GENETIC ALGORITHMS APPLIED TO MULTI-OBJECTIVE AERODYNAMIC SHAPE OPTIMIZATION

Terry L. Holst
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

A genetic algorithm approach suitable for solving multi-objective optimization problems is described and evaluated using a series of aerodynamic shape optimization problems. Several new features including two variations of a binning selection algorithm and a gene-space transformation procedure are included. The genetic algorithm is suitable for finding pareto optimal solutions in search spaces that are defined by any number of genes and that contain any number of local extrema. A new masking array capability is included allowing any gene or gene subset to be eliminated as decision variables from the design space. This allows determination of the effect of a single gene or gene subset on the pareto optimal solution. Results indicate that the genetic algorithm optimization approach is flexible in application and reliable. The binning selection algorithms generally provide pareto front quality enhancements and moderate convergence efficiency improvements for most of the problems solved.

Nomenclature

$\mathbf{F}$	set of scalar objective functions
$\mathbf{F}^*$	optimal set of $\mathbf{F}$ values
f	scalar objective function
f^*	maximum value of f
$\mathbf{G}^n$	n^{th} GA generation (active file)
IR	integer ranking array
ISEED	seed value used in the random number generator $R(0,1)$
ISELECT	user-specified integer parameter controlling which selection algorithm is used
ITRAN	user-specified integer parameter controlling the gene-space transformation option
N_C	fixed number of chromosomes in each GA generation
N_G	number of genes in each chromosome
N_O	number of scalar objective functions
P	user-specified vector with four elements that controls modification operator usage
p_1	user-specified parameter controlling probability that a specific gene will be modified using perturbation mutation operator $(0 \le p_1 \le 1)$
p_2	user-specified parameter controlling probability that a specific gene will be modified using global mutation operator $(0 \le p_2 \le 1)$
$\mathbf{R}$	basis vector matrix with elements $r_{i,j}$
$R(0,1)$	random number generator that returns a random value between 0 and 1
$\mathbf{U}$	unitary transformation matrix with elements $u_{i,j}$
$x_{i,j}^n$	i^{th} gene from the j^{th} chromosome and the n^{th} GA generation
$\mathbf{X}_j^n$	j^{th} chromosome from the n^{th} GA generation
$x_{\max_i}$	user-specified maximum limit on the i^{th} gene
$x_{\min_i}$	user-specified minimum limit on the i^{th} gene
β	user-specified parameter controlling the size of the perturbation mutations $(0 \le \beta \le 1)$

subscripts

i	gene or decision variable index
j	chromosome index
k	objective function index

superscripts

n	GA generation or population index
t	temporary chromosome values obtained after selection but before modification

Background

Numerical methods for optimizing the performance of engineering problems have been studied for many years. Perhaps the most widely used general approach involves the computation of sensitivity gradients. These methods—called gradient methods—have been utilized to produce optimal engineering performance in a wide variety of different forms. The reliability and success of gradient methods generally requires a smooth design space and the existence of only a single global extremum or an initial guess close enough to the global extremum that will ensure proper convergence.

In contrast to gradient-based methods, design space search methods such as genetic algorithms (GA)—also referred to as evolutionary algorithms (EA)—offer an alternative approach with several attractive features. The basic idea associated with the GA approach is to search for optimal solutions using an analogy to the theory of evolution. The problem to be optimized is parameterized into a set of decision variables or genes. Each set of genes that fully defines one design is called an individual or a chromosome. A set of chromosomes is called a population or a generation. Each complete design or chromosome is evaluated using a "biological-like" fitness function that determines survivability of that particular chromosome. For example, in aerospace applications, the genes may be a series of geometric parameters associated with an aerospace vehicle that is to be optimized for payload delivered to orbit, aerodynamic performance or structural weight. The fitness function takes as input all the geometric parameters and returns a numerical value for the fitness—the size of the payload, the aerodynamic performance or the structural weight.

During solution advance (or "evolution" using GA terminology) each chromosome is ranked according to its fitness vector—one fitness value for each objective. The higher-ranking chromosomes are selected to continue to the next generation while the lower-ranking chromosomes are not selected at all. The newly selected chromosomes in the next generation are manipulated using various operators (combination, crossover or mutation) to create the final set of chromosomes for the new generation. These chromosomes are then evaluated for fitness and the process continues—iterating from generation to generation—until a suitable level of convergence is obtained or until a specified number of generations has been completed.

Constraints can easily be included in the GA optimization approach either by direct inclusion into the objective function definition—a so-called penalty constraint—or, more commonly, by including one or more constraints into the fitness function evaluation procedure. For example, if a design violates a constraint, its fitness is set to zero, that is, it does not survive to the next generation. Because GA optimization is not a gradient-based optimization technique, it does not need sensitivity derivatives. It theoretically works well in non-smooth design spaces containing several or perhaps many local extrema.

General GA details including descriptions of basic genetic algorithm concepts can be found in Goldberg,[1] Davis,[2] and Beasley, et al.[3,4] Additional useful studies, which survey recent activities in the area of GA research including the presentation of model problems useful for evaluating GA performance are given in Deb,[5] Van Veldhuizen and Lamont[6] and Jiménez, et al.[7]

A disadvantage of the GA approach is expense. In general, the number of function evaluations required for a GA optimization process to converge, exceeds the number of function evaluations required by a finite-difference-based gradient optimization (see the results presented in Obayashi and Tsukahara[8] and Bock[9]). This situation is offset, to an extent, by the ease with which GAs can be implemented in parallel or distributed computing environments.

Despite being relatively new, genetic algorithms have already been applied in many applications over a broad range of engineering fields. A brief survey of single discipline applications is presented in Holst and Pulliam,[10] and will not be discussed further in this report.

Other applications involving GA search methods have been made in the area of multi-objective or multi-discipline optimization, that is, optimization problems in which two or more objectives are simultaneously and independently optimized. These methods, referred to as MOGA (multi-objective genetic algorithm) methods, are especially attractive because they offer the ability to directly compute so-called "pareto optimal sets" in a single computation instead of the limited single design point traditionally provided by other methods.

The pareto optimal set or pareto front, as it is commonly called, includes optimal solutions for each of the individual objectives, as well as a range of tradeoff solutions in between, which are themselves optimal solutions. Providing a range of solutions to a multi-objective optimization problem is a powerful approach because it allows the designer to see the effect of decision variable variation on the design space in the form of optimal tradeoffs. Thus, the designer can choose individual objective weighting factors after their full influence is quantitatively known.

In recognition of the importance of the MOGA approach, many theoretical developments have been published in recent years. In particular, Van Veldhuizen and Lamont[6] and Zitzler, et al.[11] present reasonably complete surveys of MOGA methodology with a special emphasis on how to compare GA performance from one algorithm variation to another. Other researchers, including Deb,[5] Deb, et al.,[12] Van Veldhuizen,[13] Lohn, et al.[14] and Holst and Pulliam,[15] have developed and/or utilize a suite of test problems as a standard in evaluating MOGA convergence efficiency as well as the accuracy of the final pareto front. A key aspect of pareto front development is diversity. Does a particular MOGA produce good coverage over the entire pareto front or are some regions poorly resolved while other regions have high levels of undesirable clustering? This issue is studied in Laumanns, et al.[16] and Deb, et al.[17]

Still other MOGA issues are associated with archive strategies. As the pareto front develops, many solutions are found that lie on the pareto front that cannot be retained within the fixed population size of many schemes. How to retain or archive this information for the benefit of the MOGA while maintaining reasonable bounds on the archive file has been studied in Knowles and Crone.[18,19] One last example of MOGA research lies in the area of an interactive algorithm development. Parmee, et al.[20] present a MOGA approach that allows changes to be made in GA operators as well as in objective definitions as the MOGA advances from generation to generation.

One area of MOGA research that is even more voluminous than the area of theoretical developments is that associated with GA multi-objective applications. Examples of multi-objective optimization applications include airfoil optimization by Marco, et al.,[21] Naujoks, et al.,[22] Quagliarella and Della Cioppa,[23] Vicini and Quagliarella,[24] Hämäläinen, et al.[25] and Epstein and Peigin,[26] missile aerodynamic shape optimization by Anderson, et al.,[27] and wing optimization by Anderson and Gebert,[28] Sasaki, et al.,[29] Oyama,[30] Obayashi, et al.,[31] and Ng, et al.[32]

Additional MOGA examples in the area of turbomachinery optimization include rocket engine turbopump design by Oyama and Liou[33] and compressor blade design by Benni[34] and Oyama and Liou.[35] In some of these examples the multiple objectives were obtained by considering two different aerodynamic design

points. In others the multiple objectives involved different disciplines including aerodynamics, structures, controls and/or electromagnetics.
One last area of GA research that bears mention is in the area of hybrid methods, the utilization of GA optimization in conjunction with another type of optimizer. This is an attractive area of research because GA methods are particularly good at finding a global extrema, but not particularly good in converging the optimal solution to tight levels of accuracy. Briefly, several examples where hybrid approaches have been utilized include Rai[36] and Madavan[37] for single objective problems and Giotis, et al.,[38] Tursi,[39] Vicini and Quagliarelia[40] and Brown and Smith[41] for multi-objective problems.

Various definitions and the multi-objective genetic algorithm used in the present study are described next. Details associated with each of the operators, including selection, passthrough, random average crossover, perturbation mutation and mutation are presented. MOGA effectiveness is then evaluated using several three-dimensional aerodynamic shape optimization problems.

Problem Statement: Single Objective Optimization

A single-objective optimization problem can be stated as follows: Let f be a scalar function of N_G independent variables, x_i, defined on some domain Ω

$$f = f(\mathbf{X}) = f(x_1,\cdots,x_i,\cdots,x_{N_G}) \tag{1a}$$

In this notation $\mathbf{X}$ is the vector of design space decision variables. The maximum value of f, indicated by f^*, is obtained by finding the values of $\mathbf{X} = \mathbf{X}^*$ such that[†]

$$f^* = \max\{f\} = f(\mathbf{X}^*) = f(x_1^*,\cdots,x_i^*,\cdots,x_{N_G}^*) \tag{1b}$$

The above maximization operation is subject to N_{CO} conditions or constraints indicated by

$$c_n(\mathbf{X}) \le 0 \quad n = 1,2,\cdots,N_{CO} \tag{1c}$$

The constraints placed on the decision variable vector $\mathbf{X}$ by Eqs. (1c) essentially serve to limit the design space within Ω for which the optimal solution is sought.

Problem Statement: Multi-Objective Optimization

For optimization problems involving more than one objective, which are simultaneously and independently optimized, the situation is more difficult. This is because each objective must play a role in determining the optimal solution. In the optimization process, conflicts might arise among the various objective functions, that is, the optimal values of each individual objective, in general, will not occur for the same decision variable vector, $\mathbf{X}$. As a result, the "optimal solution" for a multi-objective optimization problem is typically a range or a set of solutions, which represent tradeoffs in objective space.

To determine when one solution is better than another for multi-objective optimization problems the concept of **dominance** is utilized.[1] A vector $\mathbf{U} = \mathbf{U}(u_1,\cdots,u_i,\cdots,u_N)$ is said to dominate another vector $\mathbf{V} = \mathbf{V}(v_1,\cdots,v_i,\cdots,v_N)$ if and only if $u_i \ge v_i$ for all i and there exists at least one value of i such that $u_i > v_i$. A vector defined on some domain Ω that is not dominated by any other vector defined on Ω is said to be **non-dominated** on Ω.

[†] For the purpose of simplifying the discussion of algorithmic details, maximization is generally assumed. The logic for minimization is a straightforward modification and will not be discussed.

A multi-objective optimization problem can be stated as follows: Let $\mathbf{F}$ be a set of N_O scalar objective functions, f_k, each dependent upon the same decision variable vector, $\mathbf{X}$, which is defined on some design space Ω

$$\mathbf{F} = \mathbf{F}(f_1(\mathbf{X}),\cdots,f_k(\mathbf{X}),\cdots,f_{N_O}(\mathbf{X})) \qquad (2a)$$

As above, the decision variable vector $\mathbf{X}$ consists of N_G independent components. The multi-objective optimization problem involves finding the set of $\mathbf{X} = \mathbf{X}^*$ that produces non-dominated values for $\mathbf{F} = \mathbf{F}^*$ on Ω. This set of values $\mathbf{F}^*$ is called the **pareto optimal set** or the **pareto front.**

For each $\mathbf{F}$ the constraints

$$c_n(\mathbf{X}) \le 0 \quad n = 1,2,\cdots,N_{CO} \qquad (2b)$$

must be satisfied. Existence of these constraints serves to limit or reduce the size of Ω for which the optimal solution is sought.

Genetic Algorithm

The genetic algorithm optimization procedure utilized to solve the multi-objective optimization problem, as described by Eqs. (2), is now presented. It is closely related to the MOGA optimization procedure presented in Holst and Pulliam.[15] As mentioned in the introduction, the general idea behind GA optimization is to discretely describe the design space using a number of decision variables, x_i. In GA parlance these parameters are called genes, and the i subscript refers to the gene number. Each set of genes that leads to the complete specification of an individual design, that is, each decision variable vector, $\mathbf{X}$, is called a chromosome and is indicated by

$$\mathbf{X}_j^n = \mathbf{X}_j^n(x_{1,j}^n,\cdots,x_{i,j}^n,\cdots,x_{N_G,j}^n) \qquad (3)$$

where the j subscript, added to $\mathbf{X}$, identifies the chromosome number. In addition, the j subscript has been added to each gene, to indicate its chromosome membership. The n superscript has been added to indicate the GA generation number, which is iteratively advanced as the solution converges. Thus, $\mathbf{X}_j^n$ is the jth chromosome for the nth generation that consists of N_G genes.

For aerodynamic shape optimization problems, the design space genes are typically a series of geometric parameters, for example, airfoil thickness and camber and/or wing sweep, twist and taper. For many GA applications genes are computationally represented using bit strings and the operators used to manipulate them are designed to accommodate bit string data. In the present approach, following the arguments of Oyama,[42] Houck, et al.[43] and Michalewicz,[44] real-number encoding is used to represent all genes.

The key reason for using real number encoding is that it has been shown to be more efficient in terms of CPU time relative to binary encoded GAs.[44] In addition, real numbers are used for all genes in the present implementation because the present engineering application involves decision variables that are best described using real numbers, for example, the geometric parameters in aerodynamic shape optimization. Thus, using real number encoding eliminates the need for binary-to-real number conversions.

Initialization

Once the design space has been defined in terms of a set of real-number genes, the next step is to form an initial generation, $\mathbf{G}^0$, represented by

$$\mathbf{G}^0 = (\mathbf{X}_1^0, \cdots, \mathbf{X}_j^0, \cdots, \mathbf{X}_{N_C}^0) \qquad (4)$$

where N_C is the total number of chromosomes. Each gene within each chromosome is assigned an initial numerical value using a process that randomly chooses numbers between fixed user-specified limits. For example, the i^{th} gene in an arbitrary chromosome is initialized using

$$x_i = R(0,1)(x_{\max_i} - x_{\min_i}) + x_{\min_i} \qquad (5)$$

where $x_{\max_i}$ and $x_{\min_i}$ are the upper and lower limits for the i^{th} gene, respectively, and $R(0,1)$ is a random number generator that delivers an arbitrary numerical value between 0.0 and 1.0.

The random number generator used in the present study requires an integer input—a seed value. If the integer is positive, the next number in the current random number sequence is returned. If the integer is negative, the random number sequence is reset to a new sequence. Utilization of the same negative seed value will always reset the random number generator to the same random sequence. Each new solution begins by resetting the random number generator using a single call to $R(0,1)$ with a negative seed value—ISEED. All other calls to $R(0,1)$ during that solution use a positive seed value. Thus, a solution can be repeated by simply using the same initial seed value or rerun to determine statistical variation by using a different initial seed value.

Fitness evaluation

After a generation is established—either the initial generation or any succeeding generation in the evolutionary process—fitness values, $\mathbf{F}_j^n$, are computed for each chromosome using a suitable function evaluation. This is analogous to the objective function evaluation in gradient methods and is represented using

$$\mathbf{F}_j^n = \mathbf{F}(\mathbf{X}_j^n) \qquad (6)$$

For example, for a multi-discipline optimization (MDO) problem involving the simultaneous maximization of two separate and distinct objectives, f_1 and f_2, the fitness evaluation represented by Eq. (6) consists of the following

$$f_1^n = f_1(\mathbf{X}_j^n)$$
$$f_2^n = f_2(\mathbf{X}_j^n)$$

where, for example, the first function f_1 might be the aerodynamic drag of an aerospace vehicle (constrained to fly at fixed lift) and the second function f_2 might be the structural mass of the same vehicle. Once all the genes in a specific chromosome have been specified, f_1 can be evaluated using a suitable CFD solver to obtain the drag and f_2 can be evaluated using a suitable structural analysis routine to obtain the structural mass. In this case, of course, the optimization problem would be one of minimization.

Ranking

The purpose of the ranking operation is to determine a set of integer values for the ranking array, IR. One integer value is required for each chromosome. The ranking algorithm is quite different for multi-objective optimization problems relative to single-objective problems. As such, both situations will be discussed in this section. The ranking array values—once determined—are then used in the GA selection process.

Single Objective Optimization—The GA ranking algorithm for single objective optimization problems is quite simple. Whichever chromosome has the highest fitness is ranked number one ($IR=1$), whichever has the second highest fitness is ranked number two ($IR=2$), and so on. This ranking algorithm, more formally stated, is given by

$$\left.\begin{array}{c} ic=1 \\ if\,(f_j^n < f_{jj}^n)\ ic = ic+1 \quad jj=1,\cdots,N_C \\ IR_j^n = ic \end{array}\right\} j=1,\cdots,N_C \qquad (7)$$

where j and jj are special counters that range over all N_C chromosomes in the current population or generation level.

Multi-Objective Optimization—For multi-objective optimization problems the ranking procedure is more complex and utilizes the concept of dominance, which was defined previously.

A chromosome with a fitness vector $\mathbf{F}$ that is not dominated by any other fitness vector within the design space is said to be a non-dominated chromosome. The optimal solution set $\mathbf{F}^*$ or pareto front includes all solutions with fitness vectors that are non-dominated and that satisfy the constraints given by Eqs. (2b). The numerical approximation to the pareto front must involve a suitably complete set of discrete solutions so as to describe the optimal values of each individual objective—these are the pareto front endpoints—as well as the non-dominated tradeoff solutions in between the optimal values associated with each of the individual objectives.

With the concept of dominance in hand the actual ranking process can now be presented. There are a number of ranking procedures available for use in multi-objective optimization. The one used in the present study is called Goldberg ranking.[1] After all of the fitness values have been determined, each chromosome is checked for dominance. Those chromosomes that are non-dominated are given a number one ranking ($IR=1$) and then temporarily removed from the current generation. The remaining chromosomes that are non-dominated are given a number two ranking ($IR=2$) and then temporarily removed. This process continues until each chromosome has an integer value for the ranking array, IR. In general, with this approach, the number of different integer values contained within the ranking array will be small, at least small in comparison to N_C, as many chromosomes will be ranked near the top with a value of 1, 2 or 3.

The above procedure, by itself, represents a legitimate algorithm for the ranking operation. However, there is a refinement that provides a considerable enhancement to MOGA convergence. Before describing this refinement, two additional concepts must be defined—the active file and the accumulation file.

The **active file** is simply a specific name used for the current generation of chromosomes, that is

$$\mathbf{G}^n = (\mathbf{X}_1^n, \cdots, \mathbf{X}_j^n, \cdots, \mathbf{X}_{N_C}^n)$$

is the n^{th} generation active file for the GA iteration process. As the GA solution evolves, the active file is always fixed in size at N_C chromosomes. The first N_O elements in the active file—for the present implementation—always contain the chromosomes that posses the maximum fitness values for each of the N_O objectives.

The **accumulation file** is the list of all non-dominated chromosomes that have been found from all generations combined during the current MOGA iteration. As the MOGA solution evolves, the accumulation file typically grows in size with more chromosomes being added after each new generation. As new non-dominated chromosomes are added to the accumulation file, old chromosomes that lose their dominance must be deleted, thus ensuring that the accumulation file contains nothing but number-one ranked chromosomes. Because each chromosome stored in the n^{th}generation accumulation file is non-dominated, it serves to define the pareto front or, at least, the n^{th}generation approximation to the pareto front. The use of an accumulation file makes sense only when $N_O \geq 2$.

The multi-objective ranking procedure described above utilizes only the chromosomes in the active file, that is, each chromosome in the active file is ranked relative to the other chromosomes in the active file. But this philosophy potentially wastes a plethora of information because not every number-one ranked chromosome for multi-objective optimization problems can be retained from one generation to the next in the active file.

This is where the refinement in the ranking routine enters. Once each chromosome in the active file is ranked using the standard routine, an additional test is performed to see if any number-one ranked chromosomes in the active file are dominated by any of the chromosomes in the accumulation file. If this is the case, then the ranking number associated with the newly ranked chromosome in the active file is decremented by one. This ensures that the ranking routine produces number one rankings that are number one in a global sense.

Selection

After the ranking array is established, with or without the accumulation file option, the GA algorithm passes to the selection process to determine which chromosomes will continue on to the $(n+1)^{st}$ generation and which will not. In the present algorithm implementation, four different selection operators have been coded: (1) a simple technique called greedy selection, (2) a traditional tournament selection algorithm, (3) a bin selection algorithm based on the computation of arc-length along the pareto front and (4) a second bin selection algorithm based on subdividing the design space into a matrix of boxes. The first two selection algorithms can be used for one or more objectives; the fourth can be used for two or more objectives, and the third inherently requires two objectives. The selection algorithm actually used is controlled by a user-input parameter called ISLECT, which will be defined shortly.

Regardless of the selection algorithm chosen, the selection process picks established chromosomes—either from the n^{th} generation active file, $\mathbf{G}^n$, or from the accumulation file—and then stores the results in a temporary holding array $\mathbf{G}^t$. The chromosomes stored in $\mathbf{G}^t$ are then used by the subsequent modification operators to create $\mathbf{G}^{n+1}$, which will be discussed shortly. For the present study, results are computed using only the first, third and fourth selection algorithms, and thus, only these will be described herein.

Greedy selection—This selection algorithm is quite simple. It selects chromosomes from the n^{th} generation active file, that is, from $\mathbf{X}_j^n$. It is implemented using the following

$$\left.\left.\begin{array}{l} m=1 \\ \text{if } (IR_j^n \le it) \text{ then} \\ \quad \mathbf{X}_m^t = \mathbf{X}_j^n \\ \quad m = m+1 \\ \quad \text{if}(m > N_C)\text{stop} \\ \text{endif} \end{array}\right\} j=1,N_C \right\} it = 1,N_C \tag{9}$$

where each selected chromosome $\mathbf{X}_m^t$ is placed in a temporary holding array indicated by

$$\mathbf{G}^t = (\mathbf{X}_1^t,\cdots,\mathbf{X}_m^t,\cdots,\mathbf{X}_{N_C}^t)$$

Note how the highest ranked individuals in the n^{th} generation are selected multiple times, individuals with average ranking are selected a small number of times and individuals with the lowest rankings are not selected at all. This biasing toward individuals with the highest rankings is a key element in any GA. This baseline variation of greedy selection is implemented by setting ISELECT = 1.

Tournament selection—The tournament selection algorithm is quite simple and is used widely—one variation or another—in many GA optimization applications. The present variation is implemented as follows. Using a random process, three chromosomes are chosen from the n^{th} generation active file, that is, from $\mathbf{X}_j^n$. The ranking array values, IR, are then compared. The chromosome with the highest ranked array value is placed into a temporary holding array, $\mathbf{G}^t$. In case of ties the chromosome selected first is chosen. Next, three new chromosomes are chosen from $\mathbf{X}_j^n$, and their ranking array values are compared. As before, the chromosome with the highest ranking is added to $\mathbf{G}^t$. This process continues until N_C chromosomes have been selected from $\mathbf{X}_j^n$. At the conclusion of the tournament selection process, some of the original chromosomes within $\mathbf{X}_j^n$ may not have been selected even once, while other chromosomes may have been selected multiple times. This selection algorithm is implemented by setting ISELECT = 2.

Bin selection (version 1)—The first bin selection algorithm is different from greedy and tournament selection, as implemented here, in two general ways. First, the bin selection algorithm chooses its chromosomes from the accumulation file, not the active file. Second, the bin selection algorithm divides the distance along the pareto front into N_{bin} equal segments or "bins" using an arc-length computation and then selects an equal number of chromosomes from each segment. This ensures an equal distribution of selected points along the pareto front. The parameter N_{bin} is a user-controlled parameter typically equal to 5 or 10.

More precisely, this bin selection algorithm (version 1) is implemented as follows:

(1) Initiate optimization using greedy selection and proceed until the total number of points on the pareto front equals or exceeds N_{tot}, which is a user controlled parameter that is typically several times the value of N_{bin}.

(2) The arc-length along the pareto front is computed then divided into N_{bin} equal arc-length segments or bins. Each existing pareto front solution point is assigned to a bin according to its arc-length value.

Depending upon the distribution of points along the pareto front, some bins may be heavily populated and others lightly populated.

(3) The selection process randomly chooses a chromosome from the accumulation file keeping track of which bin it was chosen from. When the number of chromosomes chosen from a particular bin equals N_{avg} defined by

$$N_{avg} = N_C / N_{bin}$$

further selections from this bin are blocked. The process continues until N_C chromosomes have been selected. For situations when N_C is not evenly divisible by N_{bin}, the value of N_{avg} is incremented by one. Thus, certain bins may supply one more element to the next generation than other bins, but generally, the distribution will be reasonably well distributed across the entire pareto front. This baseline variation of the bin selection algorithm (version 1) is implemented by setting ISELECT = –3.

This selection algorithm does not guarantee that the pareto front endpoints will be selected and carried forward to the modification process or passed through to the next generation. Since endpoint retention can be an attractive feature for selection, a variation of this selection algorithm that first selects the pareto front endpoints and then proceeds with the standard selection algorithm is also available and is implemented when ISELECT = 3.

Since the present bin selection algorithm requires the computation of arc-length along the pareto front, it inherently requires optimization problems with two objectives, that is, the pareto fronts must be a curve in two-dimensional space. Another bin selection option, which works for two or more objectives, is available, and will be discussed next.

Bin selection (version 2)—The second bin selection algorithm is similar to the first in philosophy and has virtually identical convergence properties, but differs in how the bins are constructed. It does not use an arc-length computation along the pareto front. Instead, the design space is divided into a series of equally sized boxes or bins, each with N_O-dimensions. This approach is applicable for any number of objectives (greater than one) and thus is more general than the first bin selection algorithm.

More precisely, this bin selection algorithm (version 2) is implemented as follows:

(1) Initiate the optimization using greedy selection and proceed until the total number of points on the pareto front equals or exceeds N_{tot}—defined the same as in bin selection algorithm version 1.

(2) The design space bins are constructed next by subdividing each objective dimension in the design space into M_{seg} equal segments—where M_{seg} is a user-controlled parameter. Thus, the design space is divided into a total of $(M_{seg})^{N_O}$ bins. For example, when $M_{seg} = 5$ and $N_O = 2$, the design space is divided into 25 bins, each with the same rectangular shape.

(3) Each existing element on the pareto front, that is, each element in the accumulation file, is categorized as to which bin it occupies. Many of the design space bins will be empty, because they lie beyond the pareto front or because they occupy an interior portion of the design space. Some of the bins that contain pareto front elements may have many entries—some may have few. This, of course, depends on how the pareto front is populated and how the pareto front slices through the matrix of bins that have been created. The number of bins that have at least a single entry is counted. That resulting value is then stored in N_{bin}.

(4) The selection process randomly chooses a chromosome from the accumulation file keeping track of which bin it was chosen from. When the number of chromosomes chosen from a particular bin equals N_{avg}—defined the same as above—further selections from this bin are blocked. The process continues until N_C chromosomes have been selected. As above, for situations when N_C is not evenly divisible by N_{bin}, the value of N_{avg} is incremented by one. Thus, certain bins may supply one more element to the next generation than other bins, but generally, the distribution will be reasonably well distributed across the entire pareto front. This baseline variation of the bin selection algorithm (version 2) is implemented by setting $\text{ISELECT} = -4$.

As with bin selection algorithm (version 1), this selection algorithm does not guarantee that the pareto front endpoints will be selected and carried forward to the modification process or passed through to the next generation. A variation of this algorithm that first selects the pareto front endpoints and then proceeds with the baseline selection algorithm is also available and is implemented when $\text{ISELECT} = 4$.

Modification Operators

P Vector—After the new chromosomes have been selected and placed in the temporary holding array, $\mathbf{G}^t$, they must be modified using one of several modification operators to obtain the $(n+1)^{st}$ generation of chromosomes, $\mathbf{G}^{n+1}$. In the present implementation four modification operators are used—passthrough, random average crossover, perturbation mutation and mutation. How many chromosomes are modified with each operator is controlled by the P vector, which consists of four parameters—p_B, p_A, p_P, p_M. Each element of the P vector controls one modification operator. The value of each P vector element ranges from 0 to 1.0, and, for consistency, the sum of all four elements must always equal one. A P vector equal to 0.1, 0.3, 0.3, 0.3, for example, will cause the first 10 percent of the chromosomes to be modified using the passthrough operator, the next 30 percent to be modified using random average crossover, the next 30 percent to be modified using perturbation mutation and the last 30 percent to be modified using mutation. That is, $p_B = 0.1$, $p_A = 0.3$, $p_P = 0.3$, and $p_M = 0.3$.

The passthrough operator is always performed first. After passthrough is complete, the implementation order of the remaining operators is immaterial. Once all values of $\mathbf{G}^{n+1}$ have been established, the algorithm proceeds to fitness evaluation, ranking and onto succeeding generations until the optimization is sufficiently converged.

Passthrough—The simplest modification operator used in the present GA is "passthrough." As the name implies, a certain number of chromosomes with the highest individual fitness values are simply "passed through" to the next generation from $\mathbf{G}^t$ to $\mathbf{G}^{n+1}$ without modification. The number of chromosomes that are passed through to the next generation is controlled by the first parameter in the P vector, p_B. The passthrough operator is always performed on the first $p_B N_C$ chromosomes in $\mathbf{G}^t$. Care must be taken when choosing p_B and N_C so that $p_B N_C \geq N_O$. If this is done and if a selection option which retains end points is used, the chromosomes with the highest individual fitness values will always be passed through to the next generation, thus guaranteeing that none of the individual maximum fitness values will ever decrease during GA iteration.

Random average crossover—The next GA modification operator is called random average crossover and is implemented by first selecting two random chromosomes $\mathbf{X}^t_{j1}$ and $\mathbf{X}^t_{j2}$ from $\mathbf{G}^t$. Next, the two selected chromosomes are combined on a gene-by-gene basis using the following formula:

$$x^{n+1}_{i,j} = \frac{1}{2}(x^t_{i,j1} + x^t_{i,j2}) \quad i = 1, 2, \cdots, N_G \qquad (10)$$

where $x_{i,j}^{n+1}$ corresponds to the i^{th} gene in the j^{th} chromosome associated with $\mathbf{G}^{n+1}$ and $x_{i,j1}^{t}$ and $x_{i,j2}^{t}$ correspond to the i^{th} genes from the randomly chosen chromosomes $\mathbf{X}_{j1}^{t}$ and $\mathbf{X}_{j2}^{t}$. The number of chromosomes modified using the random average crossover operator is determined by the parameter p_A—the second element in the P vector.

Perturbation mutation—The next GA modification operator is called perturbation mutation and is implemented by first selecting a random chromosome $\mathbf{X}_j^t$ from $\mathbf{G}^t$. Next, a probability test is performed for each gene $x_{i,j}^t$ in the selected chromosome involving a call to the random number generator $R(0,1)$. If the returned random number is less than ρ_1 the gene is modified using

$$x_{i,j}^{n+1} = x_{i,j}^{t} + (x_{\max_i} - x_{\min_i})[R(0,1) - 0.5]\beta \qquad (11)$$

where β is a user-specified tolerance that controls the size of the perturbation mutation, and ρ_1 is a user-specified control parameter that statistically controls the number of genes that are modified. For sensible results the values of β and ρ_1 must be between 0 and 1.0.

Because this operator can cause the value of a particular gene to exceed one of its constraints ($x_{\max_i}$ or $x_{\min_i}$), checks are required to prevent this. The number of chromosomes modified using the perturbation mutation operator is determined by the parameter p_P—the third element in the P vector.

Mutation—The last GA modification operator used in the present study is called mutation and is implemented similarly to the perturbation mutation operator. First, a random chromosome $\mathbf{X}_j^t$ is chosen from $\mathbf{G}^t$. Next, a probability test is performed for each gene $x_{i,j}^t$ in the selected chromosome involving a call to the random number generator $R(0,1)$. If the returned random number is less than ρ_2 the gene is given a completely different value using

$$x_{i,j}^{n+1} = (x_{\max_i} - x_{\min_i})R(0,1) + x_{\min_i} \qquad (12)$$

The parameter ρ_2 is a user-specified control parameter that statistically controls the number of genes that are modified. For sensible results ρ_2 must be between 0.0 and 1.0. The number of chromosomes modified using the mutation operator is determined by the parameter p_M—the fourth element in the P vector.

Gene-space transformation—An option for accelerating GA convergence for multi-objective optimization problems involving a gene-space transformation is described next. This option, introduced in Ref. 15, worked amazingly well for model problems with simple non-convoluted pareto fronts, but performed poorly for model problems in which the pareto front was convoluted. Thus, it is of interest to see how the gene-space transformation procedure will perform on a real-world problem in which the shape of the pareto front—in gene space—is unknown.

The transformation algorithm is outlined as follows:

(1) Transform $\mathbf{G}^t$ using the gene-space transformation procedure.
(2) Perform modifications on the transformed chromosomes according to the user-input P vector values.
(3) Using the inverse of the original transformation the modified chromosomes are transformed back to obtain $\mathbf{G}^{n+1}$.

This gene-space transformation procedure, which can be viewed as a gene-space rotation of coordinates, causes a linear coupling between each of the genes, and thus, affects how they are changed in the modification process. In some cases the gene-space transformation procedure can significantly improve GA convergence. The transformation procedure, which theoretically works for any number of objectives, will now be presented for the two-objective case, that is, for $N_O = 2$.

The idea behind the transformation procedure is to perform a rotation of coordinates in gene space using an angle-preserving, length-preserving orthogonal transformation. For this purpose a simplified version of the Gram-Schmidt orthogonalization is used.[45] The current set of n^{th} generation chromosomes (those that have been newly selected and placed in the temporary holding array $\mathbf{G}^t$) can be written as

$$\mathbf{G}^t = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N_C} \\ x_{2,1} & & & \vdots \\ \vdots & & & \\ x_{N_G,1} & \cdots & & x_{N_G,N_C} \end{pmatrix}$$

where each column is one of the chromosomes in the active file holding array. It is this matrix that is to be transformed using

$$\mathbf{U}\mathbf{G}^t = \tilde{\mathbf{G}}^n \tag{13}$$

where $\mathbf{U}$ is a unitary matrix of rank N_G that needs to be constructed and $\tilde{\mathbf{G}}^n$ is the resulting transformed matrix. To construct $\mathbf{U}$ a set of basis vectors $\mathbf{R}$ needs to be established. The elements of $\mathbf{R}$ are defined by

$$r_{i,j} = \mathbf{X}_2 - \mathbf{X}_1 \quad \text{for} \quad j = 1 \tag{14a}$$

and

$$r_{i,j} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} \quad \text{for} \quad j \geq 2 \tag{14b}$$

The simple choices made for $r_{i,j}$ when $j \geq 2$ serve to simplify the transformation matrix construction without sacrificing overall generality.

As can be seen from Eq. (14a), the first coordinate direction in the new transformed coordinate system is chosen to be parallel to $\mathbf{X}_2 - \mathbf{X}_1$. As was mentioned earlier, the chromosome with the best fitness for the first objective is always placed into $\mathbf{X}_1$, and the chromosome with the best fitness for the second objective is always placed into $\mathbf{X}_2$ This convention is crucial for the success of the transformation algorithm, as it causes the first transformation coordinate to be aligned with the pareto front endpoints. Keep in mind that it is imperative that the first element of the P vector—the element that controls passthrough—is large enough so that both $\mathbf{X}_1$ and $\mathbf{X}_2$ are always passed through to the next generation without modification. It is also imperative for the selection algorithm to have an endpoint retention option.

The unitary transformation matrix $\mathbf{U}$ is constructed column by column using

first column

$$u_{i,1} = \frac{r_{i,1}}{\|r_{i,1}\|} \tag{15a}$$

second column

$$v_i = r_{i,2} - u_{2,1}u_{i,1}\,,\quad u_{i,2} = \frac{v_i}{\|v_i\|} \tag{15b}$$

n th column

$$v_i = r_{i,n} - \sum_{j=1}^{n-1} u_{n,j}u_{i,j}\,,\quad u_{i,n} = \frac{v_i}{\|v_i\|} \tag{15c}$$

Once $\tilde{\mathbf{G}}^n$ is obtained the modification operators are applied the same as without transformation to obtain $\tilde{\mathbf{G}}^{n+1}$. The final $\mathbf{G}^{n+1}$ values are then obtained using an inverse transformation indicated by

$$\mathbf{U}^T\tilde{\mathbf{G}}^{n+1} = \mathbf{G}^{n+1} \tag{16}$$

Because **U** is a unitary matrix, its inverse is simply its transpose, and thus, it is easily constructed.

The gene space transformation option is controlled using the ITRAN control parameter. If ITRAN = 0, no gene space transformation in implemented. If ITRAN = 1 the gene space transformation algorithm is activated.

Geometry Parameterization

In aerodynamic shape optimization the geometric parameterization is an important step that effectively connects the aerodynamic analysis routine to the optimization routine. An analytic parameterization suitable for wing and wing-fuselage configurations, typical of the transonic flow regime, is now described. Most of the parameters have common-sense definitions in which the name itself provides the definition, for example, the wing leading edge radius of curvature, the wing leading edge sweep or the fuselage length.

A Cartesian coordinate system (x, y, z) is used for all configurations. The coordinate system origin is at the wing root leading edge for isolated wing configurations and at the fuselage nose for wing-fuselage configurations. A plane of symmetry ($y = 0$) is inherently assumed for all configurations. The x coordinate is aligned with the freestream direction and is positive downstream, y is normal to the symmetry plane, positive to the pilot's right and z is vertical, positive upward. All length parameters are nondimensionalized using the wing root chord, that is, the wing chord length where it intersects the plane of symmetry.

Isolated wing configurations—Isolated wing geometries are parameterized using N_w airfoil defining sections or stations, each using the geometric parameterization of Sobieczky.[46] The first defining station is always at the wing root and the last is always at the wing tip. Each defining airfoil section is characterized by ten parameters. These parameters along with brief definitions are listed in Table 1. A graphical description of these parameters is presented in Fig.1. All angles are measured in degrees. Once these parameters are specified the airfoil coordinates (z as a function of x) are determined using a polynomial of the form

$$z = \sum_{n=1}^{6} a_n x^{n-1/2} \tag{17}$$

where the coefficients a_n are computed from the ten airfoil defining parameters. Two applications of Eq. 17 are required for a complete airfoil specification—one for the upper surface coordinates and one for the lower surface coordinates. Six simultaneous linear equations involving rle, xup, zup, zxxup, αte + βte/2 and zte are solved for the upper surface coordinates, and six simultaneous linear equations involving rle, xlo, zlo, zxxlo, αte - βte/2 and zte are solved for the lower surface coordinates. Because rle and zte are used for both surfaces, slope continuity at the leading edge and zero thickness at the trailing edge are always maintained.

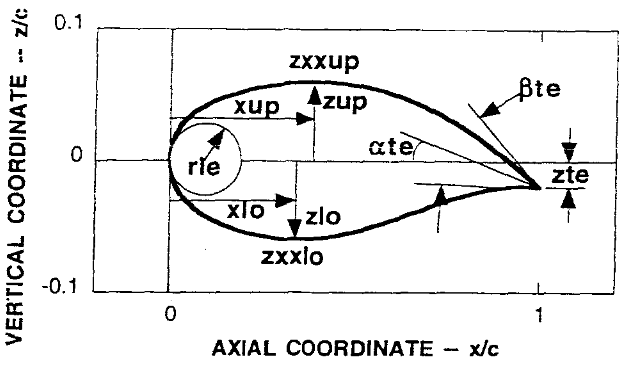


Fig. 1 Airfoil parameterization used for each wing defining station (taken from Sobieczky[46]). Angles are measures in degrees. All lengths are nondimensionalized by wing root chord.

Once the airfoil coordinates are constructed for each defining station, wing coordinates are constructed using linear lofting and then modified according to a set of planform parameters. Seven planform parameters are utilized at each airfoil defining station, producing a total of $7 \times N_w$ planform parameters. These parameters are also listed in Table 1 along with brief definitions.

Certain geometric parameters have predetermined values or are not formerly used in the wing definition process. For example, the root chord length is always unity, and the spanwise location for the root airfoil defining station is always zero. The tip airfoil defining station values for the dihedral angle and the leading edge sweep angle have no meaning, but are nevertheless included in the list for each wing parameterization.

Table 1. Definitions for the airfoil section and wing planform parameters. A value for each parameter is required at each of the N_w wing defining stations, $j = 1, \cdots, N_w$.

Symbol	Definition	Symbol	Definition
Airfoil section parameters		Planform parameters	
rle_j	Leading edge radius of curvature	c_j	Airfoil chord length
xup_j	x-location of upper surface max thickness	D_j	Dihedral angle (deg)
zup_j	Upper surface max thickness	Λ_j	Leading edge sweep (deg)
zxxup_j	Upper surface curvature at $x = \text{xup}_j$	t_j	Airfoil section thickness multiplier
xlo_j	x-location of lower surface max thickness	θ_j	Wing twist (deg)
zlo_j	Lower surface max thickness	$x_{\theta,j}$	Center of twist
zxxlo_j	Lower surface curvature at $x = \text{xlo}_j$	y_j	Spanwise defining station location
zte_j	Position of trailing edge above $z = 0$		
αte_j	Angle between trailing edge bisector and $z = 0$ plane (deg)		
βte_j	Trailing edge included angle (deg)		

Wing-fuselage configurations—Wings for wing-fuselage configurations are parameterized using the procedure as described above. Thus, only the fuselage portion of the wing-fuselage parameterization will be discussed in this section. The first step is to define a base fuselage, which is analytically constructed in three sections. The nose section is an ellipsoid of revolution. The main body section is a right circular cylinder that smoothly transitions into the nose section. And the boattail section is a sine curve of revolution smoothly connecting the main portion of the fuselage with a downstream sting. The sting—not formally part of the fuselage—is a small-radius right circular cylinder that extends to the downstream outflow boundary. A total of eight parameters—defined in Table 2—are required to specify this base fuselage. Note: The y-location of the wing root leading edge is always assumed to be on the symmetry plane. Thus, y_R is inherently assumed to be zero and is only included for completeness.

Table 2. Base fuselage parameter definitions. All lengths are nondimensionalized by airfoil root chord.

Symbol	Definition
x_R	x-location of wing root leading edge
y_R	y-location of wing root leading edge
z_R	z-location of wing root leading edge
x_B	Body length
x_N	Length of ellipsoid nose
r_B	Radius of cylindrical portion of fuselage
x_T	Length of fuselage boattail
r_S	Radius of fuselage sting

Modification of the base fuselage shape is achieved using a series of Hicks-Henne bump functions.[47] A total of N_x bump functions are distributed axially with equal spacing along the x-direction for each of N_T circumferential stations also distributed with equal spacing in the circumferential direction. This allows a total of $N_x \times N_T$ decision variables that are used to produce incremental radius perturbations Δr's to the base fuselage. Any position on the base fuselage, except the nose $x = 0$ or tail $x = x_B$, can be changed using this approach. A sketch showing the definitions of these parameters, including one possible arrangement of Δr's is shown in Fig. 2.

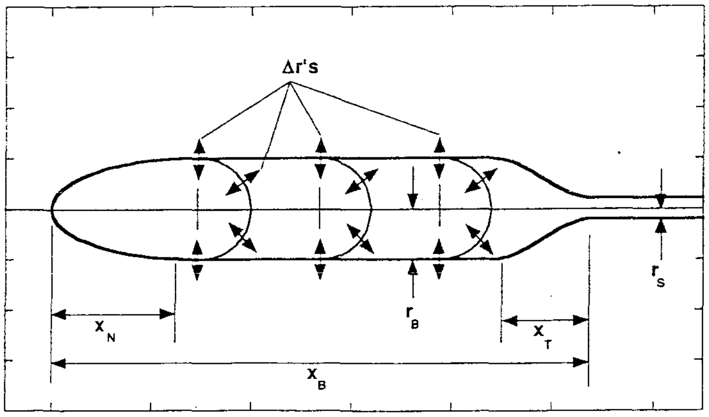


Fig. 2 Sketch showing decision variable definitions for the fuselage parameterization. For this arrangement there are three axial stations ($N_x = 3$) and four circumferential stations ($N_T = 4$).

Masking array—Associated with each decision variable, that is, each gene, is an integer value—either one or zero—stored in a masking array called *mask*. The masking array is established at the beginning of the optimization computation and tells the GA which parameters to modify in the optimization process—using the crossover and mutation operators—and which to leave unmodified. For example, the mask values for c_1, y_{N_w}, Λ_{N_w}, D_{N_w} and y_R are always zero. These parameters, as described above, are carried along with each chromosome but are never used by the GA process. The masking array can also be used to limit the size of the design space. Only those genes with mask = 1 are utilized in the search for optimal problem objectives.

Computed Results

Area Error Norm for Two-Objective Problems

In order to evaluate the accuracy and efficiency of an optimization algorithm it is important to have a proper error norm to assess the level of convergence. For single objective problems a suitable error norm is the simple arithmetic difference between the current "best fitness" and the exact answer. This, of course, works well for model problems in which the exact answer is known. For applications in which the exact answer in not known *a priori*, it can still be employed as an *a posteriori* error computation.

For multi-objective optimization a workable error norm is more elusive since a range of solution values—the so-called pareto front—is being sought. The topic of how to define easy-to-implement and accurate error norms for comparing one pareto front approximation to another for the same problem is discussed at length in Zitzler, et al.[11] and Knowles and Corne.[18] A suitable norm encompassing the optimal values of each of the individual objectives is one possibility. However, such a norm would not take into account the trade-off regions of the pareto front. Another possibility is the attainment surface approach of Fonseca and Fleming.[48] In this approach a set of equally spaced sampling lines that intersect the full breadth of the pareto front are used. Statistical measures of goodness can then be developed based upon how many intersection points one pareto front has that are superior to another pareto front.

In the present study the area between the current pareto front and the final pareto front—a numerically computed approximation of the final pareto front—is used as the error norm to determine level of convergence. As the size of this quantity goes to zero, the current solution approaches the final solution.

The area is computed numerically by dividing the region between the "exact" pareto front and the current approximation into a series of triangles as shown in Fig. 3. The area error norm is the sum of each of the individual triangular areas. Thus, an approximation to the exact pareto front that fails to match the final solution in any location will produce a non-zero value for the area error norm.

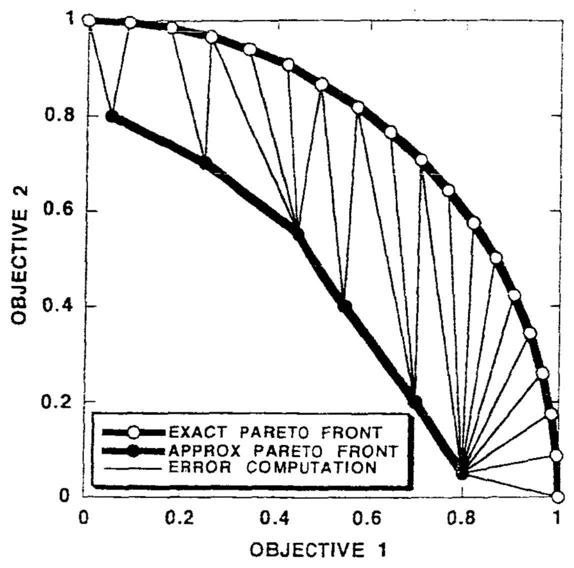


Fig. 3 Area error norm computational process for a two-objective pareto front.

Isolated wing results

The first computed result involves a two-objective optimization for an isolated wing flying in the transonic speed regime—$M_\infty = 0.84$. The two objectives—lift-to-drag ratio at fixed lift and the inverse of the nondimensional structural mass—are simultaneously maximized by the previously presented MOGA optimization procedure. The aerodynamic lift-to-drag ratio is evaluated for each candidate design, that is, for each chromosome, using the TOPS full potential solver[49] by iterating on angle of attack to obtain the lift-to-drag ratio at the specified value of lift—$C_L = 0.45$. The tolerance on the lift iteration is ±1%, but in most cases the error is much less. In effect, this is drag minimization at fixed lift.

The structural mass computation utilizes an equivalent box beam model similar to that used in Oyama.[50] Given a set of loads from the aerodynamics routine, the structures model computes the minimum-mass box beam that will support that load with a factor of safety of two. Shear and bending are included in this model but not torsion.

The design space parameterization for isolated wings with two defining stations, $N_w = 2$, as described above, consists of 34 parameters. Airfoil defining station one is at the wing root and station two is at the wing tip. Maximum and minimum constraint values for each of the geometric parameters, as well as masking array values, are given in Table. 3. Except for wing twist, all planform geometric parameters have a masking array value of zero, that is, they are not modified in the optimization process. Thus, a total of 22 genes are active within each chromosome for this optimization problem, that is, $N_G = 22$.

Table 3. Maximum and minimum gene constraints along with masking array values for the isolated wing optimization problem.

Parameter	max	min	mask	Parameter	max	min	mask
Defining airfoil section 1				Defining airfoil section 2			
rle_1	0.02	0.008	1	rle_2	0.02	0.008	1
xup_1	0.45	0.35	1	xup_2	0.45	0.35	1
zup_1	0.07	0.06	1	zup_2	0.07	0.06	1
$zxxup_1$	-0.2	-0.8	1	$zxxup_2$	-0.2	-0.8	1
xlo_1	0.45	0.35	1	xlo_2	0.45	0.35	1
zlo_1	-0.06	-0.07	1	zlo_2	-0.06	-0.07	1
$zxxlo_1$	0.8	0.2	1	$zxxlo_2$	0.8	0.2	1
zte_1	0.01	-0.01	1	zte_2	0.01	-0.01	1
αte_1	8.0	0.0	1	αte_2	8.0	0.0	1
βte_1	15.0	6.0	1	βte_2	15.0	6.0	1
Planform defining station 1				Planform defining station 2			
c_1	1.0	1.0	0	c_2	0.5	0.5	0
D_1	0.0	0.0	0	D_2	0.0	0.0	0
Λ_1	37.0	37.0	0	Λ_2	37.0	37.0	0
t_1	1.0	1.0	0	t_2	1.0	1.0	0
θ_1	3.0	0.0	1	θ_2	0.0	-3.0	1
$x_{\theta,1}$	0.5	0.5	0	$x_{\theta,2}$	0.5	0.5	0
y_1	0.0	0.0	0	y_2	2.4	2.4	0

Results from this multi-discipline optimization problem are presented in Figs. 4-9. For all computations in this series of results $N_C = 32$, $\beta = 0.1, \rho_1 = 0.2$, $\rho_2 = 0.2$ and $P = (0.04, 0.32, 0.32, 0.32)$. This P vector resulted in two passthrough chromosomes and ten chromosomes in each of the three remaining modification categories—random average crossover, perturbation mutation and mutation.

Pareto front comparisons are presented in Figs. 4. Note that each comparison contains five curves that correspond to different ISEED initialization values used in the random number generator. There is also a "master" result plotted in each part of Fig. 4 that is a concatenation of all pareto front data produced for this case—a total of twenty curves—with only the non-dominated points retained.

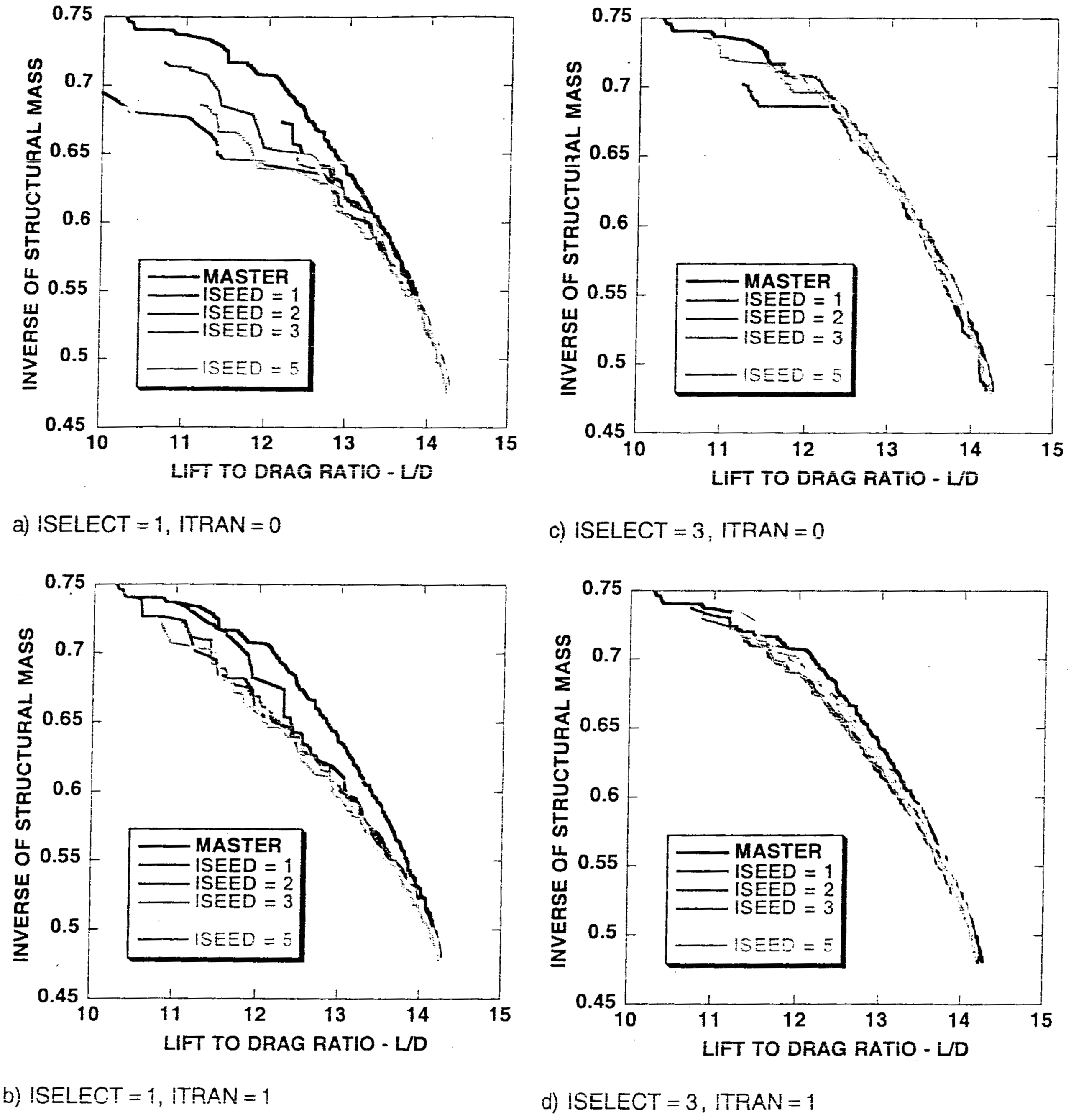


Fig. 4 Pareto front variation for transonic wing optimization in which the lift-to-drag ratio at fixed lift and the inverse of the structural mass are simultaneously maximized. Five solutions utilizing different seed values are plotted for each GA variation after 500 generations, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

There are several interesting features to note in this series of pareto front comparisons:

First of all, the variation in level of convergence across the various ISEED values is quite striking, especially for the cases involving greedy selection without gene space transformation, ISELECT = 1, ITRAN = 0 (Fig. 4a). For these cases, the optimal lift-to-drag ratio point on the pareto front converges consistently in the number of generations allotted for this computation, regardless of the ISEED value, but the minimum structural mass point does not converge consistently.

This situation is improved, somewhat, for the ISELECT = 1, ITRAN = 1 case (Fig. 4b) where the best structural mass endpoint values are obtained. However, this latter set of curves suffers dramatically in the middle portion of the pareto front where the two objectives form trade-offs with each other.

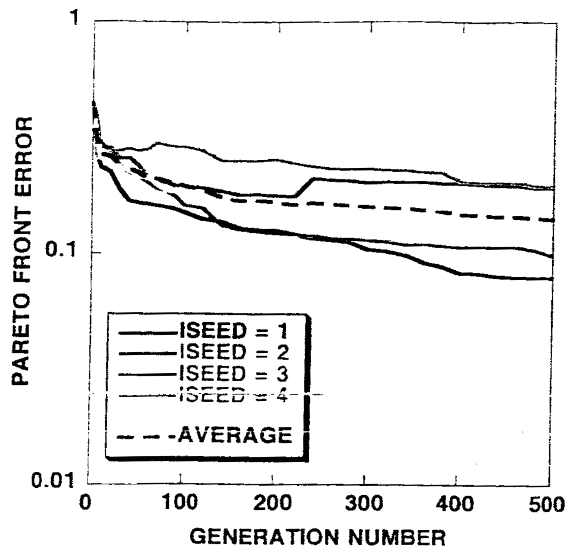
The best results for this multi-discipline isolated-wing optimization are realized when bin selection is utilized, ISELECT = 3 (Figs. 4c and 4d). The pareto fronts for both sets of curves that utilize bin selection more closely approximate the master curve than their counterparts using greedy selection. The set of curves in Fig. 4c—the ISELECT = 3, ITRAN = 0 case—are overall the best. Nevertheless, the minimum structural mass point for this set of computations still experiences inconsistency in convergence.

Convergence history comparisons for each of the pareto front curves presented in Figs. 4 are presented in Fig. 5. Computation of the pareto front error was achieved using the area error norm described above. The master pareto front displayed in Figs. 4 was used as an approximation to the "exact" solution. In addition, for each set of convergence history curves displayed in Figs. 5 there is an average convergence history curve plotted (the dashed curve), which is computed by arithmetically averaging each point in the convergence history data file.

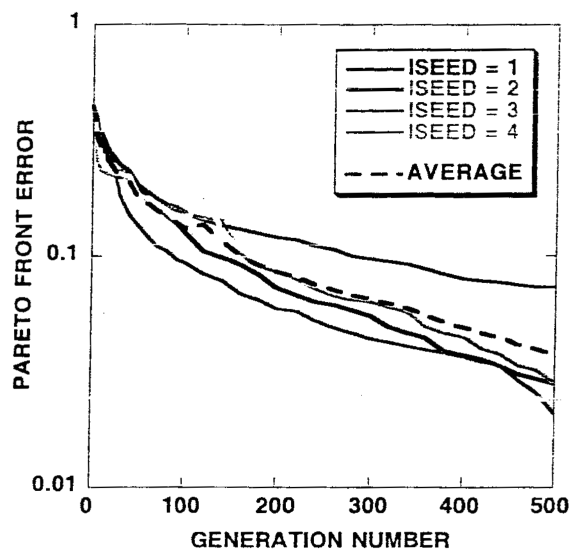
Note that occasionally the convergence history error increases with generation. This usually occurs early in the convergence process when the pareto front is still crudely defined with a small number of discrete points and is explained in the series of sketches shown in Figs. 6. Figure 6a shows a hypothetical two-objective pareto front, both the exact curve and a crude approximation to it that might exist after the search has proceeded for n generations. Figures 6b and 6c show two different scenarios for what the pareto front might look like after $n+1$ generations.

In both scenarios a single new point has been added which neither dominates nor is dominated by any of the previously existing points on the pareto front. In scenario 1 the new point's placement is less advantageous than that of scenario 2. Nevertheless, both new points are possible outcomes of the genetic algorithm search process. Despite the fact that scenario 1 represents an "enrichment" of the pareto front, the area error norm actually increases in going from the n^{th} generation to the $n+1^{st}$ generation. This is why the area error norm occasionally jumps—either up or down—during solution evolution.

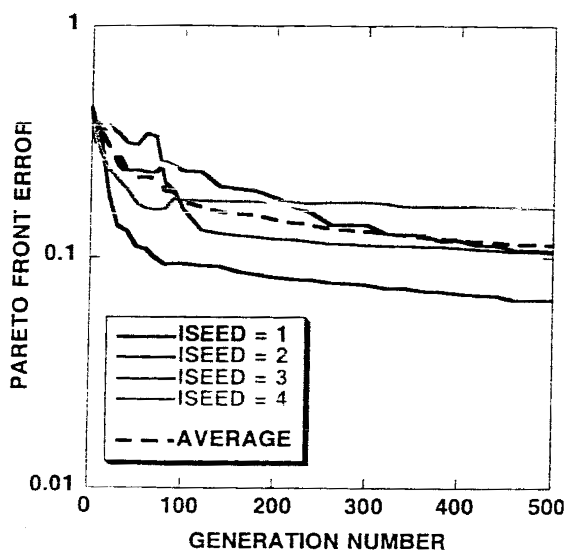
There is a moderately large amount of variance in each of the convergence history curves displayed in Fig. 5. This emphasizes the point that genetic algorithms are stochastically based. Comparison of results, especially convergence efficiency results, must be made with the proper amount of statistical averaging in place. Results from each MOGA variation demonstrate this conclusion.



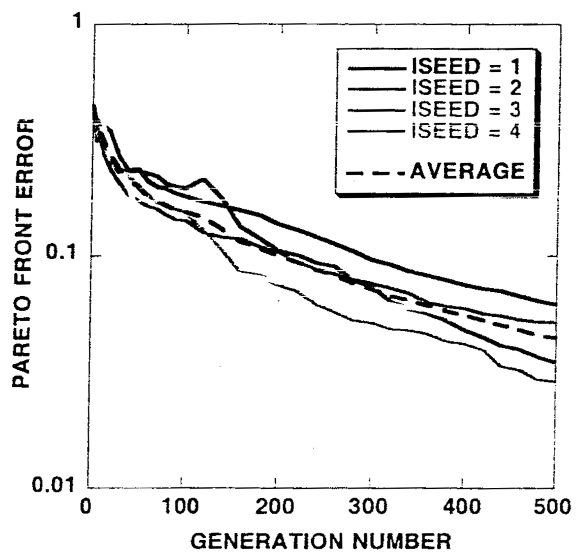
a) ISELECT = 1, ITRAN = 0



c) ISELECT = 3, ITRAN = 0



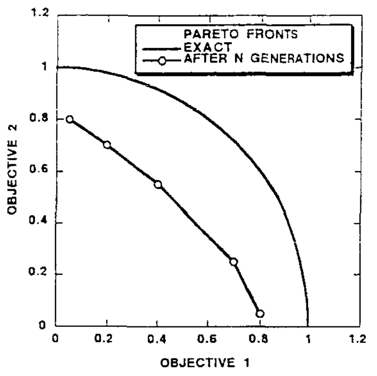
b) ISELECT = 1, ITRAN = 1



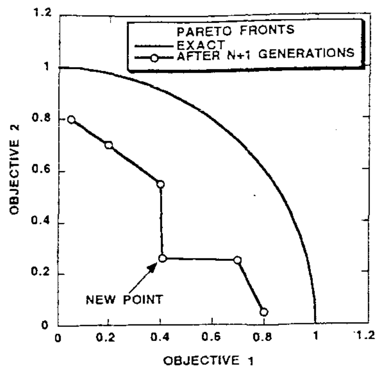
d) ISELECT = 3, ITRAN = 1

Fig. 5 Convergence efficiency comparisons for transonic wing optimization in which the lift-to-drag ratio at fixed lift and the inverse of the structural mass are simultaneously maximized. Five solutions utilizing different seed values are plotted for each GA variation, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

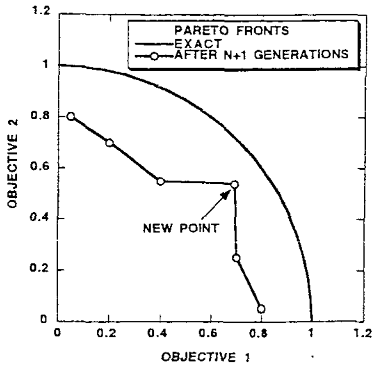
The averaged convergence history efficiencies for each of the four MOGA variations studied herein are presented on the same set of axes in Fig. 7. Note that the two variations that utilize bin selection are both faster than their greedy selection counterparts, achieving more than an order of magnitude reduction in the area error norm after 500 generations. The gene-space transformation variation produced mixed results, being slightly faster when used with greedy selection and slightly slower when used with bin selection.



a) Baseline, pareto front after n generations.



b) Pareto front after n+1 generations (scenario 1).



c) Pareto front after n+1 generations (scenario 2).

Fig. 6 Sketch of pareto front convergence history process showing two different advancement scenarios.

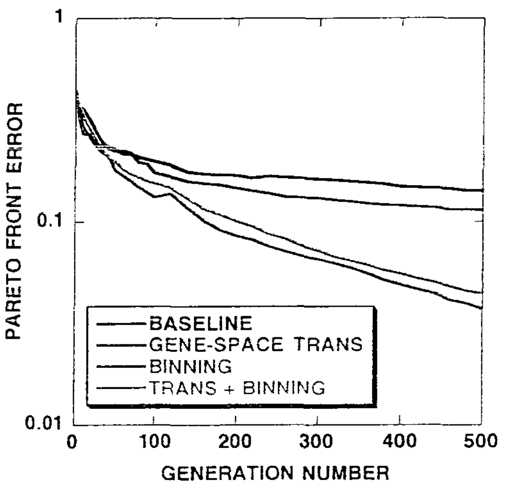


Fig. 7 Average convergence history curves for the four GA variations plotted in Figs. 4 and 5, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$ and $P = (0.04, 0.32, 0.32, 0.32)$.

Computed results for the two-objective isolated-wing optimization problem taken from the endpoints of a suitable ISELECT = 3, ITRAN = 0 pareto front are displayed in Figs. 8-10. Figure 8a shows the upper surface Mach number contours for the minimum drag point (at fixed lift), and Fig. 8b shows the same view for the minimum structural mass point. Note the extreme suction pressure (high Mach number) upstream of the shock wave in Fig. 8b. This is caused by a large amount of flow expansion on the forward part of the wing and results in a much stronger shock wave than the solution depicted in Fig. 8a.

This situation can be viewed in a more quantitative way by looking at cross-sectional plots of the pressure coefficient, which are displayed in Figs. 9 for three spanwise stations—$y/b = 0.21, 0.51, 0.78$. In each plot the pressure coefficient for the two pareto front endpoints—minimum drag and minimum structural mass—are displayed. From this series of figures it can be seen that the minimum drag case (solid curves) has a greatly reduced shock strength relative to the minimum structural mass case (dashed curves).

Both solutions have the same total lift. Despite this, the spanwise distribution of lift is different between the two cases. This can be seen from Figs. 9 by carefully examining the areas circumscribed by each cross-sectional pressure coefficient plot. The minimum structural mass solution (dashed curves) contains more area inboard and less outboard relative to the minimum drag point. This tends to move the center of pressure inboard for the minimum structural mass solution and thus, reduces the moment arm through which the wing lift acts. This, in turn, reduces the wing root bending moment and allows for a lighter structure.

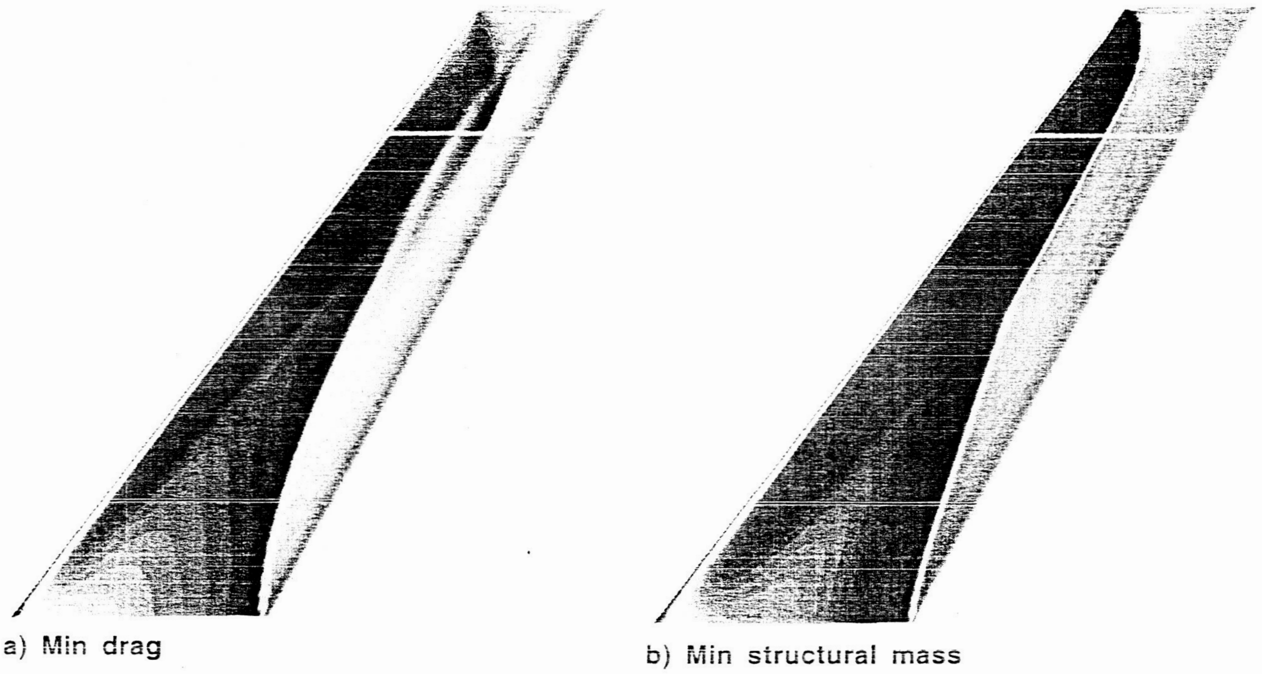


Fig. 8 Upper surface Mach number contours for a transonic wing optimization in which the drag at fixed lift and the structural mass are simultaneously minimized, ISELECT = 3, ITRAN = 0, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

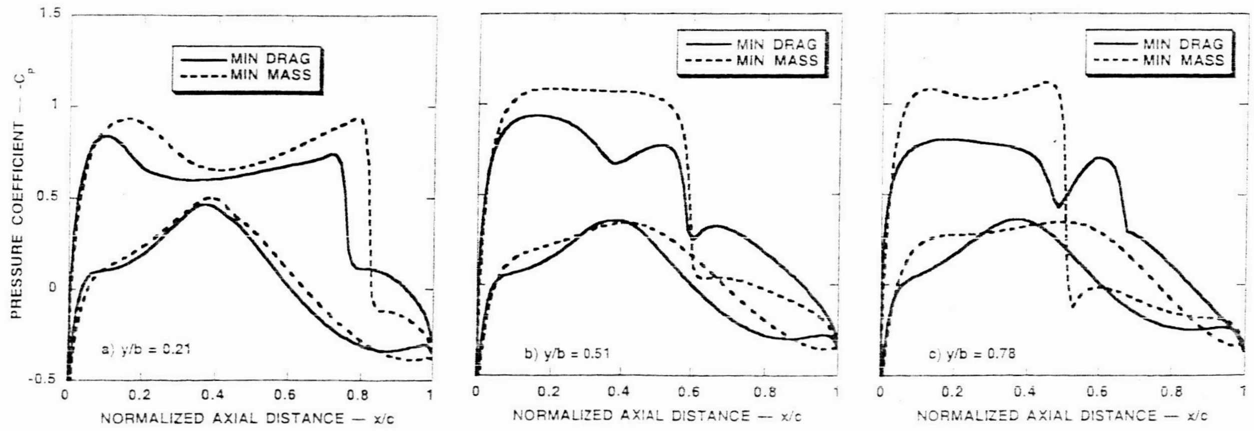


Fig. 9 Pressure coefficient distributions at selected spanwise stations for the wing optimization problem presented in Fig. 8 showing differences between the two pareto front endpoints, ISELECT = 3, ITRAN = 0, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

Another important attribute in this multi-discipline optimization problem is wing thickness—especially at the wing root. Selected airfoil cross-sectional profiles are presented in Fig. 10 at $y/b = 0.0, 0.48, 0.98$ (wing root, mid-span, wing tip). Profiles for both pareto front endpoints—minimum drag (solid curves) and minimum structural mass (dashed curves)—are presented. Note that the vertical axis had been greatly expanded to facilitate viewing of the results, and that each profile is displayed in rigged position with wing twist included. For this set of computations the minimum-drag root-airfoil thickness is the minimum allowed by the geometric constraints and the minimum-structural-mass root-airfoil thickness is the maximum allowed. This is as expected. A reduced wing-root thickness helps reduce drag by reducing shock strength and thus reduces wave drag. An increased wing-root thickness helps to reduce structural mass by more efficiently supporting the wing root bending moment with a box beam that has increased depth.

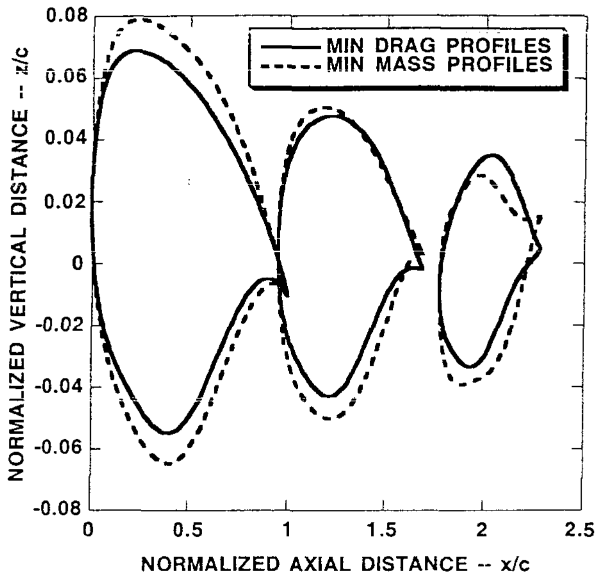


Fig. 10 Selected airfoil profiles in rigged positions—at y/b=0.0, 0.48 and 0.98—for the transonic wing optimization problem of Fig. 8 showing differences between the two pareto front endpoints, (note the expanded z/c scale), ISELECT = 3, ITRAN = 0, $M_\infty = 0.84$, $C_L = 0.45$, $N_C = 32$ $N_G = 22$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

Wing-fuselage results

The next set of computed results involves a two-objective optimization for a wing-fuselage configuration flying in the transonic flow regime—$M_\infty = 0.84$. The two objectives—lift-to-drag ratio at fixed lift and the configuration's volume—are simultaneously maximized by the previously presented MOGA optimization procedure. As before, the aerodynamic lift-to-drag ratio is evaluated for each candidate design, that is, for each chromosome, using the TOPS full potential solver[49] by iterating on angle of attack to obtain the lift-to-drag ratio at the specified value of lift—$C_L = 0.45$.

The wing-fuselage volume computation utilizes a straightforward algorithm that divides the fuselage into a series of cross-sections, computes the area of each cross-section using a simple triangularization and then computes the volume by integrating along the fuselage using trapezoid integration. The wing volume is computed using a similar algorithm that is deactivated for those wing cross-sections that lie inside the fuselage.

There are several reasons lift-to-drag ratio (at fixed lift) and wing-fuselage volume have been chosen as the two objective functions for the present optimization problem. First of all there is a clear-cut conflict between these two objectives. When the volume is maximized the lift-to-drag ratio suffers and vice versa.

Thus, we have a classical engineering trade-off problem. How the MOGA performs on such a problem with real-world objectives is the key aspect of this study, not the development of new aeronautical engineering concepts.

In addition, the vehicle volume was chosen as an objective because it is an easy quantity to understand. The direct computation of maximum volume is easy to perform, thus providing a simple check for the results obtained by the GA—at least for one endpoint on the pareto front. This problem is interesting in that it is complicated by transonic-flow shock-wave-induced nonlinearities and by a large number of decision variables that have large sensitivity variations.

The present wing-fuselage parameterization with two wing defining stations, $N_w = 2$, as described above, consists of 66 parameters. Airfoil defining station one is at the wing root and station two is at the wing tip. There are 24 Δr values used to define the fuselage—six equally-spaced axial stations, each with four equally-spaced meridinal locations. The axial positions are located at $x/c = 0.71, 1.43, 2.14, 2.86, 3.57, 4.29$ where the fuselage length is fixed at $x_B = 5.0$. The meridinal positions are located at $\varphi = -90°, -30°, 30°, 90°$ where $\varphi = -90°$ corresponds to the fuselage keel line and $\varphi = 90°$ corresponds to the crown line.

Maximum and minimum constraint values for each of the geometric parameters, as well as masking array values, are given in Table. 4. As can be seen, 43 of the 66 genes that make up each chromosome have masking array values of one, and 23 have values of zero, that is, only 43 genes are actually modified during the optimization process—$N_G = 43$. Note also that the max-min values associated with each of the Δr values are not symmetric about zero. Along the keel and crown lines, for example, any Δr value averaged between the max and min constraints, will be positive, and along the other meridinal stations (along the fuselage side), the same averaging will produce negative values. The max and min Δr constraints were chosen in this way to keep the vertical extent of the fuselage from becoming too small. A difficulty arises in the wing-fuselage line of intersection computation if the intersection line contacts the symmetry plane. This results in fuselage designs that have depth-to-width ratios that generally exceed unity—sometimes by a substantial amount.

The first wing-fuselage optimization results are presented in Figs. 11-14. For these results the freestream Mach number is 0.84 and the lift coefficient is held fixed at 0.45. The tolerance on the lift iteration is $\pm 1\%$, but in most cases the error is much less. The second binning selection algorithm with endpoint retention is used, $\text{ISELECT} = 4$, and the gene-space transformation procedure is turned off, $\text{ITRAN} = 0$. Both of the mutation probabilities are fixed at 0.2 and $P = (0.04, 0.32, 0.32, 0.32)$.

This optimization uses 34 chromosomes for each generation. The first element of the P vector is set such that two passthrough chromosomes are chosen during each generation—always the pareto front endpoints—the current minimum drag and the maximum volume chromosomes. Thus, 32 modified chromosomes require new function evaluations during each generation. These function evaluation computations are efficiently performed simultaneously on 32 processors of a parallel computer. The other three elements of the P vector are set to evenly divide the remaining chromosomes between the three remaining modification operators—random average crossover, perturbation mutation and mutation (at least as close to even as possible).

Development of the paréto front as a function of generation number is displayed in Fig. 11. Note that the maximum volume point on the pareto front, which occurs at a value of 1.42, converges quickly, achieving 96% of the theoretical maximum in as few as 50 generations, while the minimum drag point is slower in convergence. This characteristic, displayed in Fig. 11 for one ISEED value, occurred for every ISEED value utilized and is due to several factors.

Table 4. Maximum and minimum gene constraints along with masking array values for the wing-fuselage optimization problem.

Parameter	max	min	mask	Parameter	max	min	mask	Parameter	max	min	mask
Defining airfoil section 1				*Defining airfoil section 2*				*Fuselage perturbations*			
rle_1	0.014	0.014	0	rle_2	0.014	0.014	0	$\Delta r_{1,1}$	0.05	-0.02	1
xup_1	0.45	0.35	1	xup_2	0.45	0.35	1	$\Delta r_{2,1}$	0.05	-0.02	1
zup_1	0.08	0.05	1	zup_2	0.08	0.05	1	$\Delta r_{3,1}$	0.05	-0.02	1
$zxxup_1$	-0.2	-0.8	1	$zxxup_2$	-0.2	-0.8	1	$\Delta r_{4,1}$	0.05	-0.02	1
xlo_1	0.45	0.35	1	xlo_2	0.45	0.35	1	$\Delta r_{5,1}$	0.05	-0.02	1
zlo_1	-0.05	-0.08	1	zlo_2	-0.05	-0.08	1	$\Delta r_{6,1}$	0.05	-0.02	1
$zxxlo_1$	0.8	0.2	1	$zxxlo_2$	0.8	0.2	1	$\Delta r_{1,2}$	0.02	-0.05	1
zte_1	0.0	-0.0	0	zte_2	0.0	-0.0	0	$\Delta r_{2,2}$	0.02	-0.05	1
αte_1	8.0	0.0	1	αte_2	8.0	0.0	1	$\Delta r_{3,2}$	0.02	-0.05	1
βte_1	10.0	10.0	0	βte_2	10.0	10.0	0	$\Delta r_{4,2}$	0.02	-0.05	1
Planform defining station 1				Planform defining station 2				$\Delta r_{5,2}$	0.02	-0.05	1
c_1	1.0	1.0	0	c_2	0.5	0.5	0	$\Delta r_{6,2}$	0.02	-0.05	1
D_1	0.0	0.0	0	D_2	0.0	0.0	0	$\Delta r_{1,3}$	0.02	-0.05	1
Λ_1	37.0	37.0	0	Λ_2	37.0	37.0	0	$\Delta r_{2,3}$	0.02	-0.05	1
t_1	1.0	1.0	0	t_2	1.0	1.0	0	$\Delta r_{3,3}$	0.02	-0.05	1
θ_1	3.0	0.0	1	θ_2	0.0	-3.0	1	$\Delta r_{4,3}$	0.02	-0.05	1
$x_{\theta,1}$	0.5	0.5	0	$x_{\theta,2}$	0.5	0.5	0	$\Delta r_{5,3}$	0.02	-0.05	1
y_1	0.0	0.0	0	y_2	2.4	2.4	0	$\Delta r_{6,3}$	0.02	-0.05	1
Base fuselage parameters								$\Delta r_{1,4}$	0.05	-0.02	1
x_R	1.5	1.5	0	x_N	0.5	1.5	1	$\Delta r_{2,4}$	0.05	-0.02	1
y_R	0.0	0.0	0	r_B	0.35	0.35	0	$\Delta r_{3,4}$	0.05	-0.02	1
z_R	-0.15	0.15	1	x_T	0.5	1.5	1	$\Delta r_{4,4}$	0.05	-0.02	1
x_B	5.0	5.0	0	r_S	0.1	0.1	0	$\Delta r_{5,4}$	0.05	-0.02	1
								$\Delta r_{6,4}$	0.05	-0.02	1

First of all, the volume objective depends on most of the genes in a simple way. For example, if one of the Δr genes is increased, while all other genes are held fixed, the volume increases in a proportionate amount throughout the range of that gene's variation. This behavior exists no matter what value the other genes possess. In addition, the maximum volume values of many genes in the present wing-fuselage parameterization—certainly all of the fuselage Δr values—are at their upper constraint values. Finding an optimum on the design space boundary seems to be easier than finding an optimum within the design space interior.

The drag objective depends on the various genes in a more complex manner than that of vehicle volume —especially for this wing-fuselage problem. Interrelated and simultaneous changes in several genes are typically required in certain regions of the design space to achieve improvement in the drag objective. For example, when z_R is changed the position of the wing-fuselage juncture changes. To achieve optimal drag performance the wing-fuselage area ruling needs to be simultaneously adjusted, and this requires coordinated changes in various elements of the Δr array. Which elements have to be changed and by how much depends on the value of z_R.

The slow convergence of the drag pareto front endpoint for the present wing-fuselage problem, relative to the second objective, was not observed in the previous problem. In fact, the second objective for the previous isolated-wing optimization problem—minimization of the structural mass—was slower than that of the drag. There are two reasons for this apparent inconsistency. First of all, the complex interdependence of the drag objective on wing placement and the ensuing fuselage area ruling issues do not arise in the isolated-wing problem. Secondly, the dependence of the structural mass on the gene value distribution for the isolated-wing problem is every bit as complicated as for the drag objective—perhaps more so. That's because the structural mass depends on both the wing thickness distribution, especially near the wing root, as well as the wing pressure distribution—the latter quantity setting the wing's center of pressure and thus the moment arm for the wing root bending moment.

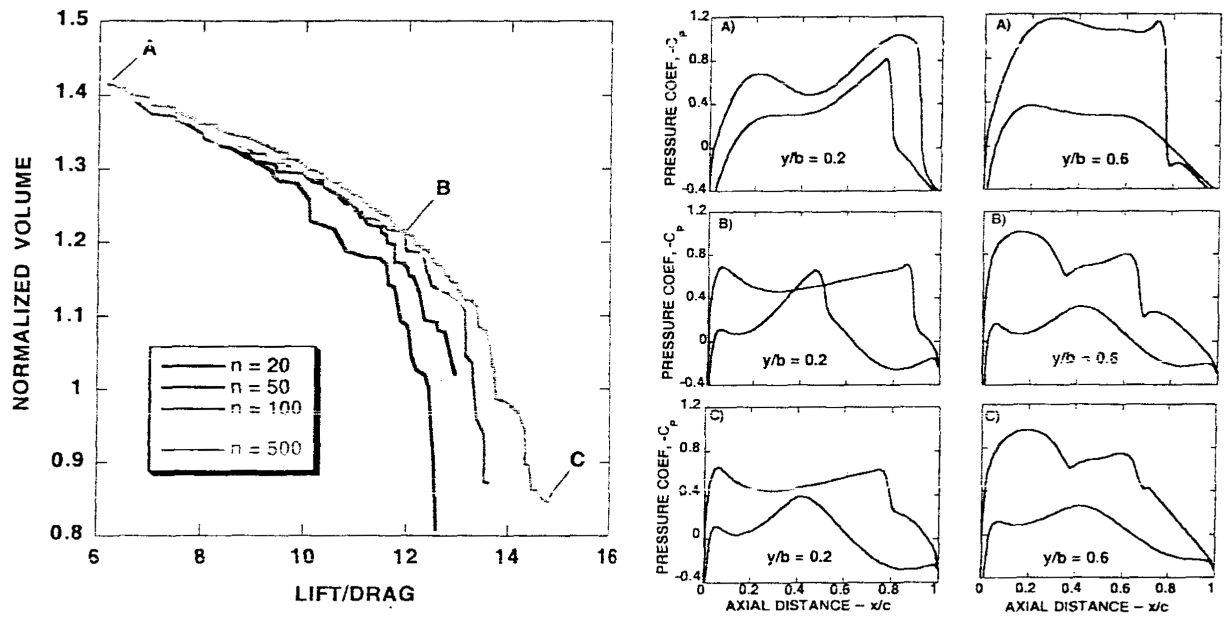


Fig. 11 Pareto front convergence and solution structure for a wing-body two-point optimization involving drag minimization and volume maximization both at constant lift, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_C = 34$, $N_G = 43$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$. **Wing sectional pressure distributions are presented at three positions along the pareto front, A) maximum volume, B) intermediate, C) minimum drag.**

Also displayed in Fig. 11 are several airfoil pressure distributions for three different solutions corresponding to three different locations along the pareto front, A) maximum volume, B) intermediate and C) minimum drag. For each solution two airfoil pressure distributions are presented—the first inboard near the wing-fuselage juncture ($y/b = 0.2$) and the second near mid-semi-span ($y/b = 0.6$). The most obvious feature from these results is the change in shock strength as the pareto front is traversed. The maximum volume solution (A) contains strong shocks, both on the upper and lower surfaces of the wing, while the minimum drag solution (C) contains mild shocks on only the wing's upper surface.

Upper surface Mach number contours for each of the three solutions presented in Fig. 11 are displayed in Figs. 12. This series of figures shows the shock wave pattern variation on the upper surface of the wing-fuselage configuration as the solution transitions from the maximum volume pareto front endpoint to the minimum drag endpoint. For the maximum volume endpoint there are strong shock waves, not only on the wing, as seen in Fig. 11, but also on the fuselage near the nose and tail. The fuselage shock waves are caused by the GA's selection of minimum values for the x_N and x_T genes. These values maximize the vehicle volume, but cause shock waves to form at both the fuselage nose and tail due to excessive over-expansion.

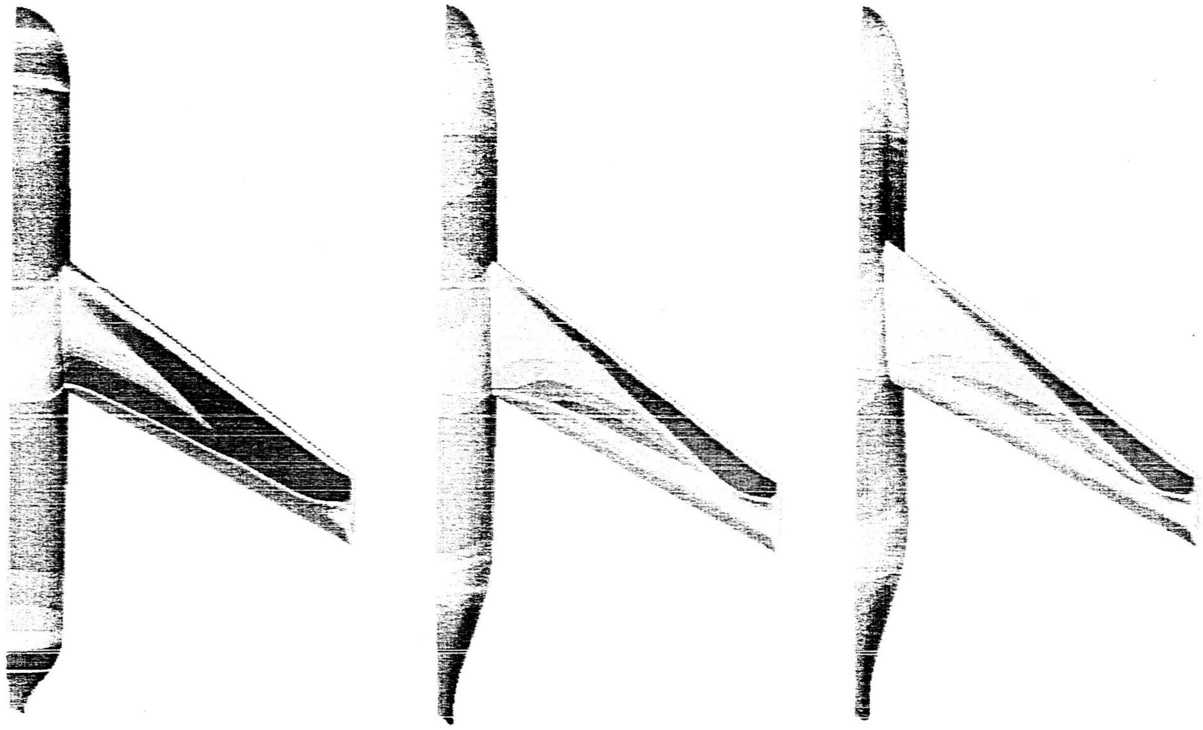


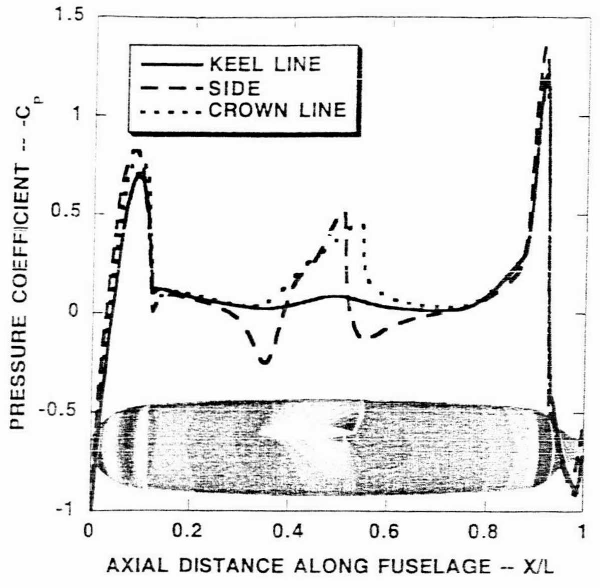
Fig. 12 Surface Mach number contours for the wing/body optimization problem presented in Fig. 11 at three positions along the pareto front.

As the solution propagates along the pareto front, trading volume to obtain a reduction in drag, the most noticeable initial change is associated with the fuselage nose and tail fairings. By choosing larger values for x_N and x_T, the drag is decreased without sacrificing a large amount of volume. This can be seen by examining the intermediate solution in Fig. 12 and noting its position on the pareto front in Fig. 11. Further reductions in drag are achieved via the complex task of fuselage area ruling in the vicinity of the wing-fuselage juncture. Such a result is observed by looking at the minimum drag solution in Fig. 12. More on this, including a comparison of selected fuselage cross-sections will be presented subsequently.

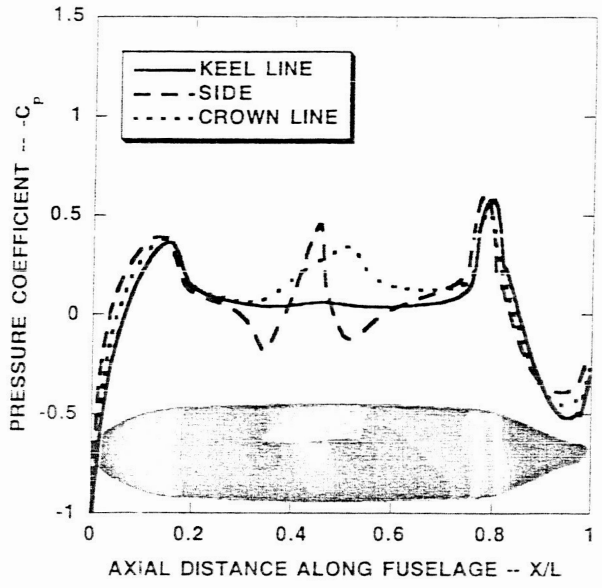
Pressure distributions along the fuselage for the three solutions described in Fig. 11 are presented in Figs. 13. Results are plotted along three fuselage meridinal stations—the keel line ($\varphi = -90°$), the side, ($\varphi = 0°$) and the crown line ($\varphi = 90°$). Each of these solutions involves a high mounted wing, that is, the MOGA optimizer has chosen a high mounted wing, regardless of the solution's position along the pareto front. More will be presented on this aspect subsequently. Because of this the side fuselage pressure distribution ($\varphi = 0°$) is plotted along a continuous meridinal line that lies below the wing for each of these three solutions. In addition, Mach number contours as viewed from the fuselage's side are also displayed in Figs. 13.

As can be seen from Fig. 13a—and previously discussed in conjunction with Fig. 12—the expansions, especially at the fuselage nose and tail, are quite large and lead to strong shock waves. As the fairings at the nose and tail are improved, see Figs. 13b and c, the expansions and resulting shocks are greatly reduced. The shock wave induced on the fuselage by wing-fuselage interference can also be seen in Fig.

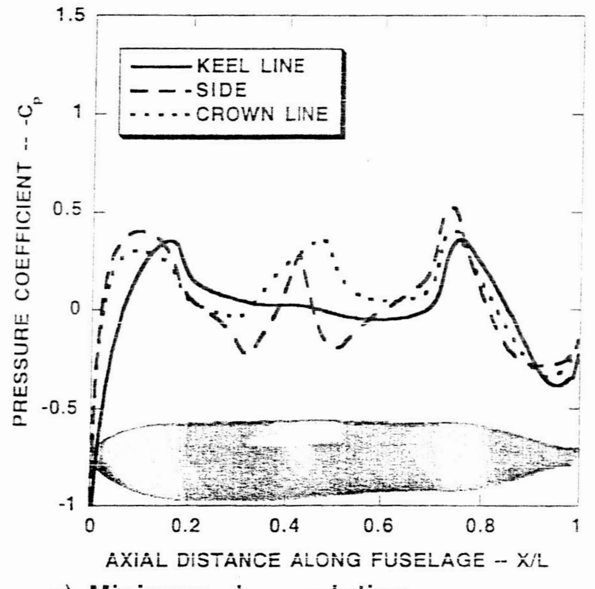
13a. Thinning of the wing, as well as the introduction of wing aft camber, reduces this effect in the intermediate solution (Fig. 13b). Fuselage area ruling reduces the wing-fuselage interference shock strength even further as the minimum drag point is approached (Fig. 13c).



a) Maximum volume solution.



b) Intermediate solution.



c) Minimum drag solution.

Fig. 13 Fuselage pressure distributions for the wing/body optimization problem presented in Fig. 11 at three positions along the pareto front.

By comparing each part of Figs. 13 with the corresponding part in Figs. 12, it can be seen that the depth of the fuselage, as measured from keel to crown is much larger than twice the half-width for any given longitudinal station. This is a direct result of the choices made in setting up the constraints for the Δr array, as was already discussed in the fuselage parameterization section, and produces fuselage cross-sections that are elongated in the z-direction. This situation is presented in a more quantitative way in Fig. 14.

A series of fuselage cross-sections are presented in Fig. 14 for the two-objective optimization problem displayed in Fig. 11. For each cross-section there are two curves plotted. The solid curve corresponds to the solution obtained at the minimum-drag pareto front endpoint, and the dashed curve corresponds to the solution obtained at the maximum-volume pareto front endpoint. For this solution the wing root leading edge is placed at $(x_R, y_R, z_R) = (1.5, 0.0, 0.15)$. Note that $x_R = 1.5$ and $y_R = 0.0$ are values set at the beginning of the computation and are held fixed, and that $z_R = 0.15$ is a value selected by the MOGA optimization for this particular chromosome from the range, $-0.15 \le z_R \le 0.15$ (see Table 4). The only fuselage cross-section in Fig. 14 that actually intersects with the wing is the one at $x/c = 1.54$. As seen from Fig. 14, the fuselage cross-section is drawn before the wing intersection is included.

From this set of plots it is easy to see how the fuselage was area ruled in the vicinity of the wing-fuselage juncture to obtain the final amount of drag reduction. Volume is subtracted along the upper portion of the fuselage near where the wing intersects, creating "pear–shaped" cross-sections in this fuselage location. Every cross-section from the maximum volume solution contains more area than the corresponding minimum drag cross-section. Each of the Δr values for the maximum volume solution is individually maximized.

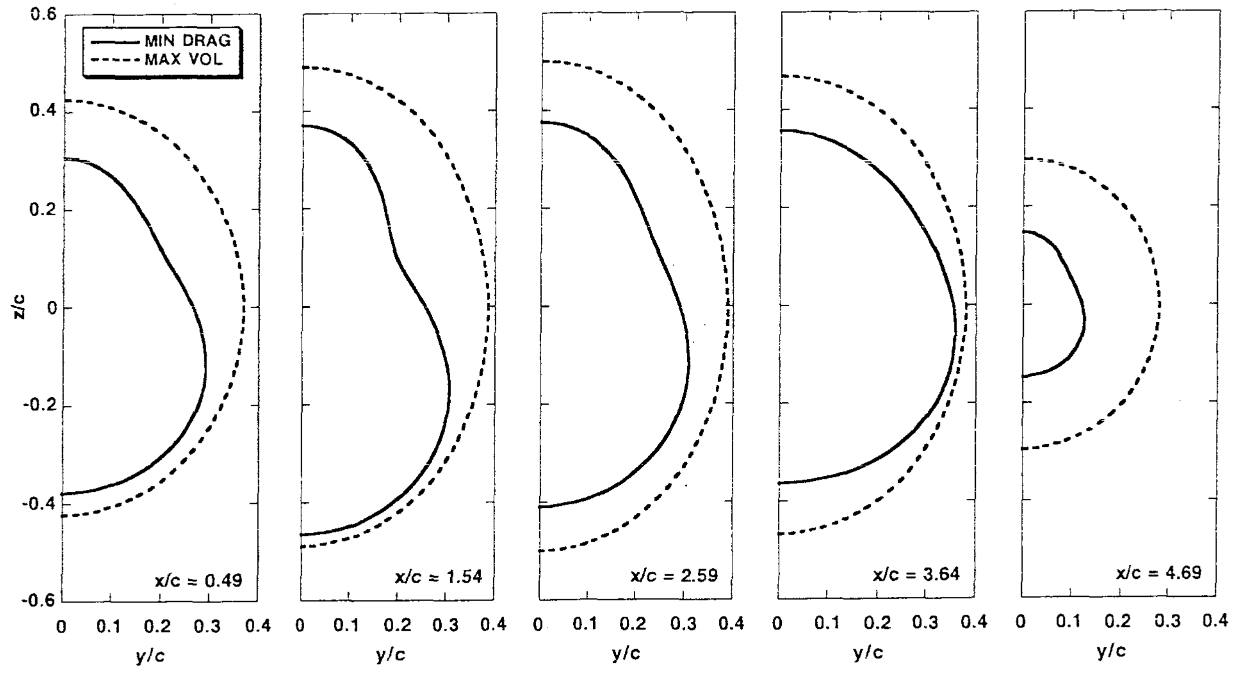


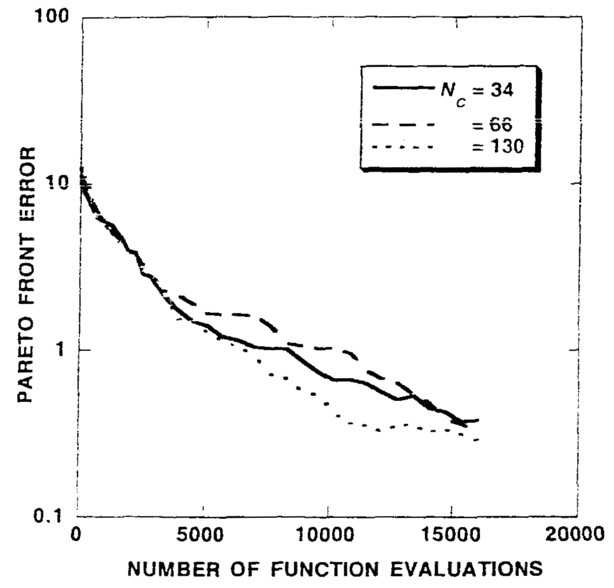
Fig. 14 Selected fuselage cross-sections for the wing/fuselage optimization problem of Fig. 11 showing differences between the two pareto front endpoints, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT $= 4$, ITRAN $= 0$, $N_O = 2$, $N_C = 34$, $N_G = 43$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$ **and** $P = (0.04, 0.32, 0.32, 0.32)$.

Convergence efficiency for the two-objective wing-fuselage optimization problem just discussed—drag minimization and volume maximization (both at fixed lift)—is presented in Figs. 15 and 16. Figure 15 shows the previously discussed area error norm plotted versus the number of function evaluations for three different convergence histories involving three different population sizes—34, 66 and 130 chromosomes, respectively. Each curve is averaged over two ISEED values in an attempt to eliminate some of the statistical variation. As can be seen each convergence history is nearly the same, indicating that convergence is not a function of population size. This is compatible with the results presented in

Holst and Pulliam[15] where a number of model problems were used to evaluate GA convergence for two-objective optimization.

For this set of computations the number of processors used to perform the function evaluations is equal to the number of chromosomes requiring function evaluations. There are always two passthrough chromosomes. The remaining chromosomes—those needing modification—are arbitrarily divided into three groups of equal size—or as close to equal as possible. One group is modified using crossover, the second using perturbation mutation and the third using global mutation. Since, two passthrough chromosomes are utilized for each population, the number of processors required to perform function evaluations is 32, 64 and 128, respectively. Using an increased number of chromosomes to define a given population allows an increased number of processors to be used for the optimization problem and thus, improves turn-around efficiency.

Figure 16 presents a comparison of the six pareto fronts computed from the convergence efficiency study presented in Fig. 15. These results are statistically similar to previous results presented in this section and suggest that there is no significant effect of population size on the final solution. Care should be taken in evaluating this conclusion, as the number of solutions needed to construct a proper statistical average has not been utilized in this case due to computer time limitations.



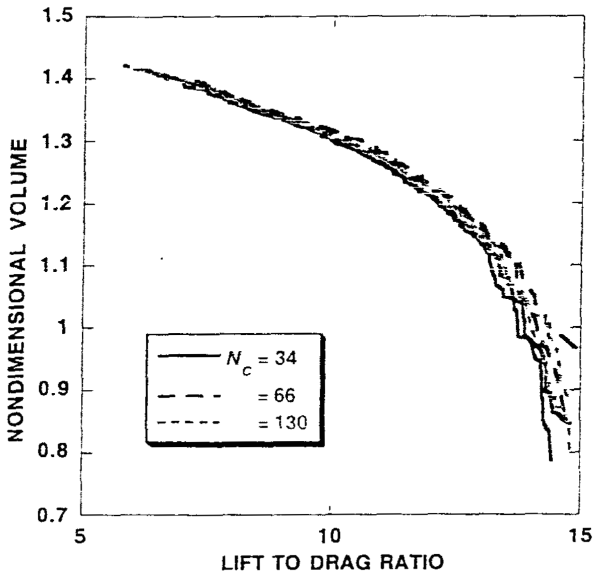
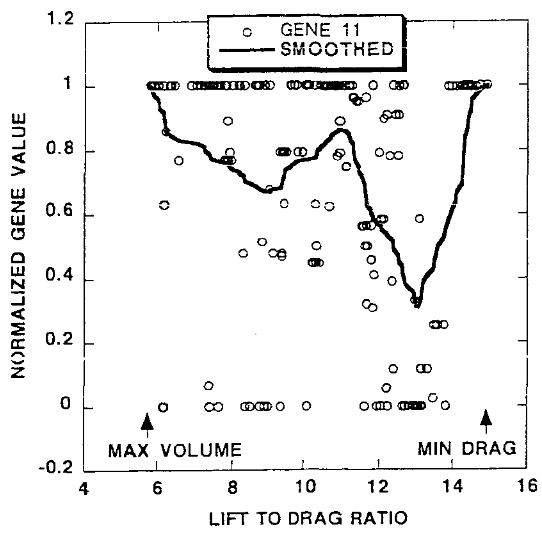


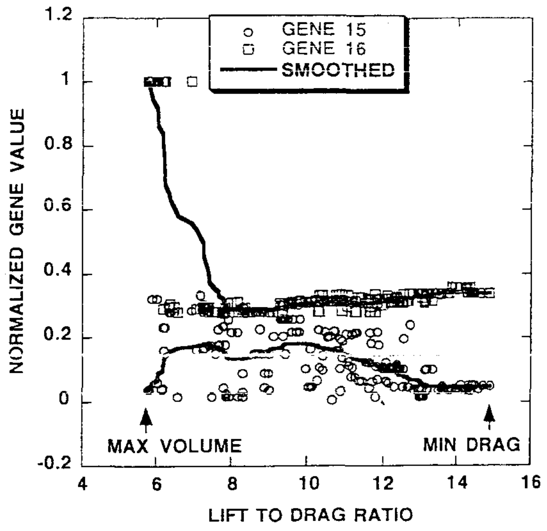
Fig. 15 Pareto front convergence history comparisons for three population sizes, each averaged over two ISEED values, wing/fuselage optimization problem. $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_G = 43$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$.

Fig. 16 Pareto front comparisons for three population sizes, each with two different ISEED values, wing/fuselage optimization problem, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_G = 43$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$.

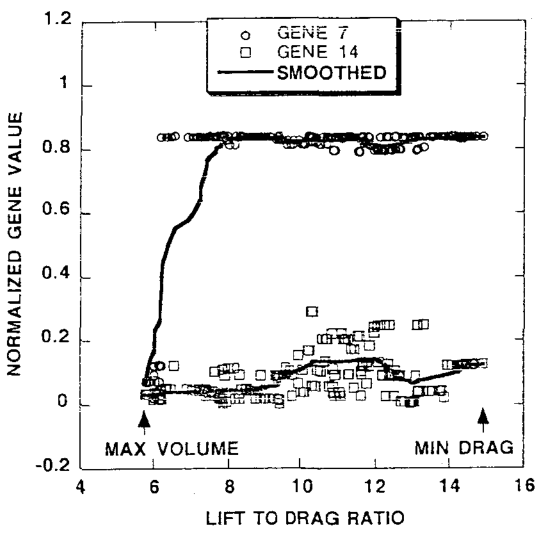
It is of interest to study how the individual genes vary along the pareto front. Selected genes from the wing-fuselage optimization problem described in Fig. 11 are plotted along the pareto front in Figs. 17 and 18. In particular, they are plotted as a function of the pareto front's first objective, lift-to-drag ratio. Each gene is normalized and then plotted so that it varies from 0 to 1. For many of the curves the discrete points are plotted along with a smoothed result. In some cases, where the number of curves on a single set of axes is large, only the smoothed curve is plotted.



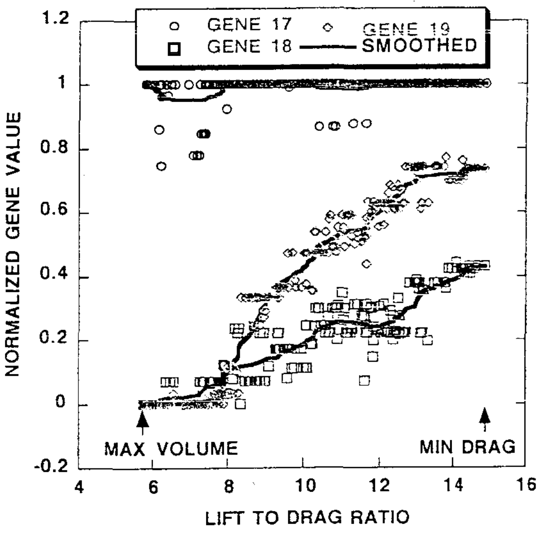
a) x-location of airfoil maximum thickness on the lower surface at the wing tip, $GENE_{11} = xlo_2$.



c) Airfoil twist at wing root and tip, $GENE_{15} = \theta_1$ and $GENE_{16} = \theta_2$.



b) Airfoil trailing edge angle at wing root and tip, $GENE_7 = \alpha te_1$ and $GENE_{14} = \alpha te_2$.



d) Vertical wing position ($GENE_{17} = z_R$), fuselage nose length ($GENE_{18} = x_N$) and fuselage boattail length ($GENE_{19} = x_T$).

Fig. 17 Normalized gene distributions along the pareto front from selected positions within each chromosome, wing/fuselage optimization problem, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_C = 34$, $N_G = 43$, $\beta = 0.1$, $\rho_1 = 0.2$, $\rho_2 = 0.2$, $P = (0.02, 0.33, 0.33, 0.32)$.

The first general observation is that a significant amount of statistical variation exists in many of the genes as they vary across the pareto front. Some genes wildly oscillate without any discernible pattern. The x-location of the wing-tip lower-surface maximum thickness—xlo_2, gene number 11—is such an example. Of the 43 available genes, xlo_2 contains the most variation. This gene is plotted in Fig. 17a and is seen to favor the maximum value in its allowable range. Nevertheless, it takes on many other values in a seemingly haphazard fashion.

The reason for this behavior is straightforward. This gene has little impact on either objective being optimized and thus has little impact on the pareto front. Any value is generally acceptable at any location along the pareto front. To test this observation several chromosomes were selected from the pareto front for which xlo_2 was against one of its constraints. The value of xlo_2 within that chromosome was then moved to the opposite constraint without changing any other gene value, and a new solution was computed. The resulting design-space point moved little and was still a member of the pareto front. Ideally, such a parameter shouldn't be utilized as a gene in the optimization process. In reality it is difficult to know *a priori* which genes will behave in this way and which will not.

Other parameters plotted in Fig. 17 include the trailing edge angle at the wing root and tip (Fig. 17b), the airfoil twist at the wing root and tip (Fig. 17c), and the vertical wing mounting position, the nose length and the boattail length (Fig. 17d). All of these gene parameters have reasonably controlled variations across the pareto front.

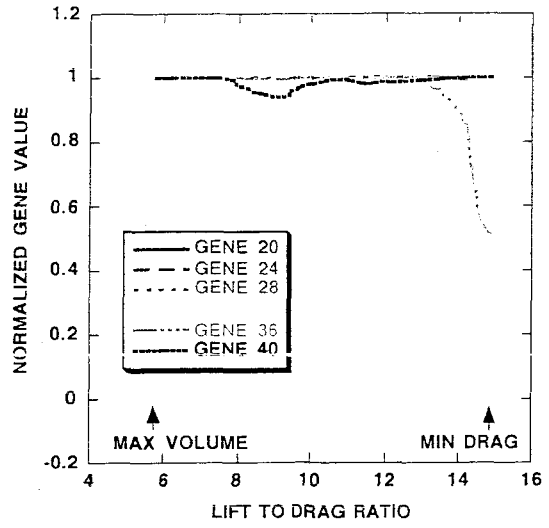
Figure 18 presents the variation of each of the Δr parameters for the wing-fuselage optimization problem—all 24 of them—across the pareto front. As already discussed, the Δr parameters are organized into four meridinal stations, each with six longitudinally distributed values. Each meridinal station is plotted on a separate set of axes in Fig. 18. Figures 18a, 18b, 18c and 18d show the Δr longitudinal variation along $\varphi = -90°, -30°, 30°$ and $90°$, respectively.

As expected, all Δr values at the maximum volume endpoint on the pareto front lie at their maximum constraint limits. Moving away from this endpoint, many of the Δr values decrease from their maximum constraint values and attain intermediate values. This is especially true for the Δr values along the $\varphi = 30°$ and $\varphi = 90°$ meridinal stations, where reductions in the various Δr values have been determined by the MOGA optimizer to achieve optimal fuselage area ruling in the vicinity of the wing-fuselage juncture.

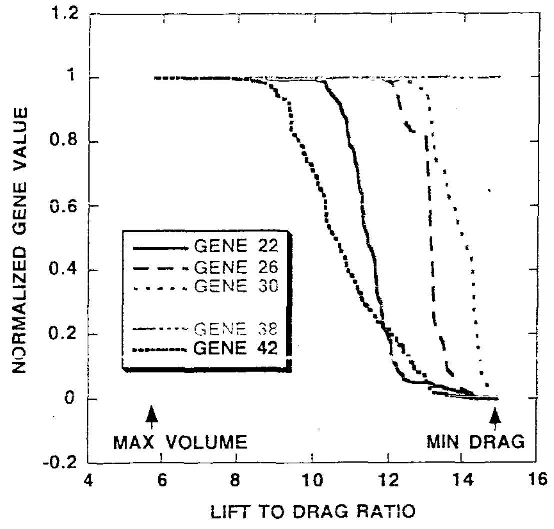
It is of interest to study the effect of wing placement on the overall wing-fuselage optimization problem. In the cases presented above, the vertical wing position, z_R, is constrained to lie between –0.15 and 0.15, while x_R and y_R are fixed at 1.5 and 0.0, respectively. As a result of the optimization, the value of z_R is chosen to lie at or near its upper constraint value, $z_R = 0.15$, over the entire pareto front (see the variation of gene 17 plotted along the pareto front in Fig. 17d). In other words, a high wing intersection is favored by the optimization regardless of position on the pareto front—at least in the context of the present design space construction.

Figures 19 and 20 present results for a wing-fuselage optimization in which the wing intersection is constrained to be low mounted, that is, the value of z_R is constrained to be –0.15. All other design space attributes and constraints are the same as in the previous wing-fuselage computations. This is implemented by setting the maximum constraint for z_R equal to the minimum constraint, $z_R = -0.15$, and by setting the corresponding z_R masking array value to zero. Thus, for this problem, $N_G = 42$, instead of the previous value of 43. Note also that $N_C = 66$ for this series of computations, that is, 64 processors were used in the optimization.

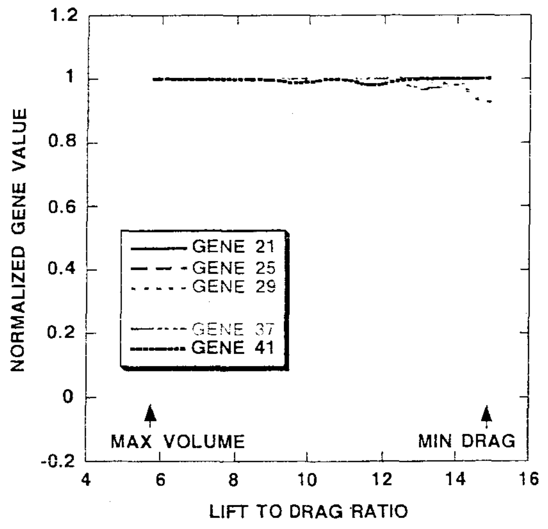
Figure 19 displays two computed pareto fronts for the original wing-fuselage optimization problem and two pareto fronts for the optimization that constrains the wing to be low-mounted. The two curves plotted for each case correspond to two different ISEED values. As can be seen, the low mounted wing results are less efficient in the vicinity of the minimum-drag pareto front endpoint. The low-mounted case achieves essentially the same level of drag, but only when vehicle volume is reduced. Note also that this effect is large in comparison to the statistical variation that exists between the two ISEED computations from each solution category. The effect of z_R on the maximum volume pareto front endpoint is negligible.



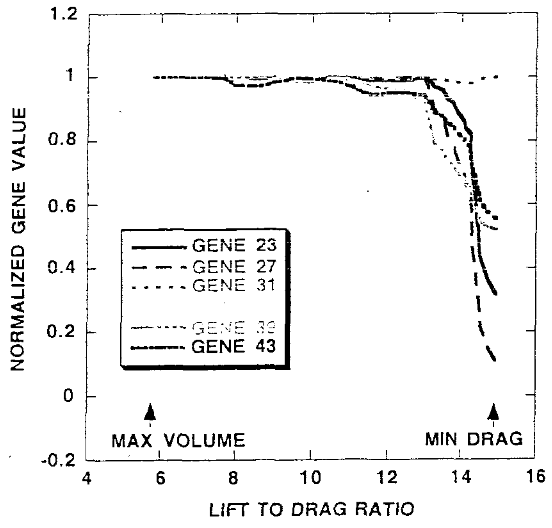
a) Axial distribution of Δr values along $\varphi = -90°$ (keel line)



c) Axial distribution of Δr values along $\varphi = 30°$



b) Axial distribution of Δr values along $\varphi = -30°$



d) Axial distribution of Δr values along $\varphi = 90°$ (crown line).

Fig. 18 Normalized gene distributions along the pareto front from the fuselage portion of the parameterization, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_C = 34$, $N_G = 43$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$, $P = (0.02, 0.33, 0.33, 0.32)$.

Also displayed in Fig. 19 are two circular symbols showing discrete chromosomes from each different category of pareto front that most closely match a Lift-to-drag ratio of 13. Selected fuselage cross sections taken from these two points are compared in Fig. 20. A total of five longitudinal stations are presented with $x/c = 0.49, 1.54, 2.59, 3.64, 4.69$. The solid curve corresponds to the original case in which the wing vertical mounting position was allowed to float. (Although for this chromosome the value of z_R that the MOGA selected was 0.15—a high mounted wing). The dashed curve corresponds to the case in which the wing was constrained to be low mounted. Despite being at the same L/D location on the pareto front, the two area rulings are quite different—quite different at each longitudinal station. Close examination shows how the MOGA modified the area ruling to account for the shifted wing mounting position, taking volume from the upper fuselage for the high-mounted case and from the lower fuselage for the low mounted case. It is also obvious that the fuselage associated with the low-mounted wing contains a smaller amount of volume as is consistent with Fig. 19.

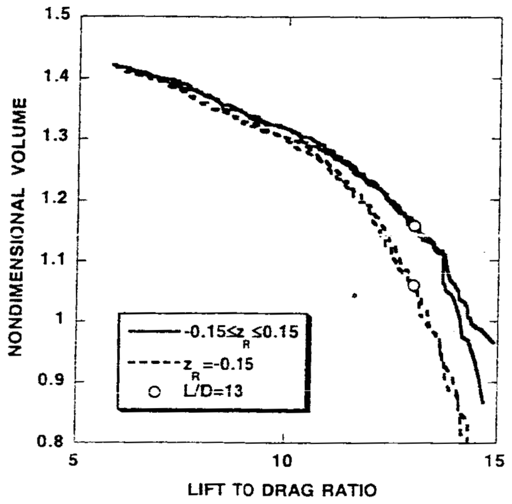


Fig. 19 Pareto front comparisons for two different wing-fuselage optimization cases: a) baseline in which wing position is allowed to float ($-0.15 \le z_R \le 0.15$) and b) wing fixed in the low-wing position ($z_R = -0.15$). Each computation performed with two ISEED values, $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_C = 66$, $N_G = 43/42$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$, $P = (0.02, 0.33, 0.33, 0.32)$.

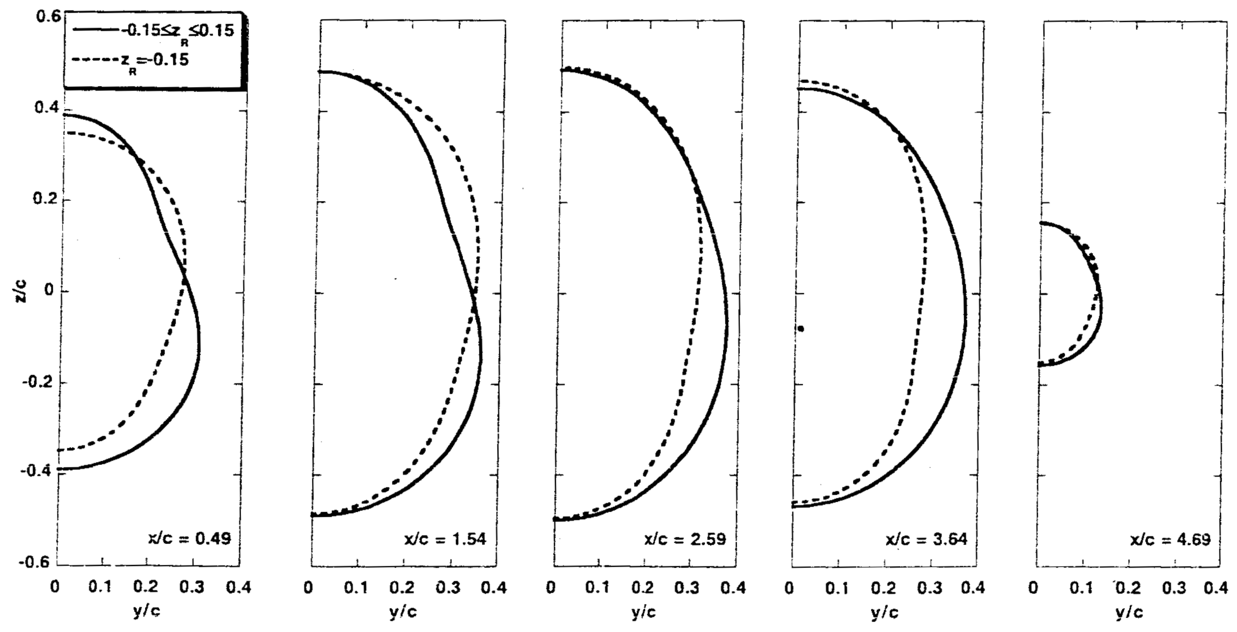


Fig. 20 Selected fuselage cross-sections for the two wing/fuselage optimization problems of Fig. 19 showing differences between the two solutions at $L/D = 13$: a) baseline in which wing position is allowed to float ($-0.15 \le z_R \le 0.15$) and b) wing fixed in the low-wing position ($z_R = -0.15$), $C_L = 0.45$, $M_\infty = 0.84$, ISELECT = 4, ITRAN = 0, $N_O = 2$, $N_C = 66$, $N_G = 43/42$, $\beta = 0.1$, $p_1 = 0.2$, $p_2 = 0.2$, $P = (0.02, 0.33, 0.33, 0.32)$.

Conclusions

A multi-objective genetic algorithm (MOGA) optimization procedure, with several selection algorithm options and a new gene-space transformation procedure, has been presented. It uses real-number encoding to represent all design space decision variables as genes and populations of fixed size to go from generation to generation. Four modification operators are utilized to advance from one generation to the next. They include passthrough, random average crossover, perturbation mutation and mutation. The standard output for this approach is a pareto front, which includes the best solutions from each objective, as well as a range of optimal "tradeoff" solutions in between.

The MOGA optimization procedure was utilized to determine a number of multi-objective optimal solutions for two problems. The first involved a transonic wing lift-to-drag maximization (drag minimization) coupled with the simultaneous minimization of structural mass (all at fixed lift). The second involved a transonic wing-fuselage lift-to-drag maximization (drag minimization) coupled with the simultaneous maximization of vehicle volume (all at fixed lift). In every case the GA optimization process was convergent. However, some of the GA variations studied converged more rapidly than others.

Of the two selection algorithms compared—greedy selection and bin selection—the latter produced the most efficient convergence across all of the problems studied. The gene-space transformation procedure produced mixed results in the area of GA convergence efficiency. In some cases it was moderately faster, for other cases moderately slower.

The new MOGA utilizes a masking array, which allows any gene (or set of genes) to be eliminated from the optimization process as a modifiable decision variable. This allows determination of the effect of a single gene (or gene subset) on the resultant pareto front.

A limited parallelization efficiency study was performed involving the use of 32 to 128 processors. Population sizes ranging from 34 to 130 chromosomes were used. The present MOGA is essentially embarrassingly parallel with regard to the number of chromosomes used for each generation, and the number of chromosomes did not affect convergence efficiency of the MOGA scheme. Thus, the turnaround time dramatically decreased with increasing number of processors.

References

1. Goldberg, D. E., "*Genetic Algorithms in Search, Optimization and Machine Learning*," Addison-Wesley, Reading, MA, 59-88, 1989.

2. Davis, L., "*Handbook of Genetic Algorithms*," Van Nostrand Reinhold, New York, 1991.

3. Beasley, D., Bull, D. R. and Martin, R. R., "An Overview of Genetic Algorithms: Part 1, Fundamentals," *University Computing*, Vol. 15, No. 2., 1993, pp. 58-69.

4. Beasley, D., Bull, D. R. and Martin, R. R., "An Overview of Genetic Algorithms: Part 2, Research Topics," *University Computing*, Vol. 15, No. 4., 1993, pp. 170-181.

5. Deb, Kalyanmoy, "Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems," *Evolutionary Computation*, Vol. 7, No. 3, 1999, pp. 205-230.

6. Van Veldhuizen, David and Lamont, Gary, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, Vol. 8, No. 2, 2000, pp. 125-147.

7. Jiménez, José, Cuesta, Pedro and Abderramán, Jesús, "Mixed Strategy in Genetic Algorithms: Domain's Reduction and Multirecombination," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

8. Obayashi, S. and Tsukahara, T., "Comparison of Optimization Algorithms for Aerodynamic Shape Design," *AIAA J.*, Vol. 35, 1997, pp. 1413-1415.

9. Bock, K.-W., "Aerodynamic Design by Optimization," Paper 20, AGARD CP-463, 1990.

10. Holst, T. L. and Pulliam, T. H., Evaluation of Genetic Algorithm Concepts using Model Problems, Part I: Single-Objective Optimization, NASA TM in preparation, 2003.

11. Zitzler, E., Thiele, L. Laumanns, M., Fonseca, C. M. and da Fonseca, V. G., "Performance Assessment of Multiobjective Optimizers: An analysis and Review," *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 2, April 2003, pp. 117-132.

12. Deb, K., Pratap, A. and Meyarivan, T., "Constrained Test Problems for Multi-Objective Evolutionary Optimization," Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, March 7-9, 2001, Zurich, Switzerland, pp. 284-298.

13. Van Veldhuizen, D., "Multiobjective Evolutionary Algorithms: Classification, Analyses, and New Innovations," AFIT/DS/ENG/99-01, June 1999.

14. Lohn, J. D., Kraus, W. F. and Haith, G. L., "Comparing a Coevolutionary Genetic Algorithm for Multiobjective Optimization, Proceedings of the 2002 IEEE Congress on Evolutionary Computation, May 2002, pp. 1157-1162.

15. Holst, T. L. and Pulliam, T. H., Evaluation of Genetic Algorithm Concepts using Model Problems, Part II: Multi-Objective Optimization, NASA TM 2003-212813, Dec. 2003.

16. Laumanns, M., Thiele, L., Deb, K. and Zitzler, E., "On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms," Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, TIK Report No. 108, June 2001.

17. Deb, K., Mohan, M. and Mishra, S., "A Fast Multi-Objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions," Kanpur Genetic Algorithms Laboratory (KANGAL), KanGAL Report No. 2003002, Feb. 2003.

18. Knowles, J. D. and Corne, D. W., "Approximating the Nondominated Front Using the Pareto Archived evolution Stategy," *Evolution Computation,* Vol. 8, No. 2, 2000, pp. 149-172.

19. Knowles, J. and Corne, D., "Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors," *IEEE Transactions on Evolutionary Computation,* Vol. 7, No. 2, April 2003, pp. 100-116.

20. Parmee, I. C., Cvetkovic, D., Gonham, C. R. and Mitchell, D., "Towards Interactive Evolutionary Design Systems for the Satisfaction of Multiple and Changing Objectives, *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

21. Marco, N, Désidéri, J.-A. and Lanteri, S., "Multi-Objective Optimization in CFD by Genetic Algorithms," Institut National de Recherche en Informatique et en Automatique, Research Report No. 3686, April 1999.

22. Naujoks, B., Willmes, L., Haase, W., Bäck, T. and Schütz, M., "Multi-Point Airfoil Optimization Using Evolution Strategies," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

23. Quagliarella, D. and Della Cioppa, A., "Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils," AIAA Paper 94-1896-CP, 1994.

24. Vicini, A. and Quagliarella, D., "Inverse and Direct Airfoil Design Using a Multiobjective Genetic Algorithm," *AIAA J.*, Vol. 35, 1997, pp. 1499-1505.

25. Hämäläinen, J., Mäkinen, A., Tarvainen, P. and Toivanen, J., "Evolutionary Shape Optimization in CFD with Industrial Applications," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

26. Epstein, B. and Peigin, S., "A Robust Hybrid GA/ROM Approach to multiogjective constrained Optimization in Aerodynamics," 16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida, AIAA Paper No. 2003-4092, June 2003.

27. Anderson, M., Burkhalter, J. and Jenkins, R., "Missile Aerodynamic Shape Optimization Using Genetic Algorithms," J. of Spacecraft and Rockets, Vol. 37, No. 5, Sept.-Oct. 2000, pp. 663-669.

28. Anderson, M. and Gebert, G., "Using Pareto Genetic Algorithms for Preliminary Subsonic Wing Design," AIAA Paper No. 96-4023-CP, 1996.

29. Sasaki, D, Obayashi, S. and Nakahashi, K., "Navier-Stokes Optimization of Supersonic Wings with Four Design Objectives Using Evolutionary Algorithm," AIAA Paper No. 2001-2531, 2001.

30. Oyama, A., "Multidisciplinary Optimization of Transonic Wing Design Based on Evolutionary Algorithms Coupled with CFD Solver," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

31. Ng, K. Y., Tan, C. M., ray, T. and Tsai, H. M., "Single and Multiobjective Wing Planform and Airfoil Shape Optimization using a Swarm Algorithm," 41st Aerospace Sciences meeting and Exhibit, Reno, Nevada, AIAA Paper No. 2003-45, Jan. 2003.

32. Obayashi, S., Yamaguchi, Y. and Nakamura, T., "Multiobjective Genetic Algorithm for Multidisciplinary Design of Transonic Wing Planform," *J. of Aircraft,* Vol. 34, 1997, pp. 690-693.

33. Oyama, A. and Liou, M.-S., "Multiobjective Optimization of Rocket Engine Pumps Using Evolutionary Algorithm," AIAA Paper No. 2001-2581, June 2001.

34. Benini, E., "Three-Dimensional Multi-Objective Design Optimization of a Transonic Compressor Rotor," 16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida, AIAA Paper No. 2003-4090, June 2003.

35. Oyama, A. and Liou, M., "Multiobjective Optimization of a Multi-Stage Compressor Using Evolutionary Algorithm," AIAA Paper No. 2002-3535, 2002.

36. Rai, M. M., "Towards a Hybrid Aerodynamic Design Procedure Based on Neural Networks and Evolutionary Methods," 20th AIAA Applied Aerodynamics Conference, St. Louis, MO, AIAA Paper No. 2002-3143, June 2002.

37. Madavan, N. K., "Aerodynamic Shape Optimization Using Hybridized Differential Evolution," 21st AIAA Applied Aerodynamics Conference, Orlando, FL, AIAA Paper No. 2003-3792, June 2003.

38. Giotis, A. P., Giannakoglou, K. C. and Périaux, J., "A Reduced-Cost Multi-Objective Optimization Method Based on the Pareto Front Technique, Neural Networks and PVM," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

39. Tursi, S., "Transonic Wing Optimization Combining Genetic Algorithm and Neural Network," 21st AIAA Applied Aerodynamics Conference, Orlando, Florida, AIAA Paper No. 2003-3787, June 2003.

40. Vicini, A. and Quagliarella, D., "Airfoil and Wing Design Through Hybrid Optimization Strategies," AIAA Paper No. 98-2729, 1998.

41. Brown, M. and Smith, R. E., "Effective Use of Directional Information in Multi-Objective Evolutionary Computation," Genetic and Evolutionary Computation Conference (GECCO 2003), July 2003.

42. Oyama, A., "Wing Design Using Evolutionary Algorithms," PhD Thesis, Dept. of Aeronautics and Space Engineering, Tohoku University, Senadi, Japan, March 2000.

43. Houck, G. R., Joines, J. A. and Kay, M. G., "A Genetic Algorithm for Function Optimization: A Matlab Implementation," North Carolina State University-IE, TR 95-09, 1995.

44. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, AI Series, Springer-Verlag, New York, 1994.

45. Noble, B., *Applied Linear Algebra*, Prentice Hall, Englewood Cliffs, New Jersey, 1969, pp. 314-318.

46. Sobieczky, H., "Parametric Airfoils and Wings," *Recent Development of Aerodynamic Design Methodologies-Inverse Design and Optimization*, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, Germany, 1999, pp. 72-74.

47. Hicks, R. and Henne, P., "Wing Design by Numerical Optimization," *J. Aircraft,* Vol. 15, 1978, pp. 407-412.

48. Fonseca, C. M. and Fleming, P. J., "On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers," *Parallel Problem Solving From Nature IV,* edited by Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P., Springer, Berlin, Germany, 1995, pp. 584-593.

49. Holst, T. L., "Multizone Chimera Algorithm for Solving the Full-Potential Equation," *J. of Aircraft*, Vol. 35, No. 3, May-June 1998, pp. 412-421.

50. Oyama, A., "Wing Design Using Evolutionary Algorithms," PhD Thesis, Dept. of Aeronautics and Space Engineering, Tohoku University, Senadi, Japan, March 2000.