THE INFLUENCE OF POPULATION SIZE AND SEARCH TIME LIMIT ON GENETIC ALGORITHM

Yuri R. Tsoy
Department of Computer Engineering,
Tomsk Polytechnic University,
84, Sovetskaya street,
Tomsk, 634034, Russia,
Tel: +7 3822 418912, Fax: +7 3822 419149,
E-mail: qai@mail.ru, qai@sibmail.com

Abstract
Genetic algorithms use numerous parameters in their work. Most researches investigated different selection strategies and genetic operators and their influence on schema dynamics, but only few of them tried to understand deeply influence of population size and generations number limit. In this paper we will try to take a deeper look on the problem of the population size and limit of generations given for the search of the solution. This research involves investigation of GA's behavior on the different types of functions. These include functions with many optimums, functions with limits in search space (non-linear programming tasks) and NP-problem functions. Author is going to show that population size increase improves the performance of GA and affects results more than change of generations number limit.

Keywords: genetic algorithm, population size, generations limit

1. Introduction

By now following parameters of genetic algorithm (GA) been investigated thoroughly: selection strategies, different variants of genetic operators, encoding problems, various models of GAs but problems of population sizing and generations limit, needed to find a solution, have not been given much attention. Partially it can be explained by a stochastic nature of genetic algorithm and absence of methods that allow predicting when and on what conditions algorithm having definite parameters will convergence. We can only say more or less confidently that the solution will be found or, vice versa, we are likely to fail.

The problem of proper population size selection can be compared figuratively with a problem of choice human group size for decision making. If group numbers is small then complex problem solution is hard to find. From the other side, in large group, when everybody is standing upon his point of view, we can expect, that on initial phases results will be worse because of discords and disputes, but final solution will be more successful outcome.

Genetic algorithms are successfully implied for optimization tasks with large number of parameters and hardly formalizable problems. There is a de-facto standard for the evaluation of the algorithm such as a number of target function evaluations. The higher bound of this characteristic can be calculated via following equation:

$$FE = G \cdot N, \tag{1}$$

where N – population size, G – generations number limit, used for the search of solution. From one hand, the more population size, the better solution found, but from the other hand, increase of the number of individuals in population can lead to the significant growth of target function evaluations.

This in its turn will influence on the overall performance rating of genetic algorithm. A question appears what is better, large population and small generations number or, vice versa, reduced population size accompanied by larger number of generations, that we need to find a solution. Proper choice of population size and generations number would allow us to analyze GA more precisely in general terms and avoid unnecessary computations.

3. Earlier publications

There were some earlier publications by other researchers, who investigated problems close to claimed. In [1] an empirical formula derived for the computation of population size for the OneMax problem solution using such data as number of loci n, initial probability to meet desirable allele p_0 and selection intensity I. The formula gained looks as follows:

$$N^* = 1 + (10.28 - 12.07I + 7.30I^2)\sqrt{n} \cdot \ln n \cdot (\frac{1}{\sqrt{p_0}} - 1),\ 0.8 < I < 1.4\ . \tag{2}$$

Also in [1] a formula for the calculation of the generations, needed for the convergence of population, was derived:

$$GEN_c \approx (\frac{\pi}{2} - \arcsin(2p_0 - 1)) \cdot \frac{\sqrt{n}}{I}\ . \tag{3}$$

Using results gained in [1], Cvetković and Mühlenberg drew a conclusion that number of individuals, needed to find a solution, is much larger for proportionate selection strategy, then that of for the truncation selection. It should be noted that that work does not take into consideration mutation rate and algorithm used only one-point crossover.

In publication [2] Spears and De Jong investigated joint influence of crossover type and population size on the performance of the genetic algorithm. They used an analysis of strings disruption by crossover and its interaction with population sizing. Following result was gained: in small populations more disruptive crossover (uniform and n-point crossover operators) is more effective, and use of less disruptive crossover (1- and 2-point crossover) in large populations lead to better performance. They also showed that this tendency does not have a single meaning and depends on complexity of the task.

The work [3] is very likely to be one of the first studies where question about sizing of the population was raised. The notion of genetic drift (genetic noise) was introduced there also. This is an effect based on stochastic property of the algorithm. Consider having population of single digit binary strings, the first half of them is "1", and the second one is "0". If we will choose strings by chance for the creation of new generation, we can expect getting equal quantity of different strings. In real as generations passed we will observe increase of heterogeneity, which can finally lead to the disappearance of the definite type strings from population. This phenomenon of loss of strings and their parts was called allele loss. In his work De Jong drew a conclusion that increase of the population size can reduce not only allele loss, but also genetic drift significantly. In [3] also shown that in large populations fitness grows more slowly on initial phases but overall result will be better.

In [4] equation for population sizing of simple genetic algorithm is derived based on the statistical decision making. The choice between two competing building blocks with different fitness and deviation is investigated. That Goldberg, Deb and Clark show that better results can be achieved on larger populations. An assumption about polynomial convergence time (quadratic or cubic) was also made.

On the base of analyzed publications following conclusions can be gained:

1. Increase of the population size improves performance of the genetic algorithm since it reduces genetic drift, allele loss and increases parallelism of the algorithm.
2. The role of generations limit and its dependency on the population size remains not clear.

4. Two rates

Let's have a look what happens when genetic algorithm works. We are given some search space where global and a set of local optimums are located. We also have a number of points in this space such that each point is an individual in population. Fitness of each individual is proportional to its distance to the optimums so that distance to the global optimum has a priority. We should recombine coordinates of individuals from generation to generation to get the point that is nearest to the global optimum. Recombination should be mostly applied to the fitted individuals.

Initially individuals distributed chaotically without any order and tendency. As some generations passed we can see quite a different picture. Individuals represented by points in search space grouped near optimums and only few of them "strayed from the herd". These are can be victims of mutation or unsuccessful recombination of coordinates of better fitted parents. Also as population gathers near optimums it's becoming more uniform in terms of coordinates values. So we have a picture that is similar to the starting one: we have search spaces, optimums and points representing individuals in this space. Everything is repeating but with one exception – population is more uniform now and values of coordinates can not provide us much heterogeneity in search via recombination as it was in the beginning. But it still can be enough to perform well in reduced search space and to reach the global optimum or some near point.

Thus two rates can be singled out:

1. the rate of search space reduction that shows how fast algorithm processes search space and locates areas with optimums;
2. the rate of population convergence that takes place because of allele loss and sometimes because of information encoding method (Standard Binary encoding or Grey codes).

A suggestion can be made that genetic algorithm will perform well and achieve good results if rate of population convergence less than rate of search space reduction. We should note that these notions can't be compared directly and should be estimated in terms of generations number needed to solution be located with allowed error or more than most of population contents become homogenous. If suggested condition is not fulfilled than search of global optimum depends on luck and good chance.

Let's see how population size and generations limit can be estimated using foregoing reasoning. If we have larger population the rate of population convergence will be less than that of in small-size population because of initially increased variety of individuals. This statement is supposed to be correct for the average fitness of population. From the other hand we can expect that search space reduction rate is also reduced but in real it is a fact only in the first generations. Later as more high-fitted points have been arrived (when their number will be greater some "threshold") other individuals will gather to these "leaders" quickly. In small populations rate of convergence is higher with recombination abilities not better then that of in large populations, assuming usage of standard crossover operators such as 1-, 2-, n-point and uniform crossover. So greater generations number is not needed since population becomes homogenous quickly. In case of large population too strict limit for the search time can force algorithm to stop without having enough time to realize its search possibilities.

One more suggestion is that population size has some "critical" value after which increase of population size will not improve search time (number of generations) and results. It would be logical to suggest that optimal population size less than this "critical" value.

Also note that high mutation rates can help avoid premature convergence in small populations but this method can also lead to the significant expansion of search space.

5. Experiments

In experiments canonical GA with 1-point crossover and without mutation was used. Mutation operator was excluded to avoid additional stochasticity. Two different types of experiments took place. In the first one the number of target function evaluations was limited by 51200 and population size was as a variable parameter together with variable generations' number limit. In the second type experiment number of generations was limited by 200 but population size value was also variable and similar to that of in the first type experiment.

Let's denote combination of population size and generations limit as NxG. For example, 512x100 means population of 512 individuals that evolve for 100 generations. Taking into considerations foregoing run conditions the following parameters was set to examine GA's behavior (for OneMax-problem minimal population size was 32 due to specific initialization):

1. First type experiment: 8x6400, 16x3200, 32x1600, 64x800, 128x400, 256x200, 512x100, 1024x50, 2048x25, 5120x10, 10240x5, 12800x4, 25600x2 – individuals x generations' number.

2. Second type experiment (200 generations limit for all cases): 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 5120, 10240, 12800, 25600 – individuals in population.

Following functions were taken to test genetic algorithm behavior in different conditions:

1. 32-bits OneMax problem. The task is to construct binary string of 32-digits from given population with "1" in each position. The population was initialized the way that each individual housed only one "1" and distribution of "1" in each position over the whole population was uniform.

2. 3-peak SAT-problem suggested by W. Spears [5]. This is a boolean satisfactory NP-complete problem and looks as follows:
$(a1\wedge\ldots\wedge a30)\vee(a1\wedge\overline{a1}\wedge\ldots\wedge\overline{a30})\vee(a1\wedge\overline{a1}\wedge\ldots\wedge\overline{a15}\wedge a16\wedge\ldots\wedge a30)$.

3. 50-variables Rastrigin's function. The formula for this function (minimization task):

$$F(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)),\ \ x\in(-5.12;\ 5.12),\ \ n=50. \tag{4}$$

4. Non-linear programming task with 2-variables and non-convex limitations. Target function (maximization task):

$$F(x_1,x_2)=x_1^2+x_2^2; \tag{5}$$

the set of limitations:

$$\begin{cases} x_2-x_1^2-1\le 0,\\ x_2-\sqrt{x_1^2}\ge 0,\\ 0\le x_1\le 2.047,\\ 0\le x_2\le 2.047. \end{cases} \tag{6}$$

6. Results and discussion

The results of runs are illustrated in the figures 1-4. Each figure contains two plots: one represents case when search was limited by number of target function evaluations ("FE Limit") and another one shows what happens if number of generations for search is the only restricting parameter ("200 Generations Limit"). Numbers near knot points on the plots indicate minimal generations' number gained to achieve corresponding results. For example, in figure 1 when population of 1024 individuals was used, algorithm needed 49 generations to achieve result of 31 "1"-digits in chromosome string.

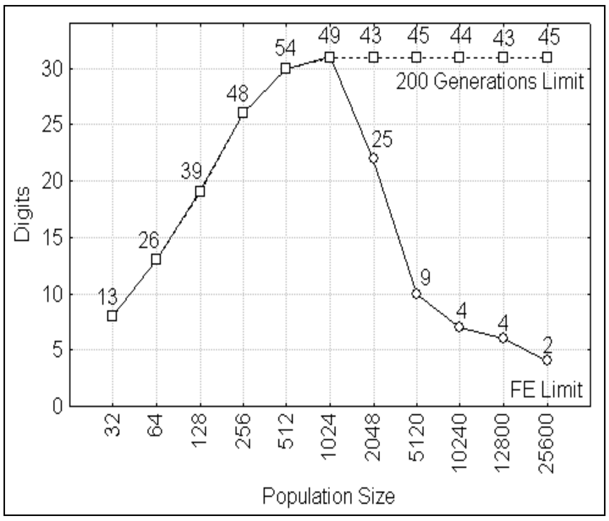

Fig.1. Results for OneMax32 task

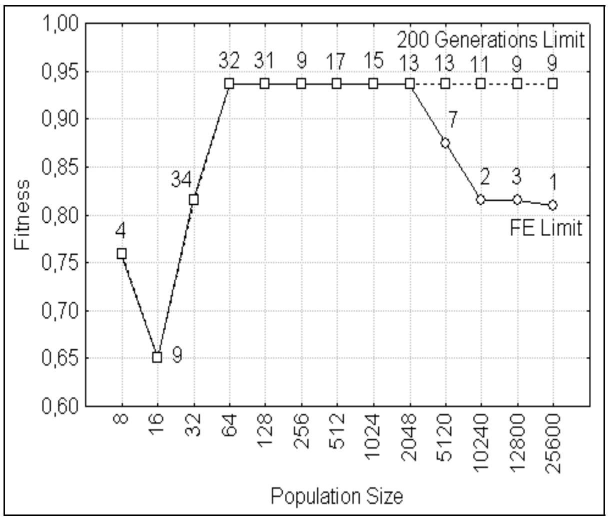

Fig.2. Results for 3-peak SAT-problem

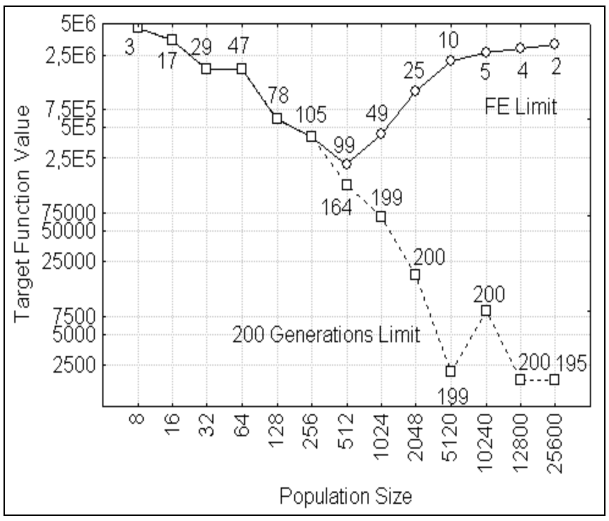

Fig.3. Results for Rastrigin's function

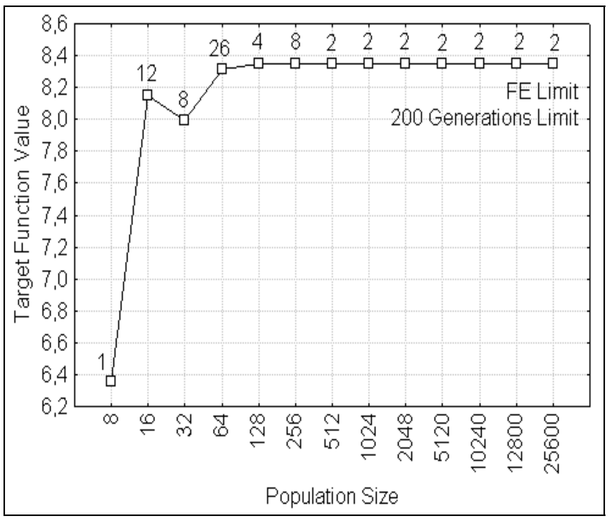

Fig.4. Results for non-linear programming task

A general tendency can be observed in all figures if move towards the increase of population size. At first, a lack of individuals leads to the premature convergence and number of generations needed to find the best solution using corresponding population size is less than limiting generations' number. It doesn't mean that after finding the best individual search stops. In fact algorithm by itself doesn't even "know" about new champion discovered and simply continues its work. As numbers of population grows results are improved. In case when there is a target function evaluations limit a lack of generations prevents algorithm from further search.

In terms of foregoing rate of reduction of search space and a rate of population convergence we can see firstly greater rate of convergence and then, when population is large enough, greater search space reduction rate.

Also a following tendency in change of generations' number enough to locate some good solution can be singled-out. Firstly, this number increases as population and fitness grows (function minimizes for Rastrigin's function). Then after some "critical" point passed it reduces till boundary value achieved and fitness of the best solution doesn't change. This can be seen in figures 1, 2 and 4. It should be noted that for the OneMax-problem value of generations' number after "critical" point is approximately 44 although minimum possible is 5. This is the illustration to the influence of the genetic drift on the results. Population for the first task was initialized such a way that genetic drift had an advantage. It also causes the fact that target string of 32 "1"-digits wasn't constructed by the algorithm (the best result is 31 "1"-s string). In case of Rastrigin's function algorithm performed not very well due to complex surface with many minima of this function that's why algorithm continued search process with increase of population size. The best result gained for 25600x200 combination is 1772.46 while global function minimum value is 0.

In general increase of population size tends to give better solution even if generations' limit becomes tighter. Thus if we are given some number of target function evaluations tuning of the population size can help achieve good results without excess expenditure of computational resources. But it can be seen that although population size increase improves performance of GA the cost is also can be high. The number of function evaluations increases dramatically: to reduce number of generations for 3-peak SAT-problem from 32 to 15, population size grows for 16 times from 64 to 1024.

The problem of choice of the appropriate population size and calculation of generations' number enough to gain satisfactory result can be solved only after mathematical model that describes dynamical behavior of GA developed. A suggestion can be made that there is no precise formula that gives one and the only good number for best population size and generations limit, because there too many variants and combinations of different genetic operators, selection strategies, encoding types and so on to take into consideration all of them.

6. Conclusions and further work

Results achieved in this work show that in most cases larger populations with less number of generations are better than small populations accompanied by greater time for search.

The size of population is one of the significant parameters of genetic algorithm since it has direct influence on its search abilities. In this work an attempt was undertaken to show that using one and the same number of function evaluations the case when population has larger size is better in terms of results gained. But excess of individuals can be bad for the computational difficulty dependent to the implemented algorithms because some selection strategies use ordering or ranking of population according to the fitness of its individuals. A question about excess of generations also arises. After some generation population does not evolve or does evolve but very slowly and insignificantly, when 50 and more generations needed to reach a small improve in result. In this case increase of population size have a sense since it brings variety and makes search process more active.

A difficulty in research is that there is no analytical formula for search space reduction and for population convergence. The main cause is a great variety of GA implementations and necessity to take into account numerous parameters that have complex influence on the whole GA performance. This problem needs further investigation. Probably creation of formulas for reduction of search space and for rate of population convergence could give us the sought for mathematical model of genetic algorithm.

References

[1] Cvetković, D., Mühlenbein, H., 1994, The optimal population size for uniform crossover and truncation selection (Technical report 94-11, GMD).

[2] De Jong, K. A., Spears, W. M., 1990, Int'l Workshop Parallel Problem Solving from Nature, 38-47.

[3] De Jong, K. A., 1975, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral Thesis, (USA: Department of Computer and Communication Sciences, University of Michigan, Ann Arbor).

[4] Goldberg, D. E., Deb, K., and Clark, J. H., 1991, Genetic Algorithms, Noise and Sizing of Populations, (University of Illinois).

[5] Spears, W. M., 1990. Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems, Masters Thesis (Department of Computer Science, George Mason University, Fairfax, Virginia.).

[6] Whitley, D., 1994, Statistics and Computing, No. 4, 65-85.

Biography: Yuri R. Tsoy was born in 1981. In 1998 he entered the Department of Computer Engineering, Tomsk Polytechnic University. In 2002 received degree of Bachelor of Techniques and Technologies. Now he is a 1-st year magistrant at the Department of Computer Engineering. Research interests: genetic algorithms, artificial neural networks, multi-agent intellectual systems, artificial life, psychology and philosophy of cognitive processes.