Statistics-based Adaptive Non-Uniform Mutation for Genetic Algorithms

Shengxiang Yang

Department of Mathematics and Computer Science
University of Leicester
University Road, Leicester LE1 7RH, UK
s.yang@mcs.le.ac.uk

Abstract. A statistics-based adaptive non-uniform mutation (SANUM) is presented for genetic algorithms (GAs), within which the probability that each gene will subject to mutation is learnt adaptively over time and over the loci. SANUM uses the statistics of the allele distribution in each locus to adaptively adjust the mutation probability of that locus. The experiment results demonstrate that SANUM performs persistently well over a range of typical test problems while the performance of traditional mutation operators with fixed rates greatly depends on the problems. SANUM represents a robust adaptive mutation that needs no advanced knowledge about the problem landscape.

1 Introduction

Holland's *schema theorem* states that building blocks receive an exponentially increasing trials in the subsequent generations. Usually with the progress of the GA, the frequency of 1's in the alleles of these loci where building blocks reside will eventually converge to 1 (or 0). SANUM makes use of this convergence information as feedback information to control the mutation by adjusting the mutation probability for each locus. Let $f_1(i,t)$ ($i = 1 \ldots L$ where L is the string length) denote the frequency of 1's in the alleles in locus i over the population at time (generation) t and $p_m(i,t)$ ($i = 1 \ldots L$) denote the mutation probability of locus i at time t. Then, $p_m(i,t)$ can be calculated from $f_1(i,t)$ as follows:

$$p_m(i,t) = P_{max} - 2 * |f_1(i,t) - 0.5| * (P_{max} - P_{min}) \tag{1}$$

where $|.|$ is an absolute function, P_{max} and P_{min} are the maximum and minimum allowable mutation probabilities for a locus respectively. Now during the evolution of the GA, after a new population t has been generated, we first calculate $f_1(i,t)$ for each locus i over the population and from this obtain $p_m(i,t)$ for gene locus i. Then we can perform SANUM operations similarly as traditional bit mutation except that SANUM uses $p_m(i,t)$ for each locus i instead of a global mutation probability p_m for all the loci.

The motivation behind SANUM lies in the fact that with the progress of the genetic search SANUM helps protecting building blocks found so far while still exploiting new building blocks. With the progress of the GA, when building blocks are partially found, SANUM decreases the mutation probabilities of those loci where these building blocks reside according to Equation (1). In this way, SANUM can protect building blocks found so far. While on the other hand, for those unconverged loci the mutation probabilities remain high within SANUM. This is useful because there may be building blocks not expressed on these loci yet. SANUM strikes to balance the construction of new building blocks and protection of found building blocks with time adaptively.

As the population converges, with traditional bit mutation fewer and fewer offsprings generated by mutating converged loci survive the next generation. That is, many mutation operations are wasted on converged loci. SANUM can save these wasted mutations and hence wasted fitness evaluations through adaptively decreasing $p_m(i,t)$ from P_{max} to P_{min} for those converged loci.

2 Experimental Results

In order to test SANUM, it is compared with traditional bit mutation with a series of recommended fixed probabilities: $1/L$, 0.01, $1.75/(N * L^{1/2})$ where N is the population size, and 0.001 over a range of typical test problems. Within SANUM, $p_m(i,t)$ varied adaptively between $P_{max} = 1/L$ and $P_{min} = 10^{-4}$ according to Equation (1). In the experiment all GAs are generational and use the fitness proportionate selection with the stochastic universal sampling and elitist model, 2-point crossover with crossover probability 0.6, and the population size of 100. For each run, the best-so-far fitness was recorded every 100 evaluations. SANUM performs persistently well on the test problems. SANUM performs much better than traditional mutation operators on royal road functions R_1 and R_2 (see Figure 1) due to the strong building blocks built in them. After certain evaluations, when the GA has built up some useful schemas, SANUM efficiently avoids mutating those converged loci, i.e., found building blocks.

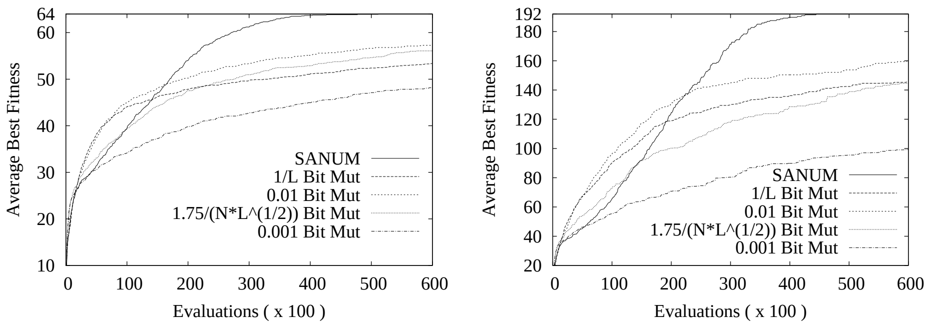

Fig. 1. Average best-so-far fitness against evaluations on Royal Road Functions (*Left*) R_1 and (*Right*) R_2. Experiment results were averaged over 100 independent runs