Single and multi-objective UAV aerofoil optimisation via hierarchical asynchronous parallel evolutionary algorithm

L. F. Gonzalez, D. S. Lee, K. Srinivas and K. C. Wong
School of AMME
University of Sydney
Sydney, Australia

ABSTRACT

Unmanned aerial vehicle (UAV) design tends to focus on sensors, payload and navigation systems, as these are the most expensive components. One area that is often overlooked in UAV design is airframe and aerodynamic shape optimisation. As for manned aircraft, optimisation is important in order to extend the operational envelope and efficiency of these vehicles. A traditional approach to optimisation is to use gradient-based techniques. These techniques are effective when applied to specific problems and within a specified range. These methods are efficient for finding optimal global solutions if the objective functions and constraints are differentiable. If a broader application of the optimiser is desired, or when the complexity of the problem arises because it is multi-modal, involves approximation, is non-differentiable, or involves multiple objectives and physics, as it is often the case in aerodynamic optimisation, more robust and alternative numerical tools are required. Emerging techniques such as evolutionary algorithms (EAs) have been shown to be robust as they require no derivatives or gradients of the objective function, have the capability of finding globally optimum solutions among many local optima, are easily executed in parallel, and can be adapted to arbitrary solver codes without major modifications. In this paper, the formulation and application of a evolutionary technique for aerofoil shape optimisation is described.

Initially, the paper presents an introduction to the features of the method and a short discussion on multi-objective optimisation. The method is first illustrated on its application to mathematical test cases. Then it is applied to representative test cases related to aerofoil design. Results indicate the ability of the method for finding optimal solutions and capturing Pareto optimal fronts.

NOMENCLATURE

α	angle-of-attack
C_d	drag coefficient
C_l	lift coefficient
C_m	pitching moment coefficient
C_p	pressure coefficient
L/D	lift on drag ratio
M_∞	free stream Mach number
γ	specific heat ratio
Re	Reynolds number
t/c	thickness ratio

1.0 INTRODUCTION

Unmanned aerial vehicles (UAV) systems are increasingly becoming important topics for aerospace research and industrial institutions. But there are difficulties in the design of UAVs because of the varied and non-intuitive nature of the configurations and missions to be performed, like high or medium altitude surveillance, or combat environments for unmanned air combat vehicles (UCAVs).

Similar to their manned counterparts, the challenge is to develop trade-off studies of optimal configurations to produce high performance aircraft that satisfy mission requirements. Particular care must be taken with aerodynamic optimisation, due to the often very long endurance and high-speed dash requirements; even small improvements in drag can represent large weight savings over an entire mission.

The goal of this study is to address these issues from a multi-criteria standpoint.

Paper No. 3030. Manuscript received 19 October 2005, revised version received 10 April 2006, accepted 19 June 2006.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

Traditional methods for solving an optimisation problem use gradient based optimisation techniques[1-3]. An emerging technique for optimisation is evolutionary algorithms (EAs)[1,4,5]. EAs are based on Darwinian evolution; whereby populations of individuals, which represent the design variables, evolve over a search space and generate offspring by the use of different mechanisms such as mutation, crossover and selection. An attractive feature of EAs is that they evaluate multiple populations of points and are capable of finding a number of solutions in a Pareto set. EAs have been successfully applied to different aircraft, wing, aerofoil and rotor blade design and optimisation problems[2,6-10].

This work describes the design and implementation of a methodology for aerofoil design and optimisation. This methodology uses a robust evolutionary technique, which is scalable to preliminary design studies with higher fidelity models for the solution and is applicable to the design and optimisation of UAVs.

2.0 CONSIDERATIONS FOR DESIGN AND OPTIMISATION OF UAVs

When considering aerofoil shape optimisation the designer or team of designers has to define appropriate robust analysis and optimisation tools to explore the search space to find a final optimal solution or set of solutions. Sometimes the search space might be complex, involve multiple local minima, be highly non-linear, be non-differentiable and involve multiple objectives. Therefore, it is desirable to use a robust optimisation tool that is able to find optimal solutions.

3.0 EAS AND MULTI-OBJECTIVE EAS

3.1 EA fundamentals

Among the emerging techniques for optimisation and multi-objective problems are evolutionary algorithms (EAs). EAs are design and optimisation algorithms that mimic the natural process of 'survival of the fittest'. Broadly speaking they operate simply through the iterated mapping of one population of solutions to another population of solutions. This contrasts with conventional deterministic search techniques such as the simplex method; conjugate gradient method and others, which proceed from one given sub-optimal solution to another, until an optimum solution is reached. Evolutionary algorithms are not deterministic, so that for identical problems and identical starting conditions, the evolution of the solution will not follow the same path on repeated applications. It is for this reason that EAs fall into the category of stochastic (randomised) optimisation methods. Other stochastic methods include the Monte-Carlo approach, directed random walk and simulated annealing. The process of evolution of EAs is of course not completely random, because the performance of the algorithm would be no better than simple guessing, and at worst would be equivalent to complete enumeration of the parameter search space. Evolutionary algorithms work by exploiting population statistics to some greater or lesser extent, so that when newer individual solutions or offspring are generated from parents, some will have inferior characteristics and some will have superior characteristics. The general working principles of the iterated mapping then reduces to generating an offspring population, removing a certain number of inferior individuals, and obtaining the subsequent population. This can be summarised as the repeated application of two operators on the population, the variation operator (the generation of offspring) and the selection operator (the survival of the fittest)[5]. The various approaches to EAs in the literature only differ in the operation of these two operators.

The origin of evolutionary algorithms for parameter optimisation seems to have appeared independently in two separate streams, genetic algorithms (GAs) and evolution strategies (ESs). The genetic algorithm was founded on principles developed by Holland[11] in 1975, and a number of research topics both in theory and application were developed. It is generally accepted however that the modern GA was founded on optimisation research by Goldberg[5]. Goldberg's initial applications of the GA were in real-world topics such as gas pipeline control. The original GA technique revolved around a single binary string (or base- 2) encoding of the chromosomes, which are the genetic material each individual carries. The binary coded GA's variation operator is comprised of two parts, crossover and mutation. Crossover interchanges portions of parental chromosomes while mutation involves the random switching of letters representing elements of a chromosome. The selection operator has taken many forms, the most basic being the stochastic fitness-proportionate (or roulette wheel) method[5].

Genetic algorithms have developed significantly in the past decade, and these developments will be considered further throughout this discussion.

Evolutionary techniques other than GAs and ESs exist, such as evolutionary programming (EP) and genetic programming (GP). evolutionary programming has been applied to real coded optimisation problems[12-14], but has not seen widespread use in this field of aerofoil shape optimisation. For comparisons between ESs and EP refer to Bäck, *et al*[4,12]. Genetic programming is devoted to the generation of computer programs rather than number sets as the solution to a given problem. Together with GAs and ESs these methodologies form the basic four 'schools' of evolutionary algorithms (GAs, ESs, EP and GP).

Of particular interest to this work are evolution strategies (ES). ES were first developed by Ingo Rechenberg and Hans-Paul Schwefel in 1964. Their first algorithm worked using only two individuals, one parent and one offspring. Each individual was real coded; each problem variable was assigned a floating point value in the chromosome. The variation operator involved applying a random mutation to each floating point value in the parental chromosome to arrive at the offspring individual. The selection operator was entirely deterministic, and was simply the result of a competition between parent and offspring to determine which remained. In the standard nomenclature this strategy is denoted the (1+1) ES, the first digit indicating the number of parents, the final digit indicating the number of offspring and the '+' indicating competition between parents and offspring. From the beginning the ES has been designed almost exclusively with real coding in mind, as opposed to original GA variants where real parameter optimisation comes about by the piecewise interpretation of the binary chromosome associated with each individual. An evolution strategy would therefore seem a logical starting point for evolutionary optimisation using real coded problem variables.

Subsequent developments in ESs introduced multi-membered populations for both parents and offspring. The first algorithm of this type was the (μ+1) ES[4,15]. This worked by applying some variation operator to the parent population to produce a single offspring. The offspring is selected by determining whether it is better than the worst member of μ and, if so, it replaces the worst member? Both the (1+1) ES and the (μ+1) ES used deterministic control of the mutation size (variations applied to design variables), which were normally distributed when applied to real coded problems. The recent developments in both GAs and ESs have greatly modified their variation and selection operators, to the point where it is not clear whether such a nomenclature division is nowadays particularly justified. The main difference that exists between them today is still the predominance of adaptive mutations in ESs, which have made them very attractive for real coded optimisation, although GA research has produced some related concepts[9,16].

Some of the advantages of EAs are that they require no derivatives or gradients of the objective function, have the capability of finding globally optimum solutions among many local optima, are easily executed in parallel and can be adapted to arbitrary solver

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

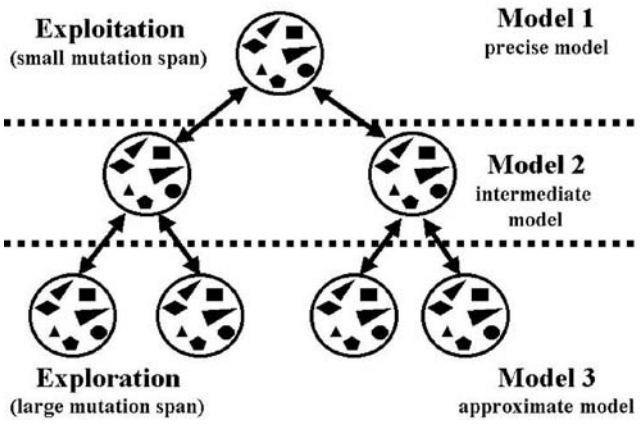


Figure 1. Hierarchical topology.

codes without major modification. Another major advantage of EAs is that they can tackle multi-objective problems directly (by considering vector fitnesses and not the traditional weighted aggregation method). It is shown below how this feature is used intensively for the capture of Pareto solutions to multi-objective optimisation problems.

4.0 ROBUST EA FOR DESIGN AND OPTIMISATION IN AERONAUTICS

One drawback of EAs is that they are slow to converge as they require a larger number of function evaluations than gradient based optimisation techniques and have poor performance with increasing number of variables. Hence the continuing challenge has been to develop robust and faster numerical techniques to overcome these difficulties and facilitate the complex task of design and optimisation in aeronautics. In that direction, the authors developed and use the hierarchical asynchronous parallel evolutionary algorithm (HAPEA) approach[10] with some extensions for multi-disciplinary and multi-objective analysis introduced recently. The foundations of the algorithm lie upon traditional evolution strategies and incorporate the concepts of covariance matrix adaptation (CMA)[17], hierarchical topology, asynchronous evaluation of candidate solutions, Pareto tournament selection for single and multi-objective problems, a constraint handling mechanism and parallel computing.

4.1 Hierarchical population topology

A hierarchical population topology, when integrated into an evolution algorithm, means that a number of separate populations are established in a hierarchical layout to solve the given problem, rather than a single 'cure-all' type single population layout. This method was proposed by Sefrioui[9] and is shown in Fig. 1. The bottom layer has a simple model and can be entirely devoted to exploration, the intermediate layer uses a more refined model and is a compromise between exploitation and exploration and the top layer a refined model concentrates on promising solutions. The combination of the different fidelity solvers helps to improve and speed-up the optimisation process.

4.2 Multi-objective EAs-Pareto optimality

Most evolutionary algorithms used today have been written with a single objective in mind, and the vast majority of early evolutionary algorithms were only written with a single objective capability.

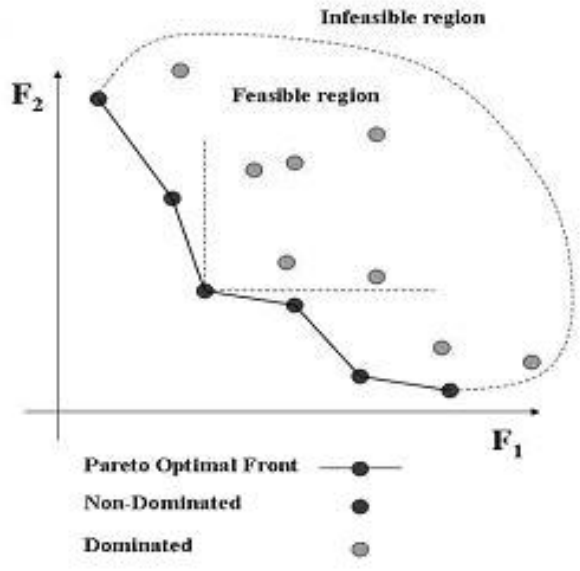


Figure 2. Pareto optimality.

Aeronautical design problems often require a simultaneous optimisation of inseparable objectives and associated number of constraints. A multi-objective optimisation problem can be formulated as:

$$Minimise: \quad f_i(x) \quad i = 1 \ldots N \qquad \ldots (1)$$

Subject to constraints:

$$g_j(x) = 0 \quad j = 1 \ldots M \quad h_k(x) \le 0 \quad k = 1 \ldots A \qquad \ldots (2)$$

Where f_i are the objective functions, N is the number of objectives and x is an N-dimensional vector where its arguments are the decision variables.

Most real world problems involve a number of inseparable objectives where there is no unique optimum, but a set of compromised individuals known as Pareto optimal (or non-dominated) solutions. The algorithm uses the Pareto optimality principle where a solution to a multi-objective problem is considered Pareto optimal if there are no other solutions that better satisfy all the objectives simultaneously. The objective of the optimisation is then to provide a set of Pareto optimal solutions that represent a trade-off of information among the objectives.

Formally, the Pareto optimal set (Fig. 2) can be defined as the set of solutions that are non-dominated with respect to all other points in the search space, or that they dominate every other solution in the search space except fellow members of the Pareto optimal set. For two solutions x and y (in minimisation form):

$$\begin{aligned} &rel(x,y) = x \text{ dominates } y \text{ if: } f_i(x) \le f_i(y),\ i = 1 \ldots M \\ &rel(x,y) = x \text{ nondominated w.r.t. } y \text{ if: } f_i(x) \le l \ge f_i(y)\ i = 1 \ldots M \\ &rel(x,y) = y \text{ dominates } x \text{ if: } f_i(x) \ge f_i(y),\ i = 1 \ldots M \end{aligned} \qquad \ldots (3)$$

For a problem in M objectives, this is called the 'relationship' operator. In practice we compute an approximation to the continuous set, by assembling $ParetoSet = \left[x_1^*, x_2^*, x_3^* \ldots . x_\mu^* \right]$

As EAs consider multiple points simultaneously, they are capable of finding a number of solutions in a Pareto set. Pareto selection ranks the population and selects the non-dominated individuals for

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

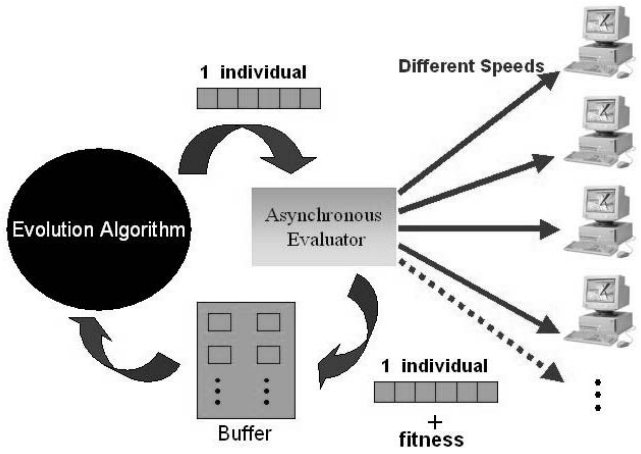


Figure 3. Parallel computing and asynchronous evaluation.

the Pareto front. A comprehensive theory and literature review on multi-objective EAs (MOEAs) is presented in Refs 1,18.

4.3 Parallelisation of the algorithm

The algorithm is similar to the hierarchical hybrid parallel multi-objective evolutionary algorithm (pMOEA) model described in Refs 19 and 20. It uses a master-slave pMOEA and incorporates the concept of isolation and migration through hierarchical topology binary tree structure where each level executes different MOEAs parameters (heterogeneous). In the optimisation procedure considered here, the parallel environment used is a cluster of PCs, wherein the master carries on the optimisation process while remote nodes compute the analysis solver. The message-passing model used is the parallel virtual machine (PVM)[21].

The algorithm has been tested in a cluster of heterogeneous CPUs, RAMs, caches, memory access times, storage capabilities and communication attributes. In this work a cluster of computers at the School of Aerospace, Mechanical and Mechatronic Engineering at the University of Sydney is used, the cluster can be configured with up to 25 machines with performances varying between 2·0 and 2·4GHz. A schematic of the parallelisation approach with asynchronous evaluation is shown in Fig. 3.

4.4 Asynchronous solution

When considering the solution to multi-objective and multidisciplinary optimisation cases several problems arise, as many methods of solution used in engineering today may take different times to complete their operation[3]. The classic example of this is the modern CFD solver. With a typical industrial code used for external aerodynamic analysis of airplanes, the time for the residual of the solution to converge to a specified level (either machine zero or an arbitrarily selected higher value) can vary over a significant range. The time taken for an iterative solution of non-linear partial differential equations is strongly dependant upon geometry. The traditional genetic algorithm and evolution strategy have mostly used a generation-based approach. A difficulty with generation-based models is that they create an unnecessary bottleneck when used on parallel computers. If the population size is approximately equal to the number of processors, and most of the candidates that are sent for solution can be successfully evaluated, then some processors will complete their task quickly with the remainder taking more time. With a generational approach, those processors that have already completed their solutions will remain idle until all processors have completed their work[19,20]. The approach used is to ignore any concept of a generation based solution. This approach is similar to work by Wakunda and Zell[22] and other non-generational approaches, however the selection operator is quite different, as it couples one-by-one (steady-state) function evaluation with a direct multi-objective fitness criterion. Whilst a parent population exists, offspring are not sent as a complete 'block' to the parallel slaves for solution. Instead one candidate is generated at a time, and is sent to any idle processor where it is evaluated at its own speed. When candidates have been evaluated, they are returned to the optimiser and either accepted by insertion into the main population or rejected. This requires a new selection operator because the offspring can not now be compared one against the other, or even against the main population due to the variable-time evaluation. The recently evaluated offspring will be compared to a previously established rolling-benchmark and if successful, it will be replaced (according to some rule) by a pre-existing individual in the population. Implementation of this benchmark via a separation of evaluation buffer (B), provides a statistical 'background check' on the comparative fitness of the solution. The length of the buffer should represent a reasonable statistical sample size, but need not be too large; approximately twice the population size is more than ample. When an individual has had a fitness assigned, it is then compared to past individuals (both accepted and rejected) to determine whether or not it should be inserted into the main population. If it is to be accepted, then some replacement strategy is invoked and it replaces a member of the main population. The 'replace-worst-always' method is exclusively used in this work.

4.5 Pareto tournament selection

The authors implement the on-the-fly selection operator by means of a Pareto tournament selection operator. To implement an optimisation algorithm that is equally applicable to both single and multi-objective problems, a suitable selection operator capable of handling either situation must be developed. The authors propose an extension of the standard tournament operator popular in many approaches[5,22,23].

Most evolutionary algorithms configured for multi-objective optimisation currently use the non-dominated sorting approach. This is a straightforward way to adapt an algorithm, is designed as a single objective optimiser, into a multi-objective optimiser, and is used by many researchers[1,5,24]. The problem with sorting approaches is that the method is not a fully integrated one. Briefly, a sorting method works by computing the set of non-dominated solutions among a large statistical sampling (either a large population or previous data), and assigning these solutions to rank one. Then ignoring these points, the process is repeated until a 'second' Pareto front is found, and this is assigned to rank two. This process continues until all points are ranked, and then the value of the rank is assigned to the individual as a new single objective fitness. A problem arises now on whether it is fair to assign individuals in the second rank numerically half the fitness of the first, and whether the third rank deserves a third of the fitness of the first. This poses a dilemma regarding the level of equality present among the solutions, as often solutions with excellent information may lie adjacent to, but not in, rank one. To solve this 'artificial scaling' problem, it is possible to introduce scaling, sharing and niching schemes[1], however all of these require problem-specific parameters or knowledge, even in adaptive approaches. It is of course always desirable to compose an algorithm that does not introduce such unnecessary parameters.

The current operator is a novel approach in that it requires no additional 'tuning' parameters, works seamlessly with the asynchronous selection buffer (B), and is very easy to encode. As illustrated in Fig. 4, to determine whether a new individual x is to be accepted into the main population, it is compared with the members of a small subset of the buffer $Q = [q_1, q_2, ..., q_n] \cdot Q$ is assembled by selecting individuals from the buffer, exclusively at random, until it is full. Then the algorithm checks to ensure that the new individual is not

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

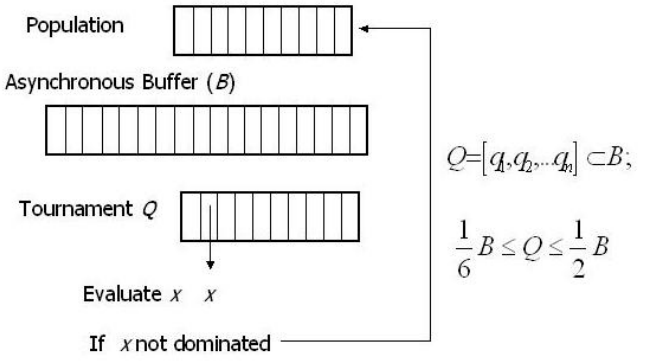


Figure 4. Pareto tournament.

dominated by any in the tournament. If this is the case, then it is immediately accepted, and is inserted according to the replacement rules. The only parameter that needs to be determined in advance is the tournament size, a parameter that would exist in a single objective optimisation anyway. Selection of this parameter requires a small amount of problem specific knowledge, and should vary between Q = ½*B* (strong selective pressure) and Q = ⅛*B* (weak selective pressure). The optimiser is not overly sensitive to this value, provided the user errs on the side of weak selective pressure (smaller tournaments) in the absence of better information. The egalitarian approach to the tournament (by selecting individuals at random) ensures good diversity among the selected individuals; no niching or forced separation of individuals has been found necessary. It can also be seen that in the event the fitness vectors have only one element (a single objective optimisation), this operator simplifies to the standard tournament selection operator[5,15].

4.6 Constraints handling mechanism

Engineering problems are usually subject to several constraints. Within the algorithm these are specified by the user. They may take the form of simple upper and lower bounds on the object variables, but many more complicated constraints exist and these must be satisfied during the optimisation process. Problems are often posed so that only certain combinations of object variables can be considered or their bounds are not simply 'upper' and 'lower' but also 'not between' and 'not if'. Object variables merely represent the genotype (numerical representation) of the individual, and further constraints will probably exist on the phenotype (physical representation) of the individual as well. Constraints such as these may be imposed on a particular solution such as weight, geometry or some other physical characteristic, which is undesirable. Often whether there has been an excursion from the phenotypic problem constraints, this can only be determined after the fitness function has been applied, and this may slow overall performance.

Two basic methods of handling constraints are considered in this work; the 'rejection' method and the 'penalty' method. The rejection method simply involves rejecting any individual, which is not compliant with the constraints, by not allowing it an opportunity to contest insertion into the main population. The merit of the rejection method is that no penalty scheme needs to be devised for handling individuals that are out of bounds, and therefore only solutions, which satisfy the constraints fully, are admitted. The disadvantage with this approach is that individuals which are close to the boundary but not within it are rejected out of hand, even though they may contain useful genetic information.

The penalty method involves adding some penalty fitness to *f* which (in the context of a minimisation problem) reduces its fitness with respect to other individuals in the population, reducing the likelihood that it will be selected next time. For example, if a certain solution-dependent value *s* must be less than a given value *v* a penalty function can be constructed as follows:

$$f' = f + h(s-v).(s-v)^2 \quad \ldots (4)$$

Where *f′* is the (possibly) penalised fitness, *f* is the original fitness and *h*(...) is the Heaviside function. The advantage of the penalty method is that individuals with good genetic material can be allowed to converge from outside the boundary to inside the boundary, if possible. In this work, multi-objective fitnesses are penalised by adding equal values of the penalty to each fitness value. This ensures that between two otherwise equal solutions, one, which is penalised, can never dominate one which is not. The disadvantage with the penalty method is of course that the penalty function needs to be devised with some care, especially considering there may be many such functions to devise. In the example given, a question of weight arises: Should a more severe term such as $10^5(s-v)^3$ or a less severe term such as $\frac{1}{4}|s-v|$ have been used instead. In these cases problem specific knowledge is required, so the user must make a 'best guess' of the penalty to apply, or run a number of cases to gain some experience with the particular case involved. Possibly the best compromise is the use of both rejection and penalty methods together, so that rejection is used on solutions that are obviously not feasible and will not lead to further improvement, while penalties are applied to solutions that show promise, but exceed allowable limits by a small margin. These two methods are used throughout this work, and they will be referred to as 'hard' (rejection) and 'soft' (penalty) bounds respectively.

5.0 PERFORMANCE ON MATHEMATICAL TEST CASES

In this section the performance of the method is evaluated on multiple test cases to determine possible performance levels of the algorithm in real situations. The test cases are all posed as easily closed-form mathematical functions.

There are a great number of test cases for single and multi-objective results presented in the literature, and are almost as varied as the number of evolutionary algorithms that have been written.

The test cases consider the convergence of the algorithm to a known Pareto front. It is difficult to directly measure the performance of an algorithm in completing a multi-objective task as the metrics to define multi-objective performance are not clear. Some work using closed-form expressions has been performed by Deb[1]. It can be determined by inspection if the entire population has converged to the Pareto front. Because inevitably a discrete approximation to a smooth surface (in the case of mathematical test functions) is considered, opinions vary regarding the fidelity of approximation that is achieved. The maximum number of points available to correctly model the Pareto front in this work is equal to the population size.

It is also desirable to have a good spread of points along the Pareto front, however this can be defined both in objective variable space and in fitness space. Because there is an explicit distortion caused by mapping between the two spaces, a method that provides an even distribution in fitness space will not provide this in objective variable space, and *vice-versa*. In this work, the authors have adopted the fitness space approach, due to the belief that the design engineer invariably would like to obtain a distribution of 'figures of merit' in advance, rather than a distribution of actual geometries. In any case, it is quite difficult to subsequently define a 'density distribution' of solutions along the Pareto front with a hope for providing an even positioning of solutions. Various metrics have been tried, including 'nearest neighbour' approaches, but invariably none work completely in the presence of discontinuities which cannot be known

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

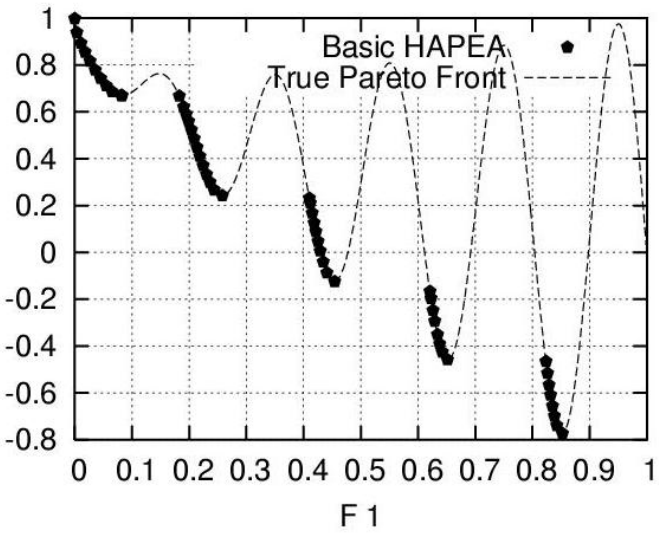


Figure 5. Pareto front ZDT3.

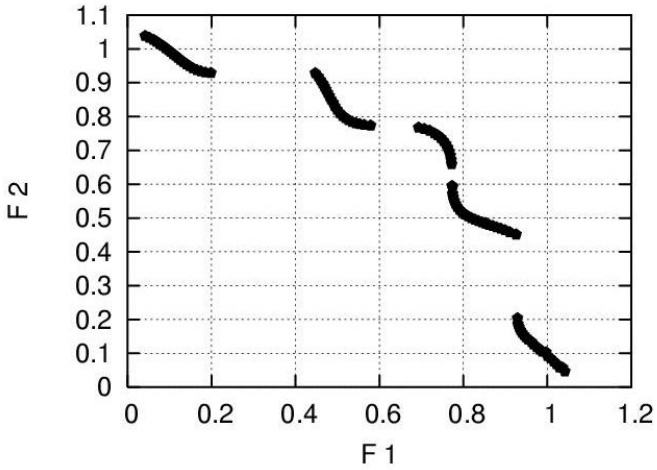


Figure 6. Pareto front TNK.

in advance. The study on multi-objective test functions is based on works by Deb[1] and Coello-Coello *et al*[18]. In these references a comprehensive set of test functions that highlight the difficulties of a multi-objective algorithm to converge to the true Pareto front is provided.

5.1 Mathematical test problems

For illustration purposes, this paper presents only two mathematical functions of interest. These are:

ZDT3

$$f_1(x) = x_1$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\mathrm{Sin}(10\pi f_1) \qquad \ldots (5)$$

$$f_2(x) = g.h$$

$$0 \le x_i \le 1$$

TNK

$$f_1(x) = x_1$$

$$f_2(x) = x_2$$

$$C_1(x) \equiv x_1^2 + x_2^2 - 1 - 0{\cdot}1\mathrm{Cos}\left(16\mathrm{Arctan}\frac{x_1}{x_2}\right) \ge 0$$

$$C_2(x) \equiv (x_1 - 0{\cdot}5)^2 + (x_2 - 0{\cdot}5)^2 \le 0{\cdot}5 \qquad \ldots (6)$$

$$0 \le x_1 \le \pi,$$

$$0 \le x_3 \le \pi$$

The first ZDT3 is an example of an optimal Pareto front that is discontinuous, and thus would not be able to be treated by a deterministic optimiser, which uses the gradient of the Pareto surface to traverse the front. It has five separate convex bands. The second TNK is an example of a constrained problem that has a discontinuous optimal Pareto front.

The first problem is an unconstrained problem with 30 variables and uses a population size of 150 and discrete recombination; the second problem has only two variables and uses a population size of 100. Both problems use a discrete recombination approach to enable the capture of complex Pareto fronts.

5.2 Numerical results

As illustrated in Figs 5 and 6 it is shown that the algorithm has correctly distributed all individuals across the Pareto front for both test problems. Furthermore it is seen that even without any explicit niching or redistribution, individuals in the population are very evenly spread across the front.

6.0 APPLICATIONS

The methodology has been applied to aerospace design problems with different complexities including inverse and direct problems for aerofoil design, multi-element aerofoil design, multidisciplinary/multi-objective wing and aircraft design[10]. In the following, single and multi-objective aerofoil shape optimisation problems are considered. Initially, the analysis tools are descibed and then the method is applied to five representative test cases in aerofoil design.

6.1 Aerodynamic analysis tools

In this work, two types of aerodynamic analysis tools are considered; a panel method (XFOIL)[25] and an Euler + Boundary Layer solver (MSES)[26].

The XFOIL software comprises a higher order panel method with coupled integral boundary layer. In some of these studies, free transition points for the boundary layer is allowed. Because some candidates may in fact cause locally sonic flow (transonic aerofoils) these will not be properly resolved by a panel method. To prevent this situation, the sonic pressure coefficient C_P^* is calculated from:

$$C_P^* = \frac{2}{\gamma \mathrm{M}_\infty^2}\left\{\left[\frac{1 + \frac{1}{2}(\gamma-1)\mathrm{M}_\infty^2}{1 + \frac{1}{2}(\gamma-1)}\right]^{\frac{\gamma}{\gamma-1}} - 1\right\} \qquad \ldots (7)$$

Where specific heat ratio γ = 1·4 for air and M is the free stream Mach number. All the reported surface $C_{P,i}$ values are examined, and if any are found to exceed the sonic value ($C_{P,i} < C_P^*$) then the candidate is rejected immediately. Also, any candidate which fails to converge the boundary-layer solution is also rejected without further consideration.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

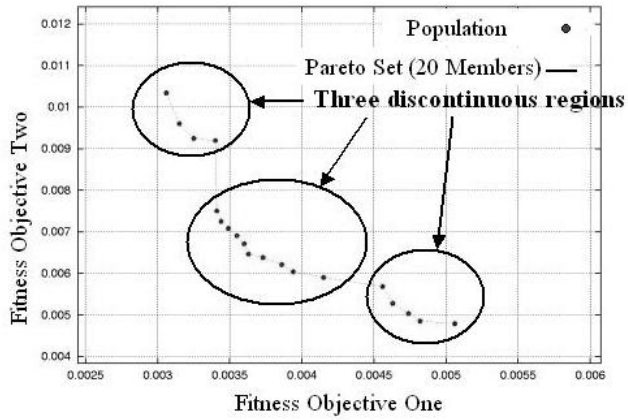


Figure 7. Pareto front for aerofoil design.

The MSES software[26] is a coupled viscous/inviscid Euler method for the analysis and design of multi-element/single-element aerofoils. It is based on a streamline-based Euler discretisation and two-equation integral boundary layer formulation which are coupled through the displacement thickness and are solved simultaneously by a full Newton method.

6.2 Representation of design variables

In most of the cases considered, the aerofoil geometry is represented by the combination of a mean line and thickness distribution, which is very common concept in classical aerodynamics[27]. Both lines are represented by Bézier curves with leading and trailing edge points fixed at (0·0, 0·0) and (1·0, 0·0) respectively, and a variable number of intermediate control points whose *x*-positions are fixed in advance and whose y-heights form the problem unknowns.

6.3 Two objective UAV aerofoil section design

6.3.1 Problem definition

This first case considers the detailed design of a single element aerofoil for a low-cost UAV application. There are two subsonic design points that are considered for optimisation; one for loitering flight and another for rapid-transit flight.

6.3.2 Design variables and fitness functions

As aerofoil design variables, four and five free control points are taken for the mean line and thickness distribution respectively. The two fitness functions to be optimised are defined as:

$$\min(f_1): f_1 = c_{d_{Transit}} + Penalty$$
$$M_\infty = 0{\cdot}60, \quad Re = 14{\cdot}0 \times 10^6 \qquad \ldots (8)$$

$$\min(f_2): f_2 = c_{d_{Loiter}} + Penalty$$
$$M_\infty = 0{\cdot}15, \quad Re = 3{\cdot}5 \times 10^6 \qquad \ldots (9)$$

6.3.3 Design constraints

The thickness of each aerofoil must exceed 12% ($t/c \geq 0{\cdot}12$) and the pitching moment must not be more severe than –0·065 ($c_m \geq -0{\cdot}065$). Both constraints are applied by equally penalising both fitness values via a linear penalty method. In addition, aerofoils generated outside the thickness bounds of 10% to 15% are rejected immediately, before analysis.

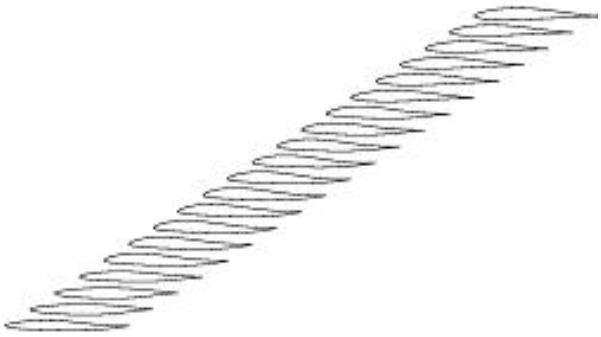


Figure 8. The ensemble of Pareto aerofoils.

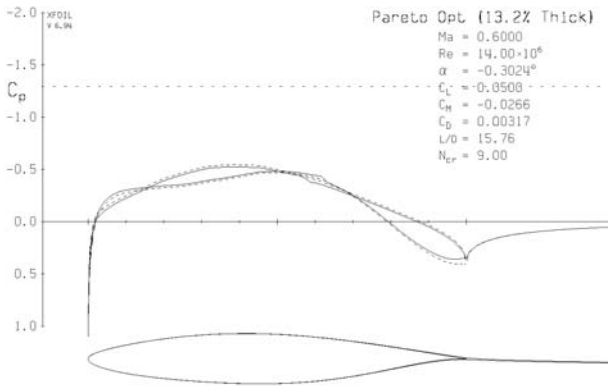


Figure 9. Objective one optimal aerofoil — Cruise C_P distribution.

6.3.4 Implementation

XFOIL is utilised as an aerodynamic analysis tool and the optimiser is configured hierarchically with the following settings:

Top Layer

A population size of 20, intermediate recombination used between two parents, 119 panels used by the solver.

Middle Layer

A population size of 20, discrete recombination used between two parents, 99 panels used by the solver.

Bottom Layer

A population size of 10, discrete recombination used between two parents, 79 panels used by the solver.

6.3.5 Numerical results

This case was run for 5,300 function evaluations of the head node, and took approximately four hours on a single 1·0GHz processor. The resulting Pareto set is shown in Fig. 7. The aerofoils comprising the Pareto front are shown in Fig. 8. It can be seen that classical aerodynamic shapes have been evolved, even considering that the optimisation was started completely from random and the evolution algorithm had no problem specific knowledge of appropriate solution types. Three aerofoils are selected for further consideration from the Pareto front of 20 members (numbers 2, 10 and 20) to illustrate the two objective extremes and compromise geometry. Figure 9 shows an objective one optimal aerofoil in the transit flow regime, and it can be seen that it has evolved a conventional low-drag

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

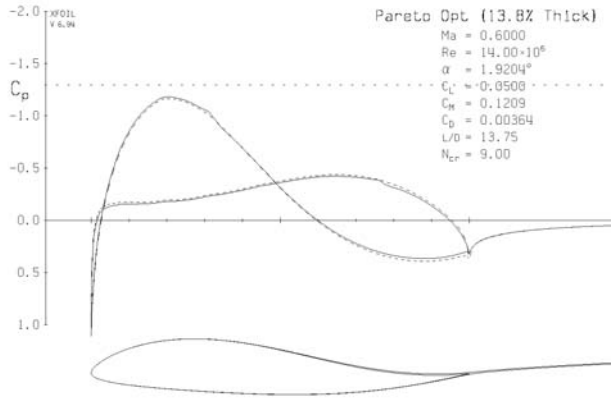


Figure 10. Compromise aerofoil — Cruise C_P distribution.

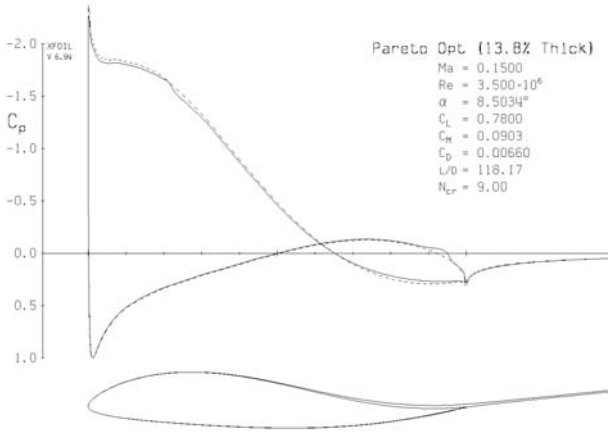


Figure 11. Compromise aerofoil — loiter C_P distribution.

pressure distribution and overall form. Figures 10 and 11 show a compromise aerofoil, having a very pronounced S-shaped camber distribution. The pressure distribution is again seen to be relatively conventional, with a marked favourable gradient on the lower surface in both flow regimes. Figure 12 shows the objective two optimal aerofoil in the loiter regime, and finally it can be seen that the pressure distribution is of the classical 'rooftop' type on the upper surface while having an almost constant favourable pressure gradient on the lower surface.

Concluding this case, it is observed that all aerofoils easily satisfy the design constraints. Without any problem specific knowledge, the evolutionary method has discovered forms (Figs 9 and 12) that would have been designed by an expert in aerodynamics, as well as an unusual but effective compromise form (Figs 10 and 11).

6.4 Two objective inverse aerofoil design

6.4.1 Problem definition

This test case considers the application of the method for multi-objective inverse aerodynamic design. The problem has two objectives and consists of minimising of the difference between computed surface pressure distributions of two pre-defined target aerofoil pressure distributions at flight conditions:

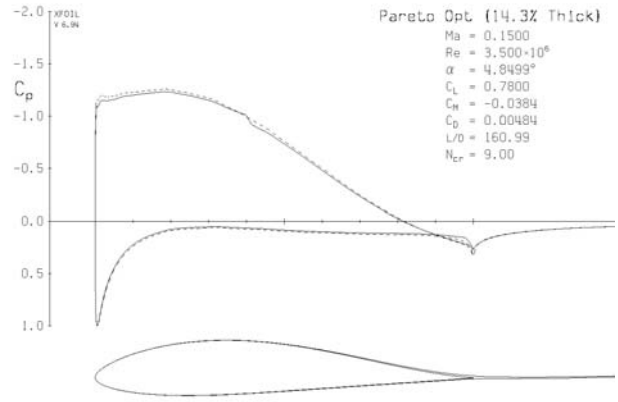


Figure 12. Objective two optimal loiter C_P distribution.

Flight Condition 1: Surface pressure distribution over a typical high-lift airfoil at subsonic conditions. NACA 0012, $M_\infty = 0{\cdot}2$, $Re = 2{\cdot}7 \times 10^6$, $\alpha = 1{\cdot}25$.

F*light Condition 2:* Surface pressure distribution over a typical transonic airfoil RAE 2822, $M_\infty = 0{\cdot}75$, $Re = 9{\cdot}0 \times 10^6$, $\alpha = 1{\cdot}0$.

6.4.2 Design variables and fitness functions

In this case four free control points on the mean line and five free control points on the thickness distribution are the selected design variables. The two fitness functions to be minimised are defined as:

$$f_1 = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(Cp_{current} - Cp_{NACA0012}\right)^2} \qquad \ldots (10)$$

$$M_\infty = 0{\cdot}20, \alpha = 5\,\text{deg}$$

$$f_2 = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(Cp_{current} - Cp_{RAE2822}\right)^2} \qquad \ldots (11)$$

$$M_\infty = 0{\cdot}75, \alpha = 1\,\text{deg}$$

6.4.3 Implementation

MSES is utilised as an aerodynamic analysis tool and the following specific parameter are considered for the evolutionary optimiser:

Top Layer

A population size of 30, intermediate recombination used between two parents, and a mesh of 173 × 36 points used by the MSES solver.

Middle Layer

A population size of 30, discrete recombination used between two parents, and a mesh of 163 × 36 points used by the MSES solver.

Bottom Layer

A population size of 30, discrete recombination used between two parents, and a maximum of 153 × 36 points used by the MSES solver.

6.4.4 Numerical results

This problem was run for 3,000 function evaluations of the head node, and took approximately ten hours on the cluster with eight machines. Figure 13 shows the Pareto front obtained for this test case. Figures 14(a,b) and 15 show a comparison of the target geometries and surface pressure distributions respectively. As illustrated, there is a good match on the computed and target surface pressure distribution.

Figure 16 shows the speed-up of the computation with the increasing number of processors.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

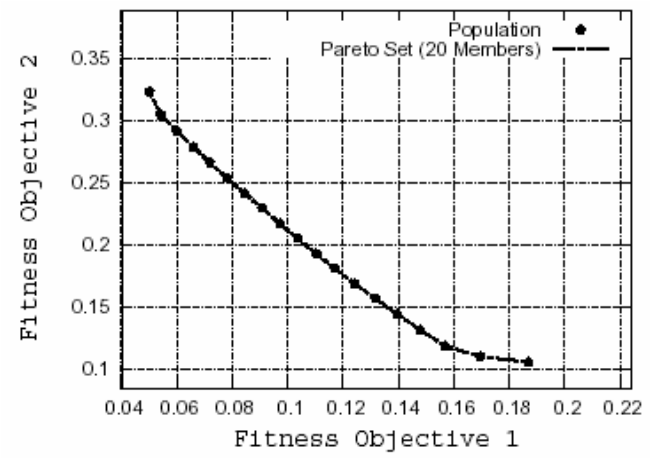


Figure 13. Pareto front for multi-point aerofoil design.

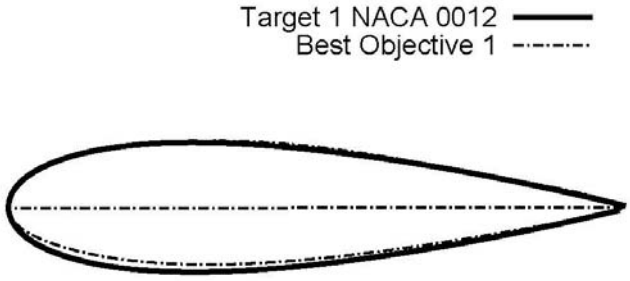


Figure 14(a). Target and computed geometries multi-point aerofoil design for objective one.

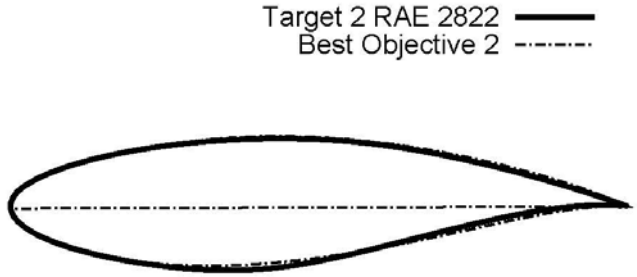


Figure 14(b). Target and computed geometries multi-point aerofoil design for objective two.

Concluding this case; the benefits of parallel computing and the capabilities of the method to find optimal shapes for inverse aerofoil shape optimisation problems is demonstrated. Without any problem specific knowledge, the method has captured the correct pressure distribution over two different aerofoils operating at different flow conditions.

6.5 Two objectives UAV aerofoil section optimisation and redesign

6.5.1 Problem definition

This case considers the detailed design of a single element aerofoil for a small UAV application similar to the RQ-7A Shadow 200 Tactical UAV. The operating conditions and data are based on Ref. 28. The aircraft maximum gross weight is approximately 320lb, it has a wingspan of approximately 12·8ft, a mean chord of approximately 2ft, length of 11ft, and a planform shape with little to no-sweep. It is assumed that the aircraft will be operating between a slow cruise 33·3ms^{-1} and fast cruise 46·6ms^{-1} approximately. These results in the airframe, flight parameters and operating conditions indicated in Table 1. These conditions assume an aircraft at mid weight-cruise during an extended cruise phase at intermediate altitude.

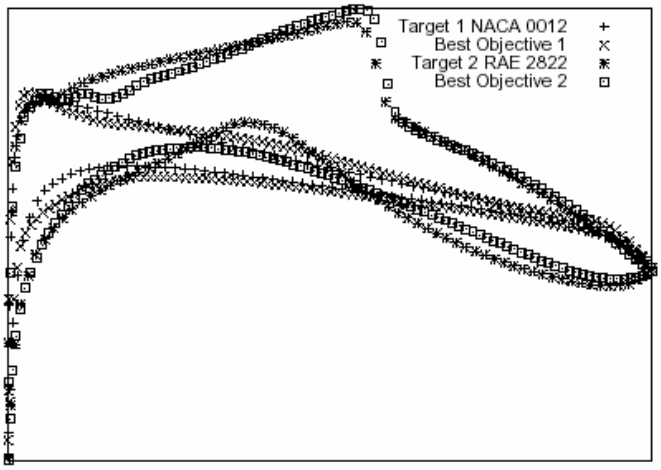


Figure 15. Target and computed pressure distribution for multi-point aerofoil design.

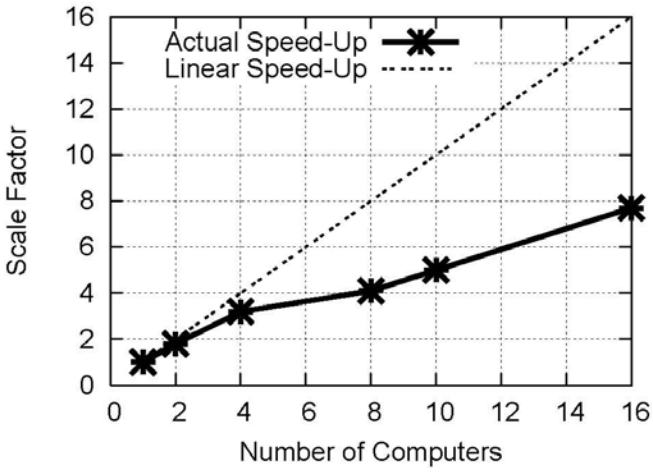


Figure 16. Speed up of the computation with increasing number of computers.

Table 1
UAV data and operating conditions

Aerofoil section	NACA 4415
Wingspan, ft	12·8
Wing chord (approx), ft	2·0
Length, ft	11·2

Description	**Flight Condition 1 Slow Cruise**	**Flight Condition 2 Fast Cruise**
$\mathbf{M_\infty}$	0·1025	0·141
Re	$1{\cdot}085 \times 10^6$	$1{\cdot}490 \times 106$
$\boldsymbol{C_l}$	1·18	0·6140
Altitude, m	3,000	3,000

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

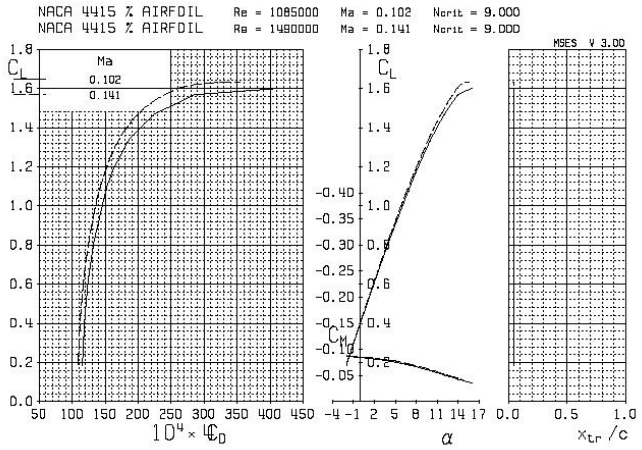


Figure 17. NACA 4415 — Polar.

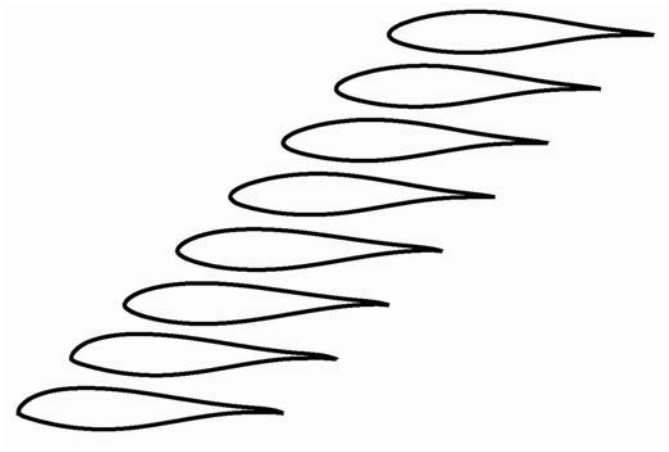


Figure 18. Pareto front for aerofoil design.

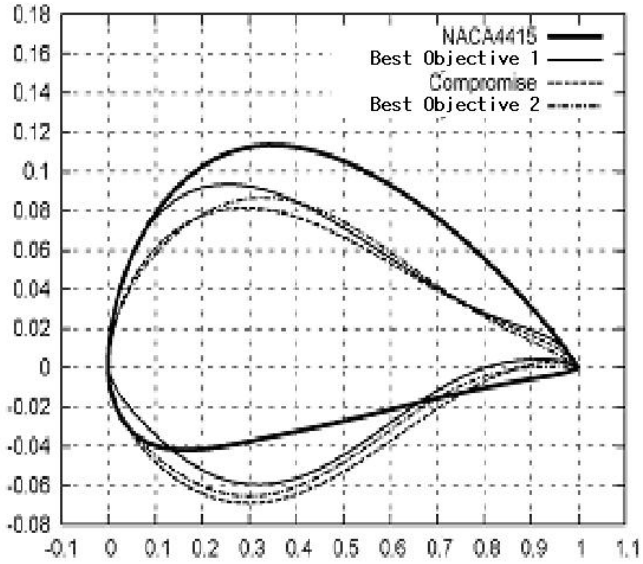


Figure 19. Comparison of selected geometries.

6.5.2 Analysis and design rationale

For the optimisation, it is assumed that the UAV has an existing aerofoil geometry and that it is operating at the two suggested design points, and then the objective is to design an aerofoil that preserve the original thickness while reducing the drag coefficient. The assumed baseline aerofoil geometry is the NACA 4415. This aerofoil is 15% thick. The combined polars for the NACA 4415 aerofoil are shown in Fig. 17. It is noted that both cruise points operate inside the invariant drag region of the aerofoil; the low speed cruise condition giving approximately C_d = 0·016 and the high speed giving approximately C_d = 0·012.

In designing a replacement aerofoil for this UAV platform, the following design factors are considered:

- Maintain approximately the same C_l so as to not impinge upon the assisted launch and landing length.
- Maintain at least the current thickness, so as not to increase the weight of the wing.
- Lower the drag at both cruise points, in a multi-objective fashion.
- Lower the pitching moment to reduce the trim drag.

6.5.3 Design variables and fitness functions

Six free control points are considered on the mean line and ten free control points on the thickness distribution. The two fitness functions to be optimised are defined as minimisation of drag (C_d) at the two flight conditions.

$$f_1 = c_d + Penalty$$
$$\mathrm{Re} = 1{\cdot}085 \times 10^6,\ C_l = 1{\cdot}184 \qquad \ldots (12)$$

$$f_2 = c_d + Penalty$$
$$\mathrm{Re} = 1{\cdot}490 \times 10^6,\ C_l = 0{\cdot}614 \qquad \ldots (13)$$

6.5.4 Design constraints

There are three types of constraints: maximum thickness, maximum thickness location and pitching moment (C_m). The maximum thickness and its location of each aerofoil must exceed 15 % ($t/c \geq 0{\cdot}15$) and be between 20 and 40% chord, respectively. If a constraint on pitching moment is applied this must not be more severe than –0·0660 (C_m –0·0660). When all constraints are considered they are added up and applied by equally penalising both fitness values via a linear penalty method.

6.5.5 Implementation

MSES is utilised as an aerodynamic analysis tool and the following parameter settings on the evolutionary optimisation algorithm:

Top Layer
A population size of 20 and a computational mesh of 215 × 36 points used by the MSES solver.
Middle Layer
A population size of 20 and a computational mesh of 165 × 2 points used by the MSES solver.

6.5.6 Numerical results

This case was run for the 2,000 function evaluations on a cluster of four computers. Figure 18 shows the ensemble of aerofoils in the Pareto front. From this front, three aerofoils; objective one optimal,

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

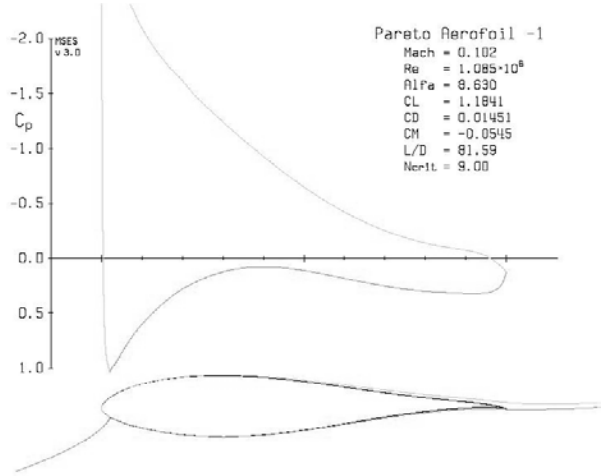


Figure 20. Pareto 01 at slow-cruise.

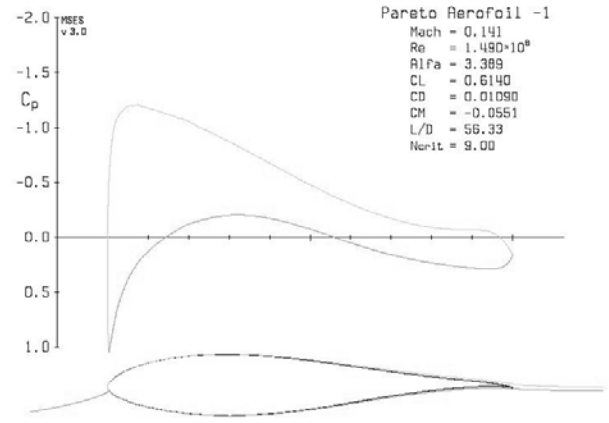


Figure 21. Pareto 01 at fast-cruise.

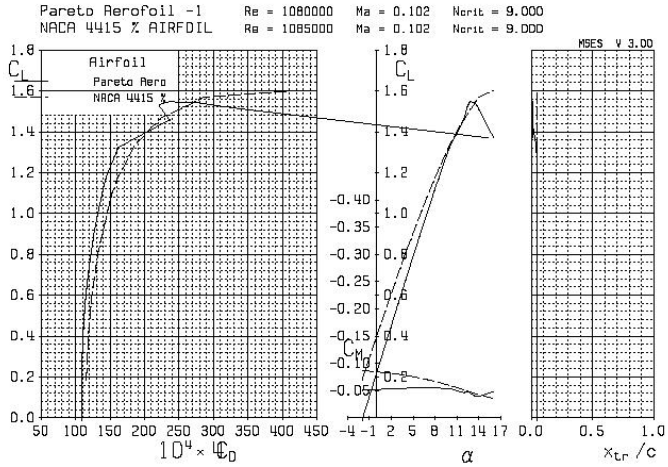


Figure 22. Comparative polars — Pareto 01 and NACA 4415 Re = 1·085 × 10^6.

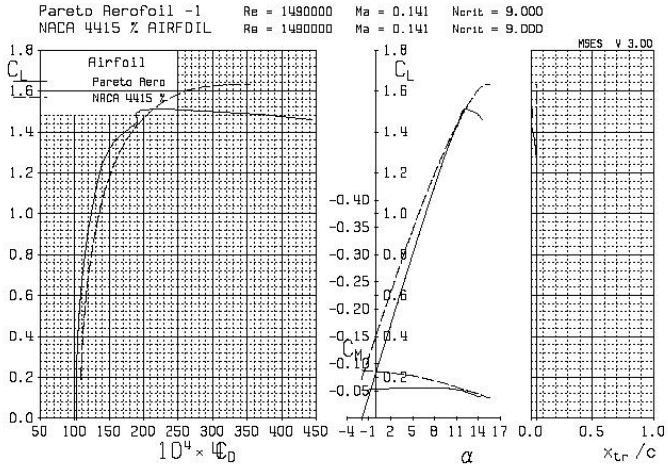


Figure 23. Comparative polars — Pareto 01 and NACA 4415 Re = 1·490 × 10^6.

objective two optimal and compromise aerofoil are selected from the middle of the front. These geometries are shown against the NACA 4415 aerofoil in Fig. 19. The compromise aerofoil is considered for further evaluation. Figures 20 and 21 show the C_p distribution for the two flight conditions. Figure 22 shows the comparative drag polars for Re = 1·085 × 10^6 and Fig. 23 that for Re = 1·490 × 10^6. When compared to the original aerofoil, the compromise aerofoil has lower drag and pitching moment coefficients. Table 2 summarises the drag reduction at the two flight conditions.

Concluding this case, it is apparent that the evolved aerofoil offer significantly lower drag at both cruise conditions with some marked differences on the overall performance and pitching moment coefficient. The requirement of constraining the pitching moment during the evolution process is necessary to avoid obtaining an aerofoil with lower drag for some flight conditions but with undesirable pitching moment characteristics. The results obtained also show the capabilities of the method to find optimal solutions and classical aerodynamic shapes for low drag. The importance of sound engineering judgement before, during and after the optimisation can not be under-emphasised; a proper definition of constraints before performing the evolutionary optimisation and the final results need to be evaluated to obtain feasible designs.

Table 2
UAV drag reduction at two operating conditions

Description	Slow cruise Flight condition 1	Fast cruise Flight condition 2
NACA 4415	0·01621	0·01173
Pareto 01	0·01451	0·01090

6.6 Adaptive aerofoil optimisation

6.6.1 Problem definition

This test case considers the application of adaptive techniques on pre-existing aerofoils to reduce the wave drag at transonic flow conditions. Ashill *et al* 1992 proposed the concept of a transonic bump with so-called shock control bump (SCB) by using geometry adaption on an aerofoil[29-31]. As illustrated in Fig. 24, the use of a SCB on an aerofoil surface causes deceleration of supersonic flow to subsonic Mach numbers by isentropic compression waves and reduction of wave drag.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

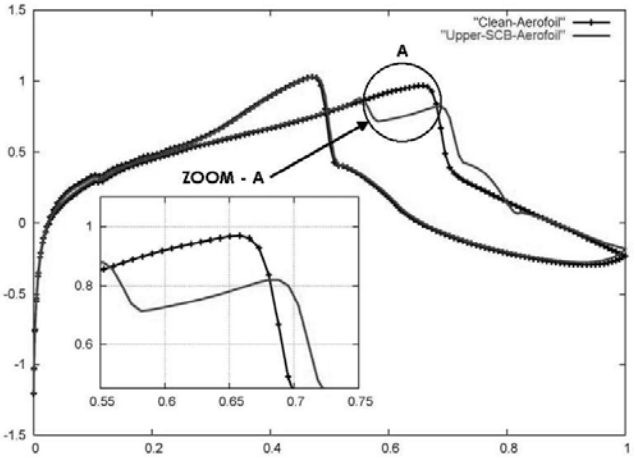


Figure 24. C_P distribution on clean aerofoil and aerofoil with upper SCB.

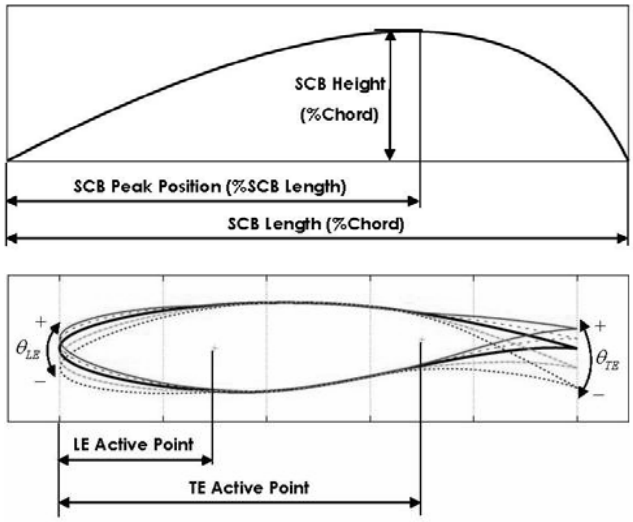


Figure 25. Design variables for SCB and LE/TE deformation.

A second approach on adaptive aerofoils, suggested in this work is the use of a leading-edge (LE) and a trailing-edge (TE) deformation.

The test case focuses on the optimisation of transonic aerofoils using a combination of these adaptive techniques. The problem considers a single objective optimisation problem where the main objective is to minimise wave drag at a fixed lift coefficient C_l = 0·175. For this problem, a combination of a LE and a TE deformation and/or a SCB on the lower and upper surface of the RAE2822 aerofoil operating at flow conditions M_∞ = 0·8, and Re = 1·863 × 10^7 are used. Two implementations are considered: The first implementation uses one SCB on the upper surface and one SCB on the lower surface. The second implementation uses these two and a LE and TE deformation.

6.6.2 Design variables and fitness function

The design variables for this problem are related to the height, length and peak position of SCB and LE/TE deformation requires two deformation angles and two active points as illustrated in Fig. 25. The upper and lower bounds for these variables are indicated in Table 3.

The $SCB_{Peak\text{-}Position}$ represents that point on the original aerofoil where the shock occurs. The fitness function is defined as minimisation of the drag coefficient: *fitness* (*f*) = min(*Cd*).

Table 3
Bounds of design variables for SCB and LE/TE deformation

SCB_{Length}	SCB_{Height}	$SCB_{Peak\text{-}Position}$
$15\%_{chord}$	$0{\cdot}35\%_{chord}$	$50\%_{chord}$
$15\%_{chord}$	$0{\cdot}35\%_{chord}$	$50\%_{chord}$

LE_{Active}	LE_{Angle}	TE_{Active}	TE_{Angle}
$15\%_{chord}$	–2°	$15\%_{chord}$	–5°
$30\%_{chord}$	+1·5°	$30\%_{chord}$	+5°

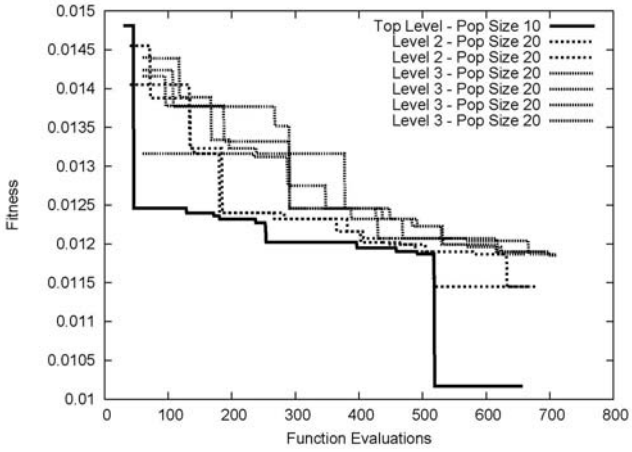


Figure 26. Fitness function *vs* function evaluations.

6.6.3 Implementation

MSES is utilised as an aerodynamic analysis tool and the following specific parameters are considered for the evolutionary optimiser:

Top Layer
A population size of 20 and a computational grid of 36 × 21^3 points used by the MSES solver.
Middle Layer
A population size of 20 and a computational grid of 24 × 19^3 points used by the MSES solver.
Bottom Layer
A population size of 20 and a computational grid of 24 × 17^3 points used by the MSES solver.

6.6.4 Results

The algorithm was allowed to run for 657 function evaluations and converged after 519 function evaluations. Figure 26 illustrates the optimisation progress. Figure 27 compares the original and the optimum aerofoil geometry. Table 4 indicates the optimum design values for this aerofoil. Table 5 shows a comparison of the drag coefficient values. The table indicates that using a combination a LE/TE deformation and a SCB produced an aerofoil with 47·0% of total drag reduction as compared to the original aerofoil. This is 14% less total drag when compared to the aerofoil optimised by using only a SCB on upper and lower surfaces. The aerodynamic performance of these aerofoils at different Mach numbers and Cl values are compared in Figs 28 and 29. In these results, it can be seen how the combination of SCB and LE/TE deformation provides a lower drag coefficient for lift coefficients between 0·1 and 0·385 as well as for Mach numbers above 0·77.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

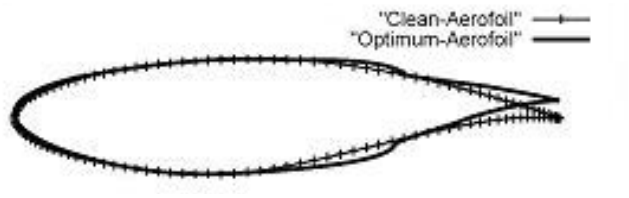


Figure 27. Comparison of clean (RAE 2822) and optimum aerofoil.

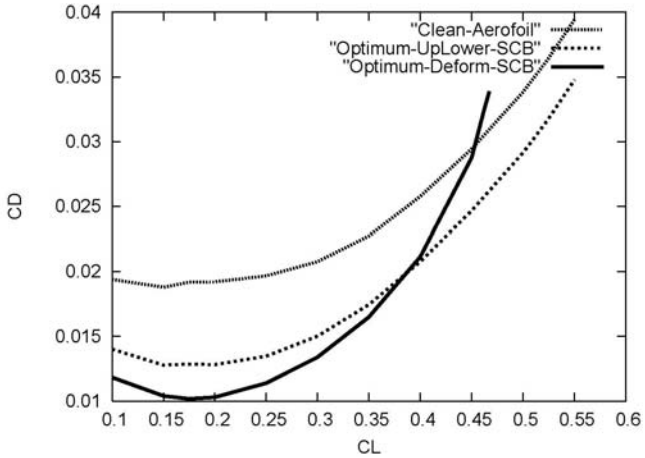


Figure 28. Drag comparison at increasing C_l. [M 0·8, Re 18·63 × 10^6].

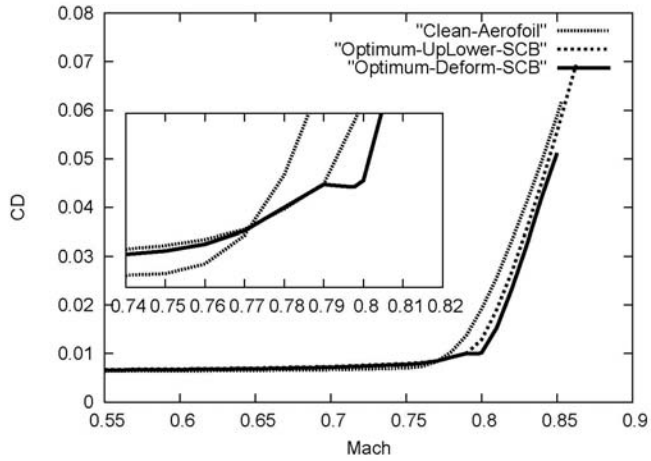


Figure 29. Drag comparison at increasing M. [C_l 0·175, Re 18·63 × 10^6].

7.0 CONCLUSION

In this study a methodology and formulation for practical UAV aerofoil design and optimisation has been presented. It has been shown how a modern evolutionary method can solve multipoint design and optimisation problems in UAV aerofoil aerodynamics without problem specific knowledge. Furthermore, very low drag, classical aerodynamic forms are evolved as solutions to single-point criteria while efficient and novel solutions are produced for compromise points on the Pareto front.

It is also shown that care needs to be taken when using a solver as a 'black box', as the broad search of evolutionary methods may exploit a non calibrated or unintended deficiency of the solver used.

Table 4
Optimum SCB and LE/TE Deformation design variables

Variables	SCB_{Length}	SCB_{Height} (% chord)	$SCB_{Peak\text{-}Position}$ (% chord)
Upper Best	21·00	0·534	67·76%$_{SCB}$
Lower Best	29·58	0·406	74·55%$_{SCB}$

Variables	LE_{Active}	LE_{Angle}	TE_{Active}	TE_{Angle}
Best found	23	+1·443°	73	+4.053°

Table 5
Comparison of aerodynamic performance between clean and optimised aerofoil

Variables	C_d	C_{d_v}	C_{d_w}	L/D
Clean aerofoil	0·0192	0·0103	$8·87\times10^{-3}$	9·119
UpLower SCB	0·0129	0·0106	$2·23\times10^{-3}$	13·587
Optimum aerofoil	0·0102	0·0091	$1·06\times10^{-3}$	17·207

As developed, the evolution algorithm-solver coupling is comparatively easy to set-up as it only requires 'payoff' information from the solver used. The benefits of using parallel computing and evolution algorithms to provide solutions for single and multi-objective problems in aircraft design are clear.

In future, more sophisticated solvers on the upper levels, namely a Navier-Stokes solver with turbulence models on an unstructured grid will be considerd. This facilitates more accurate resolution of separated flows that appear often behind plain flaps, and should remove the limitations of the integral boundary layer used to date. Work on 3D UAV wings is also underway.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Dr Eric J. Whitney, J. Periaux at INRIA and Mourad Sefrioui, Dassault Aviation for fruitful discussions on Hierarchical EAs and their contribution to the optimisation procedure and, also Prof Steve Armfield and Dr Patrick Morgan at the University of Sydney for allowing access to the cluster of computers.

Thanks to M. Drela at MIT for providing the XFOIL and MSES software.

REFERENCES

1. DEB, K. *Multi-Objective Optimisation Using Evolutionary Algorithms*, 2003, Wiley.
2. OBAYASHI, S. Multidisciplinary design optimisation of aircraft wing planform based on evolutionary algorithms, 1998, Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, La Jolla, CA, IEEE, October 1998.
3. SOBIESKI, J. and HAFTKA, R.T. Multidisciplinary aerospace design optimisation: survey of recent developments, 1996, AIAA Paper 96-0711.
4. BÄCK, T. and SSCHWEFEL, H.P. Evolution strategies I: variants and their computational implementation, In WINTER, G., PÉRIEAUX, J., GALA, M. and CUESTA, P. (Eds), *Genetic Algorithms in Engineering and Computer Science* (pp 111-126), Chichester: Wiley, 1995.
5. GOLDBERG, D. *Genetic Algorithms in Search, Optimisation and Machine Learning*, 1989, Addison-Wesley.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press

6. OYAMA, A., LIOU, M.-S. and OBAYASHI, S. Transonic axial flow blade shape optimisation using evolutionary algorithm and three-dimensionalNavier-Stokes solver, 2002, Ninth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, Atlanta, GA, September 2002.
7. PARMEE, I. and WATSON, A.H. Preliminary airframe design using co-evolutionary multiobjective genetic algorithms, 1999, Proceedings of the Genetic and Evolutionary Computation Conference, July 1999, BANZHAF, W., DAIDA, J., EIBEN, A.E., GARZON, M.H., HONAVAR, V., JAKIELA, M. and SMITH, R.E. (Eds), *2*, pp 1657-1665, Orlando, FL, USA, Morgan Kaufmann.
8. SEFRIOUI, M., PÉRIAUX, J. and GANASCIA, J.-G. Fast convergence thanks to diversity, 1996, Evolutionary Programming V Proc of the Fifth Annual Conference on Evolutionary Programming, FOGEL, L.J., ANGELINE, P.J. and BACK, T. (Eds), MIT Press.
9. SEFRIOUI, M. and PÉRIAUX, J. A Hierarchical genetic algorithm using multiple models for optimisation, *Parallel Problem Solving from Nature*, 2000, SCHOENAUER, M., DEB, K. RUDOLPH, G., YAO, X., LUTTON, E., MERELO, J.J. and SCHWEFEL, H.-P. (Eds), PPSN VI, pp 879-888, Springer.
10. WHITNEY, E., SEFRIOUI, M., SRINIVAS, K. and PÉRIAUX, J. Advances in hierarchical, parallel evolutionary algorithms for aerodynamic shape optimisation, *JSME (Japan Society of Mechanical Engineers) Int J*, 2002, **45**, (1).
11. HOLLAND, J.H. *Adaption in Natural and Artificial Systems*, 1975, University of Michigan Press.
12. BÄCK, T., RUDOLPH, G. and SCHWEFEL, H.P. Evolutionary programming and evolution strategies: similarities and differences, 1993, Proceedings of the Second Annual Conference on Evolutionary Programming, pp 11-22, Evolutionary Programming Society, San Diego, CA.
13. FOGEL, D.B. An analysis of evolutionary programming, 1992, Proceedings of the First Annual Conference on Evolutionary Programming, FOGEL, D.B. and ATMAR, W. (Eds), pp 43-51, Evolutionary Programming Society, San Diego, CA.
14. KOZA, J.R. *On the Programming of Computers by Means of Natural Selection*, 1992, MIT Press, Cambridge, MA.
15. MICHALEWICZ, Z. Genetic algorithms + data structures = evolution programs. *Artificial Intelligence*, 1992, Springer-Verlag.
16. THEIRENS. Adaptive mutation rate control schemes in genetic algorithms, 2002, Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation, pp 980-985, IEEE Press.
17. HANSEN, N. and OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies, *Evolutionary Computation*, 2001, **9**, (2), pp 159-195.
18. COLLELLO, C.A, VAN VELDHUIZEN, D.A. and LAMONT, G.B. *Evolutionary Algorithms for Solving Multi-Objective Problems*, March 2002, Kluwer Academic Publishers, New York.
19. CANTU-PAZ, E. *Efficient and Accurate Parallel Genetic Algorithms*, 2000, Kluwer Academic Publishers, New York.
20. VAN VELDHUIZEN, D.A., ZYDALLIS, J.B. and LAMONT, G.B. Considerations in engineering parallel multiobjective evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, April 2003, **7**, (2), pp 144-173.
21. GEIST, A., BEGUELIN, A., DONGARRA, J., JIANG, W., MANCHEK, R. and SUNDERAM, V. PVM: *Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, 1994, Massachusetts Institute of Technology.
22 WAKUNDA, J. and ZELL, A. Median-selection for parallel steady-state evolution strategies, *Parallel Problem Solving from Nature*, 2000, SCHOENAUER, M., DEB, K., RUDOLPH, G., YAO, X., LUTTON, E., MERELO, J.J. and SCHWEFEL, H.-P. (Eds), PPSN VI, pp 405-414, Springer, Berlin.
23. BEYER, H.-G. and DEB, K. On the desired behaviors of self-adaptive evolutionary algorithms, *Parallel Problem Solving from Nature*, 2000 SCHOENAUER, M. *et al*, (Eds), **6**, pp 59-68, Springer-Verlag, Heidelberg.
24. COLLARD, P. and ESCAZUT, C. Genetic operators in a dual genetic algorithm, 1995, Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence, November 1995, pp 12-19, Virginia, USA, IEEE.
25. DRELA, M. XFOIL *6.94 User Guide*, 2001, MIT Aero Astro.
26. DRELA, M. *A User's Guide to MSES V2.3*, February 1993.
27. ABOTT, I.H. and VON DOENHOFF, A.E. *Theory of Wing Sections*, 1980, Dover.
28. MUNSON, K. *Jane's Unmanned Aerial Vehicles and Targets*, 1998, Jane's Information Group.
29. ASHILL, P.R., FULKER, J.L and SHIRES, A. A novel technique for controlling shock strength of laminar-flow aerofoil sections, 1992, *Proceedings First European Forum on Laminar Flow Technology*, 16-18 March 1992, pp 175-183, Hamburg, Germany.
30. OH, J.T., PARK, H.C. and HWANG, W. Active shape control of a double-plate structure using piezoceramics and SMA wires, *Smart Materials and Structures*, 2001, **10**, (5), pp 1100-1106.
31. D. BUSHNELL. Shock wave drag reduction, *Annual Review of Fluid Mechanics*, January 2004, **36**, NASA Langley Research Center, NASA.

https://doi.org/10.1017/S0001924000001524 Published online by Cambridge University Press