Genetic algorithm with non-uniform genetic operators.

Alexey M. Akopyan *
S-Petersburg State University, Faculty of Physics, department of mathematical and computational physics

Fedor V. Sergeev †
S-Petersburg State University, Faculty of Physics, department of terrestrial physics

Abstract

In this paper new technique of mixing genetic material during GA run is suggested. This technique involves the feature called *gene resistance*. Resistances define individual behaviour of genes during recombination procedure and have a potential to improve GA performance. After introducing resistances in chromosomes their effect is described in concern with mutation and uniform crossover operators. Finally genetic algorithm that utilizes this new technique is presented and possible directions of future research are discussed.

Introduction

Effective mixing of properly coded information is highly essential for GA performance [1]. Much work is being done in the field of defining new and improving the existing genetic operators which implementsthis mixing. Particularly operator of crossover is supposed to be the one of the most significance for GA effectiveness. Crossover is usually defined on the pair of chromosomes and is responsible for passing genetic material between different individuals in population.

From the very beginning of GA research there were two principally different approaches for building crossover operator. First approach deals with parts of chromosomes which are identified by their position in the chrome chain. The corresponding class of operators contains n-point crossover which is very often used in implementations of GAs. Order-based crossover used

*ama@mph.phys.spbu.ru
†theodor@geo.phys.spbu.ru

in TSP solvers and linkage learning crossover operator recently suggested by Garik [4] also belongs to this class. The common feature of these operators is that they allow to efficiently manipulate building blocks which make up the solution. Theory involving building blocks and schemata [2] proved to be very fruitful in understanding of GA working.

Second approach deals with each gene in chromosome independently. This approach generates class of uniform crossover operators [3]. Uniform operators give no preference to any particular building block and often are considered less effective in schemata proceeding for this reason. From the other hand some empirical results showed that GA with uniform crossover can sometimes outperform GA with n-point operator. This gives the hope that uniform crossover still possesses some potential to contribute to GA performance.

The common property for the most of genetic operators is that they act "uniformly" at each gene position. Uniform crossover does that directly by treating each gene independently. During n-point crossover and all it's ancestors cut-points are also selected uniformly along the chromosome. Genes for mutation are picked up randomly with uniform distribution as well. The idea that made an impetus to research we want to announce in this paper was to provide GA with a possibility of non-uniform mixing of information coded in chromosomes.

Gene resistances

To bring new behaviour during recombination phase we introduce special feature called *resistance*. Resistance is the property of each gene in the chromosome. It defines the final probability for this particular gene to change. This works as follows: if gene i with resistance r_i has to be replaced by some genetic operator T, it gets replaced with the probability r_i and it remains the same with the probability $1 - r_i$. If more than one chromosome are affected by operator T (in the case of crossover for example) each gene is treated independently.

Resistances can serve to the different services. One of the possible utilization of this new feature is to help localizing the search in perspective areas after some information appears about search space properties. Once schema with high average fitness has been discovered corresponding genes can gain additional resistance to lower the chance of disrupting this schema. Resistances can also be used to direct the optimization process to explore particular areas of the search space.

Using resistances in GA. Extensions of genetic operators

It is possible to extend classical genetic operators such as mutation and

crossover so that they can work with resistances. Application of resistances for extending mutation operator is quite simple. For each gene which has to subject the mutation we check one additional random roll and in case of failing the gene preserves its value (resists). The distribution of resistances along the chromosome can be fixed or changed dynamically during GA run.

To understand the principles of extending the crossover operator we need to look what new things can happen when resistances are involved in the process. We have three different results of crossover now. First, alleles can pass between both chromosomes. This is what usually occurs during classical crossover. Second, genes in both chromosomes can resist the effect of this particular crossover operation and remain the same. And third, one of genes can successfully withstand crossover while another gene fails to resist. This is equivalent to copying first, more "strong" allele from one chromosome to another. This scheme is what we call standard extension of crossover operator since no new algorithm and operator parameters are used.

Besides standard extension of uniform crossover we also developed operator that affects each position in chromosomes taking in account the result of passing alleles in previous position. Namely this operator utilizes three different probabilities for exchanging alleles. First probability is checked if alleles have been exchanged in preceding gene. Second probability applied when no pass have been actually made, and third probability works when one of alleles has been copied from one chromosome to another (we make no difference between parents so it does not matter from which particular chromosome the gene comes). This algorithm was made to let the initially uniform crossover operator to exhibit "block" behaviour.

Let us name three probabilities from the previous paragraph p_{11}, p_{00} and p_{01}. And lets $p_{00} < p_{01} = p_{11}$. And also lets fix the initial probability for each chromosome as p_{00}. Crossover starts from the first gene and runs along the chain with p_{00} until first gene exchange occurs. Then it proceeds with probability p_{11} and because of p_{11} being greater than p_{00} there is a good chance of "block exchange". Particularly if $p_{11} = 1$ and $p_{00} = 1/l$ where l is the chromosome length, we have operator which behaves closely to 1-point crossover.

Implementation of GA with resistances. Experiments and results

To carry out our experiments we developed original program that implements the GA with resistances (RGA). This program is Win32 application built by Borland C++ Builder 3.0. Pack of classes used is rather standard for object oriented implementations of GA. It consists of *TGene*, *TChrome*, *TPopulation* and *TGASystem* classes for representing main entities of GA. There are also particular classes *TFitnessThread* and *TCrossover* for flexible

manipulating with different objective functions and crossover operators. Up to now three crossover operators are used in the testing – classical uniform crossover and two it's extended forms described above. Implementation details are probably out of subject of this paper so we omit them.

As an optimization problem we choose the task of fitting the experimental Mössbaur spectra with Lorentz curve of the form

$$L(x) = y_0 + \frac{1}{\gamma(x - x_0)^2} \tag{1}$$

Fitting had been made by minimizing the relative normalized error

$$err = C(N) \sum_{i=1}^{N} \frac{(y_i - L(x_i))^2}{y_i} \tag{2}$$

where x_i, y_i are experimental points.

This objective function was selected because we had working program written by I. Sergeev [1] (S-Petersburg State University, faculty of Chemistry) which implements classical least squares method for this problem. That provided us with possibility of comparing performance and precision of two algorithms. Since LS program (lets call it LSO) was highly optimized and used specific information to get starting point we did not really put much efforts to make RGA faster than LSO. But we strived for RGA to get precision at least the same as that of LSO. We also looked at robustness of the algorithm. Results are presented in 1.

Parameters of genetic algorithm were the following: *population size* = 50, 100, 150; *chrome length* = 32, 64; *crossover rate* = 0.5, 0.6, 0.7; *mutation rate* = 0.1, 0.05, 0.02 (some dynamical adaptation schemes was used as well); *generation gap* = 0.9. Tournament selection was used as a selection method.

Adaptation of resistances in RGA was quite simple: when a gene gets changed during reproduction it's resistance increases by small value δr. This is done to slow the elimination of new genes in population. From the other hand if gene remains the same for a long time, it's resistance drops by δr every 10 rounds.

As it can bee seen from the 1 RGA shows slightly better performance than standard genetic algorithm while specialized least squares algorithm outperforms both GA and RGA. We plan to apply dynamic parameter space scaling [5] to raise the accuracy of GA though there is a limit of accuracy that can be achieved on the given experimental set.

Conclusion: open questions and future work

Results we obtained during our experiments showed that using gene resistance has a potential to improve GA performance. It is obvious that

[1] ilya@moss.pu.ru

	LSO		GA		RGA	
x_0	166.267	165.572	166.172	165.625	165.572	165.568
y_0	194.0	167.5	190.0	216.6	167.7	170.6
γ	77.89	79.42	80.0	80.0	80.0	80.0
err	30.20	21.19	30.97	21.69	30.59	21.36

Table 1: Comparison of results obtained by least squares optimizer (LSO), standard genetic algorithm (GA) and genetic algorithm with resistances (RGA).

more experimental and theoretical work is required to clarify the scope of tasks where this feature can be used. It is also seems interesting to continue experiments with extended uniform crossover operator. We hope to get some experimental results on this topic soon.

The main problem with RGA is the adaptation of resistance distribution along the chromosomes. Intuitively it can be done by counting average fitness of alleles of each gene over population. For example one could raise resistance of those genes with high average fitness thus increasing their chance to get to next population. This will serve to the aim of more rapid convergence. From the other hand one could desire to maintain greater diversity over the population and in this case it seems to be useful to decrease the resistance of the gene which has the same value in large number of chromosomes. There are also other schemes of adopting the resistance values and we hope to investigate more of them in near future.

References

[1] Martin R.R. Beasley D., Bull D.R., *An overview of genetic algorithms: Part 2, research topics*, University Computing **4** (1993), no. 15, 170–181.

[2] Goldberg David, *Genetic algorithms in search, optimization and machine learning*, Reading, Addison-Wesley, 1989.

[3] Syswerda G., *Uniform crossover in genetic algorithms*, Proceedings of the Third International Conference on Genetic Algorithms (1989), 2–9.

[4] Harik Georges, *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*, Ph.D. thesis, University of Michigan, 1997.

[5] Belew R.K. Schraudolph N.N., *Dynamic parameter encoding for genetic algorithms*, Machine Learning (1992).