

# A new selection ratio for large population sizes

Fabien Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), bat 490 Univ. Paris-Sud  
91405 Orsay, France, fteytaud@lri.fr

**Abstract.** Motivated by parallel optimization, we study the Self-Adaptation algorithm for large population sizes. We first show that the current version of this algorithm does not reach the theoretical bounds, then we propose a very simple modification, in the selection part of the evolution process. We show that this simple modification leads to big improvement of the speed-up when the population size is large.

## 1 Introduction

Evolution Strategies [4, 1] are well-known methods for solving an optimization problem. Their main advantages are robustness and simplicity, but, because they are population based they are also well suitable for parallelization. However, it has been shown in [2, 6] that ES are not very efficient for large population sizes whereas we can notice that the number of parallel machines, supercomputers and grids, has increased, suggesting big population sizes.

We define  $\lambda$  as the population size,  $\mu$  as the number of offspring used for the recombination of the parent, and  $N$  as the dimensionality. Then, a  $(\mu/\mu, \lambda)$ -ES (Evolution Strategy) is an evolution strategy with a population size of  $\lambda$  and the  $\mu$  best offspring will be used, with equal weights, to create the new parent. A  $(1, \lambda)$ -ES is an evolution strategy in which the new parent is simply the best offspring among the population.

It is widely believed that  $(\mu/\mu, \lambda)$ -ES are more parallel (efficient for large population sizes) than  $(1, \lambda)$ -ES, which is shown by rough calculus as in e.g. [1] and by experiments with intermediate values of  $\lambda$ .

Experimentally however,  $(\mu/\mu, \lambda)$ -ES are less efficient than  $(1, \lambda)$ -ES for a (sufficiently) large population size  $\lambda$ , for the usual parametrization of the algorithms, as shown in [6]; in particular, the theoretical speedups of ES for parallel optimization (the best possible speed-ups, for optimal algorithms) are far better than the results of current implementations. In this paper, we show that changing the parametrization of self-adaptive  $(\mu/\mu, \lambda)$ -ES leads to a huge improvement for large  $\lambda$ , and that, with this improvement we can reach the theoretical bounds shown in [7]. These bounds are :

- For  $(\mu/\mu, \lambda)$ -ES, the speed-up is linear until  $\lambda$  of the same order as the dimensionality. For larger value of  $\lambda$  the speed-up becomes logarithmic as a function of  $\lambda$ .

- For  $(1, \lambda)$ -ES, the speed-up is logarithmic as a function of  $\lambda$ .

Choosing a good selection ratio  $\frac{\mu}{\lambda}$  is crucial whatever the dimensionality. Usually, the number of selected individuals  $\mu$  is function of the population  $\lambda$ ; in general,  $\mu = \frac{\lambda}{2}$  as suggested in [3] for the covariance matrix adaptation algorithm;  $\mu = \frac{\lambda}{4}$  is suggested for the covariance matrix self-adaptation algorithm in [2] which is especially devoted to big population sizes.

We work on the Self-Adaptation Evolution Strategy (SA-ES). It has been proposed in [4] and [5], and extended in [2], in the special case of large  $\lambda$ . This algorithm is very simple and provides one of the state of the art results for large  $\lambda$ . This algorithm is presented in Algorithm 1.  $N(0, Id)$  denotes standard multivariate Gaussian random variables with identity covariance matrix.

---

**Algorithm 1** Mutative self-adaptation algorithm. For large population sizes  $\tau$  is usually equal to  $1/\sqrt{N}$ ; other tuning of  $\tau$  can be used (e.g.  $1/\sqrt{2N}$ ) which is sometimes found in papers.

---

Initialize  $\sigma^{avg} \in \mathbb{R}$ ,  $y \in \mathbb{R}^N$ .

**while** Halting criterion not fulfilled **do**

**for**  $i = 1.. \lambda$  **do**

$\sigma_i = \sigma^{avg} e^{\tau N_i(0,1)}$

$z_i = \sigma_i N_i(0, Id)$

$y_i = y + z_i$

$f_i = f(y_i)$

**end for**

  Sort the individuals by increasing fitness;  $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$ .

$z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z_{(i)}$

$\sigma^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} \sigma_{(i)}$

$y = y + z^{avg}$

**end while**

---

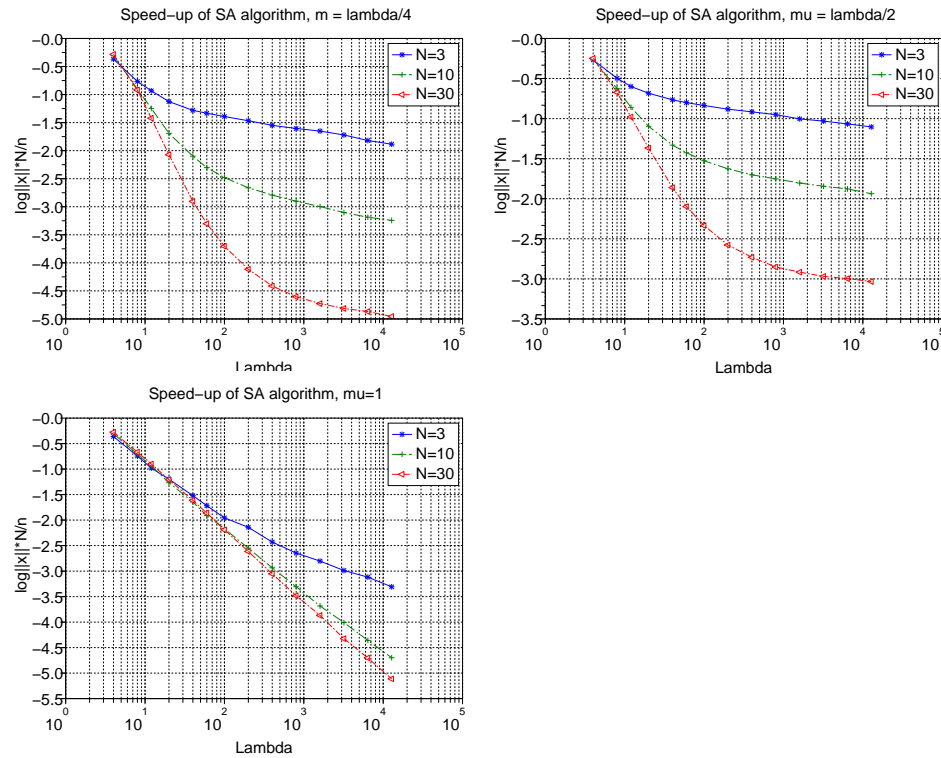
In this paper, we are considering minimization problems.

## 2 Discrepancy between theory and practice, and the tuning of the selection ratio of the Self-Adaptation Evolution Strategy

In [7], a lot of theoretical bounds have been shown, and, especially that we should reach  $\Theta(\log(\lambda))$  with a  $(\mu/\mu, \lambda)$ -ES. In practice (*i.e.* with usual parametrizations of the algorithm), the SA-ES does not reach that speed-up, as shown in Fig. 1.

In Fig. 1, we can note that the selection ratio  $\mu$  is a crucial parameter for the behaviour of the SA algorithm. Unless  $\mu = 1$ , the speed-up reaches a plateau when  $\lambda$  is very large.

$\mu = \frac{\lambda}{2}$  is never a good choice whatever the population size, and if the population

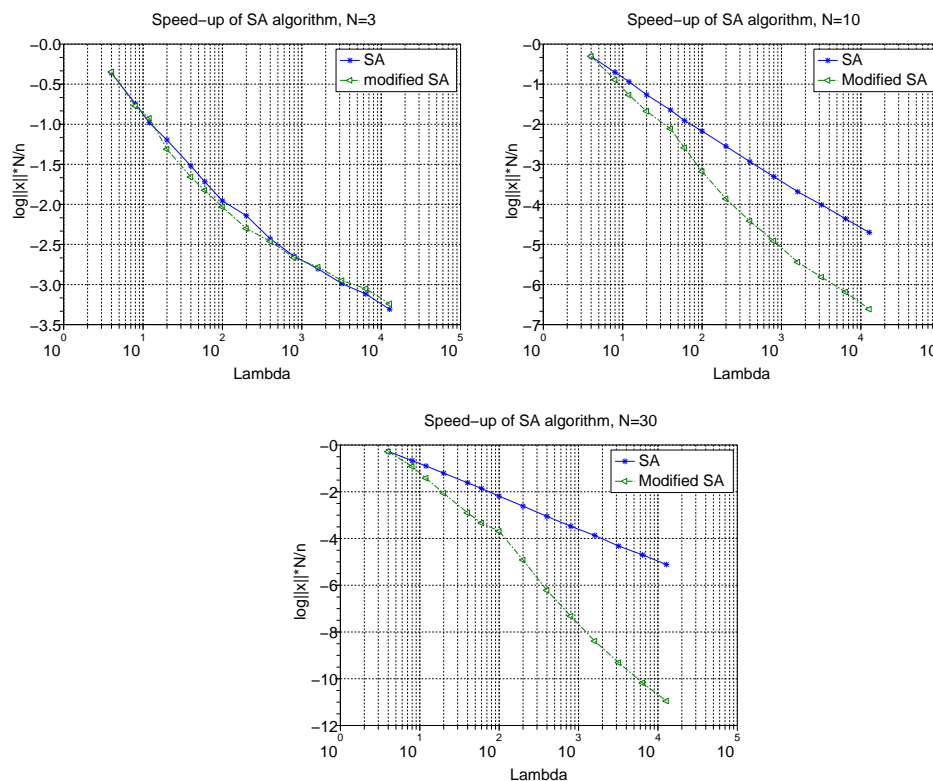


**Fig. 1.** Speed-up of the Self-Adaptation algorithm as a function of  $\lambda$ , on the sphere function. For this experiment, the initial step size is 1, the initial point is  $(1, \dots, 1)$  and the fitness function to hit is  $10^{-10}$ . Top left ( $\mu = \frac{\lambda}{4}$ ) and top right ( $\mu = \frac{\lambda}{2}$ ) do not show a logarithmic speedup whatever the dimensionality.

size becomes really large, the speedup tends to a constant.  $\mu = \frac{\lambda}{4}$  is often used in the literature because it offers a good convergence if the population size is not too large. In fact, Fig. 1 shows that in dimension 10,  $\mu = \frac{\lambda}{4}$  is better than  $\mu = 1$  until a population size of 200. In dimension 30,  $\mu = \frac{\lambda}{4}$  is also a good choice, in particular if the population size is smaller than 6400. The main problem is that the convergence rate tends to a constant (depending on the dimensionality); theoretical results suggest that we should have something better. As said previously, in [7], it is shown that for  $(\mu/\mu, \lambda)$ -ES algorithms the convergence rate should be linear as a function of  $\lambda$  if the population size is smaller than the dimension. If the population is larger than the dimension the speedup is  $\Theta(\log(\lambda))$ .

In practice, this is not true. As we can observe in Fig. 1, if the population size is large compared to the dimension, the convergence rate of  $(\mu/\mu, \lambda)$ -ES tends to a constant.  $(1, \lambda)$ -ES only shows a logarithmic convergence rate as a function of  $\lambda$ , which suggests that this is the best choice when the population size is large.

### 3 Solving the discrepancy between theory and practice by a good tuning of SA



**Fig. 2.** This figure shows the convergence rate on the sphere function for the Self-Adaptation algorithm and for the modified version of this algorithm. In the modified self adaptation algorithm, the selection ratio is chosen equals to  $\mu = \min(N, \lambda/4)$ . For the standard self adaptation algorithm, we have chosen the best selection ratio for a large population size ( $\mu = 1$ ). Results are obtained for three dimensions 3, 10 and 30. In dimensions 10 and 30, the modified self adaptation algorithm outperforms the standard version. In dimension 3, results are really close, simply because in that case, the selection ratio of both versions of the algorithm is almost the same (due to the small dimensionality).

In Fig. 2 we show that we can improve the convergence rate by using a simple rule for the selection ratio. Experiments have been done on the sphere function. The initial point is  $(1, \dots, 1)$  and the initial step size is 1. The fitness function to reach is  $f_{stop} = 10^{-10}$ . In this figure, we compare the Self-Adaptation algorithm with  $\mu = 1$  and a modified version of this algorithm. We have chosen  $\mu = 1$  for

the comparison because it is the best choice for the SA algorithm for large  $\lambda$ , as shown in the previous section.

The modified version is a SA algorithm but with a number of selected points  $\mu$  equal to the minimum between the dimensionality and the population size divided by 4. Formally, in that case,  $\mu = \min(N, \frac{\lambda}{4})$ . Intuitively, we want  $\mu = N$  for large  $\lambda$ , and  $\lambda/4$  for small value of the population size.

In dimension 3, both convergence rates are very close, because the value of the selection ratio of the Self-Adaptation algorithm and of the modified Self-Adaptation algorithm are close to each other.

In dimension 10, the modified version of the Self-Adaptation algorithm outperforms the standard version. We also have a logarithmic convergence rate as predicted by the theory. For a population size of 12800 we have a speedup of 41% over the Self-Adaptation algorithm with  $\mu = 1$ .

In dimension 30 finally, we have a really big improvement. We have a logarithmic convergence rate, as a function of the population size, and here again results for large population sizes are really good, with a speedup of 114% for a population size of 12800. In this case, for small population sizes, the convergence rate is also slightly better than  $\mu = \frac{\lambda}{4}$ .

## 4 Experiments on the Covariance Matrix Self-Adaptation Evolution Strategy

The Covariance Matrix Self-Adaptation Evolution Strategy (CMSA-ES) has been proposed in [2], and is now the state of the art algorithm for large population sizes. This algorithm is based on the SA algorithm presented previously. In the CMSA-ES, the routine used to adapt the global step size  $\sigma$  is the Self-Adaptation one. But we have here a full covariance matrix adaptation algorithm, in order to learn the shape of the fitness function. This algorithm is presented in Algorithm 2.

Following the recommendations in [2] for the tuning of the CMSA-ES algorithm, a selection ratio  $\mu/\lambda$  equals to  $1/4$  should be used. In this section, we experiment the CMSA evolution strategy with two different selection ratios: the previous one which is recommended, and we also tried our modified version  $\mu = \min(N, \lambda/4)$  as above.

### 4.1 Test functions

We do our experiments on three well-known functions. Functions are defined in Table 1. The first one is the sphere function, which is a well-known test function in optimization. The second one is the Schwefel Ellipsoid for which the adaptation of the covariance matrix is helpful. The third one is the Rosenbrock

---

**Algorithm 2** Covariance Matrix self-adaptation.  $\tau$  is equal to  $1/\sqrt{N}$ ;  $\langle . \rangle$  represents the recombination. The initial covariance matrix  $C$  is the identity matrix. The time constant  $\tau_C$  is equal to  $1 + \frac{N(N+1)}{\lambda}$ .

---

```

Initialize  $\sigma^{avg} \in \mathbb{R}, y \in \mathbb{R}^N, C$ .
while Halting criterion not fulfilled do
  for  $i = 1.. \lambda$  do
     $\sigma_i = \sigma^{avg} e^{\tau N_i(0,1)}$ 
     $s_i = \sqrt{C} \sigma_i N_i(0, Id)$ 
     $z_i = \sigma_i s_i$ 
     $y_i = y + z_i$ 
     $f_i = f(y_i)$ 
  end for
  Sort the individuals by increasing fitness;  $f_{(1)} < f_{(2)} < \dots < f_{(\lambda)}$ .
   $z^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} z^{(i)}$ 
   $s^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} s^{(i)}$ 
   $\sigma^{avg} = \frac{1}{\mu} \sum_{i=1}^{\mu} \sigma^{(i)}$ 
   $y = y + z^{avg}$ 
   $C = (1 - \frac{1}{\tau_C})C + \frac{1}{\tau_C} \langle ss^T \rangle$ 
end while

```

---

function which requires, due to its shape, continuous changes of the covariance matrix.

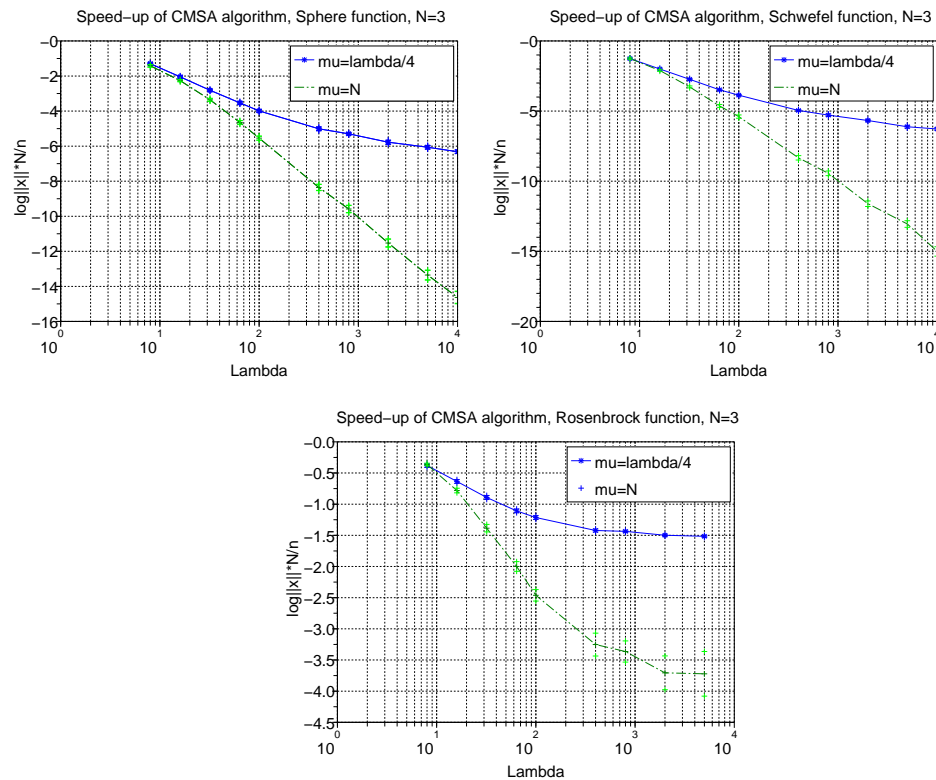
Name	Objective function	$y_{init}$	$\sigma_{init}$	$f_{stop}$
Sphere	$f(y) = \sum_{i=1}^N y_i^2$	$(1, \dots, 1)$	1	$10^{-10}$
Schwefel	$f(y) = \sum_{i=1}^N (\sum_{j=1}^i y_j)^2$	$(1, \dots, 1)$	1	$10^{-10}$
Rosenbrock	$f(y) = \sum_{i=1}^N (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2)$	$(0, \dots, 0)$	0.1	$10^{-10}$

**Table 1.** Test functions considered in this paper.

## 4.2 Results

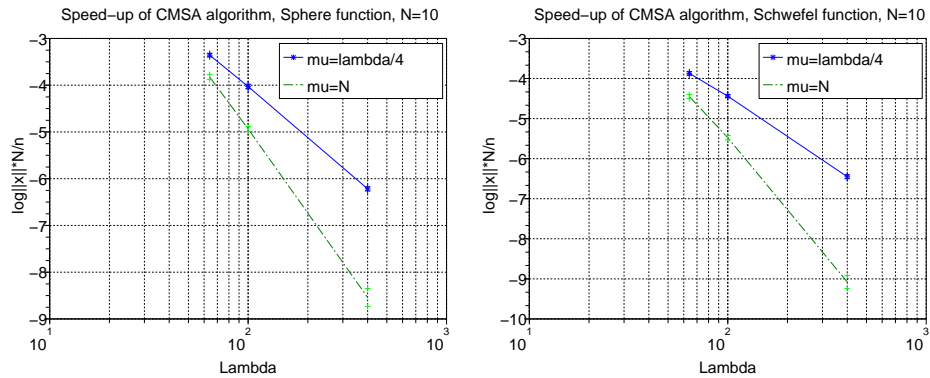
In our experiments, we have considered different dimensionalities,  $N = 3, 10, 30$ . We have compared different selection ratios for different population sizes, specially large population sizes. For each point of each experiment we take the average of 60 independent runs. We have chosen to look at the convergence rate. We measure  $\frac{N \times \log \|y\|}{number\_of\_Iterations}$  as a function of the population size. Choosing to measure the convergence rate (instead of the number of iterations as in [2])

is here justified by the fact that we know the theoretical limits of this criterion (up to a constant factor), and we know that many usual algorithms have only a bounded speed-up; we want to take care of this.



**Fig. 3.** Results of the CMSA evolution strategy in dimension 3 for the sphere function (top left), the Schwefel function (top right) and the Rosenbrock function. Convergence rate for the standard selection ratio ( $\mu = \lambda/4$ ) and the new selection ratio ( $\mu = \min(N, \lambda/4)$ ) are plotted. Choosing a selection ratio equals to  $\mu = \min(N, \lambda/4)$  is a good choice in the three experiments. We reach a logarithmic speedup on the sphere function and the Schwefel function.

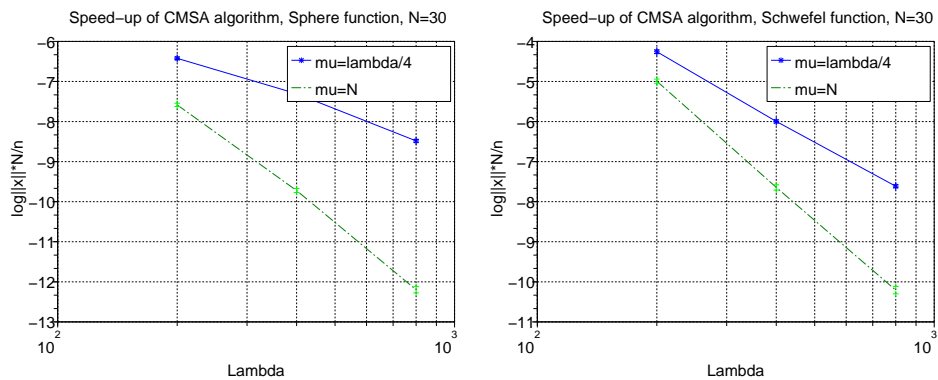
Fig 3 shows that using a bad selection ratio could be harmful, especially for large population sizes. Even for not so large population sizes (e.g.  $\lambda = 20$ ) using the selection ratio  $\mu = \min(N, \lambda/4)$  is a good choice. The best speed-ups are reached for large population sizes, more precisely we obtain a speed-up of 136% for the sphere function and 139% for the Schwefel function for a population size equals to 10000 (more than twice faster in both cases), and 146% for the Rosenbrock function for a population size of 5000.



**Fig. 4.** Results of the CMSA evolution strategy in dimension 10 for the sphere function (top left) and the Schwefel function (top right). Convergence rate for the standard selection ratio ( $\mu = \lambda/4$ ) and the new selection ratio ( $\mu = \min(N, \lambda/4)$ ) are plotted.

In Fig 4 we experiments the sphere function and the Schwefel function in dimension 10. Due to big computation time, the largest population size for this experiment is 800. Speedup of 37% is reached for the sphere function and 41% for the Schwefel function for a population size of 400.

In Fig 5, we reproduce the experiments as previously, but in bigger dimensionality, 30. Results are similar, and we reach a speedup of 44% for the sphere function and 34% for the Schwefel function for  $\lambda = 800$ .



**Fig. 5.** Results of the CMSA evolution strategy in dimension 3 for the sphere function (top left) and the Schwefel function (top right). Convergence rate for the standard selection ratio ( $\mu = \lambda/4$ ) and the new selection ratio ( $\mu = \min(N, \lambda/4)$ ) are plotted.

## 5 Conclusion

In this paper we experiment a new rule for the selected population size  $\mu = \min(N, \lambda/4)$ . This rule is a simple modification, and has very good results for the CMSA evolution strategy. We can seemingly reach a logarithmic speed-up with this rule, which is consistent with the theoretical bounds.

To summarize this paper, we have shown in section 2 that the current version of the Self Adaptation rule only reaches the theoretical speedup (logarithmic as a function of  $\lambda$ ) when the selection ratio is equal to  $\frac{\mu}{\lambda} = 1/\lambda$ , *i.e.*  $\mu = 1$ .

A selection ratio of 1/2 or 1/4 is better at first view, but it's indeed harmful when the population size  $\lambda$  is large enough (larger than the dimension).

In section 3, we have shown that having  $\mu = \min(N, \lambda/4)$  could lead to a very good speedup, better than both with  $\mu = 1$  and with  $\mu = \lambda/4$  or  $\mu = \lambda/2$ . Finally, we experiment the same rule on the CMSA algorithm (section 4), which is known to be a good evolution strategy when the population size is large.

Unfortunately, due to hard computation time, we only have experimented with small population sizes for dimensions 10 and 30. Experiments with large population sizes should be done in the future. Another future work will be to try this kind of modification on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), *i.e.* in algorithms using cumulative step-length adaptation.

## References

1. H.-G. Beyer. *The Theory of Evolutions Strategies*. Springer, Heidelberg, 2001.
2. H.-G. Beyer and B. Sendhoff. Covariance matrix adaptation revisited - the CMSA evolution strategy. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, editors, *Proceedings of PPSN*, pages 123–132, 2008.
3. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
4. I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.
5. H.-P. Schwefel. Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin, Juli 1974.
6. F. Teytaud and O. Teytaud. On the parallel speed-up of Estimation of Multivariate Normal Algorithm and Evolution Strategies. In *Proceedings of EvoStar 2009*, pages 655–664, 2009.
7. O. Teytaud and H. Fournier. Lower bounds for evolution strategies using vc-dimension. In *PPSN*, pages 102–111, 2008.