2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Airfoil Optimization by Evolution Strategies

Drew A. Curriston
Alice E. Smith
Department of Aerospace Engineering
Department of Industrial and Systems Engineering
Auburn University
Auburn, AL

Abstract— **This paper addresses subsonic airfoil optimization using Evolution Strategies (ES) and devises a means of defining the airfoil geometry that reduces unnecessary restrictions in the search space. The solution encoding uses Bezier Control Points to define the geometry of the airfoil, but does not restrict the movement of the control points as was common in previous airfoil optimization algorithms. The ES move operator combined with this improved solution encoding expands the search space to include superior solutions while also enabling a more efficient search to reduce computational cost.**

Keywords—Evolution Strategies; airfoil optimization; subsonic; Bezier; multi-objective optimization, aerodynamic optimization.

I. Introduction

Airfoil optimization has been attempted in a variety of ways for a wide range of objectives. Typically, an airfoil optimization problem tries to maximize the performance of an airfoil with respect to a specific set of performance parameters at a specified flight regime. However, the airfoil must also perform well across a wide variety of flight regimes, as would be experienced in actual flight. Adaptive optimizers have been used often in recent years to optimize airfoils, and many papers have been published with impressive results, but often the search space is restricted significantly to simplify the problem. This paper discusses the development of a subsonic airfoil optimizer that will minimize a specific set of objective functions while imposing the least amount of artificial restrictions on the problem..

II. Optimization Problem

A. *Airfoil Optimization*

In general, airfoil optimization is either conducted by direct (specify airfoil geometry and calculate performance) or indirect (specify pressure curves for desired performance and calculate corresponding geometry) methods. Indirect methods are generally solved in a deterministic way while direct methods have become more popular in recent years through the use of meta-heuristics. This paper will present an algorithm to find the optimal airfoil geometry using a direct method. In order to evaluate the performance of the algorithms, two previously published papers on airfoil optimization were selected as benchmarks for comparison. Two separate optimization problems were applied to the algorithm, each selected from one the papers discussed below. These papers were selected due to the use of the same flow solver, XFoil, in their objective function evaluation as used in this paper. By using the same flow solver, the two optimization algorithms produce the same result for the same solution, therefore improving the basis for comparison.

B. *Benchmarks for Comparison*

The first paper is "*Designing Airfoils using a Reference Point based Evolutionary Many-objective Particle Swarm Optimization Algorithm*," written by Wickramasinghe [6]. Wickramasinghe's paper attempted to optimize the geometry of the NLF0416 airfoil used on a UAV with respect to certain criteria. Six different objectives were evaluated and a Particle Swarm Optimizer was used for his algorithm. The objectives are listed below:

1. Minimize Coefficient of Drag
   (at C_L=0.5, Re = 4.0*10^6, and Mach 0.3)
2. Minimize $C_D/C_L^{(3/2)}$
   (at floating α, Re = 4.0*10^6, and Mach 0.3)
3. Minimize C_{m0}^2
   (at C_L=0, Re = 4.0*10^6, and Mach 0.3)
4. Minimize $1/C_{Lmax}^2$
   (at floating α, Re = 4.0*10^6, and Mach 0.3)
5. Minimize $1/C_L^2$
   (at α=5°, Re = 2.0*10^6, and Mach 0.15)
6. Minimize 1-x_{tr}
   (at α=5°, Re = 2.0*10^6, and Mach 0.15)

The second paper selected for comparison is "*Multi-Objective Evolutionary Optimization of Subsonic Airfoils by Meta-Modeling and Evolution Control*," written by D'Angelo [1]. In this paper a multi-objective optimization is performed using a Pareto Front to minimize the coefficient of drag and maximize the coefficient of lift. This paper also includes work in meta-modeling, artificial neural networks, and evolution control, but the comparison will focus on solution encoding and results.

III. Methods of Airfoil Parameterization

The solution encoding for this problem must provide the geometry of the airfoil, and this encoding of the geometry must always consist of smooth surfaces in order to evaluate them

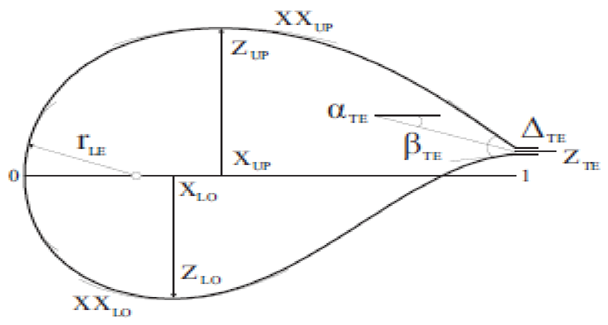


Fig. 1. *The 11 variables used in PARSEC representation of an airfoil [6]*

properly in the flow solver. Obviously, encoding the solution as a series of *x* and *y* coordinates is not feasible because it leads to surfaces that are not smooth and not easy to manipulate with a move operator. Two common representations used in the past are PARSEC and Bezier Curve representations.

A. PARSEC Encoding

PARSEC encoding consists of 11 variables, including angles and distances, that define the shape of the airfoil. These variables are depicted in Figure 1. While this lends to a smooth surface that is easily manipulated by a move operator, it also comes with some problems. The biggest disadvantage is the fact that PARSEC encoding cannot represent all geometric shapes, especially certain desirable features around the trailing edge of the airfoil. This limits the search space and may prevent an optimal or near-optimal solution from being found. The PARSEC encoding is the encoding used by Wickramasinghe in his paper.

B. Bezier Control Points

Another common representation is using Bezier Curves. These use control points and a series of Bernstein Polynomials to define the shape of the airfoil. This lends to a smooth shape and the control points can be easily manipulated, but often the range of movement on the control points is severely restricted. Figure 2 shows a Bezier Curve representation of an airfoil from Feinekos [2]. In this representation the abscissae are fixed along the chord length and the vertical range of the control point movement is limited. This will also limit the possible airfoil geometries found by the algorithm. D'Angelo's paper [1] used Bezier Control Points to encode solutions, and similarly restricted the movement of the control points to ranges defined by the user.

In 2000, Naujoks et al. conducted a multi-point airfoil optimization using Bezier control points and noted several deficiencies with the parameterization. First, the airfoil surface had a tendency to oscillate, which created a wavy surface on the top and bottom of the airfoil. This reduced the performance of the airfoil and produced far from ideal pressure curves. The article found that using three control points rather than six or 12 provided a sub-par solution. As stated in the article, "the use of only three Bezier points on the lower and upper surface might not be the

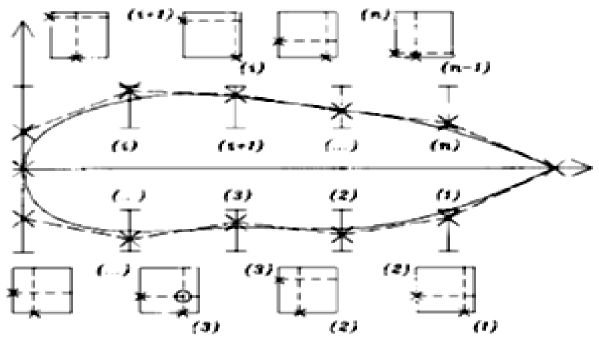


Fig. 2. Bezier Curve Representation. *The Bezier control points used in to define an airfoil. The "X" marks represent control point location and the "I" marks represent the range of movement for each control point.[2]*

ultimate choice but it provides "non-oscillating" shapes [5]." An example of this oscillatory behavior from [5] is shown in Figure 3. The second issue found was the weak representation of the leading edge of the airfoil, where the highest fidelity in the flow is required because it is the location of the greatest turn in the flow. One of the two major conclusions from [5] is the need for a parameterization that resolves these two issues. Naujoks asserted the need for future work in improving the airfoil parameterization, specifically to "involve *x*-coordinate in design optimization [5]."

The ES algorithm devised for this paper uses Bezier Control Points to encode the solution, however greatly reduces the restrictions on the movements allowed for the control points by allowing control point movment in both the *x* and *y* directions. This expands the search space to include solutions that cannot be represented by other means, while still ensuring solutions produce a smooth surface that can most likely be evaluated by the flow solver. This allowed movement of the control points in the *x* direction significantly reduces the oscillations in the airfoil surface, similar to the reductions in oscillations in polynomial interpolation through the use of unequally spaced Chebyshev nodes. The allowed movement in the *x* direction also allows the control points to be concentrated in certain regions, allowing more points and greater fidelity at the leading edge of the airfoil where it is needed. By making these changes, better solutions may be found compared to other algorithms and the solutions are not restricted to what the user is expecting the solution to look

TABLE I. SOLUTION ENCODING

Control Point	**Upper Surface *X* Value**	**Upper Surface *Y* Value**	**Lower Surface *X* Value**	**Lower Surface *Y* Value**
1	0	0.0574	0	0.0574
2	0	0.1356	0	-0.0091
3	0.3342	0.3566	0.3483	-0.25
4	0.431	-0.0142	0.5766	-0.105
5	0.7216	-0.0041	0.6884	-0.0052
6	1	-0.0911	1	-0.0911

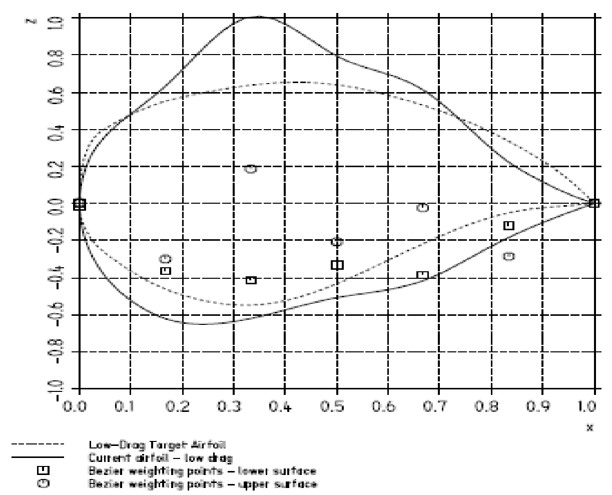
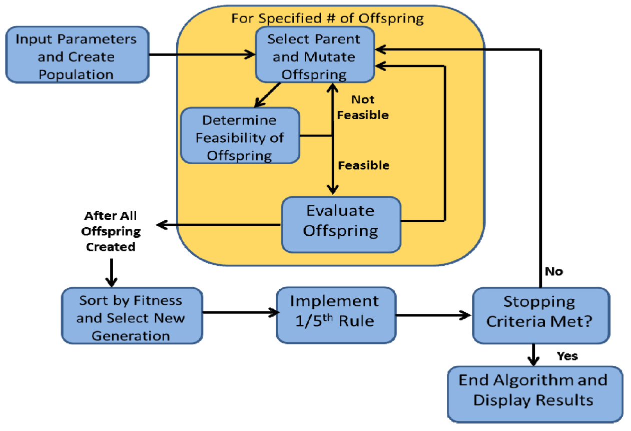


Fig. 3. An example of oscillatory behavior from [5]

like. It is entirely possible that the optimum solution is different than what the user is expecting, and imposing these restrictions may prohibit those solutions from being considered.

The only necessary restrictions placed on the movements of the control points were to ensure the airfoil kept a chord length of 1 and the surface of the leading edge was perpendicular to the flow to allow the flow solver XFoil to be used. The leading and trailing control points will remain at respective *x* coordinates of 0 and 1 to ensure a chord length of 1, however the vertical movement of the trailing point is unrestricted. The 2nd point from the leading edge on both the upper and lower surface will remain at an *x* value of 0 as well to ensure the tangent line to the leading edge of the airfoil is perpendicular to the flow. Without this restriction the flow solver could fail due to the sharp angle on the leading edge. These 2nd points on the upper and lower surfaces may move vertically in any amount as long as their *x* location stays at 0, so restrictions on possible airfoil geometries are inconsequential. If a CFD Euler or Reynold's Averaged Navier Stokes (RANS) solver were used instead of Xfoil, these restrictions could also be removed. All other points may move freely in any direction, which will allow virtually any airfoil shape to be represented by this method. An example solution encoding for an airfoil is listed below in Table 1, which simply gives the (*x*,*y*) coordinates of each Bezier Control Point. Six Bezier Control Points were used for each solution's upper and lower surfaces because this achieved a balance between sufficiently representing the airfoil and search efficiency. Up to 12 control points per surface were attempted, but generally increasing the number of control points above six did not improve results but greatly increased computation time. However, it may be beneficial to conduct a rough optimization using six control points to narrow in on a region of interest, then conduct a follow on optimization using more control points for an intensive search.



For Specified # of Offspring
Input Parameters and Create Population
Select Parent and Mutate Offspring
Determine Feasibility of Offspring
Not Feasible
Feasible
Evaluate Offspring
After All Offspring Created
Sort by Fitness and Select New Generation
Implement 1/5th Rule
Stopping Criteria Met?
No
Yes
End Algorithm and Display Results

Fig. 4. Flow Chart.

IV. EVOLUTION STRATEGIES ALGORITHM

Evolution Strategies (ES) was selected for the algorithm applied to this problem for a variety of reasons. First, because this is a continuous problem an optimizer suited for continuous problems was selected. A population based optimizer was also desired to produce multiple feasible solutions at the end of each run rather than just a single solution. The ES move operator is a much better fit to the Bezier Control Point representation than other algorithms due to the high mutation rate and ability to search outwardly more efficiently than Genetic Algorithms when the initial solutions are seeded airfoils. Because of the increased search space due to the solution encoding, pairing the ES move operator with the Bezier solution encoding was crucial in keeping reasonable computation times. Evolution Strategies also are more conducive to a micro-population, which is often used in airfoil optimization due to the high computational costs of the flow solvers. Past work has even shown that a population of one member may provide good results using ES algorithms [5]. For these reasons, Evolution Strategies was viewed as the best paradigm for this problem. The ES algorithm used very closely follows the canonical ES algorithm, which is kept simple to focus on the benefits of the airfoil parameterization used rather than complexities in the algorithm. Below is discussion on some of the major details of the algorithm. Figure 4 shows a flow chart of the algorithm's operation.

A. Initial Solutions

The initial solutions used for the first objective problem were seeded NLF0416 airfoils, matching the initial solutions used in [6]. For the second objective problem, two different sets of initial solutions were tried. The first were randomly generated Bezier Control Points, which produced random shapes that may or may not resemble an airfoil. Randomly generated initial solutions were used in D'Angelo's paper, however the method to generating the random initial solutions was not elaborated on. Random initial solutions produced good results, but needed extra computational time to move from the random shape to a generic airfoil shape. To reduce this unnecessary computation time, eight different subsonic airfoils were seeded for the initial solutions. The seeded

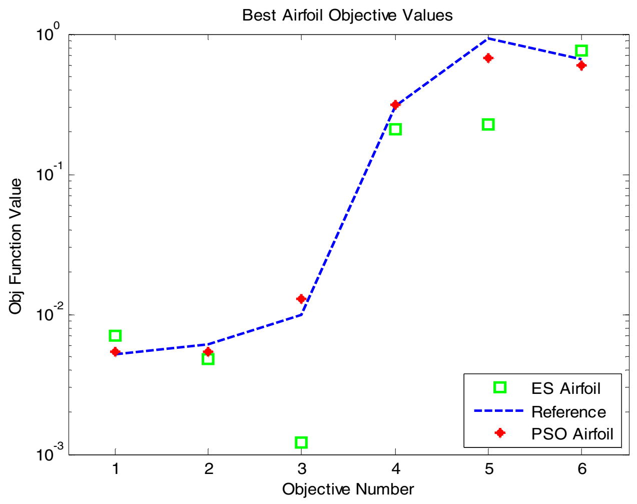


Fig. 5a. Airfoil Comparison. *The above graph compares the ES produced, PSO produced, and reference airfoils with respect to each objective function. The values for each objective are sought to be minimized.*

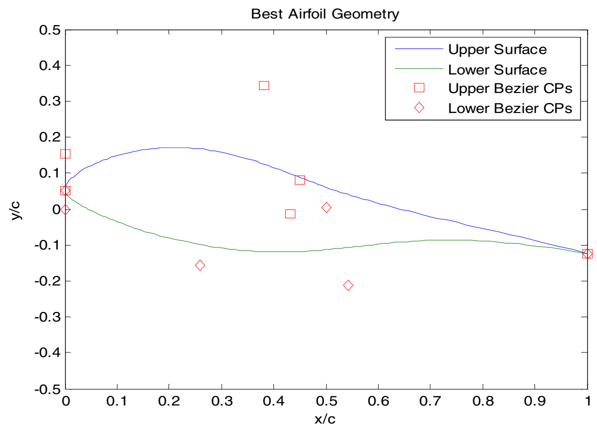


Fig. 5b. The ES produced airfoil geometry.

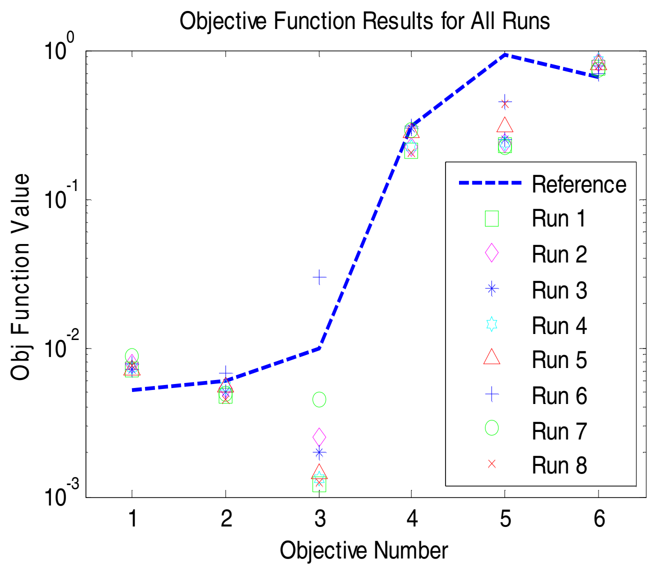


Fig. 5c. Results comparison between runs.

air:

LS(1)-0417, 64-210, 63-015), a NLF0416 airfoil, and a RAE100 airfoil. By seeding the initial solutions the computational time was reduced, but the quality of the final results had no noticeable difference from using random initial solutions.

A micro-population of eight airfoils was used for each objective problem. Micro-populations have been shown to provide comparable results with greatly reduced computational time when the function evaluations are particularly costly. Using a micro-population allowed the algorithm to perform well while keeping computation times reasonable.

B. *Move Operator*

The move operator for this algorithm is the canonical move operator for Evolution Strategies. The algorithm selects a parent through uniform probability and produces offspring by applying a mutation. The mutation consists of moving each of the Bezier Control points a random distance in the *x* and *y* directions subject to a normal distribution and a standard deviation σ. The leading two control points and the trailing edge control point are only mutated in the *y* direction for reasons stated in earlier in the paper. The one-fifth rule was applied to the standard deviation after every 50 objective function evaluations. "Sigma Restart" was also implemented, which reset the standard deviation to the initial value of 0.03 if sigma was reduced to below 0.0002. This would diversify the search if the standard deviation became so small that moves were ineffective, but would not lose superior solutions due to the elitism in the method of replacement.

C. *Replacement and Recombination*

Replacement was accomplished through (μ+λ) selection, where both the parents and offspring competed for selection in the next generation. This method provided an elitist aspect to the algorithm that allowed superior airfoils to remain in the population until they are replaced by improved offspring. Recombination was not used in the algorithm.

D. *Stopping Criteria*

The stopping criteria was completing the maximum number of generations. Because evaluating the performance of the algorithm was the main goal, completing the algorithm runs in a specified amount of time was of importance. Data was stored after each generation so the quality of the solutions and the cumulative computation time up to the specified generation could be easily found.

E. *Multiple Objectives*

The multiple objective functions were approached in different ways for the separate objective problems. For the first objective problem with six separate objective functions, a Pareto Curve would not be a suitable choice. For this reason, a simple sum of the normalized objective functions was used to combine all six into a single objective. Each objective function value was normalized against the reference value for the NLF0416 airfoil to solve scaling issues between the objectives. Weighted sums were attempted, but proved to have no noticeable difference from the non-weighted results. Wickramasinghe's paper used a Hyper-Volume approach to evaluate the six objectives at once, but this approach also came with a large computational cost.

For the second objective problem a Pareto Front was used to evaluate the two objective functions, which was the same approach used by D'Angelo. A fairly simple rank-based Pareto front was used for this algorithm, where the parents for the next generation of offspring were randomly selected from the first rank of the Pareto front from the previous generation. The concept of ε-dominance [4] was used to ensure the solutions were spread out across the front and did not cluster in certain areas. Archive sizes of 100, 300, and 500 were used with the larger archive sizes performing slightly better and with no significant difference in computation time. This approach to a Pareto front is fairly simple and straightforward, and leaves possible room for improvement such as biasing parent selection to areas less covered in previous generations, but is sufficient for the purposes of this paper.

F. *Objective Function Evaluation*

As stated earlier, the open source XFoil program developed by Dr. Drela was used as a subsonic flow solver for this algorithm. By calculating the airfoil upper and lower surface coordinates from the Bezier Control Points and inputting them with the flight conditions into XFoil, the flow solver will produce aerodynamic data such as the coefficient of lift, coefficient of drag, and point of flow separation. These XFoil outputs could then simply be read by the algorithm and translated into the appropriate objective function values.

Although XFoil is a fairly robust program, supplying mutated airfoils for evaluation sometimes resulted in problems. Reducing the restrictions on the Bezier Control Point movements exacerbated this problem, and for this reason several error control measures were implemented to protect the algorithm from XFoil convergence failures. The error control measures used in this algorithm were:

1. If the upper and lower surfaces of the airfoil crossed, the airfoil was discarded and another mutated airfoil would take its place.
2. Because XFoil uses a panel method, the angle between two adjacent panels must not be too great. If panel angles were too large, the algorithm would attempt three times to spline points to reduce the panel angles before discarding the airfoil and mutating a new one.
3. As a blanket error control measure, a timer was started each time XFoil was called upon for an evaluation. If the timer expired prior to XFoil outputting results, XFoil was terminated and a new airfoil was mutated and evaluated.

If future work uses an Euler or RANS solver instead of Xfoil, these control measures could be removed from the algorithm as long as adequate mesh generation and boundary condition techniques are used.

V. Problem One Results

The ES algorithm performed well in finding near-optimal solutions and performed extremely well in computational time compared to Wickramasinghe's PSO algorithm. Eight runs were performed to evaluate the algorithm while changing certain parameters. The specifics of the different runs are summarized below in Table 2.

A. *Objective Function Values*

The results for both the ES algorithm and the benchmark algorithm are summarized in Figure 5. Figure 7 shows an example of the converged XFoil results for the best solution. Each of the eight runs produced roughly equivalent results, with the exception of run 6, differing only slightly in objective function values. Run 6 produced inferior results simply because without sigma restart the 1/5th rule intensified the search prior to reaching the area of the optimum solution. Without sigma restart the moves reduced in size until they were approximately zero, effectively stopping the search

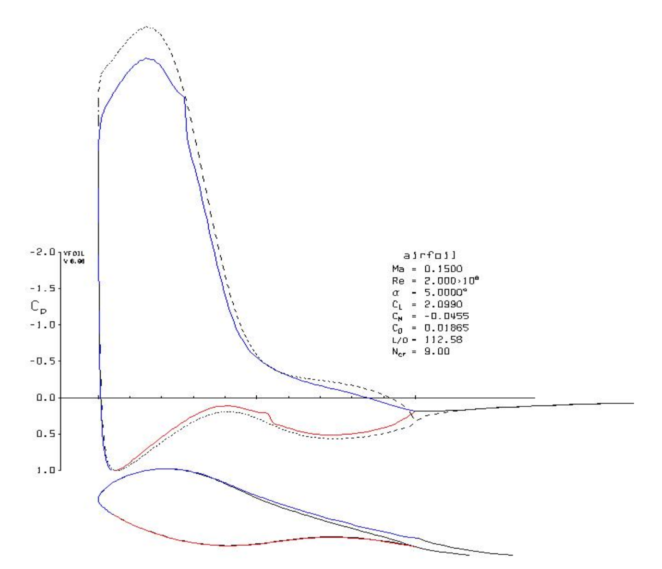


Fig. 7. XFoil converged solution example to airfoil in Figure 4b.

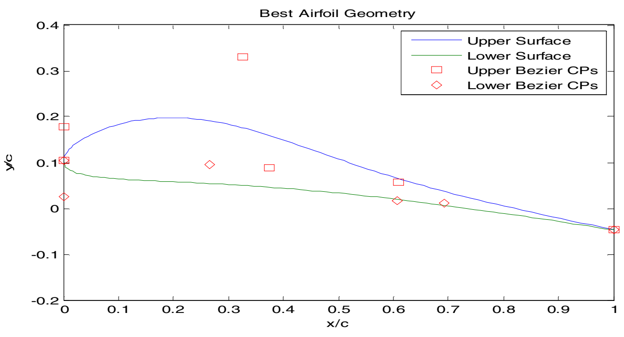


Fig. 8. Airfoil Geometry

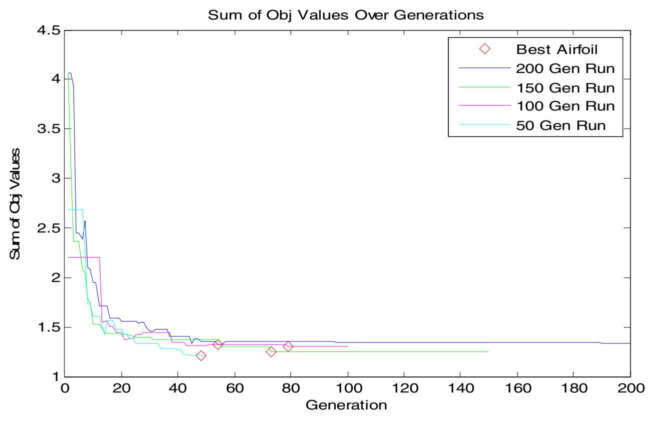


Fig. 6. Objective Function vs. Generations

before reaching optimum. The algorithm converged on a solution by about the 50th generation, so any subsequent generations generally added computation time without improving results. Figure 6 shows the sum of the objective functions over generations. Clearly by approximately generation 50 the algorithm converged to a near-optimal solution. Sigma restart improved the consistency of the algorithm by ensuring it did not easily get stuck in local optima, but did not significantly improve the final results or computation time. Using random initial solutions did not significantly impact the final results, but did slightly increase the computation time because the algorithm required approximately 10 generations to move from the random shapes to an airfoil shape equivalent to the NLF0416 airfoil. Even though the objective function results were very similar between the runs, the airfoil geometry was noticeably different in some of the runs. An example airfoil geometry from another run is shown above in Figure 8.

B. *Computation Time*

The computation time required for the algorithm was very reasonable considering the large computational cost of the flow solver. The fastest run consisting of 50 generations, which also produced the best objective function values, had a run time of 1.11 hours. All computations were performed in Matlab on a 2.0GHz dual-core processor. The PSO algorithm required approximately 36 hours to run on a 2.3GHz dual-core

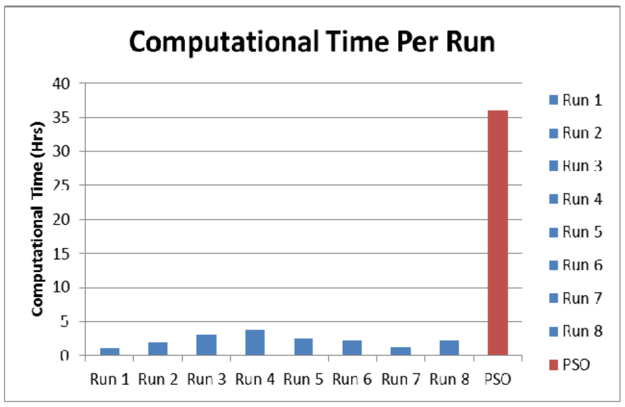


Fig. 9. Computational Time

TABLE III. RUNS AND PARAMETERS

	Run 1	Run 2	Run 3	Run 4
Generations	50	100	150	200
1/5th Rule	Yes	Yes	Yes	Yes
Sigma Restart	Yes	Yes	Yes	Yes
Initial Pts	Seeded	Seeded	Seeded	Seeded
	Run 5	Run 6	Run 7	Run 8
Generations	100	100	50	100
1/5th Rule	No	Yes	Yes	Yes
Sigma Restart	N/A	No	Yes	Yes
Initial Pts	Seeded	Seeded	Random	Random

TABLE II. OBJECTIVE FUNCTION RESULTS

Airfoil	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Obj. 5	Obj. 6
NLF0416	0.00516	0.00606	0.00982	0.30806	0.92314	0.6546
PSO	0.00536	0.00537	0.01281	0.3143	0.67285	0.5953
ES	0.00705	0.00481	0.00120	0.2087	0.2269	0.7560

processor. Because the XFoil flow solver accounted for over 80% of the computational time per run, the main difference in computation time between the ES and PSO algorithms came from the number of function evaluations required. The PSO algorithm required 10,000 function evaluations while the ES algorithm required about 2,000. Using a Hyper-Volume method for evaluating multiple objectives rather than a linear combination of the objective functions also increased run time of the PSO algorithm.

VI. PROBLEM TWO RESULTS

For problem two, the algorithm was generally unchanged with the exception of adding a Pareto archive instead of a parent population, and the parents were selected randomly from the first rank of this Pareto archive to create offspring. The algorithm still used a micro-population of eight members and made no changes to the solution encoding, move operator, or any other fundamental aspects of the algorithm. Fifteen different runs were completed in order to gain a sufficient amount of data to evaluate the performance. Three runs for each of the five scenarios below were completed:

1. 75 generations, 300 member archive
2. 150 generations, 300 member archive
3. 250 generations, 300 member archive
4. 150 generations, 100 member archive
5. 150 generations, 500 member archive

The data from these runs is too much to display in one figure, but in general the larger archives performed better with no change in computation time. Obviously, more generations led to better results, but in general returns diminished substantially after about 150 generations. There was no single run that dominated the other runs when comparing Pareto fronts, but the run with 150 generations and a 500 member archive performed well and will be used as the primary comparison against D'Angelo's results.

A. Pareto Front Comparison

Figure 11 shows the results from D'Angelo, both using a five parameter airfoil representation and a seven parameter representation. D'Angelo's best results using the five parameter representation was selected for comparison. The seven parameter result utilized Bayesian learning, and performed significantly better. The ES results for 150 generations and a 300 member archive achieved much better results in terms of the Pareto Front. Although not shown, each of the remaining 15 runs dominated at least a portion of the seven parameter Pareto front, and all but three of the runs completely dominated the five parameter Pareto front from D'Angelo.

B. Computation Time

The five parameter model from D'Angelo required approximately two hours to complete using just XFoil for a flow solver on a Pentium IV 2.8 GHz. The computational time for the seven parameter model was not stated, but it can be assumed with the higher degrees of freedom that the algorithm required at least as long as the five parameter model. The ES algorithm required approximately one hour to complete 50 generations, which resulted in a total computation time of 2.98 hours for the 150 generation, 300 member archive run. However, even though the entire algorithm required more time to complete, the Pareto front bettered the five parameter front in approximately 18 minutes and bettered the seven parameter front in approximately 25 minutes, as shown in Figure 11.

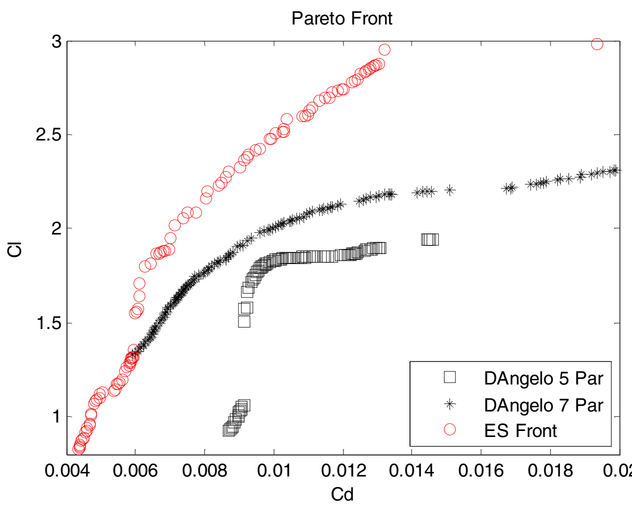


Fig. 10. Pareto Front Comparison

VII. Conclusions

The goals of this paper included developing and evaluating a less restrictive solution encoding for airfoil optimization, determining the appropriate optimization meta-heuristic and move operator for the solution encoding, comparing results obtained against previously published results in the subject, and determining the impact solution encoding can have on overall performance. The Bezier Control Point encoding was selected because of its ability to define almost any airfoil shape, and when coupled with an ES move operator many of the previous restrictions placed on the control point movements (when utilizing Genetic Algorithms or Particle Swarm

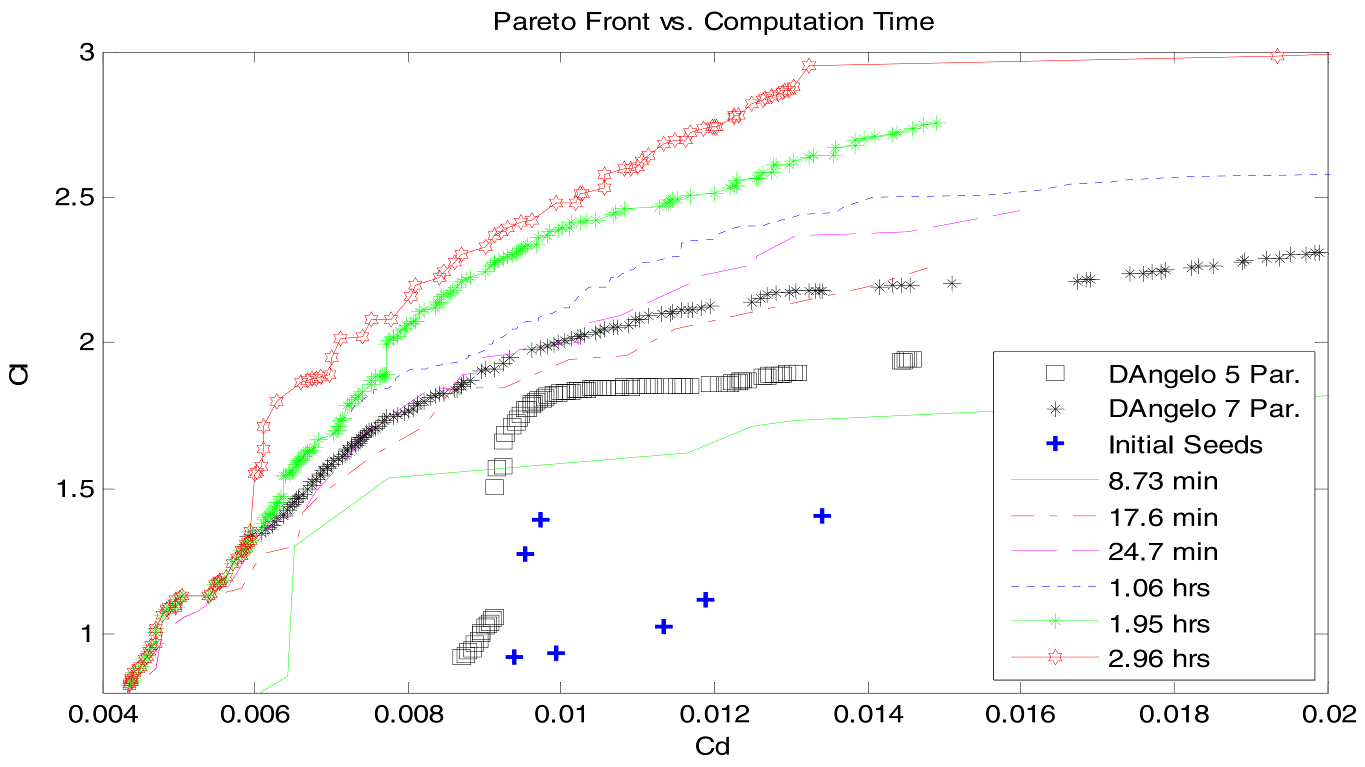


Fig. 11. Pareto Front vs. Computation Time

Optimizers) could be removed. The remainder of the algorithm was kept very simple and close to the canonical ES algorithm in order to focus on the impact of solution encoding in algorithm performance.

The results on both optimization problems showed that by using a more appropriate solution encoding and move operator, the performance of the optimizer could be greatly enhanced, both in objective function value and computation time. This is due to the ability to represent more optimal airfoil geometries than the other discussed encoding options, and a more efficient move operator for the specified problem. The oscillations in the airfoil surface and issues with weak representation of the leading edge identified in [5] were shown to not be a significant factor using the updated airfoil representation.

There are improvements that could be made to this algorithm, such as updating the many-objective evaluation method for problem 1 rather than using a sum of the six objective functions, improving the Pareto front selection code for problem 2, or adding some of the improvements that are common to ES algorithms over the canonical method. However, the results in this paper have shown that selecting the appropriate solution encoding and move operator are crucial and essential prior to adding more complex elements to the code. Bezier control points and an ES algorithm proved fitting for this specific airfoil optimization problem, and future work in this area would benefit aerodynamic optimization greatly as a whole.

Promising future work in this area includes updating the flow solver to an Euler or RANS solver that is able to tackle more complex flow problems, such as transonic flow. This update would eliminate a majority of the restrictions placed on the airfoil representation due to XFoil. More importantly, this would allow the algorithm and airfoil representation to be compared to a wide variety of more recent airfoil optimization articles which use CFD to evaluate the objective function.

NOMENCLATURE

C_D	=	coefficient of drag
C_L	=	coefficient of lift
C_{m0}	=	moment coefficient at zero lift
Re	=	Reynold's number
X_{tr}	=	normalized distance along the chord where turbulent flow starts.
α	=	angle of attack

REFERENCES

[1] D'Angelo, S., Minisci, E. "Multi-objective evolutionary optimization of subsonic airfoils by meta-modeling and evolution control." *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* Vol. 221, 2007, pp. 805-814

[2] Feinkos, G., Giannakoglou, K. "Inverse design of airfoils based on a novel formulation of the ant colony optimization method." *Inverse Problems in Engineering* Vol. 11, 2003, pp. 21-38.

[3] Jahangirian, A., Shahrokhi, A. "Inverse design of rransonic airfoils using genetic algorithm and a new parametric shape method." *Inverse Problems in Science and Engineering* Vol. 17, No. 5, 2009, pp. 681-99.

[4] Hajek, J., Smid, M., Szollos, A. "Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and e-dominance." *Advances in Engineering Software* Vol 40, 2009, pp. 419-430.

[5] Naujoks, B., Willmes, L., Haase, W., Back, T., Schutz, M. "Multi-point airfoil optimization using evolution strategies." *European Congress on Computational Methods in Applied Sciences and Engineering,*Barcelona Spain, September 11-14, 2000.

[6] Wickramasinghe, U., Carrese, R., Li, X. "Designing airfoils using a reference point based evolutionary many-objective particle swarm optimization algorithm." Paper presented at the *IEEE World Congress on Computational Intelligence,* Barcelona Spain, July 18-23, 2010.