A Relative Fitness Aware Rank-based Selection Procedure in GA

Jiling Zhong, Yan-Qing Zhang*, Member, IEEE

Department of Computer Science
Georgia State University
P.O. Box 4110
Atlanta, GA 30302
U.S.A.

Abstract

The rank-based selection procedure in GA has been considered superior to fitness proportional reproduction. But rank-based selection procedures totally ignore the information about the relative evaluations of different chromosomes. In this paper, a modified rank-based selection procedure is proposed. This selection procedure also considers the relative fitness and the distribution of different chromosomes. Another proposal is to modify the crossover function. The proposed crossover function is to K-group chromosomes instead of pairing them. Some experiments are conducted.

Keywords: genetic algorithms, selective pressure, rank-based selection, crossover..

1. Introduction

GAs exploit the idea of the survival of the fittest and an interbreeding population to create a novel and innovative search strategy. A population of strings, representing solutions to a specified problem, is maintained by the GA. The GA then iteratively creates new populations from the old by evaluating the strings and interbreeding the fittest to create new strings, which are closer to the optimum solution to the problem at hand. So in each generation, the GA creates a set of strings from the bits and pieces of the previous strings, occasionally adding random new data to keep the population from stagnating. The end result is a search strategy that is tailored for vast, complex, multimodal search spaces.

The idea of survival of the fittest is of great importance to genetic algorithms. GAs use what is termed as a fitness function in order to select the fittest string that will be used to create new, and conceivably better, populations of strings. The fitness function takes a string and assigns a relative fitness value to the string. The method by which it does this and the nature of the fitness value does not matter. The only thing that the fitness function must do is to rank the strings in some way by producing the fitness value. These values are then used to select the fittest strings. The concept of a fitness function is, in fact, a particular instance of a more general AI concept, the objective function.

The population can be simply viewed as a collection of interacting creatures. As each generation of creatures comes and goes, the weaker ones tend to die away without producing children, while the stronger mate, combining attributes of both parents, to produce new, and perhaps unique children to continue the cycle. Occasionally, a mutation creeps into one of the creatures, diversifying the population even more. Remember that in nature, a diverse population within a species tends to allow the species to adapt to it's environment with more ease. The same holds true for genetic algorithms.

It seems that there are two important issues in the evolution process of the genetic search: population diversity and selective pressure [1]. Selective pressure control is therefore one of the most important issues in genetic algorithms.

In the genetic algorithm proposed by Holland, the selection function is called proportional reproduction; it takes into account the relative fitness of the chromosomes.

Another type of selection function, Rank-based selection procedure, is widely used in genetic algorithms. But they also have their problems. Their major problem is they ignore the information about the relative evaluations of different chromosomes.

The rest of the paper is organized as follows. Section 2 introduces basic knowledge of selective pressure and rank-based selective procedures. Section 3 describes proposed algorithm. Section 4 shows experiment results. Finally, Section 5 summarizes this paper.

2. Selective pressure and rank-based selection

As observed by Whitley:" It can be argued that there are only two primary factors (and perhaps only two factors) in genetic search: population diversity and selective pressure. These two factors are inversely related. Increasing selective pressure results in a faster loss of population diversity. Maintaining population diversity offsets the effect of increasing selective pressure. In some sense this is just another variation on the idea of exploration versus exploitation that has been discussed by Holland and others."[2] In John Holland's canonical generic algorithms, fitness is defined by fi/f, where fi is the evaluation associated with string i and f is the average evaluation of all the strings in the population. This is known as fitness proportional reproduction. There can be a couple of problems with fitness proportional reproductions. First selection can be too strong in the first few generations: too many duplicates are sometimes allocated to very good individuals found early in the search. Second, as individuals in the population improve over time, there tends to be less variation in fitness, with more individuals being close to the population average. As the population average fitness increases, the fitness variance decreases and the corresponding uniformity in fitness values causes selective pressure to go down. In this case, the search begins to stagnate.

Other methods to sample a population are based on introducing artificial weights: chromosomes are selected proportionally to their rank rather than actual evaluation values [2] [3]. These methods are based on a belief that the common cause of rapid (premature) convergence is the presence of super individuals, which are much better than the average fitness of the population. Such super individuals have a large number of offspring and (due to the constant size of the population) prevent other individuals from contributing any offspring tin the next generations. In a few generations a super individual can eliminate desirable chromosomal material and cause a rapid convergence to (possibly local) optimum. There are many methods to assign a number of offspring based on ranking. In [2], a linear function and a parameter are definded:

$$\text{Prob (rank)} = q - (\text{rank} - 1)r,$$

Or a non-liner function,

$$\text{Prob (rank)} = q(1-q)^{\text{rank}-1}$$

Both functions return the probability of an individual ranked in position rank (rank = 1 means the best individual, rank = pop_size the worst one) to be selected in a single selection. Such approaches, though shown to improve genetic algorithm behavior in some cases, have some apparent drawbacks. The major problem is they ignore the information about the relative evaluations of different chromosomes.

3. Modified selective function

3.1. Take relative fitness information into account

The major problem of rank-based selective functions is that they ignore the information about the relative evaluations of different chromosomes.

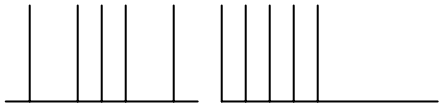

Fig. 1: Two sets of data having the same selective pressure

Take the above picture as an example.

According to the rank-based selective function, these two set of data will have the same selective pressure because they have the same ranks even their relative fitness is different.

In the proposed algorithm, the selective pressure from regular ranking function **Prob** is adjusted according to their relative fitness.

Prob' = Prob * 2/(1+exp(f-fi))

(Actually the constants in the function can be adjusted)

In which f is the average fitness; fi is the individual fitness.

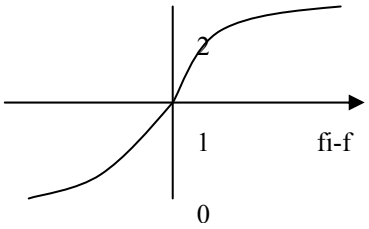

Fig. 2: Adjust the selective pressure based on chromosome's fitness.

The idea of this adjustment is to keep the selective pressure of those chromosomes which have the average fitness intact; to increase the selective pressure of those chromosomes which have the above average fitness and to decrease the selective pressure of those chromosomes which have the below average fitness.

3.2. Take fitness distribution information into account

Take the following graph as an example.

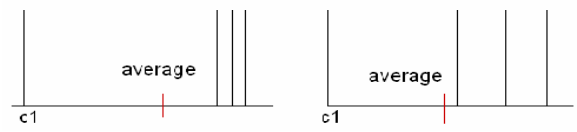

Fig. 3: Two sets of data having the same selective pressure

Assume the average of the above two data sets is the same, since the ranks for c1 in both cases are NO.1; the selective pressure for c1 is the same. But since the distribution of the two data sets is different, it might be beneficial if the selective pressure for c1 in these two cases is different.

An easy way to take the distribution into account is like this:

The population is divided into K intervals, the number of chromosomes falling into the respecting interval is counted. Then the selective pressure from regular ranking function **Prob** (or from **Prob'**) is adjusted according to their distribution.

Probj'' = Prob * 2/(1+exp(d-dj)) in which j is from 1 to K; d is the average NO of chromosomes for the whole range, dj is the average chromosome count for interval j.

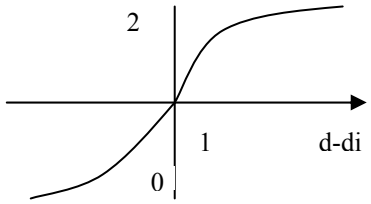

Fig. 4: Adjust the selective pressure based on chromosome's distribution.

The idea of this adjustment is to keep the selective pressure of those chromosomes which have the average distribution intact; to increase the selective pressure of those chromosomes which have the above average distribution and to decrease the selective pressure of those chromosomes which have the below average distribution.

4. Experiments and results

Benchmark function we used in this paper is the Perm(n, beta) function[6].

$f(x) = \Sigma_{k=1\sim n} \Sigma_{i=1\sim n} [i_k+beta][(x_i/i)^k-1]^2$

Suitable bounds which can be optionally imposed are x_i in [-n,n] for i=1,...,n.

The global minimum is at x_i=i with f(x)=0.

Suggested specific cases: (n,beta) = (4,50), (4,0.5), (10,10^9), (10,10^7);. The first and third are easy, and the second and fourth are hard.

Beta is a nonnegative parameter. The smaller beta, the more difficult the problem becomes since the global minimum is difficult to distinguish from local minima near permuted solutions. This problem therefore appears useful to test the ability of a global minimization algorithm to reach the global minimum successfully and to discriminate it from other local minima.

We tested these two cases and compared the original rank-based selective function with the modified selective function.

(n,beta) = (4,50) (rank-based)

```
#define POPSIZE 200
#define MAXGENS 1000
#define PXOVER 0.7
#define PMUTATION 0.1
#define TRUE 1
#define FALSE 0
#define N     4;              //Perm(N,beta)
#define BETA  50;             //Perm(N,beta)
generation    best      average
   900        0.961     1.014
   955        0.000     0.001
   956        0.000     0.000
the best result
x1 = 1.000 x2 = 2.000 x3 = 3.000 x4 = 4.000
```

(n,beta) = (4,50) (modified rank-based)

```
#define POPSIZE 200
#define MAXGENS 1000
#define PXOVER 0.7
#define PMUTATION 0.1
#define N     4;              //Perm(N,beta)
#define BETA  50;             //Perm(N,beta)
generation  best   average
   758          0.968     1.029
   884          0.000     0.002
   885          0.000     0.001
the best result
x1 = 1.000 x2 = 2.000 x3 = 3.000 x4 = 4.000
```

(n,beta) = (4,0.5) (rank-based)

```
#define POPSIZE 500
#define MAXGENS 3000
#define PXOVER 0.7
#define PMUTATION 0.1
#define N     4;              //Perm(N,beta)
#define BETA  0.5;            //Perm(N,beta)
 generation   best      average
   1448       0.990     1.527
   2999       0.011     0.015
```

```
    3000        0.011       0.015
the best result
x1 = 0.991 x2 = 2.010 x3 = 3.005 x4 = 3.984

(n,beta) = (4,0.5)   (modified rank-based)
#define POPSIZE 500
#define MAXGENS 3000
 #define NVARS 4
#define PXOVER 0.7
#define PMUTATION 0.1
#define N     4;                //Perm(N,beta)
#define BETA  0.5;              //Perm(N,beta)
 generation    best         average
    1160         0.995       1.469
    2999        0.011       0.014
    3000        0.011       0.014
the best result
x1 = 0.990 x2 = 1.999 x3 = 3.001 x4 = 4.009
```

Similar results are derived for another pair.

The result showed that for (n,beta) = (4,50) and (n,beta) = $(10,10^9)$, the modified algorithm performs better than the original rank-based selective function. Though they both find the optimal value, the modified algorithm converges faster than the original rand-based selective function. For (n,beta) =(4, 0.5) and (n,beta) = $(10,10^7)$, even the improvement is not significant, the modified algorithm still performs better than the original one.

5. Conclusions

In this paper, a modified selective function is proposed. The modifications are to curve the selective pressure according to their relative fitness and chromosome distribution. Some experiments are conducted on one benchmark function; the result shows that the modified selective function performs better than the original rank-based selective function in some cases.

References

[1] Zbigniew Michalewcz Genetic Algorithms + Data Structures = Evolution Program Springer, 1996.

[2] Whitley, D., The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[3] Baker, J. E. Adaptive selection methods in genetic algorithms. Proceedings of the First International Conference on Genetic algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[4] Kumara Sastry, David E. Goldberg Modeling Tournament Selection With Replacement Using Apparent Added Noise. Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[5] A Comparison of Selection Schemes used in Genetic algorithms, Blickle, T., Thiele, L., TIK_Report Nr. 11, December 1995, Version

[6] Kumara Sastry, David E. Goldberg Modeling Tournament Selection With Replacement Using Apparent Added Noise. Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[7] Baker, J. E. Adaptive selection methods in genetic algorithms. Proceedings of the First International Conference on Genetic algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985

[8] http://www.mat.univie.ac.at/~neum/glopt/my_problems.html Hard optimizations

[9] L.C.W. Dixon and G.P. Szego, The global optimization problem: an introduction Towards Global Optimisation 2, North-Holland, Amsterdam 1978.

[10] Whitley, D., Genetic Algorithms Toturial.

[11] Goldberg, D.E, Genetic Algorithms in Search, Optimazaiton and machine learning, Addison-Wesley, Reading, MA, 1989

[12] Baker, Reducing Bias and Inefficiency in the selection algorithms. Proceedings of the first International conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.