Self-programming of Algorithms

Piotr Wąsiewicz

Institute of Electronic Fundamentals, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
E-mail: pwas@ipe.pw.edu.pl
phone: (04822)-660-5319, fax: (04822)-252300

Abstract

In this paper self-programming of algorithms is considered. Genetic Algorithms are developed under control of Genetic Programming. It is approved that this methodology leads to finding the best structure of optimization algorithm, which searches the global optimum in multidimensional functions. The proposed approach has been implemented in C++ on a Pentium 90. Many experiments have been conducted. Some of them are presented in this paper. The results show that this new method allows to find better structures of Genetic Algorithms than standard ones or even those of Evolution Strategies.

Keywords: Genetic Programming, Genetic Algorithm, Global Optimization

1. Initial Statements

Genetic Programming (GP) [1] has been recently developed as one of Evolutionary Algorithms. Their earlier implementations are called: Genetic Algorithms (GA) and Evolution Strategies (ES). These efficient heuristics are based on evolutionary methods derived from Nature e.g. DNA genetic chromosomes, theory of evolution. Their standard structures are well known. Their search imitate Darwinian strife for survival. They start with randomly generated initial population. Each individual of this set represents a possible solution to the problem. Then, while terminate condition is not true, the following cycle is performed: number of generation ++; select Population(t) from Population(t-1); recombine Population(t), using crossover and mutation operators; evaluate Population(t). Terminate condition means that solution is good enough or there is no better results of computation for a period of time or a maximum number of cycles (epochs, generations) is reached.

GP is a methodology to solve problems by genetically breeding populations of computer programs. For a particular problem sets of functions and terminals are to be created. An initial population of LISP-like expressions is a collection of random tree-like compositions of fundamental functions and terminals. Each expression called also a program is evaluated against the problem. Genetic operators of selection and crossover are applied to create new populations of programs. Evolutionary process is continued until either a solution is found or a maximum number of generations is reached.

GAs [2] belong to techniques that can be successfully applied to NP-hard optimization problems. Generally, GAs are good at optimizing functions with many local and one global optima. It is assumed that GAs search for the global optimum [3] of such functions with cube constrains (l - a left constrain, r - a right constrain).

This problem can be described as

$$\min\left(f(X)\right)$$
$$X = \left(x_1, x_2, \ldots, x_N\right)$$
$$l_1 \le x_1 \le r_1, \ldots, l_N \le x_N \le r_N$$

In the above notations each coordinate of a point is restricted to a given interval. A point as a sequence of these variables from one to N is binary coded and called „a chromosome" e.g. there is a point coordinate x within an interval *[l, r]* and this interval is transformed into a machine word interval *[0, 2^N]* and a new binary representation c of x is received with a value *x*2^N/(l-r)*. Each coded variable has M genes, where M is a length of a machine word. A single gene can take values 1 or 0. After this operation each variable is Gray coded. In this way feasible points in the search space are transformed into such binary strings. Here a value of the given function in a given point is a fitness value of that point (individual, chromosome).

In the process of evolution, a number of cycles, standard genetic operators such as mutation, crossover are applied. The GA finds the individual with the best feasible solution (the value of a function near global optimum) located in the defined space.

ESs were first mentioned in 1964 [5]. Then, the idea to imitate principles of organic evolution was introduced in the field of experimental parameter optimization in Germany. The applications managed to deal with hydrodynamical problems like shape optimization of a bent pipe, with control problems like the optimization of a PID regulator within a highly nonlinear system and with numerical optimization problems. ES are similar to GA, but have different types of selection and do not use binary coding of individuals (points) in population. They operate in a continuous space of point coordinates[6].

In this paper, a technique based on GP controlling structures of GAs is introduced. It is difficult to find the best receipt for constructing GAs, since there are many sequences of genetic operators. A way of optimizing structures of GAs with help of GP to search quickly for global optima is presented. Efficiency comparison of developed algorithms with known methods is provided.

2. Genetic Programming Approach

During execution of GA it is important which genetic operations (e.g. crossover, mutation e.t.c.) are first performed. With the help of GP, trees made of genetic operations are created. Thus, *these tree-like structures describe an execution way of GA during one generation (epoch). After each GA epoch the best (the nearest to global optimum) chromosome is remembered. After a given number of GA epochs the best of all these chromosomes is chosen.* A value of the given function in a point obtained from the chosen decoded chromosom is a fitness value of this used earlier tree-like GA. Genetic operators of GP e.g. selection and crossover create new populations of algorithms in order to find the most appropriate GA.. The evolutionary process of GP is continued until terminate condition is true. After this the best (for searching global optimum) solution - structure of GA is showed.

A set *F* of independent operations on population and one terminal, which is just a probability of the particular operation are developed. The terminal set is ***T* = {*x*}.** The function set is following ***F* =**
{SEL, SUS, SEI, SET, SPT, BST, MUT, MT1, MT2, CRS, DCR, PCR, SCR, UCR, INV, +, -, *, /, =, <}.
The functions have the following meaning:
(SEL x) -- proportional selection, the method of „the roulette wheel",
(SUS x) -- stochastic selection with sample probing[2],a value of chromosom fitness / an average of fitnesses aims to a offspring number.
(SEI x) -- selection: in the next population exists only copies of the best individual from the previous population,
(SET x) -- tournament selection: from two at random chosen chromosomes the best wins,
(SPT x) -- tournament selection with probability: from two at random chosen chromosomes the best wins with given probability,
(BST x) -- *(1 - x)* best strings of population is remembered and will be copied after all operations-arguments of BST execution,
(MUT x) -- mutation with probability *x/100,*
(MT1 x) -- mutation with probability *x/10,*
(MT2 x) -- mutation with probability *x,*
(CRS x) -- crossover with probability *x,*
(DCR x) -- double crossover with probability *x,*
(PCR x) -- fivefold crossover with probability *x,*
(SCR x) -- „sweep" crossover with probability *x* [2],
(UCR x) -- „uniform" crossover with probability *x,*
(INV x) -- inversion of bits between two points with probability *x,*
(+ x) -- addition to probability a predefined constant *st1 : x + st1,*
(- x) -- subtraction from probability a predefined constant *st1 : x - st1,*
(* x) -- multiplication of probability by a predefined constant *st : x * st,*
(/ x) -- division of probability by a predefined constant *st : x / st,*
(= (CRS x) (MUT x)) -- predefined sequence of two GAs operators executed on the whole population, e.g. mutation is executed first, crossover - second,
(< (CRS x) (MUT x)) -- predefined sequence of two GAs operators executed on the divided population, e.g. on the first half of the population mutation is executed, on the second half of the population crossover is executed.

As an example consider a tree ***(= (CRS (MUT (- x))) (< (MT1 x)(UCR (/ x))))*** . As is seen operators of a typical tree are executed from right to left. Probability x comes from right operation to left operation.
It can be written in a file form as:
= c CRS c MUT c - c x o o o o n < c MT1 c x o o n UCR c / c x o o o o o o

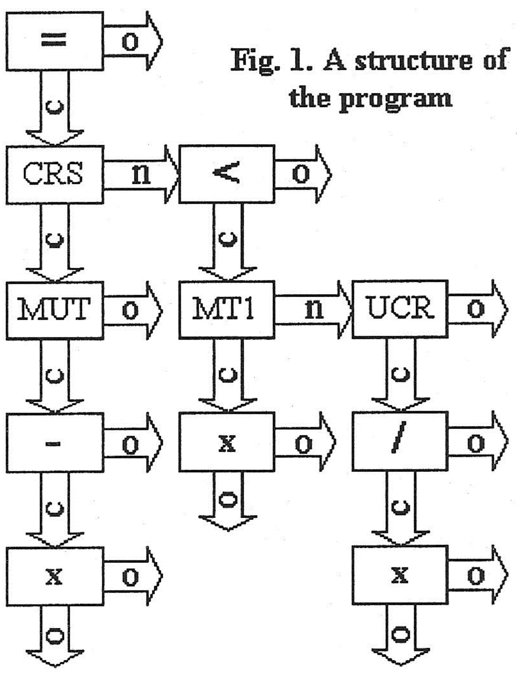

Fig. 1. A structure of the program

In *Fig. 1* a physical structure of this tree is provided. Here *"c"* means *"child"* (a first argument of a genetic operator of GP), *"n"* means *"next"* (a second argument of a genetic operator of GP) and *"o" - NULL* (without an argument)[4].

3.Results of experiments and discussions

During many experiments using this new methodology (described above) arrays of optimization results with different structures of GAs were provided. Comparisons with a program using evolution strategies by Karsten Trint and Uwe Utecht from Berlin and with a Controlled Random Search method implemented at TUW Warsaw were done. All experiments have been implemented in C++ in a DOS enviroment on the Pentium 90 computer. Descriptions of test functions, GA and GP parameters are necessary to perform computation. There were values of the algorithm parameters selected as follows. Predefined constant *st* is equal to 2. Predefined constant *st1* is equal to 0.05. At the beginning *x* is equal to 0.9. In the case of calculated probability greater than 1 or less than 0 this probability will be again equal 0.9. Five test functions were defined in the following way:

1. Rosenbrock function (10 dimensions)

RosenbrockN

$$f(X)=\sum_{i=1}^{N-1}\left[100\bullet\left(x_{1+i}-x_i^2\right)^2+\left(1-x_i\right)^2\right]$$

$$\overset{*}{x}=(1,1,1,1,\ldots)$$

$$f\left(\overset{*}{x}\right)=0$$

2. Shubert function (2 dimensions)

Shubert

$$f(X)=\sum_{i=1}^{5}i\cdot\cos\left[(i+1)\cdot x_1+i\right]\cdot\sum_{i=1}^{5}i\cdot\cos\left[(i+1)\cdot x_2+i\right]$$

$$-10\le x_i\le 10$$

$$i=1,2$$

Shubert function has 760 local minima, among them 18 global *f=-186.73.*

3. Goldstein-Price function (2 dimensions)

$$f(X)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\cdot$$

$$\cdot\left[30+\left(2x_1-3x_2\right)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$$

$$f_{\min}(0,-1)=3,$$

$$-2\le x_1\le 2,$$

4. Rastrigin function (20 dimensions)

Rastrigin

$$f(X)=nA+\sum_{i=1}^{n}\left(x_i^2-A\cos(\omega x_i)\right)$$

$$n=20, A=10, \omega=2\pi, -5.12\le x_i\le 5.12$$

$$f(0,0,0,..,0)=0$$

5. Wood function (4 dimensions)

Wood

$$f(X)=100\left(x_2-x_1^2\right)^2+\left(1-x_1\right)^2+90\left(x_4-x_3^2\right)^3+$$

$$+\left(1-x_3\right)^2+10.1\left(\left(x_2-1\right)^2+\left(x_4-1\right)^2\right)+19.8\left(x_2-1\right)\left(x_4-1\right)$$

$$f_{\min}(1,1,1,1)=0,$$

$$-3\le x_1\le 3,$$

Table 1. The most efficient GA structures found for given functions

No	The best AG for the function:	The best algorithms found for a given function. Size of GP population equal to 80. Number of GP epochs - 5. Size of GA population - 20. Number of GA epochs - 40. Step scale of fitness values for selections SEL and SUS.
1	Rosenbrock	((MT1 (SEI (SEI (CRS (BST (DCR (MT1 (SEI (SEI (MT1 (SEI (SEI (SEI (MT1 (SEI (SEI (DCR (SEI (/ x)
2	Rosenbrock	((SEI (MUT (SEI (MUT (MUT (SEI (MUT (MUT (SEI (MUT (SEI (SET (SEI (MUT (SEI (MUT (BST x)))))))))))))))))))
3	Shubert	((* (UCR (SEI (MUT (CRS x)))))))
4	Shubert	((SEI (CRS (INV (* x)))))
5	Gold.-Price	((SEI (INV (SEI x))))
6	Gold.-Price	((SEI (MUT (INV (< (PCR (PCR (< (SPT (SPT (SET (SEI (UCR (MT1 x))))))) (BST (SEI (INV (- (- (UCR x))))))))) (SEL (DCR (SCR (INV (SUS x)))))))))
7	Rastrigin	((SEI (MUT x)))
8	Rastrigin	((PCR (PCR (- (MUT (SEI (MUT (SEL (SUS (- x)))))))))))
9	Wood	((BST (+ (MT1 (= (INV (MT1 (/ (SEI (SEI x)))))) (MUT (SCR (* (CRS (SEI (+ (SEI (INV (MUT (SEI x))))))))))))))))
10	Wood	((DCR (CRS (* (SUS (MT1 (UCR (SCR (SEL (= (UCR (SCR (SEL (= (SEI x) (BST x))))) (- x)))))))))))
11	Wood	((UCR (MT1 (SCR (DCR (SEI (CRS (= x x))))))))

Table 2. The array of five optimization trials for each given function and comparison of optimization methods.

No of trial	1	2	3	4	5
The function of Rosenbrock 10					
AG -8 (20;300)	0.561645	2.22413	0.0005542	0.195918	0.99661
Cost	12060	12060	11980	11860	12020
ES (10;2000)	9.32e-23	3.04e-10	9.717e-06	3.986579	5.994e-23
Cost	20000	19970	20010	20010	20000
CRSP	4.86	2.79	3.32	3.29	3.21
Cost	24357	24834	24791	25618	23903
The function of Shubert					
AG -2 (20;300)	-186.731	-186.731	-186.731	-186.731	-186.731
Cost	3287	4981	21437	4134	4860
ES (100;1000)	-186.71	-186.3883	-186.6387	-186.5988	-186.48449
Cost	49200	4300	33600	94500	42000
CRSP	-186.7179	-186.1817	-186.73	-186.6319	-186.73
Cost	693	617	1173	656	851
The function of Rastrigin					
AG -2 (20;300)	9.78499	16.8861	15.5692	8.63879	18.231
Cost	35352	33295	18775	36320	24704
ES (20;1000)	70.64175	90.54099	44.77309	102.4803	81.58623
Cost	15920	16860	17080	14920	16500
CRSP	5.125384	5.13			
Cost	1576186	1023764			
The function of Goldstein-Price					
AG-9 (20;300)	3	3	3	3	3
Cost	16978	9231	16978	15087	13135
ES (20;1000)	3	3	3	3	3.000037
Cost	19960	20000	19360	18200	19580
CRSP	3	3	3	3	3
Cost	819	829	599	777	734
The function of Wood					
AG-11 (20;300)	0.0208407	0.18451	0.204003	1.37041	0.479571
Cost	3576	10394	4510	8184	6722
ES (20;1000)	4.275195e-12	1.732263e-11	2.235166e-11	3.482232e-09	1.115939e-13
Cost	19900	19460	19880	20000	19720
CRSP	1e-6	8e-7	1e-6	1e-6	1e-6
Cost	3515	3827	3724	3542	3290

The most efficient (only for this computation) GA structures for each given function have been automaticly generated with the help of GP. The results are summarized in *Table 1* e.g. a tree no 3 : ((* (UCR (SEI (MUT (CRS x)))))) can be easily translated into a GA structure. First, probability is assigned to 0.9. Second operation CRS is executed on the whole population with probability 0.9. Third operation MUT is executed on the whole population with probability 0.009. Fourth operation SEI is executed on the whole population with probability 0.9. Now population consists of the same copies of the best individual from previous population according to this genetic operation SEI. Fifth operation UCR is executed on the whole population with probability 0.9. At last after an operation „*" probability (greater than 1) is again assigned to 0.9. As is seen these structures are different from each other. Similarities appear rarely and emerge rather from using the same genetic functions.

Each GA structure from *Table 1* has been used to optimize each test function. The most appropriate algorithm is chosen for each given function and comparisons with ES and Controlled Random Search Procedure are provided in *Table 2*. The last method was invented by Price[3] and combines the random-search and mode-seeking (attached with the statistically estimated density function) algorithm into a single, continuous process. In this table the following notations are used. *AG-2 (20;300)* means the best AG no 2 with population of 20 points and 300 epochs. *ES (10;2000)* means Evolution Strategie ES-(1,10), which lasts 2000 epochs. *CRSP* means the Controlled Random Search method. *Cost is equal to a number of evaluations of the function f(x).*

As is seen numbers of chosen GA structures in *Table 2* are often different from these ones in *Table 1*. This means some algorithms found during optimization of one function have better performance with another function. There is an explanation: during optimization of sophisticated functions with many local and one global optima generated algorithms are often more robust and are good at searching a global optimum for many other functions. *GA structure no 2* found with a *Rosenbrock function* is the best (among GAs from *Table 1*) with *Rastrigin and Shubert functions* (the most sophisticated). For the Rosenbrock function with ten dimensions the better method is AG no 8 attached to the Rastrigin function which has 20 dimensions.

During comparison with other methods it has been observed that CRSP manage to find a global optimum of less complex functions e.g. Goldstein-Price, Wood or even Shubert functions. ES don't also find a global optimum of more complex multidimensional functions or these ones with many optima e.g. during searching optima of Shubert function ES had 5 times more points (individuals) in its population and over 3 times more epochs than GA and in spite of this it failed many times.

4. Conclusions

In this paper results of many experiments with GP improving GAs, which are applied to search a global optimum, have been presented. As follows from our considerations GP tries to choose the best sequence of genetic operators in tree forms. Now GA is not attached to only one standard scheme.

From the comparison of three methods **(Table 2.)** you can see that generated GAs are better than standard GAs and other procedures in optimizing *complex multidimensional functions with many local and one global optima* e.g. Shubert and Rastrigin functions.

Futher improvements in this field are expected. It was shown that only by changing the structure of GA it is possible to obtain better algorithms. Thus, this fact proves usefulness of using GP, even in this case.

5. Acknowlegdements

This work was supported by the Polish State Committee for Scientific Research under Grant 8T11C 04611.

The simulated experiments have been conducted with use of an improved by an author package: Genetic Programming in C++ v. 0.40 written by A.P. Fraser, University of Salford.

6.References

[1]. J.R. Koza: Genetic Programming, MIT Press 1992.
[2]. D.E. Goldberg: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA, 1989.
[3]. A. Torn, A. Zilinskas: Global Optimization, Springer Verlag Berlin, 1989.
[4]. A. P. Fraser: An Introduction to Genetic Programming in C++ (Version 0.40), ftp.io.com.
[5]. I. Rechenberg: Evolutionsstrategie'94, fromman - holzbolg, 1994.
[6]. F. Hoffmeister; T. Back: Genetic Algorithms and Evolution Strategies : Similarities and Differences, Interne Berichte und Skripten der Universitat Dortmund No. SYS 1/92.