Step Size Adaptation in Evolution Strategies – Two Approaches

Sibylle D. Müller **Nicol N. Schraudolph** **Petros Koumoutsakos**
Institute of Computational Science
Swiss Federal Institute of Technology
8092 Zürich, Switzerland
{muellers,nic,petros}@inf.ethz.ch

Nikolaus Hansen
Fachgebiet für Bionik
Technische Universität
13355 Berlin, Germany
hansen@bionik.tu-berlin.de

Abstract

We present two novel approaches for the adaptation of step sizes in evolution strategies: (i) First, step size rules are learned during the optimization process. (ii) Second, an adaptation scheme is improved for evolution strategies with large population sizes.

1 Introduction

The development of step size adaptation schemes for evolution strategies (ES) has received much attention in the ES community. Starting from an early attempt, the so-called “1/5 success rule” [Rechenberg (1973)], applied to two-membered ES's, mutative step size control [Rechenberg (1994)] and self-adaptation schemes [Bäck (1996)] were developed, followed by derandomized mutational step size control schemes [Hansen & Ostermeier (1997)]. The latter two methods have become the state-of-the-art techniques that are usually implemented in ESs. These control schemes employing empirical rules and parameters have been proven successful for solving a wide range of real-world optimization problems.

On this topic, we suggest two novel approaches:

(i) Replace a priori defined adaptation rules by a more general mechanism that can adapt the step sizes during the evolutionary optimization process automatically. The key concept involves the use of a learning algorithm for the online step size adaptation. This implies that the optimization algorithm is not supplied with a pre-determined step size adaptation rule but instead the rules are evolved by means of learning. As an initial test for our approach, we consider the application of reinforcement learning (RL) to the 1/5 success rule in a two-membered ES, as described in Section 2.

(ii) To make efficient use of selection information in large populations, a novel scheme for adaptation of the mutation distribution is devised. This approach is implemented based on an existing powerful adaptation technique called covariance matrix adaptation [Hansen & Ostermeier (1997)], presented in Section 3.

2 Learning how to Adapt Step Sizes

2.1 Reinforcement Learning

Reinforcement learning (RL) is a learning technique in which an *agent* learns to find optimal *actions* for the current *state* by interacting with its environment. The agent should learn a control strategy, also referred to as *policy*, that chooses the optimal actions for the current state to maximize its cumulative *reward*. For this purpose, the agent is given a reward by an external trainer for each action taken. The reward can be immediate or delayed. Sample RL applications are learning to play board games or learning to control mobile robots. In the robot example, the robot is the *agent* that can sense its environments such that it knows its location, i.e., its *state*. The robot can decide which *action* to choose, e.g., to move ahead or to turn from one state to the next. The goal may be to reach a particular location and for this purpose the agent has to learn a *policy*.

The learning task can be divided into discrete time steps, t. The agent determines at each step the environmental state s_t, and decides upon an action a_t. By performing this action, the agent is transferred to a new state $s_{t+1} = \delta(s_t, a_t)$ and given a reward r_{t+1}. This reward is used to update a *value function* that can be either a *state-value function* $V^{\pi}(s)$ depending on states or an *action-value function* $Q^{\pi}(s, a)$ depending on states and actions. To each state or state-action pair, the largest expected future reward is assigned by the *optimal* value functions $V^{*}(s)$ or $Q^{*}(s, a)$, respectively. The optimal state-value function $V^*(s)$ can be learned only if both the reward function r and the function δ that describes how the agent is transferred from state s_t to state s_{t+1} are explicitly known. Usually, however, r and δ are unknown. In this case, the optimal action-value function $Q^*(s,a)$ can be learned.

In this paper, we consider only RL methods for which the agent can learn the Q function, thereby being able to select optimal actions without knowing explicitely the reward function r or the state transition function $\delta(s,a)$. From this class of Temporal Difference (TD) methods, we present the SARSA algorithm in pseudocode [Sutton (1998)]:

```
Initialize $Q(s_t, a_t)$ arbitrarily.
Repeat (for each episode):
  Initialize $s_t$.
  Choose $a_t$ from $s_t$ using policy derived from $Q(s_t, a_t)$
  using e.g. an $\varepsilon$-greedy selection scheme.
  Repeat (for each step of episode):
    Take action $a_t$, observe $r_t$, $s_{t+1}$.
    Choose $a_{t+1}$ from $s_{t+1}$ using policy from $Q(s_{t+1}, a_{t+1})$
    using e.g. an $\varepsilon$-greedy selection scheme.
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$
    $s_t \leftarrow s_{t+1}, a_t \leftarrow a_{t+1}$
  until $s_t$ is terminal.
```

SARSA employs three learning parameters: α is a learning rate, γ a discount factor, and ε a greediness parameter. Constant learning rates α cannot be used because we have to deal with a non-deterministic environment. For the considered problem, a learning rate of

$$\alpha_t = \frac{1}{N_v(s_t, a_t)} \tag{1}$$

is recommended in [Mitchell (1997)] where N_v is the total number of times the state-action pair (s_t, a_t) has been visited up to and including the t-th iteration. Discount factors $\gamma < 1$ establish a preference for immediate over future rewards, and may be necessary to avoid that the expected future reward becomes infinite. We set $\gamma = 0.9$ arbitrarily. The action with the highest value function is chosen with probability $(1 - \varepsilon)$ (greedy action selection) whereas an action is selected at random with (small) probability ε. We choose $\varepsilon = 0.1$.

2.2 Combination of RL and ES

The idea is to use a RL method to learn a step size adaptation rule in ESs. We consider a two-membered ES employing the 1/5 success rule. The 1/5 success rule was originally formulated as follows [Schwefel (1995)]:

After every n mutations, check how many successes have occured over the preceding $10n$ mutations. If this number is less than $2n$, multiply the step lengths by the factor 0.85; divide them by 0.85 if more than $2n$ successes occured.

In our approach, the frequency with which step sizes are updated is kept the same (after every $10n$ mutations). Also, the step size adaptation factor of 0.85 remains constant.

RL is introduced to learn from the measured states, i.e. the success rates; the actions can be (1) to increase the step size (divide by the step size adaptation factor), (2) to decrease the step size (multiply by the step size adaptation factor), or (3) to keep the step size constant. As the success rate is determined by looking back over the last $10n$ mutations, the total number of different states is $10n + 1$, including the case of a success rate of zero. Therefore, the $Q(s,a)$ table to be learned consists of $(10n+1)\cdot 3$ state-action pairs. The reward is defined as

$$r = ||\boldsymbol{x}^{(g)} - \boldsymbol{x}^{(g-\Delta g)}|| \cdot \operatorname{sgn}\left(f^{(g)} - f^{(g-\Delta g)}\right) \tag{2}$$

where Δg is the difference in generations for which the reward is computed. $Q(s,a)$ is initialized with uniformly random numbers from the range [-1,1]. The combined algorithm is called RL-ES.

From the 1/5 success rule, we would expect a Q value table as shown for $n = 1$ in Table 1.

Success rate $[\cdot 10^1]$	Action: increase	Action: decrease	Action: keep
0,1		+	
2			+
3,4,5,6,7,8,9,10	+		

Table 1: Schema of the Q value table as it should look if the 1/5 success rule is learned. The "+" denotes the highest Q value in each row.

2.3 Results

The RL-ES algorithm is tested on the optimization of the sphere and the Rosenbrock function in several dimensions and compared with the original (1+1)-ES. The two functions are defined as

1. $f_{\text{sphere}}(\boldsymbol{x}) = \Sigma_{i=1}^{n}(x_i - 1)^2$, $\sigma^{(0)} = 1$, $\boldsymbol{x}^{(0)} = \boldsymbol{0}$,
2. $f_{\text{Rosenbrock}}(\boldsymbol{x}) = \Sigma_{i=1}^{n-1}(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$, $\sigma^{(0)} = 0.1$, $\boldsymbol{x}^{(0)} = \boldsymbol{0}$,

and the optimization is terminated as soon as $f < 10^{-10}$. If the termination criterion is not met within N_t generations, the run is called *not converged.* We determine a *convergence rate* that is the ratio of converged runs over the total number of runs. The *convergence speed* is measured by counting the number of function evaluations until convergence.

The convergence rate and the average and standard deviation for the number of generations to reach convergence are listed in Table 2 for the (1+1)-ES and the RL-ES.

Problem	(1+1)-ES	RL-ES (conv. rate; N_t)
Sph 1D	97 ± 17	337 ± 639 (1.00; 10^4)
Sph 5D	470 ± 35	1346 ± 1092 (0.99; 10^4)
Sph 20D	1928 ± 70	4313 ± 1795 (0.94; 10^4)
Sph 80D	8234 ± 80	19382 ± 9729 (1.00; 10^5)
Ros 2D	16205 ± 973	31365 ± 65135 (1.00; 10^6)
Ros 5D	140227 ± 1487	311704 ± 319148 (1.00; 10^7)

Table 2: Number of iterations until convergence is reached for the (1+1)-ES and for the RL-ES. Results are averaged over 30 runs except for the sphere function optimized by RL-ES (1000 runs). The convergence rate for the (1+1)-ES is always 1.0.

The convergence rate of the RL-ES is close to that of the (1+1)-ES and the number of iterations to convergence is larger than that of the (1+1)-ES by a factor of 2-3. Given that the RL-ES has to learn which of the three actions to choose, this factor seems reasonable.

3 Adaptation of Step Sizes for Large Populations

One of the commonly proposed advantages of ESs is that they can be easily parallelized, see e.g.[Schwefel (1995)]. ESs with λ children per generation (population size λ) are usually parallelized by distributing the function evaluation for each of the λ children to a different processor. When the number of children is smaller than the number of available processors, the advantage of using ESs in parallel cannot be fully exploited. Consequently, for a large number of processors the algorithm should be able to use a large population efficiently.

We consider a derandomized ES with covariance matrix adaptation (CMA-ES) for which experimental results [Hansen & Ostermeier (1997)] show a clear convergence velocity improvement when compared to other ESs.

When optimizing reasonably complex (e.g. highly non-separable functions), the adaptation time becomes the limiting factor for the performance of the CMA-ES if the problem dimension n exceeds a certain threshold, usually $n \geq 10$. That is, the number of generations to adapt the covariance matrix of the search distribution to the function topology is the dominant factor in the degraded performance of the algorithm. The reason is that in the CMA-ES $(n^2 + n)/2$ elements of the symmetric covariance matrix $\boldsymbol{C}$ need to be adapted while the search process itself only adjusts n variables.

For population sizes $\lambda > 20$ the adaptation time becomes practically independent of the population size. That means, the number of function evaluations required increases linearly with increasing population size. In other words, the implementation of the original CMA-ES on massively parallel computer architectures, such as Beowulf clusters with hundreds of processors, offers no substantial advantage compared to the use of twenty processors. On complex functions, the time complexity is of $\mathcal{O}(n^2)$, independent of the population size and the number of processors.

How can we use the CMA-ES efficiently on massively parallel architectures with hundreds of processors? How can we increase the efficiency of the CMA-ES, when a large population is preferable to a small one due to other reasons? To increase λ alone does not help shortening the adaptation time as pointed out above. Additionally, a faster adaptation mechanism of comparable reliability must be implemented. We increase the adaptation rate of the covariance matrix by exploiting a larger amount of information per generation.

The proposed modification for the covariance matrix update reads as follows:

$$\begin{aligned}\boldsymbol{C}^{(g+1)} &= (1 - c_{\text{cov}}) \cdot \boldsymbol{C}^{(g)} \\ &+ c_{\text{cov}}\Big(\alpha_{\text{cov}} \cdot \boldsymbol{p}_c^{(g+1)} \left(\boldsymbol{p}_c^{(g+1)}\right)^T \\ &+ (1 - \alpha_{\text{cov}}) \cdot \boldsymbol{Z}^{(g+1)}\Big) \qquad (3)\end{aligned}$$

where the change rate of the covariance matrix $c_{\text{cov}} \in [0, 1[$ and a blending factor $\alpha_{\text{cov}} \in [0, 1]$. This modified update is identical with the original method if $\alpha_{\text{cov}} = 1$.

The evolution path $\boldsymbol{p}_c$ is an exponential average of subsequently selected averaged mutation steps $\sqrt{\boldsymbol{C}}\langle \boldsymbol{z}\rangle$. Here, the matrix square root $\sqrt{\boldsymbol{C}}$ is obtained by an eigenvalue decomposition of the covariance matrix $\boldsymbol{C}$. $\langle \boldsymbol{z}\rangle$ denotes the average of selected mutation steps z_i. The outer product $\boldsymbol{p}_c(\boldsymbol{p}_c)^T$ is a symmetrical $n \times n$ matrix with rank one.

In contrast, $\boldsymbol{Z}$ is computed from $\sqrt{C}(\sum_i z_i z_i^T)\sqrt{C}$. The difference lies in the fact that here we first take the outer product of the selected mutation steps and average afterwards. The symmetric $n \times n$ matrix $\boldsymbol{Z}$ has rank $\min(\mu, n)$, and therefore contains more selection information.

An increased population usually contains more information to be exploited in order to obtain a reduced adaptation time. Compared to the original adaptation mechanism the proposed modification usually requires much fewer generations when λ is large but could be slightly less effective in small populations.

A detailed analysis of this approach can be found in [Müller, Hansen & Koumoutsakos (2002)].

4 Conclusions

We propose two novel methods for step size adaptation in evolution strategies. The first algorithm combines elements from the 1/5 success rule in a (1+1)-ES with reinforcement learning (RL). Heuristics in the (1+1)-ES are reduced and replaced with a more general learning method. Convergence speed and rate of the combined scheme called RL-ES, measured on the sphere and Rosenbrock functions in several dimensions, are compared with those of the (1+1)-ES. The RL-ES yields the same convergence rate (100 %) as the (1+1)-ES but is slower by a factor of about 2-3 on both functions, a result that meets our expectations. The second approach improves the adaptation of the mutation distribution in ESs with large populations. Its application on various test functions shows that the number of generations to convergence can be decreased from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ if $\mathcal{O}(n)$ processors are used in parallel, see [Müller, Hansen & Koumoutsakos (2002)].

References

[Bäck (1996)] Bäck, T., 1996, Evolutionary Algorithms in Theory and Practice, Oxford University Press.

[Hansen & Ostermeier (1997)] Hansen, N. & Ostermeier, A., 1997, Convergence Properties of Evolution Strategies with the Derandomized Covariance Matrix Adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES, *Proceedings of the 5th European Conference on Intelligent Techniques and Soft Computing (EUFIT '97)*, 650-654.

[Mitchell (1997)] Mitchell, T. M., 1997, Machine Learning, McGraw-Hill.

[Müller, Hansen & Koumoutsakos (2002)] Müller, S.D., Hansen, N. & Koumoutsakos, P., 2002, Increasing the Serial and the Parallel Performance of the CMA-Evolution Strategy with Large Populations, *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN 2002)*.

[Rechenberg (1973)] Rechenberg, I., 1973, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart.

[Rechenberg (1994)] Rechenberg, I., 1994, Evolutionsstrategie 94, Frommann-Holzboog, Stuttgart.

[Schwefel (1995)] Schwefel, H.-P., 1995, Evolution and Optimum Seeking, John Wiley and Sons, New York.

[Sutton (1998)] Sutton, R., 1998, Reinforcement Learning – An Introduction, MIT Press, Cambridge.