I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

EVOLUTIONARY STRATEGIES WITH ADAPTATION OF POPULATION SIZE.

Jacek Pokraśniewicz,
jacke@ipeneuro.ipe.pw.edu.pl

Warsaw University of Technology, Institute of Electronics Fundamentals
15/19 Nowowiejska St., 00-665 Warsaw, Poland
tel. (48-22) 253709, fax: (48-22) 252300

ABSTRACT

We introduce a novel approach to Evolutionary Strategies, in which the population size varies over time. This process is controlled by the stage of the optimisation and helps keep its exploration-exploitation balance. We use an average value of standard deviation of Gaussian distribution as an indicator of stage. The method leads to a reduction of the computational effort needed to achieve similar results by comparable methods. Moreover our algorithm is more robust than in the standard case.

INTRODUCTION

In the field of evolutionary algorithms there are 2 methods useful in global optimisation problems: genetic algorithms (GA) and evolutionary strategies (ES). They are successfully used especially in cases, when knowledge about the problem is scare and the problem is difficult. Although these two classes of algorithms have more similarities, they also differ significantly. GA mainly use binary coding, so they can be used to optimise functions with discrete domains (eg. integer numbers) but also to optimise functions dealing with real numbers. ES are used in problems that deal exclusively with real numbers. As we observed, they achieve better results than GA in the field of real numbers problems. Moreover they have interesting properties not implemented in GA - self adaptation properties.
In this paper we will concentrate on Evolutionary Strategies ES, and their self adaptation properties, which (in the authors' opinion) make them unique and most valuable among evolutionary algorithms.
ES were introduced by Schwefel and Rechenberg [4]. In ES a set of potential solutions is maintained. Because of inspiration from biological systems, each solution is called a chromosome, and the set of chromosomes is called a population. Each chromosome consists of two parts: co-ordinates of a point in a solution space (vector **x**), and parameters of strategy. Usually, the vector of standard deviations σ is used as the strategy parameters. Each σ_i defines the standard deviation of Gaussian distribution of x_i, which is used during a random change called mutation. There are two types of ES: $(\mu+\lambda)$-ES and (μ, λ)-ES. They differ in the selection step - in the case of $(\mu+\lambda)$-ES, the next generation is created by selecting μ individuals from μ parents and λ children, and in the case of (μ, λ)-ES only λ children are taken into account. Sketch of $(\mu+\lambda)$ Evolutionary Strategy is provided in Fig 1.

```
init(Pop(0))
t:= 0;
    while not termination_condition do
        begin
        Offs(t):= reproduce(Pop(t))
        mutate(Offs(t))
        Pop(t+1):= selectBest(Pop(t) $\cup$ Offs(t))
        t:= t+1
        end
```

Fig. 1. (μ+λ) Evolutionary Strategy scheme.

where:

init(Pop(0)) - random initialization of *Pop(0)*, *Pop(t)* - base population, *Offs(t)* - offspring, *reproduce(Pop(t))* - function that randomly chooses l offspring from m parents, *mutate(Offs(t))* stands for the mutation of the offspring, *selectBest(Pop(t)$\cup$Offs(t))* is the selection of the *m* best individuals from both the *m* parents and *l* offspring;

Mutation consists of the following steps:

1. mutate the values of s_i

   $\sigma_i := \sigma_i \cdot \exp\left(\tau_0 \cdot \mathrm{N}(0,1) + \tau \cdot \mathrm{N}_i(0,1)\right)$

   where N(0,1) denotes a random value normally distributed with zero mean and standard deviation equal to one; value of N(0,1) is sampled once for each chromosome and $\mathrm{N}_i(0,1)$ is sampled for each allele independently; τ_0 and τ are constant.
2. mutate the values of **x**

   $x_i := x_i + \sigma_i \cdot \mathrm{N}_i(0,1)$

Fig. 2. Mutation scheme.

Note that the standard deviations s_i undergo random changes and thus the designer of the algorithm does not have to choose explicitly the proper values. Moreover, the self-adaptation of values of s_i has been observed. It has been reported that values of s_i tend to decrease as the algorithm converges.

In this paper we propose to apply the self adaptation approach to the population size (both μ and λ). We shall focus on the optimisation of functions which do not change in time.

MOTIVATION

During optimisation there is a trade-off between exploration and exploitation. We assume that there exists dependence of the actual state of the optimisation and the optimal population size [3]. This dependence arises from the exploration-exploitation balance. At the beginning of the optimisation, population size should be high, since it increases the possibility to locate the neighbourhood of the global optimum. As the optimisation progresses, the most promising areas are located. At this stage, it becomes more important to exploit effectively these areas. This can be performed more effectively by choosing small values of μ. The balance between exploration and exploitation is called selective pressure (the higher selective pressure, the stronger exploitation). In the case of ES, selective pressure is adjusted by automatic change of standard deviations of the distribution used for the mutation. For the first population it seems efficient to set σ_i at high level (to allow better exploration). Simulation results show that as

optimisation goes on, values of σ_i tend (on average) to decrease. Such a property can be motivated theoretically.
In the approach we have assumed that in order to explore an entire domain it seems reasonable to use big population (to enlarge possibility of locating the neighbourhood of global optimum). However, in order to find local (or global) extreme effectively it is more efficient to use a small population to allow the search to concentrate on the neighbourhood of a single optimum (no need for any alternate solutions). Thus (1+λ)-ES can be viewed as a kind of stochastic gradient method and allows to avoid extra computational effort. Up to now, there was an assumption that μ should be constant and each ES designer had to make compromises and set an average value of μ to balance exploration and exploitation. On the contrary, we propose to allow the value of μ to vary over time: at the beginning it should be high, when the algorithm locates the neighbourhood of the extreme, the value of μ should become equal to 1.

THE ALGORITHM

The algorithm presented below, called ESOP, implements the above considerations. In order to provide „smart" changes of $\mu(t)$ we have to make its value depend on some measure of optimisation state. We believe that such convenient measure can be an average value of standard deviations over all chromosomes in the population $\underline{\sigma}$, which can be defined as:

$$\underline{\sigma} = \frac{1}{\mu(t) * n} \sum_{r=1}^{\mu(t)} \sum_{i=1}^{n} \sigma_i^r$$

where n is the dimensionality of the search space. Value of $\underline{\sigma}$ tends to decrease as optimisation progresses, and its decrease is most dramatic when almost all chromosomes belong to the neighbourhood of the same satisfactory extreme. (Because there is only slight chance to find better offspring chromosome far away from the parental one, and thus parental chromosomes with low values of σ_i produce better offspring which survive the selection). We introduce a function $f(\underline{\sigma})$ to calculate value of $\mu(t+1)$, and we define it as follows:

$$f(\underline{\sigma}) = \begin{cases} 1 & \text{if } \underline{\sigma} \le S\_MIN \\ A\_S \cdot \underline{\sigma} + B\_S & \text{if } S\_MIN < \underline{\sigma} \le S\_MAX \\ POP\_MAX & \text{if } \underline{\sigma} > S\_MAX \end{cases}$$

where:

$$A\_S = \frac{POP\_MAX - 1}{S\_MAX - S\_MIN}$$

$$B\_S = A\_S \cdot S\_MIN - 1$$

The parameters S_MIN, S_MAX, POP_MAX must be chosen arbitrarily. We will discuss their proper values below.

The whole algorithm can be written as follows:

```
init(Pop(0))
t:= 0;
for i:= 1 to N
    begin
    while not termination_condition do
        begin
        Offs(t):= reproduce(Pop(t))
        mutate(Offs(t))
        Aux(t):= selectBest(Pop(t) ∪Offs(t))
        calculate l(t+1) and m(t+1)
        if m(t+1) < m(t) then
            Pop(t+1):=selectBest(Aux(t))
        else
            begin
            l'=L/M * (m(t+1)-m(t))
            AuxOffs(t):=reproduce(Aux(t))
            Pop(t+1):=selectBest(Aux(t) ∪AuxOffs(t))
            end
        t:= t+1
        end
    IncreaseSigma(Pop(t), i)
    end.
```

Fig. 4. Sketch of the ESOP algorithm.

where: *Aux(t)* - auxiliary population needed to expand the base population, *AuxOffs(t)* - auxiliary offspring population used during the base population expansion step, in the step *calculate* $\lambda(t+1)$ *and* $\mu(t+1)$ new values of $\mu(t+1)$ and $\lambda(t+1)$ are calculated according to the *tanh* function, *IncreaseSigma(Pop(t), i)* is responsible for random increase of the values of σ_i.

Discussion of parameters.

There are several parameters in the presented algorithm, whose values must be arbitrarily chosen: S_MIN, S_MAX, POP_MAX, and λ/μ ratio. In the case of the standard evolutionary strategy there are only 3 parameters: μ, λ, POP_MAX to be set. In this chapter we will discuss the parameters and try to reduce their numbers.

The first thing is to set the proper values of S_MIN and S_MAX parameters. Their values help the f($\underline{\sigma}$) function get proper shape. The motivation of the function f($\underline{\sigma}$) was to help the strategy during the optimisation process to switch between efficient global and local optimisation. Thus the change of the number of chromosomes in the population should be done at a moment, when the neighbourhood of the global extreme has been located and the chromosomes in the population tend to group in a small area. At this moment value of $\underline{\sigma}$ decreases (the chromosomes that survive the selection, with good fitness function value, are located near parents). There is no need to have a large value of S_MAX-S_MIN, due to the positive feedback mechanism. If the process of decreasing the number of chromosomes in the population begins, it will go on because weaker chromosomes (such that might represent alternative solutions) will be deleted from the population, and only the best ones will survive. While the value of f($\underline{\sigma}$) decreases, it is observed that the value of $\underline{\sigma}$ decreases also and the value of average fitness function per population increases dramatically. The conclusion is that the values of parameters S_MIN and S_MAX should not vary much and should be set to such values of $\underline{\sigma}$, which indicate that chromosomes in the population tend to group near some extreme (hopefully global). Our experiments reveal that the efficiency of the algorithm does not change much for different small values of S_MIN and S_MAX. Good values of these parameters for all tested functions were S_MIN=0, S_MAX= 0.1. We believe that these values are good for any fitness function.

To discuss the λ/μ ratio and POP_MAX parameters let us introduce a parameter named *optimisation power OP*, which will be defined as $(1+\lambda/\mu)$*POP_MAX. When the optimisation process is performed by the ESOP algorithm, it can be observed, that some instances find a global extreme, while others do not. It can be also observed, that if the value of OP increases, the optimisation succeeds more often and for some values of OP it succeeds every time it is started. The experiments we performed reveal also, that the exact value of λ/μ ratio and POP_MAX does not induce considerable changes of the algorithm efficiency, if only the value of OP is kept unchanged.

This leads us to conclusions, that the only parameter that has to be set properly is the *optimisation power OP*. Evolutionary strategies have small chance of success if the *OP* value is too small. If this value is too great - efficiency of performing optimisation decreases (but still global extreme is found). The optimal value of *OP* depends strongly on fitness function and is very difficult to find.

We propose to set the λ/μ ratio to 1, in order to have a (1+1)-ES at the end of optimisation. We believe that such a strategy is the most efficient in local optimisation.

COMPARISON OF THE NOVEL APPROACH AND THE STANDARD STRATEGY.

In comparison experiments several test functions commonly used in algorithm comparison in the global optimisation field were used. These were: Goldstein-Price, Camelback and Branin RCOS function [7]. The definitions of test functions are provided in Fig. 4. The Goldstein-Price function has a global minimum value of $f_{min}=3$ at the point $(x_1, x_2)= (0, -1)$. The Camelback function has in the bounded region six local minima, with two global minima located at the points $(x_1, x_2)= (-0.0898, 0.7126)$ and $(0.0898, -0.7126)$, where $f_{min}=-1.0316$. The third function - the Branin RCOS function - has three global minima at the points $(x_1, x_2)= (-\pi, 12.275)$, $(\pi, 12.275)$ and $(9.42478, 2.475)$ with $f_{min}=0.397887$.

As evolutionary strategies are used by convention to solve maximisation problems, we used the negation of all test functions mentioned above. The modified Goldstein-Price function has a global maximum value of $f_{max}=-3$, the modified Camelback function has two global maxima, where $f_{max}=1.0316$, and the Branin RCOS function taken with negation - has three global maxima, where $f_{max}=-0.397887$.

$$f_{GP}(x_1,x_2) = \left(1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14 \cdot x_1 + 3 \cdot x_1^2 - 14 \cdot x_2 + 6 \cdot x_1 \cdot x_2 + 3 \cdot x_2^2)\right) \cdot$$
$$\cdot \left(30 + (2 \cdot x_1 - 3 \cdot x_2)^2 \cdot (18 - 32 \cdot x_1 + 12 \cdot x_1^2 + 48 \cdot x_2 - 36 \cdot x_1 \cdot x_2 + 27 \cdot x_2^2)\right),$$
$$\text{where } x_1, x_2 \in [-2, 2]$$

$$f_{CM}(x_1,x_2) = \left(4 - 2.1 \cdot x_1^2 + \frac{x_1^4}{3}\right) \cdot x_1^2 + x_1 \cdot x_2 + \left(-4 + 4 \cdot x_2^2\right) \cdot x_2^2,$$
$$\text{where } x_1 \in [-3,3], x_2 \in [-2,2]$$

$$f_{BR}(x_1,x_2) = a \cdot (x_2 - b \cdot x_1^2 + c \cdot x_1 - d)^2 + e \cdot (1 - f) \cdot \cos(x_1) + e,$$
$$\text{where:}$$
$$a = 1, b = \frac{5.1}{4 \cdot \pi^2}, c = \frac{5}{\pi}, d = 6, e = 10, f = \frac{1}{8 \cdot \pi}$$
$$\text{where } x_1 \in [-5,10], x_2 \in [0,15]$$

Fig. 5. Definitions of test functions.

For each setting of parameters we have carried out 40 independent runs with a randomly chosen initial population. In the tested algorithms we used two stop criteria. The algorithm terminates if one of them is achieved. The first criterion is based on the value of $\underline{\sigma}$. If $\underline{\sigma}$ decreases below some arbitrarily set threshold value it is assumed that no better extreme will probably be found and the algorithm terminates. We used threshold value equal to 10^{-6}. The second criterion is *patience*. It is defined as the number of generations in which average objective function value per population does not change. It is assumed, that if the average objective function value per population does not change through specified number of generations, it means that the algorithm could not find any better results.

The tables given below present results achieved by the ESOP strategy in comparison with those achieved by the standard $(\mu+\lambda)$ evolutionary strategy. The numbers in the table are the costs of strategy measured as the number of fitness function evaluationa at a time, when the average value of 40 independent runs of algorithm (with random initial population) is not less than the threshold value given by each table. These values were chosen to be 0.1% lower than the value of global maximum of the test function. The places marked '-' mean that the average value of 40 runs of the algorithm did not achieve threshold value at all. The threshold values are: 1.0306 for the Camelback function, -0.39828 for Branin function, and -3.003 for Goldstein-Price function. No crossover operator was used in the ESOP algorithm, nor in the case of the standard strategy.

	S_MIN =0,01	S_MIN =0,1
$\lambda/\mu=2$		
POP_MAX=20	**320**	520
POP_MAX=50	520	500
POP_MAX=100	900	800
$\lambda/\mu=5$		
POP_MAX=20	480	480
POP_MAX=50	820	840
POP_MAX=100	1460	1340
$\lambda/\mu=10$		
POP_MAX=20	641	580
POP_MAX=50	1180	1300
POP_MAX=100	2320	2161

$(\mu+\lambda)$	no of evaluations
(5+10)	-
(5+25)	**355**
(10+20)	430
(10+50)	-
(20+40)	-
(20+100)	1020

Table 1. Results for the Camelback function achieved by the ESOP algorithm (left) and the standard $(\mu+\lambda)$-ES.

	S_MIN =0,01	S_MIN =0,1
$\lambda=2$		
POP_MAX=20	620	-
POP_MAX=50	660	720
POP_MAX=100	1000	1000
$\lambda=5$		
POP_MAX=20	**600**	680
POP_MAX=50	1020	820
POP_MAX=100	1360	1480
$\lambda=10$		
POP_MAX=20	740	820
POP_MAX=50	1360	1340
POP_MAX=100	2600	2380

$(\mu+\lambda)$	no of evaluations
(5+10)	605
(5+25)	**555**
(10+20)	630
(10+50)	760
(20+40)	820
(20+100)	1220

Table 2. Results for the Branin function achieved by the ESOP algorithm (left) and the standard $(\mu+\lambda)$-ES.

From the results collected in tables 1-3 several conclusions can be formulated. The ESOP algorithm achieves usually better results than the standard $(\mu+\lambda)$ evolutionary strategy. The cost of it is usually lower than the cost of standard algorithm, but what is much more important, it is much more robust. The ESOP algorithm achieves the results comparable to the traditional $(\mu+\lambda)$-ES with the value of μ much greater than in the standard case. If the value of μ is greater, especially at the beginning of the optimisation, the chance of finding global extreme is larger (in the larger population there are higher chances for weaker chromosomes representing alternate solutions to survive). The value of μ in the case of the ESOP algorithm is initially, when the strategy *explores* the domain, equal to parameter POP_MAX, which is significantly greater than in the standard case.

	S_MIN =0,01	S_MIN =0,1
λ=2		
POP_MAX=20	-	-
POP_MAX=50	960	980
POP_MAX=100	1560	-1
λ=5		
POP_MAX=20	-	-
POP_MAX=50	1280	1340
POP_MAX=100	2480	2100
λ=10		
POP_MAX=20	**920**	921
POP_MAX=50	2181	2060
POP_MAX=100	3800	3500

(μ+λ)	no of evaluations
(5+10)	-
(5+25)	-
(10+20)	-
(10+50)	**1110**
(20+40)	1580
(20+100)	1720

Table 3. Results for the Goldstein-Price function achieved by the ESOP algorithm (left) and the standard (μ+λ)-ES.

CONCLUSIONS

We have introduced a novel approach to the ES. We allow population size to vary over time. The process is controlled by the stage of the optimisation. We assumed that the average value of σ_N can be used as an indicator of this stage. The method leads to reduction of the computational effort needed to achieve comparable results. Moreover our algorithm is more robust than in the standard case. We also reduced the number of parameters which have to be properly set from 2 in the standard case (μ, λ) to 1 (POP_MAX).

The author acknowledge financial support from the Warsaw University of Technology under Grant 503/033/016/1.

REFERENCES

[1]Pokraśniewicz J, Arabas J., Mulawka J.J., Evolutionary Strategies with Adaptively Changed Population Size, Proc. III European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, 1995, vol. 1 pp. 477-481

[2]Michalewicz, Genetic Algorithms + Data Structures =Evolution Programs, Springer, 1995

[3]Mühlenbein, H., Schlierkamp-Voosen, D., Theory and Application of the Breeder Genetic Algorithm, in [6] pp. 182-193

[4]Schwefel, H.-P., Evolution and Optimum Seeking, Wiley, 1995

[5]Torn, A., Zilinskas,A., Global Optimization, Springer, 1989

[6]Zurada, J.M., Marks, R.J., Robinson, C.J., Computational Intelligence Imitating Life, IEEE Press, New York, 1994

[7]Stuckman Bruce E., Easom Eric E., A Comparison of Bayesian/Sampling Global Optimization Techniques, IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, No. 5, September/October 1992