Solving Low Autocorrelation for Binary String (LABS) Problems using Genetic Algorithms

Practical Assignment 1

O.Aldabergenov, - November 8, 2017

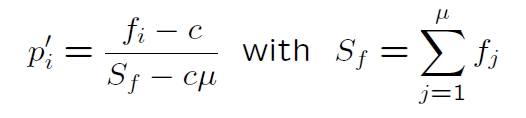
Introduction

Problem Description

Genetic Algorithm has specific way of solving problems rather than Monte Carlo search and we found difficulties on realization. Both algorithms have own constraints which are different from each other. First problem which we face was on generating population. There were two possible way of doing that: a) taking n-length bit string and equally divide it to subsets and take them as population (as shown in slide 3). b) generate multiple population of n-length bit string and in future select best of them. Secondly, problem with fitness function because at the beginning we use decoding function to evaluate subset. Last problem related to choosing right crossover and mutation probability.

Implementation

Important part of creating Genetical Algorithm is population. Starting from this point, our algorithm initialize log(n)\*100 sized cell array of population (pop\_size) which are filled by n-length bit string. Next, we evaluate each n-length bit string by using autocorrelation function (labs.m) which is already provided. Afterwards, we launch while loop which works till counts reach evaluation budget. In the loop we have parent selection part where probability calculated by using formula in Figure 1.



Only fourth population size will be selected as parents(ps\_size = pop\_size/4) to get fittest parents and some parents might be repeated. Moving to forward, we crossover parent by using probability of De Jong 1975 pcrossover=0.6 and create 3 times more new generation rather than selected parents to get better offspring. Then, we mutate it by using standard probability pm1= 1/n;

Generating population was one of the tricky moment on creating GA algorithm because slide 3 describes method which divide n-length bit string into subset. Each element become a population. Afterwards, each subset need to be decoded and evaluated. Our algorithm was wrong at the beginning, we made few mistakes. For example, code which is below:

xopt = rand(1,n) > 0.5;

fopt = labs(xopt);

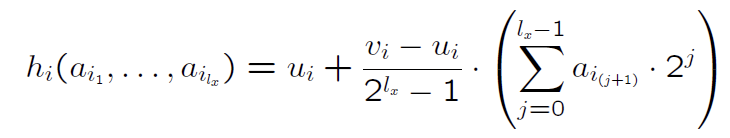
for i = 1:pop\_size

pop\_{i} = xopt((i-1)\*lx+1:i\*lx);

pop\_fit\_{i} = u + (((v-u)/(2^lx-1))\*bin2dec(num2str(pop\_{i})));

end

Firstly, we generate n-length bit string and evaluate it by using labs method (autocorrelation) and divide n-length bit string to equally subset. Eventually, it gives us pop\_size = n/lx (lx-length of bit string in subset). Next, we put each set to cell array and evaluate fitness by using decoding function.



After crossover, mutation we merge subsets and finally evaluate it by autocorrelation method. Many testing and comparing work to Monte Carlo search was done (not included to this report). We’ve thought that it works well because it almost gives us equal result, same as Monte Carlo search. However, we realize that we do not need decoding function to evaluate subsets and start use autocorrelation method. (See it below)

pop\_fit\_{i} = labs(pop\_{i});

Then again, nothing is changed. We’ve got same result. After many running algorithm we found that autocorrelation method gives us different result when you merge subset again. For example, let say that X,Y subsets of 10 length bit string . X = [0 1 0 0 0] , Y = [0 1 0 0 0]. Fitness of each subsets X,Y equals to 6.2500 respectively. Despite that this significant good result, we get 1.3514 after merging this subsets to one XY = [0 1 0 0 0 0 1 0 0 0].