On Improving Efficiency of Differential Evolution for Aerodynamic Shape Optimization Applications

Nateri K. Madavan*
NASA Ames Research Center, Moffett Field, CA 95014

Differential Evolution (DE) is a simple and robust evolutionary strategy that has been proven effective in determining the global optimum for several difficult optimization problems. Although DE offers several advantages over traditional optimization approaches, its use in applications such as aerodynamic shape optimization where the objective function evaluations are computationally expensive is limited by the large number of function evaluations often required. In this paper various approaches for improving the efficiency of DE are reviewed and discussed. These approaches are implemented in a DE-based aerodynamic shape optimization method that uses a Navier-Stokes solver for the objective function evaluations. Parallelization techniques on distributed computers are used to reduce turnaround times. Results are presented for the inverse design of a turbine airfoil. The efficiency improvements achieved by the different approaches are evaluated and compared.

Nomenclature

P	Population of real-valued parameter vectors
F	Differential Evolution amplification (or mutation) parameter
G	Additional Differential Evolution amplification (or mutation) parameter
p_c	Differential Evolution recombination (or crossover) parameter
f	Objective function
m	Population size
n	Number of design variables or parameters
g	Generation counter
i, l	Loop indices for design parameters and population members, respectively
x, y, z	n -dimensional parameter vector representing one of m members of population
$\alpha, \beta, \gamma, \eta, \nu$	Random numbers in $\{1, 2, \ldots, m\}$

I. Introduction

Aerodynamic shape optimization refers to the process of determining the shapes of airfoils, wings, or other aerodynamic surfaces that are optimal with regard to one or several desired characteristics. Major advances in the field of aerodynamic shape optimization have been achieved in recent years by combining improved methods for the simulation of complicated flow fields with efficient numerical optimization techniques and by exploiting the powerful capabilities of modern computers. Both Euler and high-fidelity Navier-Stokes solvers have been combined with various traditional optimization techniques (gradient-based methods, response surfaces, etc.) as well

* Research Scientist, NASA Advanced Supercomputing Division, M/S T27A-1. Senior Member, AIAA.

as non-traditional approaches such as neural networks and evolutionary algorithms to obtain optimal aerodynamic shapes and designs.

This article deals with an evolutionary algorithm (EA) known at Differential Evolution[1,2] for aerodynamic shape optimization. DE is a simple and robust evolutionary strategy (ES) that has been proven effective for several difficult optimization problems.[3] Over the years, DE has been applied successfully to various optimization problems in aeronautics.[4,5,6,7,8,9] Although the DE algorithm was originally developed for unconstrained optimization, it can be modified easily to handle constrained problems. Pareto-based extensions of the algorithm to multiobjective optimization have also been developed recently.[10,11,12,13] Generally speaking, population-based evolutionary approaches such as DE are easy to implement and are quite robust and capable of locating the global optimum. However, as with other EA approaches, DE often requires a considerable number of objective function evaluations to arrive at the optimal solution. This poses a serious impediment to the use of DE in aerodynamic shape optimization applications where the objective function evaluations are performed by computationally expensive analysis codes based on the Euler or Navier-Stokes equations.

The purpose of this article is to discuss various modifications to the DE algorithm that can lead to improved computational efficiency and apply them to aerodynamic shape optimization problems. Several approaches have been implemented in a DE-based aerodynamic shape optimization method that uses a Navier-Stokes solver for objective function evaluations. The shape optimization method is implemented on distributed parallel computers so that new designs can be obtained within reasonable turnaround times. Results are presented for the inverse design of a turbine airfoil. The efficiency improvements achieved by the different approaches are evaluated and compared. Some of the approaches considered here have been reported by the author in earlier work[8,9] This article extends our prior work and summarizes our efforts to date aimed at improving the efficiency of the DE algorithm.

Various other efforts to improve the efficiency of the DE algorithm have also been reported in the literature. Chiou and Wang[14] developed a hybrid DE algorithm that incorporated a gradient-based local search method for chemical engineering applications. Another hybrid algorithm combining DE with a first order gradient method was proposed by Lopez-Cruz[15] for optimal control. In aeronautics, Rogalsky et al.[6] developed a DE hybrid by incorporating a Nelder-Mead downhill simplex search and used it with a potential flow solver in the inverse design of turbomachinery airfoils. Airfoil design optimization methods that use the DE algorithm in conjunction with neural network strategies have also been reported by Rai.[7] The key ingredient in all these efforts is the use of hybridization to improve the efficiency of DE. Since the efficiency of DE is also highly dependent on the choice of the parameter settings used in the algorithm, other efforts toward improving DE efficiency have been aimed at developing strategies for the judicious selection of these parameters. Lopez-Cruz[15] proposed a parameter control strategy based on simple heuristics. Another approach using variable parameter settings was proposed by Smuc.[16] Liu and Lampinen[17] developed an adaptive DE algorithm with fuzzy logic controlled search parameters, while Zaharie[18,19] used a self-adaptive strategy for varying the parameters during the course of the optimization run.

II. Differential Evolution

DE is a conceptually simple ES developed for single-objective optimization in continuous search spaces with good convergence properties that have been demonstrated in a variety of applications.[3] Details of the algorithm can be found elsewhere;[1,2] only its main features are summarized here.

DE uses a population P that contains m n-dimensional real-valued parameter vectors, where n is the number of parameters or decision variables:

$$P = \{\bar{x}_1, \ldots, \bar{x}_m\}$$

The population is usually initialized at generation $g = 0$ in a random fashion:

$$P(0) = \{\bar{x}_1(0), \ldots, \bar{x}_m(0)\}, \; g = 0$$

The population size m is maintained constant throughout the optimization process. Differential evolution is thus similar to a (μ, λ) ES[20] with μ and λ equal to m.[21] However, the method differs from standard ES approaches in several respects as described below.

As with all ES-based approaches, mutation is the key ingredient of differential evolution. The basic idea is to generate new parameter vectors for the subsequent generation by using weighted differences between two (or more) parameter vectors selected randomly from the current population to provide appropriately scaled perturbations that modify another parameter vector (or, comparison vector) selected from the same population. This idea has been implemented in various forms. In the classical implementation that is quite popular, new trial parameter vectors $\{\bar{y}_1, \ldots, \bar{y}_m\}$ for the next generation $g+1$ are generated according to the following mutation scheme (which we will refer to as the baseline algorithm:

For $l = 1, m \; ; \; i = 1, n$ generate

$$y_l^i = x_{\alpha^l}^i(g) + F.\left(x_{\beta^l}^i(g) - x_{\gamma^l}^i(g)\right) \tag{1}$$

In the above Eqn. (1), α^l, β^l, γ^l are distinct elements of $\{1, 2, \ldots, m\}$ randomly selected for each l, and $F \in [0,1]$ is a parameter that controls the amplification of the differential variation. Other variants that either use the difference between more than two parameter vectors or keep track of the best parameter vector at each generation and use it in the mutation scheme have also been developed[1] and used with varying success in specific applications.

Following Zaharie,[19] a general expression for the mutation step in DE can be written as:

For $l = 1, m \; ; \; i = 1, n$ generate

$$y_l^i = Gx_*^i(g) + (1-G)x_{\alpha^l}^i(g) + \sum_{q=1}^{Q} F_q.\left(x_{\beta_q^l}^i(g) - x_{\gamma_q^l}^i(g)\right) \tag{2}$$

$G \in [0,1]$ is the coefficient of the convex combination of the best member of the population, $x_*(g)$, and a randomly selected member, $x_{\alpha^l}(g)$, F_q represents the DE differential amplification (mutation) factors, and Q represents the number of differential terms. In the general case, the mutation step described above involves creating each element of each new trial parameter vector as the linear combination of the best member of the population, a randomly selected parameter vector, and Q differences between pairs of randomly selected parameter vectors.This general expression incorporates the many variants of the DE algorithm as special cases. The most commonly used variant described above, and referred to as DE/rand/1/bin, is obtained by setting $G = 0, Q = 1, F_1 = F$. A second popular variant,[1] referred to as DE/best/1/bin, is obtained by setting $G = 1, Q = 1, F_1 = F$, and the DE/best/2/bin variant by setting $G = 1, Q = 2, F_1 = F_2 = F$.

DE is similar to other recombinant ES approaches in that it also uses discrete recombination. The strategy adopted in differential evolution is to modify the trial parameter vectors $\{\bar{y}_1, \ldots, \bar{y}_m\}$ to generate parameter vectors $\{\bar{z}_1, \ldots, \bar{z}_m\}$ as follows:

For $l = 1, m \; ; \; i = 1, n$ generate

$$z_l^i = \begin{cases} y_l^i \text{ with probability } p_c \\ x_l^i(g) \text{ with probability } 1 - p_c \end{cases} \tag{3}$$

where p_c is a parameter that controls the proportion of perturbed elements in the new population. Note that the mutation and recombination operations described above can lead to new vectors that may fall outside the boundaries of the variables. Various repair rules can be used to ensure that these inadmissible vectors do not enter the population. A simple strategy, which is the one adopted here, is to delete these inadmissible vectors and form new ones until the population is filled.

The selection scheme used in DE is deterministic but differs from methods usually employed in standard ES approaches. Selection is based on local competition only, with the modified trial parameter vector competing against one population member (the comparison vector) and the survivor entering the new population $g + 1$ as follows:

For $l = 1, m$

$$\bar{x}_l(g+1) = \begin{cases} \bar{z}_l & \text{if} \quad f(z_l) < f(x_l(g)) \\ \bar{x}_l(g) & \text{else} \end{cases} \tag{4}$$

where f denotes the corresponding objective function (or fitness) value. This greedy selection criterion results in fast convergence; the adaptive nature of the mutation operator, in general, helps safeguard against premature convergence and allows the process to extricate itself from local optima. The generation counter is incremented and the process is repeated until some stopping criteria are satisfied.

III. Efficiency Improvement Strategies for Differential Evolution

Various approaches that seek to improve the computational efficiency of DE for aerodynamic shape optimization applications are discussed here. Some of these approaches were described in previous work by the author and are briefly reviewed here for the sake of completeness.

A. Metamodeling Techniques

Metamodeling techniques for improving the efficiency of evolutionary approaches are based on the use of approximate models as surrogates for the actual objective functions. These surrogates can be incorporated in the optimization process and their judicious use can reduce the number of calls to the expensive analysis codes. One metamodeling approach that has received much attention is the response surface method (RSM). While traditional RSM uses low-order polynomials for function approximation, generalized response surface methods (GRSM) allow for the inclusion of a wide range of approximations, including polynomials, neural networks, kriging, multivariate adaptive regression splines, radial basis functions, and multiquadrics. Both global and local GRSM approaches have been established. In the global approach a GRSM metamodel for the entire design space is used and gradually refined as the optimization progresses. Since developing good global metamodels with validity over the entire design space can be difficult, a local approach based on local approximations and a sequential approximate optimization strategy for iteratively zooming into the region of design space around the optimum is often preferred. Typically, the optimization process is decomposed into a sequence of cycles and an optimization subproblem is defined within a trust region, i.e., a smaller part of the design space, where local metamodels are used as surrogates for the exact objective functions. The exact objective functions are evaluated only at a limited number of points in each trust region thus reducing computational cost. The trust regions are resized and/or moved as the optimization progresses. Various optimization frameworks based on such trust-region and move-limit methods have been developed to strike an appropriate balance between the use of exact and approximate function evaluations.[23]

A GRSM metamodeling strategy for DE, DE-NN, is described in Madavan[8] with metamodels based on artificial neural networks and is used here for comparison. A simple strategy is adopted in DE-NN where the metamodel is used only in the latter stages of the optimization after the population has evolved to the general vicinity of the optimal solution. This eliminates the need for a more elaborate sequential zooming strategy although it

limits the efficiency improvements. Using the neural network as a local response surface with validity only in a small region of the design space makes neural network training easier and improves network generalization abilities. A three-layer feed-forward neural network is used.

B. Memetic Methods

Generally speaking, evolutionary algorithms are not well suited for fine-tuned search close to the optimal solution. Another approach to improving the efficiency of evolutionary approaches is the use of memetic methods that apply a separate local search process to refine specific individuals. These methods are also referred to as hybrid algorithms or genetic local searchers. The general idea is to combine the global nature of evolutionary search with more efficient deterministic local search techniques. A simple strategy often adopted is to perform a two-stage optimization with the best solution determined by the GA being used as the starting point for the local search method. Alternatively, strategies that permit dynamic coupling and interaction between the GA and the local search method can be used that intuitively seem likely to be more effective.

In the context of DE, memetic methods have been explored in Madavan[9] where the DE algorithm was combined with a dynamic hill climbing[24,25] (DHC) local search technique in order to exploit the complementary advantages of both methods and achieve better computational efficiency than standard DE. This approach, referred to as DE-DHC, is described briefly below.

The core of the basic DHC algorithm is an efficient technique for locating local optima.[24] The search originates at a specified location given by the vector $\bar{x}$ of the design parameters and uses a probing vector $\bar{v}$ whose length is initially specified but increases or decreases dynamically to suit the local objective function terrain. Random move directions are tried (up to a specified maximum number) for a given probing vector length $|v|$. If the objective function value at a new point, $\bar{x} + \bar{v}$, is better than the previous value, the probing vector length is doubled and further regions of the domain are searched; if not, the vector length is halved and a more localized search is performed. In addition, a memory vector $\bar{u}$ is also used to keep track of the previous successful move direction. A linear combination $\bar{u} + \bar{v}$ of this vector and the probing vector is also evaluated as a candidate move direction. The process stops when the probing vector length falls below a specified (small) threshold value.

An important decision that has to be made for efficient optimization is the relative effort levels for the global and the local search. There are various aspects to this decision dealing with which and how many population members should be chosen for the local search, when the local search should be invoked, and when it should be terminated. The standard approach is to specify these parameters and keep their values fixed during the optimization. However, this involves considerable parameter tuning to obtain the best values for a particular application. Further, the tuning process is application-specific and will have to be repeated for a different application. Following Espinoza, et al.[26] an alternate approach is used in Madavan[9] where these parameters are not specified but adapt in response to recent performance as the algorithm converges to the optimal solution. The change in the relative coefficient of variation (CV) of the fitness function between generations is monitored.[26] CV is defined as the ratio of the mean and the standard deviation of the population fitness. In a similar fashion, the number of local search iterations to be performed before switching back to the global DE search is decided by comparing the most recent fitness improvement by local search with the latest fitness improvement by global search.[26] In general, local search typically is performed on a small subset of the population and the probability of a population member being selected for local search can also be made to adapt with the solution.[26] However, in the results reported here, only one (either the current best or randomly chosen) member of the population from a generation is selected for the local search.

C. Variable Parameter Differential Evolution

The convergence behavior of the DE algorithm can fall into one of three categories:[19] (a) Good convergence, where the algorithm approaches the global optimum in a "reasonable" number of function evaluations; (b) Poor convergence or misconvergence, where the algorithm converges to a local (not global) optimum, or gets stuck in a non-optimal state. This occurs primarily due to the population losing diversity, although DE has been known to "stagnate" in certain conditions even when the population is diverse;[22] (c) Slow convergence, where the conver-

gence velocity is slow and a "large" number of function evaluations is required to achieve the optimal solution. The general convergence behavior of the DE algorithm is highly dependent on the choice of the three DE parameters, namely, the population size, the differential amplification (mutation) factor, F, and the crossover probability, p_c. For any particular problem, these parameters are responsible for whether the convergence is good, slow, or, poor. Even when the algorithm exhibits good convergence, the choice of these three parameters can have a marked effect on the computational efficiency, i.e., the number of objective function evaluations required to obtain the optimal solution. The parameter settings are highly problem dependent and can vary widely from one problem to the next. Generally speaking, the algorithm is less sensitive to the choice of F and p_c for large population sizes, but computational efficiency is reduced usually since a larger number of function evaluations is required.

Closely tied in with the issue of computational efficiency of any evolutionary approach such as DE is its computational effectiveness. While efficiency is concerned with how rapidly the algorithm can achieve the optimal solution, effectiveness reflects the reliability or robustness of the algorithm, i.e., its ability to achieve the optimal solution irrespective of the initial population choice. For standard test problems with known optima, computational effectiveness can be measured by performing several independent runs from different starting populations and determining the fraction of runs that complete successfully. To be of any practical use, an algorithm must achieve high computational efficiency in conjunction with very high computational effectiveness (close to 100%).

Typically, the proper choice of DE parameters for a particular problem required some form of parameter tuning. Such parameter tuning can be time-consuming and expensive requiring multiple runs of the algorithm. As mentioned earlier, other approaches to address this problem have been reported, such as adaptive and self-adaptive strategies for adjusting the parameters[17,18,19] or the use of stochastically varying parameters.[16]

In the present work, a variable parameter DE approach, DE-VP is developed that extends the basic idea[16] of using stochastically varying mutation and crossover parameters in DE for improved efficiency. In the original approach[16] the parameters were varied over a very limited range (0.5-0.8 for F, and 0.3-0.7 for p_c). While good results were achieved for some specific problems our experiments with this approach over a range of test problems were generally not promising. As mentioned earlier, our focus in these experiments was to achieve accurate optimal solutions with high efficiency with high effectiveness (typically, 99-100%). The results were even worse when the parameters were varied over a wider range. Some of the results of these experiments are discussed later in the Results section.

Stemming from the experiences described above, an alternate formulation was used in developing the DE-VP algorithm. In this approach, the generalized expression for the mutation step (Eqn. (2)) is used instead of the baseline DE mutation operator (Eqn. (1)). In particular, a specific form of the generalized expression given in Eqn. (2) for the mutation step is used in the DE-VP algorithm, with $Q = 2, F_1 = F, F_2 = (1-F)$. However, the fixed parameters F and G, are replaced by the parameters F_l and G_l that are allowed to vary in a stochastic fashion during the optimization run. The mutation step in the variable parameter DE-VP algorithm is thus written as:

For $l = 1, m$; $i = 1, n$ generate

$$y_l^i = G_l x_*^i(g) + (1-G_l)x_{\alpha^l}^i(g) + F_l \cdot \left(x_{\beta^l}^i(g) - x_{\gamma^l}^i(g)\right) + (1-F_l) \cdot \left(x_{\eta^l}^i(g) - x_{\nu^l}^i(g)\right) \tag{5}$$

In the above α^l, β^l, γ^l, η^l, and ν^l are distinct elements of $\{1, 2, \ldots, m\}$ randomly selected for each l

In a similar fashion, the crossover parameter p_c is replaced by the variable parameter p_l and the crossover or recombination step is written as:

For $l = 1, m$; $i = 1, n$ generate

$$z_l^i = \begin{cases} y_l^i \text{ with probability } p_l \\ x_l^i(g) \text{ with probability } 1 - p_l \end{cases}$$

The parameters F_l, G_l and p_l are chosen for each population member as:

$$F_l = rand[0, 1] \cdot (F_{max} - F_{min})$$

$$G_l = rand[0, 1] \cdot (G_{max} - G_{min})$$

$$p_l = rand[0, 1] \cdot (p_{max} - p_{min})$$

with $F_{max} = G_{max} = p_{max} = 1.0$, $F_{min} = G_{min} = 0.0$, and $p_{min} = 0.3$

This formulation recognizes the fact that instead of the baseline DE strategy one of the many DE variants may actually be the most efficient choice for a particular problem and allows these alternate variants to influence the optimization process.

As in the DE-DHC algorithm, diversity preserving strategies (discussed below) are also incorporated in the DE-VP algorithm to allow the use of small population sizes. Results obtained for a wide range of test problems show that this approach leads to good computational efficiency while obviating the need for manual tuning of the parameters. The only parameters that need to be chosen for any optimization problem are the population size and the diversity parameter. As a general rule, the population size can be chosen to be about the same as the number of parameters (but not less than 6). The choice of the diversity parameter is not critical and a constant value was used for all the computations.

E. Other Associated Strategies

Some additional improvement strategies that have been incorporated here to enhance the efficiency of the DE algorithm are described in this subsection. It is noted that these have been used both individually or in combination with each other and in some fashion with all of the broad techniques described above.

Variable Complexity Strategies

The main idea in variable complexity modeling strategies[27] is to tailor the "complexity" of the analysis codes used in objective function evaluation to meet the needs of the optimization phase or method as it evolves toward the optimum solution. This is also referred to as multiscale or multilevel strategies in the literature. In the context of aerodynamic design, "complexity" can mean any of several attributes of the CFD analysis codes used, such as fidelity level, grid density, and convergence criteria. Variable complexity modeling is implemented here by using low fidelity, coarse grid Euler computations in the initial phases of the DE-based optimization followed by fine-grid Navier-Stokes analyses that are more appropriate as the algorithm approaches convergence. One crucial issue in using information from numerical models of varying fidelity on varying grid sizes is that the numerical inaccuracies can impart variation in the fitness function evaluations for the same candidate designs. However, the sequential strategy used here seems to work well for the aerodynamic optimization problem considered.

Diversity Preserving or Micro-GA Strategies

Another general strategy for improving efficiency is to reduce the overall population size. This must be done with care to avoid premature convergence and not sacrifice robustness or reliability. Although more generations are typically required for convergence, the overall number of objective function evaluations required can be smaller since fewer evaluations are being performed at each generation.

In order to make the DE algorithm work reliably with much smaller population sizes a diversity-preserving technique is incorporated. Diversity is usually not an issue with large population sizes, but when DE is used with small population sizes there is a very strong tendency for the population to lose diversity and get closely clustered. The DE mutation and crossover operations are then unable to generate better new individuals and the algo-

rithm converges prematurely to a local optimum and population diversity is eventually lost completely as all members of the population soon cluster around this point. In order to maintain population diversity, a degree of diversity parameter[14] is monitored in the current approach. If this measure of diversity falls below a specific tolerance level, only the current best population member is retained and the remaining population is re-initialized for the next generation. While frequent population re-initialization slows down convergence, results show that it is possible to achieve reductions in the overall number of function evaluations by using a much smaller population size than what would be normally required to maintain diversity and converge reliably to the global optimum.

Function Evaluation Reuse Strategies

This is a simple idea of storing all the evaluated solutions and their fitness values in a database for later reuse during the optimization process. This approach can save a significant amount of computation time. In particular, instead of initializing a new Navier-Stokes evaluation from freestream conditions the solution from the nearest (in an Euclidean sense) design in the database can be used as the starting solution.

Parallel Computing Strategies

The discussion so far has centered on enhancing the *algorithmic efficiency* of DE to improve overall design cycle time and cost. In addition, substantial reductions in overall design time can also be achieved by resorting to parallelization techniques on parallel computers that can perform distributed computations simultaneously on multiple processors. All the DE efficiency enhancement methods described here lend themselves to such parallelization in a straightforward manner. In the present work the various methods have been implemented on distributed parallel computers such as the SGI Origin 3000. It is noted that most of the computing time is expended in the aerodynamic analyses. Parallel implementation relies on the simultaneous computation of multiple, independent aerodynamic simulations on separate processors. A script-based procedure is used that invokes a variable number of processors depending on processor availability and the population size. A "master-slave" arrangement is used, with the master handling the tasks of setting up the simulations, neural network training as needed, and the farming out of the aerodynamic computations to the other slave processors. Since the aerodynamic computations are independent of each other, no communication between the processors is required until the computations are completed. The slave processors then communicate their results to the master which then performs the necessary calculations to determine the population members of the next generation.

IV. Aerodynamic Shape Optimization Method Implementation Details

Some details regarding other aspects of the DE-based aerodynamic shape optimization method incorporating the various efficiency enhancement approaches are described briefly in this section.

A. Constraint Handling

An efficient constraint handling mechanism is incorporated in the optimization method. It is a parameter-less approach that helps steer the algorithm away from infeasible regions of the design space. This constraint handling method was found to perform well when applied to various test problems taken from the constrained optimization literature. The constraint handling mechanism was adapted for use in aerodynamic optimization where both physical constraints (e.g. maximum airfoil thickness) as well as aerodynamic constraints (e.g. wavy surfaces) are imposed. Airfoil geometries that do not violate the constraints but for which the CFD solver fails to converge are also deemed infeasible and handled appropriately.

B. Airfoil Geometry Parametrization

Geometry parametrization and prudent selection of design variables are critical aspects of any shape optimization procedure. Since this study focuses on airfoil redesign, the ability to represent various airfoil geometries with a common set of geometrical parameters is essential. Variations of the airfoil geometry can be obtained then by smoothly varying these parameters. Geometrical constraints imposed for various reasons, such as structural,

aerodynamic (e.g., to eliminate flow separation), etc., should be included in this parametric representation as much as possible. Additionally, the smallest number of parameters should be used to represent the family of airfoils. Here, the airfoil geometry parametrization method[8] is adopted with a total of 13 geometric parameters being used to define the airfoil geometry. This parametrization provided the necessary variations in airfoil geometry required by the optimization procedure.

C. CFD Simulation Methodology

A Navier-Stokes solver was used to perform the flow simulations (direct function evaluations) that serve as inputs to the shape optimization process. The solver used is a modified version of the ROTOR-2 computer code[28] and solves the two-dimensional, Navier-Stokes equations around a single airfoil in a cascade (with spanwise periodic boundary conditions) for a given set of inlet and exit conditions. Multiple grids are used to discretize the flow domain; an inner "0" grid that contains the airfoil and an outer "H" grid that conforms to the external boundaries as shown in Fig. 1. For the analyses performed here, each inner O grid has 151 points in the circumferential direction and 41 points in the wall-normal direction. Each outer H grid has 101 points in the axial direction and 41 points in the transverse direction. For the sake of clarity, only some of the grid points are shown in Fig. 1.

The dependent variables are initialized to freestream values and the equations of motion are then integrated subject to the boundary conditions. The flow parameters that are specified are the pressure ratio across the turbine (ratio of exit static pressure to inlet total pressure), inlet temperature and flow angle, flow coefficient, and unit Reynolds number based on inlet conditions.

D. Design Objective Formulation

Various design objectives can be incorporated in the optimization method depending on the problem being solved. For the inverse turbine airfoil design shown here, the design objective function was formulated as the equally-weighted sum-of-squares error between the target and actual pressure obtained during the optimization process at various locations on the airfoil.

V. Results

Results obtained using the various DE-based approaches are presented in this section. First, the variable-parameter DE algorithm, DE-VP, is evaluated by applying it to a series of test constrained optimization problems taken from the literature. Results for the aerodynamic shape optimization of a turbine airfoil are then presented using the various DE-based approaches, DE-NN, DE-DHC, and DE-VP. Note that the results obtained using the DE-NN and DE-DHC hybrids were presented earlier[8,9] and are included here for comparison.

A. Test Optimization Problems Using The Variable Parameter DE algorithm (DE-VP)

The DE-VP algorithm was first tested using a standard suite of benchmark constrained optimization problems, often referred to as the Michalewicz[29] problem set that has also been used by others[30] to evaluate evolutionary algorithms.The properties of the test problems along with the optimal solution obtained by the DE algorithm summarized briefly in Table 1. For some of the test problems with equality constraints, each constraint was replaced by two inequality constraints that allow a small violation (usually, 0.001) of the actual equality constraint.

The convergence history of the DE-VP algorithm on test optimization problem G7 using different population sizes is shown in Fig. 2 and can be compared to the convergence history for the baseline DE algorithm shown in Fig. 3. In both figures, the results are averaged over 200 independent runs. For the baseline DE results shown in Fig. 2 the parameters were tuned to obtain the settings that resulted in the best performance. The F and p_c parameters were maintained at these settings and the population size was varied. Figure 2 shows that the baseline DE algorithm does not converge to the optimal solution for population sizes of 10 and 15. The diversity-reserving strategy in DE-VP improves convergence as compared to baseline DE with the smallest population size of 10

but fails to achieve the optimal solution. However, unlike baseline DE, DE-VP reaches the optimal solution with population size of 15. At the higher population sizes the diversity-preserving strategy seems to have little or no effect. A similar pattern is noted in Fig. 4 and Fig. 5 which compare the convergence histories for the baseline DE and DE-VP algorithms, respectively, for problem G10. Again, the diversity-preserving strategy works effectively for low population sizes (10 and 15) and improves the convergence.

In order to evaluate the computational efficiency of the DE-VP algorithm in conjunction with high computational effectiveness, 200 independent runs were made for each problem in the Michalewicz[29] problem set (except for G3 where the number was 50). The termination criterion chosen is when the algorithm reaches the optimum to within an allowable error of 0.0001% and satisfies all the constraints. The total number of function and constraint evaluations is used as a measure of the computational efficiency while the computational effectiveness or reliability is measured by the fraction of the total number of runs that converged successfully for this termination criterion. An additional requirement for successful completion was 100% computational effectiveness or reliability.

The results of these computational experiments on the Michalewicz[29] problem set are summarized in Table 2 which compares the best results (highest efficiency, 100% effectiveness) obtained using baseline DE with the variable parameter DE approaches. For the baseline DE results, substantial tuning was performed for each problem to determine the best parameter settings. The efficiency measure is the total number of function and constraint evaluations, with a lower number denoting higher efficiency. The corresponding parameter settings, population size for the variable parameter approaches, and population size, mutation, and crossover parameters for the baseline DE) are also shown. Table 2 also shows results obtained by simply using variable parameters in the baseline DE algorithm. This is similar, but not identical, to the approach of Smuc[16] since a wider range of parameters and a different diversity-preserving strategy is used here.

The results in Table 2 show that varying parameters in the baseline DE is not effective. With the exception of problem G2, all the results show much poorer efficiency than the tuned baseline DE or the DE-VP algorithm. The DE-VP algorithm performs very well in comparison to the best-tuned baseline DE algorithm. Out of 11 problems, DE-VP is substantially better that the tuned baseline DE in 6 problems (G2, G3, G4, G5, G7, G9) roughly the same in 1 (G1), and worse in 4 (G5, G7, G9, G10). In the cases where DE-VP performed poorly, it is important to note that the choice of parameters is very critical in these problems. This fact is borne out in Table 3 which shows the effect of different parameter choices on the efficiency of baseline DE for problem G10. The population size is maintained at 30, which along with parameter settings $F = 0.75$, $p_c = 1.0$ leads to the best-tuned performance. The algorithm performance drops substantially when the parameters F and p_c are varied from the tuned settings. In fact, when F is reduced from 0.75 to 0.65 with all other parameters the same the algorithm fails with effectiveness of only 1%. A similar situation occurs for problem G5. The baseline DE algorithm converges with 100% effectiveness for F and p_c in the ranges (0.7-1.0) and (0.8-1.0), respectively, and for a range of populations of size 15 and higher. (The best result, as noted in Table 2 is for a population size of 15, with $F = 0.9$, $p_c = 1.0$). However, for parameter settings outside this range it is not possible to achieve 100% effectiveness. Given the sensitivity of the baseline DE to the parameters for these problems, the overall performance of DE-VP is noteworthy since it only requires a proper choice of population size as input.

B. Aerodynamic Shape Optimization

The standard DE algorithm and the various approaches for improving its efficiency were used in the inverse design of a turbine airfoil with a specified pressure distribution. Results are shown here for the baseline DE algorithm, the hybrid variants DE-NN and DE-DHC, and the variable parameter DE-VP algorithm. The target pressure distribution was obtained at the midspan of a turbine vane from a modern Pratt and Whitney jet engine. Several flow and geometry parameters were also supplied and used in the design process. The design objective function was formulated as the equally-weighted sum-of-squares error between the target and actual pressure obtained during the optimization process at 45 locations on the airfoil.

The initial design space was chosen to be quite large to allow a wide range of airfoil shapes to be explored. A representative sampling of some of the initial airfoil geometries used in the computations is shown in Fig. 6. In

order to hold the CFD function evaluations to a reasonable number, 6 design variables (instead of 13) were used in the initial stages of the optimization process.

For the DE-NN algorithm a population size of 25 members was used. Several generations were first evolved using the baseline DE algorithm. In the early stages of evolution several of the airfoil geometries were determined to be infeasible and hence were not evaluated by the CFD solver. After 13 generations had evolved, the number of design variables was increased from 6 to 13 and the population was evolved further for another 5 generations. At this point the switch to the DE-NN algorithm was made with the results from this generation being used to train the neural network. Figure 7 shows the data used in neural network training in the form of an envelope of pressure data around the target pressure distribution. Note, however, that the network was trained directly on the sum-square-error and not on the individual pressure data. The data envelope in Fig. 7 represents the variation of the pressure distributions for the range of airfoil geometries in the neural network training set and is meant to convey the local nature of the neural network response surface in the vicinity of the optimal solution.

For the memetic DE-DHC algorithm a small population size of 10 members was used. At each generation, the CV ratio of the population was monitored and the best member of the population was picked for DHC search if deemed necessary. In order to minimize the number of function evaluations required, the DHC search was constrained to a small local region around the chosen location in design space. The diversity-preserving techniques described earlier were used to ensure that the solution did not converge prematurely although this was not much of an issue for this particular objective function landscape. The diversity-preserving strategy was also used for the DE-VP algorithm with a population size of 13 members and with the minimum and maximum parameter settings as described earlier.

The optimal pressure distributions obtained using the various DE approaches are shown in Fig. 8. All the methods produce results that are seen to agree well with the target P&W distribution. The airfoil geometries corresponding to these optimal pressure distributions are also in close agreement as shown in Fig. 9.

The convergence behavior of the various DE-based approaches are compared to the baseline DE algorithm in Fig. 10 which plots the mean objective function value of the population at each generation as a function of the number of CFD function evaluations. The baseline DE algorithm converges in about 650 function evaluations with the population mean decreasing steadily with each generation with the exception of the slight increase noted corresponding to the switch from 6 to 13 design parameters. Figure 10 also shows that convergence of the DE algorithm is slower in latter generations as the optimal solution is approached, thus illustrating the need for hybrid approaches that combines the global search and exploration capabilities of the DE algorithm with other algorithms that can converge rapidly to an optimum when initialized in the local neighborhood. By switching to the DE-NN algorithm in the latter stages, convergence to the optimal pressure distribution is achieved in 500 function evaluations. The DE-DHC algorithm converges in 420 function evaluations and exhibits slightly better performance than the DE-NN algorithm for this case. The DE-VP algorithm also converges in about 450 function evaluations. Reductions of more than 30% in the number of function evaluations are thus achieved by the DE-NN, DE-DHC, and DE-VP approaches over the baseline DE algorithm.The DE-VP algorithm appears the most advantageous since it requires only two parameters as input, the population size and a diversity limit.

VI. Summary and Conclusions

Various approaches for improving the computational efficiency of the DE algorithm have been studied in this article. These include two different hybridization strategies incorporating the use of either neural networks or local search methods, and another strategy that improves DE efficiency by reducing its sensitivity to the mutation and crossover parameter settings. The first strategy, DE-NN, uses trained neural networks to perform the function evaluations. The second strategy, DE-DHC, incorporates the DHC local search technique in the DE algorithm in order to exploit the complementary advantages of both methods and achieve better computational efficiency than standard DE. DE-DHC also incorporates techniques that permit the use of much smaller population sizes than standard DE without sacrificing robustness or reliability. The third strategy, DE-VP, uses variable parameters for the mutation and crossover steps in the algorithm and achieves higher efficiency without the need for elaborate

tuning of the parameters. Results presented here show the efficacy of the methods in reducing the number of objective function evaluations and hence the overall computational expense.

While the results reported here are promising, it is noted that the inverse airfoil design problem was chosen simply to demonstrate the general capabilities of the methods. This problem can be solved in a straightforward and more efficient manner by using other optimization techniques. Future plans include applications of the methods described to more appropriate aerodynamic problems involving, for example, multimodal landscapes, multiobjective or multipoint design.

VII. References

[1]Storn, R. and Price, K., "Differential Evolution - A Simple Evolution Strategy for Fast Optimization," *Dr. Dobb's Journal*, Vol. 22, No. 4, April 1997, pp. 18-24.

[2]K. V. Price: 'Differential Evolution: A Fast and Simple Numerical Optimizer'. In: Biennial Conference of the North American Fuzzy Information Processing Society, (NAFIPS), June 1996, ed. by M. Smith, M. Lee, J. Keller, J. Yen (IEEE Press, New York 1996) pp. 524--527.

[3]J. Lampinen: A Bibliography of Differential Evolution Algorithm. Technical Report. Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing, Lappeenranta, Finland (2001).

[4]Nho, K., and Agarwal, R. K., "Fuzzy Logic Model-Based Predictive Control of Aircraft Dynamics Using ANFIS," AIAA Paper 2001-0316, Jan., 2001.

[5]Rogalsky, T., Derksen, R.W. and Kocabiyik, S., "Differential Evolution in Aerodynamic Optimization," *Canadian Aeronautics and Space Institute Journal*, Vol. 46, No. 4, pp. 183-190, Dec. 2000.

[6]Rogalsky, T. and Derksen, R.W., "Hybridization of Differential Evolution for Aerodynamic Design," *Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada*, pp. 729-736, June 11-13, 2000.

[7]Rai, M. M., "Towards a Hybrid Aerodynamic Design Procedure Based on Neural Networks and Evolutionary Methods," AIAA Paper No. 2002-3143. June 2002.

[8]Madavan, N. K., "Turbomachinery Airfoil Design Optimization Using Differential Evolution," Computational Fluid Dynamics 2002, Proceedings of the 2nd International Conference on Computational Fluid Dynamics, Sydney, Australia, Jul. 2002. ed. by S. Armfield, P. Morgan, and K. Srinivas (Springer-Verlag, Germany, 2003), pp. 585-590.

[9]Madavan, N. K., "Aerodynamic Shape Optimization Using Hybridized Differential Evolution," AIAA Paper No. 3792, AIAA Applied Aerodynamics Conference, Orlando, FL, June 24-27, 2003.

[10]H. A. Abbass, Sarker, R., and Newton, C., "PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems," Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, New Jersey, Vol. 2, pp. 971-978, May 2001.

[11]Abbass, H. A., "The Self-Adaptive Pareto Differential Evolution Algorithm," Proceedings of the Congress on Evolutionary Computation, Honolulu, HI, Vol. 2, pp. 831-836, May 2002

[12]Madavan, N. K., "Multiobjective Optimization Using a Pareto Differential Evolution Approach," Proceedings of the Congress on Evolutionary Computation, Honolulu, HI, Vol. 2, pp. 971-978, May 2002.

[13]Rai, M. M., "Robust Optimal Aerodynamic Design Using Evolutionary Methods and Neural Networks," AIAA Paper No. 2004-778, 42nd AIAA Aerospace Sciences Meeting, Jan. 5-8, Reno, NV, 2004.

[14]Chiou, J., and Wang, F. S., "A Hybrid Method of Differential Evolution with Application to Optimal Control Problems of a Bioprocess System," *Proceedings of the IEEE l Conference on Evolutionary Computation*, IEEE, New York, NY, pp. 627-632, 1998.

[15]Lopez-Cruz, I. L., "Efficient Evolutionary Algorithms for Optimal Control," Ph.D. Thesis, Wageningen University, Wageningen, The Netherlands, 2002.

[16]Smuc, T., "Improving Convergence Properties of the Differential Evolution Algorithm," in R. Matousek, P. Osmera (eds.), Proceedings of Mendel 2002, 8th International Conference on Soft Computing, Brno, Czech Republic, pp. 80-86, 2002.

[17]Liu, J, and Lampinen, J., "Adaptive Parameter Control of Differential Evolution," in R. Matousek, P. Osmera (eds.), Proceedings of Mendel 2002, 8th International Conference on Soft Computing, Brno, Czech Republic, pp. 19-26, 2002.

[18]Zaharie, D., "Control of Population Diversity and Adaptation in Differential Evolution Algorithms," in R. Matousek, P. Osmera (eds.), Proceedings of Mendel 2003, 9th International Conference on Soft Computing, Brno, Czech Republic, pp. 41-46, 2003.

[19]Zaharie, D., "Parameter Adaptation in Differential Evolution by Controlling the population Diversity," in D. Petcu, et al. (eds.), Proceedings of the 4th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania,pp.385-397, 2002.

[20]Back. T, Hammel, U., and Schwefel, H.-P., "Evolutionary Computation: Comments on the History and Current State," *IEEE Trans. on Evolutionary Computation*, Vol. 1, pp. 3-17, 1997.

[21]Shokrollahi, M. A., and Storn, R.,'Design of Efficient Erasure Codes with Differential Evolution'. In: *Proceedings of ISIT 2000, International Symposium on Information Theory*, Sorrento, Italy, June 25-30, 2000.

[22]Lampinen, J., and Zelinka, I., "On Stagnation of the Differential Evolution Algorithm," in P. Osmera (ed.), Proceedings of Mendel 2000, 6th International Conference on Soft Computing, Brno, Czech Republic, pp. 76-83, 2000.

[23]Alexandrov, N. M., Dennis, J. E., Lewis. R. M., and Torczon, V., "A Trust Region Framework for Managing the Use of Approximation Models in Optimization," *Structural Optimization*, Vol 15, No. 1, pp. 16-23, 1998.

[24]Yuret, D., and Maza, M., "Dynamic Hill Climbing: Overcoming the Limitations of Optimization Techniques," *2nd Turkish Symposium on Artificial Intelligence and Neural Networks*, pp. 208-212, 1993.

[25]Yuret, D., and Maza, M., "Dynamic Hill Climbing," *AI Expert*, Vol. 9,. No. 3, pp. 26-31, March 1994.

[26]Espinoza, F. P., Minsker, B. S., and Goldberg, D. E., "A Self-Adaptive Hybrid Genetic Algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, International Society for Genetic and Evolutionary Computation, San Francisco, CA, 2001.

[27]Barthelemy, J-F., M., and Haftka, R. T., "Approximation Concepts for Optimum Structural Design -- A Review," *Structural Optimization*, Vol. 5, pp. 129-144, 1993.

[28]Rai, M. M., and Madavan, N. K., "Multi-Airfoil Navier-Stokes Simulations of Turbine Rotor-Stator Interaction," *ASME Journal of Turbomachinery*, Vol. 112, pp. 167-190, Jul. 1990.

[29]Koziel, S., and Michalewicz, Z., "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation*, Vol. 7, No.1, pp. 19-44, 1999.

[30]Lampinen, J., "A Constraint Hanndling Approach for the Differential Evolution Algorithm," Proceedings of the Congress on Evolutionary Computation, Honolulu, HI, Vol. 2, pp. 1468-1473, May 2002.

Test Problem Number[29]	Number of Variables	Objective Function	Total Number of Constraint Functions	Number of Linear Constraint Functions	Number of Nonlinear Constraint Functions	Optimal Solution Achieved
G1	13	nonlinear	9	9	0	-15.00
G2	2	nonlinear	2	0	2	-0.36498
G3	50	nonlinear	50(100)	0	50(100)	-1.0252
G4	5	nonlinear	6	0	6	31025.56
G5	2	nonlinear	5(8)	2(2)	3(6)	5126.484
G6	2	nonlinear	2	0	2	6961.811
G7	10	nonlinear	8	3	5	24.3062
G8	2	nonlinear	2	0	2	0.095825
G9	7	nonlinear	4	0	4	680.630
G10	8	linear	6	3	3	7049.248
G11	2	nonlinear	1(2)	0	1(2)	0.749
Note Each equality constraint was converted to two inequality constraints. Numbers in parentheses denote total number of constraints after this conversion.						

TABLE 1. Summary of constrained test optimization problems studied.

Test Problem[29]	Baseline DE Algorithm				Baseline DE Algorithm with Variable Parameters		Variable Parameter DE (DE-VP) Algorithm	
	Mutation F	Crossover Pc	Population Size	No. of Function and Constraint Evaluations	Population Size	No. of Function and Constraint Evaluations	Population Size	No. of Function and Constraint Evaluations
G1	0.80	0.15	11	13,515	35	23,504	12	13,558
G2	0.65	0.85	90	8683	15	3421	7	1611
G3	0.50	0.90	50	876,227	70	917,434	20	725,851
G4	0.75	0.90	10	4784	20	9901	10	3684
G5	0.90	1.00	15	16,452	50	132,439	25	35,122
G6	0.80	0.95	10	1871	31	6740	7	1702
G7	0.70	1.00	40	58569	90	158,921	32	68,407
G8	0.60	0.90	14	695	20	1050	6	416
G9	0.65	0.95	25	9203	45	25,469	21	11,455
G10	0.75	1.00	30	37,669	70	194,459	30	71,981
G11	0.80	0.95	19	3174	25	8223	8	2506

TABLE 2. Summary of best results in terms of efficiency for 100% computational effectiveness on various test test optimization problems using baseline DE and variable parameter DE approaches.

Population Size	Mutation, F	Crossover, Pc	No. of Function and Constraint Evaluations	Computational Effectiveness
30	0.75	1.00	37,669	100%
30	0.85	1.00	52,272	100%
30	0.65	1.00	32,176	1%
30	0.75	0.90	66,975	100%
30	0.75	0.80	130,951	100%
30	0.85	0.90	92,571	100%
30	0.85	0.80	186,305	100%
30	0.65	0.90	59,215	100%
30	0.65	0.80	102,741	100%

TABLE 3. Effect of the choice of algorithm parameters F and p_c on the computational efficiency of the baseline DE algorithm for test optimization problem G10.

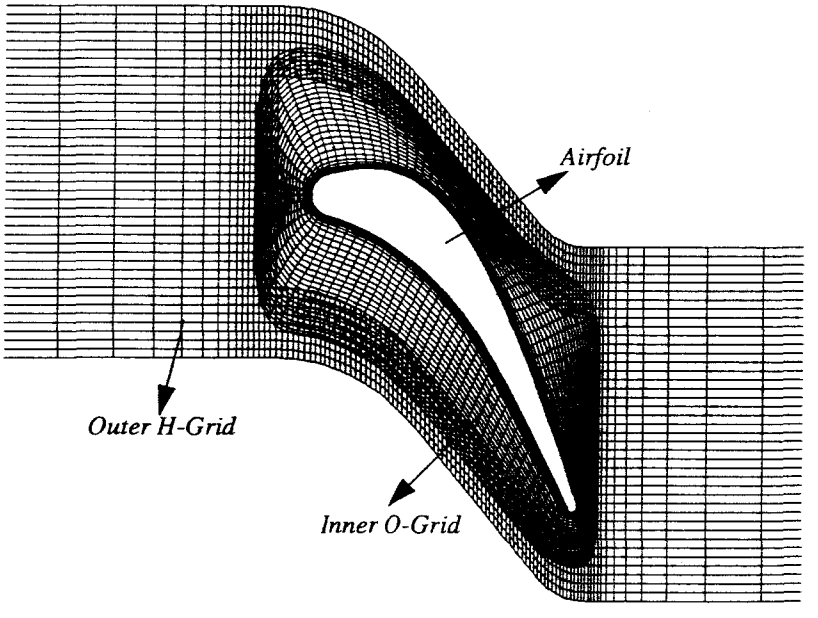

Figure 1. Representative turbine airfoil geometry and computational grid used in the CFD simulations.

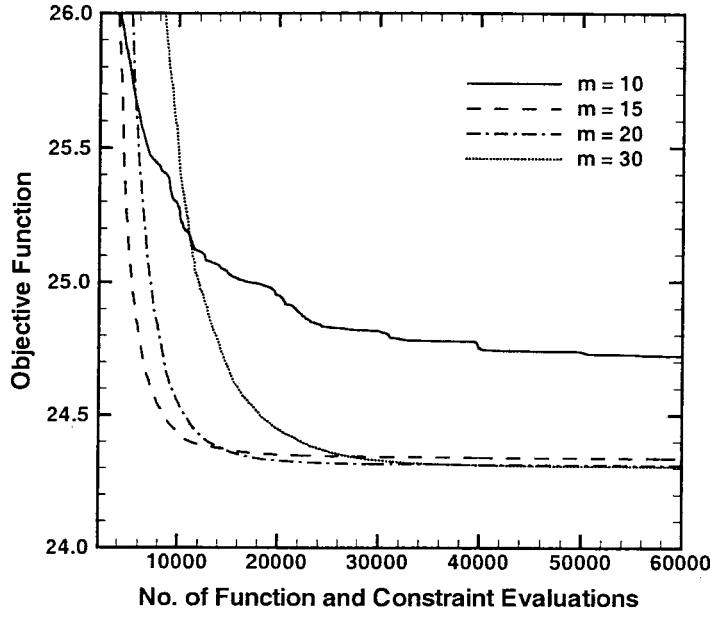

Figure 2. Convergence history of the baseline DE algorithm for the constrained test optimization problem G7 for various population sizes.

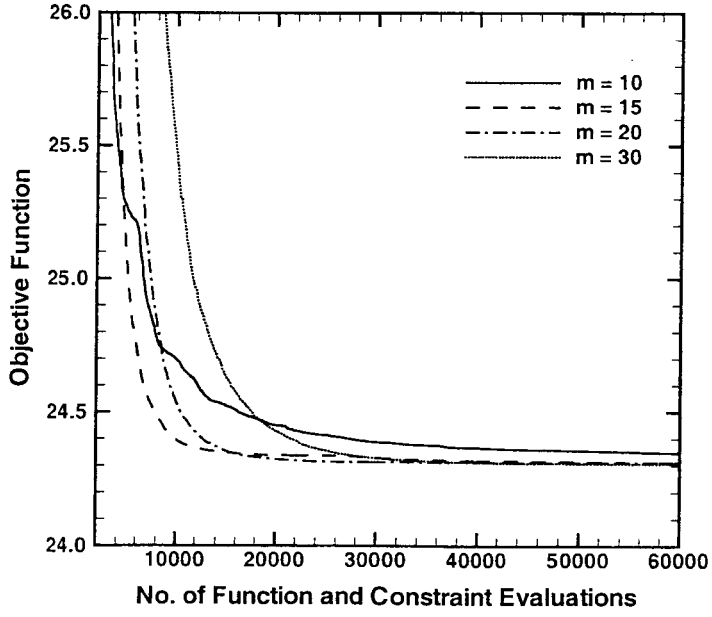

Figure 3. Convergence history of the variable-parameter DE algorithm (DE-VP) for the constrained test optimization problem G7 for various population sizes.

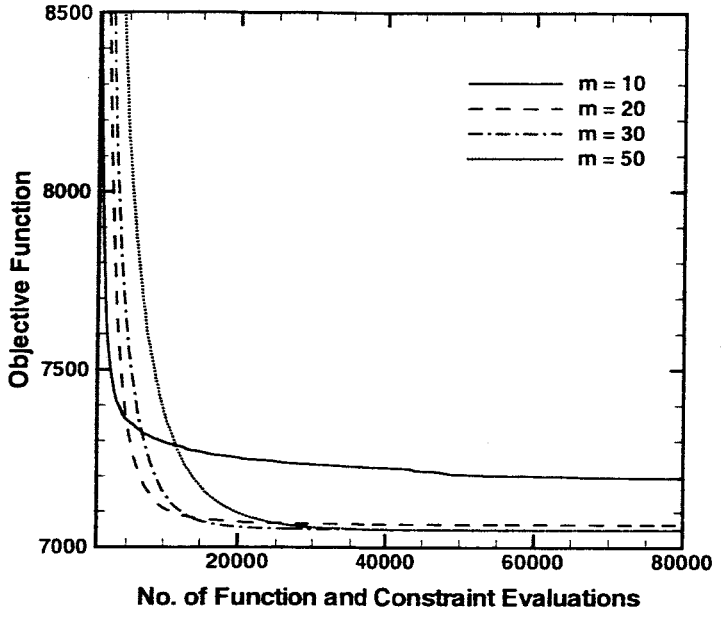

Figure 4. Convergence history of the baseline DE algorithm for the constrained test optimization problem G10 for various population sizes.

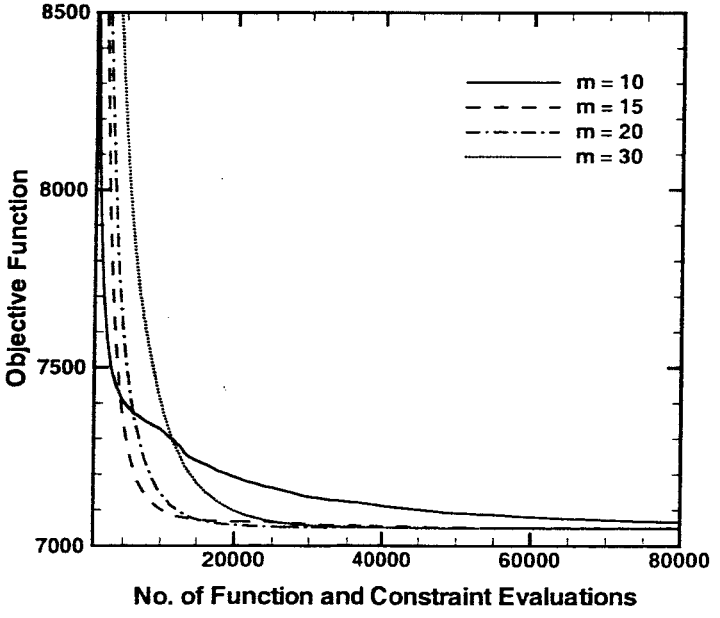

Figure 5. Convergence history of the variable-parameter DE algorithm (DE-VP) for the constrained test optimization problem G10 for various population sizes.

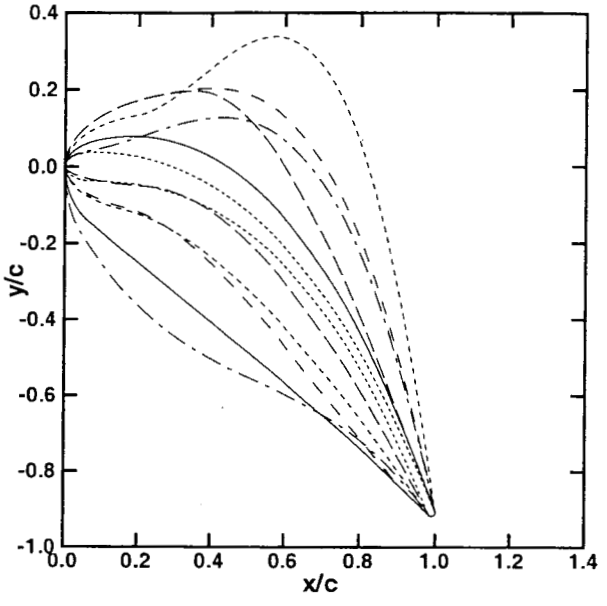

Figure 6. Sampling of airfoil geometries used in the initial population for the various DE-based algorithms.

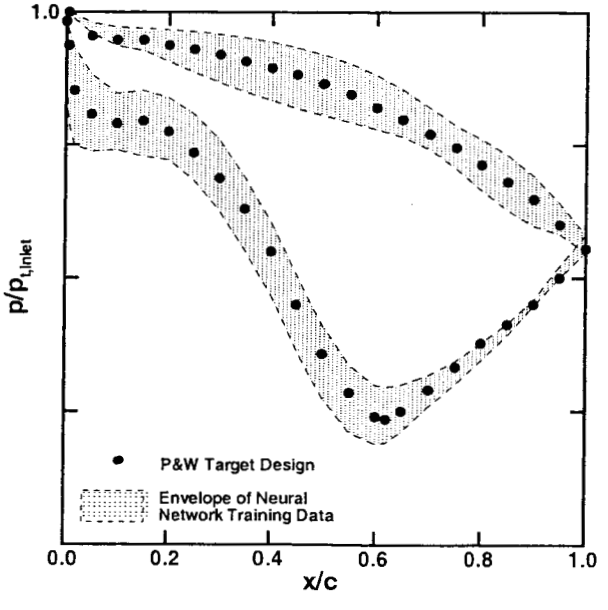

Figure 7. Envelope of airfoil loadings used to train the neural network for the metamodel DE (DE-NN) algorithm.

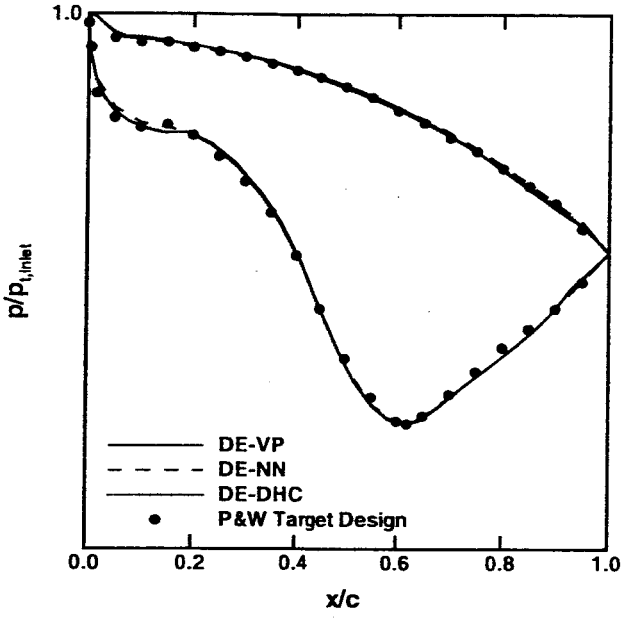

Figure 8. Airfoil pressure loading for the optimal design using the various DE-based algorithms.

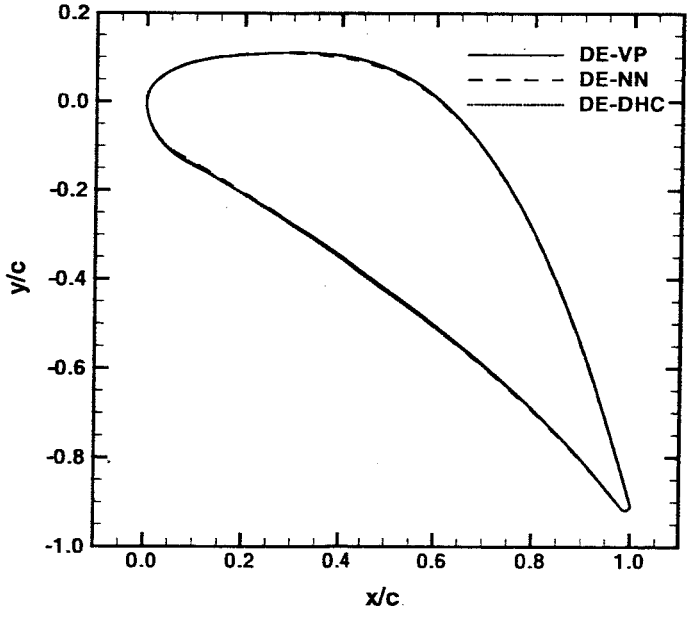

Figure 9. Airfoil geometry for the optimal design using the various DE-based algorithms.

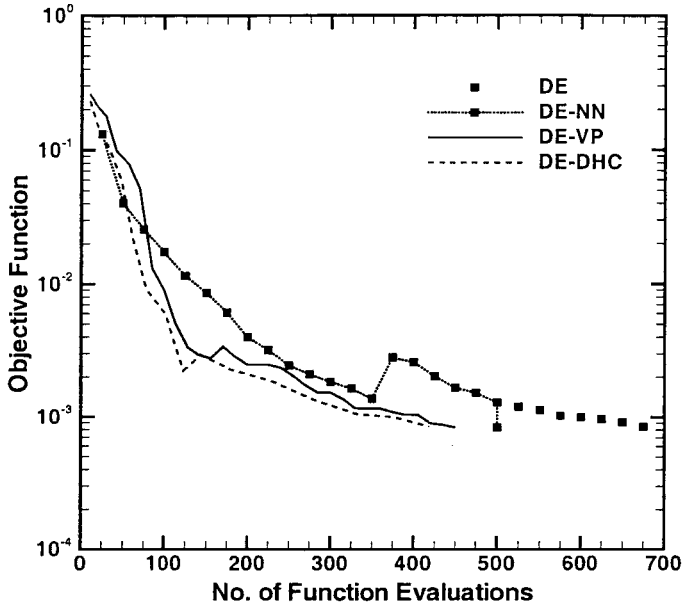

Figure 10. Convergence history for the various DE-based algorithms.