Journal of Computations & Modelling, vol.3, no.2, 2013, 111-124
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2013

Mutative Genetic Algorithms

Taisir Eldos[1]

Abstract

Genetic Algorithms have been successfully used for a long time in solving optimization problems in so many diversified fields. Most of the research effort focused on devising a suitable mapping for the problem in hand, or proposing efficient types of operations, finding an optimal set of parameters like crossover and mutation rates, mutation step size and crossover format, or selection methods towards optimizing the search in the sense of reducing the run time, computational burden, escaping local minima, etc.. In this research work, we present an intensive mutation with fitness based step size as a main player in the space exploration and exploitation. The test results show that mutation can be as good as the crossover operation in upgrading the population fitness, and the longer time it takes to execute population-wide mutation pays off in terms solution quality under time constraint and chance of getting an optimal solution, compared to the classical genetic algorithm.

Keywords: Genetic Algorithms, Knapsack Problem, Mutation, Adaptive Operator

[1] Department of Computer Engineering, Salman bin Abdulaziz University, Saudi Arabia, & Jordan University of Science and Technology, Jordan.

Article Info: *Received* : April 28, 2013. *Revised* : June 2, 2013
Published online : June 20, 2013

1 Introduction

Genetic Algorithms have been applied to a wide range of problems, with acceptable performance despite of the long run time, which is quite natural due to its evolutionary nature. GAs are exceptionally good in solving problems with large solution space and non-differentiable fitness landscape. Typically, they use two basic functions to change the individuals of a population towards more fit one generation after generation. The evolution process employs three main operators; selection, crossover and mutation. The first operator selects the fittest individuals to survive to the next and transfer the genetic content to new populations in each generation, the second operators mates highly fit pairs of individuals to produce new individuals with a good probability of having higher fitness values. Mutation, on the other hand is a low rate unary operator; one or more of the offspring is slightly disturbed to get a clone. Mutation is necessary to maintain an acceptable diversification degree in the population, which might get at risk due to successive selection of most fit individuals for crossover operations.

The encoding and parameters settings depends on the problem in hand, but the general behavior of the Classical Genetic Algorithm can be expressed by the following outline:

1. Choose the initial population of individuals randomly
2. Compute the fitness of each individual in the initial population
3. While Stopping Criteria = False,
   3.1. Select the best-fit individuals for reproduction
   3.2. Apply crossover operation to give birth to offspring
   3.3. Apply mutation operation to one of the offspring
   3.4. Compute the fitness of the new individuals
   3.5. Replace least-fit population with new individuals

The effects of genetic operators on the search vary widely, and hence a good mix is needed; selection alone tends to fill the population with copies of the best

individuals from the population, while mutation alone induces a random walk through the search space. However, selection and crossover operators together tend to cause the algorithm to converge on a local optima; a good but sub-optimal solution, while selection and mutation together create a parallel, noise-tolerant, hill climbing algorithm. The wise use of those three operations with properly sized population offer a search strategy with good possibility of converging at global optima within practically reasonable amount of time. Typical sopping criteria include; time limit, number of generations, less than threshold improvement rate, sufficient fitness achieved, etc.

Most of the previous work in this fields assumed a relatively low rate of mutation in an effort to mimic the nature from which the concept was borrowed; evolutionary theorem's survival of the fittest. A typical mutation rate of $1/n$ for n binary values chromosome, and operation carried out on the offspring following a crossover operation. Also, a light disturbance is recommended in what is known as step size. In this research, we consider the mutation as an active player in both the exploration and exploitation process, in its rate and step size aspects. We apply mutation with the following three things in mind: stand alone primary operation that gets applied to solutions rather than offspring, relatively high rate compared to the standard genetic algorithms and typically involve every individual in the population, and a step size that is determined by the fitness of the individual to be mutated or cloned.

Adaptation of strategy parameters and genetic operators has become an important and promising area of research in genetic algorithms. Today, the focus is more on solving optimization problems using adaptive techniques as away to cope with the dynamics of the solution spaces.

As an alternative to the fixed rate mutation, being relatively high or low, and to the adaptive mutation operations which most of the time adapts the rate to the dynamics of the population, the "mutative" scheme pushes the rate to the limit by operating on every individual, but with an adaptive size.

The inspiration of this idea came from the psychology, where "mutative" was coined by Strachey (1934) in relation to transference interpretations aimed at modifying the superego. To achieve this not one but a great number of mutative interpretations might be needed. Today it is more appropriate to use this happy term in a broader sense to refer to all those procedures and events in therapy, which bring about a shift, a change in the patient. In the context of optimization and using genetic algorithms in particular, a patient is a low fitness solution in the population and intensive mutation (in rate and step size) is what would bring such a solution into a higher fitness state causing the whole population to get better, which is highly likely to produce an optimal solution.

2 Literature Review

In 1975, John Holland laid the foundation for Genetic Algorithms (GAs), which have been since then used as a popular alternative to the classical optimization methodologies [1]. Since then, researchers have suggested different static mutation probabilities for GAs. These static mutation probabilities are derived from experience or by trial-and-error; 0.001, 0.005, 0.01, 1/n, 0.5/n, 2/n, 2.5/n, and even $1.75/(N * n^{1/2})$, where N and n are the population size and length of individual, respectively. It is very difficult to find an appropriate parameter setting for the optimal performance [2].

Instead of fixed operator and parameters, an ensemble of mutation strategies and control parameters from a pool may coexists throughout the evolution process, to compete in offspring production. In [3], performance improvements was demonstrated through a set of bound-constrained problems in comparison with conventional methods and several state-of-the-art parameter adaptive variants. The problem with this approach is that it needs pre-processing time to figure out strategies and parameters for such a pool.

Typical self-adaptation mutation starts with a small frequency, that increases generation after generation. Tests results show success through reducing the chance of premature convergence to a local minima due to the use of the elitism in the selection process [4]. The mutation probability increases generation after generation, test on knapsack problems shows that it outperforms state of the arts algorithms [5]. A mutation operator, based on greedy search and a distortion element [6], has increased the genetic algorithm performance in solving the TSP; two different greedy search methods tested on 14 different TSPLIB examples have shown more effective results in terms best and average error values. Another adaptive mutation rate [7] makes use of the state of the sandpile and the fitness values of the population. The results show that, at least under the proposed framework, a genetic algorithm with the sandpile mutation self-adapts the mutation rates to the dynamics of the problem and to the characteristics of the base-function.

The feasibility of using adaptive operators in genetic algorithms as opposed to using fixed parameters found using some methods of optimal ones was examined through a set of experiments, with a conclusion that although the adaptive genetic algorithms tend to need longer time to run, the price is worth to pay as the time spent finding the optimal mutation operator and rate for the non-adaptive versions can be considerable [8].

Another way to maintain sufficient diversity in the population, is a gene based adaptive mutation scheme [9], where the information on gene based fitness statistics and on gene based allele distribution statistics are correlated to explicitly adapt the mutation probability for each gene locus over time. A convergence control mechanism is combined with the proposed mutation scheme to. Experimental results show that the proposed mutation scheme efficiently improves the genetic algorithm performance.

A selective mutation method for improving the performances of genetic algorithms [10], ranks individuals then mutates one bit in a part of their strings

which is selected in correspondence with their ranks. Tests on four optimization problems have shown that it could escape local minima and find an optimal solution faster.

However, the performance of different adaptive mutation operators depends on the test problem and the gene level adaptive mutation operators are usually more efficient than the population level adaptive mutation operators, as reported by [2], in a comparative analysis of different population-level and gene-level adaptive mutation operators for genetic algorithms based on a set of benchmark optimization problems.

3 The Problem

As a proof of concept, the binary or 0/1 knapsack problem is used for evaluation. The general problem can be described as follows: given two sets of n items and m knapsack constraints (or resources), for each item j a profit p_j is assigned and for each constraint i a consumption value w_{ij} is designated. The goal is to determine a set of items that maximizes the total profit, not exceeding the given constraint capacities $\boldsymbol{c_i}$. Formally:

$$\text{Maximize} \quad z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{Subject} \quad \sum_{j=1}^{n} w_{ij} x_j \leq \quad \text{i=1, 2,} \tag{2}$$

$$\text{Where} \quad x_j \in \{0,1\} \quad \text{j=1, 2,} \tag{3}$$

This problem is included in the general class of covering and packing problems. According to Gottlieb [11], these two types of problems are structurally equivalent since we can locate the global optima on the boundaries of the feasible regions. In the particular case of the MKP, the feasible solutions contained on the boundary cannot be improved since the insertion of more items will cause the violation of resource capacities. While the uni-dimensional knapsack problem is solvable in pseudo-polynomial time (only weakly NP-Hard), the multi-

dimensional knapsack problem is strongly NP-Hard [12]. Hence, exact techniques and exhaustive search algorithms are only of practical use in solving instances of small size, making evolutionary algorithms more significant in instance of large size.

4 Intensive Mutation Approach

Research in the evolutionary algorithms has focused on applying them to specific problems, with two major lines of approach; applying a standard implementation of an evolutionary algorithm to a new problem, and proposing techniques to improve the performance of previous approaches. The focus has always been on showing a way to solve a problem with these algorithms, with less attention to how the proposed approach was able to deal with the dynamics of the solution space.

The choice of mutation rate is a vital factor in the success of any genetic algorithm, and for permutation representations this is compounded by the availability of several alternative mutation operators. It is now well understood that there is no one “optimal choice”; rather, the situation changes per problem instance and during evolution. This is why self-adaptation pops up as a choice; it has been proven to be successful for mutation step sizes in the continuous domain, and for the probability of applying bitwise mutation to binary encodings, and [8] examines whether this can translate to the choice and parameterization of mutation operators for permutation encodings.

The proposed algorithm, Mutative Genetic Algorithm, is a two-phase variant; in phase one we perform crossover operation only, just like classical genetic algorithm but with no mutation, and in the second phase, perform mutation operation only. the mutation is intensive in the sense that it involves every individual, but with a step size inversely proportional to its fitness. The least fit

among all are dropped to maintain the population size fixed. The outline of the proposed is depicted in the pseudo-code below:

1. Choose the initial population of individuals randomly
2. Compute the fitness of each individual in the initial population
3. Repeat on this 2-phase generation until termination (time limit, sufficient fitness achieved, etc.):
   3.1. Crossover Phase
      3.1.1. Select the best-fit individuals for reproduction
      3.1.2. Apply crossover operation to give birth to offspring
      3.1.3. Compute the fitness of new individuals
      3.1.4. Replace least-fit population with new individuals
   3.2. Mutation Phase
      3.2.1. Compute the mutation step size for every individual
      3.2.2. Apply mutation operation to individuals with own step sizes to create a clone
      3.2.3. Compute the fitness of new individuals
      3.2.4. Replace least-fit population with new individuals

The idea behind the variable step size mutation is that high fit solutions may need only a small perturbation to get to a better state while a large perturbation is likely to move it to a far state with possible decrease in fitness. On the other hand, low fit solutions have not much to lose, in the sense that large perturbations are likely to take them into better states, while small ones are likely to keep them around the low fitness part of the solution space. However, in the unlikely event that this leads to an unwanted outcome, they get ejected from the population soon. This approach enhances the chances of escaping local minima that the selection process is likely to fall into. With this in mind, we introduce a step size that is inversely proportional to the fitness and with pre-set limit:

$$S(i) = 1 + (L-1)\frac{F_{max} - F(i)}{F_{max} - F_{min}} \tag{4}$$

where *S(i)* is the step size of the i^{th} solution, $F(i)$ is the fitness of the i^{th} solution, $F_{\min}$ and $F_{\max}$ are the minimum and maximum fitness of the population, and the L is a limiting value, representing the largest step size, we will use $L = n/4$ for an n-value string representing a solution.

The 2-phase loop of each generation takes more time than the classical 1-phase loop due to the intensive mutation, but this extra computational burden is greatly compensated for by the outcome after every generation, which pays off immediately in better chance of more ones. The step size represents the amount of change to clone an individual; in the binary knapsack problem it represents the number of bits to be flipped. This mutation scheme adapts to the population characteristics; when the population is highly diversified and fitness span wider range, the step size automatically decreases, and this happens naturally in the early stage, while it tends to enlarge when the crossover starts giving birth to quite similar individuals.

Mutation and crossover operations are performed alternatively; the population undergoes a crossover in a generation and mutation in the next, and each stage involves all the individuals. In the crossover turn, the highest fit half of the population is mated as random pairs, while in the mutation turn, every individual is mutated with a step size determined by its fitness. the selection process at the end of each generation maintains the population fixed.

5 Results

Tests on fitness landscape using fitness distance measures and correlation measures have shown that the selection of a suitable representation is crucial when solving combinatorial optimization problems, and that encodings with a strong heuristic bias are more efficient and the addition of local optimization techniques further enhance its performance [13]. The standard test problem sets available lack

sufficient diversity particularly in the correlation structure and the constraint slackness settings, and using test problems that provide an insufficient breadth of diversity leads to questionable heuristic performance generalization, particularly since the structure of industrial problems are ill-defined [14].

To avoid encoding problems, the uni-dimensional knapsack problems is used, due to its simple straightforward encoding, and to avoid any performance bias due to ill-defined data sets, we opted to use Knapsack instances with random integral values (in the range 1 to 100 for the weight and profit vectors), with low complexity set of instances: 20, 30 and 40 objects, where the optimal solution can be found for every instance in a reasonable amount of time, and medium complexity: 40, 60, 80 and 100 for the best effort tests. Most of the tests were conducted using concurrent runs of the same code on laboratory machines with i7 CPU at 3.4 GHz, 8 MB Cache and 4 GB DRAM running C programs generated from Matlab Scripts and clock based seeds for the random number generators.

Table 1: Mutative vs. Classical Performance

	GA-Classical		GA-Mutative	
Quality	Time (min)	Generations	Time (min)	Generations
80%	7.15	10,234	7.30	10,076
85%	8.93	12,793	8.98	12,393
90%	11.35	16,246	10.96	15,120
95%	14.86	21,283	13.48	18,598

Table 1 shows how the proposed model performs in comparison with the classical genetic algorithm; GA- Classical seems to perform better only when the expectations are low, while the GA-Mutative beats it when higher quality solutions are to be found. In this experiment, Time (min) and Generations to reach

Target Quality (% of Optimal), Size=40 Objects, Capacity=1863, Optimal =3246, and population size of 40.

Table 2 compares the performance in terms of the success rate; the possibility of getting an optimal solution in a given amount of time. The two versions have equal performance in problems with small size, while the mutative version has significant increase in success rate for larger problems. We made 20 runs per instance, and population size=number of objects.

Table 2: Mutative vs. Classical Success Rate

		Success Rate	
Problem Size	Run Time (min)	GA-Classical	GA-Mutative
20	20	70 %	70 %
30	30	65 %	70 %
40	40	55 %	65 %

Table 3: Mutative vs. Classical Population Fitness Stats

		Final Population Stats (Values)	
	Initial Population	GA-Classical	GA-Mutative
Max	1857	2982	3145
Min	1225	2743	2794
Mean	1604.6	2761.7	2974.3
Range	632	239	351
Std. Dev.	172.8	81.3	126.2

Table 3, shows the population fitness characteristics of the two algorithms, the range (difference between max and min fitness values), stays relatively large, which is a sign of more diversified population when read in the context of larger standard deviation. We used 10 runs for 40 minutes each; Size=40 Objects, Capacity=1863, Optimal=3246, and Population Size=40.

Figure 1 shows the progress of the two algorithms over time, while the average is about the same in the early stages, the mutative algorithm (solid line) continues to improve the population average fitness at better rates. The setting used here is: Size=60 Objects, Capacity=2638, Optimal=3109, Population Size=40.

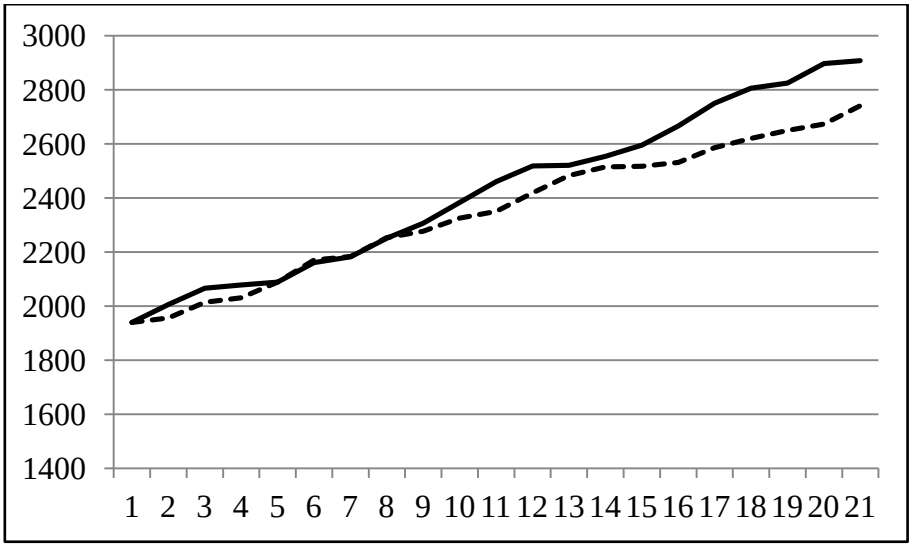

Figure 1: Mutative vs. Classical, Average Fitness Against Time (min)

6 Conclusion

Compared to the classical genetic algorithm, the variable step size intensive mutation proposed has shown slightly better performance in some aspects like the likelihood of finding global optima, which contradicts some of the findings that claims insignificant advantage of higher rates of mutation, and supports the

significance of adaptability in general. This opens the doors for better mutation application towards higher performance implementation while providing a general framework for optimization with genetic algorithms, as opposed to the genetic algorithm variants, that tends to be problem specific, through a set of tailored operations or parameters. While this approach is only slightly better than the classical implementation of genetic algorithms, several tests have shown that this improvement becomes more significant in handling larger solution spaces. Future work may focus on using MKP benchmarks, apply to other problems like TSP, reduce time by performing mutation every other generation, finding a better value for the limit *L*. Study ergodicity trend of the algorithm using a distance measure based homogeneity index.

References

[1] John Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology*, Control and Artificial Intelligence, MIT Press Cambridge, MA, USA, 1992.

[2] Imtiaz Korejo, Shengxiang Yang and Changhe Li, A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms, *The VIII Metaheuristics International Conference*, Hamburg, Germany, (2009).

[3] R. Mallipeddi, P. Suganthan, Q. Pan and M. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, *Journal of Applied Soft Computing*, **11**(2), (2011), 1679-1696, doi:10.1016/j.asoc.2010.04.024

[4] Sunith Bandaru, Rupesh Tulshyan and Kalyanmoy Deb, Modified SBX and Adaptive Mutation for Real World Single Objective Optimization, *IEEE Proceedings of Congress on Evolutionary Computation*, (2011), 1335-1342.

[5] Farhad Djannaty and Saber Doostdar, A Hybrid Genetic Algorithm for the Multidimensional Knapsack Problem, *International Journal Contemporary Mathematical Sciences*, **3**(9), (2008), 443-456.

[6] Murat Albayrak, Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms, *Journal of Expert Systems with Applications*, **38**(3), Pergamon Press, Inc. NY, USA, (2011).

[7] Carlos Fernandes, Juan Laredo, Antonio Mora, Agostinho Rosa and Juan Merelo, A Study on the Mutation Rates of a Genetic Algorithm Interacting with a Sandpile Applications of Evolutionary Computation, *Lecture Notes in Computer Science*, **6624**/2011, (2011), 32-42.

[8] Martin Serpell and James Smith, Self-Adaptation of Mutation Operator and Probability for Permutation Representations in Genetic Algorithms, *Journal of Evolutionary Computation*, **18**(3), MIT Press, USA, (2010).

[9] Shengxiang Yang and Sima Uyar, Adaptive mutation with fitness and allele distribution correlation for genetic algorithms, *Proceedings of the 21st ACM Symposium on Applied Computing*, (2006), 940-944.

[10] Sung Hoon Jung, *Selective Mutation for Genetic Algorithms*, World Academy of Science, Engineering and Technology, 2009.

[11] Jens Gottlieb, *Evolutionary Algorithms for Constrained Optimization Problems*, PhD thesis, Technical University of Clausthal, Germany, 1999.

[12] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.

[13] Jorge Tavares, Francisco Pereira and Ernesto Costa, Multidimensional Knapsack Problem: The Influence of Representation, Centre for Informatics and Systems of the University of Coimbra, *Technical Report*, (February, 2007).

[14] Raymond Hill and Chaitr Hiremath, Generation Methods for Multidimensional Knapsack Problems and their Implications, *Systemics, Cybernetics, and Informatics*, **5**(5), (2007), 59-64.