ON IMPROVING EFFICIENCY OF DIFFERENTIAL EVOLUTION FOR AERODYNAMIC SHAPE OPTIMIZATION APPLICATIONS

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

ABSTRACT
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. Several approaches that have proven effective for other evolutionary algorithms are modified and 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 standard test optimization problems and for the inverse design of a turbine airfoil. The efficiency improvements achieved by the different approaches are evaluated and compared.

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 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 as Differential Evolution (DE) for aerodynamic shape optimization. DE is a simple and robust evolutionary strategy (ES) that has been proven effective for several difficult optimization problems. However, its application in aeronautics, however, has been somewhat limited. 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 require a lot 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 explore and summarize various modifications to the DE algorithm that can lead to improved computational efficiency and apply them to aerodynamic shape optimization problems. Several promising approaches that have proven effective in the context of other EAs have been modified and 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 standard test optimization problems from the literature and for the inverse design of a turbine airfoil. The efficiency improvements achieved by the different approaches are evaluated and compared. Some of these approaches have been reported by the author in earlier work. This article extends this prior work and summarizes our efforts to date aimed at improving the efficiency of the DE algorithm.

Other efforts to improve the efficiency of the DE algorithm have also been reported in the literature. Chiou and Wang developed a hybrid DE algorithm that incorporated a gradient-based local search method for chemical engineering applications. In aeronautics, Rogalsky et al. 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 turboma-
chinery airfoils. Airfoil design optimization methods that use the DE algorithm in conjunction with neural network strategies have also been reported by Rai.\textsuperscript{7}

**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.\textsuperscript{3} Details of the algorithm can be found elsewhere;\textsuperscript{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:

\[
P = \{\tilde{x}_1, ..., \tilde{x}_m\}
\]

The population is usually initialized at generation \( g = 0 \) in a random fashion:

\[
P(0) = \{\tilde{x}_1(0), ..., \tilde{x}_m(0)\}, \quad g = 0
\]

The population size \( m \) is maintained constant throughout the optimization process. Differential evolution is thus similar to a \((\mu, \lambda)\) ES\textsuperscript{11} with \( \mu \) and \( \lambda \) equal to \( m \).\textsuperscript{12} The method however 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 rather popular, new trial parameter vectors \( \{\tilde{y}_1, ..., \tilde{y}_m\} \) for the next generation \( g + 1 \) are generated according to the following mutation scheme:

For \( l = 1, m \), \( i = 1, n \) generate

\[
y^i_l = \frac{x^i_l(g)}{\alpha^i_1} + \frac{x^i_l(g) - x^i_{\lambda}(g)}{\alpha^i_3} \quad \text{with probability } p_c
\]

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 \( l = 1, m \)

\[
\tilde{x}_l(g + 1) = \begin{cases} 
\tilde{z}_l & \text{if } f(\tilde{z}_l) < f(x_l(g)) \\
\tilde{x}_l(g) & \text{else}
\end{cases}
\]

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.

**EFFICIENCY IMPROVEMENT STRATEGIES FOR DIFFERENTIAL EVOLUTION**

A variety of approaches that seek to improve the computational efficiency of evolutionary approaches without compromising their desirable features have been proposed in the literature. Several of these were evaluated in the present study in the context of the Differential Evolution algorithm and are discussed below.

**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 incorpo-
rated 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.\(^3\)

The GRSM metamodeling strategy for DE of Madavan\(^9\) with metamodels based on artificial neural networks is used here. A simple strategy is adopted 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 here.

**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\(^{14,15}\) (DHC) local search technique in order to exploit the complementary advantages of both methods and achieve better computational efficiency than standard DE. This approach is used here and described briefly below.

The core of the basic DHC algorithm is an efficient technique for locating local optima.\(^4\) The search originates at a specified location given by the vector \(\vec{x}\) of the design parameters and uses a probing vector \(\vec{v}\) 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, \(\vec{x} + \vec{v}\), 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 \(\vec{u}\) is also used to keep track of the previous successful move direction. A linear combination \(\vec{u} + \vec{v}\) 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.\(^{16}\) 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

\[^{14}\text{Madavan, }\text{P.}\text{, et al.}\text{,}\text{2015.}\]
\[^{15}\text{SUITON}\text{,}\text{W.}\text{, et al.}\text{,}\text{2016.}\]
\[^{16}\text{Espinoza,}\text{E.}\text{, et al.}\text{,}\text{2013.}\]
function between generations is monitored. 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. 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. 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.

**Subpopulation or Island Techniques**

Another efficiency enhancement technique that has had some success in certain applications is the use of subpopulation or island genetic algorithms. These are essentially coarse-grained distributed genetic algorithms where multiple subpopulations or “islands” are evolved in a distributed fashion with interaction occurring between the islands through occasional migration of individuals. Each island searches its own space for feasible solutions. The islands can be connected through various topologies: bidirectional rings, unidirectional rings, hypercubes, etc. Different parameter settings and fitness function models of varying fidelity or resolution can be used in the various islands. For example, there can be several islands using inexpensive low-fidelity evaluations that progressively pass on individuals to fewer islands using higher fidelity evaluations. The final paper will discuss the present implementation in more detail.

**Adaptive Parameter Range Techniques**

The idea in adaptive parameter range techniques is to dynamically alter the parameter range of the space being searched. The idea originated with binary-coded genetic algorithms and is also referred to as dynamic coding or stochastic genetic algorithm. With dynamic coding, higher precision is achieved with the same chromosome length as the extent of the search space is gradually refined during the optimization process. These dynamic coding ideas have also been extended to real-coded genetic algorithms. The approach implemented here in the DE algorithm closely follows that of Oyama et al. More details will be provided in the final paper.

**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 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 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 works well for the problems considered.

**Small Population or Micro-GA Strategies**

Another strategy for improving efficiency is to reduce the overall population size. This must be done with care to avoid premature convergence and not sacrificing 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. When using DE in function optimization, an acceptable trade-off between convergence and population diversity must be determined. The DE parameters $F$ and $p_c$ can be adjusted to control the convergence speed but population diversity is a requisite for ensuring that the algorithm converges to a global, rather than local, optimum. Diversity is a function of the population size, and for a given problem, DE typically becomes more robust with convergence speeds that are less sensitive to the choice of $F$ and $p_c$ as the population size is increased. However, when DE is used with small population sizes, the population can lose diversity and get closely clustered. The DE mutation and crossover operations are then unable to generate better new individuals and the algorithm converges prematurely to a local optimum and population diversity is lost as all members of the population soon cluster around this point. In order to maintain population diversity, a degree
number of slave processors; some additional communi-

cation. In the injection island approach, an overall master
assigns to each sub-population a master and a fixed
rate processors.

Multiple, independent aerodynamic simulations on sepa-
rate processors. All the DE efficiency enhancement
methods described here lend themselves to such paral-
lelization 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 imple-
mentation relies on the simultaneous computation of
multiple, independent aerodynamic simulations on sepa-
rate processors. A script-based procedure is used that
invokes a variable number of processors depending on
processor availability and the population size. A "mas-
ter-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 aerody-
namic 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 gen-
eration. In the injection island approach, an overall master
assigns to each sub-population a master and a fixed
number of slave processors; some additional communi-
cation burden is incurred between the subpopulations
during the migration phase.

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.

Constraint Handling

An efficient constraint handling mechanism is incor-
porated 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 con-
strained optimization literature. The constraint handling
mechanism was adapted for use in aerodynamic optimi-
ization 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.

Airfoil Geometry Parametrization

Geometry parametrization and prudent selection of
design variables are critical aspects of any shape optimi-
ization procedure. Since this study focuses on airfoil
redesign, the ability to represent various airfoil geomet-
ries 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 repres-
ent the family of airfoils. Here, the airfoil geometry
parametrization method\textsuperscript{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 optimiza-
tion procedure.

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\textsuperscript{23} 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 "O" 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.

**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.

**RESULTS**

In the final paper, results obtained using the various approaches will be presented in this section. The approaches will be first tested on various unconstrained and constrained test problems taken from the optimization literature. For the test problems some effort was made to determine appropriate input parameter settings that resulted in good overall convergence. Results for aerodynamic shape optimization of a turbine airfoil will then be presented and compared using the various approaches. In order to provide a flavor of the results some prior results using a couple of the approaches described here are included. These results were presented the author in earlier work.89 The final paper will include more detailed comparisons between these and the other approaches described in this paper.

**Test Optimization Problem**

The test problem considered is the well-known Levy No. 5 function.24 The topology of this objective function \( f_1 \) (see Ref. 9 for details) is shown in Fig. 2. Note that the negative of the function is plotted so all the minima appear as maxima in the figure. There are about 760 local minima with one global minimum and the large number of local optima makes it difficult to locate the global minimum.

The performance of the memetic DE and baseline DE algorithms on this test problem is compared in Table 1. The reliability values in Table 1 denote the percentage of independent tests that converged to the correct optimum and are a measure of the robustness of the algorithms. On test problem \( f_1 \), the base DE algorithm required a population size of 25 and 2045 objective function evaluations to converge to the correct optimum with 100% reliability; the memetic DE algorithm achieves the same 100% reliability with only a population size of 5 and 598 function evaluations. With a population size this small, the base DE algorithm is unable to converge with more than 30% reliability. Thus the memetic DE method is able to achieve a marked improvement of 70% in the number of function evaluations required for the functions \( f_1 \).

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>Algorithm</th>
<th>Pop. Size</th>
<th>Function+Con</th>
<th>Function Evaluations</th>
<th>Reliability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>DE</td>
<td>5</td>
<td>n/a</td>
<td>464</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>10</td>
<td>n/a</td>
<td>821</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>15</td>
<td>n/a</td>
<td>1256</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>20</td>
<td>n/a</td>
<td>1633</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>25</td>
<td>n/a</td>
<td>2045</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>DE-DHC</td>
<td>5</td>
<td>n/a</td>
<td>598</td>
<td>100</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>DE</td>
<td>10</td>
<td>11302</td>
<td>7344</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>15</td>
<td>15428</td>
<td>8614</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>20</td>
<td>21694</td>
<td>11050</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>DE-DHC</td>
<td>5</td>
<td>4689</td>
<td>2138</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1. Performance comparison of DE and hybridized DE-DHC algorithms on unconstrained and constrained optimization test problems. Results based on an average of 100 (20 for \( f_3 \) case) independent test runs.

Figure 3 compares the convergence behavior of the memetic and standard DE algorithms on the test function \( f_1 \). The ordinate in the figure is the objective function value for the best population member at each generation. Typical results are shown from sample test runs with population sizes for both algorithms chosen for 100% reliability. In both cases, the DE algorithm exhibits a typical step convergence pattern, where the objective function value remains constant for several generations before descending to the next lower level. Due to the small population size, the memetic DE starts at higher objective function values but is able to converge rapidly. On the other hand, DE shows good convergence behavior initially, but then slows down in later generations as the optimum is approached.
Aerodynamic Shape Optimization

The standard DE algorithm and the various efficiency improving ideas were also used in the inverse design of a turbine airfoil with a specified pressure distribution. Results\(^9\) are shown here for the baseline DE and the metamodel and memetic variants. 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 sampling of some of the initial airfoil geometries is shown in Fig. 4. 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 design. A population size of 25 and 10 members was used for the metamodel and memetic DE algorithms, respectively.

For the metamodel DE algorithm, several generations were first evolved using the base 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 metamodel DE algorithm was made with the results from this generation being used to train the neural network. Figure 5 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. 5 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 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 optimal pressure distributions obtained using the two DE variants are shown in Fig. 6. Both methods 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. 7.

The above represents some of the results obtained. The final paper will evaluate and compare the various approaches described on these and additional optimization problems. Detailed measurements of algorithm convergence will also be included.

REFERENCES


Figure 1. Representative turbine airfoil geometry and computational grid used in the CFD simulations.
Figure 2. Topology of unconstrained test optimization function $f_1$ (Levy No. 5) in the search space.

Figure 3. Comparison of convergence behavior of DE and memetic DE (DE-DHC) algorithms for function F1.

Figure 4. Sampling of initial airfoil geometries.

Figure 5. Envelope of airfoil loadings used to train the neural network for metamodel DE (DE-NN) algorithm.

Figure 6. Airfoil pressure loading for the optimal design using DE variants.

Figure 7. Airfoil geometry for the optimal design using DE variants.