Chapter 1
Introduction

1.1 Background

The aircraft industry, like others, is increasingly exposed to considerable commercial pressures: Boeing Company and Airbus Industrie are struggling for supremacy on the large-size civil airliner (seating more than 100 passengers) market, while many companies, such as Bombardier, Embraer, Dornier, Dasa, are fighting for a larger share of the expanding regional jet aircraft market. Since the success of such commercial products depends on cost and timeliness as well as quality, the design process is being reengineered to save cost and time scales.

With advances in Computational Fluid Dynamics (CFD) and computer hardware, CFD has become an integral part of the aircraft design process. CFD has contributed to cut aerodynamic design cost and time scales by reducing the number of required wind tunnel tests. However, it is just an instrument for estimating aerodynamic performance of a given aircraft configuration. On the whole, the basis of the design process is trial and error, and the success of the final design depends on the knowledge and intuition of the designer. CFD technology will be able to display its ability to the full when it is coupled with numerical optimization methods by displacing any human interactions in the design procedure.

Yet, despite the fact that numerical optimization methods have been successfully used for a countless number of design problems, an application of numerical optimization to aerodynamic design still remains as a formidable challenge because of the following difficulties:

1) Objective function landscape of an aerodynamic optimization is often multimodal and nonlinear: there are many local optima, plateaus, or ridges even in a simplified problem [1]. This is because the flow field is governed by a system of nonlinear partial differential equations expressing the conservation of mass, momentum, and energy.

2) Function evaluations using a CFD code, especially a three-dimensional Euler or Navier-Stokes code, are very expensive. An aerodynamic evaluation of a simple wing with a Navier-Stokes solver, for instance, can take more than an hour of CPU time even on a vector computer.

Aerodynamic design problems with these properties require a numerical optimization tool to be very robust and efficient as well.

The application of numerical optimization methods coupled with CFD to transonic aerodynamic shape designs was pioneered by Hicks *et al.*, where airfoil shapes were designed using a gradient-based method coupled with a potential flow solver [2]. Since then, the gradient-based methods have been widely used for wing design [3], scramjet nozzle design [4], supersonic wing-body design [5], and more complex aircraft configurations [6,7]. These methods use the gradient of an objective function with respect to changes in the design variables to calculate a search direction using the finite difference methods [8] or adjoint formulations [9]. These methods are efficient in searching optimums. Not only that, the optimum obtained from these methods will be a global one, if the objective and constraints are differentiable and convex (Kuhn-Tucker condition). Distribution of an objective function of an aerodynamic design problem, however, is usually multimodal, and thus, one could only hope for a local optimum neighboring the initial design point by using the gradient-based method. Therefore, to find a global optimum, one must start the optimization process repeatedly from a number of initial points and check for consistency of the optima obtained. In this sense, the gradient-based methods are neither efficient nor robust for design automation.

Evolutionary Algorithms (EAs) are emergent optimization algorithms mimicking mechanism of the natural evolution, where a biological population evolves over generations to adapt to an environment by selection, crossover and mutation. When EAs are applied to optimization problems, fitness, individual and genes usually correspond to an objective function value, a design candidate, and design variables, respectively. One of the key features of EAs is that they search from multiple points in the design space, instead of moving from a single point like gradient-based methods do. Furthermore, these methods work on function evaluations alone and do not require derivatives or gradients of the objective function. These features lead to the following advantages:

1) Robustness: EAs have capability of finding a global optimum, because they don't use function gradients that direct the search toward an exact local optimum. In addition, EAs have capability to handle any design problems that may involve non-differentiable objective function and/or a mix of continuous, discrete, and integer design parameters.

2) Suitability to parallel computing: Since EAs are population-based search algorithms, all design candidates in each generation can be evaluated in parallel by using the simple master-slave concept. Parallel efficiency is also very high, if objective function evaluations consume most of CPU time. Aerodynamic optimization using CFD is a typical case.

3) Simplicity in coupling CFD codes: As these methods use only objective function values of design candidates, EAs do not need substantial modification or sophisticated interface to the CFD code. If an all-out re-coding were required to every optimization problem, like the adjoint methods, extensive validation of the new code would be necessary every time. EAs can save such troubles.

Owing to the above advantages over the analytical methods, EAs have become increasingly popular in a broad class of design problems (for example, see [10]). EAs have been also successfully applied to aeronautical design problems including conceptual and preliminary design of aircraft [11,12], preliminary design of turbines [13]. Table 1.1 outlines some of aerodynamic shape optimization problems where EAs coupled with CFD have been successfully applied.

Table 1.1 Aerodynamic design problems solved by EAs and CFD

Researchers	Year	Application area
Quagliarella, D. and Cioppa, A. D.	1994	Airfoil shape design using a potential solver [14]
Yamamoto, K. and Inoue, O.	1995	Airfoil shape design using a Navier-Stokes solver [15]
Cao, H. V. and Blom, G. A.	1996	Multi-element airfoil shape design using a Navier-Stokes solver [16]
Obayashi, S. and Oyama, A.	1996	Subsonic wing shape design using a Navier-Stokes solver [17]

The previous applications of EAs, however, are restricted to more or less simplified problems involving not more than 10-30 design parameters. In contrast to that, in real-world design problems, a large number of design parameters must be handled – for example, a wing shape for a generic transonic aircraft usually parameterized by more than a hundred of design parameters. Since such problem has highly multidimensional search space and extremely complicated objective function distribution, standard EAs would fail to find a globally optimum. Unfortunately, an EA capable to find a global optimum in such a large-scale design problem has not been developed yet.

1.2 Thesis Objectives

The objective of this research is to develop a new, robust, and efficient numerical design method applicable to real-world aerodynamic design problems. To achieve this goal, the following three key items will be investigated:

Goal 1: Improvement of EA search ability through the adaptive search range mechanism.

In the standard EAs, the upper and lower limits of the search regions should be given by the designer in advance to the optimization process. In general, a needlessly large search region is used in fear of missing the global optimum outside of the search region. Therefore, if the search region is able to adapted toward a promising area during the optimization process, the performance of EAs will be enhanced greatly.

Goal 2: Improvement of EAs by using the structured coding based on epistasis analysis.

When EAs are used to solve an engineering optimization problem, complexity in the objective function distribution appears as interactions among design parameters, which are often referred as *epistasis*, corresponding to the term used in biology. Therefore, if the epistatic interaction structures of the design parameters are identified in advance, the robustness and efficiency of EAs will be improved by properly defining the coding structure of the design parameters.

Goal 3: Transonic wing design optimization based on EAs

To ensure the ability of the present EA in large-scale aerodynamic design optimizations, an aerodynamic design optimization of a transonic wing shape for generic transport aircraft will be demonstrated. Aerodynamic performances of the design candidates will be evaluated by using the three-dimensional compressive Navier-Stokes equations to guarantee an accurate model of the flow field. Structural constraint was introduced to avoid an apparent solution of zero thickness wing for low drag in high speeds.

Goal 4: Supersonic wing design optimization based on EAs

To examine the ability of the EAs in supersonic wing design optimizations, an EA will be applied to an aerodynamic wing shape design for a supersonic transport.

1.3 Evolutionary Algorithms

1.3.1 What's Evolutionary Algorithm?

Evolution is a phenomenon of adapting to the environment and passing genes to next generations. In *On the Origin of Species by Means of Natural Selection* [18], Charles Darwin did not know about the underlying genetics, but he identified three basic principles driving natural evolution; reproduction, natural selection, and diversity of individuals, maintained by variations from one to the next generation. These features of natural evolution have found entrance to a broad class of evolutionary algorithms that mimic biological evolution and natural selections. It was 1960s that American and European researchers gave birth to stochastic search methods inspired by Darwinian evolution theory, all independently of each other: they are Evolutionary Programming (EP), Evolutionary Strategies (ESs), and Genetic Algorithms (GAs).

One of them, EP goes back to the work of L. J. Fogel [19]. He initially studied this method to develop the artificial intelligence and succeeded in evolving a mathematical automaton that predicts a binary time series. Later, in the middle of 80's, his son David Fogel further developed it to solve more general tasks including prediction problems, optimization, and machine learning [20]. Since this approach modeled organic evolution at the level of evolving species, the original EP does not rely on any kind of recombination. In general, each parent generates an offspring by the Gaussian mutation and better individuals among parents and offspring are selected as parents of the next generation. EP has been applied successfully for the optimization of the real-valued functions [21,22] and other practical problems [23]. For more details see [24].

ESs were developed by Rechenberg and Schwefel at the Technical University of Berlin and have been extensively studied in Europe [25-28]. While EP has derived for pure scientific interest, motivation of this study is, from the beginning, to solve engineering design problems: Rechenberg and Schwefel developed ESs in order to conduct successive wing tunnel experiments for aerodynamic shape optimization. Their important features are threefold:

1. ESs use real-coding of design parameters since they model the organic evolution at the level of individual's phenotypes.
2. ESs depend on deterministic selection and mutation for its evolution.
3. ESs use strategic parameters such as on-line self-adaptation of mutability parameters.

Some examples of early applications of ESs are: optimal dimensioning of the core of a fast sodium-type breeder reactor [29], shape optimization of vaulted reinforced concrete shells [30], arm prosthesis design [31], optimization of a thermal water jet propulsion system [32].

GAs owe their name to an early emphasis on representing and manipulating individuals at the level of genotype instead of phenotypic representation. In Holland's original work [33], GAs were proposed to understand adaptation phenomena in both natural and artificial systems and they have three key features that distinguish themselves from other computational methods modeled on natural evolution:

1. The use of bitstring for representation
2. The use of crossover as the primary method for producing variants
3. The use of proportional selection

Soon these methods are used to solve design optimization problems including discrete design parameters and then real parameter optimization problems. Early applications of GAs are optimization of gas pipeline control [34], structural design optimization [34], aircraft landing strut weight optimization [35], keyboard configuration design [36], etc. It should be mentioned that GAs have contributed to establish the schema theorem and recognize the role and importance of crossover through attentive theoretical analysis.

Though all three algorithms, EP, ESs, and GAs, originally have their distinguishing characteristics according to the level of organic evolution modeling, such classification is not useful anymore. For example, many GA practitioners have abandoned bitstring for floating-point representations. ES practitioners often use crossover operators as a reproduction operator. Application of EP is no longer limited to the evolution of finite state machines. New evolutionary approaches such as genetic programming [37] have been rising. Therefore, today all these systems are often collectively called “Evolutionary Algorithm (EA)” or “Evolutionary Computation (EC).”

1.3.2 Representation of Individuals

The first decision in applying an EA to seek optimal values for continuous variables is how to represent design parameters of an individual. Roughly speaking, there are two classes of representations: binary representations and floating-point representations.

The use of the binary representation originates in GAs that use a bitstring to model an individual. When a bitstring is used to represent an individual, however, it is required to transform real design parameters into binary numbers. Since binary substrings representing each parameter with the desired precision are concatenated to form a chromosome for EAs, the resulting chromosome encoding a large number of design variables would result in a huge string length. For example, for 100 variables with a precision of six digits, the string length is about 2000. EAs would perform poorly for such design problems. In addition, the binary representation of real design parameters presents the difficulty of so-called hamming cliffs, which comes from discrepancy between the representation space and the problem space. For instance, two points close to each other in the representation space might be far in the binary represented problem space. As a consequence, EAs using the binary representation is unable to focus the search effort in a close vicinity of the current population. It is still an open question to construct efficient evolutionary operators that suit to such a modified problem space.

The use of the floating-point representation originates in EP and ESs. In the floating-point representation, an individual is characterized by a vector of real numbers. It is more natural to use the floating-point representation for real parameter optimization problems because it is conceptually closest to the real design space, and moreover, the string length is reduced to the number of design variables. It has been reported that real-coded EAs outperformed binary-coded EAs in many design problems [38,39]. Therefore, all EAs used in this study adopt the floating-point representation. An example of binary and floating-point representations is illustrated in Fig. 1.1.

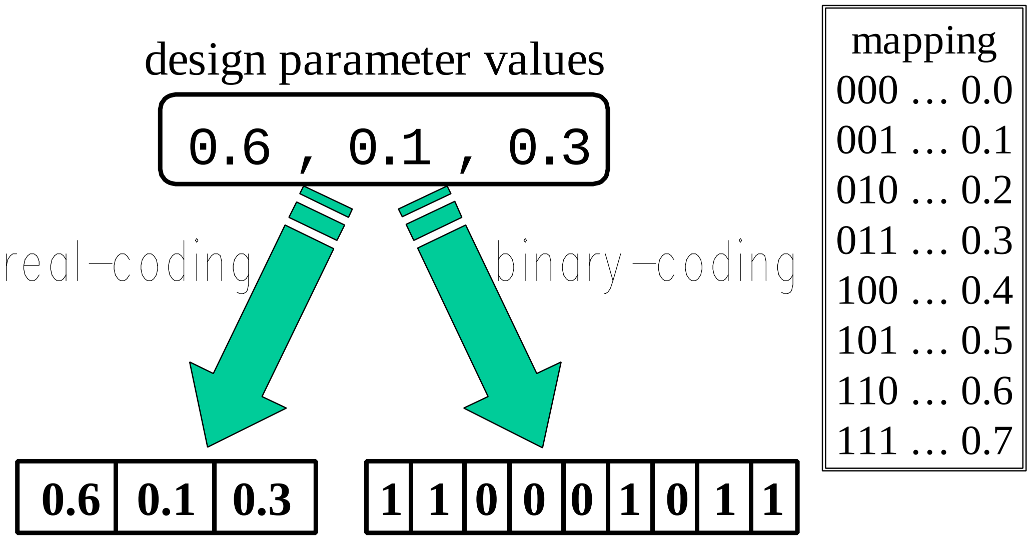

Fig. 1.1 An example of binary and floating-point representations

1.3.3 Mechanics of Evolutionary Algorithms

In spite of the diversity in EAs, they have common components: iterative selection biased by fitness, recombination, and mutation. Selection is a process in which design candidates are selected for mating based on their fitness values. It may include alternation of generations and selection of mating partners. The new design candidates are then generated by a recombination operator and a mutation operator. Figure 1.2 illustrates flowchart of EAs. Each item in the figure will be presented in the following subsections.

1.3.3.1 Initialization

An initial population of design candidates is generated. This often is accomplished by random sampling from the design space.

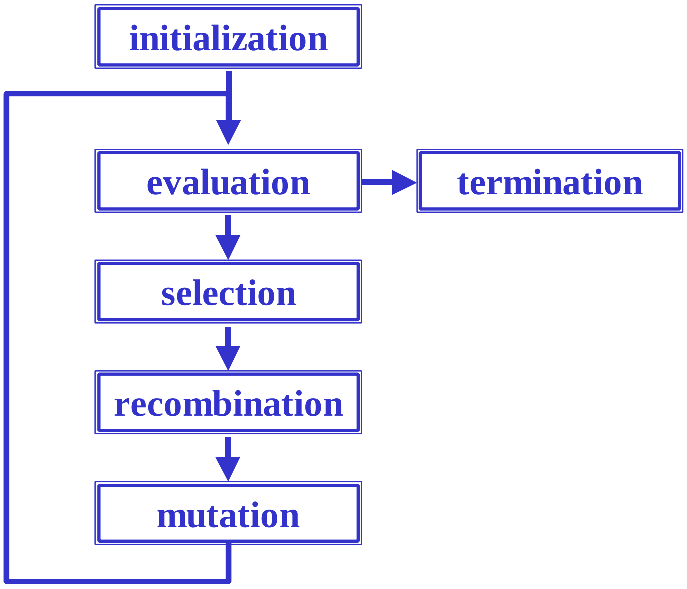

Fig. 1.2 Main flowchart of EAs

1.3.3.2 Evaluation

The fitness of all individuals is evaluated by objective function values. When CFD codes are used to calculate aerodynamic performances, this phase consumes most of the CPU time.

1.3.3.3 Selection

Selection is a process in which fitter individuals are selected to reproduce offsprings for the next generation. In nature, an individual undergo two different selection pressures before producing its offspring, i.e., survival to adult state and find of its mates. Selection of EAs models these processes.

1.3.3.3.1 Alternation of generations (survival to adult state)

This phase selects N individuals from both the parents and children to be stored in the mating pool where the total number of the parents and children is greater than the population size N. In non-overlapping systems, parents and offspring never compete with each other. The parent population is always replaced by the offspring population without any selection process; the lifetime of each individual is one generation. When a non-overlapping system is adopted, selection pressure is put only in the next parental selection phase.

Overlapping models, on the contrary, give selection pressure where parents and children compete for survival. Selection pressure, however, varies considerably from one overlapping model to another that includes strategies such as random selection, the best-N selection [40,41], and niche-formation methods. In the random selection, N individuals are selected randomly and thus, no selection pressure is put. As a result, selection pressure is required in the parental selection phase. In the best-N selection, the N best individuals are selected from both the parents and children. This selection brings selection pressure according to their fitness values. In the niche-formation methods, N individuals are drawn from both the parents and children according to their similarity with each other. Because the diversity is important for evolution, deletion of similar individuals will help to maintain it. Typical example is De Jong's crowding scheme [42] where some parents similar to the offsprings are replaced by them.

1.3.3.3.2 Selection of mating pairs (parental selection)

This phase selects pairs of individuals from the mating pool that will produce offsprings for the next generation. The commonly used is to assign each individual a probability of selection on the bias of its fitness. Goldberg and Richardson also suggest selection according to fitness modified by sharing function for the maintenance of diversity in the population[1] [43].

Fitness-proportional selection A widely-used method is the fitness-proportional selection [34]. In this method, the selection probability of each individual is calculated by dividing its fitness by the sum of the fitness of all individuals. Then, the parents are selected by either roulette-wheel selection [34] or Stochastic Universal Sampling (SUS) [44]. Figure 1.3 shows the operation of the roulette-wheel selection that assigns a portion of the wheel proportional to the selection probability and starts spinning the roulette wheel: each time, a single individual is selected.

The most important concern in a stochastic selection is to prevent loss of population diversity due to its stochastic aspect so-called *genetic drift*. For instance, in the above example, it is possible that only the individuals I-1 and I-2 are selected to produce all offsprings. The basic consideration of the SUS is to prevent genetic drift by selecting a number of parents each wheel spin. Figure 1.4 shows an example where four parents are selected at a single wheel spin from four individuals.

---

[1] Detailed description of the sharing function will be given in subsection 1.4.4.

individual	fitness	selection probability
I-1	4	0.4
I-2	3	0.3
I-3	2	0.2
I-4	1	0.1

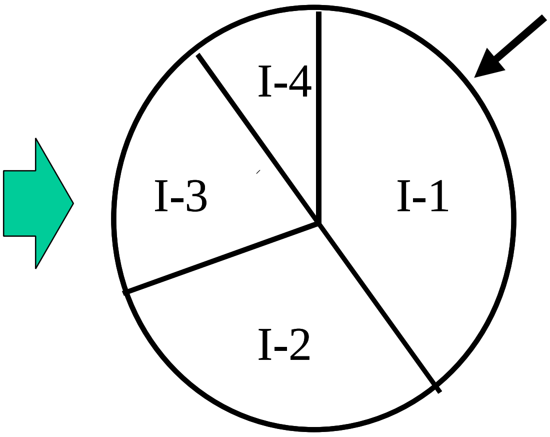

Fig. 1.3 Roulette-wheel selection

individual	fitness	selection probability
I-1	4	0.4
I-2	3	0.3
I-3	2	0.2
I-4	1	0.1

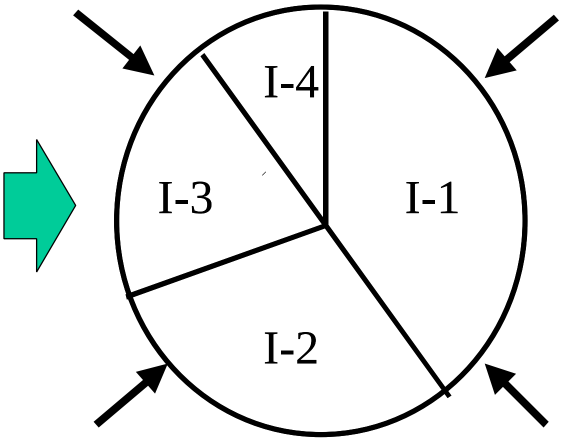

Fig. 1.4 Stochastic universal sampling

Ranking selection Ranking selection assigns selection probabilities on an individual's rank, ignoring absolute fitness value [45]. In this method, the individuals in the mating pool are sorted according to their fitness values and then assigned a count that is solely a function of their rank. The best individual receives rank 1, the second best receives 2, and so on. The selection probability is reassigned according to rank, for example, as an inverse of their rank values. In [45], Michalewicz proposed a nonlinear function to assign the selection probability as,

$$\boldsymbol{prob} = \boldsymbol{c} \cdot (1-\boldsymbol{c})^{(\boldsymbol{rank}-1)} \tag{1.1}$$

where c is a user-defined parameter. Then, the parents are selected by either roulette-wheel selection or SUS.

Tournament selection Tournament selection [46] operates by choosing some individuals randomly from a population and selecting the best from this group to survive into the next generation. Binary tournaments where tournaments are held between pairs of individuals are the most common.

Many other selection techniques are implemented into EAs. But why is the choice of selection techniques so important? The answer is that an appropriate level of selection pressure is the key to a successful evolution. An evolution under too strong selective pressure leads premature loss of diversity and results in premature convergence of an EA. It is well known that if the roulette-selection is used, diversity in a population is likely to be lost in early generations due to domination of the entire population by a few super individuals that are much better than the average fitness. On the contrary, low selective pressure can make the search ineffective. Thus, it is important to find a proper balance and various selection techniques attempt to achieve this.

1.3.3.4 Recombination

Recombination is a process in which new individuals are generated by exchanging features of the selected parents with the intent of improving the fitness of the next generation. This process is sometimes called crossover, especially when genotypic representation is used. These new individuals are then subjected to mutation. There are a number of different ways in which the recombination operation can be implemented. The following describes some of the recombination operators for the floating-point representation.

Blend crossover (BLX-α) The most common approach for recombination of two parents represented by a vector of real numbers is BLX-α proposed by Eshelman and Schaffer [47]. In this approach, children are generated on a segment defined by two parents, but the segment may be extended equally on both sides determined by a user specified parameter α. Thus, a child solution is expressed as:

$$\boldsymbol{Child}1 = g \cdot \boldsymbol{Parent}1 + (1 - g) \cdot \boldsymbol{Parent}2 \tag{1.2}$$

$$\boldsymbol{Child}2 = (1 - g) \cdot \boldsymbol{Parent}1 + g \cdot \boldsymbol{Parent}2 \tag{1.3}$$

where

$$g = (1 + 2a) u - a \tag{1.4}$$

Child1, *Child2* and *Parent1*, *Parent2* denote design parameters of the children and parents, respectively. The uniform random number u in [0,1] is regenerated for every design parameter. Figure 1.5 gives schematic view of BLX-α. For example, BLX-0.5 samples children that lie on an interval that extends 0.5d on either side of the interval between the parents. BLX-0.0, on the other hand, is equivalent to Radcliffe's flat crossover [48] that uniformly samples children between the two parents.

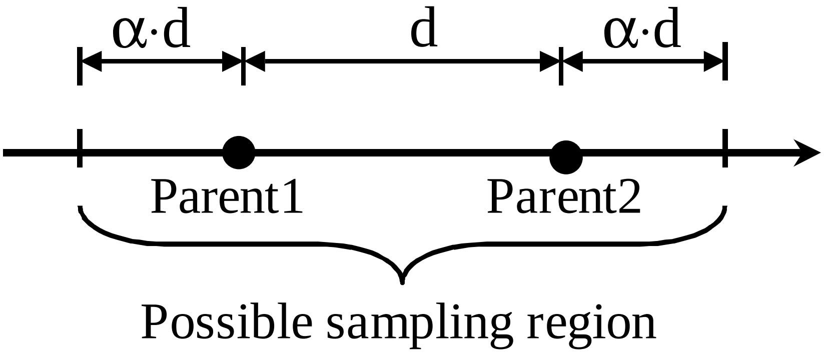

Fig. 1.5 BLX-α

When an EA is applied to a design optimization problem, what is important is balance of two conflicting goals: exploiting good solutions and exploring the search space [49]. An extreme example is the random search that explores the search space without any exploitation of the promising regions. As a result, random search enhances the reliability in multimodal situations, but reduces the convergence rate. Another extreme is gradient-based methods that exploit the region adjacent to the initial solution but neglects exploration of the search space. Such a search likely ends up with finding a local optimum. In this sense, the parameter α plays very important role in BLX-α, because α represents the proportion of exploration to exploitation. BLX-0.0, for instance, exploits the region between two parents but ignores exploration. Thus, BLX-0.5 is usually used in which both exploration and exploitation are carried out equally.

Simulated binary crossover (SBX) SBX [50] is developed by Deb *et al.* by incorporating the essence of self-adaptation mechanism of ESs into crossover operator. This approach is similar to BLX-α in the sense that children are created in proportion to the difference in parents but SBX has an unique property – that is, solution points near the parents are more likely to be selected as their offsprings than solution points distant from the parents. In SBX, the children solutions are expressed as follows:

$$\boldsymbol{Child}1 = 0.5\left[(1+b_q)\cdot \boldsymbol{Parent}1 + (1-b_q)\cdot \boldsymbol{Parent}2\right] \tag{1.5}$$

$$\boldsymbol{Child}2 = 0.5\left[(1-b_q)\cdot \boldsymbol{Parent}1 + (1+b_q)\cdot \boldsymbol{Parent}2\right] \tag{1.6}$$

where the blending parameter b_q is found so that the area under a specified probability distribution function *P(b)* from 0 to b_q is equal to a uniform random number *u* defined in [0,1]:

$$b_q = \begin{cases} (2\boldsymbol{u})^{\frac{1}{h+1}} & \boldsymbol{if\_u} \le 0.5 \\ \left(\dfrac{1}{2(1-\boldsymbol{u})}\right)^{\frac{1}{h+1}} & \boldsymbol{otherwise} \end{cases} \tag{1.7}$$

$$\boldsymbol{P}(b) = \begin{cases} 0.5(h+1)b^{h} & \boldsymbol{if\_}b \le 1 \\ 0.5(h+1)/b^{h+2} & \boldsymbol{otherwise} \end{cases} \tag{1.8}$$

where

$$b = \left|\frac{\boldsymbol{Child}1 - \boldsymbol{Child}2}{\boldsymbol{Parent}1 - \boldsymbol{Parent}2}\right| \tag{1.9}$$

The distribution index *h* is any nonnegative real number. In [51], it was reported that an EA with SBX outperformed both ESs and an EA with BLX-0.5 on some test functions.

Conventional Crossovers This kind of crossover operators include one-point, two-point, multi-point, and uniform crossovers. In the one-point crossover for the bitstring representation [34], the bits are swapped between the two parents in segments at a random point of a chromosome in between the bits. In the floating-point representation, the vector components of two parents are swapped in groups at a random space of a vector between the vector components. For example, given a five dimensional space, *Parent1* has a vector (x1, x2, x3, x4, x5) and *Parent2* has a vector (y1, y2, y3, y4, y5) and the crossover point was selected to be 3, then the offsprings will carry vectors (x1, x2, x3 y4, y5,) and (y1, y2, y3, x4, x5). An example of one-point crossover is illustrated in Fig. 1.6. Two-point, multi-point, and uniform crossovers for the floating-point implementation can be defined in the same manner.

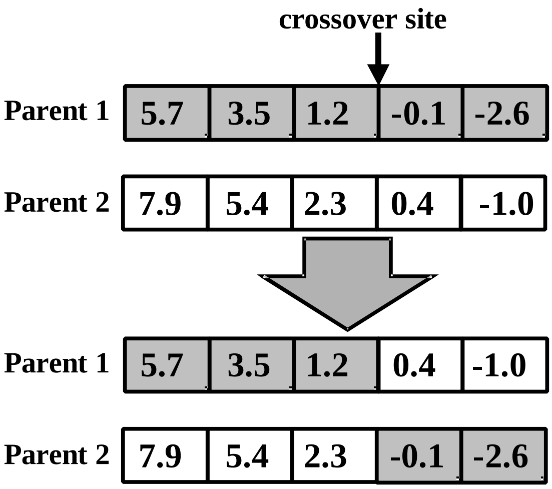

Fig. 1.6 One-point crossover for floating-point representation

Evolutionary Direction Operator In general, EAs are good at finding a neighborhood of a global optimum but poor at locating an exact optimum within the neighborhood. As a consequence, convergence of an EA becomes slower as it approaches to the optimum. Evolutionary direction operator suggested by Yamamoto and Inoue [52] can improve the convergence by estimating the direction of evolution from the selected parents. Figure 1.7 schematically shows a generation of an offspring by the evolutionary direction operator in a two-dimensional optimization problem. By using a reference point denoted as "Present", two directions are determined from Parents 1 and 2. An offspring will be generated in the quadrilateral obtained in the directions of increasing fitness values. The genes of the offspring C_o of the present individual C with the fitness value of F can be written by the genes and the fitness of the presents (C_{p1},F_{p1}) and (C_{p2},F_{p2}) as:

$$\begin{aligned} \boldsymbol{C_o} &= \boldsymbol{C} + \boldsymbol{S} \cdot \boldsymbol{sign}(\boldsymbol{F} - \boldsymbol{F_{p1}}) \cdot (\boldsymbol{C} - \boldsymbol{C_{p1}}) \\ &\quad + \boldsymbol{T} \cdot \boldsymbol{sign}(\boldsymbol{F} - \boldsymbol{F_{p2}}) \cdot (\boldsymbol{C} - \boldsymbol{C_{p2}}) \end{aligned} \tag{1.10}$$

where S and T are random numbers between 0 and 1. Since this does not require any evaluation of gradients, additional computational requirement is negligibly small.

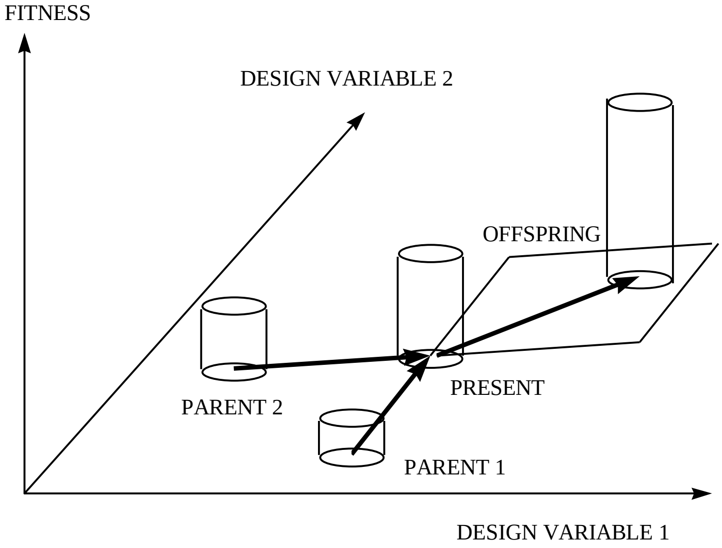

Fig. 1.7 Illustration of evolutionary direction operator

1.3.3.5 Mutation

The function of mutation is to keep diversity of a population and promote the searching in the solution space that cannot be represented by the strings of the present population. One of the most common mutation methods in real-coded EAs is uniform mutation that adds a uniform random number to each component of an individual's vector at a probability of p_c. Another mutation method is to Gaussian mutation that mutates each component of an individual's vector by adding a number from a Gaussian distribution with a mean of zero. This reflects populations in nature, where gross characteristics such as weight or height generally follow a normal distribution.

1.3.3.6 Elitist Strategy

Since evolution in EAs depends on stochastic operators, EAs do not guarantee a monotonic improvement in objective function value of the design unless deterministic overlapping systems are used. To ensure a monotonic improvement, De Jong proposed so-called elitist strategy [53], in which some of the best individuals are copied into the next generation without applying any evolutionary operators. EAs incorporating non-overlapping system usually adopt this strategy.

1.3.4 Parallelization

In biological evolution, every individual in a generation undergoes a series of evolution processes such as evaluation, selection and mating in parallel. Therefore, EAs that mimic biological evolution to solve design problems are intrinsically parallel search methods and consequently EAs can be easily parallelized. When an EA is implemented on a parallel machine, there is a choice between:

a) Preserving the global population while parallelising fitness evaluation.
b) Dividing the global population into several sub-populations and evolving a sub-population on each processor.
c) Preserving the global population while parallelising the EA operators that are restricted on the neighboring individuals.

The methods a) are categorized as global parallel evolutionary algorithms. A commonly-used approach is a master-slave implementation where each individual's fitness is evaluated simultaneously on a different processor (slave processor) and the master collects the results and applies the evolutionary operators to produce the next generations. An asset in this approach is that time saving can be easily achieved by distributing the computational effort without any change to the standard EAs since an individual's fitness evaluation process is independent of that of the other. Moreover, parallel efficiency can be very high when the fitness evaluation consumes most of CPU time.

The methods b) categorized as island distributed evolutionary algorithms [54] are positive parallel evolutionary algorithms. In these methods, a population is divided into sub-populations with some restrictions imposed on the mixing of these semi-isolated subpopulations. Some studies have reported that these methods can be more robust than the standard EAs since different subpopulations will tend to explore different portions of the search space (see for instance [55,56]) Some EA practitioners implement these methods even on serial machines. It was also reported that, however, these methods do not always produce better results [57].

The third methods c) categorized as cellular evolutionary algorithms are the extension of b). The mixing in the population is strictly restricted to the neighbors and thus quick domination in a population by a super individual is prevented.

1.4 Multiobjective Evolutionary Algorithms

1.4.1 Introduction

Many real-world design problems require simultaneous optimization of multiple and often competing objectives. Such problems are called multiobjective problems (MOPs). Although single-objective optimization problems may have a unique optimal solution, MOPs often present a set of compromised solutions, largely known as the tradeoff surface, *Pareto-optimal* solutions or *non-*

dominated solutions. These solutions are optimal in the sense that no other solutions in the search space are superior to them when all objectives are considered. In general, the goal of MOPs is to find as many Pareto-optimal solutions as possible. Once such solutions reveal tradeoff information among different objectives, the higher-level decision-maker will be able to choose a final design with further considerations.

Consider, for instance, the car design. There are a wide variety of demands from consumers such as fuel efficiency, maximum speed and cost. A perfect design might be a car that maximizes fuel efficiency and maximum speed while minimizing cost. However, these goals are conflicting in general: lower vehicle costs may not result in the desired maximum speed, an alternative car of high maximum speed may end in an expensive gas-guzzler. Therefore, the higher-level decision-maker selects one of such Pareto-optimal designs with other considerations: which profits our company the best?

According to the above discussion, the primary goal of MOP optimizations is, unlike that of single objective optimizations, to find various Pareto-optimal solutions to show the precise tradeoff information among the competing objectives. In this sense, EAs seem to be particularly suited for MOP optimizations because they can uniformly sample many Pareto-optimal solutions in parallel via its population of solutions. EAs developed for MOPs are called Multiobjective Evolutionary Algorithms (MOEAs).

Schaffer is usually recognized as a pioneer MOEA researcher, who first perceived an MOEA's population capability of finding Pareto-solutions within a single optimization run [58,59]. In his "Vector Evaluated Genetic Algorithm (VEGA)," appropriate fractions of the next generation are sequentially selected from the whole of the present generation based on their performance in each of the objectives. In a two-objective problem, for instance, VEGA would select half of the next generation's population using one of the objectives and the other half using another objective. These sub-populations were then shuffled together and evolutionary operators such as crossover and mutation are applied as usual. Although Schaffer reported some success, VEGA tends to find only extreme regions on the Pareto front since it choose specialists in each objective ignoring solutions performing "acceptably" in some of the objectives.

A breakthrough in MOEA research was given by Goldberg [34]. In his literature, he suggested to use the concept of *Pareto optimality* though selection and ranking methods. This offers capability of sampling uniform Pareto solutions to Evolutionary approaches. Thereafter, many MOEAs have been developed based on this concept [60-62] and most of the MOEA practitioners, at present, use such Pareto-based MOEAs.

1.4.2 Pareto Optimality

In real-world MOPs that usually involve conflicting objectives, there is no unique optimum, but rather a set of compromised solutions known as Pareto optimal solutions or non-dominated solutions [63]. These solutions distribute on the edge of the feasible region, showing the tradeoff information between the conflicting objectives. Figure 1.8 shows an example of MOPs, which is to minimize two conflicting objectives. In this problem, there is no single perfect solution that minimizes both f_1 and f_2. Instead, there are innumerable compromised optimal solutions such as solutions A, B, and C. In optimization terminology, all these solutions are Pareto optimal because there is no better solution on both objectives. On the other hand, one can say that the solution D is inferior to B because it has larger values than B in both objectives. The final goal in a MOP is to find such Pareto optimal solutions as many as possible to represent tradeoff information among different objectives.

The terms "dominance" and "Pareto optimality" can be mathematically defined for a general problem of simultaneously minimizing an n-components vector function ($\mathbf{f}= f_1(\mathbf{x}),\ldots,f_n(\mathbf{x})$) of an m-dimensional decision variable vector ($\mathbf{x}=x_1,\ldots,x_m$) from some universe Ω.

Definition 1 (Dominance): *A vector* $\mathbf{u} = (u_1,\ldots,u_m)$ *is said to dominate* $\mathbf{v} = (v_1,\ldots,v_m)$ *if and only if* $\mathbf{u}$ *is partially less than* $\mathbf{v}$, *i.e.,* $\forall i \in \{1,\ldots,m\}, u_i \le v_i \wedge \exists i \in \{1,\ldots,m\} : u_i < v_i$

Definition 2 (Pareto optimality): *A solution* $\mathbf{x} \in \Omega$ *is said to be Pareto-optimal with respect to* Ω *if and only if there is no* $\mathbf{x}' \in \Omega$ *for which* $\mathbf{v}=\mathbf{f}(\mathbf{x}')=(f_1(\mathbf{x}'),\ldots,f_n(\mathbf{x}'))$ *dominates* $\mathbf{u}=\mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),\ldots,f_n(\mathbf{x}))$.

1.4.3 Pareto-Based MOEAs

In this subsection, a few salient Pareto-based MOEAs are presented.

1.4.3.1 Multiobjective Genetic Algorithm (MOGA)

Fonseca and Fleming proposed multiobjective GA [60] that uses non-dominated sorting and ranking selection method. In MOGA, an individual's rank corresponds to the number of individuals in the current population by which it is dominated. Non-dominated individuals are, therefore, all assigned the same rank, while dominated ones are penalized according to how concentrated the population is in the corresponding region of the tradeoff surface. This approach is also easier to formulate, and to analyze, mathematically. This ranking procedure for a minimization problem is illustrated in Fig. 1.9.

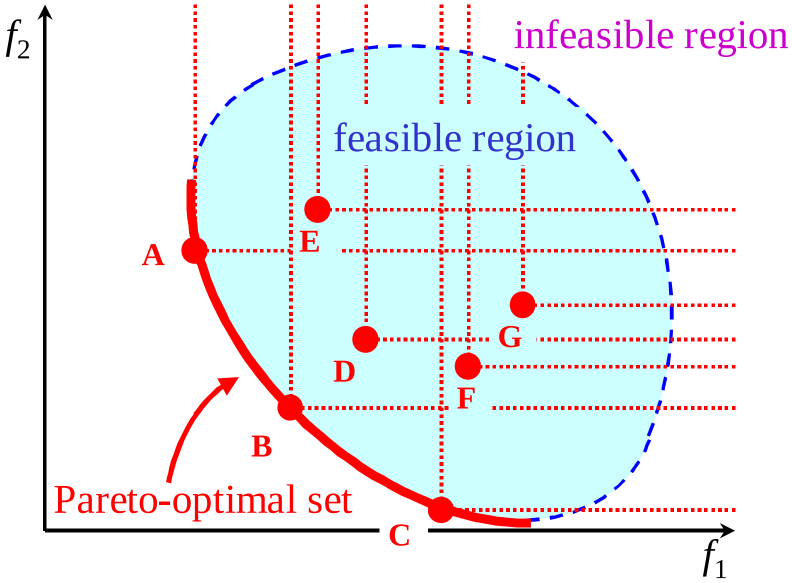

Fig. 1.8 The concept of Pareto optimality

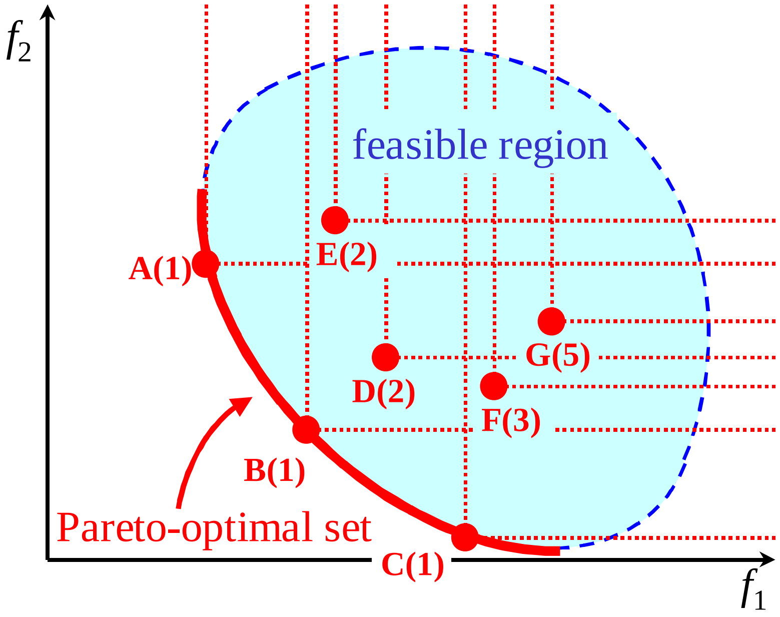

Fig. 1.9 Fonseca’s ranking method for a minimization problem

1.4.3.2 Niched Pareto Genetic Algorithm (NPGA)

Niched Pareto Genetic Algorithm (NPGA) holds a type of binary tournament selection called a Pareto domination tournament [61]. In this tournament, the two strings are compared to a sample of the population and the number of dominating points in the sample is counted for both individuals. The individual with the least number of dominating points survives. The size of the sample used can be varied depending on the level of selection pressure required. Obviously larger size samples will increase the bias towards stronger solutions. This method is analogous to performing Fonseca’s ranking method locally.

1.4.3.3 Non-Dominated Sorting Genetic Algorithm (NSGA)

Non-dominated sorting Genetic Algorithm (NSGA) uses Goldberg’s non-dominated sorting procedure [62]. The idea behind NSGA is to assign equal probability of reproduction to all non-dominated individuals in the population. This method involves first finding all of the Pareto optimal points within a population, giving them a rank of one and removing them. The remaining population members are again processed to find non-dominated individuals and these are given rank two and removed and so on until all of the population has been ranked. This ranking method is illustrated in Fig. 1.10.

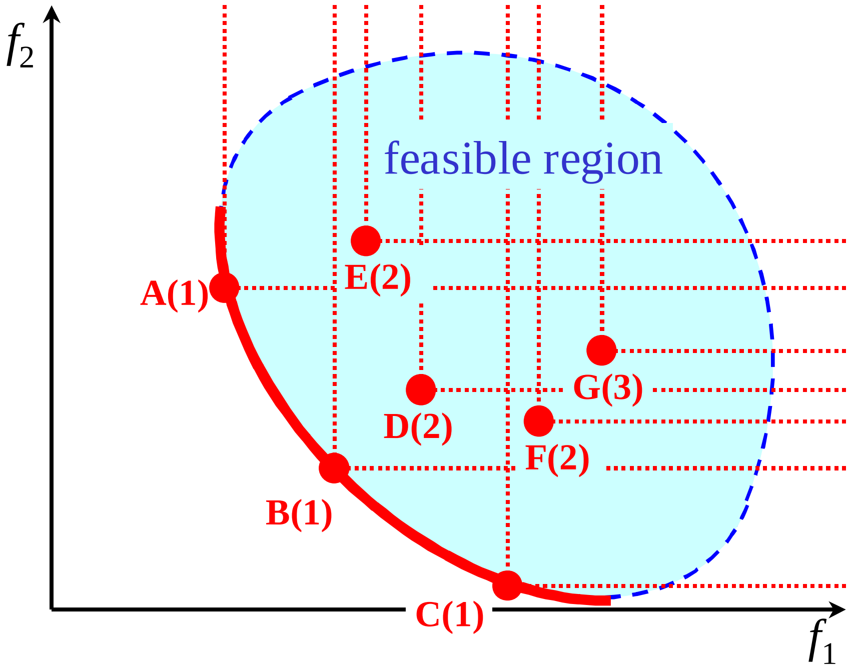

Fig. 1.10 Goldberg’s ranking for a minimization problem

1.4.4 Niching

In MOEAs, maintenance of diversity in the population is important to sample uniform Pareto-optimal solutions. To preserve this diversity, Goldberg’s *fitness sharing* [34] is used most frequently, which is a fitness scaling mechanism based on the idea that individuals in a particular niche have to share the available resources.

For a minimization problem, *i*-th individual’s shared fitness function $\boldsymbol{F}'(\boldsymbol{i})$ is equal to its prior fitness $\boldsymbol{F}(\boldsymbol{i})$ multiplied by its *niche count*, which is the sum of sharing function (*sh*) values between itself and each individual in the population:

$$\boldsymbol{F}'(\boldsymbol{i}) = \boldsymbol{F}(\boldsymbol{i}) \cdot \sum_{\boldsymbol{i}=1}^{\boldsymbol{N}} \boldsymbol{sh}(\boldsymbol{d}_{\boldsymbol{ij}}) \tag{1.11}$$

where N is a population size. The most commonly-used sharing function is given by the following equation:

$$sh(d_{ij}) = \left(\begin{array}{ll} 1 - \left(\dfrac{d_{ij}}{s_{share}} \right)^{a} & d_{ij} < s_{share} \\ 0 & others \end{array} \right. \tag{1.12}$$

where s_{share} is the niche size. If the distance between two individuals is greater than or equal to s_{share}, they do not affect each other's shared fitness.

The distance can be measured with respect to a metric in either phenotypic or genotypic space. A genotypic sharing measures the interchromosomal Hamming distance. A phenotypic sharing can further be classified into two types. One measures the distance between the design variables. The other, on the other hand, measures the distance between the objective function values. In MOEAs, a phenotypic sharing is usually preferred to distribute the population over the Pareto-optimal region. Both MOGA and NPGA perform sharing within objective function space. NSGA, on the other hand, performs sharing by measuring the vector distance between the decoded design variables, emphasizing maintenance of diversity in the parameter set.

The choice of s_{share} has a significant impact on the performance of MOEAs. Fonseca and Fleming [60] gave a simple estimation of s_{share} in the objective function space as

$$N s_{share}^{q-1} - \frac{\prod_{i=1}^{q} (M_i - m_i + s_{share}) - \prod_{i=1}^{q} (M_i - m_i)}{s_{share}} = 0 \tag{1.13}$$

where q is a dimension of the objective vector, and M_i and m_i are maximum and minimum values of each objective, respectively. When this formula is applied at every generation, the resulting s_{share} is adaptive to the population during the evolution process. Niche counts can be consistently incorporated into the fitness assignment according to rank by using the to scale individual fitness within each rank.

1.5 Documentation Organization

The remainder of this document is organized as follows. Chapter 2 proposes a new EA, named "real-coded Adaptive Range Genetic Algorithms (real-coded ARGAs)." To display advantages of the real-coded ARGAs, they will be first applied to a test function optimization problem. Then, an aerodynamic airfoil shape optimization is demonstrated to ensure the feasibility of the proposed approach in aerodynamic design problems. In Chapter 3, a tree structure of design parameters inspired by the data structure of Genetic Programming is introduced as a coding structure. The coding structure is determined by the epistasis analysis using experimental design in advance. First, feasibility of typical airfoil shape parameterization techniques will be examined though reproduction of a NASA supercritical airfoil and an aerodynamic airfoil shape optimization. Then, the present approach is applied to aerodynamic design using MOEAs coupled with a potential flow solver. Chapter 4 provides the aerodynamic optimization results of a transonic wing design by using the proposed EA. CFD analyses are performed using a three-dimensional Navier-Stokes code. In Chapter 5, the aerodynamic optimization of a supersonic wing design will be demonstrated. Chapter 6 concludes this research and discusses directions for future research.

References

[1] Obayashi, S., “Inverse Optimization Method for Aerodynamic Shape Design,” *Recent Development of Aerodynamic Design Methodologies –Inverse Design and Optimization –*, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, Germany, 1999, pp.25-54.

[2] Hicks, R. M., Murman, E. M. and Vanderplaats, G. N., “An Assessment of Airfoil Design by Numerical Optimization,” NASA TM X-3092, Ames Research Center, Moffett Field, California, July 1974.

[3] Hicks, R. M. and Henne, P. A., “Wing Design by Numerical Optimization,” *Journal of Aircraft*, Vol. 15, 1978, pp.407-412.

[4] Baysal, O. and Eleshaky, M. E., “Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics,” *AIAA Journal*, Vol. 30, No. 3, 1992, pp. 718-725.

[5] Reuther J. J. and Jameson, A., “Supersonic wing and wing-body shape optimization using an adjoint formulation,” Technical report, The Forum on CFD for Design and Optimization, (IMECE95), San Francisco, California, November 1995.

[6] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J. and Saunders, D., “Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1,” *Journal of Aircraft*, Vol.36, No. 1, January-February, 1999, pp.51-60.

[7] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J. and Saunders, D., “Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2,” *Journal of Aircraft*, Vol.36, No. 2, January-February, 1999, pp.61-74.

[8] Press, W. H., Teukolsky, S.A., Vetterling, W. T. and Flannery, B., *Numerical Recipes in Fortran 77: The Art of Scientific Computing,* second edition, the Press syndicate of the University of Cambridge, New York, 1996.

[9] Lions, J. L., *Optimal Control of Systems Governed by Partial Differential Equations,* Springer-Verlag, New York, 1971, Translated by Mitter, S.K.

[10] Miettinen, K., Makela, M. M., Neittaanmaki, P. and Periaux, J. (Eds.), *Evolutionary Algorithms in Engineering and Computer Science,* John Willey & Sons Ltd, Chichester, U.K., 1999, Chaps.17-24.

[11] Bramlette, M. F. and Cusic, R., “A Comparative Evaluation of Search Methods Applied to the Parametric Design of Aircraft,” *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989, pp.213-218.

[12] Parmee, I. C. and Watson, A. H., “Preliminary Airframe Design Using Co-Evolutionary Multiobjective Genetic Algorithms,” *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 2, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1999, pp.1657-1671.

[13] Powell, D. J., Tong, S. S. and Sholbick, M. M., “EnGENEous Domain Independent, Machine Learning for Design Optimization,” *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989, pp.151-159.

[14] Quagliarella, D. and Cioppa, A. D., “Genetic Algorithms applied to the Aerodynamic Design of Transonic Airfoils,” AIAA-94-1896-CP, June 1994.

[15] Yamamoto, K. and Inoue, O., “Applications of Genetic Algorithm to Aerodynamic Shape Optimization,” AIAA Paper 95-1650-CP, A collection of technical papers, 12th AIAA Computational Fluid Dynamics Conference, CP956, San Diego, CA, June 1995, pp. 43-51.

[16] Cao, H. V. and Blom, G. A., “Navier-Stokes/Genetic Optimization of Multi-Element Airfoils,” AIAA 96-2487, June 1996.

[17] Obayashi, S. and Oyama, A., “Three-Dimensional Aerodynamic Optimization with Genetic Algorithms,” *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, John Wiley & Sons, Ltd, Chichester, U.K., 1996, pp.420-424.

[18] Darwin. C., *On the Origin of Species by Means of Natural Selection*, Murray, London, 1859.

[19] Fogel, L. J., Owens, A. J. and Walsh, M.J., *Artificial Intelligence Thorough Simulated Evolution,* John Wiley & Sons, Ltd, Chichester, U.K., 1966.

[20] Fogel, D. B., *Evolutionary Computations: Toward a New Philosophy of Machine Intelligence,* IEEE Press, New York, 1995.

[21] Fogel, D. B. and Atmar, J. W., “Comparing genetic operators with Gaussian mutation in

simulated evolutionary processes using linear systems," *Biological Cybernetics*, Vol.63, 1990, pp.111-114.

[22] Fogel, D. B. and Stayton, L. C., "On the effectiveness of crossover in simulated evolutionary optimization," *BioSystems*, Vol. 32, No. 3., 1994, pp171-182.

[23] Sebald, A. and Schlenzing, J., "Minimax Design of Neural Net Controllers for Highly Uncertain Plants," *IEEE Transactions of Neural Networks*, Vol. 5, No. 1., pp.73-82.

[24] Back, T., Rudolph, G. and Schwefel, H. -P., "Evolutionary programming and evolution strategies: Similarities and differences," *Proceedings of the Second Annual Conference on Evolutionary Programming*, Evolutionary Programming Society, San Diego, CA, 1993, pp.11-22.

[25] Rechenberg, I., "Cybernetic Solution Path of an Experimental Problem," Ministry of Aviation, Royal Aircraft Establishment, U.K., 1965.

[26] Rechenberg, I., *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution,* Frommann-Holzboog Verlag, Stuttgart, Germany, 1973.

[27] Schwefel, H. -P., *Numerical Optimization for Computer Models*, John Willey, Chichester, U.K., 1981.

[28] Schwefel, H. -P., *Evolution and Optimum seeking*, Sixth-Generation Computer Technology Series, Wiley, New York, 1995.

[29] Heusener, G., "Optimierung natriumgekuhlter scheneller Brutreaktoren mit Methoden der nichtlinearen Programmierung," report KFK-1238, Nuclear Research Center (KfK) Karlsruhe, Germany, July 1970.

[30] Hartmann, D., "Optimierung balkenartiger Zylinderschalen aus Stahlbeton mit elastischem und plastischem Werkstoffverhalten," Doctoral Dissertation, University of Dortmund, 1974.

[31] Brudermann, U., "Entwicklung und Anpassung eines vollstandigen Ansteuersystems fur fremdenergetisch angetriebene Ganzarmprothesen," *Fortschrittberichte der VDI-Zeitschriften*, Vol.17 (Biotechnik), No. 6, December. 1977.

[32] Markwich, P., "Der thermische Wasserstrahlantrieb auf der Grundlage des offenen Clausius-Rankine-Prozesses - Konzeption und hydrothermodynamische Analyse," Doctoral Dissertation, Technical University of Berlin, 1978.

[33] Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[34] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[35] Minga, A. K., "Genetic algorithms in aerospace design," the AIAA Southeastern Regional Student Conference, Huntsville, AL, 1986.

[36] Glover, D. E., "Experimentation with an adaptive search strategy for solving a key-board design/configuration problem," Doctoral Dissertation, University of Iowa, 1986.

[37] Koza, J., *Genetic Programming: On the programming of computers by means of natural selection*, Bradford Books, Cambridge, MA, 1992.

[38] Janikow, C. Z. and Michalewicz, Z., "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms," *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, pp.31-36.

[39] Wright, A. H., "Genetic Algorithms for Real Parameter Optimization," *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1991, pp.205-218.

[40] Eshelman, L. J., "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, pp.265-283.

[41] Tsutsui, S. and Fujimoto, Y., "Forking Genetic Algorithms with blocking and shrinking modes (fGA)," P*roceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993, pp.206-213.

[42] De Jong, K. A., "An analysis of the behavior of a class of genetic adaptive systems," Doctoral Dissertation, University of Michigan, 1975.

[43] Goldberg, D. E. and Richardson, J., "Genetic algorithms with sharing for multimodal function optimization," *Proceedings of the Second International Conference on Genetic Algorithms*, Morgan

Kaufmann Publishers, Inc., San Mateo, CA, 1987, pp.41-49.
[44] Baker, J. E., "Reducing Bias and Inefficiency in the Selection Algorithm," *Proceedings of the Second International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1987, pp.14-21.
[45] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs,* third revised edition, Springer-Verlag, Berlin, 1996.
[46] Goldberg, D. E. and Deb, K., "A comparative analysis of selection schemes used in genetic algorithms" *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, pp.69-93.
[47] Eshelman, L. J. and Schaffer, J. D., "Real-coded genetic algorithms and interval schemata," *Foundations of Genetic Algorithms.2*," Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993, pp.187-202.
[48] Radcliffe, N. J., "Genetic Neural Networks on MIMD Computers, Doctoral Dissertation, University of Edinburgh, Edinburgh, U.K., 1990.
[49] Booker, L. B., "Improving Search in Genetic Algorithms," *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1987, pp.61-73.
[50] Deb, K. and Goyal, M., "A robust optimization procedure for mechanical component design based on genetic adaptive search," *Transactions of the ASME: Journal of Mechanical Design*, Vol. 120, No. 2, 1998, pp.162-164.
[51] Deb, K. and Beyer, H. -G., "Self-Adaptation in Real-Parameter Genetic Algorithms with Simulated Binary Crossover," *Proceedings of the Genetic and Evolutionary Computation Conference,* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1999, pp.172-179.
[52] Yamamoto, K. and Inoue, O., "New Evolutionary Direction Operator for Genetic Algorithm," *AIAA Journal*, Vol. 33, No. 10, Oct. 1995, pp. 1990-1992.
[53] De Jong, K. A., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Doctoral Dissertation, University of Michigan, Ann Arbor, 1975.
[54] Cohoon, J. P., Hedge, S. U., Martin, W. N. and Richards, D., "Punctuated equilibria: A parallel genetic algorithm," *Proceedings of the Second International Conference on Genetic Algorithms,* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1987, pp.148.
[55] Lin, S. C., Punch, W. F. and Goodman, E. D., "Coarse-grain parallel genetic algorithms: Categorization and a new approach," Sixth IEEE SPDP, 1994, pp.28-37.
[56] Loraschi, A., Tettamanzi, A, Tomassini, M and Verda, P., "Distributed genetic algorithms with an application to portfolio selection problems," *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Springer-Verlag Wien, New York, 1995, pp.384-387.
[57] Belding, T. C., "The Distributed Genetic Algorithm Revisited," *Proceedings of the Sixth International Conference on Genetic Algorithms,* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1995, p.114-121.
[58] Schaffer, J. D., "Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms," Doctoral Dissertation, Vanderbilt University, Nashville, TN, 1984.
[59] Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications,* Lawrence Erlbaum Associates, Publishers, Hillsdate, NJ, 1985, pp.93-100.
[60] Fonseca, C. M., "Multiobjective Genetic Algorithms with Application to Control Engineering Problems," Doctoral Dissertation, University of Sheffield, 1995.
[61] Horn, J. and Nicholas N., "Multiobjective Optimization Using the Niched Pareto Genetic Algorithm," Technical Report 930005, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Matthews Avenue, Urbana, IL61801-2996, Illinois Genetic Algorithms Laboratory (IlliGAL), July 1993.
[62] Srinivas, N. and Deb., K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, Vol.2, No.3, 1994, pp.221-248.
[63] Stewart, T. J., "A critical Survey on the Status of Multiple Criteria Decision Making and Practice," *International Journal of Management Science*, Vol.20, No5/6, 1992, pp.569-586.