Morphing Methods in Evolutionary Design Optimization

Michael Nashvili
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, United Kingdom
msc83@cs.bham.ac.uk

Markus Olhofer
Honda Research Institute Europe GmbH
Carl-Legien-Str. 30
Offenbach, Germany
markus.olhofer@honda-ri.de

Bernhard Sendhoff
Honda Research Institute Europe GmbH
Carl-Legien-Str. 30
Offenbach, Germany
bs@honda-ri.de

ABSTRACT

Design optimization is a well established application field of evolutionary computation. However, standard recombination operators acting on the genotypic representation of the design or shape are often too disruptive to be useful during optimization. In this work, we will analyze whether morphing methods between two shapes can be used as recombination operators acting on the phenotype space, thus directly on the shape or design. We introduce three different morphing methods and employ them as recombination operators in a standard evolution strategy (es). We compare their performance with an evolution strategy without any recombination operators on two target shape approximation problem. We can conclude that two of the three morphing methods can be useful during search although all morphing methods still turn out to hinder the self-adaptation of the step sizes of the evolution strategy.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; J.2 [**Physical sciences and engineering**]: Engineering

General Terms

Design, Algorithms

Keywords

Design optimization, evolution strategies, morphing methods, phenotypic recombination

1. INTRODUCTION

Design optimization is a well established application field of evolutionary computation. In particular, the optimization of aerodynamic structures, like aircraft wings [8] or turbine blades [14] using evolutionary algorithms has been successful. Although it is often treated as a parameter optimization problem, there is a clear difference between the genotype and the phenotype level in design optimization. The genotype is e.g. given by the coordinates of the control points if a spline representation is chosen. The phenotype is the actual two-dimensional or three-dimensional design or structure that can be represented by a sufficiently large set of sample points. It is evident that the topology of the genotype space is very different from that of the phenotype space.

In particular, in conjunction with evolution strategies the use of recombination operators for design or shape optimization has been controversial and in some works recombination has even been omitted [9] altogether. At the same time, on the phenotype level, i.e., on the level of the actual shapes, we have a very intuitive idea of what a shape should like that lies in between two shapes as long as the two initial shapes do not differ too much. This "intuitive idea" is captured by morphing methods which provide a smooth transition between two images, shapes or three-dimensional objects. Therefore, it seems a reasonable idea to employ morphing methods as phenotypic recombination operators in design optimization with evolution strategies. The analysis of such a shape optimization framework is the subject of this paper. The idea to employ crossover or recombination methods on phenotype space or more generally to use phenotypic information to determine optimal crossover points has been successfully used before, e.g., in the context of probability density models [7] and of neural networks [3].

In the next section, the evolutionary design optimization task will be introduced and the basic evolution strategy will be outlined. In Section 3 three different morphing methods that have been suggested in the literature for the problem of polygon shape morphing are presented. These morphing methods are used in Section 4 as recombination operators in a simple evolution strategy. Their performance will be compared with each other and with an evolution strategy without recombination for a target-spline benchmark problem [4] that is close to "real" design optimization problems. We will summarize our findings and conclude in the last section.

2. EVOLUTIONARY DESIGN OPTIMIZATION

Since the necessary computation time prohibits the use of aerodynamic shape optimization problems for testing new algorithms or operators, we use target shape optimization as a benchmark problem in this study. We define two target shape problems: a dolphin and the shape of a two-dimensional cross-section of a turbine blade. Obviously, the turbine blade shape is close to the design of the real-world problem whereas the dolphin is visually appealing and – more seriously – due to the fins it is structurally different from the blade. Both shapes (light gray curves) are shown in Figure 1.

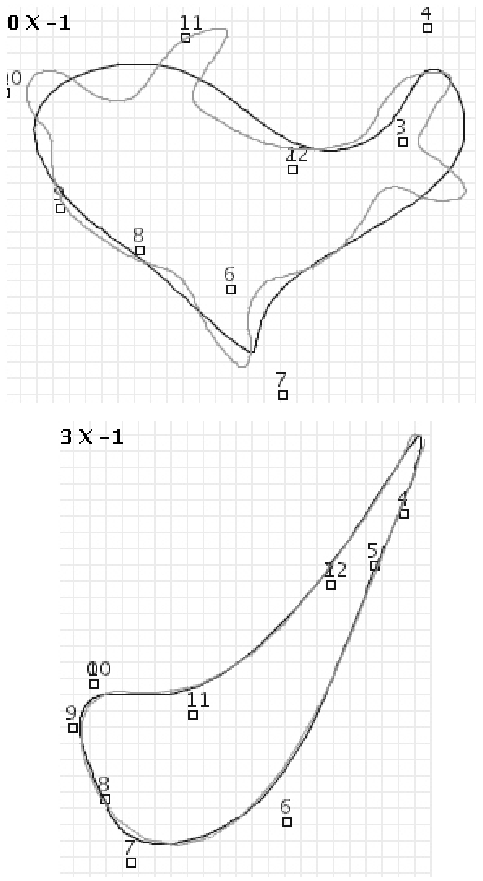

Figure 1: The target shapes (light gray curves) and the best approximations (dark gray curves) after 10000 function evaluations for the dolphin (top) and the blade (bottom) target shapes.

The goal of the optimization is to approximate the target shape as close as possible. The fitness measure is a symmetrised Hausdorff distance measure $\mathcal{H}$ between two sets of sample points. One set from the target shape $\mathcal{T} = \{t_i\}$ and one set from the individual whose fitness is calculated $\mathcal{A} = \{a_i\}$. Both sets have the same cardinality $|\mathcal{T}| = |\mathcal{A}| = n_F$. Note that each point $a_i = (x_i^a, y_i^a)$ and $t_i = (x_i^t, y_i^t)$ consists of two coordinates.

$$\mathcal{H}(\mathcal{A},\mathcal{T}) = \frac{1}{2}\left(\sum_{i=1}^{n_F} min\{|a_i - t|\,; t \in \mathcal{T}\} + \sum_{i=1}^{n_F} min\{|t_i - a|\,; a \in \mathcal{A}\}\right) \qquad (1)$$

Each shape is represented as a closed B-spline [10, 11]. The B-spline is defined by its degree, set of control points and knot points. Here, we only vary the control points. Therefore, the genotype of each individual consists of a chromosome with the (x, y) coordinates of all control points.

For the optimization we employ a standard evolution strategy with global mutative step-size adaptation. For each individual during mutation a normally distributed random number is added to the coordinates of the control points of the spline representation. The isodensity contours of the normal distribution are hyper-spheres, i.e. only one strategy parameter, the variance of the normal distribution, is adapted during the search process. The reason to use only one step size is that due to the disruptive nature of some of the recombination operators, a more stable adaptation scheme is preferred. The evolution strategy is used together with (μ,λ)-selection.

3. SHAPE MORPHING

Shape morphing is the process of deforming a source object or shape into a target through a sequence of intermediate forms. Resembling the source at the start and the target at the end of the process, the intermediate forms should be gradual blends from source to target. Although there is no precise definition of what the intermediate forms should look like, there are several common requirements that have been suggested. Carmel and Cohen-Or [2] summarize these into the following:

1. The volume and circumference of the objects should change monotonically
2. The boundary of the objects should retain the smoothness of the original objects
3. Features common to both source and target objects (e.g. head or legs), should be preserved during the process

There are several different classifications of morphing algorithms depending on the actual form of the source and target objects. Techniques have been developed to morph images, polygons and volumetric forms. For the 2D design optimization problems analyzed in this paper, we will use the polygon morphing algorithms that will be discussed in the following. For 3D problems volumetric morphing algorithms might also be applicable. The majority of shape morphing algorithms operate on a sequence of vertexes that define the object. There are only few approaches that directly use the B-spline representations of two shapes, see [13, 5].

In this paper, we discuss three morphing methods as phenotypic recombination operators. They all operate on a sequence of n_R sample points which we obtain from the spline representation of the shapes. The sample points are the vertexes of the polygon morphing algorithms.

3.1 Linear interpolation method

Given two polygons; each represented by a set of vertexes $\mathcal{A}$ and $\mathcal{B}$ with cardinality $|\mathcal{A}| = |\mathcal{B}| = n_R$, an obvious method of performing a shape blend or morph is to linearly interpolate the two sets of vertexes into a new set $\mathcal{C}$:

$$\begin{aligned} \mathcal{C}(t) &= u\,\mathcal{A} + t\,\mathcal{B} \\ &= \{u\,a_i + t\,b_i \mid a_i \in \mathcal{A};\ b_i \in \mathcal{B}\} \qquad (2) \\ &= \{c_i\}, \\ u &= 1 - t;\ \ |\mathcal{C}| = n_R. \end{aligned}$$

Note that again, each vertex (or point) consists of two coordinates $c_i = (x_i^c, y_i^c)$.

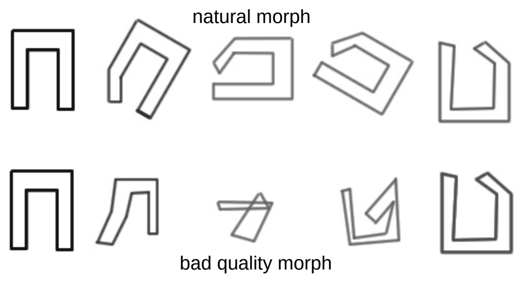

Figure 2: Linear vertex interpolation can lead to unsatisfactory intermediate shapes.

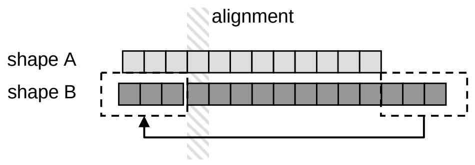

Figure 3: Correspondence between two phenotypic shape descriptions A and B and subsequent re-alignment.

This naive method however will typically produce poor results. The arbitrary correspondence between vertexes of the same index number, as shown above, will not produce desirable results in many cases. Furthermore, each vertex is linearly interpolating by the same amount. Essentially, the linear vertex interpolation method fails because it treats vertexes independently from each other. One example of morphing between two simple shapes in this way is shown in Figure 2. Most research therefore decomposes the polygon morphing problem into that of vertex correspondence and vertex interpolation. Here we use the simple interpolation suggested in equation (2) together with two different methods to align the vertexes of the two shapes.

First simply examine all sample points or vertexes on both shapes and select the two that have the minimum Euclidean distance. Using the correspondence between these two points all subsequent points in both sequences are aligned, see Figure 3. This method will be referred to as *morphing_initial* in the remainder of the paper.

A second solution to the correspondence problem is to define a cost function and to minimise it over different possible vertex correspondences, see [16]:

$$cost(i,j) = w_1\,|angle(a_i) - angle(b_j)| + w_2\,\frac{1}{n_R}|i-j|,$$

which involves the following definition of the angle cost at point a_i:

$$angle(a_i) = \frac{1}{2}\arccos(a_{i-1}, a_{i+1})\cdot\mathrm{sign}(x^a_{i-1}y^a_{i+1} - x^a_{i+1}y^a_{i-1}), \tag{3}$$

where w_1 is the *angle cost*, and w_2 is the *index cost*.[1]

The function attempts to match parts of each curve that have similar angles. The second part of the function ensures that vertexes that are close together (in terms of their indexes) correspond with each other. The minimum of the cost function is then found over all possible vertex correspondences of the two curves using the method of dynamic programming. We will refer to this method in the following as *morphing_match*.

3.2 String representation morphing

The string representation morphing algorithm [6] attempts to morph polygonal shapes by decomposing the shapes into strings and then computing the weighted mean (or linear interpolants) of the strings based on a recent algorithm by [1]. This involves using the Levenshtein algorithm [15] for computing the distance between two strings. The main steps are outlined as follows:

1. **Calculation of starting points**. The starting point is calculated in the same way as for *morphing_initial*.

2. **Curve sampling**. It is necessary to produce two sets of samples $\mathcal{A}$ and $\mathcal{B}$ from the individual and the target curves such that the distance between any two consecutive points has a fixed euclidean distance $\Delta = |a_i - a_{i-1}|$. This differs from the majority of morphing algorithms that accept polygons whereby the vertex distribution is arbitrary, but it is an intrinsic feature of this method. Most polygons can be re-sampled accurately when Δ is sufficiently small.

3. **String decomposition**. Once a sequence of uniform samples is obtained, a "string" representation for each curve is generated simply by taking the sequence of vectors from each point to the next point in the curve sequence.

4. **Levenshtein algorithm [15]**. This algorithm computes the similarity, or distance, between two strings by considering the optimal sequence of edit operations that are required in order to transform one string into the other. The edit operations are: deletion ($a \to \epsilon$), insertion ($\epsilon \to a$), and substitution ($a \to b$). These are defined with the following costs: $c(a \to \epsilon) = c(\epsilon \to a) = |a| = \Delta, c(a \to b) = |a - b|$. The substitution cost is therefore bounded by 0 and 2 Δ. The Levenshtein algorithm is implemented by means of dynamic programming and returns the minimal cost of edit operations that can transform one string, $\mathcal{A}$, into another, $\mathcal{B}$. This process also extracts the explicit sequence of optimal edit operations, S, which is known as the *edit transcript*.

5. **Weighted mean of strings [1]**. In order to generate intermediate forms of the two strings (which correspond to intermediate shapes) a subsequence S' of S of the edit operations can then be applied.

In the following, we will refer to this method as *morphing_string*.

3.3 Phenotype Decoding Method

The three shape morphing methods introduced above all result in intermediate phenotypes represented by n_R sample points. Therefore, it is necessary to reversely transcribe the sample point sequence into a B-spline. Significantly this

[1]Typical values of $w_1 = 3$, and $w_2 = 1$ are given in [16].

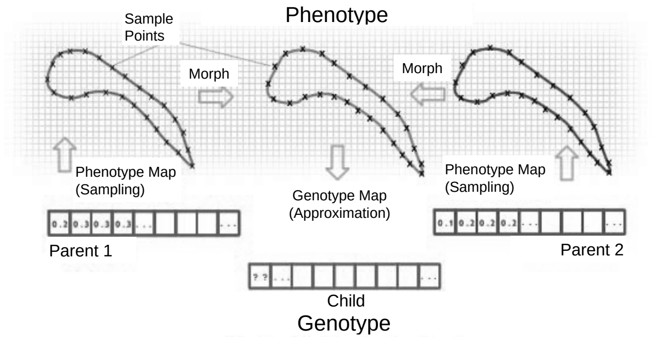

Figure 4: Flow of the phenotypic recombination process in an evolutionary algorithm.

process also modifies the result of the morphing on the phenotypic level as the B-spline it returns has a different representation in phenotype space than the sample points returned by the morphing. This mapping from phenotype to genotype is a non-trivial step in any evolutionary algorithm particularly where complex phenotypic operators are used. The intermediate genotypic representation of the intermediate shape should differ from the genotypic representation of its parents in the same ratio that the corresponding phenotypic representations differ. This strong causality constraint is essential in theory in order to ensure that the self-adaptation of the mutation in evolution strategy (es) that operates *on the genotypic level* - as the primary variation operator - is not disrupted. Fortunately, various methods exist for producing B-splines of a fixed number of control points, n, from a set of m sample points, commonly where $n > m$. The standard method of least-squares approximation is used here. A schematic overview of the steps in performing phenotypic based recombination is shown in Figure 4.

4. RESULTS

A visual comparison of the three morphing methods is shown in Figure 5.

4.1 Experimental set-up

All parameters of the experiments and the evolution strategy (es) are summarized in Table 1. The relatively small choice of the offspring population size is motivated by the computationally expensive practical optimizations. The ratio $\frac{\mu}{\lambda}$ was recommended in [12], the standard values for τ and τ' were used [12].

All results that we present in the following are the average of ten runs. The first generation was initialized by adding normally distributed random numbers with variance 0.02 to the control points of the spline representation of the initial shapes. The initial shapes of a typical first generation are shown in Figure 6 together with the dolphin target shape problem.

4.2 Blade Target Shape

The development of the fitness value, the symmetrised Hausdorff distance, with the number of function calls is shown in Figure 7 for the four recombination operators. The methods *morphing_initial*, *morphing_match* and *morphing_string* are represented by a solid line, a dashed line and a dash-dotted line, respectively. The fitness for the evo-

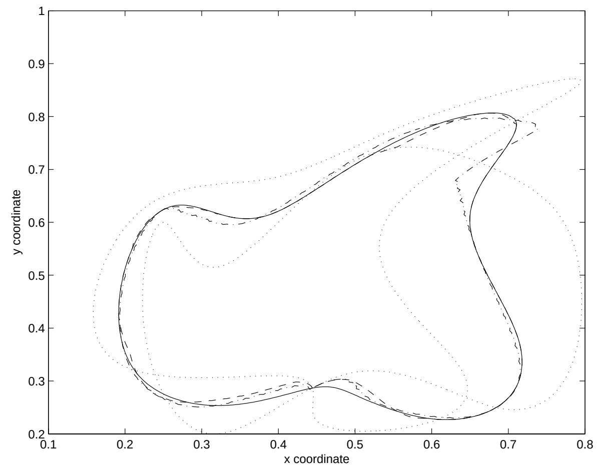

Figure 5: An example of the result of the three morphing methods. The two parent shapes are shown as dotted curves. The results of *morphing-initial*, *morphing-match* and *morphing-string* recombination operators are shown as solid, dashed and dash-dotted curves, respectively.

parameter	value
(μ, λ)	(5,35)
σ_{init}	$\sigma_{\text{init}} \in [10^{-3}, 10^{-4}]$
lower bound for σ	10^{-8}
fitness sample size (blade) n_F	66
fitness sample size (dolphin) n_F	500
phenotype sample size n_R	600
string distance Δ	0.005
number of control points	10
degree of the spline	3

Table 1: Parameter setting for all experiments.

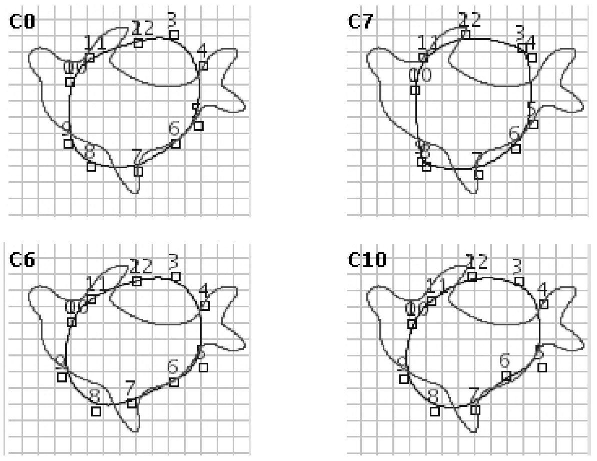

Figure 6: Four examples for the shapes from the first generation (dark gray) for the dolphin problem. Initial shapes are the same for the blade problem.

lution strategy (es) without any recombination operator is shown as a dotted line. Note that the 10.000 function calls roughly correspond to 285 generations in our experimental set-up. First we notice that the es without recombination achieves the best fitness values during the first 2000 function calls. It is "faster" than all other methods. The second observation is that *morphing_string* performs worst during the whole optimization. It converges to a relatively poor fitness value.

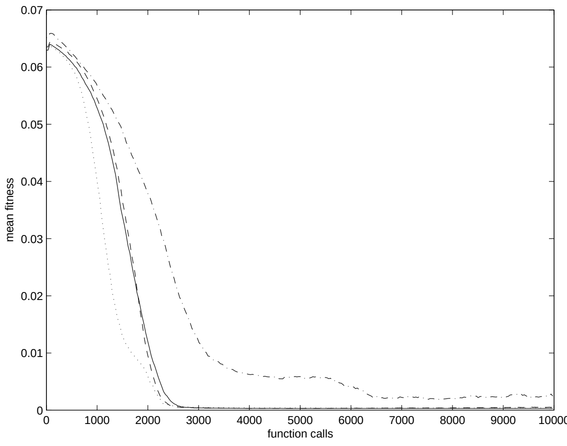

Figure 7: The Hausdorff distance (fitness) versus the function evaluations for the blade target shape problem. The es with the *morphing-initial*, *morphing-match* and *morphing-string* recombination operators are shown as solid, dashed and dash-dotted curves, respectively. The es without recombination is shown as a dotted curve.

A more detailed analysis, see Figure 8, reveals that both methods *morphing_match* and *morphing_initial* overtake the es without recombination after 2400 and 2600 function evaluations. Both methods reach about the same fitness value after 3000 evaluations. Although the difference between the morphing methods and the es without recombination seems relatively small, it is statistically significant (student t-test with $\alpha = 0.01$).

Strangely, *morphing_match* starts to diverge at later generations, as can be seen from Figure 9. The advantage of *morphing_initial* remains statistically significant for the final evaluations (significance level $\alpha = 0.001$ at 10.000 function calls).

The developments of the global step sizes for all methods (shown in Figure 10) reveal why two of the three morphing methods are not successful. Bearing in mind that the mutation operator has a very strong impact on the behavior of the es, the correlation between the performance of each algorithm and the development of the global step size is not surprising. The step size of *morphing_string* does not converge at all, it fluctuates around a relatively high level of 0.025. For the *morphing_match* method the typical adaptation pattern of the strategy parameter can be observed, however, at later generations it still converges to a value which seems to be too high. This relatively high level of σ for function calls larger than 5000, is likely to be responsible for the divergent behavior that we observed in Figure 9. Furthermore, the lower optimization "speed" of all morph-

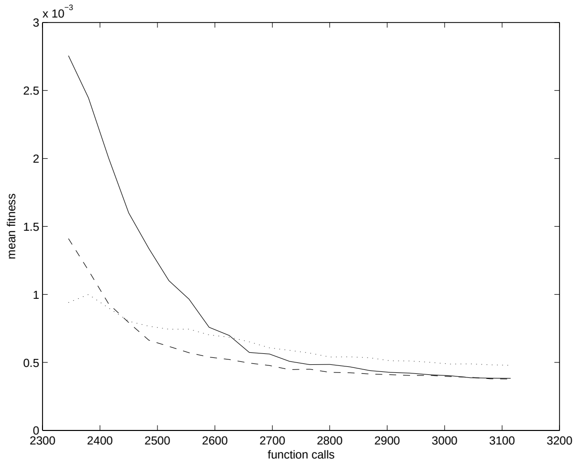

Figure 8: The fitness values between evaluations 2300 and 3100 for the blade target shape problem. The difference between *morphing_initial/match* and the es without recombination is significant (student t-test at $\alpha = 0.01$) at 3100 evaluations.

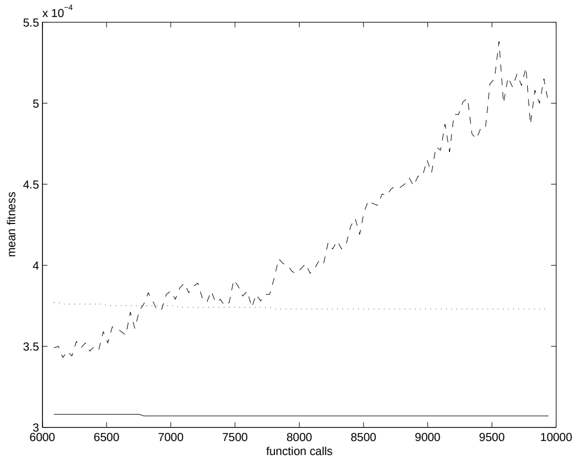

Figure 9: The fitness values for the last 4000 function evaluations for the blade target shape problem. The difference between *morphing_initial* and the es without recombination is significant (student t-test at $\alpha = 0.001$). The divergence of *morphing_match* is clearly visible.

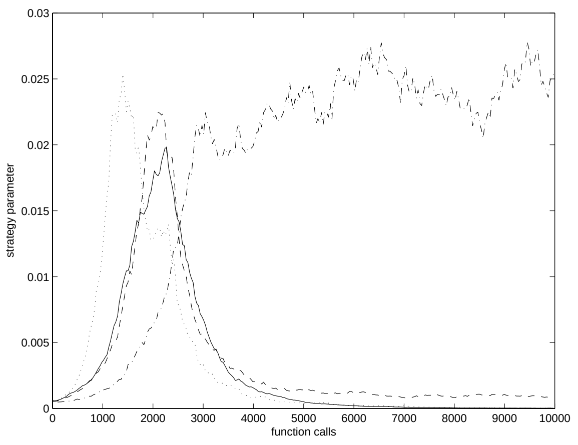

Figure 10: The development of the standard deviations of the mutation operators for all algorithms for the blade target shape problem.

ing methods can be attributed to the delayed adaptation of the strategy parameters that is also visible in Figure 10.

Finally, the resulting best approximation (dark gray curve) is shown together with the target shape (light gray curve) in Figure 1 (bottom).

4.3 Dolphin Target Shape

The behavior of the four algorithms is similar for the dolphin target shape problem. In Figure 11 the overall development of the fitness value is shown. The correspondence between algorithm and curve pattern is the same as for the blade target shape problem. We notice again that the evolution strategy (es) without recombination performs best during early generations, i.e. the first 5000 function evaluations. The *morphing-string* method performs worst during the whole optimization process and fails to converge to a satisfactory fitness value.

The first 40 generations (600 evaluations) seem to be an exception. As Figure 12 shows, here paradoxically *morphing-string* performs best. However, statistical analysis reveals that this difference is not significant (student t-test) and must be due to statistical outliers.

Similar to the blade shape problem, the *morphing-initial* method performs better than the es without recombination for evaluations larger than 8500. Indeed this performance difference is statistically significant (student t-test) with a level of $\alpha = 0.001$. However, the performance of *morphing-match* stays below the one of the es without recombination.

In Figure 14 the developments of the step sizes for all four different methods are shown. We observe a very similar pattern as for the blade shape problem. The standard deviation of the mutation operator does not adapt well when used together with the *morphing-string* recombination operator. The adaptation of the step sizes of the *morphing-match* and the *morphing-initial* methods is delayed compared to the one of the es without recombination. At the same time, it seems that the step sizes of these two morphing methods have not converged yet. Thus, one could imagine that both methods

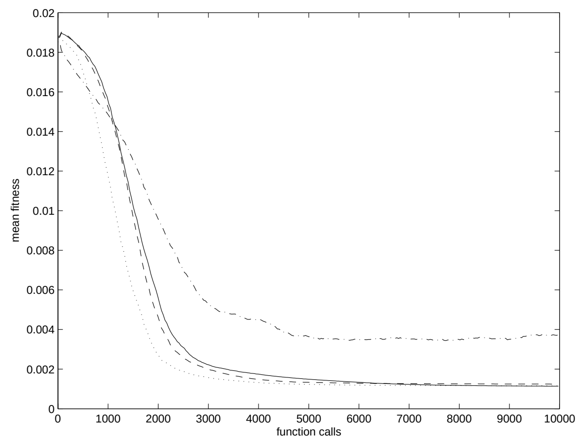

Figure 11: The Hausdorff distance (fitness) versus the function evaluations for the dolphin target shape problem. The es with the *morphing-initial*, *morphing-match* and *morphing-string* recombination operators are shown as solid, dashed and dash-dotted curves, respectively. The es without recombination is shown as a dotted curve.

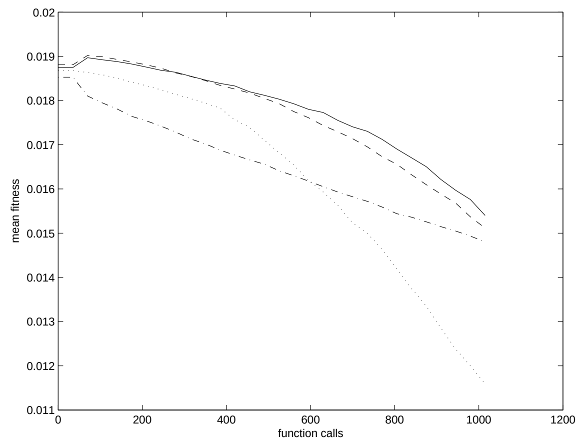

Figure 12: The fitness values for the first 1000 function evaluations for the dolphin target shape problem. The differences between the methods are not statistically significant (student t-test).

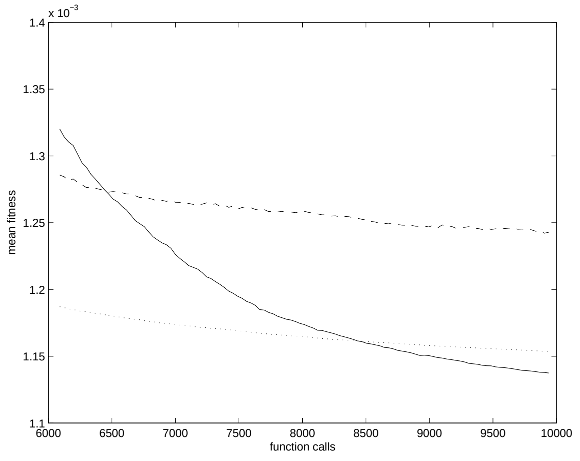

Figure 13: The fitness values between 6000 and 10000 function evaluations for the dolphin target shape problem. The difference between *morphing_initial* and the es without recombination is significant at 10000 function evaluations (student t-test at $\alpha = 0.001$).

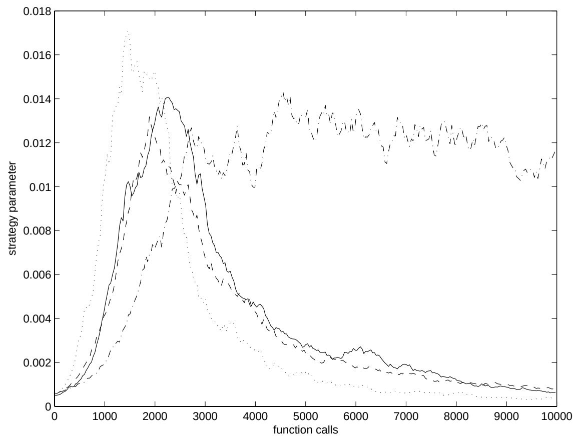

Figure 14: The development of the standard deviations of the mutation operators for all algorithms for the dolphin target shape problem.

would achieve slightly better results if more function evaluations were performed. Whether *morphing-match* would start to diverge again at some point would require further analysis.

Finally, the resulting best approximation (dark gray curve) is shown together with the target shape (light gray curve) in Figure 1 (left). One should not be mislead by the poor impression of the approximation quality. With the low number of ten control points that we fixed in this study, the dolphin cannot be approximated better. At the same time, increasing the number of control points increases the probability of local loops in the spline approximation. Local loops are difficult to detect and even more difficult to get rid off during optimization. Since loops would obscure the results, we decided to settle for a relatively low number of control points while being aware that the absolute approximation quality would be low for the dolphin.

5. SUMMARY AND CONCLUSION

In this paper, we analyzed for the first time the applicability of polygon shape morphing methods as recombination operators in a standard evolution strategy (es) for design optimization. In order to be able to achieve the number of runs that are needed to make any statistically sound statements, we used target shape design as a benchmark problem.

The results showed that two of the three morphing methods can indeed enhance the performance of the evolution strategy. This performance difference is statistically significant, however, not as decisive as one would have hoped for.

Indeed the advantage of the final approximation quality comes at the price of "slower" initial performance, i.e., the algorithms with morphing recombination operators take more evaluations to reach the same approximation quality than the es without recombination. As our analysis of the strategy parameters indicate, these performance problems are a result of the delayed (or in the case of *morphing_string* not visible) self-adaptation. In evolution strategies the mutation operator together with the self-adaptation property are the major driving forces of optimization. A recombination operator that disturbs this process cannot demonstrate its advantage even if the combination of different shapes is otherwise fruitful for the search. In particular, the problem of the reverse transcription of the phenotypic result of the recombination operator adds to the "disturbance", since control points will have changed their positions. In addition to the problems identified in this paper, the reverse transcription would basically prohibit the use of individual step-sizes or of even more complicated methods like the covariance matrix adaptation.

The particularly poor performance of *morphing_string* cannot be fully explained by the results of this work. However, a visual inspection of some of the recombination results seems to indicate that the morphing method itself can lead to strange results.

In this work, we concentrated on phenotypic morphing methods as recombination operators for evolution strategies. Standard n-point genotypic recombination operators do not work well for this problem with evolution strategies. Intermediate recombination on the genotype level results in performance values similar to the ones discussed in this paper (results for both methods are not shown here).

In the future, we plan to analyze whether a stable self-adaptation property can be realized even when morphing recombination operators are used. Furthermore, we will analyze morphing methods that directly act on the B-spline representation. The use of morphing recombination with genetic algorithms would also constitute an interesting subject of research.

6. REFERENCES

[1] H. Bunke, X. Jiang, K. Abegglen, and A. Kandel. On the weighted mean of a pair of strings. *Pattern Analysis and Applications*, 5:23–30, 2002.

[2] E. Carmel and D. Cohen-Or. Warp-guided object-space morphing. *The Visual Computer*, 13:465–178, 1997.

[3] B. Carse and T. Fogarty. Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm. In T. Fogarty, editor, *Evolutionary Computing – Selected papers from AISB*, volume 1143 of *Lecture Notes in Computer Science*, pages 1–22. Springer Verlag, 1996.

[4] W.-W. Chang, C.-J. Chung, and B. Sendhoff. Target shape design optimization with evolutionary computation. In *Congress on Evolutionary Computation (CEC)*, volume 3, pages 1864–1870. IEEE Press, 2003.

[5] S. Cohen-Or, G. Elber, and R. Bar-Yehuda. Matching of freeform curves. *Computer Aided Design*, 29:369–378, 1997.

[6] X. Jiang, H. Bunke, K. Abegglen, and A. Kandel. Curve morphing by weighted mean of strings. In *Proceedings of the Sixteenth Conference on Pattern Recognition (ICPR 2002)*, volume 4, pages 192–195. IEEE Press, 2002.

[7] M. Kreutz, A. M. Reimetz, B. Sendhoff, C. Weihs, and W. von Seelen. Optimisation of density estimation models with evolutionary algorithms. In A. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Computer Science 1498, pages 998–1007. Springer, 1998.

[8] S. Obayashi, Y. Yamaguchi, and T. Nakamura. Multiobjective genetic algorithm for multidisciplinary design of transonic wing planform. *Journal of Aircraft*, 34(5):690–693, 1997.

[9] M. Olhofer, T. Arima, T. Sonoda, M. Fischer, and B. Sendhoff. Aerodynamic shape optimisation using evolution strategies. In I. Parmee and P. Hajela, editors, *Optimisation in Industry*, pages 83–94. Springer Verlag, Berlin, 2002.

[10] M. Olhofer, Y. Jin, and B. Sendhoff. Adaptive encoding for aerodynamic shape optimization using evolution strategies. In *Congress on Evolutionary Computation (CEC)*, volume 2, pages 576–583, Seoul, Korea, May 2001.

[11] L. Piegl and W. Tiller. *The Nurbs Book*. Springer Verlag, 1997.

[12] H. Schwefel. *Evolution and Optimum Seeking*. John Wiley & sons, New York, 1995.

[13] T. Sederberg and E. Greenwood. Shape blending of 2-d piecewise curves. In M. Daehlen, T. Lyche, and L. Schumaker, editors, *Mathematical Methods in CAGD*, volume 3, pages 1–3. Vanderbilt University Press, 1995.

[14] 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*, 126:350–359, 2004.

[15] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21:168–173, 1974.

[16] H. Yohan, Y. Koiso, and T. Nishita. Morphing using curve and shape interpolation techniques. In B. Barsky, Y. Shinagawa, and W. Wenping, editors, *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications (Pacific Graphics '00)*, pages 348–358. IEEE Press, 2000.