Application of the Multi-objective Alliance Algorithm to a Benchmark Aerodynamic Optimization Problem

Valerio Lattarulo*, Timoleon Kipouros†, Geoffrey T. Parks‡
Engineering Design Centre
Department of Engineering
University of Cambridge
Cambridge, Trumpington Street, CB2 1PZ, UK
Email: [vl261, tk291, gtp10]@cam.ac.uk

Abstract—This paper introduces a new version of the multi-objective Alliance Algorithm (MOAA) applied to the optimization of the NACA 0012 airfoil section, for minimization of drag and maximization of lift coefficients, based on eight section shape parameters. Two software packages are used: Xfoil which evaluates each new candidate airfoil section in terms of its aerodynamic efficiency, and a Free-Form Deformation tool to manage the section geometry modifications. Two versions of the problem are formulated with different design variable bounds. The performance of this approach is compared, using two indicators and a statistical test, with that obtained using NSGA-II and multi-objective Tabu Search (MOTS) to guide the optimization. The results show that the MOAA outperforms MOTS and obtains comparable results with NSGA-II on the first problem, while in the other case NSGA-II is not able to find feasible solutions and the MOAA is able to outperform MOTS.

I. INTRODUCTION

Real-world optimization problems are generally characterized by several objectives in conflict. These problems share similar characteristics: they have many design variables and constraints, the objective and constraint functions are highly non-linear and contain many local optima. Metaheuristic approaches are often used to tackle such problems and many different methodologies have been defined for this purpose. Some of the most popular are: Genetic Algorithms [1], Differential Evolution [2], Simulated Annealing [3], Particle Swarm Optimization [4], Ant Colony Optimization [5] and Tabu Search [6]. These algorithms are used for single-objective optimization but multi-objective (MO) variants of all have been developed, with MO evolutionary approaches being the most widely used [7], [8].

The Alliance Algorithm (AA) is a recently developed single-objective optimization algorithm that has been applied successfully to various problems [9], [10], [11], [12]. A MO variant [13] was then developed, the performance of which has been compared with NSGA-II [14] and SPEA 2 [15]. That study revealed a certain complementarity because the three approaches offered superior performance for different classes of problems. Since then, a mixed-integer version of the MOAA with hybrid components has been developed. This

was able to outperform a hybrid version of NSGA-II on a satellite constellation refueling optimization problem [16]. The knowledge acquired in solving benchmarks and complex real-world problems led to the development of a new version of the MOAA which is presented in this paper.

This new MOAA version is applied to the optimization of the NACA 0012 airfoil section. The problem has two objectives: to minimize the drag coefficient and maximize the lift coefficient. The problem has eight design variables which modify the airfoil section shape. Xfoil [17] is used to evaluate the aerodynamic performance of each candidate airfoil. A Free-Form Deformation (FFD) tool is used to alter the section geometry. The results are compared, using the epsilon and hypervolume indicators and Mann-Whitney statistical test, with those obtained using NSGA-II and MOTS [18]: the former is a very well-known and widely used state-of-the-art MO optimizer, the latter has previously yielded several good results for this family of design problems [19], [20], [21], [22].

The rest of the paper is structured as follows: Section II introduces the airfoil optimization problem; Section III presents the new version of the MOAA; Section IV introduces the indicators and statistical test used for performance comparison; Section V describes the parameters used by the three algorithms; Section VI reports their performance and discusses the results; Section VII concludes the paper and suggests possible future work.

II. AIRFOIL OPTIMIZATION

The benchmark aerodynamic test case was defined in [23], where the FFD technique [24] is used for 2D airfoil geometry management. The scheme is based on trivariate Bernstein polynomials and can express global or local geometrical deformations. This in-house FFD implementation is then coupled with a quick and reliable flow evaluation tool, Xfoil, for the prediction of the lift, drag and moment coefficients. Xfoil is a freeware code developed by Drela [17] for the design and analysis of subsonic airfoils. It deploys a linear-vorticity stream function panel method coupled with a two-equation lagged dissipation integral boundary layer formulation and a
Karman-Tsien correction for calculated compressibility effects. The force coefficients are calculated via the momentum deficit in the wake. Reliable predictions can be obtained in the presence of attached flows and even small separated regions, in a fraction of the time required by a Navier-Stokes solver (about a minute for a complete lift curve), making it ideal for representing aerodynamic design problems realistically and demonstrating the capabilities of computational design tools.

The objective functions are defined in Eq. 1:

\[
F_{C_l} = \frac{C_l}{C_{ldatum}} \\
F_{C_d} = \frac{C_d}{C_{datum}}
\]

Here: \(C_l\) is the lift coefficient, \(C_d\) is the drag coefficient, \(C_{ldatum}\) is the datum value of \(C_l\), and \(C_{datum}\) is the datum value of \(C_d\). These datum values for the NACA0012 airfoil section are respectively 1.4644 and 0.03051.

Fig. 1 illustrates the stages during the aerodynamic shape optimization process for a 2D airfoil section. The FFD parameterisation technique [24] applies modifications with respect to a reference datum geometry. In our test problem the leading and trailing edge positions are fixed, in order to ensure the same airfoil chord length and angle of attack for all the candidate designs explored during the optimization. The eight design variables in the optimization study correspond to the horizontal and vertical movements of the FFD control points, as shown in Fig. 2.

\[
\begin{align*}
F_{C_l} &= \frac{C_l}{C_{ldatum}} \\
F_{C_d} &= \frac{C_d}{C_{datum}}
\end{align*}
\]

The complete formulation of the problem also includes two hard constraints on the thickness of the airfoil section at 25% and 50% of the chord, with respect to the datum, in order to ensure practicality of the optimal solutions by allowing sufficient space for the two spars along the span of the wing.

Two versions of the problem are formulated with different design variable bounds: ±0.3 and ±0.6. Hard constraints are imposed on the objective functions in order to limit the feasible region to plausible designs. In the tests, designs with values of \(F_{C_l} < -3\) or \(F_{C_d} > 2\) are automatically discarded. Without these constraints XFOIL accepts several designs that are not physically realistic.

### III. MULTI-OBJECTIVE ALLIANCE ALGORITHM

The MOAA is a metaheuristic optimization algorithm inspired by the metaphorical idea of a number of tribes struggling to conquer an environment that offers resources that enable them to survive. The tribes are characterized by two features: the skills and resources necessary for survival. Tribes try to improve skills by forming alliances, which are also characterized by the skills and resources needed, but these now depend on the tribes within the alliance. The two main search elements of the algorithm are the formation of alliances and the creation of new tribes. One MOAA cycle ends when the strongest possible alliances of existing tribes have been created. The algorithm then begins a new cycle starting with new tribes whose creation is influenced by the previous strongest alliances.

#### A. The Main Entities

Two main entities play important roles in the MOAA: tribes and alliances. A tribe \(t\) is a tuple \((x_t, s_t, r_t, a_t)\) composed of:

- a point in the solution space \(x_t\);
- a set of skills \(s_t = [s_{t,1}, s_{t,2}, \ldots, s_{t,N_S}]\) dependent on the values of the \(N_S\) objective functions \(S_i\) evaluated at \(x_t\):

\[
s_{t,i} = S_i(x_t) \quad \forall \; i = 1, \ldots, N_S
\]

- a set of resource demands \(r_t = [r_{t,1}, \ldots, r_{t,N_R}]\) dependent on the values of the \(N_R\) constraint functions:

\[
r_{t,i} = R_i(x_t) \quad \forall \; i = 1, \ldots, N_R
\]

- an alliance vector \(a_t\) containing the \(ID_s\)s of the tribes allied to tribe \(t\). Initially an alliance is composed of just one tribe, thus \(a_t(1) = ID_t\).
An alliance is a mutually disjoint partition of tribes. The tribes within an alliance perform actions as a unique entity. Each alliance \( a \) forms a new point \( x_a \) in the solution space defined by the tribes in the alliance. The sets of skills \( s_a \) and resource demands \( r_a \) of the alliance consist of the objective and constraint functions \( S \) and \( R \) evaluated at \( x_a \).

B. Algorithm Steps

The procedure followed by the MOAA can be divided into several steps which can be performed differently according to the problem at hand and user preference. A detailed description of the previous version of the algorithm can be found in [13].

1) Solution Generation: In the MOAA’s first cycle the tribes (solutions) are chosen randomly (with a uniform distribution):

\[ x_{t,i} = U(L_i, H_i) \quad \forall \ i = 1, \ldots, P \]  \hspace{1cm} (4)

where \( x_{t,i} \) is the \( i \)th component of tribe \( t \), \( L_i \) and \( H_i \) are respectively the lower and upper bounds on the \( i \)th variable, \( U(L_i, H_i) \) is the uniform distribution between \( L_i \) and \( H_i \); and \( P \) is the total number of variables.

In subsequent cycles, new solutions are sampled from a normal distribution with defined mean and standard deviation \( \sigma \) as follows:

\[
\begin{align*}
&\text{if } p_1 \leq P_1 : \quad x_t = b_r \\
&\text{if } p_1 > P_1 \land p_2 \leq P_3 : \quad x_{t,i} = N(b_{r,i}, \sigma) \\
&\text{if } p_1 > P_1 \land p_2 > P_3 : \quad x_{t,i} = b_{r,i}
\end{align*}
\]  \hspace{1cm} (5)

Here: \( p_1 \) and \( p_2 \) are random variables between 0 and 1, \( P_1 \) and \( P_3 \) are probability factors between 0 and 1; \( r \) is a random integer between 1 and \( N_P \), the number of Pareto-optimal (PO) solutions found; \( b_{r,i} \) is the normalized \( i \)th variable of the \( r \)th PO solution found.

One of the PO solutions found is chosen randomly and normalized; with probability \( P_1 \) a new solution is simply a copy of this PO solution; otherwise, with probability \( P_2 \), the variables are modified, on an individual basis, with a normal distribution with standard deviation \( \sigma \) around the chosen point, or, with probability \( 1 - P_2 \), they assume the same values as those of the chosen point. This cycle is repeated until \( N \) tribes have been generated.

An important feature of this new version of the MOAA is the adaptive nature of \( \sigma \); this parameter adaptively decreases in order to produce high diversity at the beginning of the optimization and low diversity at the end. This mechanism enhances the initial exploration of the solution space and the final convergence of the solutions already found. This task is accomplished with the following equations:

\[
\begin{align*}
\sigma_{\text{factor}} &= \frac{\sigma_{\text{end}}}{\sigma_{\text{init}}} \\
\sigma &= \sigma_{\text{init}} \cdot \sigma_{\text{factor}}
\end{align*}
\]  \hspace{1cm} (6)

Here: \( \sigma_{\text{init}} \) is the initial value of \( \sigma \); \( \sigma_{\text{end}} \) is the final value of \( \sigma \); \( E_c \) is number of function evaluations from the previous cycle; \( E_{t\text{ot}} \) is the total number of function evaluations. The value of \( \sigma \), starting with an initial value, is multiplied every cycle by \( \sigma_{\text{factor}} \) in order to approximately reach the final value. The value of \( \sigma \) is not updated if the size of the previous Pareto front is larger than the current Pareto front in order to prevent convergence in a few restricted areas. This adaptive process allows a gradual transition from global search to local search.

2) Token Phase and Ally Choice: In this phase an alliance/tribe (A/T) is chosen randomly and given the chance to forge an alliance by being given a token. Meanwhile all the other A/Ts wait their turn. The A/T \( t \) with the token chooses another tribe to become an ally, thus forming a new alliance.

3) Verification: In this phase an alliance (a point in solution space \( x_a \)) is created. In this case, \( x_a \) is made up of components drawn from the tribes within the alliance: given an alliance of \( N_a \) tribes, each component of \( x_a \) has a \( 1/N_a \) probability of being equal to the corresponding component of any tribe in the alliance. This part is similar to the uniform crossover used in GA and in ES [25] with the difference that the number of tribes involved in the creation of an alliance is not fixed: once an alliance is formed other tribes can join the alliance. There is an additional probability \( P_3 \) that the component is then modified as per the following equations:

\[
\begin{align*}
&c = U_d(1, N_a) \\
&\text{if } p_3 \leq P_3 : \quad x_{a,i} = N(x_{c,i}, \sigma_{\text{alliance}} \cdot d_{\text{tribes,i}}) \\
&\text{else : } x_{a,i} = x_{c,i}
\end{align*}
\]  \hspace{1cm} (7)

Here: \( c \) is the index of the chosen tribe within the alliance; \( p_3 \) is a random variable between 0 and 1; \( P_3 \) is a probability factor between 0 and 1; \( U_d(1, N_a) \) is the discrete uniform distribution between 1 and \( N_a \); \( x_{c,i} \) is the value of the \( i \)th component of the alliance; \( x_{c,i} \) is the value of the \( i \)th component of the chosen tribe; \( \sigma_{\text{alliance}} \) is a constant factor; \( d_{\text{tribes,i}} \) is the distance between the highest and lowest values of variable \( i \) of the tribes within the alliance. This operation is repeated until all the components of \( x_a \) are defined. As the standard deviation for the creation of an alliance depends on the distance between the highest and lowest values of the corresponding variable among the tribes within the alliance, the standard deviation for an alliance of tribes that are close together is small (local search) and for tribes that are far apart it is large (global search). Generally at the start of an optimization the tribes within an alliance are far apart and then they start to come closer together. This behavior can be viewed as an initial global search followed by progressively more localized search.

The new alliance will only be confirmed if at least one skill in \( s_a \) of \( x_a \) is better than one skill in \( s_{t_1} \) of the solution representing the A/T with the token \( x_{t_1} \) and one skill in \( s_{t_2} \) of the tribe chosen to become an ally.

The resource function \( R(x) \) plays no role here because the problem tackled in this particular study is unconstrained: the constraints are already inserted in the objective function and the other two algorithms were already using this particular objective function.

4) Alliance and Data Structure Update: There are two possible outcomes from the Verification Phase: the chosen
tribe joins the entity with the token forming a new alliance, or the tribe does not join and the new alliance is not confirmed. Next there is an update of the data structures necessary for the low level system to function, such as the necessity to provide a unique ID to the created alliances. The cycle termination conditions are also checked. The cycle finishes when each A/T has tried to form a new alliance with every other tribe and remains unchanged (because there is no advantage in changing). If this condition is not met, the algorithm returns to the Token Phase.

5) Selection of the Strongest Alliances and Termination: At the end of the interactions between tribes, many alliances will have been formed but only the strongest A/Ts will conquer the environment. Therefore the A/Ts selected are the non-dominated points in objective space. These correspond to the best solutions to the problem found thus far. They can be used as the input to another MOAA cycle or, if the algorithm has ended, they represent the final results.

There is a limit \( n \) to the number of best solutions saved in the archive of PO solutions. If the number of non-dominated solutions exceeds this, then all the solutions with at least one neighbor within a neighborhood distance \( d \) (in objective space) are eliminated. The initial value of \( d \) is 0 and then changes adaptively as follows:

\[
\alpha = \frac{N_f \cdot \text{range}_{\text{tot}}}{E_{\text{tot}}} \cdot \frac{n_p - n}{n} 
\]

Here: \( N_f \) is a constant factor; \( E_{\text{tot}} \) is the total number of solution evaluations; \( \text{range}_{\text{tot}} \) is the Euclidean distance in objective space between the maximum and minimum values of the PO solutions; \( n_p \) is the current number of PO solutions. This formulation takes into account that the reality that the rates of convergence have to depend on the number of function evaluations that can be afforded: slower convergence is possible if many function evaluations are possible; faster convergence is necessary with few function evaluations. Moreover it also depends on the actual ranges of the Pareto front: larger ranges require faster rates in order that the number of accepted solutions in the archive does not explode; conversely, smaller ranges require slower rates. In this way a balance around \( n \) is struck between the new PO solutions found by the algorithm every cycle and the solutions removed from the archive of PO solutions.

The MOAA is terminated when a specified limit \( E_{\text{tot}} \) on the number of solution evaluations is reached. The output of the algorithm is then the best solutions and the Pareto front found.

6) Extended Archive: Another innovation of this new version of the MOAA is the use of an Extended Archive. The archive with the PO solutions records only the non-dominated solutions, not considering many solutions that could help maintain diversity among solutions and the convergence of the algorithm. This issue is addressed by allowing other solutions to join the archive in the following way:

- For each dimension in objective space, the PO solutions and the solutions found in the cycle are sorted by increasing objective values.
- The distance (the difference in objective values for the relevant objective) between consecutive PO solutions is calculated as is the distance between the last PO solution (that with the worst objective value) and the solution from the current cycle with the worst value of that objective, in order to include the edge of the function.
- If the distance between two solutions is greater than the mean distance multiplied by a factor (discussed below), then non-dominated sorting of the solutions from the current cycle within that particular range is done and the non-dominated solutions from this group are added to the archive.

The factor changes adaptively increasing over time because the gaps between the solutions become smaller reaching similar values: increasing the value only large gaps (in comparison with the average) will be taken into consideration. The factor \( A_{f_i} \) for objective \( i \) starts with value of 1 and is then updated by adding the number of the new entries to the archive \( \text{sol}_{\text{new},i} \) resulting from this process divided by the number of solutions already in the archive \( n_p \).

\[
A_{f_i} = A_{f_i} + \frac{\text{sol}_{\text{new},i}}{n_p} \tag{9}
\]

This number \( n_p \) is updated only at the end of the process, thus the same value is used for all objectives. Using this method, the factor is updated only if there are gaps that have been filled with some solutions from the current cycle: if many solutions are added, then many gaps have been filled and the factor’s value can be increased faster. Furthermore, the factor update is closely related to the number of solutions in the archive: if there are few solutions then the number of possible gaps between them is small and the factor’s value can increase faster; conversely, if there are many solutions, there many possible gaps and the factor’s value has to increase slowly in order to fill them all.

IV. Indicators and Statistical Test

The performance measures chosen to evaluate the algorithms are the epsilon indicator [26] and hypervolume indicator [27] provided in the PISA package [28]. The epsilon indicator [26] makes use of the Pareto-dominance concept and measures, given a reference set of points (ideally the true Pareto front, if available), the minimum amount \( \epsilon \) necessary to translate all the points of the found Pareto front to weakly dominate the reference set. The hypervolume indicator [27] calculates the difference between the hypervolume of the space dominated by the found Pareto front and the hypervolume of the space dominated by a reference set (again, ideally the true Pareto front). This indicator needs a reference point which is dominated by all the found points in order to bound the hypervolume.

The statistical test chosen for the evaluation of results is the Mann-Whitney test, provided in the PISA package [28]. This is
a non-parametric rank-based test that can be used to compare two independent sets of sampled data. It outputs p-values that estimate the probability of rejecting the null hypothesis of the study question when that hypothesis is true. Here the p-values can be interpreted as the probability that the MOAA is superior to NSGA-II or MOTS only by chance. When \( H_0 \) holds, the two algorithms being compared have similar performance.

V. ALGORITHM PARAMETERS

The parameters used for NSGA-II are specified in [14]. The parameters used for MOTS are specified in [23]. The MOAA parameters used are shown in Table I. These have already proven their effectiveness with standard benchmark functions such as the ZDT [29] and DTLZ [30] families. Thus, these values are general and have not been overfitted to the current problem. Although some of these parameters need further investigation to find better general values and provide appropriate justifications, there are some exceptions:

- The number of tribes \( N \) is a number between 5 and 10, in order to allow sufficient algorithm cycles.
- The initial standard deviation \( \sigma_{\text{init}} \) is 0.3 because a normal distribution with this standard deviation behaves approximately like a uniform distribution (global search).
- The final standard deviation \( \sigma_{\text{end}} \) needs to be in the range 0.01–0.001 in order to allow local searches.
- The total number of PO solutions archived and output can be chosen by the user.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>6</td>
<td>Number of tribes</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>0.5</td>
<td>Probability 1 for the creation of tribes</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>0.2</td>
<td>Probability 2 for the creation of tribes</td>
</tr>
<tr>
<td>( \sigma_{\text{init}} )</td>
<td>0.3</td>
<td>Initial standard deviation</td>
</tr>
<tr>
<td>( \sigma_{\text{end}} )</td>
<td>0.001</td>
<td>Final standard deviation</td>
</tr>
<tr>
<td>( P_3 )</td>
<td>( 2/P )</td>
<td>Probability for the creation of alliances</td>
</tr>
<tr>
<td>( \sigma_a )</td>
<td>0.1</td>
<td>Std for the creation of alliances</td>
</tr>
<tr>
<td>( N_{\text{tot}} )</td>
<td>100</td>
<td>Total number of PO solutions</td>
</tr>
<tr>
<td>( N_f )</td>
<td>10</td>
<td>Factor for neighborhood evaluation</td>
</tr>
</tbody>
</table>

VI. RESULTS

In this section the performance of the MOAA is compared with that of NSGA-II and MOTS. The three algorithms were tested with limits of 6000 function evaluations per run. Two different tests (Tests 1 and 2) were performed with different design variable bounds: \( \pm 0.3 \) and \( \pm 0.6 \). Each test was repeated 20 times for each algorithm. The reference set used for the comparison comprised the PO solutions found by the three algorithms. The same parameter values were used in all the cases. The performances of the algorithms were compared using the mean and standard deviation of the indicators and the p-value of the statistical test. For both the indicators and p-values smaller values indicate better performance. In Test 1 the performances of the three algorithms are compared; in Test 2 only MOAA and MOTS are compared because NSGA-II was unable to find feasible solutions.

Table II shows the means and standard deviations of the epsilon and hypervolume indicators for Test 1. MOAA and NSGA-II obtain comparable results for the hypervolume indicator, while NSGA-II gives the best performance for the epsilon indicator followed by MOAA. MOTS obtains the worst results for both the epsilon and hypervolume indicators.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Epsilon</th>
<th>Hypervolume</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std</td>
</tr>
<tr>
<td>MOAA</td>
<td>0.0765</td>
<td>0.0352</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>0.0472</td>
<td>0.0091</td>
</tr>
<tr>
<td>MOTS</td>
<td>0.1794</td>
<td>0.0965</td>
</tr>
</tbody>
</table>

Table III shows the means and standard deviations of the epsilon and hypervolume indicators for Test 2. MOAA outperforms MOTS achieving means for both indicators that are half those achieved by MOTS. On the other hand, the indicators have lower standard deviations for the MOTS runs. This indicates that the MOTS runs are more consistent, but on average find worse performing designs.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Epsilon</th>
<th>Hypervolume</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std</td>
</tr>
<tr>
<td>MOAA</td>
<td>0.0672</td>
<td>0.0864</td>
</tr>
<tr>
<td>MOTS</td>
<td>0.2838</td>
<td>0.0646</td>
</tr>
</tbody>
</table>

Fig. 3 shows the results of the 20 runs in both the tests. In Test 1, the MOAA obtains both the best and the worst
PO solutions; the average MOAA Pareto front is comparable with that found by NSGA-II. MOTS finds only a few PO solutions comparable with those found by MOAA and NSGA-II; most of the solutions found by MOTS are clear dominated by MOAA/NSGA-II solutions. NSGA-II performs the most consistently, finding similar Pareto fronts each time. However, in the second test NSGA-II is not able to find feasible designs. In the second test, MOTS finds a number of Pareto fronts, but the solutions on these are dominated by solutions found by the MOAA. The MOAA finds fewer PO designs than MOTS but the solutions are qualitatively superior. An important difference between MOAA and MOTS is that in different runs MOTS finds several Pareto fronts that are quite similar to each other, while the MOAA finds different Pareto fronts which are generally better than all the fronts found by MOTS.

![Comparison of the best results from 20 runs in the two tests](image)

**Fig. 4.** Comparison of the best results from 20 runs in the two tests

Flow patterns for the datum geometry and randomly selected compromise optimal designs found by the MOTS and MOAA algorithms in Test 2 are presented in Fig. 5. Here, the pressure coefficient distributions are shown for the upper (negative values) and lower (positive values) airfoil surfaces, together with the boundary conditions and the corresponding direct flow coefficient values. The shapes of the optimal designs found by the two optimizers are similar. The physical mechanism that is activated effectively smothers out the adverse pressure gradient, and hence limits the amount of separated flow, resulting, of course, in improved aerodynamic performance.

### A. Discussion

These results have shown that the MOAA was able to find the best Pareto front in both the tests. NSGA-II could not find Pareto fronts in the second test, due to the infeasibility of much of the search space: the initial population consists only of infeasible solutions and the algorithm is not then able to reach a feasible region. The MOAA and MOTS overcome this problem in two different ways: MOTS is a ‘local search’ algorithm and needs fewer resources to find a feasible solution and then to explore the feasible region around that solution; MOAA needs a smaller initial population (it can explore different regions with fewer resources) and then it creates alliances favouring feasible solutions. In Test 2, the comparison between the two algorithms shows that the MOAA finds a smaller number of PO solutions than MOTS in individual runs but these solutions, in most cases, dominate the MOTS solutions. The performance of the MOAA is less consistent than the other two approaches (hence the generally higher indicator standard deviations), in particular in Test 2 when it does not always find the same Pareto front. Further studies are needed to understand if the lack of consistency is frequent or occasional and if this problem needs to be solved. This characteristic has also some advantages, however, because, in the case of problems with many infeasible regions, the MOAA is able to better exploit the solution space and find different Pareto fronts in different runs. Over multiple runs, this leads to better performance than MOTS, which almost always focuses on the same local minima. Overall, the MOAA can therefore be considered a good choice for the proposed problem.

### TABLE IV

MANN-WHITNEY TEST BETWEEN THE ALGORITHMS

<table>
<thead>
<tr>
<th>Test</th>
<th>Indicator</th>
<th>p(A &gt; B)</th>
<th>p(A &gt; C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>eps</td>
<td>6.43 × 10⁻⁴</td>
<td>0.9988</td>
</tr>
<tr>
<td></td>
<td>hyp</td>
<td>7.36 × 10⁻⁷</td>
<td>0.3625</td>
</tr>
<tr>
<td>Test 2</td>
<td>eps</td>
<td>8.37 × 10⁻⁵</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>hyp</td>
<td>1.02 × 10⁻⁴</td>
<td>0</td>
</tr>
</tbody>
</table>
VII. CONCLUSIONS AND FUTURE WORK

This paper has presented a new version of the MOAA applied to the optimization of the NACA 0012 airfoil section for minimization of drag and maximization of lift coefficients. Two versions of the problem were formulated with different design variable bounds, and the performance of the MOAA was compared with that of NSGA-II and MOTS using the epsilon and hypervolume indicators and Mann-Whitney statistical test. The results show that the MOAA outperformed MOTS in both the tests and obtained good results, which were comparable to those of NSGA-II, in the first test. Moreover, NSGA-II was unable to provide solutions in the feasible region in the second case. In Test 2, the MOAA was able to find considerably better solutions than MOTS. Overall, it is evident that the MOAA is a sensible choice of optimizer for this type of problem. Initially, four tests were chosen for this study with different design variable bounds ($\pm 0.3$, $\pm 0.6$, $\pm 0.8$ and $\pm 1.0$) but in the last two tests the MOAA exploited a model weakness in XFOIL, finding impressive Pareto fronts consisting of several designs that were, sadly, not physically realistic. In future work, these two remaining tests will be repeated once the problem in XFOIL has been solved, and the number of design variables and constraints will be increased in order to increase the difficulty of the problem to further test the capabilities of the optimizers under consideration.

REFERENCES
