex functions. Thus mutation based GA fails for difficult problems.

In furtherance of our experiments, in this section we discuss the effect of mutation alone based GA with varying population sizes on robustness of GA:

- When mutation alone with selection is applied with varying population sizes, it is observed from Figures 1 and 2 that performance is poorer for small population sizes. As shown in Table 1, as the population size increases the mean value also increases. In case of small population size, the graph is noisy. This is due to the fact that probability of getting an optimal solution in the next generation is small. A small population has a low probability of taking step in right direction. When an incorrect step is taken, it requires several generations to come back to the optimal point, thus causing the GA to spend a large number of function evaluations (S) or generations (T) in searches before returning to optimal point as shown in Figures 1 and 2. Whereas with large population sizes diversity in the initial population is expected to be large and the best solution is expected to be close to optimal point. The dispersion value for small population sizes (P=20) is large considered with other population sizes as shown in Table 4. This is due to the fact that high mutation rate (M=0.2) combined with low population sizes disrupt the schemata in every generation more frequently.

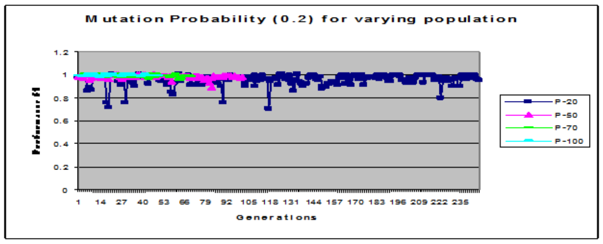


Fig. 1. Maximum performance (f1) measure versus generations with different population sizes for mutation alone based GA.

- There is a considerable increase in mean values of each function as the population size increases as shown in Table 2. Thus there exists a minimum population size, below which GA will have difficulty in reaching optimal point. The observation leads us to conclude that mutation based GA perform somewhat poorly for small population size and moderately for large population.

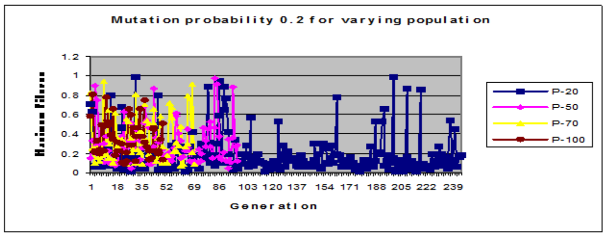


Fig. 2. Maximum performance (f3) measure versus generations with different population sizes for mutation alone based GA.

Table 1. Mean function values for different mutation / crossover probability settings

Function	Only Mutation				Only Crossover				Mutation with Crossover			
	M= 0.01	M= 0.05	M= 0.09	M= 0.2	C= 0.25	C= 0.5	C= 0.75	C= 0.95	C=0.9, M=0.01	C=0.9, M=0.1	C=0.1, M=0.01	C=0.1, M=0.1
f1	0.974	0.972	0.985	0.983	0.984	0.98	0.997	0.991	0.997	0.994	0.988	0.986
f2	2686	2608	2772	2779	3167	3283	3461	3561	3474	3400	3002	2728
f3	0.205	0.256	0.279	0.292	0.926	0.978	0.979	0.671	0.991	0.962	0.744	0.368
f4	0.217	0.249	0.285	0.306	0.924	0.968	0.988	0.996	0.995	0.952	0.881	0.477

TABLE 2. Mean function values for varying population sizes

Function	Only Mutation M= 0.2				Only Crossover C=0.75				Mutation with Crossover M=0.01, C=0.75			
	P=20	P=50	P=70	P=100	P=20	P=50	P=70	P=100	P=20	P=50	P=70	P=100
f1	0.95	0.985	0.986	0.997	0.842	0.907	0.938	0.949	0.9	0.932	0.988	0.973
f2	2587	2689	2765	2794	2398	2941	3368	3393	2393	2740	2484	2539
f3	0.1766	0.29	0.34	0.391	0.752	0.903	0.798	0.826	0.758	0.967	0.889	0.952
f4	0.166	0.164	0.201	0.247	0.728	0.894	0.809	0.83	0.764	0.956	0.893	0.947

TABLE 3. Standard deviation of function values for different mutation / crossover probability settings

Function	Only Mutation				Only Crossover				Mutation with Crossover			
	M= 0.01	M= 0.05	M= 0.09	M= 0.2	C= 0.25	C= 0.5	C= 0.75	C= 0.95	C=0.9, M=0.01	C=0.9, M=0.1	C=0.1, M=0.01	C=0.1, M=0.1
f1	0.02	0.032	0.014	0.016	0.012	0.019	0.006	0.003	0.004	0.005	0.0075	0.0138
f2	294.8	509.4	441.1	453.9	373.2	49.37	3.79	29.93	143.48	143.96	335.04	401.31
f3	0.285	0.161	0.197	0.209	0.008	0.01	0.05	0.0001	0.0103	0.059	0.352	0.3614
f4	0.267	0.127	0.172	0.193	0.007	0.013	0.067	0.00002	0.0101	0.019	0.332	0.3458

TABLE 4. Standard deviation of function values for varying population sizes

Function	Only Mutation M= 0.2				Only Crossover C=0.75				Mutation with Crossover M=0.01, C=0.75			
	P=20	P=50	P=70	P=100	P=20	P=50	P=70	P=100	P=20	P=50	P=70	P=100
f1	0.047	0.014	0.011	0.005	0.014	0.016	0.002	0.0049	0.057	0.0478	0.053	0.0918
f2	510.4	473.9	415.5	396.8	234.8	329.6	390.7	500.35	220.83	239.58	285.54	258.74
f3	0.198	0.205	0.190	0.189	0.097	0.164	0.16	0.183	0.0786	0.059	0.0587	0.0511
f4	0.188	0.200	0.195	0.199	0.098	0.172	0.161	0.184	0.077	0.058	0.059	0.052

(In Tables 1 to 4: M = Mutation Probability, C = Crossover Probability, P = Population Size)

B. Crossover Alone

In GA literature crossover is considered to be a powerful operator. Role of crossover is of constructive i.e. of preserving in nature. Crossover constructs higher order hyper planes from lower order hyper planes that have higher observed average fitness. Crossover is used where qualities of construction and survival are required for good performance. The results obtained are discussed below:

- Mean function values for C = 0.75 is high compared to other crossover probabilities for all function as shown in Table I. Mean values obtained are higher for all crossover probabilities when compared with mutation based GA as shown in Table 1.
- It is seen that as generations progress, the function value becomes stagnant. This is due to the fact that crossover loses its power as the population loses its diversity (i.e. number of common alleles increases) and it limits the types of exploration that crossover can perform. Standard deviation readings for crossover in Table 3 shows us that there is a significant decrease in standard deviation for all crossover probabilities when compared with mutation alone based GA.

In furtherance of our experiments, in this section we discuss the effect of crossover alone based GA with varying population sizes on the robustness of GA:

- From Figures 3 and 4 it is seen that for population less than N = 50, the function value degrades and the optimal solution is not reached. This is due to the fact that the population loses its diversity at an early stage due to which the function value stagnates as there is no mechanism to add diversity in the population. Performance of GA improves as population size increases. This is due to the fact that diversity in the population also increases, increasing the power of the crossover operator. Mean function values increase as population size is increased as shown in Table 2.

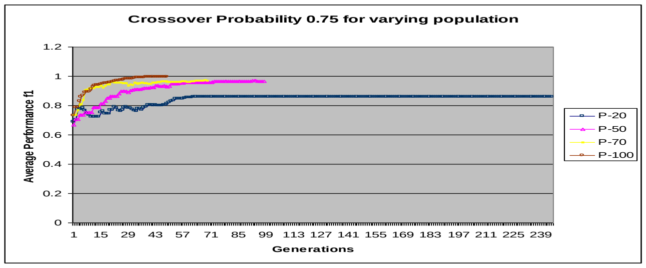
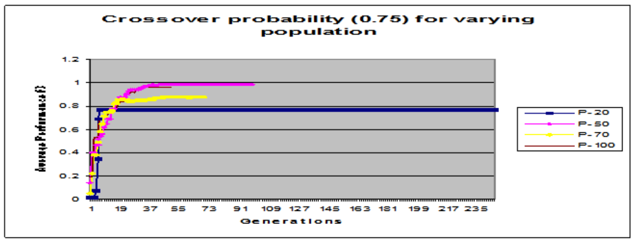


Fig. 3. Average performance (f1) measure versus generations with different population sizes for mutation



alone based GA.

Fig. 4. Average performance (f3) measure versus generations with different population sizes for mutation alone based GA.

- For large population sizes, the initial population will have a larger probability of having more individuals in the population having fitness close to optimal. The task of selection operator is to select such good solutions, in the middle of noise, from the initial population. Thus as these good solutions are selected repeatedly, the population loses diversity after each generation and performance stagnates. This is shown by the observation that mean values progressively decreases and standard deviation increases for large population sizes as shown in Table 3 and Table 4.

C. Crossover and Mutation Combined

It has been discussed earlier that role of mutation is to increase diversity in the population thus preventing the solutions getting trapped in local optima. Role of crossover is to construct and preserve good building blocks. Crossover guarantees preservation of alleles, while mutation guarantees diversity. We have performed experiments with different parameters with these two operators, since they are complimentary in nature. Based on our results we have found similarity in the performance of all the four functions described in Section 3. The following are the observations:

- Mean function values for all combinations of crossover and mutation probabilities are high when compared with crossover and mutation alone based GAs for all functions as shown in Table I.
- There is a significant increase in the mean value for C = 0.9 and M = 0.01 as shown in Table 1. Is also seen from the results obtained that as mutation rate is increased to a value of M = 0.1, the function value starts to resemble a random noisy search, and that the mean value decreases considerably. This is due to the fact that mutation destroys already found good solutions or building blocks in the population.
- Dispersion value for crossover probability C = 0.1 and mutation probability M = 0.1 for all the function is highest as shown in Table 3, which tells us that the data series is noisy in nature due high mutation probability and low crossover probability.

In furtherance of our experiments, in this section we discuss the effect of crossover with mutation based GA under varying population sizes on the robustness of GA:

- For varying population sizes keeping number of function evaluations (S) constant, results have shown that there is an 'optimum' population size below and above which average performance degrades as shown in Figure 5 and 6. This is because, for small population sizes there is less diversity in the population and since mutation rate is small (0.01), there will be very little diversity added to the population in each generation. Hence GA performance degrades. In case of large population sizes, the performance does not reach optimal mark due to the limitation in number of generation (T) and as the number of function evaluations (S) is held constant, it stops near the optimal point without actually reaching optimality. For all population sizes, the performance is much better compared to crossover alone and mutation alone based GAs.
- Mean values for all the functions are highest for population size 50 as shown in Table 2. Standard deviation for all functions increase as the population sizes increase, in keeping with increase in population diversity.

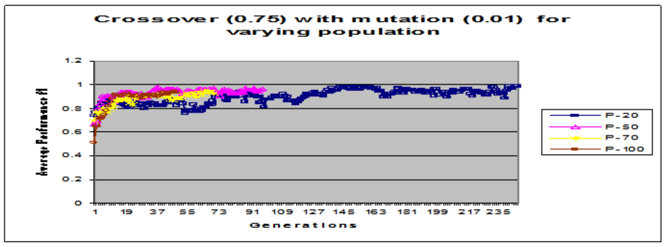


Fig. 5 Average performance (f1) measure versus generations with different population sizes for crossover & mutation alone based GA.

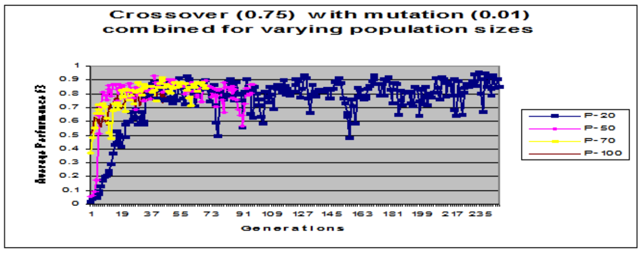


Fig. 6 Average performance (f3) measure versus generations with different population sizes for crossover & mutation alone based GA.

5. CONCLUSIONS

In this paper we have studied the effect of changes in GA parameters from a view of varying population sizes keeping number of function evaluation constant. Our aim is not to find the best population size for a particular problem, but to come up with general conclusions. We have applied GA for four different functions (one-variable uni-modal function, two-variable function from De Jong test function bed, two-variable uni-modal function and a two-variable four peaked function). We can derive the following conclusions from our study:

- Results have shown that for simple problems mutation based approaches perform better. As complexity of problem increases, possibility of mutation alone based GAs to achieve global optima decrease. Mutation based GA fail miserably for complex problems.
- Results obtained by mutation based GA are poor for both small and medium population sizes.
- Crossover based GAs perform better than mutation based GAs for all the functions we had tested. But they have a tendency to get stuck at local optimum.
- For simpler problems mutation based GAs requires larger population size as compared to crossover based GAs to achieve the same level of optimality.
- In order to achieve good performance when the total number of function evaluations is held constant, correct population sizing is important.
- In general, a high crossover rate combined with low mutation rate with a correct population size is a reliable approach.

6. FUTURE WORK

We have been able to demonstrate empirically, the role of some parameters in the process of natural selection and randomized, structural recombination in artificial genetic search. In our zeal to keep things simple we have however, neglected several others interesting natural operators. Our research work is limited in that sense. There are many knowledge-augmenting natural genetic operators, that can be implemented and tested. For example, it is seen that in nature genetic constitution does not easily forget the lessons learned in previous environmental shifts. The redundant memory of diploidy permits multiple solutions to be carried along, with only one particular solution being expressed. In this way old lessons are remembered and tested occasionally. Analysis and implementation of various advanced genetic operators like diploidy-dominance, inversion, intra chromosomal duplication and deletion can be implemented for further improvement of GA. We strongly feel that more energy is needed to be spent in finding correct implementation of these operators. So that they limit the intensive use of computational recourses made by GA and reduces time to reach optimal solutions.

7. REFERENCES

[1] D. E. Goldberg, K. Deb, “A comparative analysis of selection schemes used in genetic algorithm,” In: Rawlins, Gregory J.E. (Ed.), Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, Inc., pp. 69–93, 1991.

[2] R. G. Harik, F. G. Lobo, “A parameter-less genetic algorithm,” IEEE transactions on evolutionary computation. 1999. [Online]. Available: http://w3.ualg.pt/~flobo/papers/plga-gecco99.pdf

[3] O. Boyabalti, I. Sabuncuoglu, “Parameter selection in genetic algorithms” System, Cybernatics & Informatics. Volume 2-Number 4, pp. 78-83, 2007. [Online]. Available: http://www.iiisci.org/journal/CV$/sci/pdfs/P409090.pdf

[4] V. A. Cicirello, S. F. Smith, “Modelling GA Performance for Control Parameter Optimization,” Proceedings of Genetic & Evolutionary Computing Conference. GEECO-2000. [Online]. Available: http://.ri.cmu.edu/publication_view.html?pub_id=3329

[5] Y. J. Cao, Q. H. Wu, “Optimization of control parameters in genetic algorithms: a stochastic approach,” International journal of systems science, volume 30, number 2, pp. 551-559, 1999. [Online]. Available: http://www.informaworld.com/smpp/content~db=all~content=a713866076~frm=abslink

[6] A. E. Eiben, Z. Michalewich, M. Schoenaur, J. E. Smith, “Parameter control in evolutionary algorithms,” Proceedings of Genetic & Evolutionary Computing Conference, 1999.

[7] N. Radcliffe, “Forma analysis and random respectful recombination,” Proceedings of 4th International conference on genetic algorithms, 1999.

[8] H. Kargupta, K. Deb, D. E. Goldberg, “Ordering genetic algorithms and deception,” Parallel problem solving from nature 2. pp. 47-53, 1992. [Online]. Available: http://www.illigal.uiuc.edu/pub/papers/Publication

[9] J. C. Culberson, “Mutation-Crossover Isomorphism’s and the construction of discriminating functions,” Evolutionary Computation 2(3). 279-311, 1994.

[10] D. E. Goldberg. Genetic algorithm in search, optimization & machine learning. New York: Addison Wisley, 1989.

[11] W. M. Spears, K. A. De Jong, “On the virtues of uniform crossover,” Proceedings of the Fourth International Conference on Genetic Algorithms, 230-236. La Jolla, CA: Morgan Kaufmann, 1991.

[12] W. M. Spears, K. A. De Jong, “An analysis of multi-point crossover,” Proceedings of the Fourth International Conference on Genetic Algorithms, 230-236. La Jolla, CA: Morgan Kaufmann, 1993.

[13] D. E. Goldberg, "Sizing populations for serial and parallel genetic algorithms,” In: Schaffer, J.D. (Ed), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, Los Altos, CA, pp. 70–79, 1989.

[14] H. Kargupta, K. Deb, D. E. Goldberg, “Ordering genetic algorithms and deception,” Parallel problem solving from nature 2. pp. 47-53, 1999.

[15] S. Rana, “The distributional baises of crossover operators,” Proceedings of Genetic & Evolutionary Computing Conference, 1999.

[16] K. A. De Jong, , W. M. Spears, “An analysis of the interacting roles of population size and crossover in genetic algorithms,” Proceedings of the International Conference on parallel problem solving from nature. Springer. pp. 38-47, 1990.

[17] W. M. Spears, “Adapting crossover in evolutionary algorithms,” Proceedings of the Fourth International Conference on Evolutionary programming, 1995. [Online].

Available: http://www.cs.uwyo.edu/~wspears/papers/ep95.pdf

[18] W. M. Spears, "Adapting crossover in a genetic algorithm," Artificial intelligence center internal report # AIC-94-019, 1995. [Online]. Available: http://www.cs.uwyo.edu/~wspears/papers/adapt.cross

[19] H. Muhlenbein, "How genetic algorithms really work I. Mutation and Hillclimbing," Foundation of genetic algorithms II pp. 15-25, 1992. [Online]. Available: http://muehlenbein.org/mut92.pdf

[20] D. M. Tate, A. E. Smith, "Expected allele coverage and role of mutation in genetic algorithms," Proceedings of the 5th International Conference on Genetic Algorithms, pp. 31- 37, 1993.

[21] M. Srinivas, L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE transactions on Systems, Man & Cybernatics, Vol. 24, No. 4, 1994. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=286385&tag=1

[22] R. Hinterding, H. Gielewski, T. C. .Peachey, "The nature of Mutation in Genetic Algorithms," Proceedings of the 5th International Conference on Genetic Algorithms, 1995.

[23] M. D. Vose, "A closer look at mutation in genetic algorithms," Annals of Mathematics and Artificial Intelligence, Vol. 10, No. 4, pp 423-434, 1994.

[24] H. E. Aguirre, K. Tanaka, "Parallel varying mutation genetic algorithms," IEEE transactions, 2002.

[25] K. Deb. Optimization for Engineering Design. Algorithms and Examples. Prentice Hall of India. New Delhi, 2000.

[26] V. Kapoor, S. Dey, A. P. Khurana, "Empirical analysis and random respectful recombination of crossover and mutation in genetic algorithms," International Journal of Computer Applications. Special issue on Evolutionary Computation for Optimization. ECOT, 2010. pp. 5-30, 2010. [Online]. Avaliable: http://www.ijcaonline.org/specialissues/ecot/number1/1530-133