Microchannel Optimization Using Multiobjective Evolution Strategies

Ivo F. Sbalzarini, Sibylle Müller, and Petros Koumoutsakos

Institute of Computational Sciences, ETH Zürich, Switzerland,
{sbalzarini,muellers,petros}@inf.ethz.ch

Abstract. We implement multiobjective evolutionary algorithms for the optimization of micro-fluidic devices. In this work we discuss the development of multimembered evolution strategies with step size adaptation in conjunction with the Strength Pareto Approach. In order to support targeting, an extension of the Strength Pareto Evolutionary Algorithm is proposed. The results suggest a novel design for micro-fluidic devices used for DNA sequencing.

1 Introduction

Evolutionary Algorithms (EAs) such as Evolution Strategies or Genetic Algorithms have become *the* method of choice for optimization problems that are too complex to be solved using deterministic techniques such as linear programming or gradient (Jacobian) methods. The enormous number of applications ([Beasley (1997)]) and the still growing interest in this field are due to several advantages of EAs compared to gradient based methods for complex problems. EAs require little knowledge about the problem being solved, they are easy to implement, robust and inherently parallel. To solve a certain optimization problem, it is enough to require that one is able to evaluate the objective (cost) function for a given set of input parameters. The property of parallelism becomes more and more important with the increasing power and availability of large parallel computer systems. Because of their universality, ease of implementation and fitness for parallel computing, EAs often take less time to find the optimal solution than gradient methods. However, most real-world problems involve simultaneous optimization of several, often mutually concurrent objectives. Multiobjective EAs are able to find optimal trade-offs in order to get a set of solutions that are optimal in an overall sense. In multiobjective optimization, gradient based methods are often impossible to apply. Multiobjective EAs however can always be applied and they inherit all of the favorable properties from their single objective relatives.

Section 2 of this paper introduces main concepts of single objective EAs. Section 3 extends these ideas to multiobjective cases and introduces the principles of dominance and Pareto optimality. Section 4 describes the Strength Pareto

Approach used in this work and in section 5 we extend it with a targeting capability. In section 6 the results of both single and multiobjective optimization of a microchannel flow are shown and discussed.

2 Single objective Evolutionary Algorithms

The basic idea for single objective EAs is to imitate the natural process of biological evolution. The problem to be solved is therefore described using a certain number of *parameters* (design variables). One then creates a group of $\lambda (> 0)$ different parameter vectors and considers it as a *population of individuals*. The quantity λ is called the *population size*. The quality of a certain vector of parameters (i.e. an *individual* in the population) is expressed in terms of a scalar valued *fitness function* (objective function). Depending on whether one wants to minimize or maximize the objective function, individuals (i.e. parameter vectors) with lower or greater fitness are considered better, respectively. The algorithm then proceeds to choose the $\mu, (\mu < \lambda)$ best individuals out of the population to become the parents of the next generation (natural *selection*, survival of the fittest). Therefore, μ denotes the *number of parents*. The smaller μ is chosen compared to λ, the higher the *selection pressure* will be. Out of the μ individuals chosen to be parents for the next generation, one then creates a new population of λ offspring $\mathbf{x}_i^{g+1}$ by applying *mutation* on the parents $\mathbf{x}_j^g$ as follows:

$$\mathbf{x}_i^{g+1} = \mathbf{x}_j^g + \mathcal{N}(0, \Sigma) \ , i = 1, \ldots, \lambda \ , j \in \{1, \ldots, \mu\} \tag{1}$$

where $\mathcal{N}(0, \Sigma)$ denotes a vector of jointly distributed Gaussian random numbers with zero mean and covariance matrix Σ. The standard deviations (i.e. the square roots of the diagonal elements σ_i^2 of Σ) of the additive random numbers determine "how far away from its parent a child will be" and are called *step sizes* of the mutation. Now, the first iteration is completed and the algorithm loops back to the evaluation of the fitness function for the new individuals. Several different techniques for adaptation and control of the step size have been developed (see e.g. [Bäck (1997a)], [Bäck (1997b)], [Bäck (1993)], [Hansen & Ostermeier (1996)] or [Hansen & Ostermeier (1997)]). In the following subsections, some of the single objective Evolution Strategies used in this work are outlined.

2.1 The (1+1)-ES

One of the simplest and yet powerful evolution strategies is the "one plus one evolution strategy", denoted by (1+1)-ES. In this strategy, both the number of parents and the population size (i.e. number of offspring) are set to one: $\mu = \lambda = 1$. Mutation is accomplished by adding a vector of usually uncorrelated Gaussian random numbers, i.e. $\Sigma = diag(\sigma_i^2)$ is a diagonal matrix. Step size adaptation can be performed according to Rechenberg's 1/5-rule: if less than 20% of the generations are successful (i.e. offspring better than parent), then decrease the step size for the next generation; if more than 20% are successful, then increase the step size in order to accelerate convergence. This adaptation is done every $N \cdot L_R$ generations where N is the number of parameters (i.e. dimension of search space) and L_R is a constant, usually equal to one. Selection is done out of the set union of parent and offspring, i.e. the better one of the two is chosen to become the parent of the next generation.

2.2 The (μ, λ)-ES

A slightly more advanced method is to take one or more parents and even more offspring, i.e. $\mu \geq 1$ and $\lambda > \mu$. Mutation is accomplished in a sihyper-ellipsoidmilar way as with the (1+1)-ES. Besides the 1/5 rule, another method for step size adaptation becomes available which is called *self-adaptive mutation* ([Bäck (1997a)]). In this method, the mutation steps are adapted every generation. They are either increased, decreased or kept the same, each with a probability of 1/3. On the average, 1/3 of the offspring will now be closer to their parents than before, 1/3 keeps progressing at the same speed and 1/3 explores further areas. Depending on how far away from the optimum we currently are, one of these three groups will do better than the others and therefore, more individuals out of it will be selected to the next generation, where their step sizes are inherited. The algorithm adapts the step size by itself, i.e. by means of mutation and selection.

2.3 The $(\mu/\mu_I, \lambda)$-CMA-ES

The Covariance Matrix Adaptation is a sophisticated method for online adaptation of step sizes in (μ, λ)-ES with intermediate recombination (i.e. averaging of parents). It was first described by [Hansen & Ostermeier (1996)] and further improved and evaluated by [Hansen & Ostermeier (1997)]. For a complete description of the algorithm, the reader is referred to the latter publication. The basic idea is to adapt step sizes and covariances in such a way, that the longest axis of the of mutation distribution always aligns in the direction of greatest estimated progress. This is done by accumulating information about former mutation steps and their success (*evolution path*) and searching it for correlations. Besides this very sophisticated method for step size adaptation, a CMA-ES also includes mutation (with Σ now being a full matrix) and selection.

3 Multiobjective Evolutionary Algorithms

As soon as there are many (possibly conflicting) objectives to be optimized simultaneously, there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality. Consider for example the design of an automobile. Possible objectives could be: minimize cost, maximize speed, minimize fuel consumption and maximize luxury. These goals are clearly conflicting and therefore there is no single optimum to be found. Multiobjective EAs can yield a whole set of potential solutions - which are all optimal in some sense - and give the engineers the option to assess the trade-offs between different designs. One then could for example choose to create three different cars according to different marketing needs: a slow low-cost model which consumes least fuel, an intermediate solution and a luxury sports car where speed is clearly the primer objective. Evolutionary Algorithms are well suited to multiobjective optimization problems as they are fundamentally based on biological processes, which are inherently multiobjective.

After the first pioneering work on multiobjective evolutionary optimization in the eighties ([Schaffer (1984)], [Schaffer (1985)]), several different algorithms have been proposed and successfully applied to various problems. For comprehensive overviews and discussions, the reader is referred to [Fonseca & Fleming (1995)], [Horn (1997)], [Van Veldhuizen & Lamont (1998)] and [Coello (1999)].

3.1 Dominance and Pareto-Optimality

In contrast to fully ordered scalar search spaces, multidimensional search spaces are only partially ordered, i.e. two different solutions are related to each other in two possible ways: either one dominates the other or none of them is dominated.

DEFINITION 1: *Consider without loss of generality the following multiobjective optimization problem with m decision variables x (parameters) and n objectives y:*

$$\begin{aligned} &\text{Maximize } \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m)) \\ &\text{where} \quad \mathbf{x} = (x_1, \dots, x_m) \in X \\ &\qquad\qquad \mathbf{y} = (y_1, \dots, y_n) \in Y \end{aligned} \tag{2}$$

and where $\mathbf{x}$ *is called* decision (parameter) vector*,* X parameter space*,* $\mathbf{y}$ objective vector *and* Y objective space*. A decision vector* $\mathbf{a} \in X$ *is said to* dominate *a decision vector* $\mathbf{b} \in X$ *(also written as* $\mathbf{a} \succ \mathbf{b}$*) if and only if:*

$$\begin{aligned} \forall i \in \{1, \dots, n\} &: f_i(\mathbf{a}) \geq f_i(\mathbf{b}) \\ \wedge\ \exists j \in \{1, \dots, n\} &: f_j(\mathbf{a}) > f_j(\mathbf{b}) \end{aligned} \tag{3}$$

Additionally, we say $\mathbf{a}$ covers $\mathbf{b}$ $(\mathbf{a} \succeq \mathbf{b})$ *if and only if* $\mathbf{a} \succ \mathbf{b}$ *or* $\mathbf{f}(\mathbf{a}) = \mathbf{f}(\mathbf{b})$*.*

Based on this convention, we can define nondominated, *Pareto-optimal* solutions as follows:

DEFINITION 2: *Let* $\mathbf{a} \in X$ *be an arbitrary decision (parameter) vector.*

1. *The decision vector* $\mathbf{a}$ *is said to be* nondominated *regarding a set* $X' \subseteq X$ *if and only if there is no vector in* X' *which dominates* $\mathbf{a}$*; formally:*

$$\nexists \mathbf{a}' \in X' : \mathbf{a}' \succ \mathbf{a} \tag{4}$$

2. *The decision (parameter) vector* $\mathbf{a}$ *is called* Pareto-optimal *if and only if* $\mathbf{a}$ *is nondominated regarding the whole parameter space* X.

If the set X' is not explicitly specified, the whole parameter space X is implied.

Pareto-optimal parameter vectors cannot be improved in any objective without causing a degradation in at least one of the other objectives. They represent in that sense *globally optimal* solutions. Note that a Pareto-optimal set does not necessarily contain all Pareto-optimal solutions in X. The set of objective vectors $\mathbf{f}(\mathbf{a}'), \mathbf{a}' \in X'$, corresponding to a set of Pareto-optimal parameter vectors $\mathbf{a}' \in X'$ is called *"Pareto-optimal front"* or *"Pareto-front"*.

3.2 Difficulties in multiobjectve optimization

In extending the ideas of single objective EAs to multiobjective cases, two major problems must be addressed:

1. How to accomplish fitness assignment and selection in order to guide the search towards the Pareto-optimal set.
2. How to maintain a diverse population in order to prevent premature convergence and achieve a well distributed, wide spread trade-off front.

Note that the objective function itself no longer qualifies as fitness function since it is vector valued and fitness has to be a scalar value. Different approaches to relate the fitness function to the objective function can be classified with regard to the first issue. For further information, the reader is referred to [Horn (1997)]. The second problem is usually solved by introducing elitism and intermediate recombination. *Elitism* is a way to ensure that good individuals do not get lost (by mutation or set reduction), simply by storing them away in a external set, which only participates in selection. *Intermediate recombination* on the other hand averages the parameter vectors of two parents in order to generate one offspring according to:

$$\begin{array}{lll} \mathbf{x'}_j = \alpha \mathbf{x}^g_{j1} + (1-\alpha)\mathbf{x}^g_{j2} & , j, j_1, j_2 \in \{1, \dots, \mu\} & \\ \mathbf{x}^{g+1}_i = \mathbf{x'}_j + \mathcal{N}(0, \Sigma) & , i = 1, \dots, \lambda & , j \in \{1, \dots, \mu\} \end{array} \tag{5}$$

Arithmetic recombination is a special case of intermediate recombination where $\alpha = 0.5$.

4 The Strength Pareto Approach

For this work, the *Strength Pareto Approach* for multiobjective optimization has been used. Comparative studies have shown for a large number of test cases that, among all major multiobjective EAs, the *Strength Pareto Evolutionary Algorithm*

(SPEA) is clearly superior ([Zitzler & Thiele (1999)] [Zitzler, Thiele & Deb (2000)]). It is based on the above mentioned principles of Pareto-optimality and dominance. The algorithm as proposed by [Zitzler & Thiele (1999)] was implemented in a restartable, fully parallel code as follows:

Step 1: Generate random initial population P and create the empty external set of nondominated individuals P'.
Step 2: Evaluate objective function for each individual in P in parallel.
Step 3: Copy nondominated members of P to P'.
Step 4: Remove solutions within P' which are covered by any other member of P'.
Step 5: If the number of externally stored nondominated solutions exceeds a given maximum N', prune P' by means of clustering.
Step 6: Calculate the fitness of each individual in P as well as in P'.
Step 7: Select individuals from $P + P'$ (multiset union), until the mating pool is filled.
Step 8: Adapt step sizes of the members of the mating pool.
Step 9: Apply recombination and mutation to members of the mating pool in order to create a new population P
Step 10: If maximum number of generations is reached, then stop, else go to Step 2.

4.1 Fitness assignment

In Step 6, all individuals in P and P' are assigned a scalar fitness value. This is accomplished in the following two-stage process. First, all members of the nondominated set P' are ranked. Afterwards, the individuals in the population P are assigned their fitness value.

Step 1: Each solution $i \in P'$ is assigned a real value $s_i \in [0, 1)$, called *strength.* s_i is proportional to the number of population members $j \in P$ for which $i \succeq j$. Let n denote the number of individuals in P that are covered by i and assume N to be the size of P. Then s_i is defined as: $s_i = \frac{n}{N+1}$. The fitness f_i of i is equal to its strength: $f_i = s_i \in [0, 1)$.
Step 2: The fitness of an individual $j \in P$ is calculated by summing the strengths of all external nondominated solutions $i \in P'$ that cover j. Add one to this sum to guarantee that members of P' always have better fitness than members of P (note that the fitness is to be minimized):

$$f_i = 1 + \sum_{i, i \succeq j} s_i \ , f_i \in [1, N) \tag{6}$$

4.2 Selection and step size adaptation

Step 7 requires an algorithm for the selection of individuals into the mating pool and Step 8 includes some method for dynamical adaptation of step sizes (i.e. mutation variances). For this paper, selection was done using the following binary tournament procedure:

Step 1: Randomly (uniformly distributed random numbers) select two individuals out of the population P.
Step 2: Copy the one with the better (i.e. lower for SPEA) fitness value to the mating pool.
Step 3: If the mating pool is full, then stop, else go to Step 1

Adaptation of the step sizes was done using the self-adaptive mutation method (c.f. section 2.3). Each element of P and P' is assigned an individual step size for every parameter, i.e. $\Sigma = diag(\sigma_i^2)$ is a diagonal matrix for each individual. The step sizes of all members of the mating pool are then either increased by 50%, cut to half or kept the same, each at a probability of 1/3.

4.3 Reduction by Clustering

In Step 5, the number of externally stored nondominated solutions is limited to some number N'. This is necessary because otherwise, P' would grow to infinity since there always is an infinite number of points along the Pareto-front. Moreover, one wants to be able to control the number of proposed possible solutions, because from a decision maker's point of view, a few points along the front are often enough. A third reason for introducing clustering is the distribution of solutions along the Pareto-front. In order to explore as much of the front as possible, the nondominated members of P' should be equally distributed along the Pareto-front. Without clustering, the fitness assignment method would probably be biased towards a certain region of the search space, leading to an unbalanced distribution of the solutions. For this work, the *average linkage method*, a clustering algorithm which has proven to perform well on Pareto optimization, has been chosen. The reader is referred to [Morse (1980)] or [Zitzler & Thiele (1999)] for details.

5 Strength Pareto Approach with Targeting

Compared to other methods like for example the Energy Minimization Evolutionary Algorithm (EMEA) (c.f. [Jonathan, Zebulum, Pacheco & Vellasco (2000)]), the SPEA has two major advantages: it finds the whole Pareto-front and not just a single point on it and it converges faster. The latter is a universal advantage, whereas the former is not. There are applications where a target value can be specified. One then wants to find the point on the Pareto-front which is closest to the user-specified target (in objective space). This eliminates the need to analyze all the points found by SPEA in order to make a decision. EMEA offers such a possibility, but it converges slower than SPEA and it is unable to find more than one point per run. Hence we wish to extend SPEA with some targeting facility that can be switched on and off depending on whether one is looking for a single solution or the whole front, respectively. We added this capability to SPEA by the following changes to the algorithm:

1. Between Step 6 and Step 7 the fitness values of all individuals in P and P' are scaled by the distance D of the individual from the target (in objective space) to some power q:
$$f_i = f_i \cdot D_i^q$$
This ensures that enough nondominated members close to the target will be found so that the one with minimal distance will appear at higher probability. The parameter q determines the sharpness of the concentration around the target.
2. Another external storage P_{best} is added, which always contains the individual out of P' which is closest to the target. Therefore, between steps 4 and 5, the algorithm calculates the distances of all members of P' to the target and picks the one with minimal distance into P_{best}. At all times, P_{best} only contains one solution.
3. At the end of the algorithm, not only the Pareto-front is output, but also the solution stored in P_{best}. Note that due to clustering and removal in P', the solution in P_{best} is not necessarily contained in P'. It is therefore an optimal solution which otherwise would not have appeared in the output.

The algorithm has been implemented and tested for convex and nonconvex test functions. Figures 1 to 4 show some results for the nonconvex test function $\mathcal{T}_2$ as proposed in [Zitzler, Thiele & Deb (2000)]:

$$\begin{array}{ll} \text{Minimize} & \mathcal{T}_2(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x})) \\ \text{subject to} & f_2(\mathbf{x}) = g(x_2, \ldots, x_m) h(f_1(x_1), g(x_2, \ldots, x_m)) \\ \text{where} & \mathbf{x} = (x_1, \ldots, x_m) \\ & f_1(x_1) = x_1 \\ & g(x_2, \ldots, x_m) = 1 + 9 \cdot \sum_{i=2}^m x_i/(m-1) \\ & h(f_1, g) = 1 - (f_1/g)^2 \end{array} \tag{7}$$

where m is the dimension of the parameter space and $x_i \in [0,1]$. The exact Pareto-optimal front is given by $g(\mathbf{x}) = 1$. The parameters of the algorithm were set as summarized in table 1.

The chosen target value is slightly off-front. Therefore, the targeting error will never be zero. Figure 1 shows the final population after 250 generations without targeting. The diamonds indicate members of the external nondominated set (Pareto-optimal front), whereas members of the regular population are denoted by crosses. In figure 2 the same run has been repeated with targeting. Figure 3

Parameter	Value
Dimension of parameter space (m)	5
Size of population (λ)	50
Size of mating pool (μ)	30
Size of nondominated set (N')	70
Number of generations	250
Target value for (f_1, f_2)	(0.5, 0.7)
Concentration parameter q	4

Table 1. Settings for targeting SPEA

shows the targeting error as a function of the generation number. The dashed line indicates the theoretical minimum of the distance. After about 80 to 100 generations, the point on the front which is closest to the target has been found with good accuracy. Figure 4 shows the path of P_{best} towards the target. The jumps are due to the fact, that the individual stored in P_{best} gets replaced as soon as another individual is closer to the target.

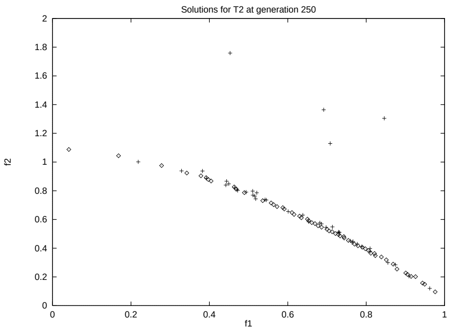

Fig. 1. Final population without targeting

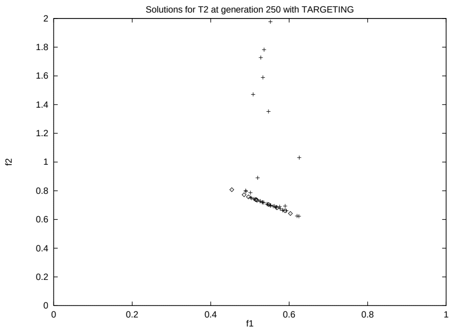

Fig. 2. Final population with targeting

The best objective value that was achieved was: $\mathbf{f}(P_{best}) = (0.5265, 0.7247)$, its Euclidean distance from the target is $3.6287 \cdot 10^{-2}$, which is equal to the theoretical minimal distance within the given computational accuracy.

6 Microchannel flow optimization

Both single and multiobjective EAs have been applied to a fluidic microchannel design problem. Bio-analytical applications require long thin channels for DNA sequencing by means of electrophoresis. In order to pack a channel of several meters in length onto a small square plate, curved geometries are required. However, curved channels introduce dispersion and therefore limit the separation efficiency of the system. The question is now how to shape the contour of

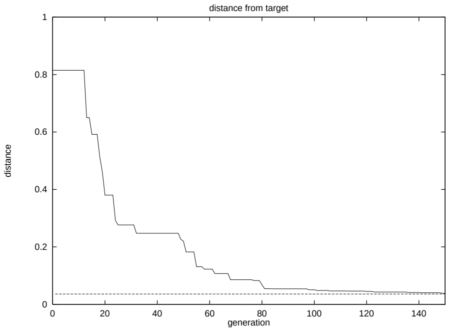

Fig. 3. Distance between P_{best} and target

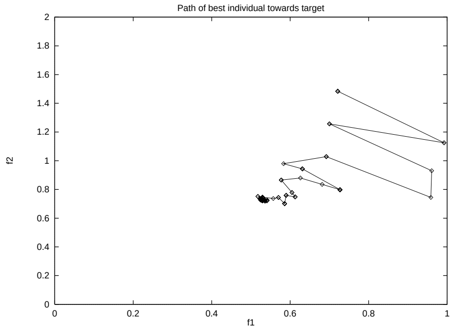

Fig. 4. Path of P_{best} towards the target

the channel in order to minimize dispersion. A detailed description of the problem as well as an optimization solution using gradient methods can be found in [Mohammadi et al. (2000)].

6.1 Single objective optimization

The goal of this optimization run was to minimize the final skewness of the flow inside the channel, i.e. it was required that the iso-values of the advected species a be normal to the flow field U by time T when they exit the channel. The objective function defined by [Mohammadi et al. (2000)] is therefore:

$$J = \int_{\Omega} \left(\nabla a(x, T) \times U(x)\right)^2 \, dx \tag{8}$$

with Ω being the cross section of the channel exit. The shape of the 90 degrees turn is described by 11 parameters. Therefore, the parameter search space is of dimension 11. The objective space is scalar since it is a single objective problem.

The calculation of the flow field and evaluation of the objective function was done by an external flow solver provided by [Mohammadi et al. (2000)]. Both a (1+1)-ES and a $(3/3_I,12)$-CMA-ES were applied to the problem and their convergence was compared. The results were statistically averaged from 5 runs with different initial conditions, i.e. starting points.

Since the CMA-ES has a population size of 12, it performs 12 function evaluations per generation. Figure 5 shows the convergence normalized to the same number of function calls. Figures 6 and 7 show the corresponding solutions after 20 and 180 generations of the best 1+1 run out of the ensemble (the lines are iso-potential lines of the electric field). After 20 generations the contour of the channel gets a clearly visible dent in it. After 80 evaluations of the objective function, the algorithm has found a double-bump shape to be even better and after 180 calls to the solver, no further significant improvement is observed. The value of the objective function has dropped to about 10^{-6} for the best run out of the ensemble. This means, that dispersion is almost zero and the channel will have very good separation properties.

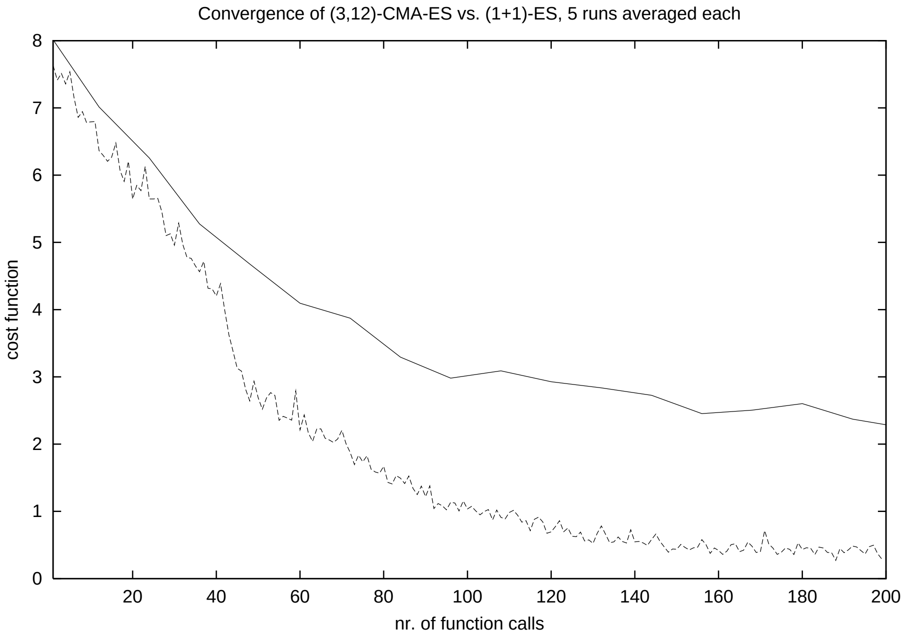

Fig. 5. Convergence of (3,12)-CMA-ES [solid line] and (1+1)-ES [dashed line] vs. number of evaluations of the objective function.

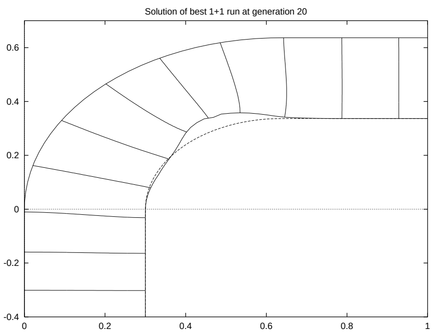

Fig. 6. Solution at generation 20

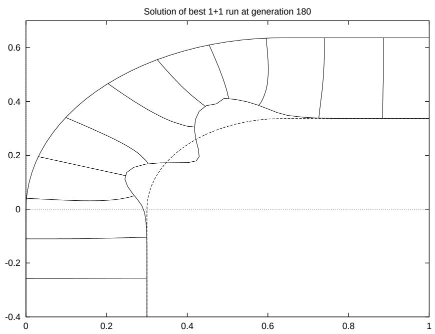

Fig. 7. Solution at generation 180

6.2 Multiobjective optimization

We then introduced the total deformation of the channel contour as a second objective to be minimized simultaneously in order to minimize manufacturing costs. The second objective thus reads:

$$K = \sum_{i=1}^{11} p_i^2 \tag{9}$$

where p_i are the shape parameters of the channel as introduced by [Mohammadi et al. (2000)]. The first objective remained unchanged. The algorithm used for this optimization was a SPEA with a population size of 20, a maximum size of the external nondominated set of 30 and a mating pool of size 10.

Figure 8 shows the Pareto-optimal trade-off front after 80 generations of the algorithm and figures 9 and 10 show the corresponding solutions, i.e. optimized shapes of the channel. One is now free to choose whether to go for minimal skewness at the expense of a higher deformation (c.f. figure 9), choose some intermediate result or minimize deformation in order to minimize manufacturing costs and still get the lowest skewness possible with the given amount of deformation (c.f. figure 10).

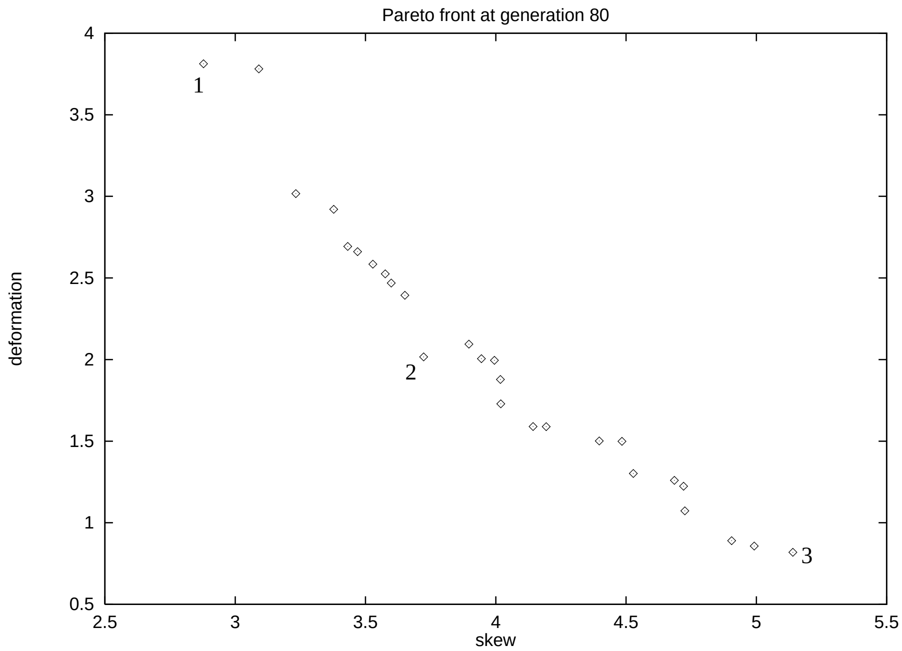

Fig. 8. Pareto-front of nondominated solutions after 80 generations..

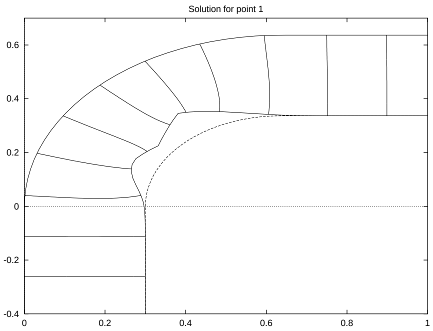

Fig. 9. Solution at point 1

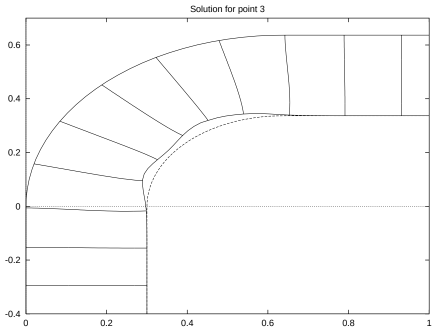

Fig. 10. Solution at point 3

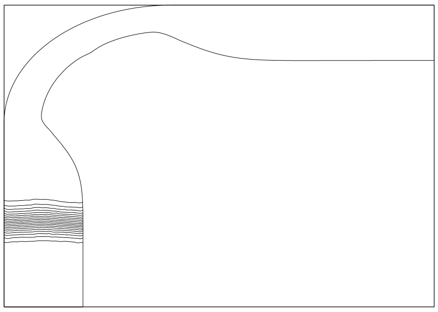

Fig. 11. First optimized shape using gradient methods

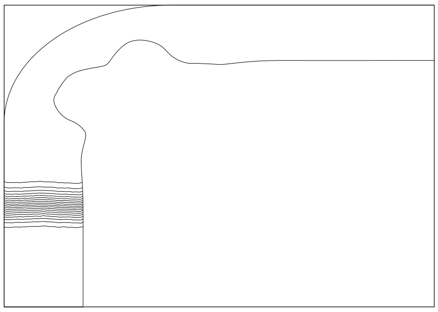

Fig. 12. Second optimized shape using gradient methods

6.3 Comparison with gradient based methods

Figures 11 and 12 show two classes of optimized shapes obtained by [Mohammadi et al. (2000)] using gradient methods. It is interesting that the gradient technique offers the same two designs, namely the single-dented (fig. 11) and the double-dented (fig. 12) shapes, which we found with the evolution strategy after 40 or 180 generations, respectively. Therefore, we obtain qualitatively similar results from both methods. Using the gradient method, the skewness is reduced by one order of magnitude [Mohammadi et al. (2000)] which is comparable to the numbers obtained by evolutionary optimization. While trial and error procedures were used in the gradient methods to obtain various solutions, evolution strategies provide us with a number of solutions (Pareto front) in a fully automated fashion. Unlike the gradient based methods which require an explicit formulation of the optimization problem in hand, the evolution strategy provides a straightforward optimization procedure. Moreover, the small cost of computation implies that evolution strategies are a reliable method leading to greater flexibility and shorter "time-to-solution".

7 Conclusions and future work

Single and multiobjective evolutionary algorithms have been implemented and assessed. The SPEA has successfully been extended to support targeting in objective space. It has been shown that these algorithms are easy to apply to fluid dynamical problems and that their solutions are comparable to those found by gradient based methods. In cases where gradient methods cannot be applied or where they would involve too complex mathematical calculations, evolution strategies are a good alternative to solve an optimization problem or reduce the time needed to do so as part of hybrid processes.

Future and present work addresses the acceleration of convergence of these algorithms and their implementation in hybrid processes.

References

[Bäck (1993)] Bäck, T. 1993 Optimal mutation rates in genetic search, In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, IL, July 1993*, Morgan Kaufmann, San Mateo, CA, pp. 2-8.

[Bäck (1997a)] Bäck, T. 1997a Self-adaptation, In Bäck, T., Fogel, D. B. and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, **97/1**, C7.1, IOP Publishing Ltd. and Oxford University Press.

[Bäck (1997b)] Bäck, T. 1997b Mutation parameters, In Bäck, T., Fogel, D. B. and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, **97/1**, E1.2, IOP Publishing Ltd. and Oxford University Press.

[Beasley (1997)] Beasley, D. 1997 Possible applications of evolutionary computation, In Bäck, T., Fogel, D. B. and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, **97/1**, A1.2, IOP Publishing Ltd. and Oxford University Press.

[Coello (1999)] Coello, C. A. C. 1999 A comprehensive survey of evolutionary-based multiobjective optimization, *Knowledge and Information Systems*, **1(3)**, 269-308.

[Fonseca & Fleming (1995)] Fonseca, C. M. & Fleming, P. J. 1995 An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*, **3(1)**, 1-16.

[Hansen & Ostermeier (1996)] Hansen, N. & Ostermeier, A. 1996 Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC '96)*, 312-317.

[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.

[Horn (1997)] Horn J. 1997 Multicriteria decision making, In Bäck, T., Fogel, D. B. and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, **97/1**, F1.9, IOP Publishing Ltd. and Oxford University Press.

[Jonathan, Zebulum, Pacheco & Vellasco (2000)] JONATHAN, M., ZEBULUM, R. S., PACHECO, M. A. C., VELLASCO, M. B. R. 2000 Multiobjective Optimization Techniques: A Study Of The Energy Minimization Method And Its Application To The Synthesis Of Ota Amplifiers, *Proceedings of the second NASA/DoD Workshop on Evolvable Hardware*, July 13-15, 2000, Palo Alto, CA, 133-140.

[Mohammadi et al. (2000)] MOHAMMADI, B., MOLHO, J. I. & SANTIAGO J. A. 2000 Design of Minimal Dispersion Fluidic Channels in a CAD-Free Framework, *Center for Turbulence Research Annual Research Briefs 2000.*

[Morse (1980)] MORSE, J. N. 1980 Reducing the size of the nondominated set: Pruning by clustering, *Computers and Operations Research*, **7 (1-2)**, 55-66.

[Schaffer (1984)] SCHAFFER, J. D. 1984 Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, Unpublished Ph.D. thesis, Vanderbilt University, Nashville, Tennessee.

[Schaffer (1985)] SCHAFFER, J. D. 1985 Multiple objective optimization with vector evaluated genetic Algorithms, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, sponsored by Texas Instruments and the U.S. Navy Center for Applied Research in Artificial Intelligence (NCARAI), 93-100.

[Van Veldhuizen & Lamont (1998)] VAN VELDHUIZEN, D. A. & LAMONT, G. B. 1998 Multiobjective evolutionary algorithm research: A history and analysis, Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

[Zitzler & Thiele (1999)] ZITZLER, E. & THIELE, L. Nov. 1999 Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach *IEEE Transactions on Evolutionary Computation*, **3(4)**, 257-271.

[Zitzler, Thiele & Deb (2000)] ZITZLER, E., DEB. K. & THIELE L. 2000 Comparison of Multiobjective Evolutionary Algorithms: Empirical Results *Evolutionary Computation*, **8(2)**, 173-195.