Using incremental evaluation and adaptive choice of operators in a genetic algorithm

Alexander Kosorukoff
Dept. of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
kosoruko@uiuc.edu

Abstract

Genetic algorithm (GA) is an optimization procedure that uses a simplified model of natural evolution to achieve an objective of optimization represented by a fitness function. The number of fitness function evaluations is a major factor determining the efficiency of a GA. This paper addresses the issue of incremental evaluation which, if applicable, substantially reduces the complexity of evaluation. Incremental evaluation can be used together with evaluation relaxation methods using quick estimates of the real value of the fitness function. Using incremental evaluation in a genetic algorithm puts more emphasis on the rational choice of trade-off between the amounts of time spent on local and global search. This paper contains utility analysis of crossover and mutation operators for one-max problem. It shows the utilities of these innovation operators change significantly during the run. Experiments with adaptive choice strategy maximizing the expected utility of operators show significantly faster convergence of the algorithm to the optimum solution.

1 Introduction

Genetic algorithm is an optimization procedure that uses a simplified model of natural evolution to achieve an objective of optimization represented by a fitness function. The important idea of evolution is a combination of local and global search operators within one computational procedure (Goldberg, 1989). From this idea a question immediately follows: how much time to spend on the local search and how much time to spend on the global search? This problem examined elsewhere (Goldberg & Voessner, 1999). A particular case of this general problem is the problem "crossover vs mutation" caused many arguments in the field. This problem can also be viewed from a slightly different perspective: where and which operator is better to apply? This formulation leads to adaptive probabilities of crossover and mutation instead of using some predetermined and fixed probabilities. Several authors suggested giving credit to the operators each time they produce innovation (Davis, 1989; Srinivas & Patnaik, 1994; Julstrom, 1995).

In this paper, the problem "crossover vs mutation" is formulated as a decision problem using multiattribute utility theory (MAUT) (Keeney & Raiffa, 1993). The multilinear utility function of operators is based on the model of GA innovation (Goldberg, 1983; Goldberg, 1998). After assessing the utility function, it is easy to use MAUT to find a rational trade-off between crossover and mutation.

The initial motivation for this work came from the comparison of human creativity and genetic algorithms. While mutation serves only an auxiliary role in genetic algorithms, it seems that most people use mutation-like incremental improvements much more often than radical crossover-like innovations. What is the rationale behind this human behavior? A possible reason becomes apparent if we take into account that the cost of evaluation of a new individual might differ depending on how this new individual was produced. It is often the case, that the result of a small mutation is easier to evaluate than the radically innovative result of crossover. In this paper, the costs of evaluation were taken into account and associated with each innovation operator through the use of utility function. For the purpose of this research, we are especially interested in the case where the computational cost of *incremental evaluation* of mutants is small in comparison with *full evaluation* of a significantly different individual. Such fitness functions will be further referred as *in-*

crementally evaluated fitness functions. The simplest example of this class is the one-max function: after flipping one bit of the genotype, it is not necessary to count ones, it is enough just to update (increment or decrement) the fitness of the original individual by one. The mutation of one bit in the genotype of length $l = 100$ requires an incremental evaluation taking only 1/100-th of a full evaluation. In this case the choice "mutation vs crossover" becomes not so obvious: low cost and low innovation vs high cost and high innovation.

The rest of the paper is organized as follows. Section 2 describes a decision problem of the optimal choice of operators. Section 3 contains general description of strategy and methods chosen to solve the problem. Section 4 describes the assessment of innovation utility of the operators. Section 5 describes the algorithm of adaptive choice of the operators. Section 6 presents the results of applying this algorithm to the one-max fitness function. And, finally, section 7 concludes the paper.

2 Problem statement

The decision problem in this paper is the optimal choice between two genetic innovation operators, crossover and mutation, representing global and local search, respectively. Although it seems quite simple, the problem has a long history of arguments between different schools of evolutionary computation in favor of using crossover or mutation, so that opinions of researchers in the field are highly polarized. The most usual approach in American school of evolutionary computation is the probabilistic combination of the two operators where crossover is dominant and used with high probability, while mutation serves an auxiliary role and applied rarely. In a classical implementation of GA these operators are coupled together, so it is often difficult to compare their contribution in the result of optimization.

The main conflicting attributes in our decision problem are the innovation rate produced by the two operators and the computational cost associated with their application. The choice of the function to be optimized will influence both attributes. Although the methodology described here is quite general, for the purpose of this paper, we are primarily interested with decomposable additive functions, where the evaluation of individual after mutation can be done incrementally, in other words, at a lower computational cost than the evaluation of an individual after crossover. Moreover, the fitness of the solution is calculated as a sum of fitnesses of its individual parts. An well known example of such fitness function is a one-max problem, usually used as as a test function and benchmark for genetic algorithms.

We would like to test a conjecture that mutation operator for fitness functions that can be evaluated incrementally (like one-max function) has advantage over crossover, and crossover has advantage in other cases. If this conjecture is true, then the strong preference of crossover over mutation (current practice in GA field) is not always the best decision. If the conjecture will be refuted this will support traditional way to design GAs. It might happen that in different stages of evolution different operators will be preferrable, then the best combination can be learned adaptively during the run using online learning algorithms.

3 Strategy and methodology

In the previous section we described the problem. The first step to address it is to uncouple operators and analyse the effect of each of them separately. The outcome of this will be a genetic algorithm in which crossover and mutation applied to individuals independently, unlike a traditional scheme where mutation immediately follows crossover.

After the effect of each operator can be analysed independently, we can gather statistics on applications of each operator and measure its contribution to the overall innovation rate. The operators have different costs and innovation rates, so it is not easy to determine which operator should be preferred and when. We would like to use multiattribute utility theory for this, constructing a utility function of two attributes: innovation rate, cost of application[1]. Later this initial set of attributes was changed to a set of three attributes described later.

We had chosen a one-max problem for our experiments, since it is the simplest function that can be evaluated incrementally. Apparently, the interesting results with one-max function can motivate further research of more complex fitness functions. The innovation rate of operators for one-max problem can be measured experimentally. The cost of application of each operator can be estimated theoretically for one-max problem. The set of design decisions includes pure crossover, mutation, or probabilistic combinations of them blended with different probabilities. The optimal solution will be defined by probabilities of application of each operator to achieve the desired trade-off between conflicting objectives. This includes the

[1]Cost is measured in terms of computational complexity of fitness evaluation

boundary cases where it will be best to use only one of the two operators and not to use the other.

The optimal solution will not necessarily be fixed, we expect the possibility that it will be different for different stages of evolutionary process. Apparently, the highly random population at the initial stage is quite different than the almost converged and more regular population in the late stages of evolution, so why optimal choices of operators should be the same? In order to account this, we apply the decision procedure every generation to update our operator preferences iteratively.

4 Utility function assessment

To approach the problem described in section 2 systematically, we use multiattribute utility theory (MAUT). We construct multiattribute utility function and use decision theory to find the optimum combination of operators that will maximize the expected utility. We will use this combination in our decision making.

Our first step was to get some intuition about the form of utility function and conduct the first experiments that will help to select a good set of relevant attributes. Intuitively the ratio of our initial two attributes, the innovation rate to the cost of innovation, seems to be a good efficiency measure. Apparently this kind of utility function is not representable by multilinear utility function usually used in MAUT (Keeney & Raiffa, 1993). However we would like it to be representable. This observation leads us to substitution of a cost C as an attribute with the number of possible applications at a fixed computational cost F. We can calculate $F = 1/C$ with a precision up to a constant coefficient. Now our initial guess utility function is representable as $U(I, F) = I * F$.

Before performing an assessment of utility function we perform experiments to measure the value of the selected attributes for mutation and crossover. In these experiment, the chromosome length and population size both equal to 100. The results averaged over five runs are shown in Table 1. Additional attribute, scrap rate is added, because later in the experiments we noticed its relevance.

According to MAUT, in order to choose the form of the utility function we must check if the assumptions of mutual and utility independence hold for the selected set of attributes (Keeney & Raiffa, 1993). We can easily check that the attributes chosen are preferentially independent. Indeed, at any fixed level of F and S we prefer the operator that will give us higher innovation rate I. Conversely, among the two operators with the same F and the same innovation rate I we certainly will choose the one with lower scrap rate S. In addition, under fixed I and S we always prefer less costly operator (with higher F). We can show utility independence conducting a set of lotteries, where the value of one of the attributes F, I, or S is uncertain. Analysis of these lotteries show that decisions taking into account a certainty equivalence of uncertain attribute doesn't change under different combinations of the pair of other fixed attributes. These lotteries show that utility function over the chosen set of attributes has the property of utility independence. In addition to this, we noticed that the innovation rate I and the scrap rate S are additive independent from each other.

Table 1: Value of attributes of our decision choices

Choice option	Crossover	Mutation
Attributes:		
Applications per fixed cost (F)	1	100
Innovation rate (I)	≈ 0.0251	≈ 0.00351
Scrap rate (S)	≈ 0.0056	≈ 0.00527

Table 2: Value of attributes of our decision choices

Attribute	Range	Value function
Applications per fixed cost (F)	$0 - 100$	$U(F) = F/100$
Innovation rate (I)	$0 - 0.03$	$U(I) = I/0.03$
Scrap rate (S)	$0 - 0.03$	$U(S) = 1 - S/0.03$

Now we can proceed and assess the value functions of each selected attribute independently: U(F), U(I), and U(S). Exact assessment of these functions is complicated, however our approximate assessment show that these value functions can be assumed risk independent and linear in the first approximation (shown in table 2). The linear value function respect the preferences toward lower cost, higher innovation rate, and lower scrap rate. Range of all value functions is $0 - 1$ under assumptions that the values of the attributes will be in the ranges specified in table 2.

The multiattribute utility function under these assumptions is the following:

$$U(F, I, S) = k_f U(F) + k_i U(I) + k_s U(S) +$$

$$+ k_{fi} U(F) U(I) + k_{fs} U(F) U(S)$$

where

$$k_f + k_i + k_s + k_{fi} + k_{fs} = 1$$

is a normalization constraint

The scaling coefficients $k_f, k_i, k_s, k_{fi}, k_{fs}$ can be evaluated by the usual MAUT assessment procedure. Apparently, for different problems this assesment can be different, because the trade-offs between the speed of optimization and the quality of solution can vary. We made a coefficient assessment for our particular case, using the lottery questions quantifying tradeoffs between the different attributes. Thus we got the following lotteries:

$$U(F^*, I^0, S_0) \approx (U(F^*, I^*, S^*), 0.00, U(F_0, I_0, S_0))$$

$$U(F_0, I^*, S_0) \approx (U(F_*, I^*, S^*), 0.06, U(F_0, I_0, S_0))$$

$$U(F_0, I_0, S^*) \approx (U(F_*, I^*, S^*), 0.04, U(F_0, I_0, S_0))$$

$$U(F^*, I^*, S_0) \approx (U(F^*, I^*, S^*), 0.04, U(F_0, I_0, S_0))$$

$$U(F_0, I^*, S^*) \approx (U(F_*, I^*, S^*), 0.10, U(F_0, I_0, S_0))$$

$$U(F^*, I_0, S^*) \approx (U(F_*, I^*, S^*), 0.04, U(F_0, I_0, S_0))$$

They correspond to the following constraints on the coefficients:

$$k_f = 0$$

$$k_i = 0.06$$

$$k_s = 0.04$$

$$k_f + k_i + k_{fi} = 0.66$$

$$k_i + k_s + k_{is} = 0.10$$

$$k_f + k_s + k_{fs} = 0.34$$

From these constraints we determine coefficients:

$$k_f = 0$$

$$k_i = 0.06$$

$$k_s = 0.04$$

$$k_{fi} = 0.6$$

$$k_{is} = 0$$

$$k_{fs} = 0.3$$

The following utility function is the result of our coefficient assessment for the test problem, optimization of one-max function:

$$U(F, I, S) = 0.06U(I) + 0.04U(S) +$$
$$+ 0.6U(F)U(I) + 0.3U(F)U(S)$$

Using values from tables 1 and 2 we can calculate utilities of crossover and mutation:

$$U(crossover) \approx U(1, 0.0251, 0.0056)$$

$$U(F) = 0.01$$

$$U(I) \approx 0.84$$

$$U(S) \approx 0.81$$

$$U(crossover) \approx 0.09$$

$$U(mutation) \approx U(100, 0.00351, 0.00527)$$

$$U(F) = 1$$

$$U(I) \approx 0.117$$

$$U(S) \approx 0.824$$

$$U(mutation) \approx 0.36$$

Apparently $U(crossover) < U(mutation)$, so the optimal strategy according to decision theory seems to always use mutation. However the experiments with only mutation produced inferior results: the fitness of solution rises quickly to some suboptimal level and can't reach the optimum for a long time. The latter result contradicts to our high estimate of the utility of mutation. The possible reason for this result could be that the attributes of the decision problem are not static, they can change during the optimization run in such a way that utility of mutation is higher during the run, but at the final stage becomes extremely low such that mutation can hardly finalize the experiment.

This hypotesis about dynamic changes of the attributes during the run was confirmed by new experiments. Figure 1 shows significant changes in the values of the two attributes: innovation and scrap rates. The attribute F is the only constant attribute during the run. From this figure, it immediately follows that average values of the changing attributes that we estimated earlier do not capture the situation: choosing one static decision based on the averaged values of attributes is not a good strategy in this situation. However, since we have an analytical expression for our utility function, we can revise our decision for each generation based on new information about the current values of the attributes. Taking this adaptive strategy we will keep track of changing values of our

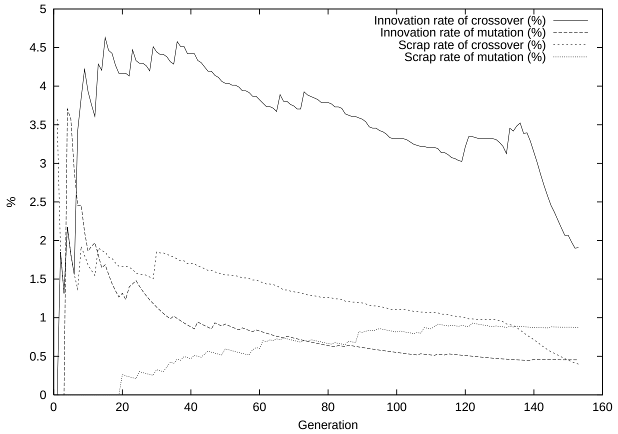

Figure 1: Change of innovation and scrap rates during the run

Table 3: Value of attributes of our decision choices

Attribute	Range	Value function
Applications per fixed cost (F)	$0 - 100$	$U(F) = F/100$
Innovation rate (I)	$0 - 0.05$	$U(I) = I/0.05$
Scrap rate (S)	$0 - 0.05$	$U(S) = 1 - S/0.05$

attributes. Of course, our value functions for innovation and scrap rates should be corrected, since the graph shows that these parameters can go as high as 5% during the run, so we update their ranges and value functions. Updated value functions for adaptive strategy are shown in table 3. The following sections will show that an adaptive decision strategy is indeed can be implemented and works.

5 Decision algorithm

The decision algorithm is simple. We estimate innovation rates as historical frequencies of innovation produced by the two operators, estimate their utilities and make a probabilistic choice between the operators. We learn the optimal probability of crossover p_x by perceptron algorithm. We start from $p_x(0) = 0.5$ (corresponding to no knowledge about relative efficiency of crossover and mutation). If the calculated utility of crossover is higher than the calculated utility of mutation, we promote p_x additively: $p_x = p_x + d$, where d is a fixed learning rate parameter ($d = 0.05$ in our experiments). If the calculated utility of mutation is higher, we demote p_x: $p_x = p_x - d$. We keep the innovation and scrap rate frequencies current, updating them after each generation. The value of p_x is constrained to be in the interval $[\delta, 1-\delta]$, where δ is some margin ($\delta = 0.05$ in our experiments). This constraint is necessary because we learn a dynamic concept, the optimal probability of p_x that is changing from generation to generation. Without this constraint, p_x would have converged to zero at the beginning of the run because of initially low utility of crossover. In this case, there is no way to detect the fact that the utility of crossover increased later in the run just because we stopped using crossover and have no idea what its effects might be.

The margin δ and the learning rate d are the parameters of our learning algorithm. There is a trade-off connected with these parameters. Higher values of δ and d correspond to the faster adaptation to a dynamic change, however they also mean lower quality of approximation of the optimal value of p_x. Additional research is needed to explore this trade-off. By now, experiments have shown that the extreme values of δ and d produce inferior results: one of the extremes correspond to the classic GA practice $p_x(0) = 0.75$, $\delta = d = 0$ (fixed probability of crossover 75%). The other extreme is $\delta = 0.05$, $p_x(0) = \delta$, $d = 1 - 2\delta$ produced results inferior to those presented in the following section.

6 Results

In this section we will compare the results obtained by running the genetic algorithm with different fixed values of probabilities for sampling of operators and with the dynamically adjusted probabilities.

We estimate the overall performance of the algorithm by the total cost needed to achieve the theoretical fitness maximum. For the traditional algorithm with crossover probability $p_x = 0.75$ the cost averaged for five runs was 2568.30. For the adaptive algorithm that use perceptron rule to adjust probability of application of operators the cost was 1334.18, that is approximately twice lower than in the traditional case.

The analysis of adaptive sampling of operators shows that the utility of application of mutation operator is usually higher during the whole run, however the situation changes near the end of the run. This is detected by adaptive algorithm and it changes its preferred innovation operator. Probably this is one of the reasons of improved performance. The typical example of adaptive sampling is shown in Figure 2. Initial probability of crossover $p_x = 0.5$. We can see how the algorithm decreased it to $p_x = 0.05$. We limited the range of it to be between 0.05 and 0.95 to detect the situation, where the situation suddenly changes and the previously efficient operator becomes inefficient.

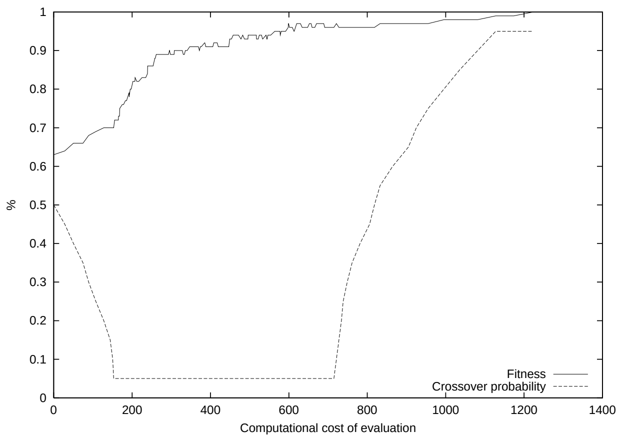

Figure 2: Adaptive sampling of operators

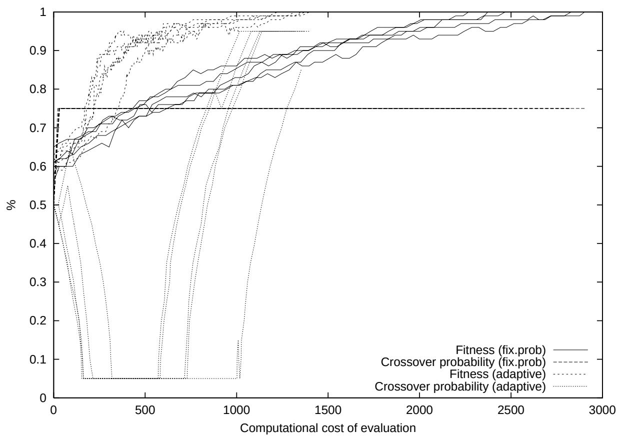

Figure 3: Comparison of adaptive sampling vs sampling with fixed probabilities

We can see this at the Figure 2, where the algorithm changes the preferred operator after about 700 fitness evaluations because the mutation operator becomes inefficient at this point.

Figure 3 shows the comparison of five experiments following traditional sampling of operators with five experiments with adaptive sampling. It can be seen that adaptive strategy saves us approximately half of fitness evaluations. However this graph also shows that there is a room for improvement of the learning algorithm: in many cases the faster reaction on the change of relative utility of the operators would lead to efficiency improment.

Analysis of adaptation shows that the utility of mutation is highest at the beginning of the run and diminishes to the end of the run, until it becomes lower than the one of the crossover. This has a nice explanation in terms of building blocks theory of GA. The building blocks are not formed yet at the beginning of the run, so the expensive action of crossover combining building blocks together seems to be premature at this stage and its positive value is not realized, while the utility of mutation is highest at this stage. However, at the final stage of the run, the crossover operator becomes more useful than mutation, because most of the building blocks are already formed and need only be combined together that can be efficiently done by crossover operator. The utility of mutation at the end of the run is low and experiments with mutation alone show that it takes much longer to assemble these building blocks without using crossover.

7 Conclusion

This research has addressed one of the decision problems of engineering of genetic algorithm: the optimal trade-off between global and local search operators, crossover and mutation, respectively. It includes experimental results that provide a useful information for developers of genetic algorithms. Practical result of this research is a genetic algorithm that dynamically adjusts its preferences depending on the current utility of the operators. In experiments, this adaptive adjustment has shown two-fold reduction of the computational cost.

The possible extension of this research is testing if we will observe the same patterns in the optimization of more complex functions. Since decreasing utility of mutation and increasing utility of crossover is consistent with schema theorem, we expect to observe the same patterns of utility change for optimization of other test fuctions. Another extention is trying different machine learning algorithms to learn which operator to apply in the particular moment. The learning of the concept that is changing over time is a difficult problem and was beyond the scope of this work.

Acknowledgements

This work was supported by a research assistantship from the Technology Entrepreneur Center, University of Illinois at Urbana-Champaign.

References

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. *International Conference on Genetic Algorithms '89*, 61–69.

Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning*. PhD dissertation, University of Michigan.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E. (1998). *The design of innovation: Lessons from genetic algorithms, lessons for the real world* (IlliGAL Report 98004). Department of General Engineering, UIUC.

Goldberg, D. E., & Voessner, S. (1999). Optimizing global-local search hybrids. *Genetic and Evolutionary Computation Conference '99*, 220–228.

Julstrom, B. A. (1995). What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. *International Conference on Genetic Algorithms '95*, 81–87.

Keeney, R. L., & Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value tradeoffs.* Cambridge, MA: Cambridge University Press.

Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(4), 656–667.