Genetic Algorithms in a dynamically changing environment

B. Dilimulati & I. Bruha
McMaster University, Department of Computing and Software, Hamilton, Ontario, Canada

Abstract

Genetic Algorithms (GAs) are search methods based on principles of natural selection and genetics. GAs attempt to find optimal solutions to a given problem by manipulating a population of candidate solutions (individuals). In the real world, we always encounter the problems that need to be solved in a changing environment. This means that our algorithm needs to be dynamic or even adaptive to the changing environment. In this paper, we mainly deal with the adaptive GAs that have a new genetic operator called *transformation* instead of the traditional crossover. We use a dynamic problem generator to create a dynamically changing landscape and study the behavior of the transformation-based GAs in different parameter settings, such as transformation rate, mutation rate and segment replacement rate.
Keywords: Genetic Algorithm, transformation operator, dynamically changing environment.

1 Introduction

Genetic Algorithms (GAs) are mainly used to solve optimization problems, see e.g. [1, 3, 4, 7]. In fact, there are various optimization methods such as exhaustive search, analytical optimization, line optimization methods, and natural optimization methods; natural optimization methods include simulated annealing, ant colony optimization, and genetic algorithms.

In traditional GAs, the operator set is usually fixed but in the real world we always encounter problems that need to be solved in a changing environment. Such problems include target recognition (the sensor performance varies on environmental conditions); scheduling problems (available resources vary over

WIT Transactions on Information and Communication Technologies, Vol 38, © 2007 WIT Press
www.witpress.com, ISSN 1743-3517 (on-line)
doi:10.2495/DATA070071

time); financial trading models (market conditions can change abruptly); investment portfolio evaluation (the assessment of investment risk change in time). These types of problems may experience simple dynamics where the fitness peaks representing the optimal problem solution drift slowly from one value to the next one, or complicated dynamics where the fitness peaks change more dramatically with the current peaks being destroyed and new remote peaks arising from valleys. To solve this kind of problems, our algorithm has to be dynamic or even adaptive to the changing environment.

In recent years there has been a significant research in upgrading the genetic algorithms to work more efficiently in dynamic environments [2, 5, 6, 8–10, 12]; most of this research could be grouped into one of these categories:

1. *Increasing diversity after change:* The GA runs in a standard fashion but as soon as a change in the environment has been detected explicit actions are taken to increase diversity and thus to facilitate the shift to the new optimum.
2. *Maintaining diversity throughout the run:* Convergence is avoided and it is hoped that a spread-out population can adapt to changes more easily.
3. *Memory based approaches:* The GA exhibits a memory that is able to recall useful information from past generations that seems especially useful when the optimum repeatedly returns to previous locations.
4. *Multi-population Approaches:* Multiple subpopulations are used, some to track known local optima, some to search for new ones.

Most of the adaptive GAs are trying to improve the diversity of the population so that the change in the fitness landscape can be detected by some individuals, importing random immigrants and placing sentinels.

The paper is organized as follows. Section 2 introduces the mechanism of transformation-based genetic algorithms. Section 3 briefly surveys the dynamic problem generator that is used for testing the performance of TGA. In Section 4, the performance of TGA is compared with other GAs, and also the TGA performance in different parameter settings is tested.

2 Transformation-based Genetic Algorithm

Transformation [13, 14] is a genetic operator inspired by the biological issue that, when incorporated into the genetic algorithms, can promote diversity in the population; in nature this operator occurs in colonies of bacteria. Usually, transformation consists in the transfer of small pieces of cellular DNA between organisms. These pieces of DNA (called *gene segments*) [13] are extracted from the environment and added to recipient cells.

Transformation-based GA (TGA) starts with a randomly generated initial population of individuals and a randomly generated initial *gene segment pool* [13]. Gene segments are used for transforming selected individuals; they thus act as foreign DNA pieces in bacterial transformation. In each generation, we select individuals to be transformed and apply transformation by using the gene

segments in the gene segment pool; then mutation is carried out. After that, the gene segment pool is updated by the individuals from the old population to create part of the new segments, and the rest of the segments are generated at random. One can see that the crossover operator in the standard GA is replaced by the transformation operator in the TGA.

After selecting individuals, we use the transformation mechanism to produce new individuals. We randomly select a segment from the gene segment pool, and also randomly choose a point of transformation in the selected individual. The segment is incorporated in the genome of the individual (chromosome), replacing the genes after the transformation point. It should be noted that the chromosome is seen as a circle. Figures 1 and 2 illustrate this transformation mechanism.

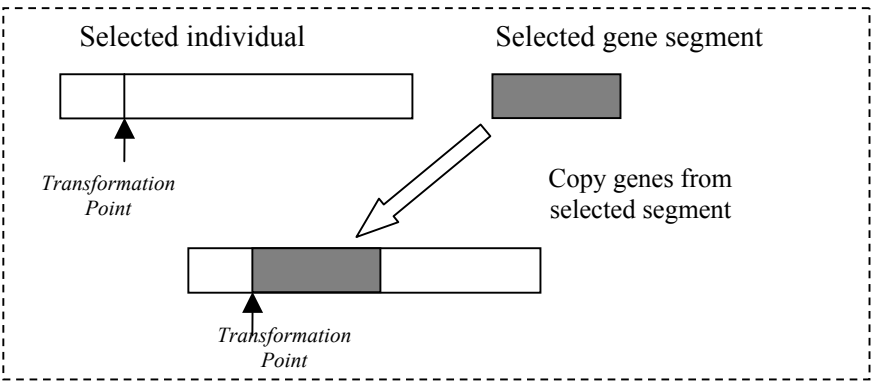

Figure 1: Transformation mechanism (the gene segment lies in the middle of the chromosome).

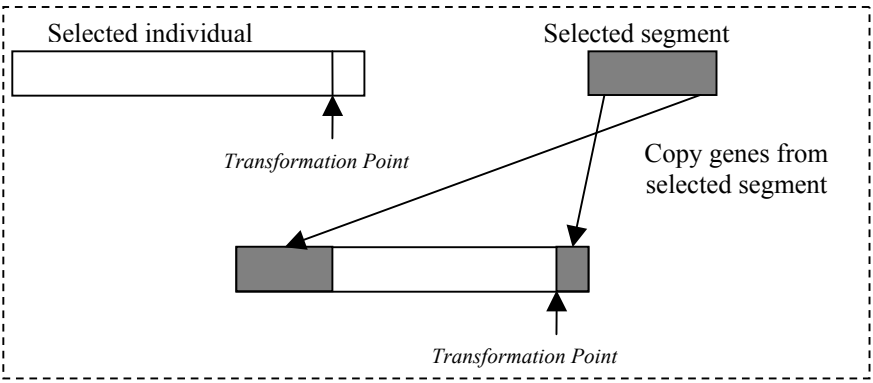

Figure 2: Transformation mechanism (the gene segment lies in the two ends of the chromosome).

3 Dynamic problem generator

Researchers working in the field of applying GAs in dynamic environments have developed various dynamic test functions; this paper introduces the dynamic problem generator that was developed by Morrison and De Jong [11].

The process of generating a dynamic problem can be divided into two steps: (i) constructing the shape of the fitness landscape; (ii) changing the landscape according to the user specified settings.

The morphology of the fitness landscape is the "field of cones" of different heights and various slopes that are randomly scattered across the landscape [11]. The static function can be specified for any number of dimensions. In the 2-dimensional case:

$$f(X,Y) = \max_{i=1,N} H_i - R_i * \sqrt{(X - X_i)^2 + (Y - Y_i)^2}$$

where N is the number of cones in the environment, (X_i, Y_i) specifies the location of each cone, H_i is the height of each cone, R_i is the slope of each cone (tangent value of the base angle).

In this dynamic problem generator, the features of the fitness landscape change in the discrete step sizes. To control the generation of a variety of different step sizes the following function was used:

$$Y_i = A * Y_{i-1} * (1 - Y_{i-1})$$

where A is a user-specified constant, Y_i is the value at iteration i.

4 Experiments

First, we have performed a comparative study between TGA and other common GAs, including *standard genetic algorithm* (SGA) [11] and *triggered hypermutation-based genetic algorithm* (HGA) [5]. Second, we have tested the TGA performance in various parameter settings. In this study, we use the offline performance as a measure for comparing the efficiency of different GAs.

(1) TGA, HGA, and SGA Performance in Dynamic Landscape

As we have already stated, we use the dynamic test problem generator to create the dynamic landscapes. We have fixed the peak heights so that the best fitness value is the same throughout all the generations. We exploit the following dynamics: moving the peak locations, and changing the peak slopes randomly. Consequently, we get a changing landscape.

We have tested the three algorithms TGA, HGA, and SGA in the environment where landscape changes every 20 generations. The test results are shown in Fig. 3. For all three algorithms, population size = 50, highest fitness value = 1873.00; SGA parameters are: crossover rate = 0.7, mutation rate = 0.01; TGA parameters are as follows: transformation rate = 0.6, segment replacement rate = 0.5, mutation rate = 0.001; HGA parameters were set as: crossover rate = 0.5, mutation rate = 0.01, hypermutation rate = 0.2.

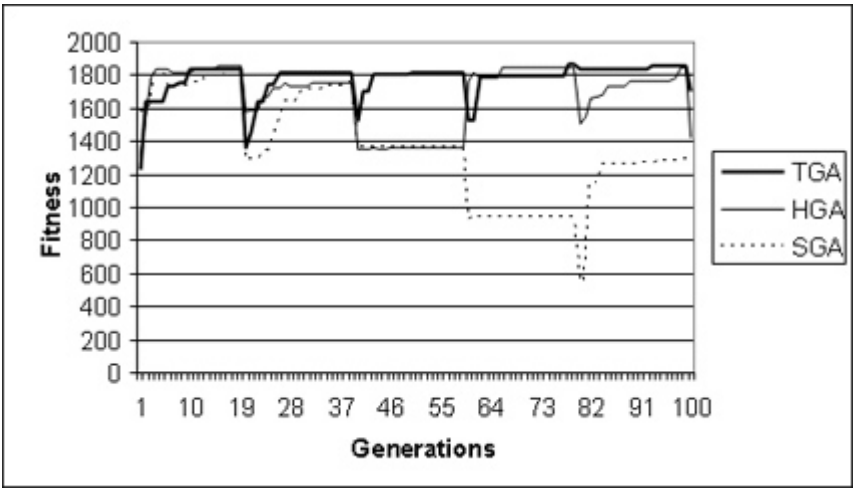

Figure 3: Highest fitness values of TGA, HGA, and SGA in a dynamic landscape.

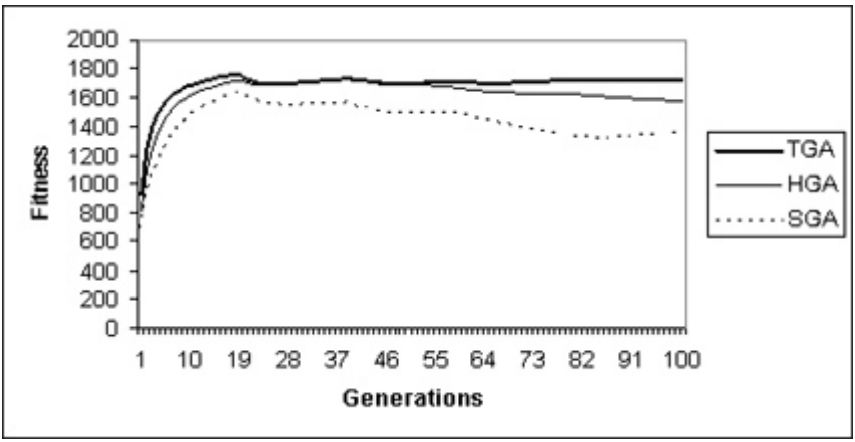

Figure 4: Offline performances of TGA, HGA, and SGA in dynamic landscape.

We also compare the offline performance of these three algorithms with the above parameter settings. We have repeated the test 20 times and calculated the average of the offline performances; see Fig. 4.

One can see according to the two figures that SGA behaves poorly in this dynamic landscape. The reason is that individuals tend to converge around the optimal peak before the landscape change, but after the landscape change, these individuals find themselves in lower fitness. In the next generations, SGA generates new individuals around a previous region which is now in a low fitness area. Consequently, SGA lacks *population diversity* (the extent to which

individuals spread evenly throughout the search space) which is vital in dynamic environments. The SGA genetic operators are crossover and mutation. The individuals created through crossover are most probably in the same region as their parents. The only mechanism that can increase the population diversity is mutation, but the SGA mutation rate is usually very small. Thus, SGA has a smaller chance of finding optimal peak in the dynamic environment.

One can also see that HGA exhibits almost identical performance as TGA, and far better than SGA. The reason is that HGA keeps track of the population fitness in every generation. If it finds some significant decline in population fitness, then the landscape has changed. In this case, HGA dramatically increases the mutation rate and consequently increases the population diversity.

(2) TGA performance in various parameter settings

In this section, we discuss the TGA performance in various parameter settings: *(a)* transformation rate, *(b)* segment replacement rate, and *(c)* mutation rate.

(a) TGA performance in various transformation rates

This test compares the offline performance of TGA with the landscape changes in every 50 generations. We repeat the tests 20 times and calculate the average of the offline performances; see Fig. 5. TGA parameters are set as follows: segment replacement rate = 0.5, mutation rate = 0.001.

One can see that TGA with the transformation rate of 0.7 performs better. The reason is that when the transformation rate is small then only small portion of the population is transformed. Therefore, the number of newly generated individuals is not large enough to increase the population diversity. When the transformation rate increases to 0.7, the number of new individuals becomes larger and, consequently, the population diversity also increases. But the transformation rate cannot be too high. If it were too high, then some individuals with higher fitness value would also be transformed. This could cause the destruction of these individuals (chromosomes), thus the overall performance of the algorithm would decrease.

(b) TGA performance in various segment replacement rates

The offline performance of TGA in various combinations of the segment replacement rates and the landscape change durations is tested. Again, the tests are repeated 20 times, and the average of the offline performances is calculated, see Tab. 1; the greatest values are in italic. Here population size = 50, number of generations = 100; TGA parameters are: transformation rate = 0.7, mutation rate = 0.001.

One can see that the segment replacement rate of 0.3 is good for all the dynamics. The reason is that only 30% of the segments are generated from the old population and the rest of them are generated randomly. This large number of randomly generated gene segments increases the population diversity. Consequently, the algorithm behaves well in the dynamic environments. If the segment replacement rate is too small, then almost all the gene segments are

generated randomly. In this case, it is possible that some segments with better genes may be replaced by randomly generated segments, thus decreasing the performance of TGA.

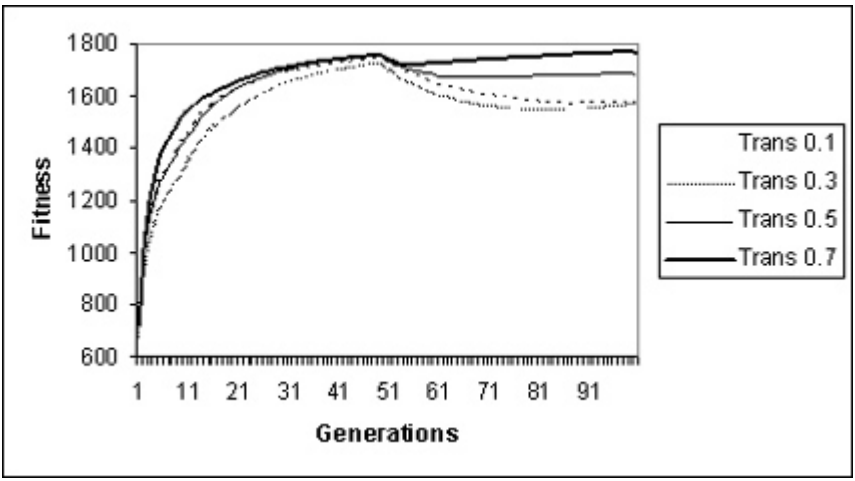

Figure 5: Offline performance of TGA with various transformation rates (Trans) in a landscape that changes every 50 generations.

Table 1: Offline performance of TGA in various segments replacement rates and various landscape change durations (LCD).

	LCD=10	LCD=20	LCD=50	LCD=100
segment.repl.rate = 0.1	1507.41	1566.26	1631.99	1682.25
segment.repl.rate = 0.2	1527.42	1583.00	1641.45	*1700.45*
segment.repl.rate = 0.3	1549.31	1588.26	*1683.25*	*1708.45*
segment.repl.rate = 0.5	1513.71	1570.31	1622.51	*1710.73*
segment.repl.rate = 0.7	1524.96	1564.39	1626.33	1645.3

(c) TGA performance in various mutation rates

Mutation is one of the most important genetic operators in traditional GAs. In this study, we have found that even the mutation can be replaced by the transformation in TGA. The offline performance of TGA in various combinations of mutation rates and landscape change durations is compared, see Tab. 2; the greatest values are in italic. In this test, GA parameters are identical to the previous experiment; segment replacement rate = 0.3.

One can see that TGA behaves better if there is no mutation or the mutation rate is very small. When the mutation rate increases then TGA performance decreases. Changing the genes may result in the destruction of good chromosomes, thus decreasing the algorithm performance.

Table 2: Offline performance of TGA in various mutation rates and various landscape change duration (LCD).

	LCD=10	LCD=20	LCD=50	LCD=100
mutation.rate = 0.0	1522.42	1576.56	1639.64	*1689.47*
mutation.rate = 0.001	1530.45	1578.65	1637.46	*1680.53*
mutation.rate = 0.005	1531.96	1580.83	1621.53	1650.25
mutation.rate = 0.01	1526.42	1576.56	1624.94	1641.48
mutation.rate = 0.05	1491.22	1528.63	1570.64	1603.43
mutation.rate = 0.1	1485.25	1435.52	1490.28	1510.31

5 Conclusion

The genetic algorithm called transformation using new genetic operator (inspired by biology) is presented. It is an alternative operator to the crossover one. In the transformation-based genetic algorithm (TGA), an individual is generated from a single parent and a gene segment. This differs from other GAs that use the crossover operator.

We have mainly focused on the methodology and implementation of general-purpose GAs rather than carrying out a huge set of experimentation. We have carried out some experiments and used the offline performance as a measure of algorithm efficiency. Analysis of these experiments includes the following.

• TGA with a higher transformation rate of 0.7 performs better. The reason is that higher transformation rate of 0.7 causes more new individuals to be generated; these new individuals replace the poor individuals in the old population, so it increases the overall performance of the algorithm. But the transformation rate cannot be too high. If it is too high, then some individuals with higher fitness value will also be transformed.

• Smaller segment replacement rate exhibits better performance in all dynamics. If the segment replacement rate is small, then only a small part of the segments is generated from old population and the rest of them are generated randomly. This large number of randomly generated gene segments increases the population diversity. Consequently, the algorithm behaves well in the dynamic environments.

• We have found that even the mutation operator can be replaced by the transformation in TGA. The mutation is used to increase the diversity of the population. In TGA, randomly generated gene segments can increase the diversity of the population, so the mutation can be replaced by the transformation operator if we set a proper segment replacement rate.

Future work may consist of (i) the study TGA with the variable length gene segments; (ii) the number of segments in the segment pool is also an important factor in TGA, therefore the relation between the population size and the segments pool size need to be studied, too.

References

[1] Affenzeller, M. & Wagner, S., Offspring Selection: A New Self-Adaptive Selection Scheme for Genetic Algorithms. *Adaptive and Natural Computing Algorithms*, pp. 218-221, 2005.

[2] Branke J., Evolutionary Approaches to Dynamic Optimization Problems – Updated Survey. *GECCO workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pp. 27–30, 2001.

[3] Bruha, I. & Kralik, P., Embedding a Genetic Algorithm in Attribute-Based Rule-Inducing Learning. *Soft Computing (SOCO-99), Symposium ICSC,* Genova, Italy, pp.631-635, 1999.

[4] Bruha, I., Kralik, P. & Berka, P., Genetic Learner: Discretization and Fuzzification of Numerical Attributes. *Intelligent Data Analysis Journal*, **4**, pp. 445-460, 2000.

[5] Cobb, H. G., An Investigation Into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environment. *NRL Memorandum Report 6760*, 1990.

[6] Cobb, H.G. & Grefenstette, J. J., Genetic Algorithms for Tracking Changing Environments. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.523-530, 1993.

[7] Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.

[8] Grefenstette, J. J., Genetic algorithms for changing environments. *Parallel Problem Solving from Nature,* **2**, ed. R. Manner and B. Manderick, Elsevier Science, pp. 137-144, 1992.

[9] Grefenstette, J. J., Evolvability in dynamic fitness landscapes: A Genetic Algorithm Approach. *Proceedings of the Congress on Evolutionary Computation*, **3**, IEEE Press, pp. 2031-2038, 1999.

[10] Lund, H.H., Adaptive Approaches Towards Better GA Performance in Dynamic Fitness Landscapes. *Technical Report DAIMI PB-487*, Aarhus University, 1994.

[11] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition,* Springer Verlag, 1996.

[12] Morrison, R.W., *Designing Evolutionary Algorithms for Dynamic Environments*, Springer Verlag, 2004.

[13] Simoes, A., & Costa, E., On Biologically Inspired Genetic Operators: Transformation in the Standard Genetic Algorithm. *Proceedings of the Genetic and Evolutionary Computation Cenference, GECCO-2001*, ed. W. B. Langdon et al, Morgan Kaufmann, pp. 584-591, 2002.

[14] Simoes, A., & Costa, E., Parametric Study to Enhance Genetic Algorithm's Performance when Using Transformation. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02),* ed. B. Langdon et al, Morgan Kaufmann, 2002.