Comparison of Steady-State and Generational Evolution Strategies for Parallel Architectures

Razvan Enache, Bernhard Sendhoff, Markus Olhofer, Martina Hasenjäger

2004

Preprint:

This is an accepted article published in Parallel Problem Solving from Nature – PPSN VIII. The final authenticated version is available online at: https://doi.org/[DOI not available]

X. Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*,
Lecture Notes in Computer Science 3242, pages 253-262. Springer, 2004.

Comparison of Steady-State and Generational Evolution Strategies for Parallel Architectures

Razvan Enache[1] Bernhard Sendhoff[2] Markus Olhofer[2] Martina Hasenjäger[2]

[1] École Nationale Supérieure de Télécommunications
46 rue Barrault 75013 Paris France
[2] Honda Research Institute Europe
Carl-Legien-Strasse 30, 63073
Offenbach am Main, Germany

Abstract. Steady-state and generational selection methods with evolution strategies were compared on several test functions with respect to their performance and efficiency. The evaluation was carried out for a parallel computing environment with a particular focus on heterogeneous calculation times for the assessment of the individual fitness. This set-up was motivated by typical tasks in design optimization. Our results show that steady-state methods outperform classical generational selection for highly variable evaluation time or for small degrees of parallelism. The 2D turbine blade optimization results did not allow a clear conclusion about the advantage of steady-state selection, however this is coherent with the above findings.

1 Introduction

Evolution strategies (ES) are a class of evolutionary algorithms designed for real-valued parameter optimization that are known to show very good performance on high-dimensional and multi-modal problems. Evolution strategies have been used successfully on design problems such as turbine blade optimization [4, 7]. Since the computation time needed to evaluate an individual in real world applications can be very long, evolution strategies are often parallelized on multi-computer clusters. The "basic" parallelization of evolution strategies is very simple if one processor can be reserved for each individual. If the evaluation of each individual takes about the same time, this type of parallelization is maximally efficient. However, if one of the two constraints is not fulfilled, efficiency can degrade rapidly. When every processor but one waits for the last evaluation, the cluster's efficiency deteriorates to that of a single processor. There are two main reasons why in practical applications this breakdown of efficiency occurs frequently. First, even if a sufficient number of processors is available, computing power has often to be shared with other projects. Second, the evaluation times often vary drastically between individuals. For example, computational

fluid-dynamics (CFD) calculations work iteratively until a convergence criterion is satisfied. However, the number of iterations can vary between designs. Furthermore, design optimization is often used in conjunction with meta-models [3]. In this case, the evaluation time of individuals for which the meta-model is used is of orders of magnitude smaller than the evaluation time of the ones for which the CFD tool is used (milliseconds compared to hours).

Therefore, there is a need to explore possibilities to replace generational selection methods in evolution strategies by steady-state type selection algorithms, which integrate a newly evaluated individual directly in the population without waiting for the evaluation results of the whole generation. The overall target is to achieve parallelization without a break-down of efficiency if individual evaluation times vary drastically.

The first steady-state ES, the $(\mu+1)$, was proposed by Rechenberg [5]. More recently two steady-state selection methods have been proposed, which claim to be compatible with the self-adaptation principle of evolution strategies: median selection [8, 9] and continuous selection [6]. Both are non-elitist selection mechanisms.

On a single processor, these steady-state ESs have already been shown to perform better on many test functions than the typically used generational (μ, λ) or $(\mu + \lambda)$ methods. However, the performance and the efficiency on parallel architectures have not been evaluated yet to the extent necessary to be able to replace generational selection with steady-state selection in a practical application of optimization, like in the design optimization of turbine blades.

This comparison is the target of this paper. We test the performance of all three selection methods for varying degrees of parallelism and for different distributions of the evaluation time in Sec. 4. Before we do so, we will recall the steady-state selection methods for ES in the next section and define our experimental set-up in Sec. 3. In Sec. 5, we will extend our comparison to include a real optimization case, the design of the 2D cross-section of a turbine blade. In Sec. 6 we will conclude this paper.

2 Selection Methods in Evolution Strategies

The idea of steady-state selection is not new to evolutionary strategies. Indeed, the first type of multi-membered ES proposed by Rechenberg, the $(\mu+1)$-ES, was a steady-state algorithm! However, it turned out that due to a tendency to reduce the mutation strength one of the key features of ES – the self-adaptation of the strategy parameters – could not be realized with this algorithm [1], hence this direction of research was abandoned for the time being in favor of the study of $(\mu \dot{+} \lambda)$ strategies. Recently interest in steady-state ESs has revived again and with median selection [9, 8] and continuous selection [6] two algorithms have been proposed that take different courses in tackling the problem of endowing steady state ESs with self-adaptation.

In generational selection the problem of self-adaptation is solved by introducing an offspring population of λ individuals from which μ individuals are selected

to become the parent population of the next generation. We consider the case of comma selection where in each generation the complete parent population is replaced by newly created individuals.

In steady-state ES, we cannot rely on an offspring population since only one newly evaluated individual is available at a time. Thus we have to exploit information that is available from previously evaluated individuals.

Median selection [9, 8] does this by observing that in generational selection only the best offspring individuals have the chance to be selected at all for generating the next parent population. This defines a fitness threshold that separates individuals which can be discarded immediately from those that may be selected. In the same spirit median selection only accepts individuals to the parent population whose rank with respect to the fitness of the last n_m evaluated individuals exceeds a certain threshold rank. In order to avoid elitism and the related disadvantages always the oldest parent is replaced when a new individual is admitted to the parent population. Note that, similarly to generational selection, median selection handles two populations: a parent population containing the last μ selected individuals and a history buffer containing the last n_m evaluated individuals.

Continuous selection [6] proceeds differently. Here the basic idea is that each individual has a limited reproduction capacity, i.e. it is only able to produce a finite number of offspring. Practically, continuous selection maintains a pool of n_c previously evaluated individuals from which the best μ have the chance to produce new offspring. Either the offspring individuals are fitter than the parent, in which case the parent will eventually be replaced, or the parent fails to produce fitter offspring within a reasonable time span which means that it should be discarded anyhow. The selection mechanism is controlled by a replacement strategy that replaces the worst individual in the pool unless the reproduction capacity of the parent expires. In this case the parent is replaced.

Note that median selection was proposed for derandomized ESs while continuous selection was proposed for the general case. For use with derandomized ESs, the latter can easily be modified by omitting steps that aim at suppressing random fluctuations in the strategy parameters.

3 Experimental Setup

3.1 Parameters of the Evolution Strategy

In order to cope with the long computation times for design optimization tasks, usually small population sizes are chosen. Since the results of our comparison of the three selection methods discussed above shall carry over to this kind of application, we also used small population sizes for the test functions.

For the generational selection method, we used $\mu = 5$ parents and $\lambda = 20$ offspring. The free parameters of the steady-state selection were chosen as similar as possible to their equivalent in generational selection. Thus for median selection the parent population also consisted of $\mu = 5$ parents and for determining

the fitness threshold the last $n_m = \lambda = 20$ evaluated individuals were taken into account. The threshold rank was given by $\theta = n_m \frac{\mu}{\lambda} = 5$. In continuous selection the selection pool consisted of $n_c = \lambda = 20$ individuals. The reproduction capacity was $\gamma = \frac{\lambda}{\mu} = 4$. This is equal to the average number of times an individual reproduces in generational selection.

For all three selection algorithms we used a derandomized self-adaptation method, the covariance matrix adaptation [2], with default parameters, as recommended in [2]. The damping factor for the adaptation of the global step size was $d_\sigma = 1 + \frac{1-\frac{N}{T}}{c_\sigma} = 26.35$, for $T = 12000$ evaluations. The maximal condition value of the covariance matrix was 10^{14} and the minimal effective step size 10^{-15}.

The strategy parameters, i.e. the covariance matrix, the evolution path, and the global step size were associated with each individual. Therefore, every individual had an associated search distribution which was the result of the adaptation to the series of mutations that created that individual's genotype. We did not use any type of recombination operator in this study.

The objective variables of the initial population were initialized with random numbers uniformly distributed in the interval $[x_{min}, x_{max}]$ and the initial global step size was given by the values for σ in Tab. 1.

We conducted experiments with a large number of widely used test functions that are listed in Tab. 1. The test functions were evaluated in dimension $N = 100$. Note, that the optimization target was minimization.

Table 1. Test functions and initialization parameters.

Name	x_{min}	x_{max}	σ	Target	Formula
Sphere	-5.12	5.12	1	1e-10	$\sum_{i=1}^{N} x_i^2$
Ellipsoid	-5.12	5.12	1	1e-10	$\sum_{i=1}^{N} 10^{6\frac{i-1}{N-1}} x_i^2$
Cigar	-5.12	5.12	1	1e-10	$x_1^2 + \sum_{i=2}^{N} 10^6 x_i^2$
Tablet	-5.12	5.12	1	1e-10	$10^6 x_1^2 + \sum_{i=2}^{N} x_i^2$
Cigar-Tablet	-5.12	5.12	1	1e-10	$x_1^2 + \sum_{i=2}^{N-1} x_i^2 + 10^8 x_N^2$
Two axes	-5.12	5.12	1	1e-10	$\sum_{i=1}^{\lfloor N/2 \rfloor} 10^6 x_i^2 + \sum_{i=\lfloor N/2 \rfloor}^{N} x_i^2$
Weighted sphere	-5.12	5.12	1	1e-10	$\sum_{i=1}^{N} i x_i^2$
Different powers	-5.12	5.12	0.1	1e-15	$\sum_{i=1}^{N} \lvert x_i \rvert^{2+10\frac{i-1}{N-1}}$
Rosenbrock	-5.12	5.12	0.1	1e-10	$\sum_{i=1}^{N-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$
Rastrigin	-5.12	5.12	100	1e-10	$10N + \sum_{i=1}^{N} x_i^2 - 10\cos(\omega x_i)$
Parabolic ridge	0	0	1	-1e10	$-x_1 + 100 \sum_{i=2}^{N} x_i^2$
Sharp ridge	0	0	1	-1e10	$-x_1 + 100\sqrt{\sum_{i=2}^{N} x_i^2}$
Ackley	-32.7	32.7	1	1e-10	see below
Schwefel	-65.5	65.5	1	1e-10	$\sum_{i=1}^{N} \left(\sum_{j=1}^{i} x_j \right)^2$

Ackley function : $-a \cdot \exp\left(-b\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \right) - \exp\left(\frac{1}{N} \cdot \sum_{i=1}^{N} \cos(c x_i) \right)$

3.2 Evaluation Time and Computing Resources

The test functions require a very small and constant evaluation time. To study the effect of parallelism in a context closer to real world applications, we simulated two different types of variable evaluation times:

A. The evaluation time of an individual was considered to be a random variable drawn from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 90s$ and $\sigma = 10s$, where s denotes seconds. This setting corresponds to a situation with only small variations in the fitness function evaluation times.
B. The evaluation time of an individual was considered to be a random variable drawn from an exponential distribution $\mathcal{E}(x) = \lambda \exp(-\lambda x)$ with $\lambda = 90$. In this case the fitness function evaluation time varies largely.

In both sets of experiments we varied the degree of parallelization starting with only one processor up to the case where the number of processors was identical to λ, the number of offspring in the generational selection method.

3.3 Performance Measure

In order to achieve a fair comparison, we limited the duration T of every run so that the *computing power* $P = p \cdot T$ was constant for any number of processors p. This allows taking into account the differences in the computational efficiency of the ESs, see Fig. 1.

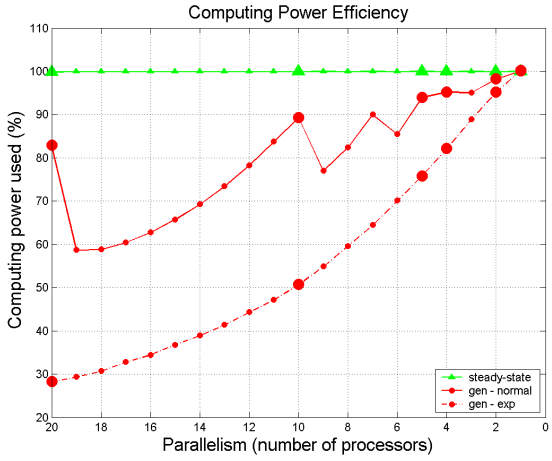

Fig. 1. Usage of the computing power by generational selection for normally (gen-normal) and exponentially (gen-exp) distributed evaluation times and by steady-state selection (steady-state). For steady-state selection the curve is independent from the time distribution. On the contrary generational selection is very sensitive to the distribution of the evaluation time. In any case efficiency is lower for generational selection.

The measure used to characterize the performance of an ES was the fitness value reached by using the given computing power. The simulations were performed for two values of the computing power, $P_1 = 100$ processor-hours and $P_2 = 300$ processor-hours. If we consider the above mentioned average of 90 seconds per fitness function evaluation, this is equivalent to a maximally efficient algorithm with 4000 and 12000 fitness function evaluations, respectively. However, the number of evaluations performed depends on the efficiency of the ES.

All performance values are averages $\bar{m}$ of the best fitness values over 20 independent runs. For a more complete image, the corresponding standard deviations $\bar{\sigma}$ are also given.

4 Results

As stated in Sec. 3.2, two sets of experiments were conducted, corresponding to normally and exponentially distributed evaluation times (set A and set B, respectively).

For the set A of experiments, represented by solid curves in Fig. 2, we observe that with one exception both steady-state methods outperformed the generational selection algorithm for all degrees of parallelization. The exception was the maximally efficient parallelization with generational selection, i.e., the number of processors was equal to the number of offspring, in our case $\lambda = 20$.

The set B of experiments is represented by dashed curves in Fig. 2. We see that for any degree of parallelization both steady-state methods performed similarly irrespective of the probability distribution of the evaluation times. Thus, whether there was a large or a small variation in evaluation times among individuals had no impact on the performance of steady-state methods. This was very different for the generational selection algorithm. We observe a sharp decrease in the performance for large variations (exponential distribution) in particular for the 20 processor case.

Fig. 2 shows the results of both sets of experiments for the special case of the Schwefel test function, cf. Tab. 1. However, for most of the other test functions the curves look qualitatively similar (not shown here). Moreover, similar results were obtained for different computing powers (Fig. 3).

The average relative performance of the steady state selection methods with respect to generational selection is shown for all test functions in Fig. 4 for the case of 20 processors and $P_2 = 300$ processor-hours.

For all test functions represented in the lower left quadrant of the plot steady-state selection showed better results than generational selection: both means and standard deviations were smaller than for generational selection. As expected, for small variations in the evaluation time generational selection outperformed steady-state selection methods (note that the case for $p = \lambda$ processors is shown), see Fig. 4 (right). However, for the exponentially distributed evaluation times, which result in large differences, the situation is reversed. On nearly all test

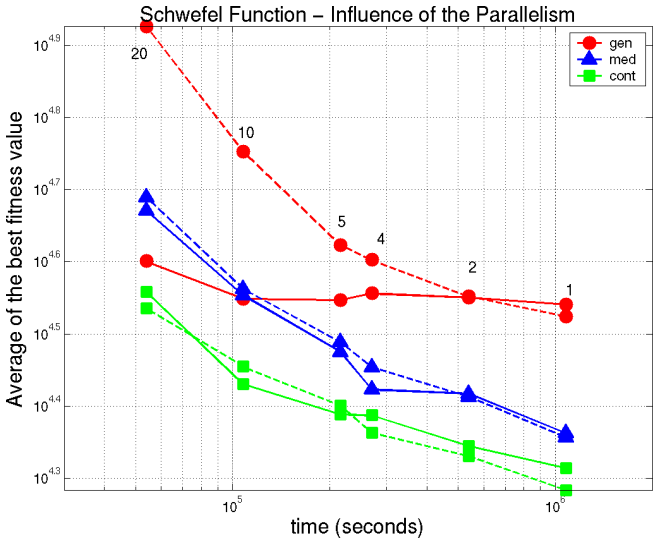

Fig. 2. Comparison of normally (solid curves) and exponentially (dashed curves) distributed evaluation times for the Schwefel function. The points on each curve correspond, from left to right, to 20, 10, 5, 4, 2 and 1 processor(s), respectively. Here gen/med/cont refer to generational/median/continuous selection.

functions steady-state selection shows better performance than generational selection, see Fig. 4 (left).

5 Real world case: 2D Blade Optimization

In order to analyze whether the results obtained for the various test functions carry over to real-world application problems, we apply the three selection methods to the two-dimensional geometry optimization of an outlet guide vane in the turbine stage of a small turbofan engine.

The contour of the two-dimensional cross-section of the vane is represented by a 3^{rd} B-spline curve, whose 22 control points are subject to optimization. Thus, we have to solve a 44 dimensional optimization problem.

The fitness values are determined by analyzing the fluid dynamic properties of the proposed vane geometries using a Navier-Stokes flow solver, see [7, 4] for details of the system set-up. The calculation time for a single fitness evaluation for this problem is of the order of 3-5 minutes on an Opteron 2GHz processor. Variations in the calculation time are mainly due to different starting conditions of the flow solver and also depend on the vane geometry.

The results are shown in Fig. 5. The differences between the evaluation times in this experiment were surprisingly small (roughly normally distributed with mean $\mu = 257s$ and standard deviation $\sigma = 42s$). As expected from the previous results on the test functions, we observe in Fig. 5 a behavior which is similar to

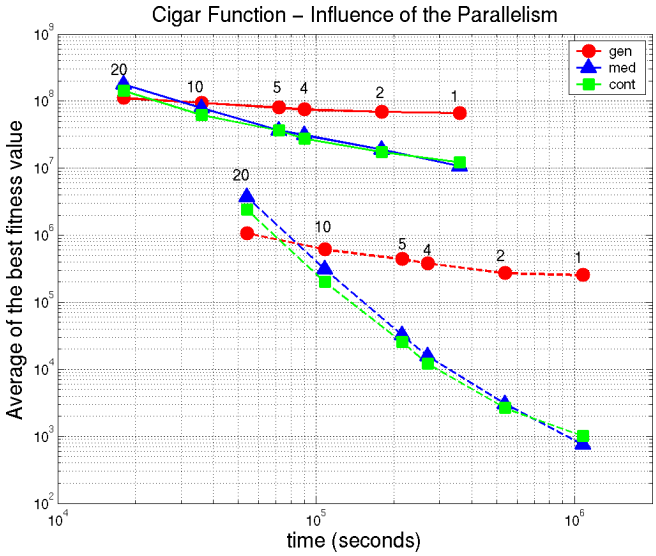

Fig. 3. The influence of parallelism on the cigar function. The lower (dashed) curves correspond to $P_2 = 300$ processor-hours and the upper (solid) curves correspond to $P_1 = 100$ processor-hours. The points on each curve correspond, from left to right, to 20, 10, 5, 4, 2 and 1 processor(s), respectively. Here gen/med/cont refer to generational/median/continuous selection.

the set A of experiments. Therefore, steady-state selection did not significantly improve the optimization. Indeed the curves for all three selection methods look very similar. In order to give a clear recommendation for steady-state selection for the design optimization application additional investigations are necessary. We will come back to this point in the last section.

6 Conclusions

This paper is concerned with the performance evaluation and the comparison of three different types of selection methods for evolution strategies. Two steady-state methods have been analyzed and compared to the standard generational selection algorithm for different distributions of the evaluation time and for degrees of parallelism ranging from a single processor to the maximally efficient parallelization for generational selection, i.e. with $p = \lambda$ processors.

From the two sets of experiments – one with normally and one with exponentially distributed evaluation times – we observe that steady-state selection methods outperform generational selection in all but one case. This case, which we termed "maximally efficient" parallelization, is realized if the number of processors equals the number of individuals and if the evaluation times are fairly similar between individuals. As we argued already in the introduction, these two criteria will not always be met in applications. Therefore, steady-state selection

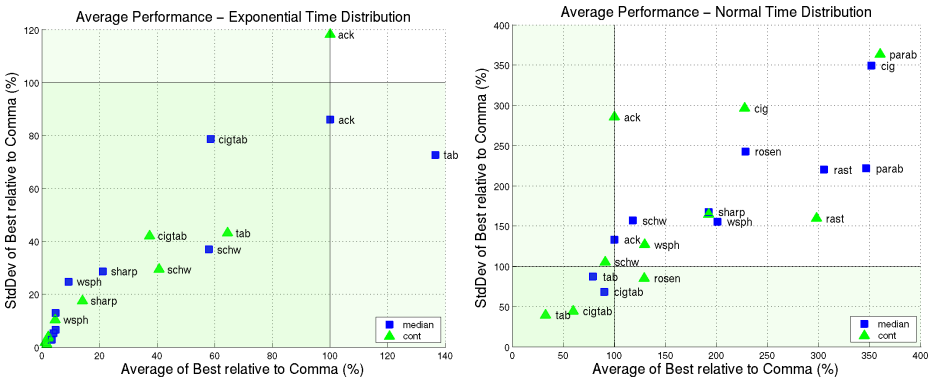

Fig. 4. Average performance of steady state selection relative to generational selection for $p = \lambda = 20$ processors for an exponential time distribution (left) and a normal time distribution (right). The plot shows the statistics $\bar{m}_{ss}$ and $\bar{\sigma}_{ss}$ of the steady-state methods as a percentage of the corresponding statistics $\bar{m}_c$ and $\bar{\sigma}_c$ for the generational selection, i.e., the points have the coordinates $\frac{\bar{\sigma}_{ss}}{\bar{\sigma}_c}$ and $\frac{\bar{m}_{ss}}{\bar{m}_c}$.

methods present a more flexible alternative to generational selection. Furthermore, they are more efficient and in most cases also show higher performance.

Unfortunately, the first results for the design optimization task in a "real-world" optimization environment, do not provide a picture as clear as for the test functions. The main reason is that the variations of the evaluation time between individuals were smaller than what we recorded for previous design optimization tasks. Therefore, we can only conclude that steady-state selection methods perform similarly as generational selection. In order to verify their advantage, additional experiments are necessary. We are currently using our meta-model assisted design optimization environment [3] as a test bed. Furthermore, we plan to run additional experiments for a smaller number of processors.

Acknowledgments We would like to thank E. Körner for his support.

References

1. H.-G. Beyer and H.-P. Schwefel. Evolution strategies. *Natural Computing*, 1:3–52, 2002.
2. N. Hansen and A. Ostermeier. Completely derandomized self-adapatation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
3. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.
4. M. Olhofer, T. Arima, T. Sonoda, M. Fischer, and B. Sendhoff. Aerodynamic shape optimisation using evolution strategies. In *Optimisation in Industry III*, pages 83–94. Springer, 2001.
5. I. Rechenberg. *Evolutionstrategie '94*. Frommann-Holzboog, Stuttgart, 1994.

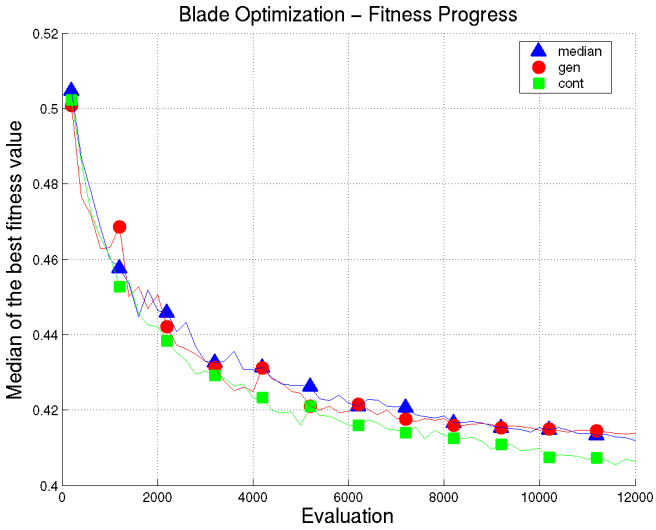

Fig. 5. Average performance of steady state selection relative to generational selection on the blade optimization problem. The median of the best fitness value is plotted against the evaluation number for median, generational and continuous selection.

6. T. Runarsson and X. Yao. Continuous selection and self-adaptive evolution strategies. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC)*, pages 279–284. IEEE Press, 2002.
7. T. Sonoda, Y. Yamaguchi, T. Arima, M. Olhofer, B. Sendhoff, and H. A. Schreiber. Advanced high turning compressor airfoils for low reynolds number condition, Part 1: Design and optimization. *Journal of Turbomachinery*, 2004. In press.
8. J. Wakunda and A. Zell. Median-selection for parallel steady-state evolution strategies. In *Proceedings of the 6th International Conference Parallel Problem Solving from Nature (PPSN VI)*, pages 405–414. Springer, 2000.
9. J. Wakunda and A. Zell. A new selection scheme for steady-state evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 794–801. Morgan Kaufmann, 2000.