Spring 2010

Hybrid Intelligent Optimization Methods for Engineering Problems

Yasin Volkan Pehlivanoglu
Old Dominion University

HYBRID INTELLIGENT OPTIMIZATION METHODS FOR ENGINEERING PROBLEMS

by

Yasin Volkan Pehlivanoglu
B.S. June 1993, Istanbul Technical University
M.S. May 2006, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

AEROSPACE ENGINEERING

OLD DOMINION UNIVERSITY
May 2010

Approved by:

Oktay Baysal (Director)

Ali Beskok (Member)

Gene Hou (Member)

Abdurrahman Hacioglu (Member)

ABSTRACT

HYBRID INTELLIGENT OPTIMIZATION METHODS FOR ENGINEERING PROBLEMS

Y. Volkan Pehlivanoglu
Old Dominion University, 2010
Director: Dr. Oktay Baysal

The purpose of optimization is to obtain the best solution under certain conditions. There are numerous optimization methods because different problems need different solution methodologies; therefore, it is difficult to construct patterns. Also mathematical modeling of a natural phenomenon is almost based on differentials. Differential equations are constructed with relative increments among the factors related to yield. Therefore, the gradients of these increments are essential to search the yield space. However, the landscape of yield is not a simple one and mostly multi-modal. Another issue is differentiability. Engineering design problems are usually nonlinear and they sometimes exhibit discontinuous derivatives for the objective and constraint functions. Due to these difficulties, non-gradient-based algorithms have become more popular in recent decades.

Genetic algorithms (GA) and particle swarm optimization (PSO) algorithms are popular, non-gradient based algorithms. Both are population-based search algorithms and have multiple points for initiation. A significant difference from a gradient-based method is the nature of the search methodologies. For example, randomness is essential for the search in GA or PSO. Hence, they are also called stochastic optimization methods. These algorithms are simple, robust, and have high fidelity. However, they suffer from similar defects, such as, premature convergence, less accuracy, or large computational time. The premature convergence is sometimes inevitable due to the lack of diversity. As the generations of particles or individuals in the population evolve, they may lose their diversity and become similar to each other. To overcome this issue, we studied the diversity concept in GA and PSO algorithms.

Diversity is essential for a healthy search, and mutations are the basic operators to provide the necessary variety within a population. After having a close scrutiny of the diversity concept based on qualification and quantification studies, we improved new mutation strategies and operators to provide beneficial diversity within the population. We called this new approach as multi-frequency vibrational GA or PSO. They were applied to different aeronautical engineering problems in order to study the efficiency of these new approaches. These implementations were: applications to selected benchmark test functions, inverse design of two-dimensional (2D) airfoil in subsonic flow, optimization of 2D airfoil in transonic flow, path planning problems of autonomous unmanned aerial vehicle (UAV) over a 3D terrain environment, 3D radar cross section minimization problem for a 3D air vehicle, and active flow control over a 2D airfoil.

As demonstrated by these test cases, we observed that new algorithms outperform the current popular algorithms. The principal role of this multi-frequency approach was to determine which individuals or particles should be mutated, when they should be mutated, and which ones should be merged into the population. The new mutation operators, when combined with a mutation strategy and an artificial intelligent method, such as, neural networks or fuzzy logic process, they provided local and global diversities during the reproduction phases of the generations. Additionally, the new approach also introduced random and controlled diversity. Due to still being population-based techniques, these methods were as robust as the plain GA or PSO algorithms. Based on the results obtained, it was concluded that the variants of the present multi-frequency vibrational GA and PSO were efficient algorithms, since they successfully avoided all local optima within relatively short optimization cycles.

ACKNOWLEDGEMENTS

There are many people who have contributed to the successful completion of this dissertation. The author sincerely thanks Dr. Oktay Baysal for his guidance and support as my academic adviser. This work was funded by Turkish Air Force; therefore, I also thank the Turkish Armed Forces for providing an opportunity to perform this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Optimization Concept and Basics

Optimization means "an act, process, or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible." From the same family, the word *optimum* means "the amount or degree of something that is most favorable to some end; especially: the most favorable condition for the growth and reproduction of an organism: greatest degree attained or attainable under implied or specified conditions" [1]. The word *optimum*, meaning *best*, is synonymous with *most* or *maximum* in the former case and with *least* or *minimum* in the latter.

Mathematically, the general standard optimization problem can be defined as

$$\min_{x \in R^n, u(x) \in R^m} f(x, u(x)) \tag{1.1}$$

subject to;

$$\begin{aligned} &h(x, u(x)) = 0 \\ &g(x, u(x)) \le 0 \\ &x_L \le x \le x_U \\ &G(x, u(x)) = 0 \end{aligned} \tag{1.2}$$

where f to be minimized is a vector function representing the objective function(s), x is the column vector defining the design variables from a set of X, $u(x)$ is a column vector including the state equations. This expression is solved by subjecting to vector function h as equality constraints, vector function g as inequality constraints, upper and lower bounds of x, and vector function G as another equality constraint group including state equations. Briefly this standard form includes three elements such as objective function, design variables, and constraints. The aim of the optimization process is to find the proper design variable values which yield the maximum or the minimum objective function value under constrained conditions.

1.2 Solution Methodologies

The general search methodologies to find the optimum solution are gradient-based and non-gradient based algorithms. Fig. 1.1 depicts the solution strategies. Gradient-based algorithms search the next candidate solution within the design space in accordance with the derivative information. Non-gradient based algorithms are linear programming techniques

and heuristic search methods. Heuristic optimization algorithms use the information currently gathered by the algorithm to help decide which solution candidate should be tested next or how the next solution candidate can be produced. There are a lot of heuristic search methods in the literature.

The most popular ones are genetic algorithms (GA), particle swarm optimization (PSO), simulated annealing (SA), and ant colony optimization (ACO). These search methods are usually based on a model of some natural phenomenon or physical process. These algorithms are also called artificial intelligence. The other popular types of artificial intelligence methods are fuzzy logic (FL) and neural networks (NN).

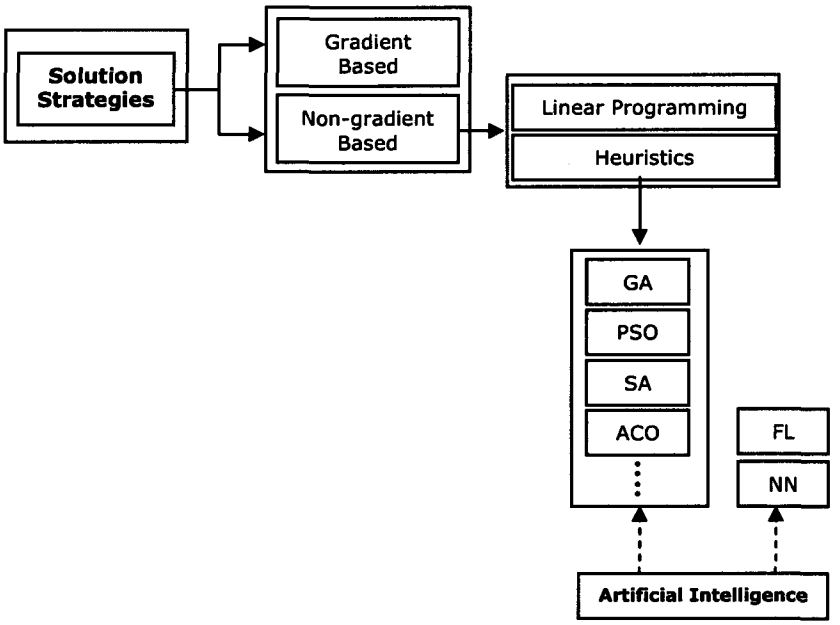

Figure 1.1 The solution strategies

Gradient-based algorithms trade in guaranteed convergence of the solution for a relatively shorter runtime. This does not mean that the results obtained using them are global, they may probably be the local optima. Often heuristic algorithms may require more cost function evaluations than comparable, gradient-based algorithms. They, however, provide attractive characteristics, such as ease of implementation for both continuous and discrete problems, efficient use of large numbers of parallel processors, no requirement for the continuity in response functions, and more robust solution generations for searching global or near global solutions. Therefore, heuristic search methods are preferred to study. On the other hand, a solution a little bit inferior to the best possible one is better than one which needs

several years to be found. So, heuristic algorithms need to be adjusted to be more efficient than the current situations.

Although they are called artificial intelligence techniques, their long computational times requirement probably originates from their basic, but essential nature: randomness and chance. However, the other artificial intelligence techniques such as fuzzy logic or neural networks do not have random nature, and they are almost deterministic methods. So, the use of deterministic and stochastic algorithms in a hybrid manner within robust algorithm architecture may be promising to solve optimization problems in a shorter time.

1.3 Common Problems in Optimization Process

There are some important complications, the major problems that may be encountered during the optimization solution process. Some of the subjects are in general and some of them are observed especially on nature-inspired approaches like genetic algorithms. Neglecting even a single one among them during the design or process of optimization can render the whole efforts invested useless, even if highly efficient optimization techniques are applied. By giving clear definitions to some of these topics, we want to raise the focused areas that are studied in this dissertation. The most countered complications are premature convergence mostly originating from loss of diversity, ruggedness and weak causality, deceptiveness, neutrality and redundancy, epistasis, noise and robustness, overfitting and oversimplification, and dynamically changing fitness. Most of them are related to objective function (fitness) landscape. The detailed information can be found in [2].

Among these complications probably the most popular problem is premature convergence. An optimization process comes to end if it cannot generate new solution candidates anymore or if it continually produces solution candidates from a small subset of the problem space. One of the problems in global optimization is that it is too difficult to determine whether the best solution currently known is a local or global optimum. This actually becomes only problematic if there are multiple optima which mean the problem is multimodal. An optimization algorithm prematurely converges to a local optimum if it is no longer able to explore other sections of the search space than the currently examined area and there at least one another region exists that contains a solution superior to the currently discovered one. The main reason for the premature convergence might be the lack of diversity within the searched space. Much of the efficiency in natural algorithms is based on the huge diversity of candidates interacting in manifold ways. In population-based global optimization algorithms, starting with diverse individuals and maintaining a set of diverse solution

candidates are very important. The lack of diversity means closing by a state where all the solution candidates under investigation are similar to each other. Preserving diversity is directly linked with maintaining a good balance between exploitation and exploration and has been studied by researchers from many aspects.

The operations which generate new solutions from existing ones have a very large impact on the diversity in the population. In the context of optimization, exploration means global character finding new points in areas of the search space which have not been investigated before. Exploration is a metaphor for search operations which find totally new and maybe better solution candidates. Exploitation, on the other hand, has a local character and it is the process of improving and combining the traits of the currently known solution(s). Exploitation operations often incorporate small changes which lead to new individuals very close to the already tested solution candidates.

Optimization algorithms that favor exploitation over exploration may have higher convergence speed but run the risk of not finding the optimal solution and may get stuck at a local optimum. However, those algorithms which perform excessive exploration may never improve their solution candidates far enough to find the global optimum or it may take them very long time to discover it accidentally.

1.4 Basic GA Structure

As a stochastic method, GA is an emergent optimization algorithm mimicking of the natural evolution, where a biological population evolves over generations to adapt to an environment by selection, recombination, and mutation [3]. When GA is applied to optimization problems, fitness, individual, and genes usually correspond to an objective function value, a design candidate, and design variables, respectively.

The brief algorithm can be expressed as the following

- Determine fitness function, f
- Determine stopping criteria
- Determine population size, P
- Determine fitness scaling method
- Determine selection mechanism
- Determine crossover fraction and method
- Determine mutation ratio and method
- Determine elite count
- Initialize the population

Repeat, g=1

Evaluation - compute fitness values

Check the convergence criterion

Fitness assignment - scale fitness's

Selection - select parents

Apply crossover

Apply mutation

g=g+1

- ⊾End

Basic genetic algorithm cycle is depicted in Fig. 1.2

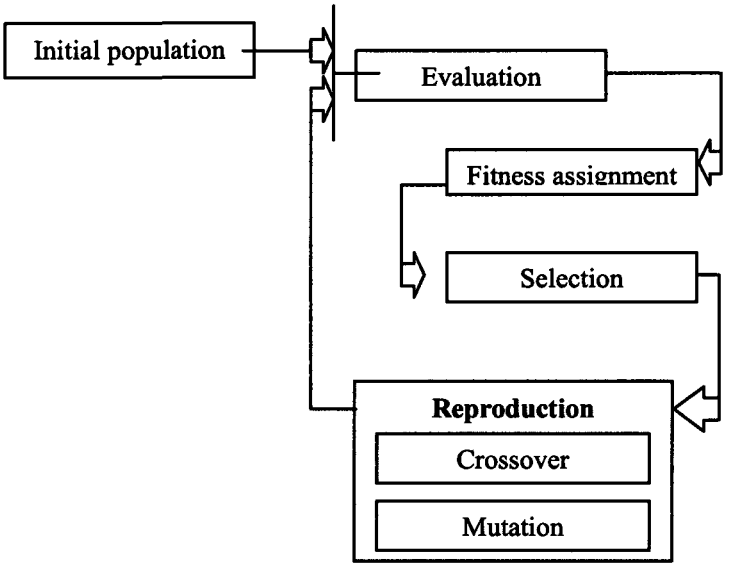

Figure 1.2 Basic genetic algorithm cycle.

At the beginning of the optimization, GA needs a group of probable solutions named as initial population. It is possible to mention about two ways of generating the initial population. The first way consists of using randomly produced solutions formed by a random number generator. This method is especially preferable for the problems about which no a priori knowledge exists or for assessing the performance of an algorithm. The second method employs a priori knowledge about the given optimization problem. In this way, GA starts the optimization with a set of approximately known solutions and therefore converges to an optimal solution in less time than with the previous method [4].

The principle behind genetic algorithms is essentially natural selection. Selection provides the driving force in a genetic algorithm. With too much force, genetic search will terminate prematurely; with too little force, evolutionary progress will be slower than

necessary for adaptation. Up to now, many selection methods have been proposed, examined, and compared [5-7]. Common selection types are roulette wheel, tournament, uniform, stochastic uniform, and remainder selection methods. Before a selection step there is a scaling operation. In this phase the fitness values are scaled. Ranking, proportional (cost weighting), top, and shift linear are popular fitness scaling methodologies. We will not focus on selection methods, so detail information can be found in [8].

In genetic algorithms, accumulated information is explored by the selection mechanism, while new regions of the search space are explored by means of genetic operators such as crossover and mutation operators. We analyze them in detail in the methodology section.

1.5 Critique on GA and Remedies in Literature

Although genetic algorithms are robust and have high fidelity, they may sometimes be computationally expensive when each evaluation of the cost function requires intensive computation. However, there are different ways to accelerate and improve the performance of the genetic algorithms in literature. These are multi-level approaches [9], multi-processing [10], hybridization of genetic algorithms with other algorithms like gradient-based algorithm, neural network, or fuzzy algorithms [11-18], and the use of improved genetic operators.

In GAs, multi-processing is related to the evaluation of candidate solutions on a cluster of interconnected processors or parallelization of the evaluation software, using data partitioning or subdomaining in computational mechanics. In hybrid methods a stochastic algorithm such as GA is chosen due to its global features; and, in order to reduce its prohibitive simulation time while keeping its advantages, it is coupled with a deterministic algorithm such as gradient-based method. A Multi-level approach includes multi level evaluation, search, or parameterization. In multi level evaluation approach different evaluation software is assigned to each level such as panel solvers for exploration at the beginning stage of the optimization process and high-fidelity models such as Euler or Navier-Stokes solvers for refinement at the proceeding stages. Instead of using a real flow solver or optimization tool, the use of a surrogate model may be accepted as a multi-level approach. In multi-level search, each level is associated with a different search technique. Stochastic search techniques, such as GAs, are preferably used at the lower levels for the exploration of the design space, leaving the refinement of promising solutions to the gradient-based ones at the higher levels. In multi-level parameterization each level is associated with a different set of design variables. At the lowest or coarse level, a sub-problem with just a few design variables is solved. At the higher or fine levels, the number of design parameters is increased.

When facing constrained optimization problems, the term "different parameterization" may imply different handling of constraints.

Currently there are many genetic operators proposed in the GA literature. However, it is difficult to provide rigorous guidelines for which method should be used to which problem; and there are still opportunities to improve new operators. Crossover and mutation operators are essential to the genetic algorithms. As summarized here, crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. Mutation adds to the diversity of a population and one of the most important factors that determine the performance of the genetic algorithm is the diversity of the population. Most popular mutation operators are Gaussian [19], uniform operators [20]. The other types can be found in [21-23].

Apart from designing new mutation operators, researchers have focused less on investigating how to apply mutation operators during the process and determining what kind of diversity should be provided within the population. So, the key factor is diversity. After dealing with PSO, we focus on diversity concept and its implications.

1.6 Definition of Basic PSO

As in GA, Particle Swarm Optimization (PSO) method is a population-based stochastic optimization algorithm that originates from the "nature" and "evolutionary computations." PSO algorithms search the optimum within a population called "swarm." It benefits from two types of learning, such as "cognitive learning" based on individual's own history and "social learning" based on swarm's own history accumulated by sharing information among all particles in the swarm. Since its development in 1995 by Kennedy *et al.* [24], it has attracted significant attention. Most of the investigations on this topic are related to either the mathematical analysis focusing on how it works, or improving the PSO to get faster and more reliable solutions. The impetus for the latter is typically the trapping of the solution candidates to local optima, or the so-called premature convergence.

Let S be the swarm size, D be the particle dimension space, and each particle of the swarm has a current position vector $\boldsymbol{X}_i$, current velocity vector $\boldsymbol{V}_i$, individual best position vector $\boldsymbol{P}_i$ found by particle itself. The swarm also has the global best position vector $\boldsymbol{P}_g$ found by any particle during all prior iterations in the search space. Assuming that the function f is to be minimized and describing the following notations in t^{th} iteration, then the definitions are as follows:

$$X_i(t) = (x_{i,1}(t), x_{i,2}(t), ..., x_{i,D}(t),),\ x_{i,j}(t) \in R^D,\ ^{i=1,2,...,S} \quad (1.3)$$

$$V_i(t) = (v_{i,1}(t), v_{i,2}(t), ..., v_{i,D}(t),),\ v_{i,j}(t) \in R^D\ ,\ ^{i=1,2,...,S} \quad (1.4)$$

where each dimension of a particle in the swarm is updated using the following equations:

$$v_{i,j}(t) = v_{i,j}(t-1) + c_1 r_1 (P_i(t-1) - x_{i,j}(t-1)) + c_2 r_2 (P_g(t-1) - x_{i,j}(t-1)) \quad (1.5)$$

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t) \quad (1.6)$$

In Eq. (1.5), c_1 and c_2 denote constant coefficients, r_1 and r_2 are elements from random sequences in the range of (0, 1). The parameter c_1 controls the influence degree of "cognitive" part of an individual, and c_2 determines the effect of a "social" part of the swarm. The personal best position vector of each particle is computed using the following expression:

$$P_i(t) = \begin{Bmatrix} P_i(t-1), & if\ f(X_i(t)) \geq f(P_i(t-1)) \\ X_i(t), & if\ f(X_i(t)) < f(P_i(t-1)) \end{Bmatrix} \quad (1.7)$$

Then, the global best position vector is found by,

$$P_g(t) = \arg\min_{P_i(t)} f(P_i(t)) \quad (1.8)$$

1.7 Improved PSO Algorithms in Literature

PSO may have some issues related to convergence speed, prematurely converged solutions, deficient accuracy or lack of diversity. Although numerous modifications have been proposed so far to overcome these issues [25], we will highlight some improvement methods. These may be broadly divided into two categories as focusing on hardware or software improvements. The hardware focus is related to parallel computing. The software studies may be further classified as hybridization with other search algorithms, inspiration from other stochastic-based algorithms, rearranging and manipulating the reproduction descriptions mainly related to the velocity equation, parameter and neighborhood topology manipulations within the reproduction phase, and finally the population sizing and grouping.

Since each particle's cost function can be evaluated independently in the swarm, PSO is ideally suited for synchronized execution on a cluster of computers in parallel. Integrating PSO algorithms with other search algorithms is called hybridization. A common hybrid application is to use PSO with another population-based algorithm such as a genetic algorithm. Yet another technique is to use a gradient-based algorithm as an integrated part. Manipulating or rearranging Eq. (1.5) and Eq. (1.6) is an important part of algorithm improvements. In these manipulations the velocity equation is extended with an extra term or shortened depending on the approach. Parameter re-descriptions constitute another bundle of

modification packages. Since Shi *et al.* [26] suggested the use of an inertia weight *(w)* multiplying with the previous velocity in Eq. (1.5), the parameter number to be adjusted is increased to three. The inertia weight determines the effect of a previous velocity. The common PSO parameters, w, c_1, c_2 may be constant, or linearly or nonlinearly changeable, let's say periodically, adaptively, chaotic or randomly changeable depending on the time or other reference such as cost function value, velocity, metropolis criteria. Neighborhood topology originates from the selection of position vectors $\boldsymbol{P}_i$ and $\boldsymbol{P}_g$. In the original algorithm the particle is going to be attracted by individual best position and global best position. It means no other individual best position vectors will have an attractive effect to individual itself and also the best one will never be changed at that iteration. At this point, different neighborhood topology descriptions are introduced. Population itself is also studied in many different aspects. Dynamic population sizing or dividing the population into subgroups may be promising to solve some of the optimization problems.

Inspirations from other stochastic-based algorithms such as GA and simulated annealing or opposition-based learning algorithm are other popular methodologies to improve the basic PSO algorithm. Commonly used GA reproduction operators including selection, crossover, elitism, and mutation can be deployed in PSO algorithm architecture. Among GA operators, mutation is the most utilized. Because the drawback of PSO is due to the lack of diversity, which forces the swarm particles to converge to the position found so far which may not even be a local optimum. Without an effective diversity enhancing mechanism, the optimization algorithm may not be able to efficiently explore the search space. Mutation operators introduce new individuals into a population by manipulating a current individual, thus adding diversity into the population and probably preventing stagnation of the search in local optima. However, the mutation application procedure brings some new adjustments to the algorithm such as the criteria of mutation applications, the position where mutation will be applied, and the selection of random probability distribution sequence. Describing diversity thresholds, mutation probability percentages, similarity and closeness to each other in terms of Euclidian distance may be the criteria for mutation applications. Gaussian, Cauchy, Beta distributions, chaotic distribution based on logistic maps, or mixed types such as cloud distribution are possible random distribution sequences. Randomly selected current positions, velocities, or even individual best position and global best position vectors may be mutated [27-37]. However, elitism concept is integrated into the algorithm in the case of mutating the individual best or global best position vectors.

Although mutations provide diversity within the population they may also cause a peripheral stagnation search. Therefore, the type of mutation operator and its application

strategy are important decision parts for the mutation applications. Apart from designing new mutation operators, researchers have relatively put less effort into investigating how to apply mutation operators during the process and determining what kind of diversity should be provided within the population in PSO process.

1.8 Dissertation Objectives

This research aims to demonstrate the effect of diversity concepts within GA and PSO optimization processes. To determine its effects quantitative expressions must be described and tested. After studying the quantification of the diversity concept, it should then be qualified in terms of local and global search capabilities or random and controlled applications.

Within this scope, mathematical expressions for diversity need to be described, compared, and analyzed for selected benchmark test functions. Special attention will be paid to mutational operations, because the main source for diversity in the population or swarm originates from mutations. To classify diversity in terms of local and global search capabilities, or random and controlled applications, new qualitative mathematical expressions will be described. To introduce controlled diversity concept, popular deterministic intelligent methods, such as, fuzzy logic and neural networks will be coupled with mutational operations.

After quantitative and qualitative studies of the diversity concept, a new mutation strategy will be constructed, tested, and approved for improved search capability. New methods need to be implemented on selected problems to determine the performance of the newly developed methods for aeronautical engineering applications.

2. METHODOLGY

2.1 Control Parameters in GA

The important control parameters of a simple GA include the population size P, crossover fraction P_c, and mutation rate R_m. Usually populations are of constant size. Depending on the preferred genetic way, there may be other important control parameters. Crossover fraction determines the number of children, N_c generated by crossover operations in the next generation. It also automatically determines the number of individuals mutated by mutation operator, N_m. Mutation rate determines the number of mutated genes, S_m related to the length of chromosome (individual), n within the population. N_c, N_m, and S_m are integer values and computed as follows

$$N_c = P_c.P \tag{2.1}$$

$$N_m = (1-P_c).P \tag{2.2}$$

$$S_m = n.R_m.N_m \tag{2.3}$$

2.2 Genetic Operators

Search is one of the most universal problem-solving methods for problems in which one cannot determine a priori the sequence of steps leading to a solution. Typically, there are two types of search behaviors: a global search and a local search. Global search explores the entire solution space and is capable of achieving escape from a local optimum. Local search exploits the best solution and is capable of climbing upward toward a local optimum. An ideal search should posses both types simultaneously.

In essence, genetic operators perform a random search and cannot guarantee to yield improved offspring. In conventional meaning, the *crossover* operator is used as the principal operator. It performs a random search to try to explore the area beyond a local optimum. In general understanding, the *mutation* operator which produces spontaneous random changes in various chromosomes is used as a background operator and it performs a local search to try to find an improved solution. However, we will show that a mutation operator with a proper application strategy can perform a search to explore and exploit the solution space during the generation process.

Crossover

Because it serves well as a paradigm for other genetic operators, it would be better to look first at crossover. In biological systems, crossover is a process yielding recombination of genes via exchange of segments between pairs of chromosomes. Crossover proceeds in two steps; within the first step, two individuals as parents are randomly selected from the current population. In the second step, a new structure is formed from parents by exchanging their genes; so the crossover recombines features from two parents to produce offspring. Sometimes the crossover would recombine the best features from the parents, resulting in superior offspring, but not always.

There are different types of crossover operators. Some well used genetic operators based on crossover are uniform, single point, two-point, intermediate, heuristic, arithmetic, and blend crossover known as BLX-β. In the next sections these crossover operators are going to be compared. Therefore, each method is explained in the followings. Before proceeding, assume that the parents as father (F) and mother (M), a child as (C), and the fitness values of them are selected as

$$x_j^F, x_j^M, \; j=1,2,\ldots,n \tag{2.4}$$

$$f^F, f^M \tag{2.5}$$

Uniform crossover A child is generated as the following

$$\left.\begin{array}{ll} if & r<0.5 \quad x_j^C = x_j^F \\ if & r \geq 0.5 \quad x_j^C = x_j^M \end{array}\right|^{j=1,2,\ldots,n} \tag{2.6}$$

where r is a random number between (0,1). In uniform crossover approximately half of the genes of the child come from each parent.

Single point crossover A child is generated as the following

$$x_j^C\Big|^{j=1,2,\ldots,n} = [x_1^F, x_2^F, \ldots, x_p^F, x_{p+1}^M, x_{p+2}^M, \ldots, x_n^M] \tag{2.7}$$

where p is a random integer number between (1, n).

Two-point crossover A child is generated as the following

$$x_j^C\Big|^{j=1,2,\ldots,n} = [x_1^F, x_2^F, \ldots, x_p^F, x_{p+1}^M, x_{p+2}^M, \ldots, x_q^M, x_{q+1}^F, x_{q+2}^F, \ldots, x_n^F] \tag{2.8}$$

where p and q are random integer numbers between (1, n).

Intermediate crossover A child is generated as the following

$$\left.\begin{aligned} x_j^C &= x_j^F + \alpha(x_j^M - x_j^F) \\ \alpha &= u.R \end{aligned}\right|^{j=1,2,\ldots,n} \tag{2.9}$$

where u is a random number between (0,1) and R is the ratio determined by the user. The ratio controls the range over the children.

Heuristic crossover A child is generated as the following

$$\left.\begin{aligned} &if \quad f^F < f^M \quad x_j^C = x_j^M + R(x_j^F - x_j^M) \\ &if \quad f^M < f^F \quad x_j^C = x_j^F + R(x_j^M - x_j^F) \end{aligned}\right|^{j=1,2,\ldots,n} \tag{2.10}$$

where R is the same as the previous one as ratio determined by the user. The ratio is selected as 1.2 or more.

Arithmetic crossover A child is generated as the following

$$x_j^C = \lambda x_j^F + (1-\lambda)x_j^M \Big|^{j=1,2,\ldots,n} \tag{2.11}$$

where λ is a random number between (0,1).

Blend crossover The children is generated as the following

$$\left.\begin{aligned} x_j^{C_1} &= \beta x_j^F + (1-\beta)x_j^M \\ x_j^{C_2} &= (1-\beta)x_j^F + \beta x_j^M \end{aligned}\right|^{j=1,2,\ldots,n} \qquad \beta = Tu - 0.5 \tag{2.12}$$

where u is a random number between (0,1), T is the constant determined by a user. In this method extrapolation between parent values is possible.

Mutation

Mutation is one of the most familiar of the genetic operators. In genetics, mutation is a process wherein one gene is randomly replaced by or modified to another to yield a new structure. Generally there is a small probability of mutation at each gene in the structure. Like crossover operators, there are different types of mutation methods. Some of them are uniform, adaptive feasible and Gaussian mutation. In the next sections these mutation operators are going to be compared. Therefore, each method is explained in the following. Before proceeding, assume that the current generation is labeled with g, the maximum generation is G, a child is M, and the individual selected for mutation is indexed with P such as

$$x_j^P \Big|^{j=1,2,\ldots,n} \tag{2.13}$$

Gaussian mutation A child is generated as the following

$$\beta_2 = \beta_1 - \gamma \beta_1 \frac{g}{G} \tag{2.14}$$

$$\beta_3 = \beta_2 (x_j^U - x_j^L) \Big|^{j=1,2,\ldots,n} \tag{2.15}$$

$$x_j^M = x_j^P + \beta_3 .u \Big|^{j=1,2,\ldots,n} \tag{2.16}$$

where β_1 is a user defined scale factor, γ is other user defined shrink factor, u is the random number generated from standard normal distribution. U and L are the upper and lower bound indexes for the genes. Mutation rate is not applicable to this type of operator because all genes in chromosome are mutated.

Uniform mutation A child is generated as the following

$$\Delta x_j = x_j^U - x_j^L \Big|^{j=1,2,\ldots,n} \tag{2.17}$$

$$\begin{aligned} x_i^M &= x_i^L + \Delta x_i .u \Big|^{i=1,2,\ldots,I_M} \\ I_M &= nR_m \end{aligned} \tag{2.18}$$

where u is the random number generated between (0,1). In this application not all genes are mutated. Instead, randomly selected I_M genes are mutated depending on user defined R_m.

Adaptive feasible mutation A child is generated as the following

$$x_j^M = x_j^P + \alpha \beta d \Big|^{j=1,2,\ldots,n} \tag{2.19}$$

where α is the step size, β is the scale factor, and d is the search direction. These values are not user defined and computed during the iterations. As it is in Gaussian type applications, all genes are mutated. However this operator is a bit different from others. It is deterministic and not stochastic based. The direction is found by a kind of line search.

2.3 Quantification of Diversity

The word *diversity* means "the condition of being diverse, variety", or "the quality or state of being composed of many different elements or types." [38]. Diversity refers to Euclidian distance among the members in a group. We can describe three different diversity scales. These are the diversity of design variable, $D_{v,g}$; the diversity of individual, $D_{i,g}$; and the diversity of population, $D_{P,g}$ such as

$$D_{v,g}\Big|_{g=1,2,\ldots,G}^{v=1,2,\ldots,S} \tag{2.20}$$

$$D_{i,g}\Big|_{g=1,2,\ldots,G}^{i=1,2,\ldots,p} \tag{2.21}$$

$$D_{P,g}\Big|_{g=1,2,\ldots,G}^{P=1} \tag{2.22}$$

where S is the problem dimension, g is the number of generation, G is the maximum number of generations, p is the population size, and P is the number of population which is typically 1 if there is only one population instead a group. It is possible to determine each diversity type as a single value based on the general and selected positions in the population. This can be achieved by three different definitions: the difference between a member and the best member; the difference between a member and average member value; and the difference among the members.

Diversity of Design Variables

The diversity of any design variable at each generation can be computed by using the following equations;

$$D_{v,g}^{j}\Big|_{g=1,2,\ldots,G}^{j=1,2,\ldots,S} = \left[\sum_{i=1}^{p}(x_{i,j}(g)-x_j^e(g))^2\right]^{\frac{1}{2}} \tag{2.23}$$

$$D_{v,g}^{j}\Big|_{g=1,2,\ldots,G}^{j=1,2,\ldots,S} = \left[\sum_{i=1}^{p}(x_{i,j}(g)-x_j^b(g))^2\right]^{\frac{1}{2}} \tag{2.24}$$

$$D_{v,g}^{j}\Big|_{g=1,2,\ldots,G}^{j=1,2,\ldots,S} = \left[\sum_{i=1}^{p}(x_{i,j}(g)-\overline{x}_j(g))^2\right]^{\frac{1}{2}}$$
$$\overline{x}_j(g)=\frac{1}{p}\sum_{i=1}^{p}x_i^j(g) \tag{2.25}$$

where x^e_j is the j^{th} variable value belonging to elite individual at g^{th} generation, x^b_j is the j^{th} variable value belonging to the best individual in that population at g^{th} generation, and $\overline{x}_j$ is the averaged j^{th} variable value at g^{th} generation.

Diversity of Individuals

We can categorize individual diversity into two types such as individual diversity and individual dependent diversity. Any individual diversity at each generation can be computed by using the following equations;

$$D_{i,g}\Big|_{g=1,2,\ldots,G}^{i=1,2,\ldots,p} = \left[\sum_{j=1}^{S}(x_{i,j}(g)-x_j^e(g))^2\right]^{\frac{1}{2}} \tag{2.26}$$

$$D_{i,g}\Big|_{g=1,2,\ldots,G}^{i=1,2,\ldots,p} = \left[\sum_{j=1}^{S}(x_{i,j}(g)-x_j^b(g))^2\right]^{\frac{1}{2}} \tag{2.27}$$

$$D_{i,g}\Big|_{g=1,2,\ldots,G}^{i=1,2,\ldots,p}=\left[\sum_{j=1}^{S}(x_{i,j}(g)-\bar{x}_j(g))^2\right]^{\frac{1}{2}}$$
$$\bar{x}_j(g)=\frac{1}{p}\sum_{i=1}^{p}x_{i,j}(g) \tag{2.28}$$

Individual dependent diversity is the diversity between two different individuals in the population. Assume that x_k is the k^{th} individual and x_i is the i^{th} individual, then dependent diversity is given by

$$D_{i,g}^{k}\Big|_{g=1,2,\ldots,G}=\left[\sum_{j=1}^{S}(x_{i,j}(g)-x_{k,j}(g))^2\right]^{\frac{1}{2}}$$
$$D_{i,g}^{k}=D_{k,g}^{i} \tag{2.29}$$

Diversity of Population

The diversity of population can be computed by using many different ways. At first, we can calculate this value based on the distance between the design variables as we did in the previous sections. Secondly, we can compute the population diversity based on the fitness values. The population diversity related to design variable distances is symbolized as $D^x_{P,g}$, the population diversity related to fitness values is symbolized as $D^f_{P,g}$. $D^x_{P,g}$ can be computed by the following expressions. The first one is related to the difference between the individuals and the best one. It is defined as

$$D_{P,g}^{x}=\frac{1}{p}\sum_{i=1}^{p}\left[\sum_{j=1}^{n}(x_j^i(g)-x_j^b(g))^2\right]^{1/2} \tag{2.30}$$

The second one may be based on the difference directly between the individuals such as

$$D_{P,g}^{x}=\frac{1}{p}\sum_{i=1}^{p}\sum_{j>i}^{p}\left[\sum_{k=1}^{n}(x_k^i(g)-x_k^j(g))^2\right]^{1/2} \tag{2.31}$$

The last one may be constructed of the difference between the individuals and the averaged ones such as

$$D_{P,g}^{x}=\frac{1}{p}\sum_{i=1}^{p}\left[\sum_{j=1}^{S}(x_j^i(g)-\bar{x}_j(g))^2\right]^{1/2} \tag{2.32}$$
$$\bar{x}_j(g)=\frac{1}{p}\sum_{i=1}^{p}x_j^i(g)\Big|^{j=1,2,\ldots,S} \tag{2.33}$$

This description is a kind of averaged population diversity because we divide the total sum with the population size. Additionally, Eq. (2.32) may be manipulated such that

$$D^{x}_{P,g} = \sum_{i=1}^{p}\left[\sum_{j=1}^{s}(x^{i}_{j}(g)-\overline{x}_{j}(g))^{2}\right]^{1/2} \tag{2.34}$$

In this way we can take care of the population size as multiplier. Another approach is fitness value based description. Similarly the diversity, $D^{f}_{P,g}$ can be determined based on the differences between general and selected individual fitness values. The first one is related to the difference between the individuals' fitness values and the best fitness value. It is defined as

$$D^{f}_{P,g} = \left[\sum_{i=1}^{p}(f^{i}-f^{b})^{2}\right]^{1/2} \tag{2.35}$$

The second one may be based on the difference directly between the individuals' fitness values such as

$$D^{f}_{P,g} = \frac{1}{p}\sum_{i=1}^{p}\left[\sum_{j>i}^{p}(f^{i}-f^{j})^{2}\right]^{1/2} \tag{2.36}$$

The last one may be constructed of the difference between the individuals' fitness values and the averaged fitness value such as

$$D^{f}_{P,g} = \left[\sum_{i=1}^{p}(f^{i}-\overline{f})^{2}\right]^{1/2} \tag{2.37}$$

$$\overline{f} = \frac{1}{p}\sum_{i=1}^{p} f^{i} \tag{2.38}$$

2.4 Quantification in a Simple Test Case

In the previous paragraphs we showed that there are many possible definitions for diversity indicators. It may be beneficial to see some of them for better understanding of indication character. For this reason a simple test case is constructed. One of the benchmark test functions, Rastrigin test function, is selected as a test case and some diversity definitions are compared in terms of generation and diversity values. Rastrigin test function is described as the following

$$10n + \sum_{i=1}^{n}[x_{i}^{2} - 10\cos(2\pi x_{i})] \tag{2.39}$$

search range	$\mathbf{x}^{*}$	$\mathbf{f(x}^{*}\mathbf{)}$
$[-5.12, 5.12]^{D}$	[0,0,…,0]	0

The problem dimension is selected as 10. A regular genetic algorithm is used to search optimal x values. The features of genetic algorithm used in test case are tabled in the next

table. The diversity indicator changes are depicted in Fig. 2.1. The diversity definitions related to the positions of individual genes are plotted on the left side. Other diversity definitions related to individual fitness values are shown on the right side. The definition given in Eq. (2.30) is labeled as D_x-1, Eq. (2.31) is labeled as D_x-2, and Eq. (2.32) is labeled as D_x-3. Similarly the definition given in Eq. (2.35) is labeled as D_f-1, Eq. (2.36) is labeled as D_f-2, and Eq. (2.36) is labeled as D_f-3.

Table 2.1 Genetic algorithm features in quantification case

P	*G*	P_c	**Mutation**
10	2000	0.8	Gaussian: β_1, 0.5; γ, 0.75
Fitness scaling	**selection**	**Elite count**	**Crossover**
Rank	Roulette	1	Arithmetic

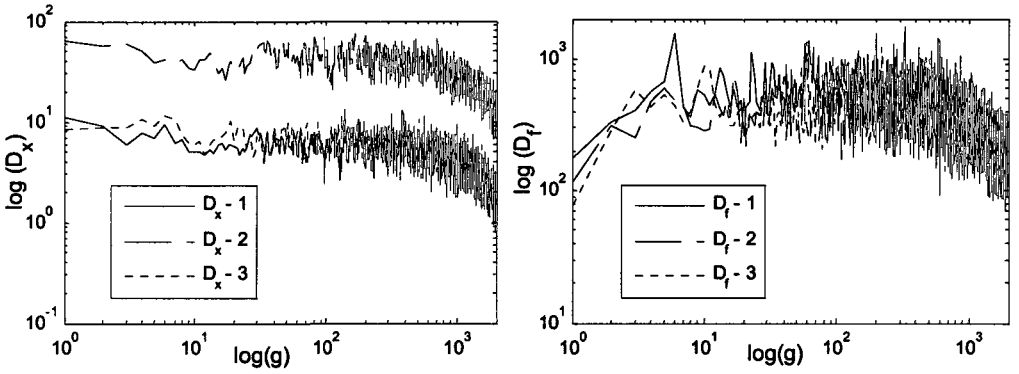

Figure 2.1 The diversity indicator changes versus generations.

Fig. 2.1 shows that there is no significant difference among the diversity indicators. There is just scale variety between D_x-1 and D_x-2. However the curve characteristics are almost the same. The same conclusion is valid for fitness based diversity indicators. Therefore the diversity definition given in Eq. (2.32) and the second version of it, Eq. (2.34) are selected as diversity indicators for the rest of studies.

2.5 Diversity and Crossover Operator Relationships

The diversification is provided by genetic operators during the generations. Both crossover and mutation operators may probably have different diversification characteristics. The crossover operator and diversity relationship is investigated in the next test case bundle. The same test function given in Eq. (2.39) is used in test cases. As diversity indicator, Eq. (2.32) is used. The features of a regular genetic algorithm are expressed in Table 2.2. We excluded mutation operations in reproduction phases to see the pure effect of crossover operators on diversity changes. So, all individuals are generated by crossover operator within the population. 20 runs are fulfilled and the average diversity indicator values are plotted for

each crossover operator. Selected operators are scattered, single point, two-point, intermediate, heuristic, and arithmetic crossover operators.

Table 2.2 Genetic algorithm features in crossover case without mutation

P	G	P_c	**Mutation**
10	200	1	No mutation
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

The diversity and crossover operator characteristic relationships are depicted in Fig. 2.2. This figure is divided into two sections such as binary coded and real coded genetic processes. Scattered, single and two-point crossover operators are used in binary coded process and their relationships with diversity are shown on the left side of the figure. Intermediate, heuristic, and arithmetic crossover operators are real coded processes and their relationships with diversity are shown on the right side of the figure. For both types the relationships are similar characteristic behaviors. In binary coded process the diversity shows linear change in logarithmic scale until a certain generation number. After about 23rd iteration the diversity is significantly decreased. Then the diversity becomes constant. This type behavior is almost the same for all crossover operators in binary coded process. However, other operators in real coded process behave in different modes. The diversity shows linear change in logarithmic scale until a certain generation and then becomes stagnant. We do not see a sudden decrease; instead, we observe a gradual decrease in diversity values. From this point we can conclude that real type coding provides higher diversity among the individuals than binary type coding. On the other hand, the difference among the crossover operators within the same coding type is almost negligible. It seems it does not matter what kind of a crossover operator is used to generate an individual from the point of view of diversity.

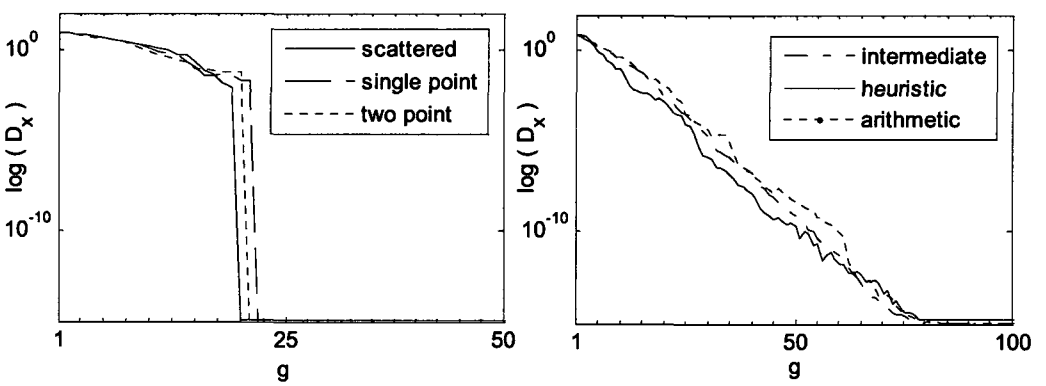

Figure 2.2 The diversity and crossover method relationships.

In the next step, we want to determine the effect of a fixed type mutation operator has on diversity for the same crossover operators. For that reason, a new test case bundle is taken into consideration. Rastrigin test function with the same constraints is used for test cases. The

features of a regular genetic algorithm are expressed in Table 2.3. 20 runs are fulfilled and the average diversity indicator values are plotted for each crossover operator. A common mutation operator is selected as a uniform mutation operator which is applicable to both binary and real coding systems. The mutation rate, R_m is equal to 0.05.

Table 2.3 Genetic algorithm features in crossover case with uniform mutation

P	*G*	P_c	**Mutation**
10	200	0.8	Uniform
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

The diversity and crossover operator characteristic relationships are depicted in Fig. 2.3. Similar to previous analysis, this figure is divided into two sections such as binary coded and real coded genetic processes. Interestingly, mutation operator significantly changes the diversity and crossover operator relations. Without mutation there was a strong difference between binary and real coded processes. However, a mutation operator weakens the difference among the crossover operators. Both binary and real coded crossover operators show almost the same diversity-generation characteristics. Diversity values are decreased linearly in logarithmic scale until a certain generation and then become stagnant with a certain periphery. It seems that a mutation operator keeps the diversity level at the same degree with some exceptional points. Additionally, binary coded system operators show relatively higher diversity values than real-coded ones. This is an expected result due to sensitivity of binary coding to the digital changes.

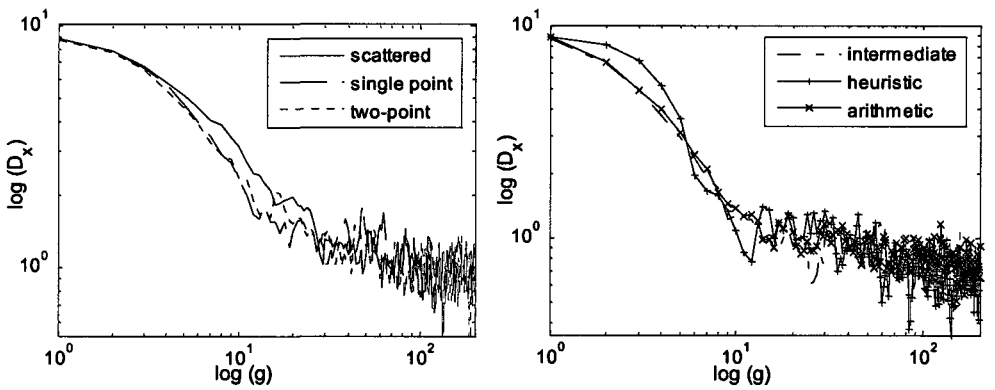

Figure 2.3 The diversity and crossover method relationships under common mutation effect.

The last analysis will be based on the crossover ratio, P_c. We observed that there is no serious difference among the crossover operators under a common mutation operator effect. However, we kept the crossover ratio as 0.8 during the test runs. It means only 20 percent of the total population is going to be mutated with a certain mutation ratio. Therefore, we need to look at the effect that the crossover ratio has on the diversity changes. To do this analysis

we select a heuristic crossover operator with R 1.2 and a Gaussian mutation operator as genetic operators. We change the crossover ratio from 0.2 to 1.0. The mutation operator parameters are selected as β_1, 0.5; γ, 0.75. The features of regular genetic algorithm are expressed in Table 2.4. 20 runs are fulfilled and the average diversity indicator values are plotted for each crossover ratio.

Table 2.4 Genetic algorithm features in crossover case with Gaussian mutation

P	*G*	**Mutation**	
10	200	Gaussian	
Fitness scaling	**selection**	**Elite count**	**Run**
rank	roulette	1	20

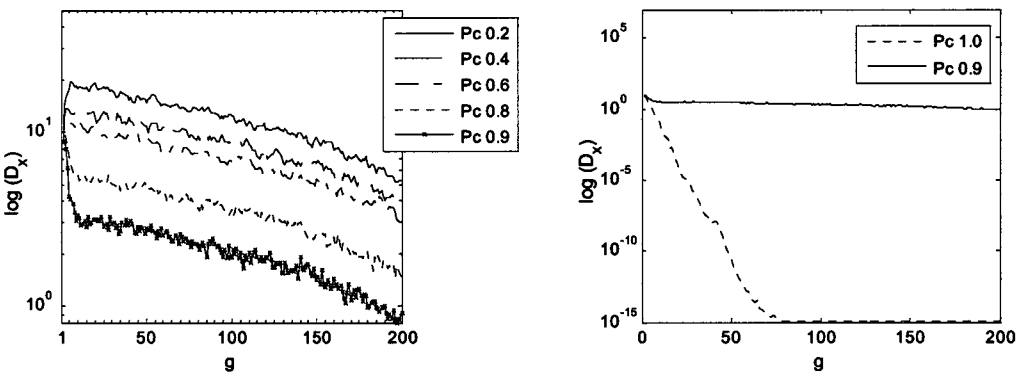

Figure 2.4 The diversity and crossover fraction relationships under common mutation effect.

The evaluation of Fig. 2.4 shows that the crossover fraction significantly effects the diversity changes. Low level fraction causes high level diversity. Actually the real parameter here is $(1\text{-}P_c)$ value. This value determines the number of individuals which will be mutated by mutation operator. If P_c is equal to 1 it means that there is no individual to be mutated so the diversity significantly decreases. This big gap can be observed on the right side of the figure. Ten percent mutated individuals per population (P_c is equal to 0.9) increase the diversity with logarithmic scale.

2.6 Diversity and Mutation Operator Relationships

As we show that the crossover operator has little effect on the level of diversity. The main effect comes from the number of mutated individuals. Now, we need to analyze the relationship between diversity and a mutation operator type. For the test bundle, Rastrigin test function with the same constraints is used. As a diversity indicator, Eq. (2.32) is used. The features of a regular genetic algorithm are expressed in Table 2.5. 20 runs are fulfilled and the average diversity indicator values are plotted for each mutation operator. A common crossover operator is selected as a heuristic crossover operator with R 1.2. The crossover

fraction is set to P_c 0.8. The selected mutation operators to be tested are Gaussian with β_1, 0.5; γ, 0.75, uniform with R_m 0.05, and adaptive feasible mutation operators.

Table 2.5 Genetic algorithm features in heuristic crossover case

P	*G*	**Crossover**	
10	200	Heuristic, P_c 0.8	
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

According to Fig. 2.5 the mutation operator determines the level of diversity. Uniform and adaptive feasible mutation operators show similar behaviors. However, a Gaussian mutation operator has similar characteristic with different scales. The diversity decreases fast to a certain degree and then becomes stagnant with different periphery. This is an expected result because of mutation rates. In Gaussian mutation applications all genes of selected parents are mutated. It means R_m is equal to 1. However, the mutation rate is restricted to R_m 0.05 in uniform mutation applications. This ratio directly affects the diversity. In adaptive feasible mutation applications, all genes are also mutated, but the deterministic search direction seems to decrease the diversity. On the other hand, the number of mutated individuals within the population is the same.

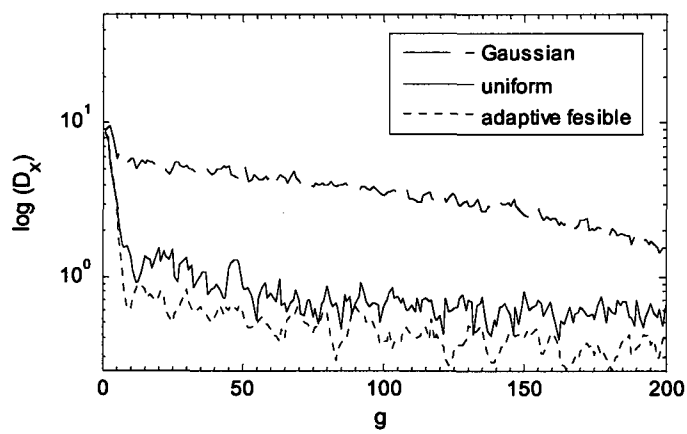

Figure 2.5 The diversity and mutation operator relationship under common crossover operator effect.

2.7 Diversity and Population

Up to now, we observed that the diversity originates mainly from the number of mutated individuals and secondly from the type of mutation operator. The first parameter determines the scale and the second parameter determines the periphery. For the next analysis we will look directly at the population itself. In this case, diversity indicators given in Eq. (2.32) and Eq. (2.34) are used separately under the same conditions. In principle, the small

populations run the risk of seriously under-covering the solution space, and so increase the chance of premature convergence to a poor solution. On the other hand, larger populations allow the exploration of fewer generations per unit of computational efforts [39]. We analyze the population size and diversity under common crossover and mutation genetic operators. The features of a regular genetic algorithm are expressed in Table 2.6. 20 runs are fulfilled and the averaged diversity indicator values computed via Eq. (2.32) are plotted for each population size. A common crossover operator is selected as a heuristic crossover operator with R as 1.2. The crossover fraction is set to P_c 0.8.

Table 2.6 Genetic algorithm features in population case

Mutation	G	**Crossover**	
Uniform Gaussian	200	Heuristic, P_c 0.8	
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

The diversity and population size characteristic relationships are depicted in Fig. 2.6. The analysis is divided into two sections. At first, the diversity is observed under different population sizes and uniform mutation operator with R_m 0.05. Upper side of Fig. 2.6 shows this case's relationship. Interestingly, the increase in population size does not bring more diversity within the population. The same conclusion is observed on the lower side for Gaussian mutation with β_1, 0.5; γ, 0.75, too. At the beginning of the generations the population size causes slightly higher diversity level, and then this effect disappears through the proceeding generations. However, this type of diversity is averaged population diversity. Therefore, we need to look at the values of non-averaged population diversity given in Fig. 2.7.

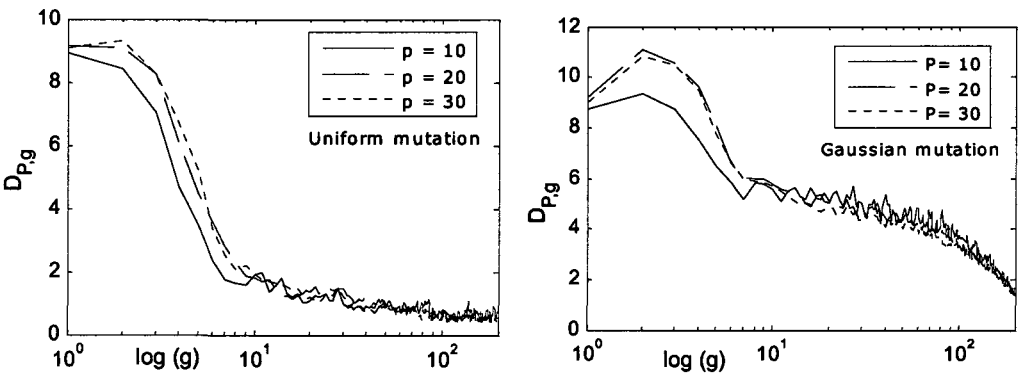

Figure 2.6 The diversity and population size relationship in accordance with Eq. (2.32).

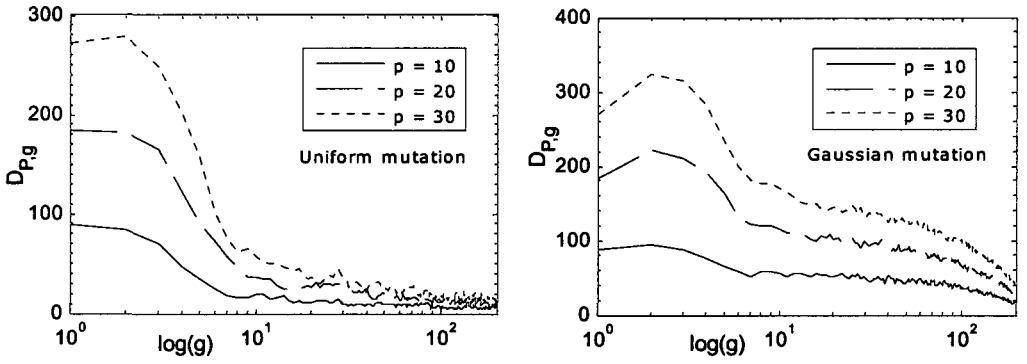

Figure 2.7 The diversity and population size relationship in accordance with Eq. (2.34).

In Fig. 2.7, we see that the population size increases the population diversity with multiplier effects. As a result, more population size does not mean more averaged population diversity, but means higher population diversity. Additionally, we need to put this point that more population size means higher convergence rate. This takes us to other inference; more diversity in the population may not mean faster convergence rate. Therefore, in addition to quantification of diversity, we need to focus on the qualification of the diversity. Before this subject, we will have a closer look on the mutation criteria.

2.8 Mutation with Threshold Diversity

In regular genetic algorithms we see that the diversity is controlled by the number of mutated individuals in terms of the ratio of population size and the mutation operator type. Instead of this classical approach we can control the diversity directly itself. For example, one of the current mutation operators or a new type may be activated only when the diversity of the population becomes less than a certain threshold value, D_{low}. The new mutation application is named as threshold mutation operator and described in the following equation.

$$x_j^i = \left[\begin{matrix} if\ D < D_{low} & x_j^i = x_j^i + \xi u \\ if\ D > D_{low} & x_j^i \end{matrix} \right]\Bigg|_{j=1,2,\ldots,n}^{i=1,2,\ldots,P} \tag{2.40}$$

where ξ is the scaling factor, u is a random number generated by a Gaussian distribution. The diversity description given in Eq. (2.40) is based on the differences between the average dimensional position and the current dimensional particle positions, D_x. In this approach, the whole population is mutated under threshold diversity criteria. In the first generations the diversity will be crossover operator dependent. However, when the diversity is decreased, the mutation operator is going to be activated and this activation continues until the desired diversity is provided. The effects of this approach can be compared and analyzed with different parameters settings. For the test bundle, Rastrigin test function given in Eq. (2.39)

with the same constraints is used. The features of a new genetic algorithm are expressed in Table 2.7. 20 runs are fulfilled and the average diversity indicator values are plotted for each threshold diversity value. Fig. 2.8 shows the relation between threshold diversity value and population diversity. It is observed that the threshold diversity value determines the population diversity level. Naturally, the population diversity is decreased due to generational crossover operations. When the mutation operator is activated, the population diversity is increased and the diversity level is kept on threshold diversity level with a certain periphery. Higher threshold diversity value means higher population diversity level. Another interesting point inferred from the figure is the periodicity. It seems that the determination of diversity level brings an unclear, but dominant periodic mutation applications. This observation is more understandable in Fig. 2.9.

Table 2.7 Genetic algorithm features in mutation with different threshold case

Mutation	***G***	**Crossover**	***P***
Threshold with ξ 0.5	2000	Blend with T 2	10
Fitness scaling	**Selection**	**Elite count**	**Run**
Rank	Roulette	1	20

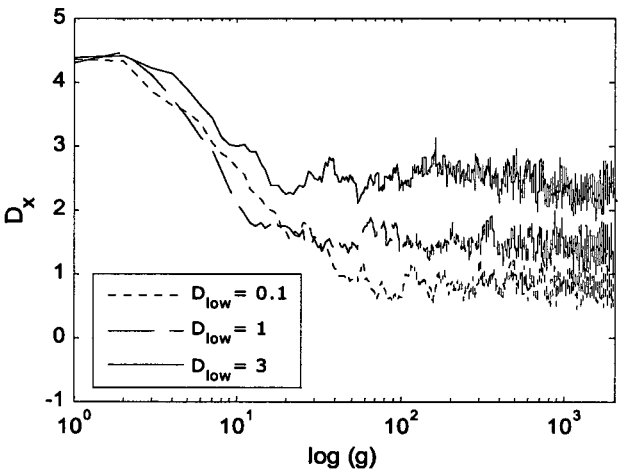

Figure 2.8 The diversity and different threshold diversity relationship under common crossover operator and scale factor.

Fig. 2.9 shows the effect of a scale factor on the population diversity under a fixed threshold diversity value. The features of a genetic algorithm are expressed in Table 2.8. Different scale factors cause different population diversity characteristics. Higher scale factors provide larger band diversity changes and cause more mutation operator activity. On the other hand, lower scale factor generates at a lower diversity band and with rare mutation operator activation. Both figures also show that each parameter in Eq. (2.40) effects both the diversity band width and the activation frequency. This may result in an interaction between these parameters. However, we need to get a closer look at the periodicity of diversity.

Table 2.8 Genetic algorithm features in mutation with fixed threshold case

Mutation	***G***	**Crossover**	***P***
Gaussian with D_{low} 0.1	2000	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

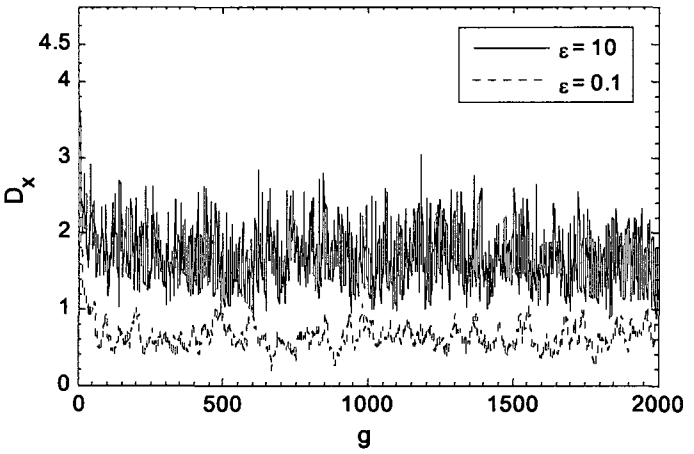

Figure 2.9 The diversity and different scale factor relationship under common crossover operator and threshold diversity.

2.9 Wavelet Analysis

Before the analysis of signal it had better to deal with signal analysis. We can express any signal, s in terms of detail signals such as

$$s = \sum_{j \in Z} D_j \tag{2.41}$$

$$D_j = \sum_{k \in Z} \alpha_{j,k} \psi_{j,k} \tag{2.42}$$

where $\alpha_{j,k}$ is a coefficient and $\psi_{j,k}$ is a wavelet function. This expression is usually divided into approximate and detail signals such as

$$s = A_J + \sum_{j \le J} D_j \tag{2.43}$$

where A_J is the approximate signal and D_j are the detail signals [40]. The biggest part of the signal energy is carried by an approximate signal. The remaining energy is shared by detail signals sometimes called noise signals. There are different wavelet functions in literature. In our analysis we preferred to use Meyer wavelets [41].

The wavelet analysis is based on the genetic process described in Table 2.9 for the same optimization problem. The whole diversity signal can be seen in Fig. 2.10.

Table 2.9 Genetic algorithm features in wavelet analysis

Mutation	***G***	**Crossover**	***P***
Threshold with D_{low} 1, ξ 0.5	10000	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

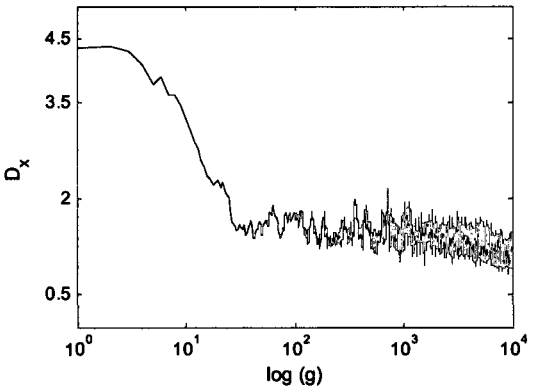

Figure 2.10 The diversity signal.

The diversity change in terms of the last 2000 generations is included for steady result. This signal can be analyzed by using discrete approximation of Meyer wavelets at 10-level. So, the diversity signal is expressed by the following

$$D_x = A_{10} + \sum_{j=1}^{10} D_j \tag{2.44}$$

The resulted discrete wavelet transform for the diversity can be seen in Fig. 2.11. The figure includes only approximation signal A_{10} and detail signals D_j, j; 7, 8, 9, and 10. The remaining signals are noise signals. The picture shows that the main diversity signal caries some locally periodic signals and noises.

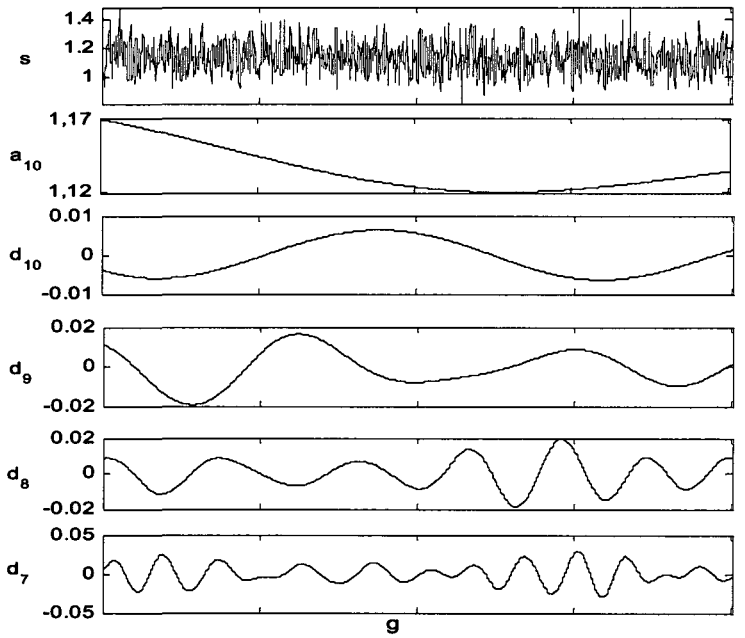

Figure 2.11 Wavelet analysis of a part of diversity signal.

Up to now we analyzed that the description of threshold diversity brings new parameters such as periodicity and band width, let's say a kind of amplitude. For the next step, we focus on the relationship between threshold value and the fitness value.

2.10 Threshold Effect on Fitness

Different threshold diversity values result in different convergence speed. To test this issue we can use the same test set up as the previous one. The features of genetic algorithm are expressed in Table 2.10. 20 runs are fulfilled and the last value of average diversity indicator is plotted for each threshold diversity value. Fig. 2.12 shows that the relation between threshold diversity and best fitness value is nonlinear. It means we need to tune up the threshold diversity for the best convergence and this relationship may also have more than one local minimum. For the next step, we will compare the new type mutation activation and the regular mutation application with regular mutation operators.

Table 2.10 Genetic algorithm features in threshold effect

Mutation	***G***	**Crossover**	***P***
New Gaussian with ξ 0.5	10000	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

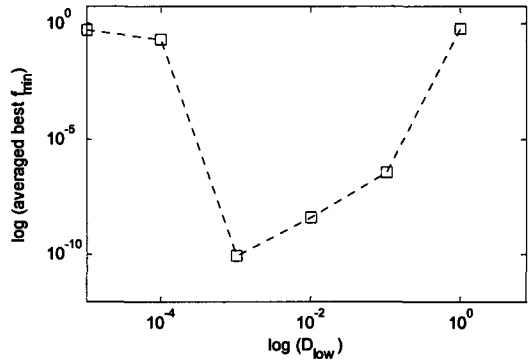

Figure 2.12 Averaged best objective function values versus different threshold values.

2.11 Comparison of Mutation Applications

It will be beneficial to compare new type mutation activation and application method with the previous mutation application and operator approaches. For that reason the same optimization problem is included. The features of genetic algorithms are expressed in Table 2.11 and 2.12. In regular genetic algorithms, the uniform mutation operator is with R_m 0.05, Gaussian mutation is with β_1, 0.5; γ, 0.75. 20 runs are fulfilled and the averages of the best objective function values are plotted for each mutation type. Fig. 2.13 depicts the results.

Table 2.11 New type genetic algorithm features

Mutation	***G***	**Crossover**	***P***
Threshold with D_{low} 0.1 and ξ 0.5	5000	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

Table 2.12 Regular genetic algorithm features

Mutation	***G***	**Crossover**	
Uniform	5000	Heuristic with P 1.2	
Gaussian	***P***	P_c 0.8	
Adaptive feasible	10		
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

It seems that new mutation activation strategy provides a better convergence result than the regular mutation activation methodologies for the current test function, *Rastrigin*. New strategy decreases the required generations at least 40 % comparing with the best of regular ones. Regular mutation applications could not save the population from premature convergence.

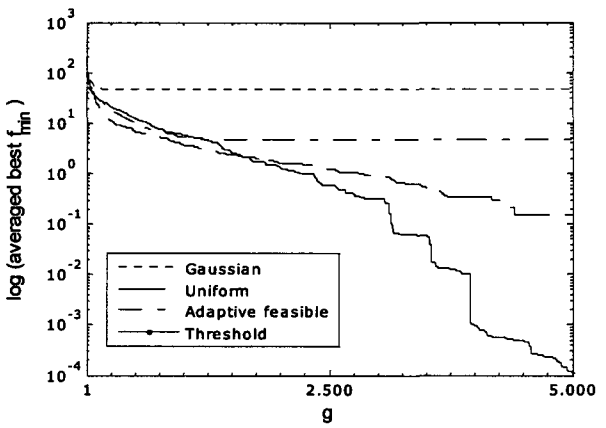

Figure 2.13 Averaged best objective function values versus different threshold values.

Mutation activation dependent on diversity may be beneficial due to high and guarantied diversification. However, finding a proper threshold value may not be easy and may take extra study. Additionally, fixed diversity rate may not be proper for more accurate convergence in different test functions. Instead of using threshold diversity, we may analyze the periodicity and amplitude features in terms of a new mutation concept which includes both of them.

2.12 Periodic Mutation Activation

At the beginning of the genetic process we determine the crossover fraction and then naturally the number of mutated individuals in regular genetic algorithms. This fraction is fixed at each generation. Instead of this approach we can describe the same fraction such as

$$P_c = \begin{bmatrix} 1 \; if \; g = mf \\ 0 \; if \; g \neq mf \end{bmatrix} \Bigg|^{m=1,2,3,\ldots} \tag{2.45}$$

where f is the frequency, let's say mutation application frequency, n is the integer number. This strategy means that all individuals within the population are going to be mutated in every f generations, in other generations all individuals are generated by only crossover operations. We did not describe any mutation operator; we just described the mutation application strategy. Now, we can describe a new mutation operator which has an amplitude factor.

Vibrational mutation operator Within the periodic mutation strategy, a child can be generated as the following

$$x_j^i = \begin{bmatrix} x_j^i[1+w\beta(1-u)], & if \; g = mf, \; ^{m=1,2,\ldots} \\ x_j^i, & if \; g \neq mf, \; ^{m=1,2,\ldots} \end{bmatrix} \Bigg|^{\substack{i=1,2,\ldots,P \\ j=1,2,\ldots,n}} \tag{2.46}$$

where w is a user defined weight number, β is the scale factor which may be user defined or generation dependent, u is a random number generated between (0,1), and f is a user defined application frequency. Apart from other mutation applications, vibrational mutation is not applied in every generation, instead it is applied periodically. The other important point is the applicants. The whole genes of all individuals are mutated in a corresponding generation.

2.13 Effects of Vibrational Mutations

We need to analyze the effect of vibrational mutations on the population diversity and objective function. For this reason we use the same optimization problem based on Rastrigin test function. However, we will decrease the problem dimension to 3 for visual observation of the individual trajectories. In other analyses we will use larger problem dimension.

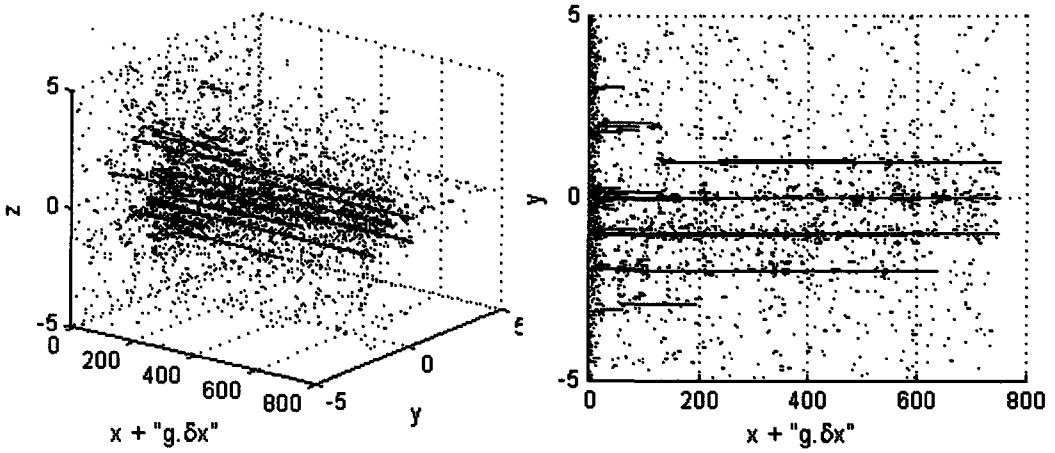

Figure 2.14 Individuals` positions of regular genetic algorithm process.

The individuals' trajectories in 4-dimensional domain can be seen in Fig.2.14 and 2.15. In these plots the x axis of an individual is ridden on the generations (g) with a certain range, δx. Totally 20 runs including 60,000 individual positions are plotted in each figure. The features of used genetic processes are given in Table 2.13 and 2.14. In Fig. 2.14 we can see some drawbacks of the regular genetic process. It is clear that there are multiple local minimums and the algorithm prematurely converges to these minimums. Activated mutations based on uniform mutation operator cause random but global perturbations within the population and it seems that they don't provide beneficial diversity for the population. The other point is the homogeneous distribution of the mutated individuals. On the other hand, the dispersions of the individuals of vibrational genetic process at each mutation period are clearly observed like fishbone during the generations in Fig. 2.15. The dispersions become smaller during the iterations. The mutated individuals are heterogeneously distributed. There are also a few local minimum locations observed until a certain generation. However these local convergences disappear after 200th generation. Almost all runs are converged to a global minimum with certain accuracies.

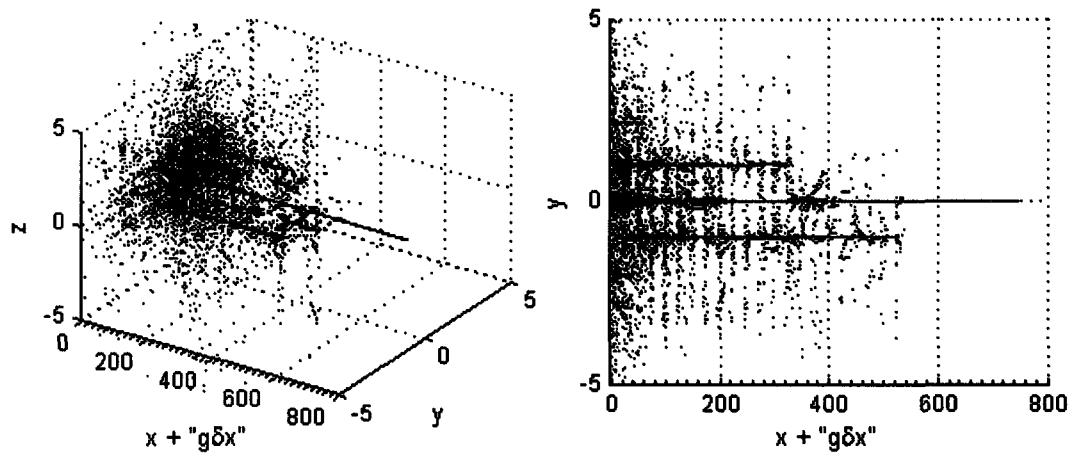

Figure 2.15 Individuals` positions of vibrational genetic algorithm process.

Table 2.13 Vibrational genetic algorithm features

Mutation	***G***	**Crossover**	***P***
Vibrational with f 10, w 1, and β 5	300	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

Table 2.14 Regular genetic algorithm features

Mutation	***G / P***	**Crossover and fraction**	
Uniform with R_m 0.05	300 / 10	Heuristic with P 1.2, P_c 0.8	
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

The diversity change in the population belongs to vibrational genetic process can be seen in Fig. 2.16. The features of the algorithm are given in Table 2.15. The problem

dimension is increased to 10 in this test case. Fig. 2.16 points to three different issues, which are periodicity, adaptive periphery, and adaptive decrease in diversity.

It is clear that the periodic mutation applications cause a periodic change in population diversity. However, the diversity is not fixed at any level. It adaptively decreases to certain levels. This process is not linear. The diversity bandwidth is also changing depending on the generations. The amplitude of change in diversity starts from large fluctuations and ends with small fluctuations. This process matches with the individuals trajectories depicted in Fig. 2.15. When the population gets closer to a global minimum the dispersions also get closer, but there are still dispersions with small peripheries in the population.

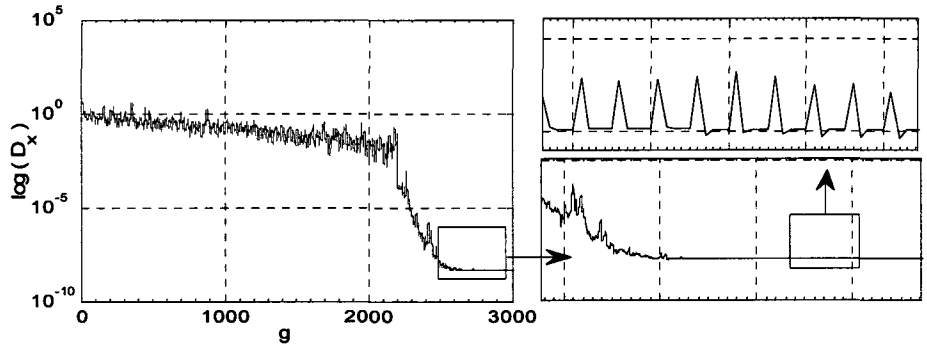

Figure 2.16 Zoomed diversity changes versus generations in vibrational genetic algorithm process.

The effect of vibrations on the best fitness value can be observed in Fig. 2.17. This figure is a repetitive version of Fig. 2.13 except vibrational mutations process. We see that vibrational genetic process outperforms its rivals in Rastrigin function test cases.

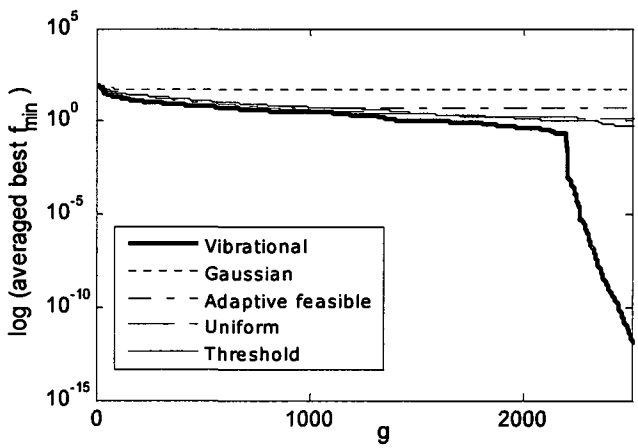

Figure 2.17 Best objective function value changes versus generations in vibrational genetic algorithm process.

Table 2.15 Vibrational genetic algorithm features

Mutation	***G***	**Crossover**	***P***
Vibrational with f 5, w 1, and β 0.5	3000	Blend with T 2	10
Fitness scaling	**selection**	**Elite count**	**Run**
Rank	roulette	1	20

2.14 Qualification of Diversity

Up to now, we analyzed the quantification of diversity within the population with different genetic operators including mutation strategies. It seems that diversification is useful and required for fast and more accurate convergence. However, diversity sometimes may have destructive effects on the population. For example; Gaussian mutation operator causes more diversification in the population as shown in Fig. 2.5, however, its performance is not as good as others as seen in Fig. 2.13. Therefore, we need to analyze the quality of diversity. We think that the quality of diversity can be categorized depending on the distribution character in the search space and/or the determination process. At first, we will focus on the distribution character of the diversity such as global or local diversity and then determination process such as random or controlled diversity.

Global and Local Diversity

If the mutation applications cause a wider range of distribution of the current population in search space it can be called global diversity. Assume that the population, P and the diversity D are described as follows

$$P = c_i \Big|^{i=1,2,\ldots,p} \tag{2.47}$$

$$c_i = \left[c_{i,j}\right]^{j=1,2,\ldots,J} \tag{2.48}$$

$$\mathcal{D} = \frac{1}{p}\sum_{i=1}^{p}\left[\sum_{j=1}^{J}(c_{i,j}-\overline{c}_j)^2\right]^{1/2} \tag{2.49}$$

$$\overline{c}_j = \frac{1}{p}\sum_{i=1}^{p} c_{i,j}\Big|^{j=1,2,\ldots,J} \tag{2.50}$$

where c_i is i^{th} individual curve in P, $c_{i,j}$ is j^{th} discrete point on i^{th} curve. Suppose that the current diversity of the current population is described as D_c, and the new diversity after mutation applications is called as D_n. We describe that if $D_n \gg D_c$ then D_n is called global diversity. An example current population can be seen in Fig. 2.18. In this population the individual curves are close to each other with a certain range. These individuals are globally distributed after vibrational mutation applications as seen in Fig. 2.19. The curves are farther than each other in terms of Euclidean distances; therefore, the new diversity is larger than the

previous one. Vibrational mutation operator described in Eq. (2.46) cause global diversity within the population. Each gene of each chromosome is modified with a certain range.

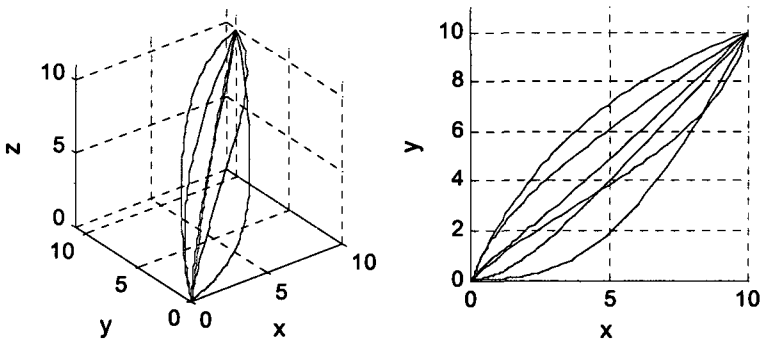

Figure 2.18 Population in search space.

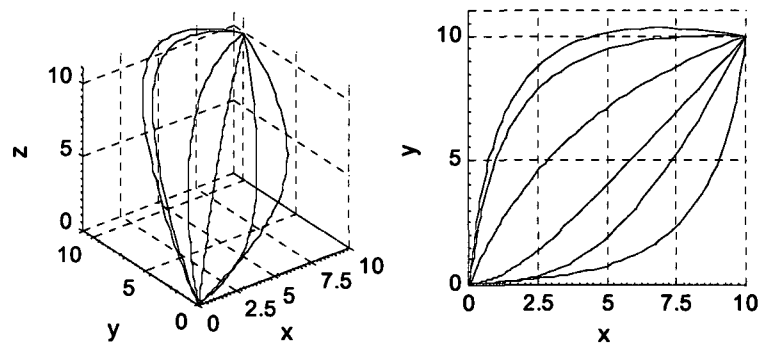

Figure 2.19 Globally distributed population in search space.

Local diversity concept can be based on any individual and its neighborhood. Instead of taking any individual we prefer to get the elite individual. Before taking the next step, let's re-describe individual dependent diversity, $D_{m,n}$ term peculiar to this case. $D_{m,n}$ can be described as the following

$$\mathcal{D}_{m,n} = \left[\sum_{j=1}^{J}(c_{m,j} - c_{n,j})^2\right]^{1/2}$$
$$\mathcal{D}_{m,n} = \mathcal{D}_{n,m} \tag{2.51}$$

where m is the m^{th} individual curve, n is the n^{th} individual curve, and $D_{m,n}$ is individual dependent diversity between these two individuals. Assume that we have three individual curves labeled as c_m, c_n, and c_k in the population. The individual dependent diversities for these curves are called $D_{m,n}$, $D_{m,k}$, and $D_{k,n}$. Let's describe localized diversity, ΔD. If the following criteria

$$\mathcal{D}_{m,n}, \mathcal{D}_{n,k} < \Delta D \tag{2.52}$$

is valid for the curves c_m, c_n, and c_k then we can say that there is a local diversity within the population. An example local diversity can be seen in Fig. 2.20. In this population there are

some individuals which are close to each other. During the genetic process local diversity is going to be common phenomenon within the population. This situation may results from local or global convergence. Therefore, it may or may not be beneficial to the population.

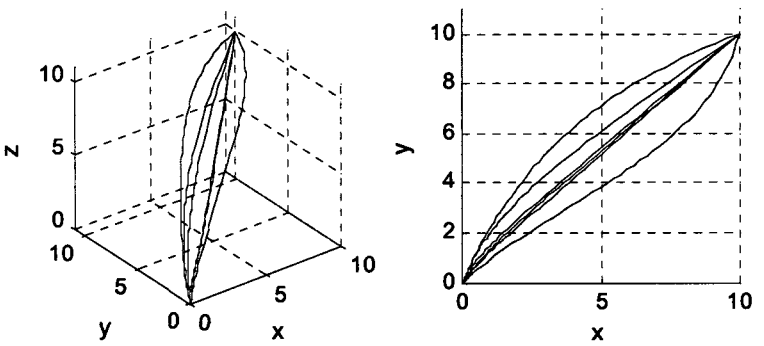

Figure 2.20 Locally distributed population in search space.

Periodic Localization

We can intentionally generate local diversity by using vibrational mutation operator. The application can be done in accordance with the following equation;

$$x_j^i = \begin{bmatrix} x_j^e[1+w\beta(1-u)], & if\ g = mf, & ^{m=1,2,\ldots} \\ x_j^i, & if\ g \neq mf, & ^{m=1,2,\ldots} \end{bmatrix} \Bigg|_{j=1,2,\ldots,n}^{i=1,2,\ldots,q} \tag{2.53}$$

where x^i is the i^{th} individual, n is the problem dimension, w is a user defined weight number, β is the scale factor which may be user defined or generation dependent, u is a random number generated between (0,1), and f is a user defined application frequency. The base vector x^e is the elite individual of the population and q is the number of new individuals locally generated by a vibrational mutation operator.

Multi-frequency Vibrational Mutation Applications

We need to point out that Eq. (2.53) is different from Eq. (2.46). The first one is globally; the second one is locally applied. It is possible to select different frequencies, scale factors, and weight numbers. We can describe this situation such as

$$x_j^i = \begin{bmatrix} x_j^i[1+w_1\beta_1(1-u)], & if\ g = mf_1, & ^{m=1,2,\ldots} \\ x_j^i, & if\ g \neq mf_1, & ^{m=1,2,\ldots} \end{bmatrix} \Bigg|_{j=1,2,\ldots,n}^{i=1,2,\ldots,p} \tag{2.54}$$

$$x_j^i = \begin{bmatrix} x_j^e[1+w_2\beta_2(1-u)], & if\ g = mf_2, & ^{m=1,2,\ldots} \\ x_j^i, & if\ g \neq mf_2, & ^{m=1,2,\ldots} \end{bmatrix} \Bigg|_{j=1,2,\ldots,n}^{i=1,2,\ldots,q} \tag{2.55}$$

We can figure the population in terms of g generations. Assume that f_1 is equal to 5, f_2 is equal to 3. For the first seven generation includes the following populations;

g	1	2	3	4	5	6	7
P	$P_I^i \Big\vert^{i=1,2,\ldots,n}$	$P_C^i \Big\vert^{i=1,2,\ldots,n}$	$P_C^i \Big\vert^{i=1,2,\ldots,n-q}$ $P_{ML}^i \Big\vert^{i=1,2,\ldots,q}$	$P_C^i \Big\vert^{i=1,2,\ldots,n}$	$P_{MG}^i \Big\vert^{i=1,2,\ldots,n}$	$P_C^i \Big\vert^{i=1,2,\ldots,n-q}$ $P_{ML}^i \Big\vert^{i=1,2,\ldots,q}$	$P_C^i \Big\vert^{i=1,2,\ldots,n}$

where P_I is the initial population, P_C is the population generated by crossover operations, P_{ML} is the population locally generated by the second mutation operator, P_{MG} is the population globally generated by the first mutation operator.

At first, the population is composed of randomly generated initial individuals. In the second generation the population only includes the individuals generated by crossover operations. The third generation corresponds to the second frequency; $g \leftarrow nf_2$, $n=1$. In the 3rd generation, the population includes *(n-q)* individuals generated by crossover operations and q individuals generated by the second vibrational mutation operator. In the 4th generation the population only includes the individuals generated by crossover operations. The fifth generation corresponds to the first frequency; $g \leftarrow nf_1$, $n=1$. In the 5th generation the population only includes the individuals generated by the first vibrational mutation operator. The sixth generation corresponds to the second frequency; $g \leftarrow nf_2$, $n=2$. In the 6th generation, the population includes *(n-q)* individuals generated by crossover operations and q individuals generated by the second vibrational mutation operator. In the 7th generation the population only includes the individuals generated by crossover operations. This periodic genetic process goes on until the stopping criterion is satisfied.

Random and Controlled Diversity

Vibrational mutation operators generate new individuals and these new individuals are absolutely random-based estimations for better solutions. Application of the first mutation operator provides global but random diversity within the population. Application of the second mutation operator cause local but random diversity surrounding by elite individual neighborhood in the population. However, we may improve the quality of estimations generated by vibrational mutation operators. We can call this secondary process as filtration or *controlled* diversity.

Control action on mutated individuals before placing them into next population can be done by using artificial intelligent methods. There can be two possible options for control process. These options are called as *correction* and *permission* steps. Fig. 2.21 depicts the control processes.

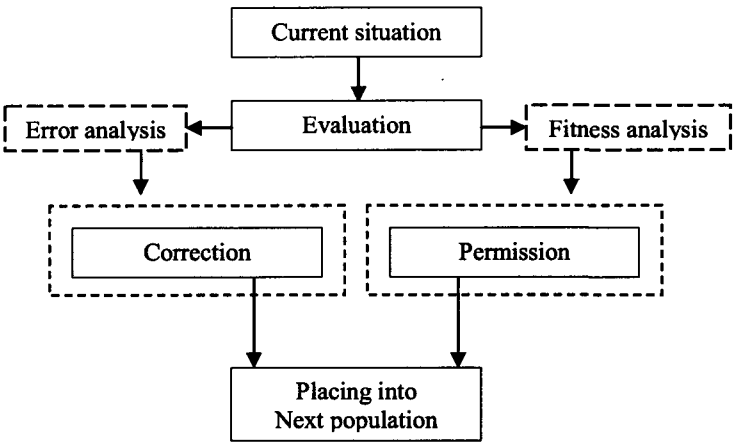

Figure 2.21 Control options on mutated individuals.

In correction process the input as current situation is corrected in accordance with evaluation result. Therefore the current situation is going to be changed. After correction phase the input, let's say mutated individual is placed into next population. In permission process the input is just evaluated and then the decision is made if the input is going to be placed into the next population or not. The second process is simpler than the previous one. However, permission requires multiple inputs for ordering and placing. In this diagram there are some questions to be answered. How can we do evaluation, correction, and permission?

The evaluation can be constructed on objective function value or error value if the problem has a target design. For evaluation step based on objective function value the remedy lies on the feature of genetic algorithms; it is a clear fact that they are population-based algorithms. In each generation the algorithm produces design input parameters and resulted outputs based on objective function computations. These couples are also sample points for the design space. Normally, sample points are selected in accordance with design of experiments (DoE) methods. DoE is a systematic procedure for choosing a set of samples, with the general goal of maximizing the amount of information gained from a limited number of samples. But genetic process produces solution examples instead of sample points. As a result, after each generation we had growing sample points. It is possible to use these sample points naturally generated by genetic process to construct approximations models. In the current state of art applications, during the optimization, the complicated and time-consuming objective function computation and analysis can be replaced by a surrogate model with much higher computational efficiency and acceptable accuracy. There are different approximation models. Most popular ones are Response Surface Methodology (RSM) [42], Kriging [43-44], and Radial Basis Function Neural Network [45-48]. In this model, due to being orders of magnitude cheaper to run, approximation models can be used in lieu of computationally

expensive solver codes during an evolutionary search. However, since the fitness function evaluations are performed using the approximate solutions, the performance of an evolutionary search depends on the success of approximation models. In our technique, which is different from the current surrogate model usage, an approximation model is utilized within the mutation steps to evaluate and control the mutated individuals. Correction steps can be done in accordance with error analysis. In some optimization problems there may be design target values. For example, in inverse design of an airfoil the target is pre-determined pressure coefficient distribution. By comparing the target and the current situation the error analysis can be done. This error analysis is used to correct a mutated individual. By using other artificial intelligent methods such as fuzzy logic we can correct the mutated individual.

A permission step is formed by an ordering process. After fitness function evaluations of mutated individuals they are sorted and put in order. Selected ones are allowed to take place into the next population. Control phases can be applied to global or local diversity. So, the algorithm may use random global diversity, random local diversity, controlled global diversity, and controlled local diversity. All these options can be applied in a single way or together. Another option is to start from local diversity and arrive in a globally controlled diversity. This option requires global search engines such as simulated annealing or particle swarm optimization methods. In the following figure each method is depicted. Here, $\mathcal{A}_{\text{func}}$ is an approximation function, $\mathcal{G}_{\text{search}}$ is global search engine, vmo_1 is the vibrational mutation operator given in Eq. (2.54), *and* vmo_2 is the vibrational mutation operator given in Eq. (2.55). Applications of these methods are in the next sections.

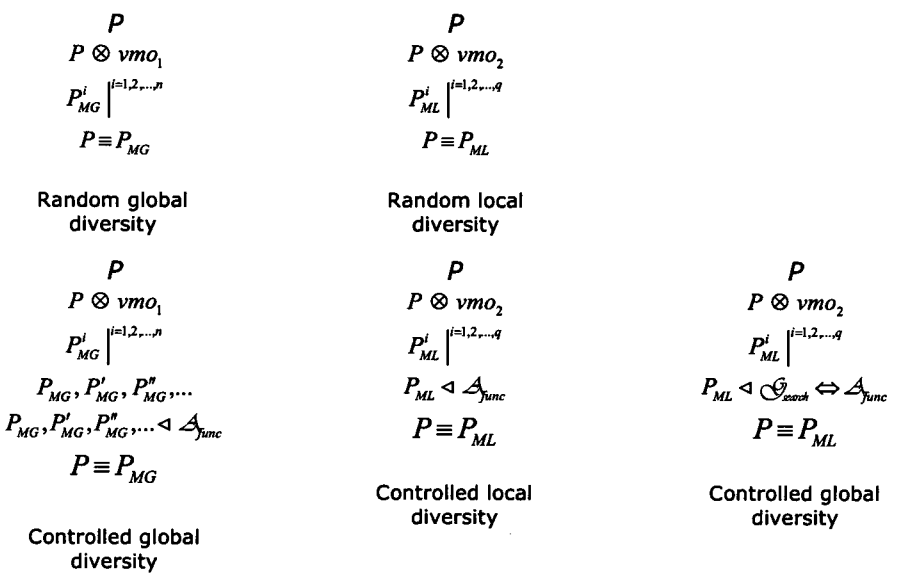

Figure 2.22 Qualification of diversity.

2.15 Quantification of Diversity in PSO

Similar to GA analysis we can describe three different diversity scales. These are the diversity of design variable, $D_{v,t}$; the diversity of particle, $D_{p,t}$; and the diversity of swarm, $D_{S,t}$ such as

$$D_{v,t}\Big|_{t=1,2,\ldots,T}^{v=1,2,\ldots,d} \tag{2.56}$$

$$D_{p,t}\Big|_{t=1,2,\ldots,T}^{p=1,2,\ldots,s} \tag{2.57}$$

$$D_{S,t}\Big|_{t=1,2,\ldots,T}^{S=1} \tag{2.58}$$

where t is the number of generation, T is the maximum number of generations, and S is the number of swarm which is typically 1 if there is only one population instead of a group. It is possible to determine each diversity type as a single value based on the general and selected positions in the swarm. This can be achieved by three different definitions: the difference between a member and the best member; the difference between a member and average member value; and the difference among the members. We will mainly focus on particle diversity and swarm diversity. We can categorize particle diversity into two types such as particle diversity and particle dependent diversity. Any particle diversity at each generation can be computed by using the following equation;

$$D_{p,t}\Big|_{t=1,2,\ldots,T}^{p=1,2,\ldots,s}=\left[\sum_{j=1}^{d}(x_{i,j}(t)-\overline{x}_j(t))^2\right]^{\frac{1}{2}} \tag{2.59}$$

$$\overline{x}_j(t)=\frac{1}{S}\sum_{i=1}^{s}x_{i,j}(t) \tag{2.60}$$

Particle dependent diversity is the diversity between two different particles in the swarm. Assume that x_k is the k^{th} particle and x_i is the i^{th} particle, then dependent diversity is given by

$$\begin{aligned} &D_{p,t}^{i,k}\Big|_{t=1,2,\ldots,T}=\left[\sum_{j=1}^{d}(x_{i,j}(t)-x_{k,j}(t))^2\right]^{\frac{1}{2}} \\ &D_{p,t}^{k,i}=D_{p,t}^{i,k} \end{aligned} \tag{2.61}$$

The swarm diversity may be constructed of the differences between the particles and the averaged one such as

$$D_{S,t}=\frac{1}{S}\sum_{i=1}^{s}\left[\sum_{j=1}^{d}(x_{i,j}(t)-\overline{x}_j(t))^2\right]^{1/2} \tag{2.62}$$

This description is a kind of averaged population diversity because we divide the total sum with the swarm size. We plan to use the diversity description given in Eq. (2.62) in the following analysis.

2.16 Mathematical Analysis of Basic PSO

Ozcan and Mohan [49] analyzed the updating Eq. (1.5) and (1.6) in an analytical way. Let's reissue the analytical analysis to get a further step. For the sake of simple mathematical analysis assume that $c_1.r_1$ and $c_2.r_2$ are constants which are equal to φ_1 and φ_2, respectively, and $\boldsymbol{P}_i\ (t)$, $\boldsymbol{P}_g\ (t)$ are also constants, such as p_i, and p_g, respectively. Then, Eq. (1.5) and (1.6) become,

$$v_{i,j}(t) = v_{i,j}(t-1) + \varphi_1(p_i - x_{i,j}(t-1)) + \varphi_2(p_g - x_{i,j}(t-1)) \tag{2.63}$$

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t) \tag{2.64}$$

by getting $x_{i,j}(t\text{-}1)$ using Eq. (2.64) and substituting it into Eq. (2.63) we get the following particle path equation

$$x_{i,j}(t) - (2 - \varphi_1 - \varphi_2)x_{i,j}(t-1) + x_{i,j}(t-2) = \varphi_1 p_i + \varphi_2 p_g \tag{2.65}$$

$$x - \alpha x' + x'' = \beta \tag{2.66}$$

where

$$\alpha = (2 - \varphi_1 - \varphi_2) \tag{2.67}$$

$$\beta = \varphi_1 p_i + \varphi_2 p_g \tag{2.68}$$

This is a linear non-homogeneous second order differential equation. The general solution of Eq. (2.65) can be derived by getting complementary and particular solutions. The resulted solutions including initial boundary conditions can be found such that

$$\xi_{i,j}(\tau) = \chi\tau^t + \varepsilon\phi^t + \gamma \tag{2.69}$$

where

$$\chi = (0.5 - \frac{(\varphi_1 + \varphi_2)}{2\eta})x_{i,j}^{(0)} + \frac{v_{i,j}^{(0)}}{\eta} + \frac{\beta}{2\eta} - \frac{\beta}{2(\varphi_1 + \varphi_2)} \tag{2.70}$$

$$\varepsilon = (0.5 + \frac{(\varphi_1 + \varphi_2)}{2\eta})x_{i,j}^{(0)} - \frac{v_{i,j}^{(0)}}{\eta} - \frac{\beta}{2\eta} + \frac{\beta}{2(\varphi_1 + \varphi_2)} \tag{2.71}$$

$$\tau = (\alpha + \eta)/2 \tag{2.72}$$

$$\phi = (\alpha - \eta)/2 \tag{2.73}$$

$$\eta = \sqrt{\alpha^2 - 4} \tag{2.74}$$

$$\gamma = \beta/(\varphi_1 + \varphi_2) \tag{2.75}$$

This result may expose several different cases depending on α value, specifically the sum of φ_1 and φ_2 values. Value of η becomes a complex number for $0 < \varphi_1 + \varphi_2 < 4$. Converting Φ

and τ into polar forms and substituting them into Eq. (2.69) we can obtain the simplified following equations;

$$x_{i,j}(t) = \kappa \sin(\theta t) + \lambda \cos(\theta t) + \gamma \tag{2.76}$$

$$\theta = \tan^{-1}(\|\eta\|/|\alpha|) \tag{2.77}$$

$$\kappa = (2v_{i,j}^{(0)} - (\varphi_1 + \varphi_2)x_{i,j}^{(0)} + \beta)/\|\eta\| \tag{2.78}$$

$$\lambda = x_{i,j}^{(0)} - \gamma \tag{2.79}$$

This is the general form for the particle path. We can make some assumptions such that there is only one dimension, $p=p_i= p_g$, $\varphi_1 + \varphi_2= \varphi$ and $2>\varphi>0$ or $4> \varphi>2$, $x^{(0)}=x_0$, $v^{(0)}=v_0$. Then the path of any particle in the swarm is directed by the following equations;

$$x(t) = A_1 \sin(\theta t) + A_2 \cos(\theta t) \tag{2.80}$$

$$A_1 = \frac{(2v_0 - \phi x_0 - \phi p)}{\sqrt{|\phi^2 - 4\phi|}} \tag{2.81}$$

$$A_2 = (x_0 - p) \tag{2.82}$$

$$\theta = arc\tan(\sqrt{|\phi^2 - 4\phi|}/|2-\phi|) \tag{2.83}$$

As Clerc and Kennedy [50] concluded, the particle as seen in discrete time "surfs" on an underlying continuous foundation of sine waves and each time it jumps from one wave to another by using random numbers in periphery of a random number distribution. At this point, let's redefined the Kronecker delta function such that;

$$x(t) = \begin{Bmatrix} x(t) + A_3\delta & \delta = 1 \quad if\ t > nf, n = 1,2,... \\ x(t) & \delta = 0 \quad if\ t < nf, n = 1,2,... \end{Bmatrix} \tag{2.84}$$

here f is to be the periodicity to make an impulse, and A_3 is the third amplitude which is possibly negative or positive. The resulting graphs for Eq. (2.80) and Eq. (2.84) with positive A_3 can be plotted in Fig. 2.23. The purpose of adding an impulse depending on the time frame is to catch the big wave, or let's say to extend the search domain from one local area to another local area for the particle. The effect of an impulse will be analyzed in further sections.

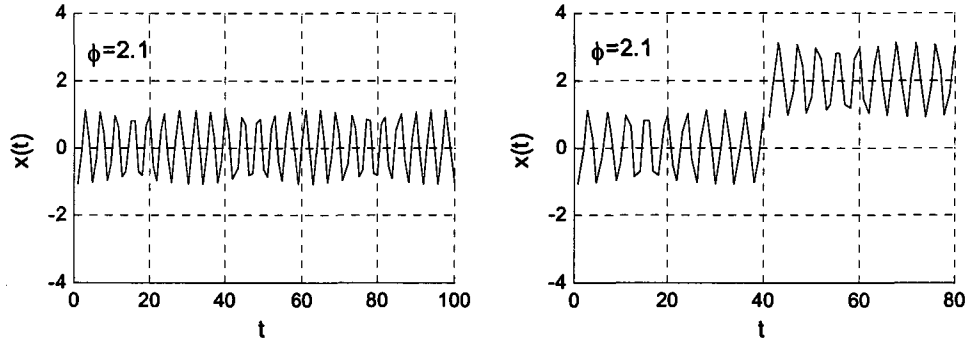

Figure 2.23 Trajectory of a particle in a simplified PSO algorithm

2.17 Improved PSO Algorithms

When a new algorithm is proposed into literature it is expected to be compared with the current state of art algorithms on commonly used benchmark test functions. Obviously it is difficult to make a fair comparison because each algorithm may have peculiar tuning which results in important differences on the cost functions. However well-defined and straightforward algorithms may provide relatively good inference. Three well known PSO algorithms are selected as comparative optimization algorithms. These are constriction factor PSO (c-PSO), linearly decreased inertial weight PSO (w-PSO), and Gaussian mutation based PSO (g-PSO).

c-PSO

The particle swarm with a constriction factor is introduced by Clerc [50], which has investigated the use of a parameter called the constriction factor. With the constriction factor K, the particle velocity and position dimensions are updated via:

$$v_{i,j}(t) = K[v_{i,j}(t-1) + c_1 r_1 (P_i(t-1) - x_{i,j}(t-1)) + c_2 r_2 (P_g(t-1) - x_{i,j}(t-1))] \tag{2.85}$$

$$K = \frac{2}{\left|2 - \psi - \sqrt{\psi^2 - 4\psi}\right|}, \psi = c_1 + c_2, \psi > 4$$
$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t) \tag{2.86}$$

A particularly important contribution of this factor is that if it is correctly chosen, it guarantees the stability of PSO without the need to bind the velocities. Typically values of *2.05* are used for c_1 and c_2, making ψ is equal to *4.1* and K is equal to *0.729*.

w-PSO

Shi and Eberhart [51] introduced the idea of a time-varying inertia weight. The idea was based on the control of the diversification and intensification behavior of the algorithm. The velocity is updated in accordance with the following expression;

$$\begin{aligned} v_{i,j}(t) &= w(t) v_{i,j}(t-1) + c_1 r_1 (P_i(t-1) - x_{i,j}(t-1)) + c_2 r_2 (P_g(t-1) - x_{i,j}(t-1)) \\ x_{i,j}(t) &= x_{i,j}(t-1) + v_{i,j}(t) \end{aligned} \tag{2.87}$$

The inertia weight, w, is decreased linearly starting from one point and ending to another point related to maximum iteration number, G. Normally the starting value of the inertia weight is set to *0.9* and the final to *0.4*. However we tuned them to [0.6, 0.2] range for better performance.

g-PSO

First Gaussian mutation based PSO algorithm is introduced by Higashi and Iba [27], which use a mutation operator that changes a particle dimension value using a random number drawn from a Gaussian distribution. A particle is selected for mutation using a mutation rate that is linearly decreased during a run. However it is improved by several researchers. Pant *et al.* [52] developed Gaussian mutation operator application technique for updating the position of the swarm particles. The mutation operator is activated only when the diversity of the swarm becomes less than a certain threshold, d_{low}. The velocity is updated via Eq. (2.87) in addition to the following criteria;

$$if \;\; d < d_{low} \;\; then \;\; x_{i,j}(t) = x_{i,j}(t-1) + \xi.rand \tag{2.88}$$

where ξ is the scaling factor, *rand* is a random number generated by Gaussian distribution. The diversity description is based on the differences between the average dimensional position and the current dimensional particle positions.

2.18 Quantification in a Simple Test Case

It may be beneficial to see the change of swarm diversity for better understanding of algorithm characters. For this reason a simple test case is constructed. One of the benchmark test functions, *Ackley* test function is selected as a test case and selected PSO algorithms are compared in terms of generation and swarm diversity values. Ackley test function is multimodal test function and it is described as the following

$$-20\exp\left[-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d}x_j^2}\right]-\exp\left[\frac{1}{d}\sum_{j=1}^{d}\cos(2\pi x_j)\right]+20+e \tag{2.89}$$

The search range and optimal values; x^*, $f(x^*)$ are given in the following

search range	x^*	$f(x^*)$
$[-30, 30]^D$	[0,0,…,0]	0

The problem dimension, d is selected as 3, the swarm size, s is selected as 5, and the maximum generation, T is fixed as 200. The experimental set up for c-PSO, w-PSO, and g-PSO are given in Table 2.16. All algorithms are run 40 times and the results are averaged. The resulted plots can be seen in Fig. 2.24. We can see averaged best (elite) f_{min} values versus generations on the left side and the swarm diversity values versus generations on the right side.

Table 2.16 The features of PSO algorithms

Algorithm	w	c_1	c_2	d_{low}	ξ
c-PSO	-	2.05	2.05	-	-
w-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	-	-
g-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	10^{-3}	6

Fig. 2.24 shows that g-PSO outperforms the other ones. At the end of maximum generation g-PSO reaches about the level of 10^{-4} for objective function values. However the other two algorithms are about the level of 10^{-1} for objective function values. It seems that threshold diversity based mutation applications decreased 40 percent of the required generation number. On the other hand, the swarm diversity values have different characteristics. Both constriction factor based PSO and linearly decreased inertia weight based PSO have linearly decreased swarm diversity in logarithmic scale of objective function values. w-PSO shows higher level in diversity than c-PSO. However, g-PSO has a completely different characteristic in diversity changes. The algorithm decreases the swarm diversity until a certain generation number; then, the mutation operator takes a role and keeps the swarm diversity in a fixed level. Interestingly, the swarm diversity changes in a periodic manner. It seems that the update rules of PSO always decrease the swarm diversity in a linear fashion in logarithmic scale. It implies that the swarm tries to make similar each particle to each other during the generations without outer disturbance. This characteristic behavior may significantly cause premature convergence within the swarm.

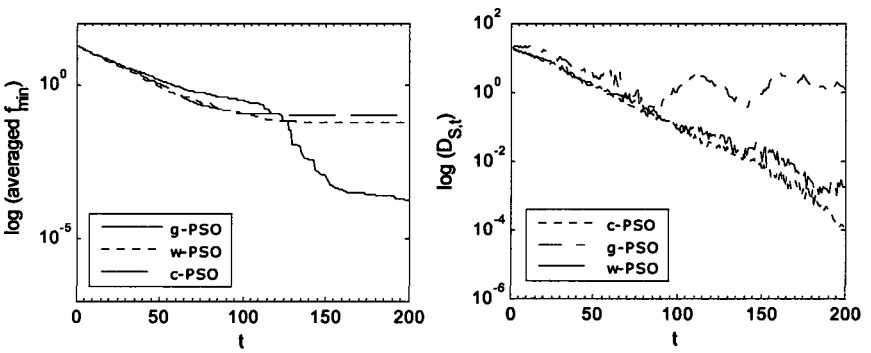

Figure 2.24 The comparative results of selected PSO algorithms.

2.19 Signal Analysis

It may be beneficial to have a close look at swarm diversity changes in g-PSO. The swarm diversity change is a kind of signal based on time series. The wavelet analysis is applied to the same optimization problem with T=1000. The whole diversity signal can be seen in Fig. 2.25

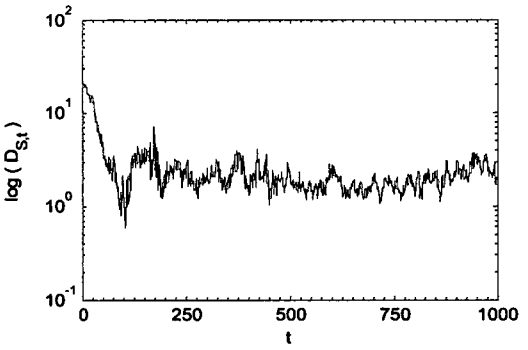

Fig. 2.25 The swarm diversity signal in g-PSO optimization process

The diversity change in terms of the last 800 generations is included for steady result. So, the diversity signal is expressed by the following

$$D_{s,t} = A_9 + \sum_{j=1}^{9} D_j \tag{2.90}$$

where A_j is the approximate signal and D_j are the detail signals. As expressed in the previous chapter, the biggest part of the signal energy is carried by an approximate signal. The remaining energy is shared by detail signals sometimes called noise signals. There are different wavelet functions in literature. In our analysis we preferred to use Meyer wavelets at 9-level. The resulted discrete wavelet transform for the diversity can be seen in Fig. 2.26. The figure includes only approximation signal A_9 and detail signals D_j, j; 6, 7, 8, and 9. The remaining signals are ignored due to noisy natures. The picture shows that the main diversity signal caries some locally periodic signals and noises.

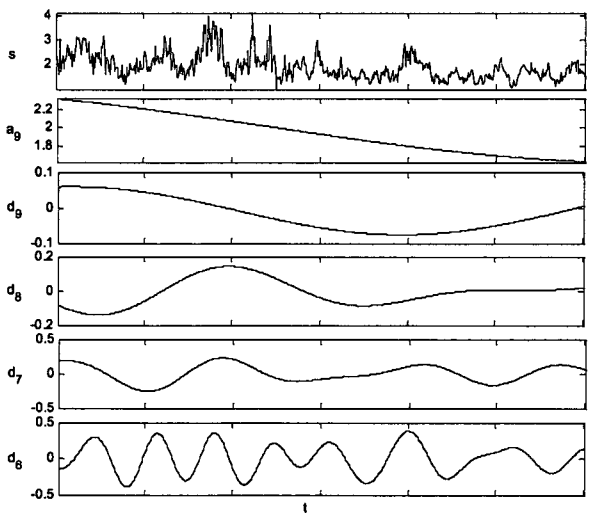

Figure 2.26 Wavelet analysis of a part of swarm diversity signal

Wavelet analysis showed that the description of threshold diversity brings new parameters such as periodicity and band width; let's say a kind of amplitude. Mutation activation dependent on diversity may be beneficial due to high and guarantied diversification. However, finding a proper threshold value may not be easy and may take extra study. Additionally, fixed diversity rate may not be proper for more accurate convergence in different test functions. Instead of using threshold diversity, we may analyze the periodicity and amplitude features in terms of a new mutation strategy which includes both of them.

2.20 Periodic Mutation Activation

At the beginning of the PSO process we determine the mutation probability, P_m and then naturally the number of mutated individuals in a regular manner. This probability is fixed at each generation. Instead of this approach we can describe the same probability such as

$$P_m = \begin{bmatrix} 1 \; if \; t = nf \\ 0 \; if \; t \neq nf \end{bmatrix} \Bigg|^{n=1,2,3,\ldots} \tag{2.91}$$

where f is the frequency, let's say mutation application frequency, n is the integer number. This strategy means that all particles within the swarm are going to be mutated in every f generations, in other generations all particles are generated by only update operations. We put an attention that we did not describe any mutation operator, we just describe the mutation application strategy. Now, we can describe a new mutation operator which has an amplitude factor.

2.21 Vibrational PSO Algorithm: v-PSO

The traditional general form of the mutation which was applied in the previous g-PSO algorithm can be written as $x_{i,j}(t) = g(x_{i,j}(t))$; where g is the mutation operator providing the offspring vector. Instead of this strict form of a mutation operator it can be described including mutation strategy as

$$x_{i,j}(t) = \mathcal{F}(g(x_{i,j}(t)), f) \tag{2.92}$$

where F is the generalized mutation function, f is a user defined frequency. Right after update applications, in every f^{-1} period of the generations applying the mutation operator to all particle dimensions of the whole swarm, particles in the population spread throughout the design space. Mutation operator is given by

$$x_{i,j}(t) = x_{i,j}(t)[1 + A.(0.5 - rand).\delta],\ {}^{j=1,2,\ldots,d}_{i=1,2,\ldots,s}\ and\ \delta = \begin{cases} 1 & if\ t = nf, n = 1,2,\ldots \\ 0 & if\ t \neq nf \end{cases} \qquad (2.93)$$

where A is a user defined scale factor called an amplitude and it may be selected as a fixed number or computed during the generations, *rand* is a real random number specified by random number generator in accordance with $N[0, 1]$. In the applications Gaussian probability density function is used. However other density functions can also be used in Eq. (2.93). The aim of designing such a mutation strategy is to catch the big wave for escaping from local traps and getting the correct search pattern. We need to point out that Eq. (2.93) resembles Eq. (2.84). The velocity and the positions are updated via Eq. (2.87) except the generations corresponding to the mutation period. This new algorithm is named as vibrational PSO (v-PSO).

2.22 Comparison of Algorithms with a Simple Case

New algorithm is compared with the previous algorithms in the same test function case. In addition to Table 2.16 the features of v-PSO is given in Table 2.17. The results are depicted in Fig. 2.27. On the left side of the figure the averaged best objective function values versus generations are shown. Vibrational PSO outperforms the other algorithms. It decreases the required generations 50 percent as compared with c-PSO and w-PSO, and 25 percent as compared with g-PSO. Additionally the accuracy of the solution is about 10^{-7} level and it is almost twice more accurate than g-PSO. On the right side of the figure the swarm diversity versus generations are observed. The swarm diversity of v-PSO is decreased during the generations; however its change is fluctuated and periodic due to periodic mutation applications. The amplitude of change is also changed.

Table 2.17 The features of v-PSO algorithm

Algorithm	W	c_1	c_2	f	A
v-PSO	0.05	2	1.5	25	1

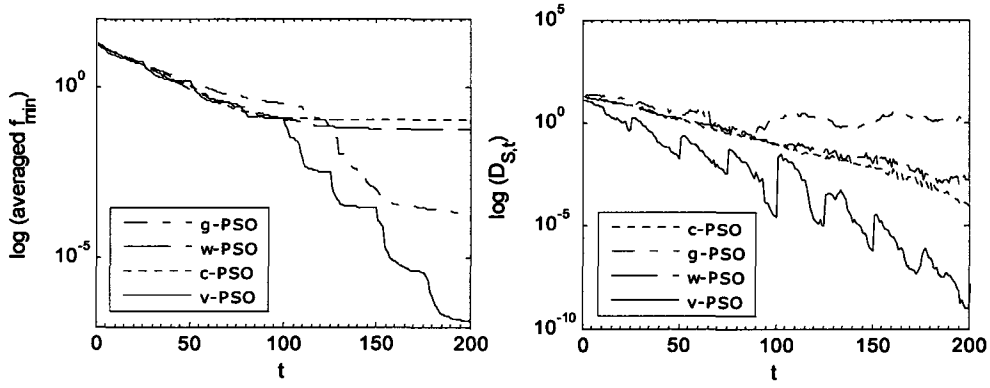

Figure 2.27 The comparative results of PSO algorithms.

Fig. 2.27 is also showed that the performance of the algorithm is directly related to the diversity. However, this relation is not subject to the quantification, it is subject to the qualification. Therefore, we need to analyze the qualification of the diversity.

2.23 Qualification of Diversity

Up to now, we analyzed the quantification of diversity within the swarm with different swarm algorithms including a new mutation strategy. It seems that diversification is useful and required for fast and more accurate convergence. However, high level diversity does not bring better performance. Better performance is related to the quality of diversity. We think that the quality of diversity can be categorized depending on the distribution character in search space and/or the determination process. It can be global or local diversity. It can be random or controlled diversity. At first, we will focus on the distribution character of the diversity in terms of global or local diversity.

Global and Local Diversity

If the mutation applications cause a wider range of distribution of the current swarm in search space it can be called global diversity. Suppose that the diversity of the current swarm is described as $D_S(t)$, and the new diversity after mutation applications is called as $D_S(t+1)$. We describe that if

$$D_S(t+1) >> D_S(t) \tag{2.94}$$

is valid, it means that the mutation applications provide a global diversity in the swarm. Local diversity concept can be based on a particle and its neighborhood. Assume that we have two particles labeled as x_m and x_k in the swarm. The particle diversities for these particles are named as $D^m_p(t)$ and $D^k_p(t)$. Let's describe local difference in diversity, ΔD_L. If the following criteria

$$D_p^m(t) - D_p^k(t) < \Delta D_L \tag{2.95}$$

$$D_{p,t}^{m,k} \Rightarrow \Delta D_L \tag{2.96}$$

are valid for the particles; then, we can say that there is a local diversity within the population. During the optimization process local diversity is going to be common phenomenon within the swarm. This situation may results from local or global convergence. Therefore, it may or may not be beneficial to the population.

Periodic Localization

We can intentionally generate local diversity by using additional vibrational mutation operator for a better local search. The application can be done in accordance with the following equation;

$$x_{i,j}(t) = x_j^e(t)[1 + A'.(0.5 - rand).\delta], \; {}^{j=1,2,\ldots,d}_{i=1,2,\ldots,q} \; and \; \delta = \begin{Bmatrix} 1 & if \; t = nf', n = 1,2,\ldots \\ 0 & if \; t \neq nf' \end{Bmatrix} \qquad (2.97)$$

where A' is a user defined scale factor called an amplitude, f is the application frequency, the base vector x^e is the global best called elite particle of the swarm, and q is the number of new individuals locally generated by new vibrational mutation operator.

An example periodic localization for the previous test case can be seen in Fig. 2.28. In this application; s is taken as 10, T is taken as 200, and q is taken as 2. The other features of the algorithm are given in Table 2.18. In the figure the particle diversities versus generations are depicted. The solid lines belong to elite based particles generated by using vibrational mutation operator. The dotted lines are the other particles' diversities in the swarm. On the right side the generations between 25 and 50 are zoomed. It is clearly observed that the elite based particles are closer to each other than the other particles.

Table 2.18 The features of v-PSO algorithm

Algorithm	w	C_1	c_2	f	A
v-PSO	0.05	2	1.5	15	0.5

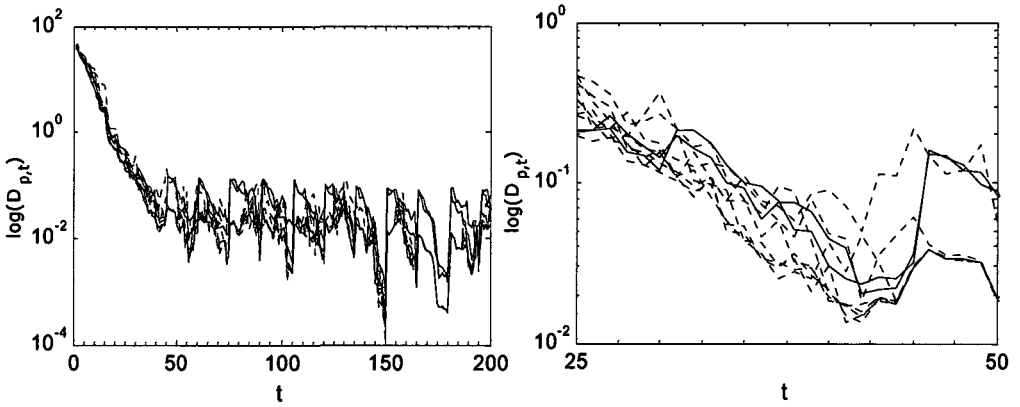

Figure 2.28 Local diversity among the particle diversities

Multi-frequency Vibrational Mutations

We need to point out that Eq. (2.93) is different from Eq. (2.97). The first one is global; the second one is locally applied. It is possible to apply both mutation operators by selecting different frequencies and amplitudes. We can describe this situation such as

$$x_{i,j}(t) = x_{i,j}(t)[1 + A_1 .(0.5 - rand).\delta],\ {}^{j=1,2,\ldots,d}_{i=1,2,\ldots,s}\ and\ \delta = \begin{cases} 1 & if\ t = nf_1, n = 1,2,\ldots \\ 0 & if\ t \neq nf_1 \end{cases} \quad (2.98)$$

$$x_{i,j}(t) = x^e_j(t)[1 + A_2 .(0.5 - rand).\delta],\ {}^{j=1,2,\ldots,d}_{i=1,2,\ldots,q}\ and\ \delta = \begin{cases} 1 & if\ t = nf_2, n = 1,2,\ldots \\ 0 & if\ t \neq nf_2 \end{cases} \quad (2.99)$$

We can figure the swarm in terms of t generations. Assume that f_1 is equal to 5, f_2 is equal to 3. For the first seven generation includes the following swarm combinations;

t	1	2	3	4	5	6	7
S	$S_I^i \Big\vert^{i=1,2,\ldots,s}$	$S_U^i \Big\vert^{i=1,2,\ldots,s}$	$S_U^i \Big\vert^{i=1,2,\ldots,s-q}$ $S_{ML}^i \Big\vert^{i=1,2,\ldots,q}$	$S_U^i \Big\vert^{i=1,2,\ldots,s}$	$S_{MG}^i \Big\vert^{i=1,2,\ldots,s}$	$S_U^i \Big\vert^{i=1,2,\ldots,s-q}$ $S_{ML}^i \Big\vert^{i=1,2,\ldots,q}$	$S_U^i \Big\vert^{i=1,2,\ldots,s}$

where S_I is the initial swarm, S_U is the swarm generated by update operations, S_{ML} is the swarm locally generated by the second mutation operator, S_{MG} is the swarm globally generated by the first mutation operator. At first, the swarm is composed of randomly generated initial particles. In the second generation the swarm only includes the particles generated by update operations described in Eq. (2.87). The third generation corresponds to the second frequency; $t \leftarrow nf_2$, $n=1$. In the 3rd generation, the swarm includes *(s-q)* particles generated by update operations and q particles generated by the second vibrational mutation operator. In the 4th generation the swarm only includes the particles generated by update operations. The fifth generation corresponds to the first frequency; $t \leftarrow nf_1$, $n=1$. In the 5th generation the swarm only includes the particles generated by the first vibrational mutation operator. The sixth generation corresponds to the second frequency; $t \leftarrow nf_2$, $n=2$. In the 6th generation, the swarm includes *(s-q)* particles generated by updating operations and q particles generated by the second vibrational mutation operator. In the 7th generation the swarm only includes the particles generated by update operations. This periodic process goes on until the stopping criterion is satisfied.

Random and Controlled Diversity

Vibrational mutation operators generate new particles and these new particles are absolutely random based estimations for better solutions. Application of the first mutation operator provides global but random diversity within the swarm. Application of the second mutation operator causes local but random diversity surrounded by elite individual neighborhood in the swarm. However, we may improve the quality of estimations generated by vibrational mutation operators. We can call this secondary process as filtration or *controlled* diversity. Control action on mutated individuals before placing them into the next swarm can be done by using artificial intelligent methods. There can be two possible options for control process. These options are called *correction* and *permission* steps. In the correction process the input as a current particle situation is corrected in accordance with

evaluation results. Therefore, the current particle situation is going to be changed. After the correction phase, the corrected input, let's say mutated particle, is placed into next swarm. In the permission process the input is evaluated and then the decision is made if the input is going to be placed into next swarm or not. The second process is simpler than the previous one. Therefore, we focus on permission step. However, permission requires multiple inputs for ordering and placing.

It is a clear fact that PSO is a population-based algorithm. In each generation the algorithm produces design input parameters and resulted outputs based on objective function computations. These couples are also sample points for the design space. Normally, sample points are selected in accordance with design of experiments (DoE) methods. But PSO process produces solution examples instead of sample points. As a result, after each generation we had growing sample points. It is possible to use these sample points naturally generated by PSO process to construct approximation models. In current state of the art applications, during the optimization, the complicated and time-consuming objective function computation and analysis can be replaced by a surrogate model with much higher computational efficiency and acceptable accuracy. There are different approximation models. The most popular ones are RSM, Kriging, and Radial Basis Function Neural Network. However, since the objective function evaluations are performed using the approximate solutions, the performances of evolutionary search depend on the success of approximation models. In our technique, which is different from the current surrogate model usage, the approximation model is utilized within the mutation step to evaluate and control the mutated individuals. The permission step is formed by ordering process. The best ones are allowed to take place into next swarm.

In the test case the Matlab function *newrb* is used to generate neural networks. In a set of training data, the input parameters are the position points of particles in the swarm, while the output parameters are their objective function values to be improved.

RBF NN in controlled diversity

Control process can be applied in global or local diversity applications. Therefore, we have four options including global controlled diversity, local controlled diversity, global random diversity, and local random diversity. In our applications we prefer to use global random diversity and local controlled diversity. The steps of neural network coupled v-PSO for optimization are the following; firstly, cost function calculations in the current swarm are performed to get their objective function values. Secondly, networks are trained after the threshold generation point by using the particles in the current and previous swarms and their

cost function values. However the training set is limited to a certain number. For this training, particles in the swarms are used as the input and their cost function values are used as the output. Then, the mutations according to Eq. (2.99) are applied and temporal neural network swarm including T_s particles is generated. After neural network population generation, by using trained neural nets, the cost function values of temporal network swarm are estimated and some of them (for example the best q of T_s) which have smaller cost function values than the global best particle has in the current swarm are randomly placed in the new swarm to be used as candidates at the next step of the algorithm as follows

$$\begin{aligned} f^{NN} &= \mathcal{N}_{func}(T_s) \\ \left[f^{NN}\ order\right] &= sort(f^{NN}) \\ x_k(t) &= T_s^{order(i)},\ \substack{k=rand[1-s] \\ i=1,2,\ldots,q} \end{aligned} \tag{2.100}$$

Similar to GA applications there are some important points which require careful tuning during the online use of neural network model in v-PSO. These are: the number of generations required for training the neural network and the selection of generations to be used for training.

3. APPLICATIONS AND RESULTS

In this section, we applied new algorithm improvements to different problems in order to check their efficiencies. These problems are mostly selected from aerospace engineering. However, the solution algorithms improved in this dissertation are also applicable to other engineering disciplines. The selected problems are as follows;

- Effect of surface parameterization on VGA in 2D airfoil design.
- Inverse design of 2D airfoil in subsonic flow conditions.
- Aerodynamic optimization of 2D airfoil in transonic flow regime.
- Path planning of autonomous UAV in 3D terrain environments.
- Radar cross section minimization of 3D air vehicle for simple and complex shapes.
- Optimization of parameters for benchmark test functions.

3.1 Effect of Surface Parameterization on VGA in 2D Airfoil Design

In this study, two different curve representation methods; Parsec and Bezier representation methods are tested via vibrational genetic algorithm [VGA] to understand the effect of representation method on the type of optimization process in 2D airfoil design.

Introduction

The main goal in aerodynamics is commonly to increase the efficiency like the ratio of lift over drag. Aircraft wings are the primary subject to optimization efforts. Airfoils are the basic elements of wing geometry; they determine a large share of wing flow phenomena though they are just 2D sections of the physical wing surface. Given a designer's refined knowledge about the occurring flow phenomena, his goal may be to obtain certain pressure distributions on wing surfaces: This may be reached by inverse approaches with a shape resulting from the effort, or by applying optimization strategies to drive results toward ideal values [53]. In an aerodynamic optimization problem the optimum design is an unknown shape, and the performance of the optimization process depends on how well the geometry representation method can approach the optimum shape. To pose the airfoil shape optimization problem, the design variables that control the geometrical shape of airfoil are needed. The goal is to propose functions with a minimum set of input parameters for shape variation, function structure and their parameters chosen to address special aerodynamic or fluid mechanic phenomena. However, a method with more design parameters should have a more complete set of shapes, and therefore can approach the design target better [54].

There are different functions to describe airfoil sections. In addition to well known airfoil descriptions the aircraft industry has also developed their own mathematical tools to

shape specific wing and blade sections. Among these geometrical representations it is possible to make a categorization depending on the type of the mathematical functions. The classification can be divided into three categories; polynomial function based representation methods, sinusoidal function based representation methods, and others. Bezier functions [55], NURBS [56], Parsec method based functions [57], and NACA 4- and 5-digit series functions [58] are well known polynomial type representation methods of airfoil sections. B-spline functions [59] are also other well known type of polynomial functions. Mathematically, any continuous function defined on a closed interval can be represented by an infinite series of normal modes which form a complete set of bases. The set of Fourier sine functions is an example of such a complete set. There are several well known shape functions for wing section modifications. Hicks-Henne functions, Wagner functions, and aerofunctions are some of them [60]. Joukowski transformation, mesh point and grid parameterization methods are given examples of the category of other methods [61]. In mesh pint method for a numerical computation, mesh points are used to represent the airfoil surface. The mesh-point method uses the coordinates of these points directly as the design parameters.

Different representation methods may have different performances depending on the optimization algorithms. There are some studies which include the comparisons among representation methods related to gradient based optimization algorithms [62, 63]. Both studies showed that the mesh-point method can reach the highest accuracy among other methods such as Parsec, Hicks-Henne, B-Spline, and the Class function / Shape function Transformation. This is an expected result; because the accuracy is proportional to the number of design parameters. More design parameters usually mean a more complete design space and hence a better capability of approaching the design target. More design parameters mean also more sensitivity to local perturbations. Gradient based optimization algorithms are mainly based on local sensitivities related to perturbations [64]. On the other hand, conventional gradient-based algorithms may be ineffective in some optimization problems with non-differentiable, highly nonlinear, and many local minima cost functions because of local minimums or the difficulty in calculating gradients. Search methods that require no gradient information and can achieve a global optimal solution, offer considerable advantages in solving these difficult optimization problems [65]. These advantages are robustness, suitability to parallel computing, and simplicity. Owing to these advantages over the analytical methods, especially genetic algorithms have become increasingly popular in a broad class of design problems [66]. Although there are some successful applications to compute optimal solutions for aerodynamic problems, sometimes, premature or slow rate convergence may prevent GAs from reaching global optimal solutions. This may be directly because of the applied optimization algorithm itself or a selected representation method.

There are some studies [67, 68] which include global optimization methods to analyze the accuracy of some parameterizations such as B-spline and basis function approaches or Parsec method and its derivations. In this study, two different curve representation methods, Parsec and Bezier representation methods, are tested via a vibrational genetic algorithm to show the effect of representation method on search type optimization process in 2D airfoil design. At first, representation methods are tested in low speed flow conditions within the inverse design problem and then the same representation methods are tested in transonic flow conditions within the optimization problem. For both cases a vibrational genetic algorithm is used as an optimization tool.

Surface Parameterization

Bezier and Parsec representation methods are polynomial function based representation methods. Although there are some similarities between Bezier and Parsec methodologies, these curves are different in nature. A selected 2D curve, an airfoil can be represented by a Bezier curve representation with a set of *(m+1)* control points. Its expression is given by the following equations

$$x(t) = \sum_{i=0}^{m} C_m^i \, t^i \, (1-t)^{m-i} \, x_i \tag{3.1}$$

$$y(t) = \sum_{i=0}^{m} C_m^i \, t^i \, (1-t)^{m-i} \, y_i \tag{3.2}$$

$$C_m^i = \frac{m!}{i!\,(m-i)!} \tag{3.3}$$

where t is the parameter of the curve whose values vary uniformly between [0-1]. The (x_i, y_i) are the coordinates of the control points (c. p.) which define the airfoil points $(x(t), y(t))$. The two control points (0,0) and (1,0) at the leading and trailing edges are fixed. It is commonly considered that the x_i control points are kept fixed, and the parameters coded in the genetic algorithm are only the y_i coordinates of the control points. Fig. 3.1 shows the Bezier curve representation of an example airfoil.

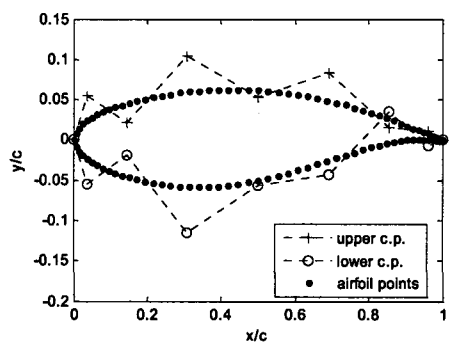

Figure 3.1 An airfoil in Bezier form.

An airfoil family, "Parsec," has been proposed to parameterize an airfoil shape in a different way but familiar to geometry. A remarkable point is that this technique has been developed to control important aerodynamic features effectively by selecting the design parameters based on the knowledge of flows around airfoil. The Parsec-11 basic set parameterizes upper and lower airfoil surfaces using polynomials in coordinates x, z as

$$z = \sum_{n=1}^{6} a_n x^{n-1/2} \tag{3.4}$$

where a_n are real coefficients. Instead of taking these coefficients as design parameters, the Parsec airfoils are defined by basic geometric parameters; leading edge radius, upper and lower crest location including curvatures, trailing edge ordinate, thickness, direction, and wedge angles as shown in Fig. 3.2. The real coefficients, a_n, are computed by solving simple simultaneous equations related to each design parameter. An algebraic equation system can be written in accordance with design parameters and solved to get coefficients for each curve. For the upper surface of the airfoil, the design parameters can be related to the polynomials as

$$\begin{aligned}
z_{TE} &= z(1) = \sum_{n=1}^{6} a_n \\
z_{UP} &= z(x_{up}) = \sum_{n=1}^{6} a_n x_{up}^{n-1/2} \\
\tan(\alpha_{TE}) &= z'(1) = \sum_{n=1}^{6} \frac{2n-1}{2} a_n \\
z'(x_{up}) &= \sum_{n=1}^{6} \frac{2n-1}{2} a_n x_{up}^{n-3/2} \\
z_{XXUP} &= z''(x_{up}) = \sum_{n=1}^{6} \frac{2n-1}{2}\frac{2n-3}{2} a_n x_{up}^{n-5/2} \\
r &= \frac{1}{2} a_1^2
\end{aligned} \tag{3.5}$$

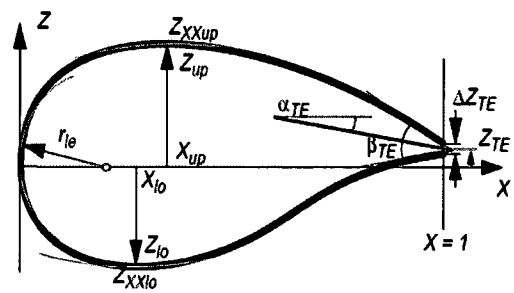

Figure 3.2 Design parameters for the Parsec airfoil.

The similar equations are valid for the lower surface of the airfoil. By using these equations, the following algebraic equation system can be written

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_{up}^{1/2} & x_{up}^{3/2} & x_{up}^{5/2} & x_{up}^{7/2} & x_{up}^{9/2} & x_{up}^{11/2} \\ 1/2 & 3/2 & 5/2 & 7/2 & 9/2 & 11/2 \\ \frac{1}{2}x_{up}^{-1/2} & \frac{3}{2}x_{up}^{1/2} & \frac{5}{2}x_{up}^{3/2} & \frac{7}{2}x_{up}^{5/2} & \frac{9}{2}x_{up}^{7/2} & \frac{11}{2}x_{up}^{9/2} \\ -\frac{1}{4}x_{up}^{-3/2} & \frac{3}{4}x_{up}^{-1/2} & \frac{15}{4}x_{up}^{1/2} & \frac{35}{4}x_{up}^{3/2} & \frac{63}{4}x_{up}^{5/2} & \frac{99}{4}x_{up}^{7/2} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} z_{TE} \\ z_{UP} \\ \tan(\alpha_{TE}) \\ 0 \\ z_{XXUP} \\ \sqrt{2r} \end{bmatrix} \tag{3.6}$$

Parsec and Bezier curve representation methods are still popular fashions although there are some criticized points related to Bezier or Parsec. In each method some improvements are implemented to basic descriptions and compensated inefficient sides of each method.

CFD Solvers

Once an aerodynamic shape representation is defined, a numerical optimization method is coupled with a suitable flow analysis tool (flow solver). Two types of CFD solvers are used in this study. These solvers are 2D vortex-panel solver for incompressible, inviscid, subsonic flows and Euler equations solver for inviscid, compressible, transonic flows. Panel method based solver is used in inverse design problems. The other one is used in an optimization problem. The Euler equations solver program uses elliptic partial differential equation solution method to generate structural grids around the airfoil. The produced grid structure example around NACA 4-digit airfoil is given in Fig. 3.3. Within the program, the flux values are calculated by using a cell-centered finite volume space discretization method on a structured O-mesh and Roe flux difference splitting method. The steady state solution is reached by pseudo-time marching the Euler equations using an explicit six-stage Runge-Kutta scheme.

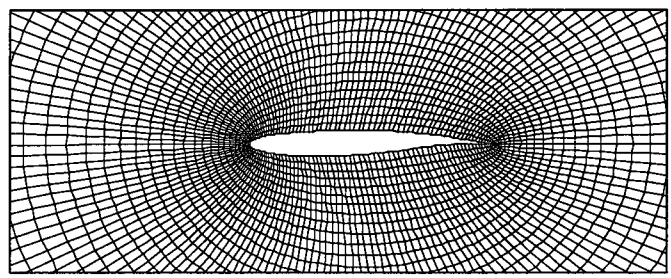

Figure 3.3 Grid structure for NACA 4412

Case Studies and Results

In the first case, representation methods are tested in low speed flow conditions within the inverse design problem. In the second case, the same representation methods are tested in transonic flow conditions within the optimization problem. For both cases a vibrational genetic algorithm (VGA) is used as an optimization tool.

Case 1

Initial populations for both representation methods are generated by using random number operator based on NACA-0012 symmetric airfoil. Bezier curves for upper and lover airfoil lines are governed by 26 control points. Four control points of them are known points such as leading edge and trailing edge. Therefore, the total unknown number of control points for an airfoil is 22. The sample initial population for Bezier curves is given in Fig. 3.4. Parsec curves are totally directed by 10 parameters. The y coordinate and thickness of trailing edge are fixed as zero. The sample initial population for Parsec curves is given in Fig. 3.4.

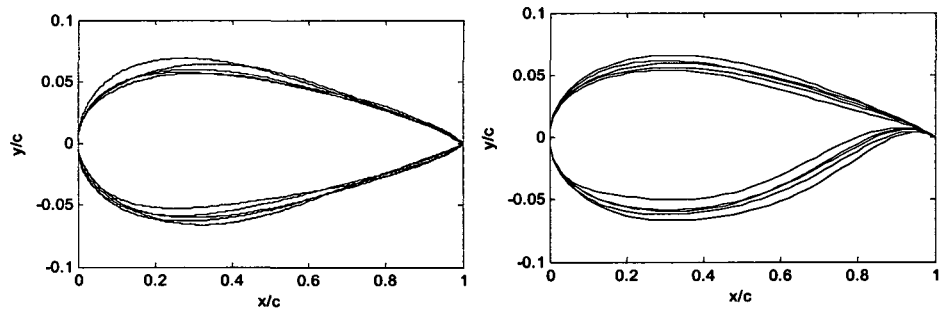

Figure 3.4 Initial populations based on Bezier curves (left) and Parsec curves (right).

For both processes angle of attack is fixed *0*. The fitness function Φ_i for i_{th} individual among population is defined as

$$\Phi_i = 1/f_i$$
$$f_i = \int (Cp^i - Cp^r) ds \tag{3.7}$$

where Cp^i is the pressure coefficient value of i_{th} individual, Cp^r is the pressure coefficient value of the reference curve. As a reference curve *Cp* distribution around Rae2822 asymmetric airfoil is selected. Initial/reference *Cp* distributions together with initial/target airfoil shapes are shown in Fig. 3.5.

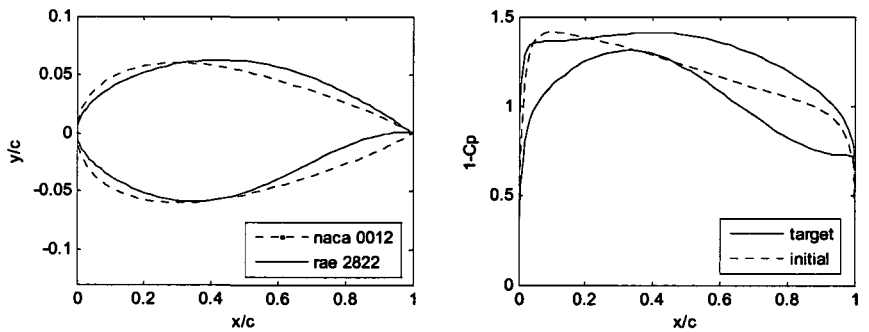

Figure 3.5 Initial and reference *Cp* distributions together with initial and target airfoil shapes

The features of a common optimization tool, VGA, are selected as follows: maximum generation is limited to 100, population number is 10, selection method is roulette, elite count is 1, mutation is vibrational mutation operator with frequency 4 and weight 0.5, cross over is modified Blx-α with α 0.8. At the end of the optimization processes the resulted typical *Cp* distributions and airfoil shapes for both representations are depicted in Fig. 3.6. The comparison between Bezier and Parsec are shown in Fig. 3.7. The plot shows the convergence histories of the averaged fitness values (over 20 runs). Fig. 3.7 emphasizes the superiority of the Parsec representation method. Regarding the average best individual fitness value, Parsec gives better results than the Bezier representation method while the second method shows more than 300% improvement in the final fitness value. The maximum, minimum and average fitness values for two methods are shown in Table 3.1. It is clear from this table that Parsec representation method is more efficient than the other one.

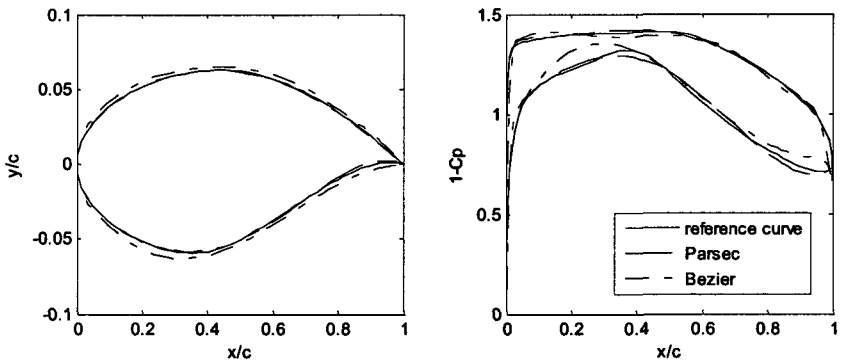

Figure 3.6 At the end of the optimization processes the resulted typical *Cp* distributions and airfoil shapes.

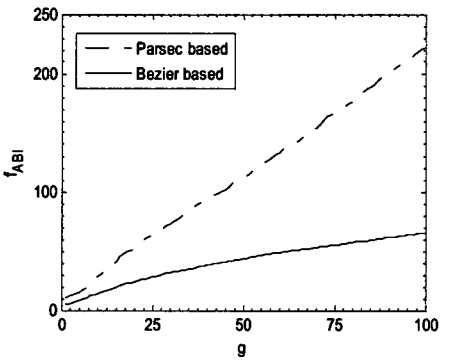

Figure 3.7 Comparison between Bezier and Parsec representations in accordance with $f_{\text{average best individual}}$ (f_{ABI})

Table 3.1 Fitness values and methods

Representation method	Bezier	Parsec
Max	78	402.5
Min	48.8	150.5
Average	65.4	221.2

Case 2

In the second test case shock wave reduction problem of Rae2822 airfoil at 2° angle of attack and Mach number 0.75 was investigated via two representation methods within VGA process. Initial populations for both representation methods are generated by using random number operator based on Rae2822 airfoil. Bezier curves are governed by 22 control points. Parsec curves are directed by 10 parameters. Sample initial populations for Parsec and Bezier curves are given in Fig. 3.8.

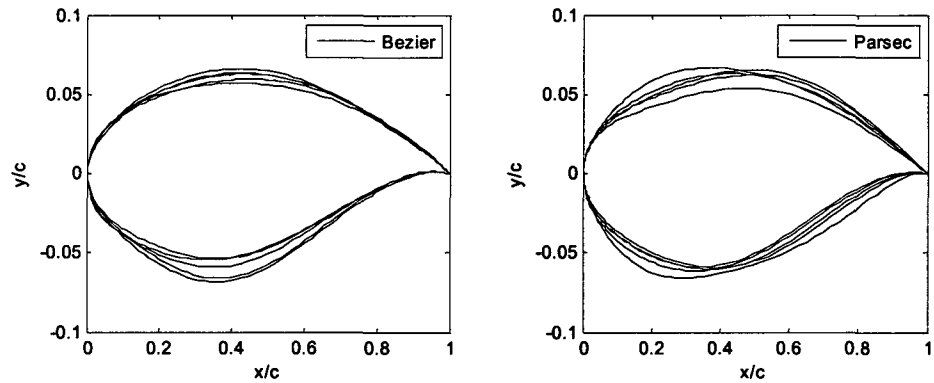

Figure 3.8 Initial populations based on Bezier and Parsec curves.

The fitness function, f, for i_{th} individual among population is defined as

$$f = \left[\frac{C_D}{C_L} + 10\ \left(C_L^* - C_{L2}\right)^2 + 100\ \left(t^* - t\right)^2\right]^{-1} \tag{3.8}$$

$$C_{L2} = \begin{cases} C_L^* & if \quad C_L \geq C_L^* \\ C_L & if \quad C_L < C_L^* \end{cases} \tag{3.9}$$

where C_L is the lift coefficient, C_D is the wave drag coefficient and t is the thickness ratio of the candidate airfoil, as C_L^* and t^* are the design lift coefficient (equal to 0.75) and thickness ratio (equal to 0.12) respectively.

The features of the common optimization tool, VGA, are selected as follows: maximum generation is limited to 100, population number is 10, selection method is roulette, elite count is 1, mutation is vibrational mutation operator with frequency 4 and weight 0.5, cross over is modified Blx-α with α 0.8. The original airfoil of Rae2822 and typical optimized ones (Parsec and Bezier representation based ones, from one of 10 independent runs of VGA), and their pressure coefficient *Cp* distributions are shown in Fig. 3.9 correspondingly. The comparison of optimization processes between Bezier and Parsec are shown in Fig. 3.10. The plot shows the convergence histories of the averaged fitness values (over 20 runs). Fig. 3.10 emphasizes the superiority of the Bezier representation method. Regarding the average best individual

fitness value, the Bezier method based optimization process gives slightly better results than the Parsec method based optimization process while the first method shows more than 40% improvement in the required iteration number for cost function value 23.5.

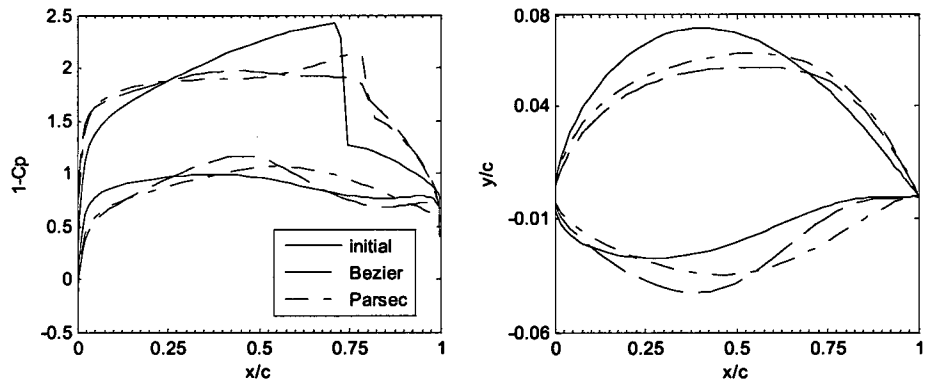

Figure 3.9 At the end of the optimization processes the resulted typical *Cp* distributions and airfoil shapes.

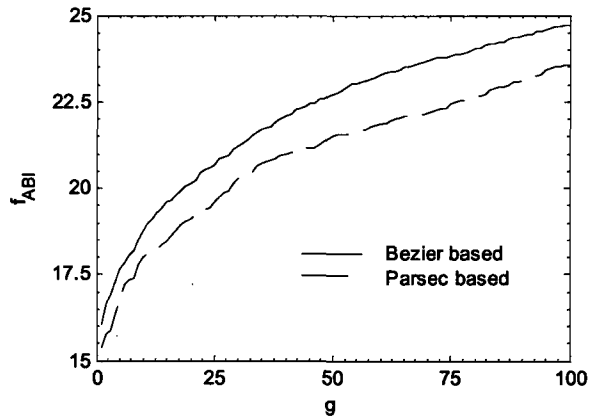

Figure 3.10 Comparison between Bezier and Parsec representations in accordance with average best individual.

Conclusion

In this study, Bezier and Parsec representation methods are tested in two different flow conditions: subsonic and transonic flows. In the first test case, both representation methods are compared via VGA optimization tool under the subsonic flow conditions. In the second test case, both representation methods are compared via VGA optimization tool under the transonic flow conditions. From these cases it is concluded that the Parsec method is more global and more efficient than the Bezier method in subsonic flows. However, the Bezier method is more flexible and more efficient than the Parsec method within transonic flows.

3.2 VGA Enhanced with FL and NN in 2D Airfoil Optimization

A new optimization algorithm called multi-frequency vibrational genetic algorithm (mVGA) is significantly improved and tested for two different test cases: an inverse design of an airfoil in subsonic flow and a direct shape optimization of an airfoil in transonic flow.

The algorithm emphasizes a new mutation application strategy and diversity variety, such as global random diversity and local controlled diversity. The local controlled diversity is based on either a fuzzy logic controller or an artificial neural network depending on the problem type.

Introduction

The present study introduces the application of a new multi-frequency vibrational genetic algorithm (mVGA) to speed up the optimization algorithm and overcome such problems as deficient diversity and premature convergence during the optimization. The principal role of this multi-frequency approach is to answer the question of which individuals should be mutated and when they should be mutated. Then, depending on the nature of the problem at hand, the present approach employs fuzzy logic or neural network concepts to provide local but controlled diversity within the population in addition to random global diversity. To demonstrate, mVGA and its variants are applied to two different test cases. First, a new fuzzy-logic coupled mVGA is tested on an inverse design problem at low flow speed conditions. Secondly, a new neural-network coupled mVGA is tested on an airfoil shaping problem at transonic flow conditions that mitigates the adverse shock wave effects.

Methodology

The present multi-frequency vibrational genetic algorithm (mVGA), which is given details in section 2, is an iterative algorithm for which a flow chart is presented in Fig. 3.11.

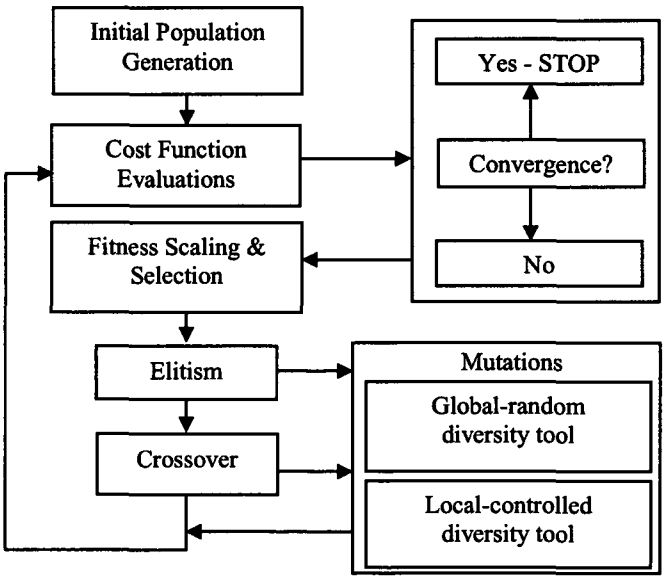

Figure 3.11 Flow chart of mVGA algorithm

An initial population is generated by using a random number operator based on baseline shape or parameters. To describe the method mathematically, let S be the population size, D be the individual (or chromosome) dimension space, f be the objective function, and $\boldsymbol{Z}_i$ be the current position vector including genes, $z_{i,j}(t)$, described in t^{th} iteration:

$$\begin{aligned} &Z_i(t) = (z_{i,1}(t), z_{i,2}(t), \ldots, z_{i,D}(t),), \\ &z_{i,j}(t) \in R^D, \; {}^{i=1,2,\ldots,S} \end{aligned} \tag{3.10}$$

The second step is to evaluate the fitness of the current population via a defined cost function f. Then, the cost weighting fitness scaling and roulette selection procedure [69] for mating are determined. The elitism concept is applied next to ensure that the best objective function value within a population is not reduced from one generation to the next. The procedure for the elite fitness value, f^e, and elite individual, $\boldsymbol{Z}^e$, is as follows:

$$f^e(t) = \arg\min_{Z_i(t)} f(Z_i(t)) \;\; \& \;\; Z^e(t) = Z_i(t) \tag{3.11}$$

$$Z^e(t) = \begin{Bmatrix} Z^e(t-1), & \text{if } f^e(t) > f^e(t-1) \\ Z^e(t), & \text{if } f^e(t) \le f^e(t-1) \end{Bmatrix} \tag{3.12}$$

The crossover technique denoted by BLX-α and described in [70] with α=0.5, is applied for the new individuals. The present mVGA mutation strategy is applied right after this crossover phase. At this step, there are two tools. As the first tool, the goal of the first mutation application is to provide a global random diversity in the population. For this reason, all the genes in all the chromosomes are mutated as follows:

$$z_{i,j}(t) = \begin{Bmatrix} z_{i,j}(t)[1 + w_1 \beta_1 (1-u)]_{j=1,2,\ldots,D}^{i=1,2,\ldots,S}, & \text{if } t = nf_1, \; {}^{n=1,2,\ldots} \\ z_{i,j}(t), & \text{if } t \ne nf_1, \; {}^{n=1,2,\ldots} \end{Bmatrix} \tag{3.13}$$

where f_1 is the application frequency, β_1 is a user defined amplitude parameter, u is a random real number between (0-1), and w_1 is a user defined scale factor. Implementing the mutation starts from the first gene position of the first chromosome, and continues throughout the genes at the same positions in the other chromosomes. As a second tool, the goal of the second mutation application is to provide a local but controlled diversity in the population. A fuzzy logic controller or a neural network application can be used at this stage depending on the problem at hand. For example, in an inverse design problem, the target is defined in the beginning. Therefore, using fuzzy logic is proper. However, in a direct optimization problem, such a target is not provided. Hence, a neural network application can be used to provide a local-controlled diversity within the population. In applying the fuzzy logic, modified elite genes, $z^{m.e.}$, are generated as given below:

$$z_s^{m.e}(t)=\left\{\begin{array}{ll} z_s^e(t)+w_2\beta_2, & if\ t=nf_2,\ ^{n=1,2,\ldots} \\ z_s^e(t), & if\ t\neq nf_2,\ ^{n=1,2,\ldots} \end{array}\right\} \tag{3.14}$$

$$\beta_2=\mathscr{F}_{func}(Z^e) \tag{3.15}$$

where f_2 is the application frequency, β_2 is the amplitude, w_2 is a user-defined scale factor, and s is a randomly determined gene number. Instead of fixing the value of β_2, it is estimated by the fuzzy logic controller function, $\mathscr{F}_{func}$. The modified elite gene is placed in an elite individual and this new individual is randomly located within the population as follows and applied I times:

$$Z^{m.e}(t)=(z_1^e(t),z_2^e(t),\ldots,z_s^{m.e},\ldots,z_D^e(t)) \tag{3.16}$$

$$(Z_k(t))_j=Z^{m.e}(t)\Big|_{j=1,2,\ldots,I}^{k=rand[1-D]} \tag{3.17}$$

In the neural network application, all the genes of an elite individual are mutated as follows:

$$P_{i,j}(t)=\left\{\begin{array}{ll} z_j^e(t)[1+w_2\beta_2(1-u)] & if\ t=nf_2,\ ^{n=1,2,\ldots} \\ z_j^e(t) & if\ t\neq nf_2,\ ^{n=1,2,\ldots} \end{array}\right\}_{j=1,2,\ldots,D}^{i=1,2,\ldots,N} \tag{3.18}$$

where u is a random real number between (0-1), β_2 is a user-defined constant amplitude. A newly generated temporal population $\boldsymbol{P}$ includes N individuals. The objective function values of this population, $\boldsymbol{f}^{NN}$, are predicted via trained neural network function, $\mathscr{N}_{func}$, and the best I of them are randomly placed within the population:

$$\begin{aligned} &\boldsymbol{f}^{NN}=\mathscr{N}_{func}(\boldsymbol{P}) \\ &\left[\boldsymbol{f}^{NN}\ order\right]=sort(\boldsymbol{f}^{NN}) \\ &(Z_k(t))_i=\boldsymbol{P}_{order(i)}\Big|_{i=1,2,\ldots,I}^{k=rand[1-D]} \end{aligned} \tag{3.19}$$

The frequencies f_1, f_2, and I are user-defined constants and their typical values are 5, 2, and 3, respectively. The special functions $\mathscr{F}_{func}$ and $\mathscr{N}_{func}$ will be explained in detail when presenting the case studies. After the mutation operations, a new population is evaluated via the cost function which is determined by the real solver. The algorithm repeats all of the above steps as necessary until the convergence criterion are satisfied.

Results

The mVGA algorithm will now be applied first to an inverse design problem, followed by a direct shape optimization of an airfoil in transonic flow. The algorithm, however, will be coupled with a fuzzy-logic controller for the first application. The lack of a target in a direct optimization problem makes use of fuzzy logic difficult for the second application. However,

using neural networks may be appropriate to provide the necessary local but controlled diversity in the genetic population. Therefore, for the second application, the algorithm will be coupled with a neural-network controller. The benefits for each coupling will be rather clear when the results are presented.

Fuzzy Logic Coupled mVGA

The airfoil family, known as "Parsec," was proposed to parameterize an airfoil shape. As mentioned before this technique aims to control the important aerodynamic features by selecting the design parameters based on the a priori knowledge of a flow around an airfoil. Parsec airfoils are defined by basic geometric parameters, such as, the leading edge radius (r_{LE}), upper and lower crest locations including curvatures (x_{up}, z_{up}, z_{xxup}, x_{lo}, z_{lo}, z_{xxlo}), trailing edge ordinate (z_{TE}), trailing edge thickness (Δz_{TE}), direction (α_{TE}), and wedge (β_{TE}) angles. The number of design parameters can be decreased to 9 by setting Δz_{TE} and z_{TE} to be equal to zero. The initial populations for the present case are generated by using the random number operator based on the NACA-0012 symmetric airfoil.

Objective function and flow solver

The angle of attack is assumed to be zero during the optimization process. The fitness function f_i for i^{th} individual among population is defined as,

$$f_i = \frac{1}{\sum_{j=1}^{L} (Cp_j^i - Cp_j^t)^2} \tag{3.20}$$

where Cp^i is the pressure coefficient value of i_{th} individual computed by a panel solver [71], Cp^t is the pressure coefficient value of the target curve, and L is the number of panels. A Cp distribution around the Rae2822 asymmetric airfoil is selected as the target pressure distribution. Shown in Fig. 3.12 are the initial and reference Cp distributions, and the initial and the target airfoil shapes.

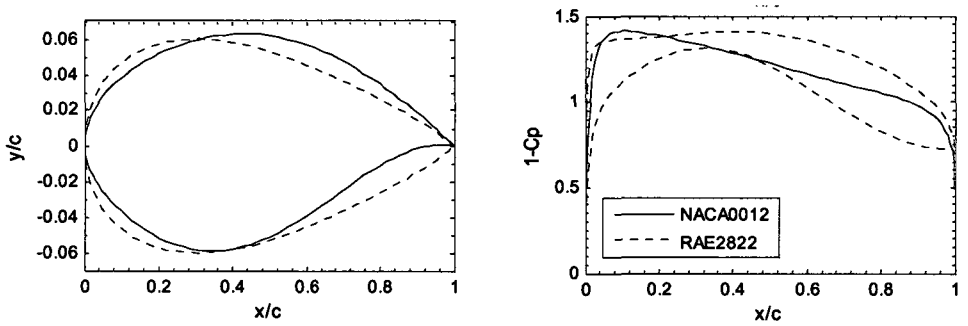

Figure 3.12 Initial and reference target airfoil shapes; initial and reference Cp distributions

Fuzzy logic controller and $\mathcal{F}_{func}$

Fuzzy logic is a form of multi-valued logic derived from a fuzzy set theory to deal with reasoning that is approximate rather than precise. It has been applied to diverse fields from control theory to artificial intelligence. This alternative, non-analytical technique has been recently employed to carry out a multi-objective optimization in different ways [72]. The fuzzy logic system employs a set of fuzzy linguistic rules, which may be provided by experts or can be extracted from numerical data. These rules are expressed as a collection of "if-then" statements. Therefore, a fuzzy rule base, $\boldsymbol{R}$, can be expressed as:

$$\begin{aligned} \boldsymbol{R} &= [R_i]^{i=1,2,...,N} \\ R_i &= \{if\ [c_i\ is\ A]\quad then\ [u_i\ is\ \beta]\} \end{aligned} \tag{3.21}$$

where c_i and A are the fuzzy inputs to the system; namely, i^{th} input variable and the consequent fuzzy set, respectively. On the other hand, u_i and β are the fuzzy output; namely, i^{th} design variable and the consequent fuzzy set, respectively. In the antecedent of rule R_i, the term $A=[A^1_j, ..., A^n_j]$ represents the vector of the fuzzy sets referring to the input fuzzy vector c. The membership functions of both the antecedent and consequent, A and β, respectively, have been chosen to be Gaussian; the inference engine employs a product inference for the rule implication. For the present inverse design case, the fuzzy system output represents the design variable amplitude and the fuzzy inputs are the error parameters, which can be described as follows:

$$e_j = Cp^t_j - Cp^e_j,\ ^{j=1,2,...,L} \tag{3.22}$$

where e_j is the error distribution of the elite individual, Cp^t is the target pressure distribution around the target airfoil geometry, and Cp^e is the pressure distribution of the elite individual. Input fuzzy vector, c, can be derived from error distribution by computing the center of distribution coordinates such as c_x and c_z for an elite airfoil using the following expressions:

$$c_x = \frac{\sum_{j=1}^{L-1} \frac{(e_j + e_{j+1})(x^2_{j+1} - x^2_j)}{2}}{\sum_{j=1}^{L-1} (e_j + e_{j+1})(x_{j+1} - x_j)} \tag{3.23}$$

$$c_z = \frac{\sum_{j=1}^{L-1} \frac{(e_j + e_{j+1})^2 (x_{j+1} - x_j)}{4}}{\sum_{j=1}^{L-1} (e_j + e_{j+1})(x_{j+1} - x_j)} \tag{3.24}$$

Since each Parsec parameter has an aerodynamic meaning, an error analysis is easily interpreted for both the upper and the lower curves of the airfoil. The sensitivities of the *Cp* distributions along with the variations on upper and lower curves are evaluated and the general characteristics of each Parsec parameter are determined. During this analysis, the centroids of error spectra and the error distributions are both taken into consideration. The characteristic behaviors of example Parsec parameters (x_{up}. and z_{up}) for the upper curve are shown in Fig. 3.13.

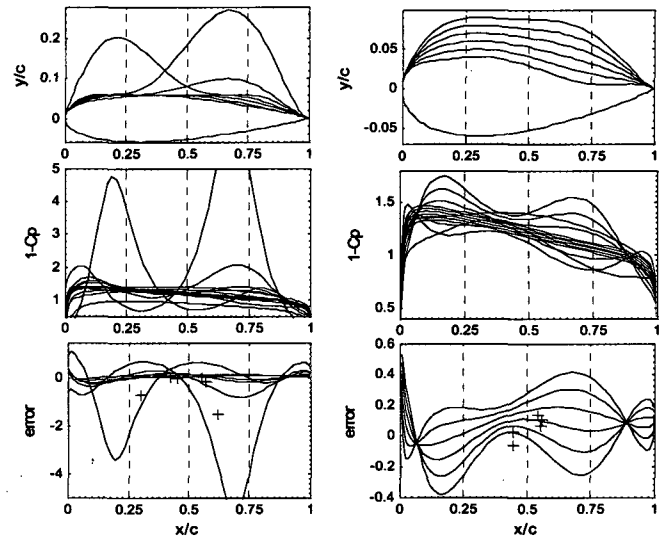

Figure 3.13 Airfoil shapes, pressure coefficient and error distributions, and error centroids (+) for selected Parsec parameters x_{up} (left side) and z_{up} (right side)

Just for illustration purpose, consider the z_{up} parameter. When the position of the central gravity (c) of the error distribution of z_{up} parameter is evaluated, it is possible to make global conclusions. An example inputs/output membership functions for z_{up} parameter is given in Fig. 3.14.

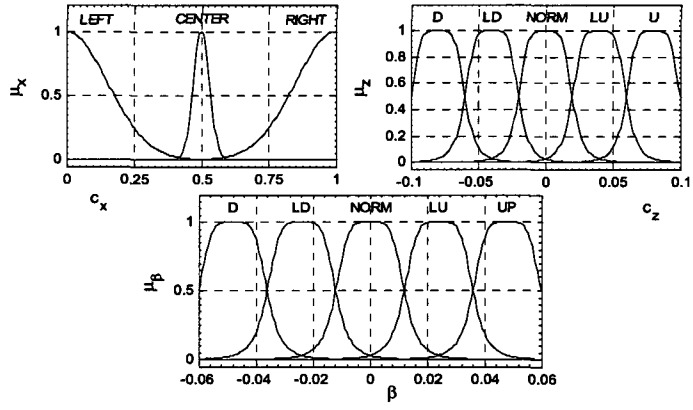

Figure 3.14 Membership functions of inputs and output for z_{up} parameter; μ_x, μ_z, μ_β, respectively. D: down, LD: little down, NORM: normal, LU: little up, U: up.

The order of magnitude of an error is between -0.5 and 0.5. The x coordinate of c is in the middle between 0.4 and 0.6. According to these error distributions and the position of the centroids, rules and membership functions are generated using Matlab 7.0 fuzzy logic toolbox for each of the Parsec parameters. The $\mathscr{f}_{func}$ expression for z_{up} parameter in terms of fuzzy rules is given in Table 3.2.

Table 3.2 The rules of z_{up} parameter amplitude estimation

If	c_z	LD	&	c_x	CENTER	Then	β	LU
If	c_z	NORM	&	c_x	CENTER	Then	β	NORM
If	c_z	U	&	c_x	CENTER	Then	β	D
If	c_z	D	&	c_x	CENTER	Then	β	UP
If	c_z	LU	&	c_x	CENTER	Then	β	LD

In addition, the rule surface for the same parameter is given in Fig. 3.15, and the flow chart of the local-controlled diversity tool as $\mathscr{f}_{func}$ is given in Fig. 3.16.

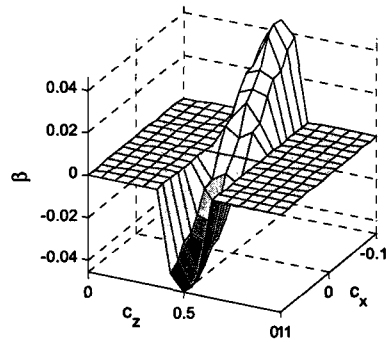

Figure 3.15 The fuzzy rule surface of z_{up} parameter

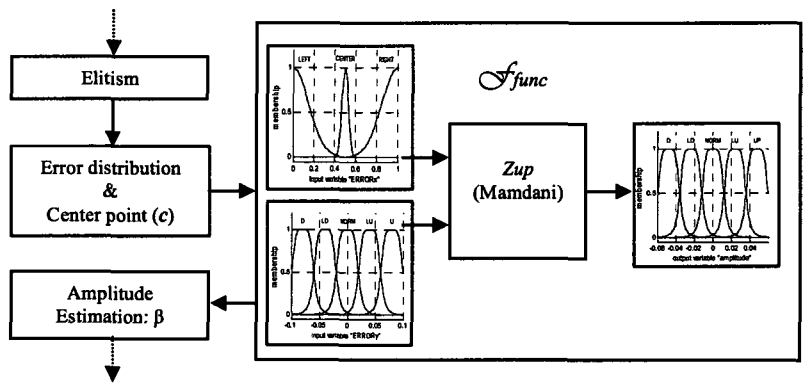

Figure 3.16 Flow chart of local-controlled diversity tool as F_{func}

An inverse design application

An inverse design problem for an airfoil typically has the resulting pressure distribution around the geometry as a given, then the method searches for a geometry that supports this distribution. The efficiency of fuzzy logic coupled mVGA is now tested through an

application to an inverse design problem. The algorithm features of VGA and fuzzy-coupled mVGA are given in Table 3.3.

Table 3.3 The features of VGA and mVGA

Method	G	*S/I*	Fitness scaling	Selection	f_i - w_i - β_i	Crossover	Elite count
VGA	200	10/0	rank	roulette	f_1 - w_1 - β_1 5/0.5/1	Blx-α	1
mVGA	200	10/3	rank	roulette	f_1 - w_1 - β_1 5/0.5/1 f_2 - w_2 - β_2 2/1/F_{func}	Blx-α	1

The comparisons between VGA and m-VGA results are shown in Fig. 3.17 and 3.18. Fig. 3.17 provides the average best individual (over 20 runs) against computational fluid dynamics (CFD) solver calls. This comparison clearly demonstrates the superiority of the present mVGA method. mVGA not only finds better results than VGA, but also it obtains more than 50% improvement in the final fitness value. Observed in Fig. 3.18, the resulting airfoil shape by mVGA is much closer to the target airfoil shape. In particular, the leading edge (A) and the lower curve (B) sections show the better fit to the corresponding target airfoil sections.

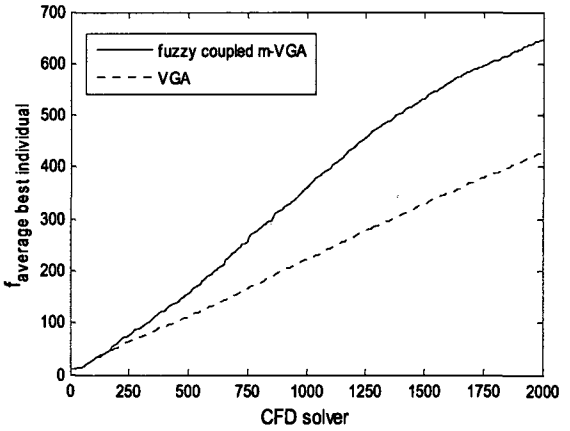

Figure 3.17 Comparison between VGA and mVGA in accordance with $f_{average\ best\ individual}$

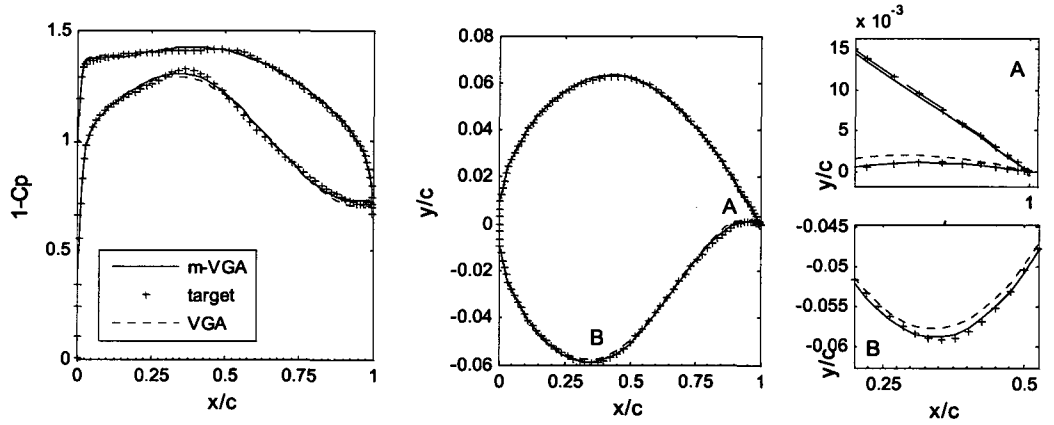

Figure 3.18 The *Cp* distributions and the airfoil shapes at the end of the optimization processes

Neural Network Coupled mVGA

In the second test case, the shape of an airfoil is optimized for a transonic flow at M=0.75 past an airfoil placed at 2° angle of attack and 6.5×10^6 Reynolds number. The initial shape is selected to be NACA-64A410 and the optimized shape is expected to weaken the shock wave formed on the upper surface in order to increase the lift to drag ratio. The optimization will be performed first by VGA and then the neural-network-coupled mVGA in order to demonstrate the enhancement with the latter.

Shape parameterization

An airfoil can be represented using the Bezier curves with a set of control points. The two control points (0,0) and (1,0) at the leading and trailing edges are fixed. It is commonly assumed that the x_i are fixed, then the design parameters are only the z_i coordinates of the control points. The initial population needed for the present method is generated by using a random number operator based starting with the NACA-64A410 airfoil.

Objective function and flow solver

The objective function, f is to be maximized, where

$$f = \left[\frac{C_D}{C_L} + 10 \left(C_L^* - C_{L2} \right)^2 + 100 \left(t^* - t \right)^2 \right]^{-1}$$
$$C_{L2} = \begin{cases} C_L^* & \text{if} \quad C_L \geq C_L^* \\ C_L & \text{if} \quad C_L < C_L^* \end{cases} \tag{3.25}$$

and C_L is the lift coefficient, C_D is the wave drag coefficient. C_L^* is the design lift coefficient and t^* is design maximum thickness ratio, which are taken for the demonstration case to be 0.885 and 0.1, respectively. Since the lift and drag coefficients are computed based on the pressure distribution around an airfoil, the pressure distribution needs to be computed at each generation of the design process. The pressure distribution around an airfoil for inviscid compressible flow can be determined by solving, for example, the Euler equations presented below

$$\nabla(\rho u \boldsymbol{U}) = -\frac{\partial p}{\partial x} \tag{3.26}$$

$$\nabla.(\rho v \boldsymbol{U}) = -\frac{\partial p}{\partial y} \tag{3.27}$$

In the equations above, ρ is the fluid density, p is the pressure, u is the flow speed in x-direction, v is the flow speed in y-direction, and $\boldsymbol{U}$ is the flow velocity vector. For 2D flows around an airfoil, the pseudo-time-dependent, compressible Euler equations are solved using a

finite volume method and the Roe flux-splitting scheme [73]. An elliptic PDE (Partial Differential Equation) method is used to generate a structured grid around the airfoil. The time required for grid production and flow solution around an airfoil varies depending on the geometry, however, and takes approximately 60 seconds when computed on an Intel(R) Core(TM) Duo CPU T7100 1.8GHz processor.

Neural networks and $\mathcal{N}_{func}$

Radial Basis Function (RBF) networks are designed to find a surface in a multidimensional space that provides a best-fit to the training data [74]. They are one of the popular approximation models that can be used as surrogates of the complicated and computationally expensive CFD methods. Before the description or the RBF network, some prerequisite definitions will be given here. An example radial basis function neural network is shown in Fig. 3.19. The input is z and the output is $y=\mathcal{N}_{func}(z)$, where $\mathcal{N}_{func}$ represents the processing by the entire radial basis function neural network. The input to the i^{th} receptive field unit, which is sometimes called a radial basis function, is z, and its output is denoted with $R_i(z)$. Assuming that there are n_R receptive field units, the output of the radial basis function neural network may be represented as follows:

$$y = \sum_{i=1}^{n_R} w_i R_i(z) + b \tag{3.28}$$

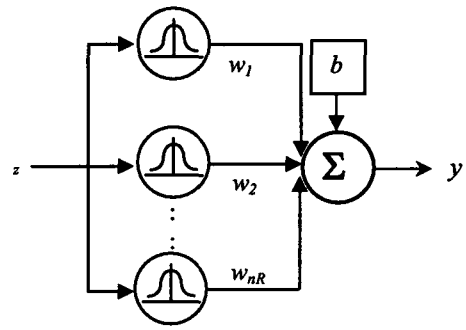

Figure 3.19 Radial basis function neural network model

There are several possible choices for the receptive field units, $R_i(z)$. A sample choice may be the following:

$$R_i(z) = \sum_{j=1}^{D} \exp\left[-\frac{1}{2}\left(\frac{z-c}{\sigma}\right)^2\right] \tag{3.29}$$

where c is the center and σ is the variance. Let's assume that there are N sample points for the problem with D design variables, the sampled data ($\boldsymbol{Z}_s$, $\boldsymbol{Y}_s$) can be defined as follows:

$$Z_S = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,D} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,D} \\ \vdots & \vdots & \cdots & \vdots \\ z_{N,1} & z_{N,2} & \cdots & z_{N,D} \end{bmatrix}, \; Y_S = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{3.30}$$

where $\boldsymbol{Z}_s$ is the design matrix with each row vector representing a sample point as input, and $\boldsymbol{Y}_s$ is the column vector that contains the value of the response at each sample point as output. The set of sample points are used to train the neural network. In Eq. (3.28), the unknown coefficients, w_i and b, can be calculated by the least squares regression technique:

$$[w_1, w_2, \ldots, w_m, b] = (H^T H)^{-1} H^T Y_s \tag{3.31}$$

$$H = \begin{bmatrix} R_1(z_1) & R_2(z_1) & & R_m(z_1) & 1 \\ R_1(z_2) & R_2(z_2) & & R_m(z_2) & 1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ R_1(z_n) & R_2(z_n) & & R_m(z_n) & 1 \end{bmatrix} \tag{3.32}$$

In the present test case, the Matlab function *newrb* is used to generate neural networks. In a set of training data, the input parameters are the control points of individuals (that is, parameterized airfoil geometry for aerodynamic optimization) in the population, while the output parameters are their aerodynamic performance values (that is, C_L or C_D) to be improved. There are some important points which require careful tuning during the use of the neural network model in the present methodology, mVGA. These are the number of generations required for training the neural network, and the selection of generations to be used for training. The first issue is related to the training cost and the second one determines the quality of the prediction. Instead of gathering all previous exact evaluations, it is more effective to gather a certain amount of collected data (N) based on the recent generations to avoid the computationally expensive task. Additionally, the selection of the neighboring patterns, such as the last certain generation, provides a better reliability. The reason is because the last certain generation is locally closer in terms of Euclidean distance in the design space to the current individuals and their perturbed variants.

The flow chart of the local-controlled diversity tool as $\mathcal{C}_{func}$ is presented in Fig. 3.20. The steps of the neural network coupled with the present mVGA methodology are as follows. First, the cost function calculations in the current population are performed to get their fitness values. Secondly, the two networks are trained after the threshold generation point (the threshold is selected as the 10^{th} generation in the present implementation) by using the individuals in the current and previous populations and their fitness values. The first network is trained for the C_L estimation and the second is for the C_D estimation. However, the training set needs to be limited. For the present test case, it is limited with the last 100 data set. For this training, individuals in the population are used as the input and their fitness values are

used as the output. Lastly, the mutations, as in Eq. (3.18), are applied and the temporal neural network population is generated. It is taken 40 as N in the application. After the generation of the neural network population, the fitness values of network population are estimated by using trained neural nets. Then, some of them, which have greater fitness values than the elite individual (in the present use, the best two are selected) are placed in the new population to be used as candidates at the next step of the algorithm.

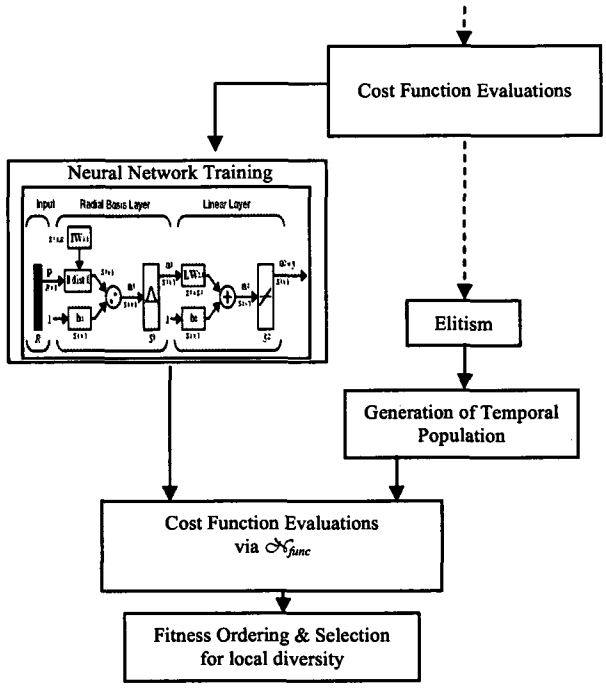

Figure 3.20 Flow chart of local-controlled diversity tool as $\mathcal{H}_{func}$

Aerodynamic shape optimization case

The efficiency of the neural-network-coupled mVGA is tested and it is compared with the corresponding VGA results. The features of the VGA and m-VGA algorithms are provided in Table 3.4.

Table 3.4 The features of VGA and mVGA

Method	G	S/I	Fitness scaling	Selection	f_i - w_i - β_i	Crossover	Elite count
VGA	100	10/0	rank	roulette	f_1 - w_1 - β_1 4/0.5/1	Blx-α	1
mVGA	50	10/3	rank	roulette	f_1 - w_1 - β_1 4/0.5/1 f_2 - w_2 - β_2 1/0.5/1	Blx-α	1

A comparison of the average best individuals (over independent 10 runs) versus the CFD solver calls is given in Fig. 3.21. VGA reaches the objective function value of 24.52 within 1000 CFD computations while the neural-network-coupled mVGA obtains the same value within 330 CFD calculations. mVGA clearly shows better performance than VGA after the 10^{th} generation, because a local-controlled diversity tool based on neural networks is

activated after this iteration. This result indicates that m-VGA decreases the CFD solver calls by about 67% with respect to VGA.

The initial NACA-64A410 airfoil and the optimized shapes (from one of 10 independent runs of each optimization algorithm), and their corresponding *Cp* distributions are shown in Fig. 3.22. Both optimization algorithms keep the lift coefficient and maximum thickness ratio fixed at their design values.

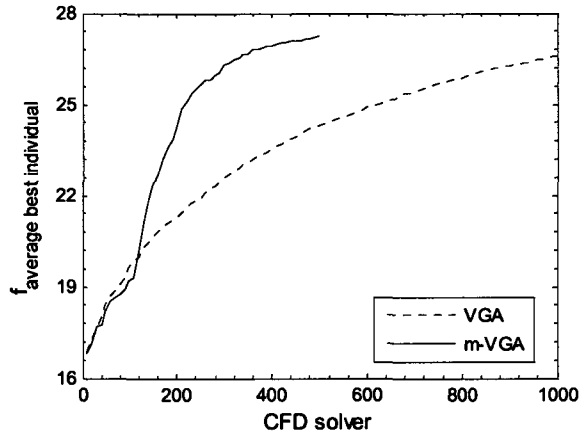

Figure 3.21 Comparison between VGA and NN coupled mVGA in accordance with $f_{average\ best\ individual}$

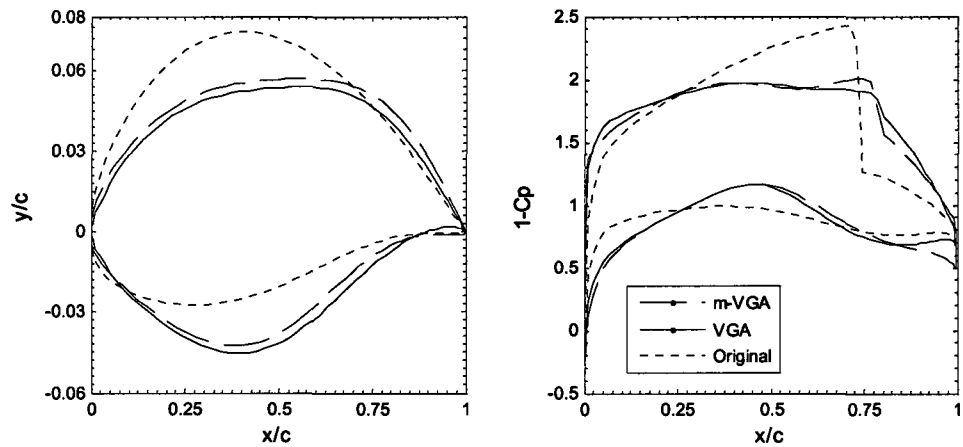

Figure 3.22 The resulted typical *Cp* distributions and airfoil shapes at the end of the optimization processes

Conclusions

The present paper introduced a new multi-frequency vibrational genetic algorithm (mVGA) to speed up the optimization algorithm and overcome problems, such as, deficient diversity and premature convergence. Then, depending on the nature of the problem at hand, the present approach employed fuzzy logic or neural network concepts to provide local but controlled diversity within the population in addition to random global diversity. The average

best-individual-fitness values of all algorithms were recorded for a fair comparison among them. To demonstrate their merits, mVGA and its variants were applied to two different test cases. First, a new fuzzy-logic coupled mVGA was tested on an inverse design problem at low flow speed conditions. Secondly, a new neural-network coupled mVGA was tested on an airfoil shaping problem at transonic flow conditions that mitigated the adverse shock wave effects.

3.3 A New GA for Path Planning of Autonomous UAV

This study introduces a new evolutionary algorithm that can be used to solve path planning problems of autonomous unmanned aerial vehicles (UAVs). This new algorithm combines fuzzy c-means clustering and Voronoi diagram along with multi frequency vibrational genetic algorithm (mVGA) to find the optimal paths. In this study clustering method and Voronoi diagram concepts are used within initial population phase of mVGA process.

Introduction

The number of applications that are considering the use of intelligent unmanned vehicles is increased within civilian or military purposes. Unmanned aerial vehicles have become an indispensable platform for many applications where manned operation is considered unnecessary or too risky. These applications include goals such as reconnaissance, search and rescue, weather observation, target detection, or target destruction. For both military and civilian applications, there is a desire to develop more sophisticated UAV platforms where the emphasis is placed on development of intelligent capabilities. There are many activities that must be followed or carried out by an UAV system to enable the execution of the task of autonomous navigation. These activities are mapping and modeling the environment, path planning and flight control systems. Between these three problems, it can be argued that path planning is one of the most important problems in the navigation process. Its intention is to find out the optimal or suboptimal safe flight trajectory starting from departure location and ending to arrival location in the proper duration which UAV is able to accomplish the pre-arranged task and avoid the hostile threats.

Regardless of the domain, the path planning problem can be broken up into two major categories: solving the problem of creating a planned path to follow in a static environment, or creating a plan for a dynamic environment. In the first environment, the starting point, ending point, set of regions to avoid, and set of regions to visit, are all known before the proposed plan is constructed. In the second case, some or all of the set of regions to avoid, may be initially unknown. It is called a dynamic path planning which includes global path

construction together with local path plannings that make the UAV avoid collision in real time state. Changing the planning problem from fixed obstacles to moving ones changes the problem from a deterministic problem to a stochastic problem. The selection of an appropriate algorithm in every stage of the path planning process is very important to ensure that the navigation process will run smoothly. In computing, data structures play an important role and greatly influence the computational complexity and efficient implementations of an algorithm. Because the optimal path planning relies too heavily on time consuming optimization techniques such as numerical computation, it is usually solved offline based on the known information before takeoff.

There are different groups of path planning algorithms in the literature. Researchers have fulfilled extensive analysis and simulation studies. The first group is called skeleton approach. This approach is also called the roadmap, or highway approach. These algorithms are usually based on configuration space representations. Widely used skeletons are the visibility graph, Voronoi diagram, subgoal network, and the silhouette [75]. The second group of approaches is cell decompositions. A world space is triangulated into a mesh. An optimized path graph is extracted from this mesh. Dijkstra's shortest path searching algorithm is usually used to search the shortest path in the path graph. The third group is called the potential field approach, in which a scalar function is constructed from the information of obstacles and contractions. The path is determined by performing steepest gradient descent on the potential function [76]. In recent years, a path planning problem is introduced to new or hybrid techniques such as fuzzy logic [77, 78], neural network [79], ant colony optimization (ACO) [80, 81], Dijkstra's algorithm and ant system algorithm [82], the artificial potential field approach with simulated annealing [83, 84], particle swarm optimization (PSO) [85], genetic algorithm with simulated annealing [86], and genetic algorithms (GA) [87-94].

Each method has its own strength over others in certain aspects. Unfortunately, there are some deficiencies such as computational complexity, local optimization, and adaptability when we use those algorithms to deal with path planning in complex environments. For instance, in a large-scale background environment, the computational cost of algorithms based on skeleton or cell decomposition approaches is very high and the approximate optimal solutions cannot always be obtained. The potential field methods provide simple and effective motion planning for practical purposes. However, a potential field approach has a major problem which is that a vehicle may be trapped at a local minimum before reaching its goal. Even in global search methods such as genetic algorithms, ACO or PSO it often takes a long time for planners to escape from local optimal solution once the evolution converges to a local optimal plan. This is called premature convergence. Although there are some successful

applications to compute near-optimal paths for UAVs, sometimes, premature convergence may prevent search algorithms from reaching global optimal solutions. The main reason for the premature convergence is usually addressed by the lack of diversity. Parallel evolution including several populations is one approach to overcome premature convergence via increasing the diversity. However, this approach also increases the computational cost. In addition to premature convergence problem, almost for all reported genetic algorithms, the individuals of the first population are assumed to be generated randomly. This is simple but will lead to large quantities of infeasible paths if used in path planning. Although some of these infeasible paths might become feasible solutions after certain genetic operations, they cause several problems such as more computational time or meaningless work. To overcome these problems, this article emphasizes a new Voronoi supported multi-frequency vibrational genetic algorithm (mVGA). Basic contributions of this study are the improvement of initial population by using a clustering method and Voronoi diagram in three-dimensional (3D) space, and genetic algorithm enhancement by using quantification and qualification of diversity in the population

Optimization Method

In path planning problems mVGA is selected as an optimization method. The details of the optimization method are given in the previous sections.

Initial Population Enhancement

The aim of the optimization process is to optimize design variables which represent the optimal path. The time of optimization process and also the result of it heavily depend on the initial points. Improper initial population may cause time consuming process and premature convergence. Therefore, proper and robust initial population is very important for the optimization. Before proceeding to initial population enhancement method we will explain the preliminaries such as representations of path and terrains.

Path Representation

Path parameterization plays a key role since it defines the design variables. Bezier curves or splines have been widely adopted when computing smooth, dynamically feasible trajectories for UAVs. Smooth path is an important feature for UAV flight dynamics because aerial vehicles can not fly on line segments like land vehicles. Additionally, smooth path prevents the vehicle from sharp turnings which would be harmful for the structure. The other advantage of employing Bezier curve is that the path can be represented using a relatively smaller number of parameters than using a complete geometric description of the path. This results in a small computational cost. Coordinate transformation between aerial vehicle and

the terrain model is not taken into consideration because the test cases are implemented in demonstrative terrain models.

Terrain Modeling

Two different 3D surface representation methods; trigonometric function based terrain modeling and city surface based terrain representation are used to simulate the topology of the configuration space. The function of trigonometric based terrain modeling is given by

$$z(x,y) = \sin(y+a) + b\sin(x) + c\cos(d(\sqrt{x^2+y^2})) \\ +e\cos(y) + f\sin(f(\sqrt{x^2+y^2})) + g\cos(x) \tag{3.33}$$

where *a, b, c, d, e, f,* and *g* are real constant numbers. Different constants generate different terrain models. On the other hand, 160 buildings (rectangular prisms with different heights) are used to simulate city type environment. It is possible to model the real city or city regions by using city map and building height info. An example model is generated via *x, y* matrixes based on the random building locations and *z* matrix based on the random building heights. Both terrain models are depicted in Fig. 3.23 and 3.24.

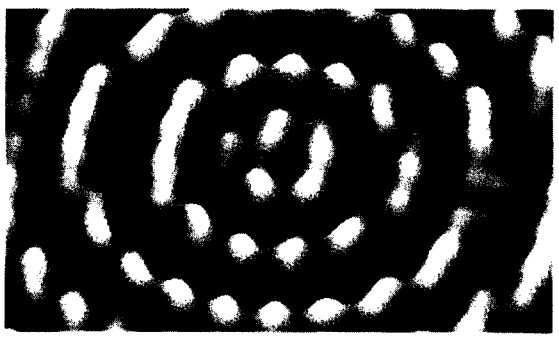

Figure 3.23 Sinusoidal terrain modeling

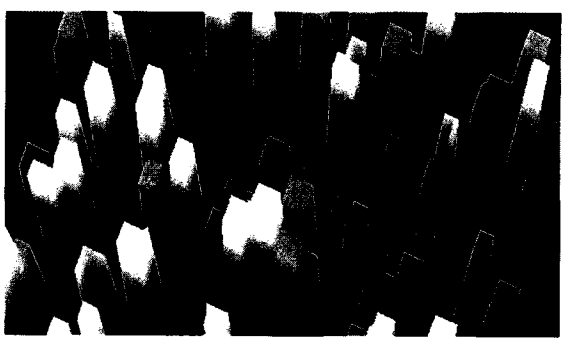

Figure 3.24 City type surface modeling.

These surface models are used to simulate environments of path planning problems. Sinusoidal terrain model is a mathematically continuous model. A city type terrain model is a discrete and discontinuous model. However, GA can handle both types of models.

Initial Population

Initial population is described as follows

$$P = P_r + P_v \tag{3.34}$$

where P_r is the random based population, P_v is the remaining population. Random based population includes the individuals generated by using random number operators. The design variables; (x_i, y_i, z_i) are randomly generated within the terrain model bounds and amplitude bound of UAV. However, the first and the last point coordinates are fixed to start point and end point, respectively. The remaining population includes the individuals generated by using Voronoi diagram.

Voronoi diagram based individuals

A Voronoi diagram for a set of N points p_i, $1 \leq i \leq N$, in the Euclidean plane is a partitioning of the plane into N polygonal regions, one region associated with each point p_i. A point p_i is referred to as a Delaunay point. The Voronoi region $V(p_i)$ associated with point p_i is the locus of points closer to p_i than to any of the other $N-1$ points. The Voronoi edge separating $V(p_i)$ from $V(p_j)$ is composed of the points equidistant from p_i and p_j. Not all Voronoi edges are bounded; some extend to infinity. The intersection of Voronoi edges occurs at vertices called Voronoi points. The construction of Voronoi diagrams is reviewed in [95].

As expressed in introduction section, Voronoi diagrams are efficiently used to find threat avoiding paths. However, Voronoi diagrams are two dimensional; therefore, they don't care about the altitude restrictions. Additionally, these diagrams include straight lines and UAV cannot fly on these straight lines due to flight mechanics. To solve 2D plane problem, Krozel *et al.* [96] studied to polygonize the counter data of mountains. Vertices from mountain polygon obstacles are used as Delaunay point locations to model the obstacles. To solve line segments problems, Jong *et al.* [97] used B-spline path templates. In his study, in conjunction with the high-level path planner, the proposed algorithm finds the corresponding local path segments and stitches them together, while preserving the smoothness of the composite curve. As a high level path planner he used a cell decomposition algorithm. Different from these studies we propose a simpler method.

Voronoi diagram needs to N Delaunay points. To get these points we can use a simple approach; filtering and clustering. At the beginning, the terrain model can be filtered in accordance with the altitude limitation of UAV. Simply, if the point of the terrain model is higher than the UAV altitude limitation then this point is placed in obstacle/dangerous points group keeping the location info of the group in memory; otherwise, it is excluded. Secondly, dangerous points are clustered and cluster centers are found by using fuzzy c-means clustering method [98]. The numbers of cluster center, N, is predicted before the clustering process. Thirdly, cluster centers are used as threats to construct Voronoi diagram. A Voronoi diagram is constructed from the known cluster center locations. However it would be assumed that it is difficult to get data clustered exactly. Therefore there are some defective cluster centers. These centers cause unsafe Voronoi edges. But this is just the beginning of the optimization process and genetic process eliminates these deficiencies during the optimization. In the following figures an example part of sinusoidal terrain model (Fig. 3.25), filtration of model points (3.26), and generated cluster centers (3.27) are depicted.

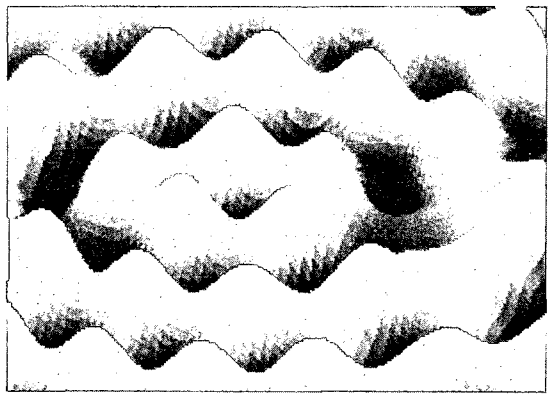

Figure 3.25 Example part of sinusoidal surface modeling.

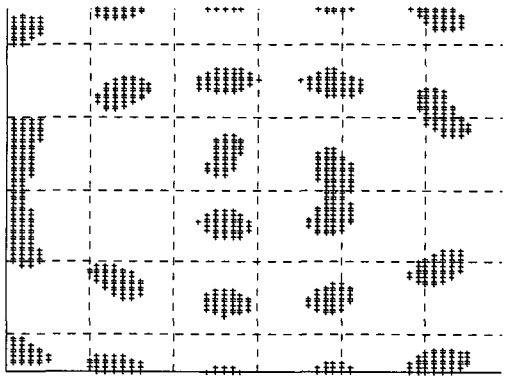

Figure 3.26 Filtered dangerous points of sinusoidal surface modeling.

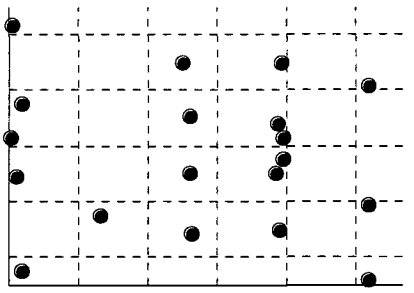

Figure 3.27 Cluster centers of dangerous points.

These cluster centers are used as Delaunay points to construct Voronoi diagram. Example Voronoi diagram is depicted in Fig. 3.28.

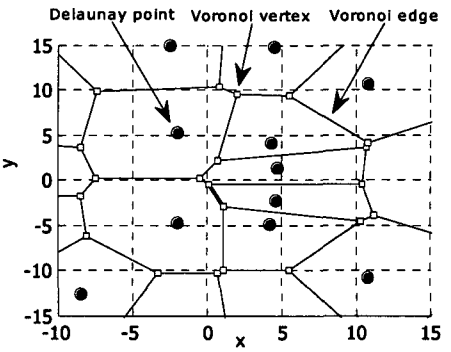

Figure 3.28 Voronoi diagram based on cluster centers.

After constructing a Voronoi diagram, the start location and end location need to be connected to the Voronoi vertices. We simply connect the start location and the end location to the nearest several nodes of the Voronoi diagram. Similarly, proper Voronoi vertices between start and end points of the path are selected and these vertices are used as control points of the Bezier curves. An example implementation is depicted in Fig. 3.29 and resulted path is shown in Fig. 3.30.

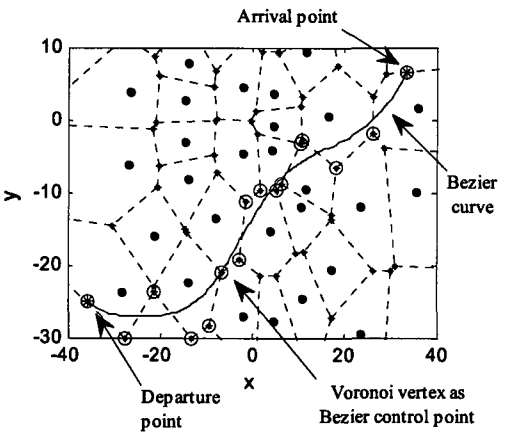

Figure 3.29 Voronoi vertices based Bezier curve.

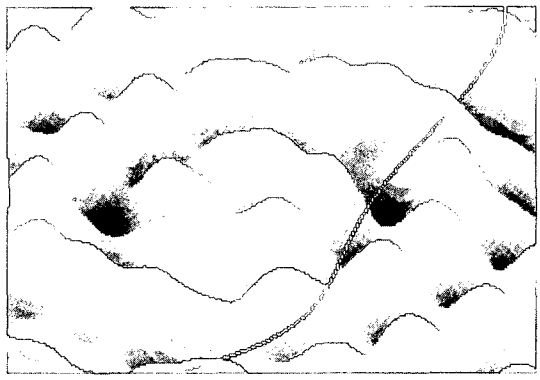

Figure 3.30 Constructed path (white color) based on Bezier curve.

These constructed path solutions are Voronoi based initial individuals. The number of Voronoi vertices on the constructed path determines the number of control points of Bezier curve, shortly (x_i, y_i). The third coordinate (z_i) is determined as altitude limitation of UAV.

Fitness Function

The evaluation function of an individual measures the cost of a candidate path. The fitness function is designed to accommodate three different optimization goals: minimize the distance flown, maintain a smooth trajectory preventing sharp turns, and satisfy the clearance providing the safe distance for UAV from terrain. We have proposed a linear combination of these three factors. The general description of the fitness value is given below

$$f = \frac{1}{\sum_{j=1}^{3} a_j F_j} \tag{3.35}$$

where Fj is the suboptimization goal, a_j weighting constant. The first concern is the length of the curve F_1, and is calculated by the given expression

$$F_1 = \sum_{i=1}^{n-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2]^{1/2} \tag{3.36}$$

where n is the curve discretization number; x, y, and z values are the curve discrete coordinates. The second concern is the passing ratio of the curve through the terrain boundary F_2, and is calculated by the given expression:

$$F_2 = \sum_{i=1}^{n} [\begin{array}{c} if\ (z_{Curve} - z_{Surface}) < S_D \\ or\ z_{Curve} > z_{Al} \\ punishment + 1 \end{array}] \tag{3.37}$$

where S_D is a safe distance determined by user, z_{Al} is an altitude limitation for UAV, z_{Curve} is the discrete path curve coordinate, and $z_{Surface}$ is the terrain model coordinate. This expression penalizes the curve that passes through the solid boundary. So, the penalty value is proportional to the number of discretised curve points located under the solid surface. The third concern is the curvature angle ratio of the curve F_3, which is calculated by the given expression:

$$F_3 = \sum_{i=1}^{m-1} [\begin{matrix} if\ \theta_{i,i+1} < \theta_T \\ punishment + 1 \end{matrix}] \tag{3.38}$$

where $\theta_{i,i+1}$ is the angle between the extension of the line segment connecting Bezier control points i and $i+1$, θ_T is the safe turning angle determined by user for the UAV. This concern is designed to prevent the aerial vehicle from exceeding the lateral and vertical acceleration limits. Because the flight envelope determines the maximum radius of turns for flying objects. The weight constants, a_j, are determined experimentally. In test cases a_2 is selected as high number.

Test Cases and Results

The new improved algorithm is tested and compared in different environments. The first environment is selected as a sinusoidal terrain model. The second environment is selected as a city type surface modeling. For both test cases the job is to fly from departure point to an arrival point under described conditions.

Sinusoidal Terrain Results

The test case is given in Fig. 3.31. The mission of UAV is to depart from departure point and arrive at arrival point under certain conditions. At first, four different genetic algorithms are tested in this case. These algorithms are described in accordance with mutation types. The features of these algorithms are tabled in Table 3.5. The first two algorithms are labeled as regular genetic algorithms such as RGA_1 and RGA_2. In these algorithms classical mutation strategy applications are performed including Gaussian and uniform mutations. VGA is named as vibrational genetic algorithm and the first vibrational mutation operator is applied to get global diversity in the population. mVGA is labeled as multi-frequency vibrational genetic algorithm and both vibrational mutation operators are applied to get global and local diversity in the population. Blx-α is used for all algorithms. The data for fitness function is given in Table 3.6.

Table 3.5 Genetic algorithms' features

Method	Mutation	Crossover
RGA_1	Gaussian; β_1, 0.5; γ, 0.75	Blx-α; α,0.5
RGA_2	Uniform; R_m, 0.05	
VGA	Vibrational: f_1, 5; w_1, 1;β_1, 0.5	
m-VGA	Vibrational: f_1, 5; w_1, 1; β_1, 0.5; f_2, 2; w_2, 1; β_2, 2.5	

Fitness scaling	selection	Elite	T	Run	S
Rank	roulette	1	100	20	10

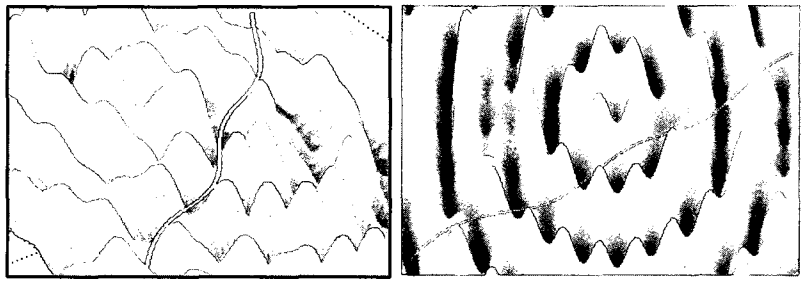

Figure 3.31 An example constructed path (white color) with different views by mVGA.

Table 3.6 Fitness function data

S_D	θ_T	z_{AI}	a_1	a_2	a_3
0.05	60°	5	5	10	1

All algorithms are run 20 times and the averaged best individual fitness function values versus generations are plotted in Fig. 3.32. According to this figure VGA and m-VGA outperformed regular genetic algorithms. VGA reaches the value of 0.002 (1/averaged f_{min}) at 83rd generation while RGA_2 reaches the same value at 100th generation. Therefore, VGA decreases 17% the required generation number. mVGA looks very efficient in this case. mVGA reaches the value of 0.002 at 12th generation. It decreases 78% the required generation number.

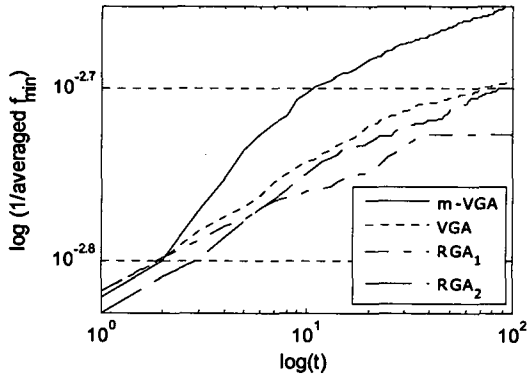

Figure 3.32 The fitness values of algorithms versus generations

Additionally, mVGA is supported by initial population enhancement and the resulted values are compared with pure mVGA. The algorithms are again run 20 times and the averaged best individual fitness values are plotted in Fig. 3.33.

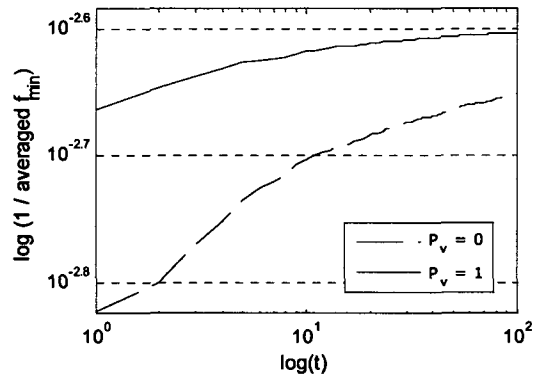

Figure 3.33 The fitness values of algorithms including m-VGA and Voronoi supported m-VGA versus generations

Fig. 3.33 shows that the enhancement of initial population cause significant improvement in fitness function values. It improves the algorithm almost 100 %. mVGA reaches the value of 0.0022 at 100th generation while Voronoi supported mVGA starts with the value of 0.0021. On the other hand, the number of individuals generated by Voronoi diagram; P_v and the algorithm performances are depicted in Fig. 62. It seems that the main point is to add a reasonable individual within the population. Because increase in P_v does not results in performance improvement.

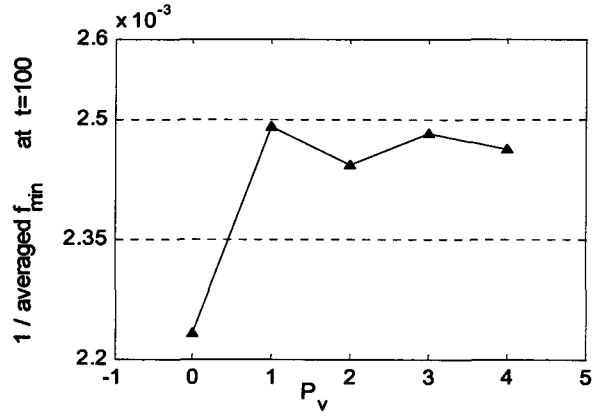

Figure 3.34 The resulted fitness values of algorithms at 100th generation versus the number of individuals generated by Voronoi diagram; P_v.

City Type Terrain Results

The test case is given in Fig. 3.35. Similar to previous test cases, the mission of UAV is to depart from departure point and arrive at arrival points above the city under certain conditions. At first, two different genetic algorithms including VGA and mVGA are tested in this case. The features of these algorithms are tabled in Table 3.7. The data for fitness function is given in Table 3.8.

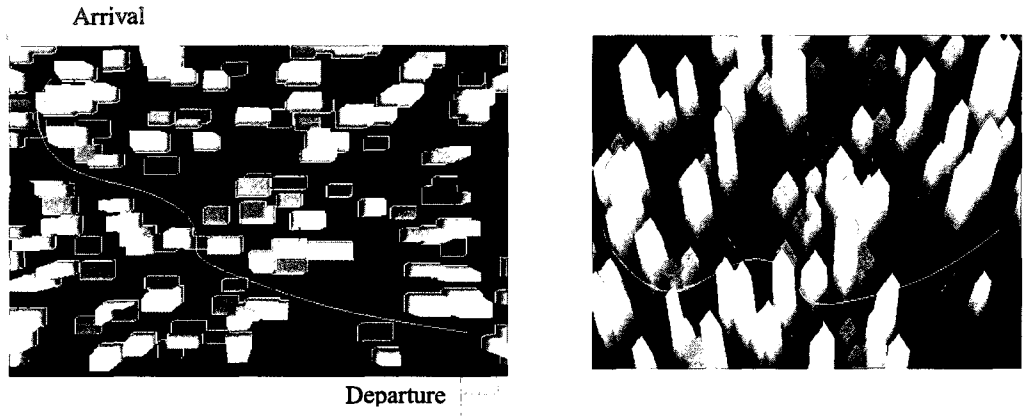

Figure 3.35 An example constructed path (white color) with different views by m-VGA.

Table 3.7 Genetic algorithms' features

Method	Mutation					Crossover
VGA	Vibrational: f_1, 5; w_1, 1;β_1, 0.5					Blx-α; α,0.5
m-VGA	Vibrational: f_1, 5; w_1, 1; β_1, 0.5; f_2, 2; w_2, 1; β_2, 2.5					
	Fitness scaling	**selection**	**Elite**	***T***	**Run**	***S***
	Rank	roulette	1	100	20	10

Table 3.8 Fitness function data

S_D	θ_T	z_{Al}	a_1	a_2	a_3
0.05	60°	5	5	10	1

All algorithms are run 20 times and the averaged best individual fitness function values versus generations are plotted in Fig. 3.36.

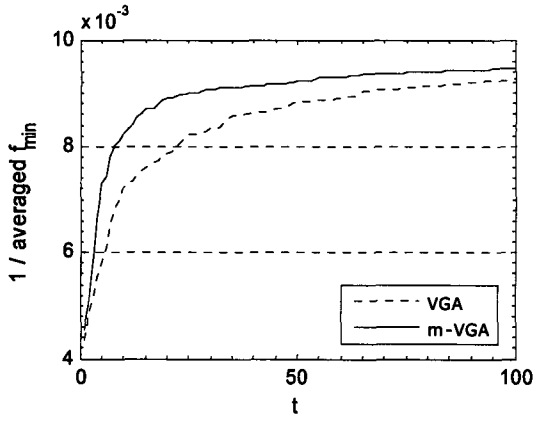

Figure 3.36 The fitness values of algorithms versus generations

According to this figure mVGA outperformed VGA. VGA reaches the value of 0.0092 (1/averaged f_{min}) at 100th generation while m-VGA reaches the same value at 50th generation. Therefore, m-VGA decreases 50% the required generation number. The efficiency of mVGA is a bit lower than the previous case (sinusoidal terrain case). This may originate from the terrain data. The sinusoidal terrain model is an exact model and the computation of fitness

values are based on continuous functions. However, a city terrain model is a discrete model. Therefore, it is considered as the reason for the lower efficiency.

Additionally, m-VGA is supported by initial population enhancement and the resulted values are compared with pure mVGA. The algorithms are again run 20 times and the averaged best individual fitness values are plotted in Fig. 3.37. Fig. 3.37 shows that the initial effect is not significant as the previous case. However, the effect of Voronoi based individuals on the process is very significant. mVGA reaches the value of 0.0083 at 100^{th} generation while Voronoi supported m-VGA reaches the same value at 11^{th} generation. This means 89% decrease in required generation number.

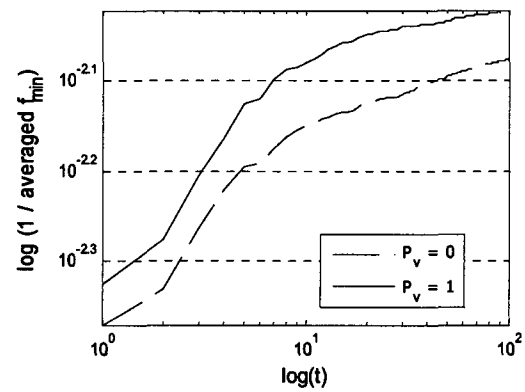

Figure 3.37 The fitness values of algorithms including m-VGA and Voronoi supported m-VGA versus generations

Conclusion

Although genetic algorithms are global search methods they are suffered from premature convergence and low convergence rates. Instead of classical applications this article showed that periodic mutation applications based on vibrational mutation operators provide classified but efficient diversity in the population. In addition to a new mutation strategy, the initial population is improved by using Voronoi diagrams, and hence the convergence is seriously accelerated.

3.4 VGA Enhanced With NN in RCS Problems

Within this study, multi frequency vibrational genetic algorithm [mVGA] is used to accelerate the genetic algorithm for radar cross section minimization problem.

Introduction

In today's World there is no point in considering military aerodynamic configuration development without including a stealth feature. It plays a key role in the configuration

layout. For aerodynamic configuration design, the key element is a radar cross section, *RCS*, with some consideration of infrared, mainly from the back of the aircraft. Clearly, flat surfaces normal to the incoming waves are bad, and reflect strongly back to the radar transmitter, thus surfaces should be angled to reflect the waves in other directions. The optimal shapes have sharp corners in directions where the *RCS* is minimized and exhibit corrugations. Designers work to decrease *RCS* target values [levels] in different sectors. These sectors are front sector, sides, and rear sector which is often emphasizes infrared [99, 100]. Optimization of *RCS* has commonly relied on evolutionary algorithms mainly genetic algorithms [101-104]. There are also some gradient-based optimization techniques [105, 106]. An optimization under the concerns of aerodynamics and electromagnetic is very complex and multi-objective process. In this test case only the RCS feature is considered and the shape of the aircraft is optimized to minimize the RCS under the limitations of aircraft configuration. RCS is a function of target configuration, frequency, incident polarization, and receiver polarization [107]. The frequency is fixed as 0.015 GHz. Incident and receiver polarization angles are taken as fixed values. Thus the target configuration is determined as optimization issue.

Optimization Method

The present neural network coupled multi-frequency vibrational genetic algorithm is an improved version of the multi-frequency vibrational genetic algorithm (mVGA). It is an iterative algorithm for which a flow chart is presented in Fig. 3.38. An initial population is generated by using a random number operator based on baseline shape or parameters. In the present test case, the Matlab function *newrb* is used to generate neural network. In a set of training data, the input parameters are the design variables of individuals (such as parameterized air vehicle geometry for RCS minimization) in the population, while the output parameters are their RCS performance values to be improved. There are some important points which require careful tuning during the use of neural network model in the present methodology, mVGA. The steps of the neural network coupled with the present mVGA methodology are as follows. First, the cost function calculations in the current population are performed to get their fitness values. Secondly, the neural network is trained after threshold generation point (the threshold is selected as the 10^{th} generation in the present implementation) by using the individuals in the current and previous populations and their fitness values. However, the training set needs to be limited. For the present test case, it is limited with the last *100* data set. For this training, individuals in the population are used as the input and their fitness values are used as the output. Lastly, the mutations are applied and the temporal neural network population is generated. It is taken 40 as N in the application. After the generation of the neural network population, the fitness values of network

population are estimated by using trained neural net. Then, some of them, which have greater fitness values than the elite individual (in the present use, the best two are selected) are placed in the new population to be used as candidates at the next step of the algorithm.

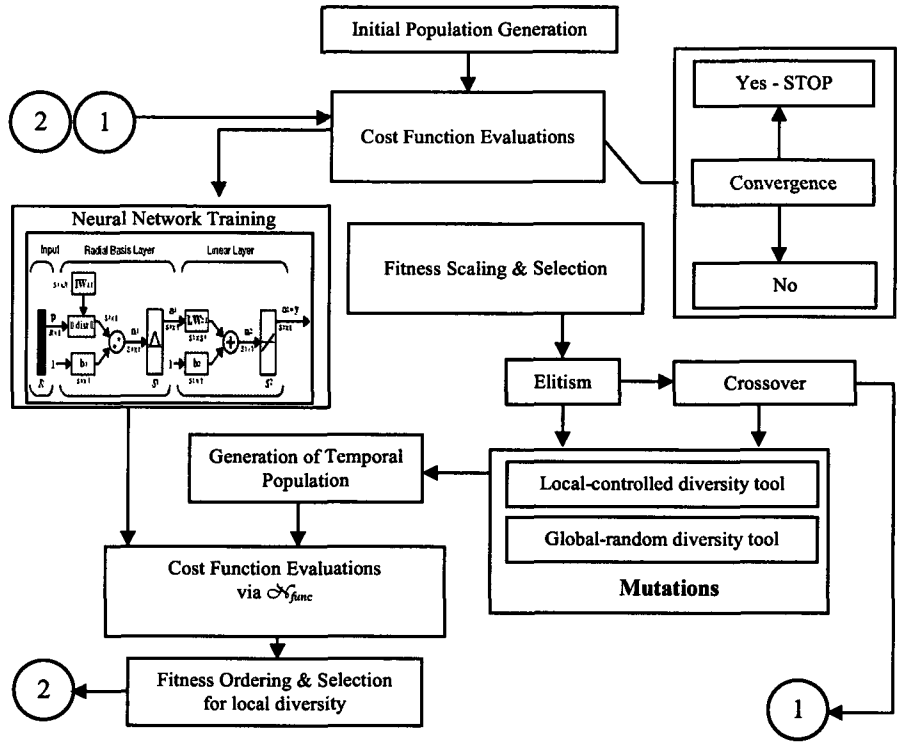

Figure 3.38 Flow chart of mVGA algorithm

Model Representation and RCS Solver

Radar cross section reduction problems of modified Harpy air vehicle is considered. A three dimensional model of vehicle is represented by using 727 points and 1415 triangular flat facets. The original model is depicted in Figure 3.39. The circular section of fuselage is parameterized by Bezier curve including four control points by considering y and z axes symmetries. The design parameters are selected as winglet angle which is between y axis and the winglet z axis, and three Bezier control points of fuselage including y and z coordinate values. The wing geometry and remaining control points of Bezier curve are kept fixed. Initial populations are generated by using a random number operator based on the original vehicle form.

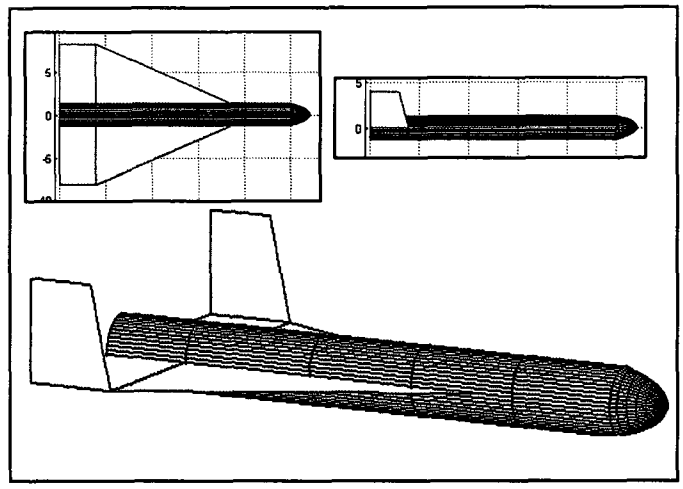

Figure 3.39 Original shape of Harpy air vehicle in different views

The radar cross section value of *ith* triangular flat plate, σ_i, in the plane φ equal to $0°$ and the total *RCS* value, σ_t, at φ, θ angles are given by the following equations, respectively:

$$\sigma_i(\theta,j)=\frac{4\pi(a_i b_i/2)^2}{\lambda^2}(\cos\theta)^2\left[\frac{(\sin\beta)^4}{\beta^4}+\frac{(\sin 2\beta-2\beta)^2}{4\beta^4}\right]$$
$$\beta=ka_i\sin\theta\cos j \qquad (3.39)$$
$$\sigma_t=\sum_{i=1}^{1415}\sigma_i(\theta,j)$$

where θ and φ angles define the direction of propagation of the incident waves, k is $2\pi/\lambda$, and λ is wavelength [108]. The triangle is oriented in Fig. 3.40. As approximate method, physical optics (PO) approximation based on the code, Pofacets 3.0.1 is used to obtain *RCS* data by computing the scattered field of the collection of triangular simple facets [109].

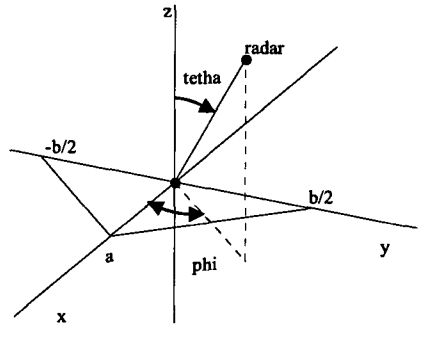

Figure 3.40 Coordinates of a perfectly conducting triangular plate

Fitness Function and Results

RCS reduction problem of modified Harpy air vehicle at incidence angles φ equal to $0°$ and θ equal to $0°$, observation angles φ from $0°$ to $360°$, θ equal to $90°$ cut; TM-polarized incident wave at a frequency of *10 GHz* was investigated via VGA and mVGA separately. The fitness value is based on bistatic RCS values. However, the fuselage volume is bound with a design fuselage volume value due to structural and missionary concerns, and the winglet angle is limited to certain values due to aerodynamical and stability concerns. The objective function value, f to be minimized,

$$\begin{aligned} \min f &= \left[C_r \max\{\sigma_i\} + C_a\left(\alpha^* - \alpha_2\right)^2 + C_v\left(V^* - V\right)^2 \right] \\ \alpha_2 &= \begin{cases} \alpha^* \; if \; 45^o < \alpha < 135^o \\ \alpha \; if \; otherwise \end{cases} \end{aligned} \tag{3.40}$$

where σ_i is the RCS values for the observation angles φ from $0°$ to $360°$, α^* is $90°$ which is the design winglet angle, V^* is the design volume value of the original vehicle form. The weighted factors are selected as 1, 0.1, and 100 respectively in the implementations. The features of VGA and mVGA are given in Table 3.9. In both optimization processes rank method is used as fitness scaling, roulette method is used as selection method. The population size is taken as 10, the number of maximum generation is taken as 100.

Table 3.9 The features of algorithms

Method	Mutation	Crossover
VGA	$fr_1 - w_1 - \beta_1$ *4 / 1/0.5*	Blx-α 0.5
m-VGA	$fr_1 - w_1 - \beta_1$ *4 / 1/0.5* $fr_2 - w_2 - \beta_2$ *2 /1/ 0.4*	Blx-α 0.5

The comparison of the results among VGA and mVGA is shown in Fig. 3.41. The plot gives objective function values belong to average best individuals, f_{abi}, (over independently 20 runs) against generations. The original and minimized air vehicle *RCS* distributions and optimized form (from one of 20 independent runs of mVGA optimization algorithm) are shown in Fig. 3.42 and Fig. 3.43 correspondingly. Fig. 3.44 shows the evolutionary fuselage sections.

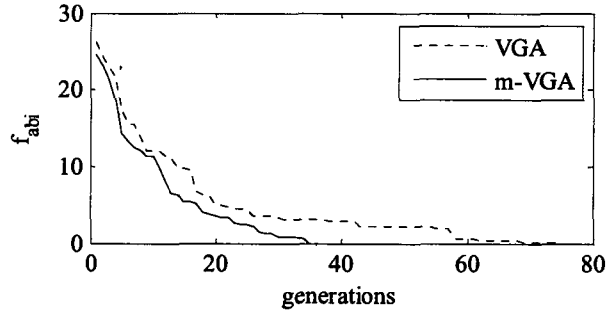

Figure 3.41 Fitness value change against generations

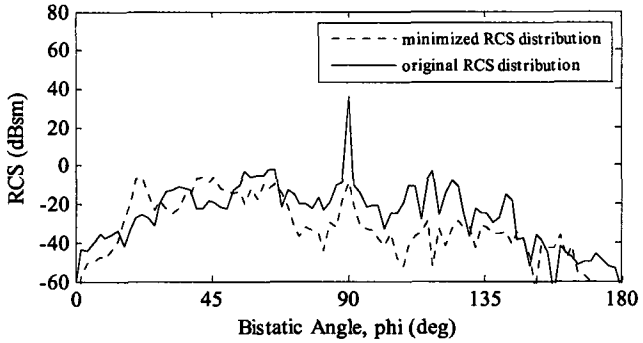

Figure 3.42 Original and minimized *RCS* distributions

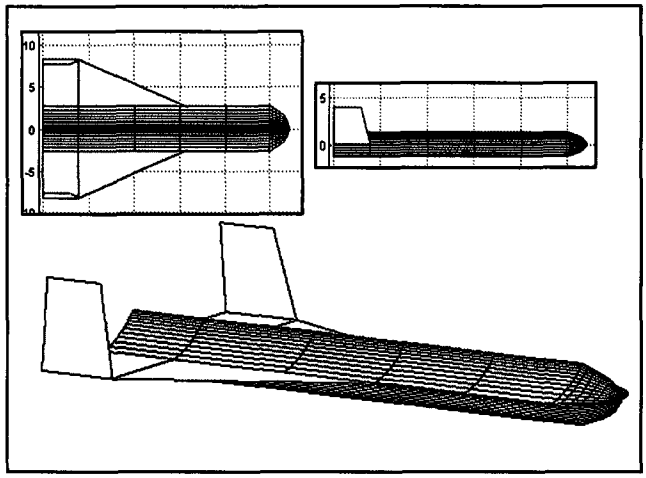

Figure 3.43 Optimized shape of Harpy air vehicle in different views

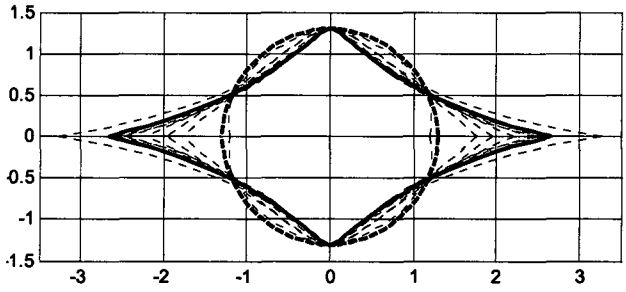

Figure 3.44 Evolutionary forms of fuselage cross section

As can be seen in Fig. 3.41 VGA reached the objective function value which is below zero within 750 objective function computations based on computational *RCS* code, while mVGA reached the same value within 350 *RCS* calculations. The difference among them is clear after the threshold generation point. This result indicates that mVGA approach decreased *RCS* solver calls about 53% with respect to VGA process. In Figure 3.42 it is plotted that the maximum *RCS* value of the original air vehicle form is around *32 dBsm* at φ is equal to *90°*. This value is significantly decreased below to zero at the end of 35th generation

of mVGA optimization process. The circular shape of fuselage is changed to triangular form having sharp corners. The winglet angle is increased to *98°* from original value *90°*.

Conclusion

The present study introduced a new neural network coupled multi-frequency vibrational genetic algorithm (mVGA) to speed up the optimization algorithm and overcome problems, such as, deficient diversity and premature convergence. Then, the present approach employed neural network concept to provide local but controlled diversity within the population in addition to random global diversity. The average best-individual-fitness values of employed algorithms were recorded for a fair comparison between them. To demonstrate its merits, mVGA was applied to RCS minimization test case.

3.5 Optimization of Parameters for Benchmark Test Functions

In this section, v-PSO and the comparative PSO algorithms are tested using different test functions including unimodal and multimodal benchmark functions. The selected test bed is given in Table 3.10. These functions have different characteristics which may result in different convergence criteria. Usually increase in dimensions for the test functions makes the convergence more difficult except for the Griewank function. An interesting phenomenon of Griewank function is that it is more difficult for lower dimensions than higher dimensions.

Table 3.10 Definitions of test functions

F	Test function	*f(x)*	search range	*x**	*f(x*)*
f_1	Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-30, 30]^D$	[0,0,...,0]	0
f_2	Cosine mixture	$\sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n}\cos(5\pi x_i)$	$[-1, 1]^D$	[0,0,...,0]	-0.1n
f_3	Ellipsoidal	$\sum_{i=1}^{n}(x_i - i)^2$	$[-n, n]^D$	[1,2,...,n]	0
f_4	Exponential	$-(\exp(-0.5\sum_{i=1}^{n} x_i^2))$	$[-1, 1]^D$	[0,0,...,0]	-1
f_5	Griewank	$1+\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]^D$	[0,0,...,0]	0
f_6	Rastrigin	$10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]^D$	[0,0,...,0]	0
f_7	Rosenbrock	$\sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^D$	[1,1,...,1]	0
f_8	Schwefel	$418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^D$	[420.9687..., 420.9687]	0
f_9	Zakharov	$\sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n}\frac{i}{2}x_i)^2 + (\sum_{i=1}^{n}\frac{i}{2}x_i)^4$	$[-5.12, 5.12]^D$	[0,0,...,0]	0

Parameter Settings for PSO Algorithms

The test procedure includes two different bundles. The first bundle contains the algorithm sensitivity to different function dimensions as D is equal to 10, 20, and 30 with fixed generation rate as 10,000 and fixed population rate as 20. The second bundle contains the behaviors of the algorithms related to different population sizes as 10, 20, and 30 with fixed dimensions as D is equal to 30 and fixed function evaluations as 200,000. All algorithms are run 100 times and the results are averaged. Peculiar settings are in the followings: c_1 and c_2 are equal to 2.05 for c-PSO; c_1 and c_2 are equal to 2, $w_{initial}$ and w_{final} are equal to 0.6, 0.2, respectively for w-PSO; c_1 and c_2 are equal to 2, $w_{initial}$ and w_{final} are equal to 0.6, 0.2, respectively, d_{low} is equal to 0.5 for g-PSO; c_1 is equal to 1.5, c_2 is equal to 2, w is equal to 0.05, f is equal to 10, A is equal to 1, elite count is 3 for v-PSO.

Fixed Iteration Rate-Fixed Population Size

The resulting values are tabulated in Table 3.11(a) and 3.11(b). The performances of four algorithms with different dimensions are tabulated in terms of means and averaged CPU times for nine test functions. The test runs were executed on an off-the-shelf laptop computer that has an Intel Centrino Duo processor with 32-bit accuracy. The mean value is calculated in accordance with 95% confidence interval ratio. The best results among four algorithms are shown in bold. The averaged global best individual values versus generations for D is equal to 30 are shown in Fig. 3.45.

Table 3.11(a) First test bundle results

		c-PSO		w-PSO	
f	D	Mean	t_{CPU}	mean	t_{CPU}
f_1	10	0.1040±0.0659	1.612	2.5e-15±9.9e-17	1.5219
	20	1.7684±0.2544	1.964	0.0231±0.0322	1.7922
	30	3.8334±0.3806	2.406	0.0266±0.0374	2.163
f_2	10	-0.9483±0.0158	1.557	-0.9970±0.0041	1.4688
	20	-1.533±0.0485	1.938	-1.9512±0.0162	1.74
	30	-2.0261±0.0692	2.318	-2.8167±0.0286	2.11
f_3	10	**0±0**	4.288	7.8e-33±1.2e-32	1.9672
	20	9.6e-30±7.0e-30	6.586	**2.4e-29±2.5e-29**	2.125
	30	6.3e-26±1.1e-25	7.893	**3.6e-28±2.8e-28**	2.458
f_4	10	-1±2.2e-18	1.468	-1±0	1.371
	20	-1±2.15e-17	1.78	-1±1.7e-17	1.593
	30	-1±5.14e-14	2.136	-1±5.4e-18	1.877
f_5	10	0.0915±0.0093	2.174	0.0742±0.0075	2.075
	20	0.0586±0.0478	2.577	0.0291±0.005	2.396
	30	0.2238±0.1616	3.043	0.0153±0.0035	2.835
f_6	10	8.6760±0.7852	2.133	5.1837±0.5259	2.055
	20	38.4849±2.1134	2.635	23.7496±1.6545	2.483
	30	84.6011±4.6683	3.146	53.5088±2.5935	3.004
f_7	10	1.2868±0.3710	1.714	3.5771±1.6449	1.707
	20	**2.6809±1.7206**	2.08	25.5461±9.9943	2.101
	30	**7.7424±4.5362**	2.524	36.2898±8.5554	2.549
f_8	10	836.04±61.8981	2.732	842.7311±59.72	2.185
	20	2.375e+3±112.69	4.222	2.1577e+3±101	3.093
	30	4.146e+3±177.38	5.451	3.62e+3±158.35	3.945
f_9	10	2.043e-247±0	2.136	2.3e-230±0	2.186
	20	1.7e-62±2.36e-62	2.35	1.0e-40±1.9e-40	2.369
	30	4.85e-20±9.4e-20	2.642	2.3e-11±2.6e-11	2.685

Table 3.11(b) First test bundle results

		g-PSO		v-PSO	
f	D	Mean	t_{CPU}	mean	t_{CPU}
f_1	10	3.258e-4±4.5e-5	1.634	**1.84e-15±2.9e-16**	1.717
	20	0.0013±1.8e-4	1.95	**2.84e-15±1.5e-16**	1.957
	30	0.0027±3.7e-4	2.307	**4.93e-15±3.4e-16**	2.296
f_2	10	-0.9998±3.1e-5	1.568	**-1±0**	1.644
	20	-1.9979±2.1e-4	1.883	**-2±0**	1.921
	30	-2.9942±5.1e-4	2.255	**-3±0**	2.26
f_3	10	1.308e-7±3.1e-8	1.852	2.5073e-22±1e-22	2.031
	20	4.8e-6±7.0e-7	2.081	1.0444e-16±2e-17	2.282
	30	2.98e-5±3.5e-6	2.378	9.2336e-14±1e-14	2.614
f_4	10	-1±1.5e-6	1.422	**-1±0**	1.55
	20	-0.9999±9.2e-6	1.625	**-1±3e-18**	1.771
	30	-0.9998±1.9e-5	1.901	**-1±1e-17**	2.039
f_5	10	0.0549±0.0046	2.125	**0.0209±0.006**	2.251
	20	0.0222±0.0041	2.483	**0.0026±0.002**	2.553
	30	0.0166±0.004	2.946	**8.8568e-4±0.001**	2.991
f_6	10	0.0398±0.0389	2.111	**0±0**	2.21
	20	2.8259±0.6476	2.537	**0±0**	2.621
	30	14.1188±2.24	3.033	**5.6843e-16±1e-15**	3.068
f_7	10	5.5279±2.9	1.6742	**1.0818±1.349**	1.922
	20	32.501±9.5	2.072	6.5075±4.554	2.317
	30	68.0813±19.3	2.495	31.5511±8.43	2.75
f_8	10	815.1561±64.3	2.232	**620.8131±50.4**	2.371
	20	2.1798e+3±120.9	3.021	**1.3384e+3±68.5**	3.128
	30	3.47e+3±149.3	3.781	**2.1395e+3±103.3**	3.971
f_9	10	1.819e-5±3.1e-6	2.132	**0±0**	2.924
	20	0.0011±9.6e-5	2.358	**7.6e-102±1e-101**	2.625
	30	0.01±6.15e-4	2.673	**2.5575e-41±3e-41**	2.915

The best performance belongs to v-PSO in 22 test cases over 27 test cases of the function evaluations $f_1, f_2, f_4, f_5, f_6, f_8, f_9$. The best performances of f_3 seem to be c-PSO and w-PSO in accordance with the dimension. The best performances of f_7 belong to c-PSO and v-PSO related to dimension. Usually c-PSO has shorter generation numbers to converge a local optimum than w-PSO and g-PSO. This is a typical result because c-PSO guaranties convergence to an optimum but not necessarily a global one. After a while it loses its diversity and tackles local optimum positions. However c-PSO shows the best performance in unimodal Rosenbrock test function. The second best performance of Rosenbrock function is v-PSO. w-PSO and g-PSO seems to have similar performance characteristics. g-PSO has the second best performance for f_1, f_2, f_5, f_6, f_8 function evaluations. c-PSO is the second best algorithm for f_3 and f_9. The CPU times required to 200,000 function evaluations for each algorithm are close to each other except f_8 function evaluation. Although c-PSO outperformed for 10-dimensional case of f_8 function the required CPU time for c-PSO is the longest one among others. The accuracy and the efficiency superiorities of v-PSO can be clearly seen in Fig. 3.1. It converges within 10^2 or 10^3 generation number to the values which other ones converge about 10^4 generation number.

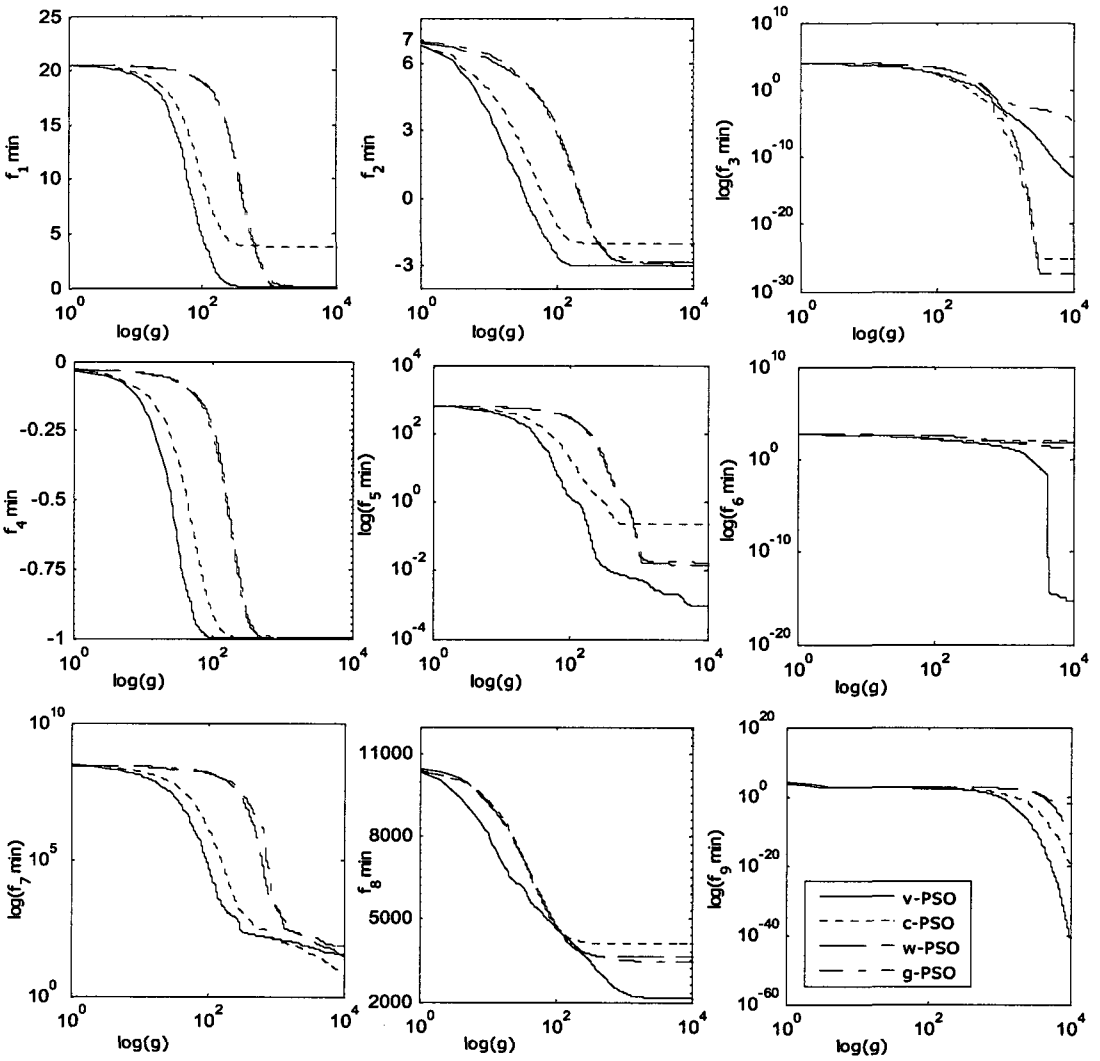

Figure 3.45 Averaged best function values versus generations for test functions.

Fixed Evaluation Rate-Fixed Dimension Rate

The averaged best function values versus population sizes such as *10*, *20*, and *30* for *D* is equal to *30* are shown in Fig. 3.46 and in Table 3.12. The mean value is calculated in accordance with *95%* confidence interval ratio. The best results among four algorithms are shown in bold. The similar results are observed comparing with the previous analysis. v-PSO has the best performance in the function evaluations of $f_1, f_2, f_4, f_5, f_6, f_8, f_9$. w-PSO has the best performance in f_3 function evaluation. c-PSO and v-PSO are the best related to population sizes in f_7 function test. Fig. 3.46 shows an interesting result in the relationship between the performance and the population size. Generally v-PSO has similar performances for all population sizes. It means that v-PSO provides better efficiency in lower population sizes. However, the other algorithms' performances significantly depend on the population. Typically, larger population size means better performance for them except for v-PSO. The

main reason for this result is that v-PSO provides enough diversity within the population even in low population size. The other algorithms need larger population size to have enough diversity in the population.

Table 3.12 Second test bundle results

f	P	c-PSO	w-PSO	g-PSO	v-PSO
		mean	mean	Mean	mean
f_1	10	9.412±0.6149	1.5913±0.4344	0.0803±0.096	**4.583e-15±3e-16**
	20	4.6684±0.4644	0.0902±0.067	0.0026±3.7e-4	**4.9738e-15±3e-16**
	30	2.2994±0.2776	0.0116±0.023	0.0027±3.9e-4	**5.0449e-15±3e-16**
f_2	10	-1.4844±0.0918	-2.6113±0.05	-2.996±3.3e-4	**-3±0**
	20	-1.9673±0.0767	-2.833±0.03	-2.9937±6.1e-4	**-3±0**
	30	-2.1842±0.071	-2.8714±0.03	-2.9928±7.3e-4	**-3±0**
f_3	10	0.0037±0.0037	**9.35e-26±9e-26**	4.4497e-5±4e-6	6.5765e-17±1e-17
	20	9.5e-27±1.2-e26	**4.58e-28±3e-28**	3.49e-5±1.3e-6	8.7906e-14±1e-14
	30	6.6e-27±1.3e-26	**1.70e-28±1e-28**	2.9598e-5±4e-6	2.8231e-12±5e-13
f_4	10	-1±8.8e-5	-1±1e-17	-0.9999±1e-5	**-1±1e-17**
	20	-1±2e-13	-1±3e-18	-0.9998±1.74	**-1±1e-17**
	30	-1±1e-16	-1±2e-18	-0.9997±2e-5	**-1±1e-17**
f_5	10	0.8154±0.275	0.0339±0.01	0.0223±0.005	**0.0016±0.002**
	20	0.1233±0.064	0.0167±0.003	0.0155±0.003	**0.0011±0.001**
	30	0.0431±0.01	0.0153±0.003	0.0153±0.003	**5.9021e-4±0.001**
f_6	10	104.3609±5.25	69.9753±3.63	12.1629±2.31	**0±0**
	20	81.8152±3.98	52.8919±2.42	13.5722±1.77	**1.1369e-15±1e-15**
	30	73.6367±3.7	44.8626±2.45	15.4020±1.98	**5.6843e-16±1e-15**
f_7	10	83.5977±18.45	105.1452±56.29	133.3642±62.5	**30.9127±7.5**
	20	**3.0857±0.75**	59.968±23.31	93.8152±53.8	28.2183±6.7
	30	**12.6086±4.19**	74.1099±20.47	72.2343±17.6	23.0551±5.2
f_8	10	4.6531e+3±142	3.9629e+3±161	3.6985e+3±194	**2.2246e+3±123**
	20	3.978e+3±149	3.5837e+3±147	3.5119e+3±188	**2.1888e+3±110**
	30	3.7372e+3±140	3.4045e+3±134	3.5136e+3±152	**2.2258e+3±107**
f_9	10	1.39e-5±2.7e-5	1.021e-9±2e-9	0.0059±3e-4	**6.39e-58±8e-58**
	20	4.38e-21±7e-21	2.57e-11±2e-11	0.0105±6e-4	**1.48e-41±2e-41**
	30	2.13e-19±2e-19	1.2e-7±1.2e-7	0.0161±0.001	**4.52e-32±4e-32**

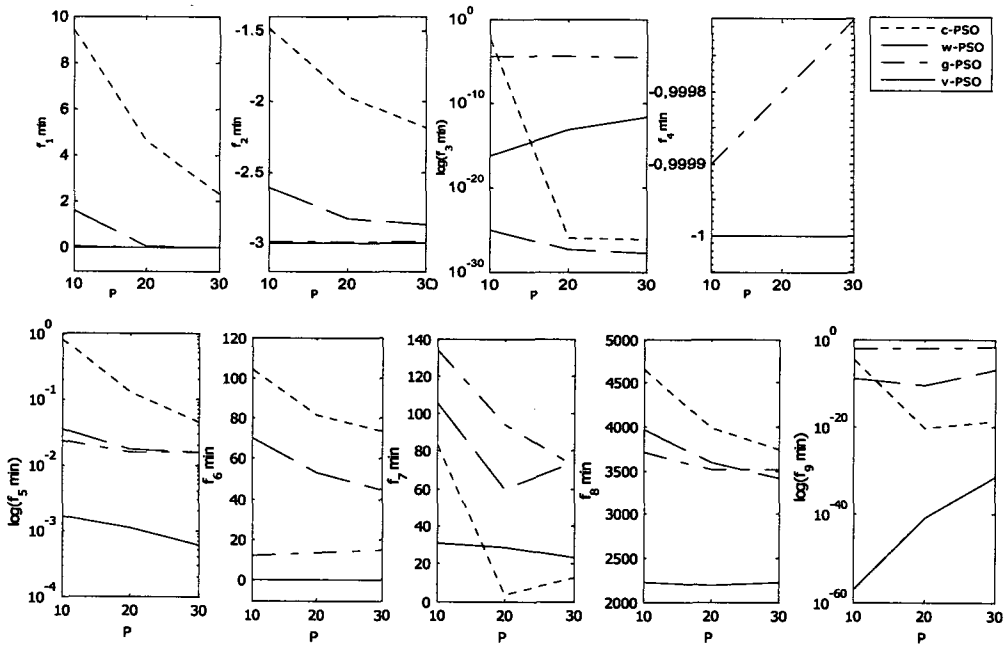

Figure 3.46 Averaged best function values versus population sizes for selected test functions.

Conclusions

Vibrational PSO is shown to be a good remedy to catch the big wave toward the global optimum. Its efficiency is clearly observed in the multi-modal functions tested herein. v-PSO tends to decrease the required objective function evaluations down to %50 or even less and it yields more accurate results than the other selected algorithms for comparison.

Further, the performance of these selected algorithms seems to depend on the population size. Their performance improves when the population size is increased, which, in turn, affects the diversity in the population. What is demonstrated in the present paper is that v-PSO is virtually independent of the population size. It does not need more individuals because the periodic mutations provide enough diversity in the swarm.

3.6 Aerodynamic Optimization of 2D Airfoil in Transonic Flow

New improved v-PSO algorithm variants are directly applied to an engineering problem and compared with other current three algorithms namely c-PSO, w-PSO, and g-PSO. The selected engineering issue is shock wave reduction problem in 2D aerodynamic optimization.

Background and Literature Review

Several different search methods are studied to solve shock wave reduction problem besides gradient-based algorithms, such as simulated annealing, ant colony optimization, artificial neural networks, and genetic algorithms to solve aerodynamic problems. Another approach that is gaining popularity is PSO algorithm. In recent studies PSO algorithms are applied to design wing in structural and aerodynamical objectives. Tan *et al.* [110] demonstrated the fast convergence capabilities of both the swarm and the evolutionary algorithm (EA) of airfoil shape optimization problems. In his comparative study the EA exhibited marginally higher efficiency over the swarm for single objective airfoil design problems while the swarm performed better on the multiobjective examples. Ng *et al.* [111] and Ray *et* al. [112] have demonstrated that PSO algorithm is capable of handling different forms of airfoil shape optimization problems. Venter et al. [113, 114] presented that the particle swarm optimization algorithm is able to reliably find the optimum design and it is capable of dealing with the unique challenges posed by multidisciplinary optimization as well as the numerical noise and truly discrete variables present in the optimization of a typical transport aircraft wing. He also demonstrated an example of using parallel computing in a synchronous and asynchronous manner. Khuranal *et al.* [115] showed that particle swarm optimization and artificial neural networks can be coupled in a surrogate model approach. However, these studies also showed that PSO algorithm needs to be more efficient than in the current forms. Our improved algorithms are used to solve shock wave reduction problems of

NACA64A410 airfoil at 2° angle of attack, Mach number with *0.75*, and *Re* number with $6.5x10^6$.

Airfoil Representation and Flow Solver

An airfoil curve is represented by two Bezier curves including upper and lower curves and each Bezier curve is governed by m control point which is equal to *12*. Initial populations are generated by using a random number operator based on NACA64A410 airfoil control points. The two control points (0, 0) and (1, 0) at the leading and trailing edges are fixed. It is commonly considered that the x_i control point axis is fixed, and the design parameters are only the y_i coordinates of the control points. Therefore, totally 22 design parameters for upper and lower curves are taken to be optimized.

During the optimization process relatively cheap computation solver is used. In this approach, the pressure coefficient (*Cp*) and pressure distribution around an airfoil for inviscid compressible flow can be determined by solving Euler equations instead of full Navier-Stokes equations. For 2D flows around an airfoil, using finite volume technique and Roe flux splitting scheme the time dependent compressible Euler equations were solved. Similarly computationally cheap elliptic partial differential equation solution method is used to generate a structured O-mesh grid domain around an airfoil. However, CFL3D v6.4, Navier-Stokes code for solving 2D flows on structured grids was used to perform the numerical validation of the simulations. CFL3D solves the time-dependent conservation law form of the Reynolds-averaged Navier-Stokes equations. The spatial discretization involves a semi-discrete finite-volume approach. Due to high Reynolds number, the flow was assumed fully turbulent. One-equation Spalart-Allmaras model was used to compute turbulent eddy viscosity. The commercial CFD meshing software, GridGen v15.06, was used to generate the computational grid with clustering in the normal direction. The resolution of the utilized C-type computational grid is 281x121. Normal spacing for the first grid line of the surface of the airfoil was 0.00001c where c is the airfoil chord length. The generated grid structures for original airfoil shape are depicted in Fig. 3.47(a) and 3.47(b).

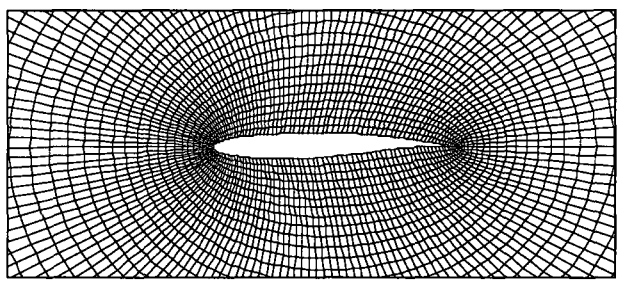

Figure 3.47(a) Structured O-type grid around NACA64A410 airfoil

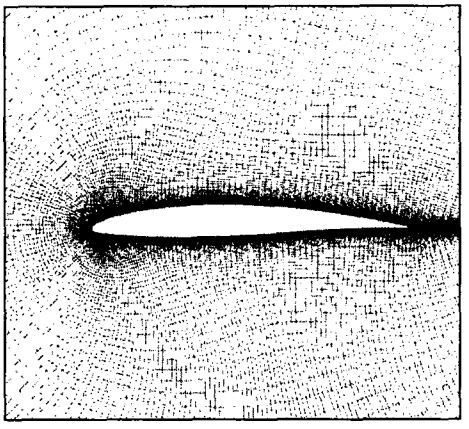

Figure 3.47(b) Structured C-type grid around NACA64A410 airfoil

Objective Function

The objective function values are based on drag (C_D) and lift (C_L) coefficients ratio. However, the maximum thickness of an airfoil is bound with design maximum thickness value. The objective function f to be minimized,

$$f = \left[\frac{C_D}{C_L} + 10\ \left(C_L^* - C_{L2}\right)^2 + 100\ \left(t^* - t\right)^2\right]$$
$$C_{L2} = \begin{cases} C_L^* & \text{if} \quad C_L \geq C_L^* \\ C_L & \text{if} \quad C_L < C_L^* \end{cases} \tag{3.41}$$

where C_L^* and t^* are the design lift coefficient, equal to *0.885*, and design maximum thickness ratio, equal to *0.1*, respectively. During the optimization processes the same flow conditions are taken into consideration.

Methodology

The baseline airfoil shape is optimized in accordance with given flow conditions by using five PSO algorithms including c-PSO, w-PSO, g-PSO, v-PSO, and mv-PSO. v-PSO does just include the first vibrational mutation operator given in Eq. (2.44). mv-PSO includes both vibrational mutation operators. It is also supported by neural network application expressed in terms of controlled diversity. The swarm size is selected as 10; the maximum generation number is selected as 100. The problem dimension is fixed to 22 as the control points of Bezier curves. The other features peculiar to the algorithms are given in Table 3.13. All algorithms are run 20 times and the averaged global best particle values versus computational fluid dynamics (CFD) calls are taken into consideration for fair comparison. Additionally, the schematic diagram of mv-PSO (it also includes v-PSO) is depicted in Fig. 3.48.

Table 3.13 The features of PSO algorithms

Algorithm	w	c_1	c_2	D_{low}	ξ	f	A
c-PSO	-	2.05	2.05	-	-	-	-
w-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	-	-	-	-
g-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	10^{-3}	6	-	-
v-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	-	-	f_1=15	A_1=0.5
mv-PSO	w_{ini}= 0.6; w_{end}= 0.2	2	2	-	-	f_1=15, f_2 =2	A_1=0.5; A_2 =0.5

In mv-PSO application, two networks are trained after threshold generation point (in the implementation threshold is selected as 10^{th} generation) by using the particles in the current and previous swarms and their cost function values. The first network is trained for C_L estimation. The second network is trained for C_D estimation. However the training set is limited to certain number (in the implementation the training set is limited with the last 100 data set which means the last 10^{th} generations).

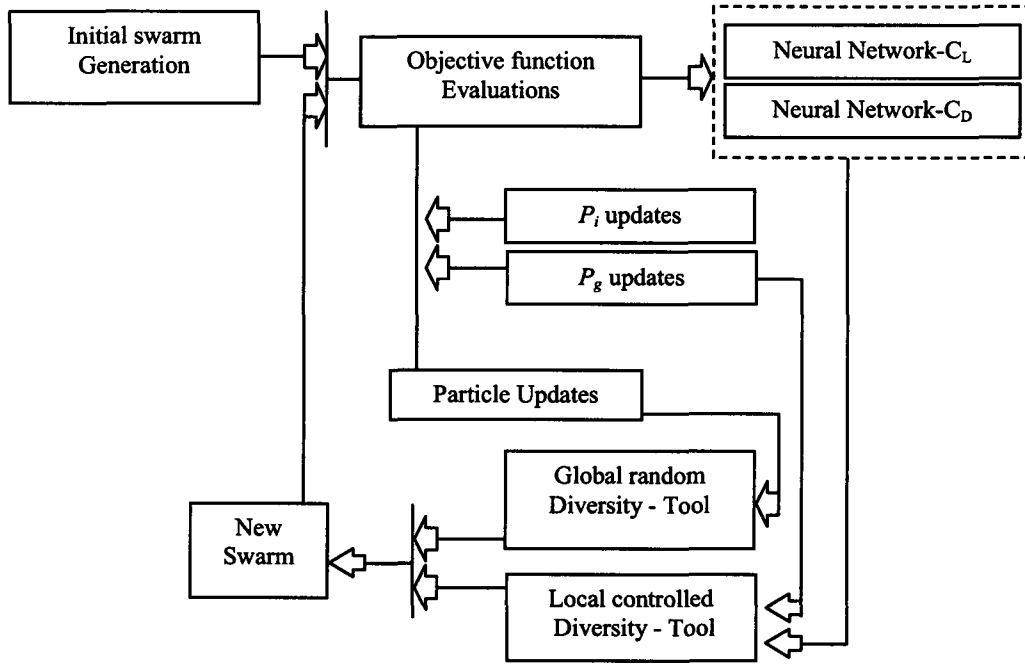

Figure 3.48 Flow chart of mv-PSO

Results

The optimization process results are depicted in Fig. 3.49. In this figure averaged global best particle objective function value and CFD calls are plotted. Among the classical PSO algorithms including c-PSO, w-PSO, and g-PSO the best performance belongs to w-PSO algorithm. It reaches the value of 24.08 in terms of 1/averaged f_{min} at 100^{th} generation which means 1000 CFD calls. c-PSO is the poorest algorithm reaching the value of 22.13 at 100^{th} generation. c-PSO is seriously suffered from the lack of diversity and trapped by local optimums. The performance of g-PSO is close to w-PSO. However, it is not good as w-PSO.

On the other hand, v-PSO outperforms the regular PSOs. It reaches the value of 24.1 at 56th generation which means 560 CFD calls. This result also means 44% decrease in required generations comparing with w-PSO. After 45th generation v-PSO shows the efficiency of mutation applications and escapes from trapped regions. At the end of optimization process v-PSO reaches the value of 24.89.

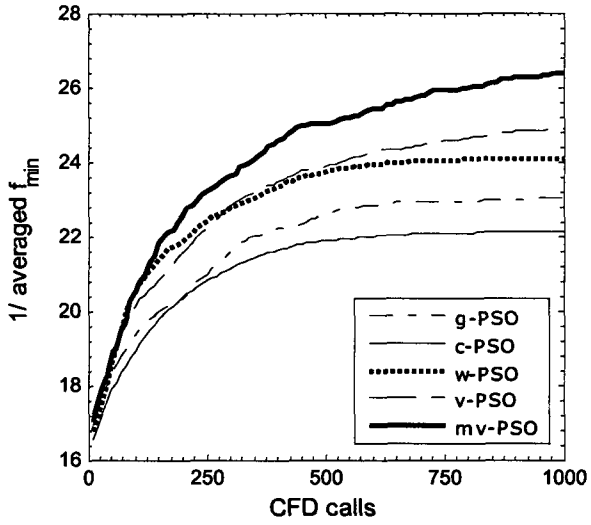

Figure 3.49 Optimization process results for PSO algorithms

mv-PSO does outperform all algorithms. After 13th generation the controlled diversity tool shows its affects on optimization process. This is an expected situation because the activation of neural nets starts with 10th generation. After this threshold generation, the local controlled diversity tool places new individuals into a swarm and these particles accelerate the optimization process. The random global diversity tool also provides global search capability to the algorithm. mv-PSO reaches the value of 24.19 at 36th generation. This means 64% decrease in required CFD calls comparing with w-PSO. Additionally, it reaches the value of 24.89 at 44th generation which means 440 CFD calls while v-PSO reaches the same value at 1000 CFD calls. In this comparison mv-PSO decreases 56% of the required CFD calls. mv-PSO provides the value of 26.39 at 100th generation.

In Table 3.14 and Table 3.15 the original and an example optimized airfoil features are presented in accordance with Euler solver and CFL3D solver, respectively. The validation results show that using Euler solver in optimization process provides correct direction for the search. This is beneficial because using a cheap CFD code is crucial for a short design cycle. According to the Euler solver results, the optimization process keeps the lift coefficient almost fixed while decreasing the drag coefficient by 40%. This results in a 65% increase in C_L/C_D ratio. On the other hand, CFL3D results show that the optimization process has a little bit deficiency on keeping the lift coefficient fixed. The process decreases lift almost by 3% as

compared with the original value. However, it also decreases the drag coefficient by 57% which is very effective. This results in a 122% increase in C_L/C_D ratio.

Table 3.14 Aerodynamic feature comparisons in accordance with Euler solver

	Euler solver		
	C_L	C_D	C_L/C_D
Original	0.8847	0.0578	15.3062
optimized	0.8860	0.035	25.31

Table 3.15 Aerodynamic feature comparisons in accordance with CFL3D solver

	CFL3D solver		
	C_L	C_D	C_L/C_D
Original	0.7036	0.03332	21.11
optimized	0.6814	0.01450	47

In Fig. 3.50(a) the original and optimized airfoil shapes are depicted. In Fig. 3.50(b) the typical pressure coefficient distributions are shown. Based on Euler solver results of *Cp* distribution here we can say that the optimization process almost eliminates the shock wave. However, CFL3D results show that the situation is a bit different. Fig. 3.51 shows Mach counters around airfoils based on CFL3D solution result. On the left side of this figure, Mach counters are cumulated around 0.6c on the original airfoil surface. This area is a strong normal shock wave region. On the left side, we see that strong normal shock is divided and the first part is located around 0.5c on the optimized airfoil surface. This is a relatively softer shock wave. After a soft shock wave, the flow accelerates and goes down through the trailing edge. We recall that this result is an example result. In stochastic optimization process the results are almost always different due to a random nature of algorithms. Therefore, each time different airfoil shape is formed at the end of optimization process, as shown below.

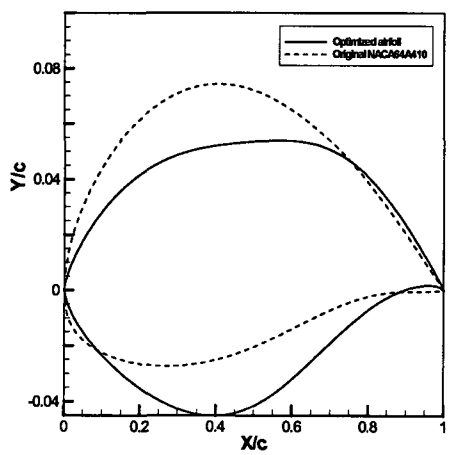

Figure 3.50(a) Original and typical optimized airfoil shapes.

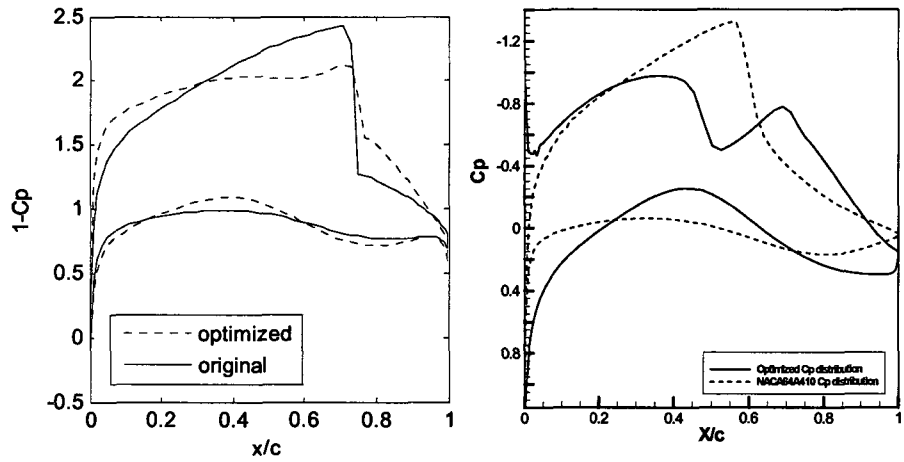

Figure 3.50(b) Typical pressure coefficient distributions for original and optimized airfoils (Euler based on the left side, CFL3D based on the right side)

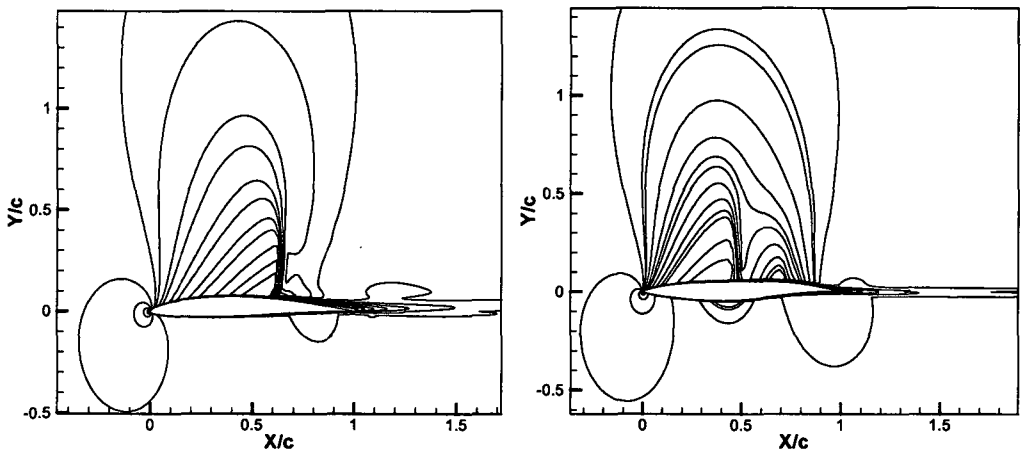

Figure 3.51 Mach number counters around original (left side) and optimized airfoils.

Conclusions

In this application v-PSO decreases the required objective function evaluations down to %44 compared with the closest result achieved by w-PSO. However, the diversity provided by vibrational mutation operator is a global random diversity. In some cases this type of diversity may not be enough to catch the correct wave. In addition to global random diversity the local controlled diversity combining with elite particles in the swarm and neural nets may help faster convergence. mv-PSO is a promising algorithm for faster and more accurate results in long but real optimization processes. This neural network supported algorithm provides a 64% decrease in required CFD calls. Both v-PSO and mv-PSO are still population based evolutionary algorithms and promising candidate solution methods for other optimization problems.

3.7 Optimization of AFC on an Airfoil at Transonic Regime

Introduction

Engineering design of an airplane wing roughly consists of three stages: 1) conceptual design such as determining the span length, maximum thickness, taper ratio, sweep angle, and aspect ratio; 2) preliminary design including airfoil shape and its optimization; and 3) detailed design including the detailed plan for manufacturing of the wing. In a preliminary design phase, designers start with a good baseline design and then improve its performance by using some optimization techniques. For transonic commercial aircraft wing design, the primary goal is to improve the wing performance at the cruise conditions without severe penalty at off-design conditions. The main issue in this stage is a shock wave reduction problem. A wave drag is caused by the formation of shock waves around the wing. Shock waves radiate away a considerable amount of energy that is experienced by the aircraft as drag. The magnitude of the rise in drag is impressive, typically peaking at about four times the normal subsonic drag. Due to this energy consumption, it is highly beneficial to eliminate the effects of shock wave at design phases. It may be worth recalling that a mere 10% reduction in the total drag of an aircraft translates into a saving of billions in annual fuel cost for the commercial aircraft in the world [116]. Optimization is a key concept to reduce the effects of shock wave and it is heavily based on the reforming of an airfoil shape in a passive way. Although, passive control enhances the aerodynamic performance at the design point, it may have harmful effects at off-design conditions. Furthermore, it may not be practically applicable for the current designs in service. An Active flow control may be an alternative remedy. In this context, active flow control may offer new solutions for the performance maximization of existing designs [117].

Active Flow Control (AFC) has been the subject of the major research areas in fluid mechanics for more than the past two decades. There are lots of studies to enable active flow control benefits on airfoils at subsonic speeds [118-120]. However, there are a few studies related to shock wave reduction problems based on active flow control at transonic flows. The small disturbance close to the shock can result in large changes in the aerodynamics of the airfoil at transonic and supersonic speeds [121]. Experimental study by Smith and Walker [122] has shown that applications of strong suction in the strong adverse pressure gradient increases lift. Qin and Zhu [123] showed that lift could be increased by application of suction in the vicinity of the shock; however, this is obtained with an increase in drag. Injection of momentum accelerates the inviscid outer flow over the airfoil ahead of the shock induces weak compression waves that soften the adverse pressure gradient [124]. In those studies, only one actuator was used for suction/blowing. As a result of this study, it was seen that one

actuator is not enough to obtain the desired goal. Vadillo and Ramesh [125] have investigated the same result that three synthetic jet actuators on the upper surface of the airfoil were needed to achieve the goal of enhancement in L/D with minimum change in transonic drag. Although AFC provides such degrees of improvements, the sensibility of aerodynamic performance to active flow control design parameters makes it a nontrivial and expensive problem [126]. Therefore, the designer has to optimize a number of different parameters related to active flow control. Yagiz and Kandil [127] evaluated the capability of weakening the shock waves to improve the aerodynamic performance in transonic conditions by using surface suction/blowing on airfoils via gradient-based optimization process. They selected the suction/blowing speed and angle as design parameters for different number of control points keeping the locations fixed. However, the missing parameter, the location of actuator is also an important design variable for such a case.

The desired goal in this study is to improve the aerodynamic performance on airfoils in transonic conditions by using optimization of surface suction/blowing parameters. For this reason a non-gradient based global optimization algorithm, Particle Swarm Optimization method is used to search the optimal design variables. We also compared the results by using gradient-based algorithm for selected cases. During the optimization process time-dependent turbulent Spalart-Allmaras model in CFL3D developed at NASA Langley Research Center is used for solving 2D flows on structured grids. Computations were performed for flow past a NACA64A010 airfoil in transonic flow at Mach 0.78, angle of attack 0.5° with and without AFC. The NACA64A010 airfoil was tested by Smith and Walker [122] at transonic speeds with surface suction. This test was used to validate the numerical study.

Numerical Model for Flow Analysis

An existing Navier-Stokes solver, CFL3D v6.4, for solving 2D/3D flows on structured grids was used to perform the numerical simulations. CFL3D solves the time-dependent conservation law form of the Reynolds-averaged Navier-Stokes equations. An implicit approximately factored, finite volume, upwind and multigird algorithm is used for the solution. Due to high Reynolds number, the flow was assumed fully turbulent. One-equation Spalart-Allmaras model was used to compute turbulent eddy viscosity. The governing equations, which are the thin-layer approximations to the three-dimensional time-dependent compressible Navier-Stokes equations, can be written in terms of generalized coordinates as

$$\frac{\partial \hat{\boldsymbol{Q}}}{\partial t}+\frac{\partial(\hat{\boldsymbol{F}}-\hat{\boldsymbol{F}}_v)}{\partial \xi}+\frac{\partial(\hat{\boldsymbol{G}}-\hat{\boldsymbol{G}}_v)}{\partial \eta}+\frac{\partial(\hat{\boldsymbol{H}}-\hat{\boldsymbol{H}}_v)}{\partial \zeta}=0 \tag{3.42}$$

A general, three-dimensional transformation between the Cartesian variables (x, y, z) and the generalized coordinated (ξ, η, ζ) is implied. The variable J represents the Jacobian of the transformation:

$$J = \frac{\partial(\xi,\eta,\zeta,t)}{\partial(x,y,z,t)} \tag{3.43}$$

In Eq. (3.42), $\boldsymbol{Q}$ is the vector of conserved variables, density, momentum, and total energy per unit volume, such that

$$\hat{\boldsymbol{Q}} = \frac{\boldsymbol{Q}}{J} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \tag{3.44}$$

The inviscid flux terms are

$$\hat{\boldsymbol{F}} = \frac{\boldsymbol{F}}{J} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ \rho U w + \xi_z p \\ (e+p)U - \xi_t p \end{bmatrix} \hat{\boldsymbol{G}} = \frac{\boldsymbol{G}}{J} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ \rho V w + \eta_z p \\ (e+p)V - \eta_t p \end{bmatrix} \hat{\boldsymbol{H}} = \frac{\boldsymbol{H}}{J} = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho W u + \zeta_x p \\ \rho W v + \zeta_y p \\ \rho W w + \zeta_z p \\ (e+p)W - \zeta_t p \end{bmatrix} \tag{3.45}$$

The contra variant velocities are given by

$$\begin{aligned} U &= \xi_x u + \xi_y v + \xi_z w + \xi_t \\ V &= \eta_x u + \eta_y v + \eta_z w + \eta_t \\ W &= \zeta_x u + \zeta_y v + \zeta_z w + \zeta_t \end{aligned} \tag{3.46}$$

The viscous flux terms are

$$\hat{\boldsymbol{F}}_v = \frac{\boldsymbol{F}_v}{J} = \frac{1}{J}\begin{bmatrix} 0 \\ \xi_x\tau_{xx} + \xi_y\tau_{xy} + \xi_z\tau_{xz} \\ \xi_x\tau_{xy} + \xi_y\tau_{yy} + \xi_z\tau_{yz} \\ \xi_x\tau_{xz} + \xi_y\tau_{yz} + \xi_z\tau_{zz} \\ \xi_x b_x + \xi_y b_y + \xi_z b_z \end{bmatrix} \hat{\boldsymbol{G}}_v = \frac{\boldsymbol{G}_v}{J} = \frac{1}{J}\begin{bmatrix} 0 \\ \eta_x\tau_{xx} + \eta_y\tau_{xy} + \eta_z\tau_{xz} \\ \eta_x\tau_{xy} + \eta_y\tau_{yy} + \eta_z\tau_{yz} \\ \eta_x\tau_{xz} + \eta_y\tau_{yz} + \eta_z\tau_{zz} \\ \eta_x b_x + \eta_y b_y + \eta_z b_z \end{bmatrix}$$

$$\hat{\boldsymbol{H}}_v = \frac{\hat{\boldsymbol{H}}_v}{J} = \frac{1}{J}\begin{bmatrix} 0 \\ \zeta_x\tau_{xx} + \zeta_y\tau_{xy} + \zeta_z\tau_{xz} \\ \zeta_x\tau_{xy} + \zeta_y\tau_{yy} + \zeta_z\tau_{yz} \\ \zeta_x\tau_{xz} + \zeta_y\tau_{yz} + \zeta_z\tau_{zz} \\ \zeta_x b_x + \zeta_y b_y + \zeta_z b_z \end{bmatrix} \tag{3.7}$$

The shear stress and hear flux terms are defined in tensor notations (summation convention implied) as

$$\tau_{x_i x_j} = \frac{M_\infty}{\mathrm{Re}_{\tilde{L}_R}}\left[\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)+\lambda\frac{\partial u_k}{\partial x_k}\delta_{ij}\right]$$
$$b_{x_i} = u_j\tau_{x_i x_j} - \dot{q}_{x_i} \tag{3.48}$$
$$\dot{q}_{x_i} = -\left[\frac{M_\infty \mu}{\mathrm{Re}_{\tilde{L}_R}\,\mathrm{Pr}(\gamma-1)}\right]\frac{\partial a^2}{\partial x_i}$$

The pressure is obtained by the equation of state for a perfect gas

$$p = (\gamma-1)\left[e-\frac{\rho}{2}(u^2+v^2+w^2)\right] \tag{3.49}$$

The above equations have been nondimensionalized in terms of the free-stream density, ρ_∞ , the free-stream speed of sound, a_∞, and the free-stream molecular viscosity, μ_∞. The chain rule is used to evaluate derivatives with respect to (x, y, z) in terms of (ξ, η, ζ). Consistent with the thin-layer assumption, only those derivatives in the direction normal to the wall (ζ) are retained in the shear stress and heat flux terms. Eq. (3.42) is closed by the Stokes hypothesis for bulk viscosity ($\lambda+2\mu/3=0$) and Sutherland's law for molecular viscosity. The details of the code can be found in the reference by Rumsey *et al.* [128].

The actuator is modeled as a boundary to accurately compute the mass flow through a solid boundary. A constant rate of change in mass flow, C_{qu}, is established. Mass flow coefficient, C_q, is gradually increased from zero to a constant value within T_{st} time for stable initiation and then C_q will remain fixed. It is also defined as the ratio between the boundary and free stream mass flows.

$$C_q = \int_0^{T_{st}} C_{qu}\,dt$$
$$C_q = \frac{(\rho u)}{(\rho u)_\infty} \tag{3.50}$$

A two-level multi-grid technique was used to achieve the convergence acceleration. The calculation is initiated from a steady state solution obtained for the flow in the absence of any jets. Then, the control cases are started from this steady solution until the residual value goes to 10^{-9}.

Optimization Algorithms

In this study two different optimization algorithms are used. The first one is v-PSO. The details of the algorithm were given in the previous section. During the optimization processes the swarm size, S, is taken as *10*, the inertia weight, w, is decreased linearly starting from *0.6* and ending to *0.3* related to maximum iteration number, G which is equal to 100. The

mutation frequency, f_m, is equal to *5*, scale factor, A, is equal to *0.5*, c_1 is equal to *2*, c_2 is equal to 2. In computational phase two-level parallelization is implemented. The first parallelization is applied in the swarm. Each particle objective function value within the swarm is computed on different processors in a parallel way. The flow solver code is also appropriate for parallel computing. Therefore, the second level parallelization is applied in flow solver.

As a second optimization algorithm, sequential quadratic programming (SQP) is used. SQP is one of the most powerful methods among the mathematical nonlinear programming techniques. However, it has a major distinguishing disadvantage convergence toward local optimum point. Non-gradient based methods such as PSO have the quality to escape from the local minimums. To escape from the local minimum, many initial points for all cases were used in this study. In this method, we first generate a quadratic approximation to the objective function using the Taylor series expansion of the objective function. The solution of the quadratic problem is used to determine the search direction at a given point. The quadratic problem is expressed as follows:

$$\begin{aligned} &\min \nabla f(x)^T s + \frac{1}{2} d^T H s \\ &\text{subject to} \\ &\qquad \nabla g_j(x)^T s + g_j(x) \le 0 \end{aligned} \tag{3.51}$$

The search direction vector, s, is the design variable for this quadratic problem. The matrix H is initially the identity matrix, which is a positive definite matrix. To approach the Hessian of the objective function H is updated on the subsequent iterations [129]. Gradient based optimization process is summarized as follows:

- $i = 0, x = x^0$
  - $i = i + 1$
    - Calculate $f(x^{i-1}), g_j(x^{i-1}), j = 1, m$
    - Identify the set of critical constraints, J_c
    - Evaluate $\nabla f(x^{i-1}), \nabla g_j(x^{i-1}), j \in J_c$
    - Determine a search direction, s
    - Investigate a one dimensional search to find, α
    - Set $x^i = x^{i-1} + \alpha s$
  - Check for convergence, if not go to second step.
- Stop

Design Parameters and Objective Function

A nonlinear-constrained optimization problem for an airfoil can be expressed as follows:

$$\begin{aligned} &\min_{X\in R^D} f = \frac{C_D}{C_L} \\ &\text{subject to :} \\ &\quad -C_L + C_L^* \le 0 \\ &\quad x^L \le x \le x^U \end{aligned} \tag{3.52}$$

where C_L^* is the design lift coefficient. In PSO algorithm the cost function description can be converted into unique equation by using weighting number such as

$$f_{x\in R^D} = \frac{C_D}{C_L} + 10\left(C_L^* - C_{L2}\right)^2;\ C_{L2} = \begin{Bmatrix} C_L^* & if & C_L \ge C_L^* \\ C_L & if & C_L < C_L^* \end{Bmatrix} \tag{3.53}$$

In both optimization processes the following design parameters are used: mass flow coefficient, C_q, centre location of actuator, x_c, and suction/blowing angle relative to the local tangent, β. Depending on the number of actuators used in AFC the design parameter vector x is composed of different combinations based on given parameters. In the first three test cases the centre locations of the actuators are kept fixed, the velocities and angles are selected as design variables. However, for the last three cases the locations of the actuators are also selected as additional design variables. For all cases the width of the actuator is kept fixed as *0.035c* used in experiment. The design variables and fixed locations for the first three cases are depicted in Fig. 3.52. The first location, *0.7075c*, is selected as a validation point. This location is used by Smith and Walker [128] in their experimental studies as the hinge line of the trailing edge flap. The third location, *0.5125c*, is placed in behind of the shock wave. Under selected flow conditions the center of normal shock wave is occurred about *0.5100c*. The second location, *0.61775c*, is placed between the first and the third locations.

Grid Generation

Two dimensional, 10% thick symmetric NACA64A010 airfoil was utilized. Two different grid morphologies are used during the simulations. The first one is the grid with clustering in the normal direction and in the vicinity of jets to resolve the details of the flow for the first three cases. The second grid is used for the last three cases and the travel area of the actuator locations between *0.5500c* and *0.9600c* is made dense. The resolution of the first utilized C-type computational grid is *407x121*; the second one is *449x121*. Normal spacing for the first grid line of the surface of the airfoil was *0.000001c*. Fig. 3.53 shows the grids

used in the simulations. The grid is divided into four sub blocks to implement the parallel computing for faster computation.

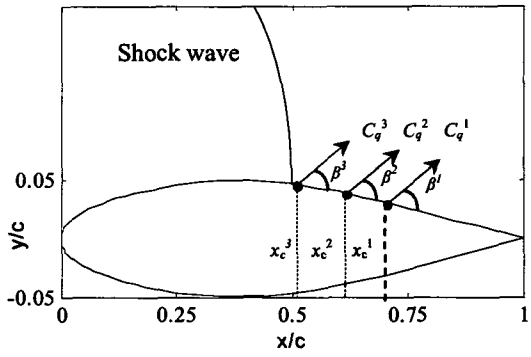

Figure 3.52 The design parameters for active flow control

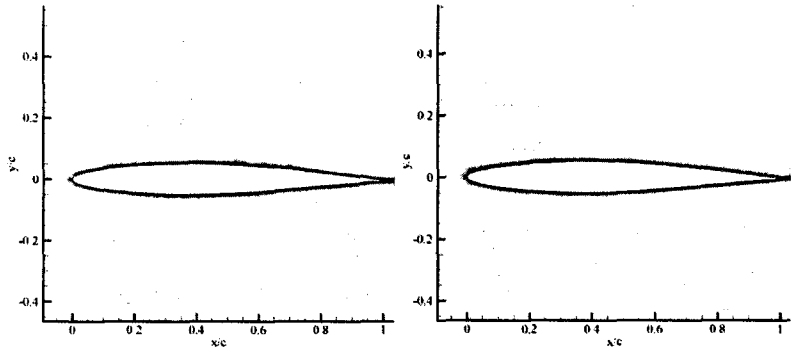

Figure 3.53 Computational grids used in simulations

The NACA64A010 airfoil was tested by Smith and Walker [128] at different transonic speeds with surface mass injection downstream of the hinge line of the trailing edge flap. The validation cases used in this study had a Reynolds number of 2.9 million based on airfoil chord, $M_\infty = 0.78$, $\alpha = 0.5°$, corresponding to one of the wind tunnel experiments. The region of suction was located between $0.69c$ and $0.725c$ which is downstream of the shock position without active flow control. The suction coefficient was -0.06429 and the suction angle was 84° to the airfoil surface.

Fig. 3.54 shows the comparison of the pressure distributions for both computation and experiment with and without flow control. Also, the solution sensitivity to the grid used is illustrated in Table 3.16. In numerical tests, three sets of grid and another computational result given by Quin *et al.* [130] have been used. Normalized first cell height, y^+ values, based on the height of the first wall-bounded cell, are below unity for all meshes considered here. The solutions obtained on the course and fine grids are reasonably good. The results obtained on these three different grid sizes are reasonably grid-converged results and prove the solution sensitivity. The results are seen to be in qualitative agreement with experiment. As can be

seen from Table 3.16 the measured lift for the controlled case and drag for the non-control values are different than the present results. The reason may be due to the fixtures mounted on the airfoil in the experiment, which was not calculated in the computation.

Table 3.16 Grid sensitivity for NACA64A010 aerofoil test cases

	grid size	C_L	C_D
without control	449 * 121	0.2121	0.01050
	407 * 121	0.2111	0.01059
	241 * 69	0.2121	0.01071
	Experiment [7]	0.2000	0.01300
	Computation [16]	0.2166	0.01110
with control	449 * 121	0.2821	0.01406
	407 * 121	0.2773	0.01376
	241 * 69	0.2751	0.01378
	Experiment[7]	0.2400	0.01400
	Computation[16]	0.2795	0.01380

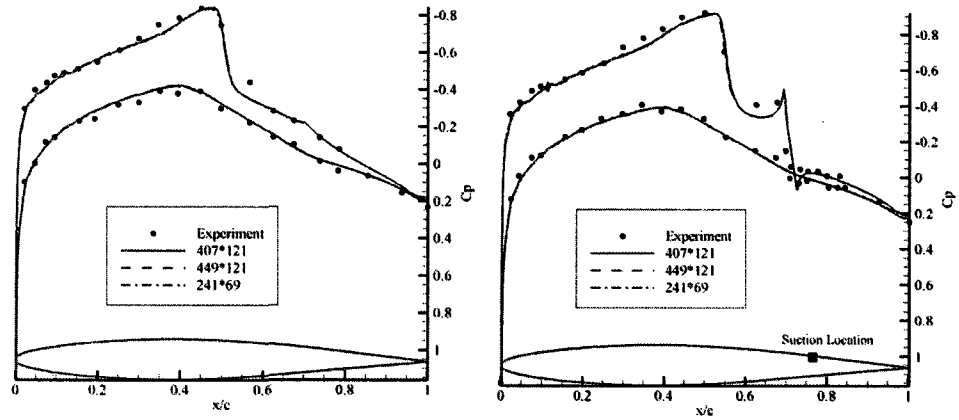

Figure 3.54 Comparison of pressure distributions for NACA64A010 aerofoil without and with suction

Results - Fixed location(s) Optimizations

The effect of mass flow coefficient and angle for different number of actuators at fixed locations are investigated. For that reason three cases depending on the number of actuators are studied. At first only one actuator is considered. Then, the number of actuators is increased to two and finally three actuators are used to control the flow on the airfoil upper surface.

One-actuator Optimization

The center of suction/blowing actuator is placed on *0.7075c* point. The design parameter vector and bounds are described as follows

$$\begin{gathered} x = [C_q \; \beta]^T \\ -0.1 \le C_q \le 0.025 \\ 3° \le \beta \le 176° \end{gathered} \tag{3.54}$$

Due to the global nature of the PSO algorithm and relatively small number of design parameters, the optimization process takes only two generations. At the end of optimization C_q has taken the maximum suction velocity as *-0.1* and β became the minimum angle as *3°* which is almost parallel to the local airfoil surface. The changes of aerodynamic coefficients such as C_L and C_D versus computational fluid dynamics (CFD) calls belong to the best particle are depicted in Fig. 3.55. According to these figures both coefficients are increased due to suction operation. However the aerodynamic performance based on L/D is also increased. The reason is that the increment in C_L is larger than the increment in C_D. After suction operation at optimal values C_L is increased *9.76%*, on the other hand C_D is increased *2.17%*. As a result, the re-described objective function value (1/*f*) increases by *7.43%*. The change in aerodynamic performance versus CFD calls in PSO is shown on the left side of Fig. 3.56. The right figure in Fig. 3.56 depicts the same objective function change versus CFD calls in SQP optimization process. In this gradient based optimization process totally six different initial points (i.p.) are tested to escape from local optimums. After these six SQP optimization processes the same optimal values found in PSO process are determined. In PSO process a total of *20* CFD calls are needed to reach the optimal values. However in SQP processes a total of *66* CFD calls are needed to get the same optimal values. Additionally SQP processes are done in accordance with *try and error* approach. Therefore, the sequential computations are executed. Although sequential computations are implemented only the best SQP process and PSO process are compared in terms of computation time on the lower part of Fig. 3.56. One CFD call takes approximately *24* minutes based on Intel 2.4-gigahertz quad-core processor. According to this figure PSO approach is much more time-efficient than SQP approach.

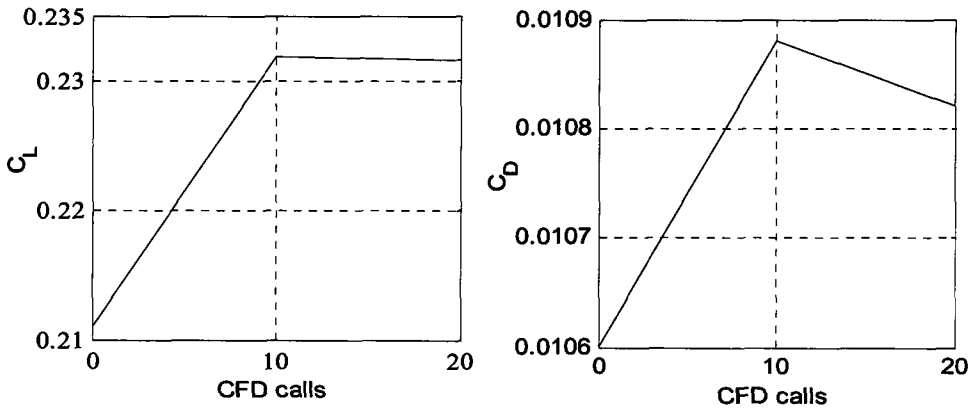

Figure 3.55 The change of aerodynamic coefficients during the generations

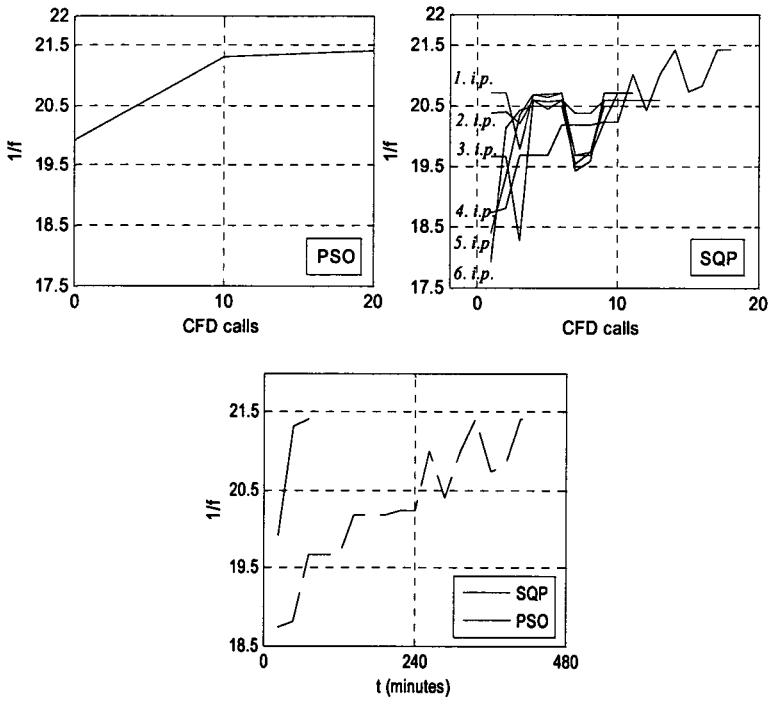

Figure 3.56 The change of aerodynamic performance in PSO and SQP

Two-actuator Optimization

The centers of suction/blowing actuators are placed on *0.7075c* and *0.61775c* points. The design parameter vector and bounds are described as follows

$$\begin{aligned} \boldsymbol{x} &= [C_q^{\,1}\ \beta^1\ C_q^{\,2}\ \beta^2]^T \\ -0.1 &\le C_q^{\,i} \le 0.025 \\ 3^\circ &\le \beta^i \le 176^\circ \end{aligned} \Big|^{i=1,2} \tag{3.55}$$

PSO optimization process takes only four generations. At the end of optimization $C_q^{\,1,2}$ have taken the maximum suction velocity as *-0.1* and $\beta^{1,2}$ became the minimum angle as *3°*. The changes of aerodynamic coefficients such as C_L and C_D versus CFD calls belong to the best particles are depicted in Fig. 3.57. Similar to previous operation both coefficients are increased due to suction operations. After suction operations at optimal values C_L is increased *16.86%*, on the other hand C_D is increased *3.21%*. As a result $1/f$ is increased *13.25%*. The change in aerodynamic performance versus CFD calls in PSO is shown on the left side of Fig. 3.58(a). The right figure in Fig. 3.58(a) depicts the same objective function change versus CFD calls in SQP optimization process. Totally five different initial points are tested to escape from local optimums. After these five SQP optimization processes the same optimal values found in PSO process are determined. In the PSO process a total of *40* CFD calls are needed to reach the optimal values. However in SQP processes a total of *48* CFD calls are needed to get the same optimal values. Although sequential computations are implemented

only the best SQP process and PSO process are compared in terms of computation time in Fig. 3.58(b). Similar to previous optimization process the PSO approach is much more time-efficient than the SQP approach.

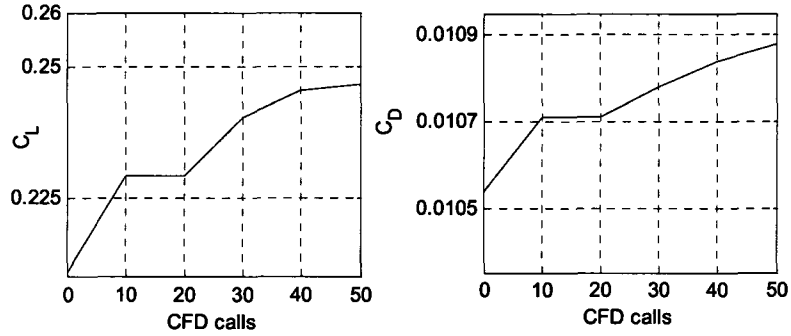

Figure 3.57 The change of aerodynamic coefficients during the generations

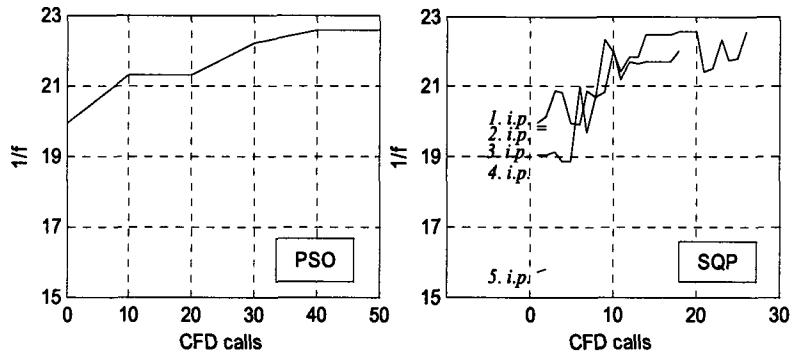

Figure 3.58(a) The change of aerodynamic performance in PSO and SQP

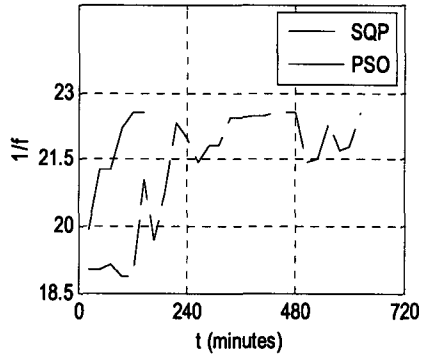

Figure 3.58(b) The change of aerodynamic performance versus time in PSO and SQP

Three-actuator Optimization

The centers of suction/blowing actuators are placed on *0.7075c*, *0.61775c*, and *0.5125c* points. The design parameter vector and bounds are described as follows

$$\begin{aligned} x &= [C_q^1\ \beta^1\ C_q^2\ \beta^2\ C_q^3\ \beta^3]^T \\ -0.1 &\le C_q^i \le 0.025 \\ 3^\circ &\le \beta^i \le 176^\circ \end{aligned} \Big|^{i=1,2,3} \qquad (3.56)$$

PSO optimization process takes eleven generations. At the end of optimization $C_q^{1,2,3}$ has taken the suction velocity as *-0.0864*, *-0.1*, and *-0.0977*, respectively. $\beta^{1,2,3}$ became the minimum angle as *3°*. The changes of aerodynamic coefficients such as C_L and C_D versus CFD calls belong to the best particle are depicted in Fig. 3.60. After suction operations at optimal values C_L is increased *22.03%*, on the other hand C_D increases by *4.82%*. As a result $1/f$ increases by *16.46%*. The change in aerodynamic performance versus CFD calls in PSO is shown on the left side of Fig. 3.59. The right figure in Fig. 3.59 depicts the same objective function change versus CFD calls in SQP optimization process. Totally four different initial points are tested to escape from local optimums based on the previous experience. After these four SQP optimization processes slightly different optimal values are determined. In SQP process all C_q variables are determined as *-0.1* suction velocity, however all angle values are the same as they are in the PSO process. In SQP process the aerodynamic performance is *16.26%* which is slightly smaller than it is in the PSO process. A total of *110* CFD calls are needed to reach the optimal values in PSO process. However in the SQP processes a total of *65* CFD calls are needed to get the same optimal values. The best SQP process and PSO process are compared in terms of computation time on the lower part of Fig. 3.59. Although SQP is more efficient than the PSO process in terms of CFD calls, the PSO approach is more time-efficient than the SQP approach.

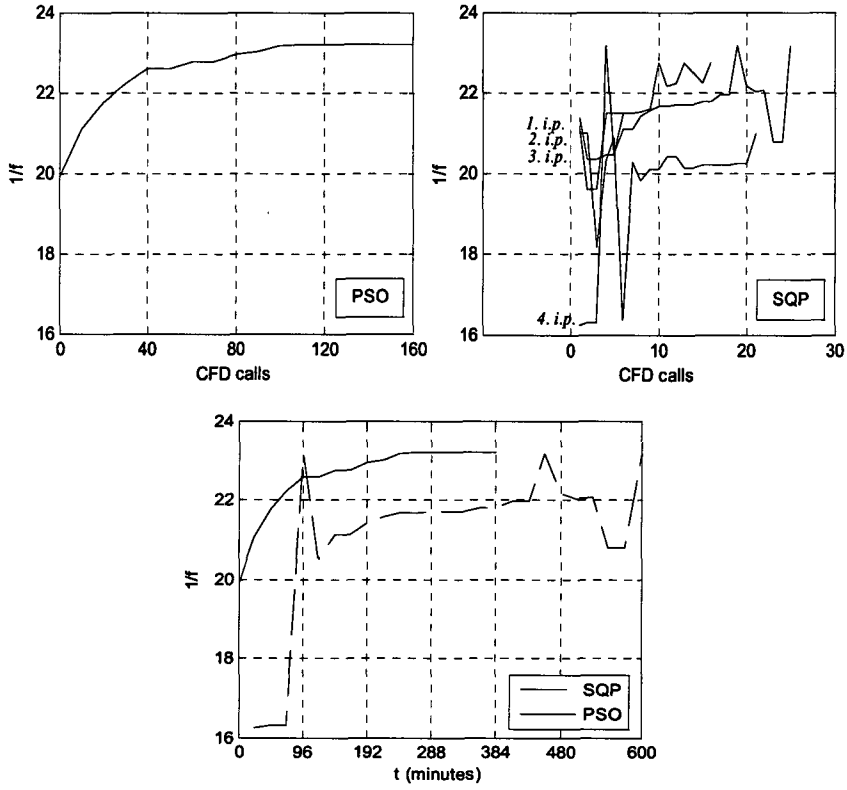

Figure 3.59 The change of aerodynamic performance in PSO and SQP

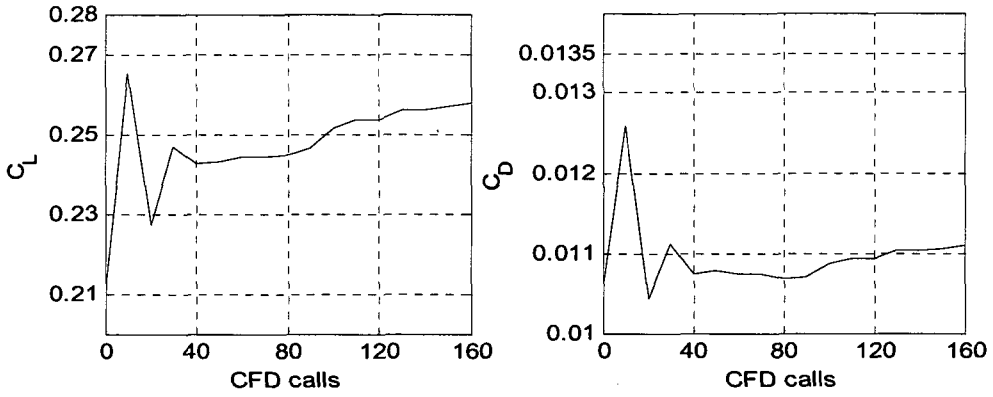

Figure 3.60 The change of aerodynamic coefficients during the generations

Results - Variable Location Optimizations

The effect of mass flow coefficient, angle, and the location of center of actuator for a different number of actuators are investigated. Similar to previous case studies three cases depending on the number of actuators are studied. At first, only one actuator is considered. Then, the number of actuators is increased to two, and finally three actuators are used to control the flow on the airfoil upper surface.

One-actuator Optimization

The design parameter vector and bounds are described as follows

$$\begin{gathered} x = [C_q \; \beta \; x_c]^T \\ -0.1 \le C_q \le 0.025 \\ 3° \le \beta \le 176° \\ 0.55c \le x_c \le 0.96c \end{gathered} \tag{3.57}$$

PSO optimization process takes only four generations. At the end of optimization C_q has taken the maximum suction velocity as -0.1, β became the minimum angle as $3°$, and the location arrived at $0.5561c$ which is close to shock wave. These are the same as the fixed location case except the location. However the remaining story is different. The changes of aerodynamic coefficients such as C_L, C_D and the aerodynamic performance $1/f$ versus CFD calls belong to the best particle are depicted in Fig. 3.61. After suction operation at optimal values and optimal location C_L increases by 8.02%, on the other hand C_D decreases by 1.33%. As a result $1/f$ increases by 9.46%. This result is reasonable better than fixed location optimization case for one-actuator. Another valuable point is that the drag is decreased due to location change.

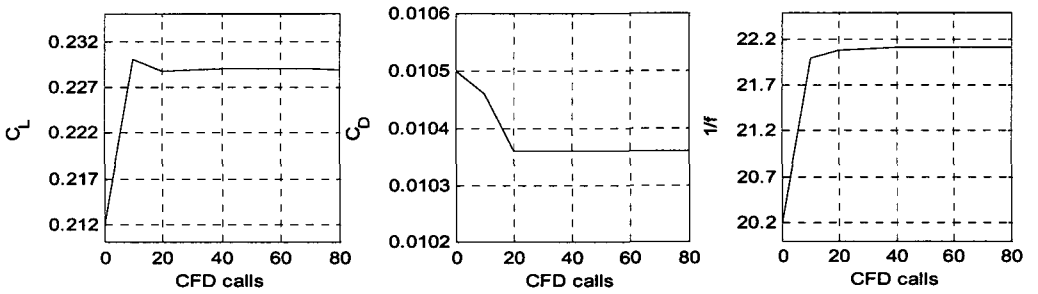

Figure 3.61 The change of aerodynamic coefficients and performance during the generations

Two-actuator Optimization

The design parameter vector and bounds are described as follows

$$\begin{aligned} x &= [C_q^1\ \beta^1\ x_c^1\ C_q^2\ \beta^2\ x_c^2]^T \\ -0.1 &\le C_q^i \le 0.025 \\ 3° &\le \beta^i \le 176° \quad |^{i=1,2} \\ 0.70c &\le x_c^1 \le 0.90c \\ 0.55c &\le x_c^2 \le 0.65c \end{aligned} \tag{3.58}$$

To avoid the geometrical interaction $0.05c$ distance is kept between the actuators. PSO optimization process takes six generations. At the end of optimization both C_q variables have taken the maximum suction velocity as -0.1, similarly both β angles became the minimum angle as $3°$, the first location x_c^1 arrived at $0.5724c$ which is close to shock wave and the second location x_c^2 arrived at $0.7000c$. These are the same as the fixed location case except the locations. The changes of aerodynamic coefficients such as C_L, C_D and performance $1/f$ versus CFD calls belong to the best particle are depicted in Fig. 3.62. After suction operations at optimal values and optimal locations C_L increases by 17.68%, on the other hand C_D increases by 1.71%. As a result $1/f$ increases by 15.69%. This result is much better than fixed location optimization case for two-actuator.

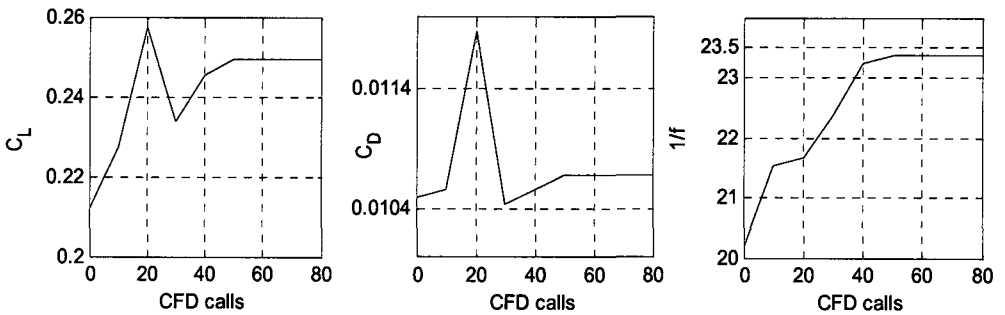

Figure 3.62 The change of aerodynamic coefficients and performance during the generations

Three-actuator Optimization

The design parameter vector and bounds are described as follows

$$\begin{aligned} \boldsymbol{x} = [C_q^{\,1}\ \beta^1\ x_c^{\,1}\ C_q^{\,2}\ \beta^2\ x_c^{\,2}\ C_q^{\,3}\ \beta^3\ x_c^{\,3}]^T \\ -0.1 \le C_q^{\,i} \le 0.025 \\ 3° \le \beta^i \le 176° \ \Big|^{i=1,2,3} \\ 0.85c \le x_c^{\,1} \le 0.96c \\ 0.70c \le x_c^{\,2} \le 0.80c \\ 0.55c \le x_c^{\,3} \le 0.65c \end{aligned} \tag{3.59}$$

Similar to previous case study, to avoid the geometrical interaction among the actuators *0.05c* distance is kept between the actuators. PSO optimization process takes eleven generations. At the end of optimization $C_q^{\,1,2,3}$ have taken the suction velocity as *-0.1*, *-0.077*, and *-0.1*, respectively. $\beta^{1,2,3}$ became the minimum angle as *3°*, the first location $x_c^{\,1}$ arrived at *0.5889c* which is close to shock wave, the second location $x_c^{\,2}$ arrived at *0.7700c*, and the third location $x_c^{\,3}$ arrived at *0.9600c*. The changes of aerodynamic coefficients such as C_L, C_D and the performance $1/f$ versus CFD calls belong to the best particle are depicted in Fig. 3.63. After suction operations at optimal values and optimal locations C_L increases by *37.48%*, on the other hand C_D increases by *14.29%*. As a result, $1/f$ increases by *20.7%*. This result is much better than fixed location optimization case for three-actuator.

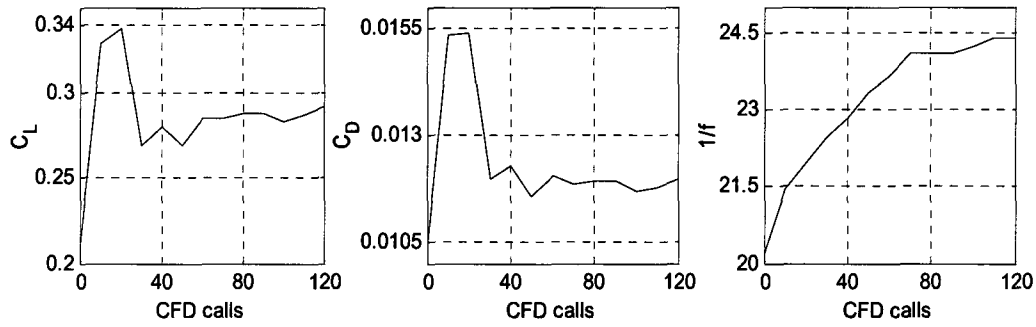

Figure 3.63 The change of aerodynamic coefficients and performance during the generations

Comparative Evaluations

Both studies showed that the number of actuators and their locations have important effects on an aerodynamic performance. In Fig. 3.64 the relationship between the number of actuators and performance is depicted. Usage of more actuators on the airfoil surface provides more increase in aerodynamic performance. However this increase looks like an exponential curve. Selection of location of the actuator is also another emphasized point. Instead of determining the locations based on guess usage of optimized locations provide more efficient results. As can be seen in Fig. 3.64 using only two actuators at proper locations can provide almost the same level aerodynamic performance increase as the usage of three actuators.

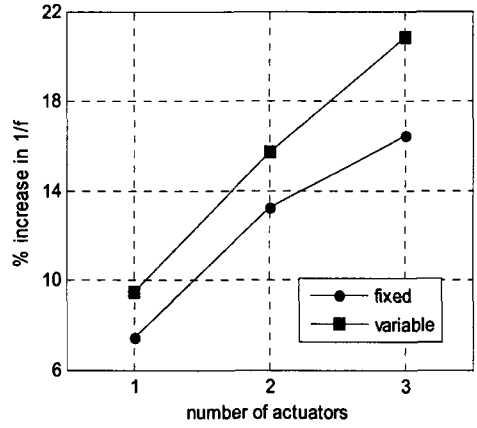

Figure 3.64 The effect of location and the number of actuators on an aerodynamic performance

On the other hand, using of actuator causes to change in the location of shock wave center. In all cases the location of shock wave center proceeded through downstream. On the left side of Fig. 3.65(a) the pressure counters are shown for fixed location case of one-actuator usage. Due to suction operation at *0.7075c* point a local suction pocket is generated. Because of this local suction pocket the shock wave is located to *0.5625c* which is much farther than the original place, *0.5100c*. On the right side of Fig. 3.65(a) the pressure counters are shown for optimized location case of one-actuator usage. Similar to fixed location case suction operation at *0.5561c* point a local suction pocket is generated. Because of this local suction pocket the shock wave is forwarded to *0.5350c*. The effects of both suction operations on pressure coefficient can be seen in Fig. 3.65(b). Selected fixed location causes a sharp increase and then sharp decrease in C_p distribution. However suction at the optimal location causes relatively small and smooth decrease and then determinately increase in C_p distribution. In both operations forwarded shock wave locations are clearly observed comparing with no-suction case.

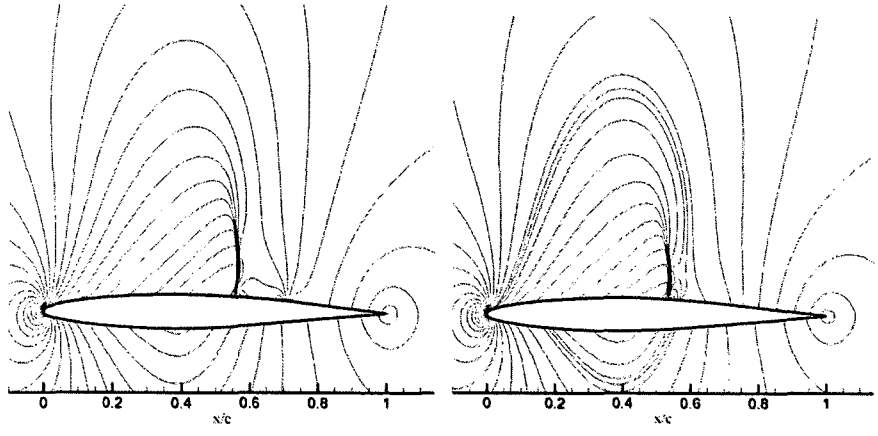

Figure 3.65(a) The effect of actuator on pressure counters for one-actuator case

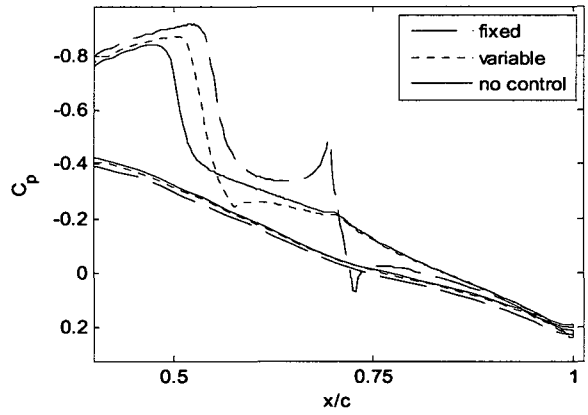

Figure 3.65(b) The effect of actuator on pressure coefficient for one-actuator case

On the left side of Fig. 3.66(a) the pressure counters are shown for fixed location case of two-actuator usage. Due to suction operations at *0.7075c* and *0.61775c* points two separated local suction pockets are generated. Because of these local suction pockets the shock wave is located to *0.5400c*. On the right side of Fig. 3.66(a) the pressure counters are shown for optimized locations case of two-actuator usage. Suction operations at *0.5724c* and *0.7000c* points a unified local suction pocket is generated. Because of this local strong suction pocket the shock wave is forwarded to *0.5500c*. The effects of both suction operations on pressure coefficient can be seen in Fig. 3.66(b). Selected fixed locations cause to sharp increases and then sharp decreases in C_p distributions. However suction operations at the optimal locations cause to relatively smooth decrease and then determinately increase in C_p distribution. In both operations forwarded shock wave locations are clearly observed comparing with no-suction case.

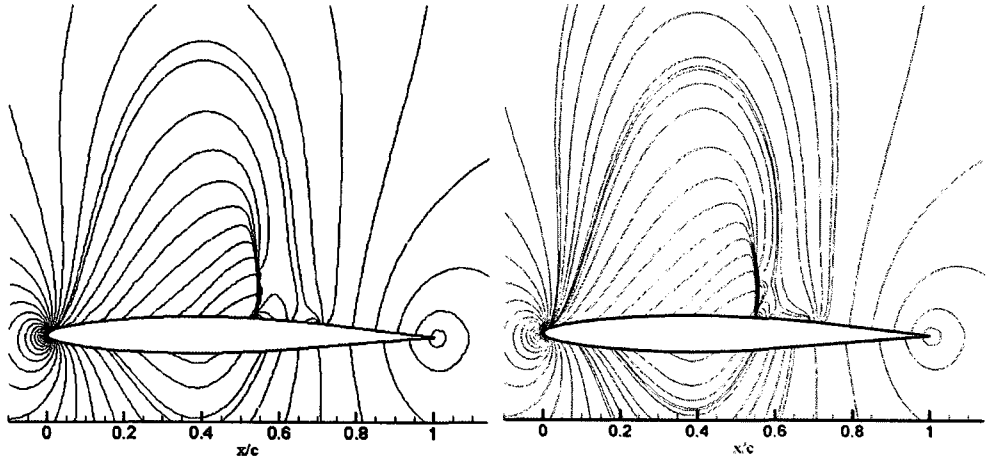

Figure 3.66(a) The effect of location on pressure counters for two-actuator case

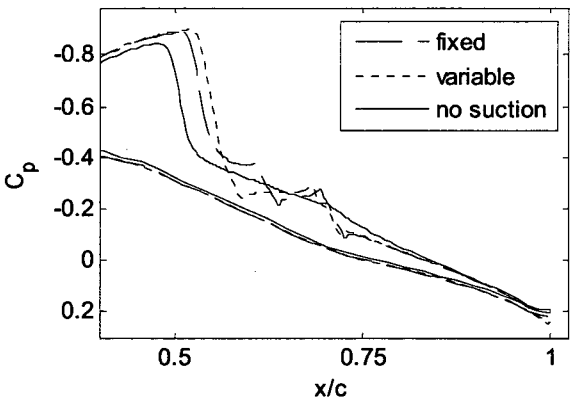

Figure 3.66(b) The effect of location on pressure coefficient for two-actuator case

On the left side of Fig. 3.67(a) the pressure counters are shown for fixed location case of three-actuator usage. Due to suction operations at *0.7075c*, *0.61775c*, and *0.5125c* points three separated local suction pockets are generated. Because of these local suction pockets the shock wave is located to *0.5525c*. On the right side of Fig. 3.67(a) the pressure counters are shown for optimized locations case of three-actuator usage. Suction operations at *0.9600c*, *0.7700c*, and *0.5889c* points one unified local suction pocket and additionally one separated pocket are generated. Because of these local strong suction pockets the shock wave is forwarded to *0.5650c* which is the farthest point among the other cases. The effects of both suction operations on pressure coefficient distributions can be seen in Fig. 3.67(b). Similar to previous case selected fixed locations cause to sharp increases and then sharp decreases in C_p distribution. However suction operations at the optimal locations cause to relatively smooth decrease and then determinately increase in C_p distribution. In both operations forwarded shock wave locations are clearly observed comparing with no-suction case.

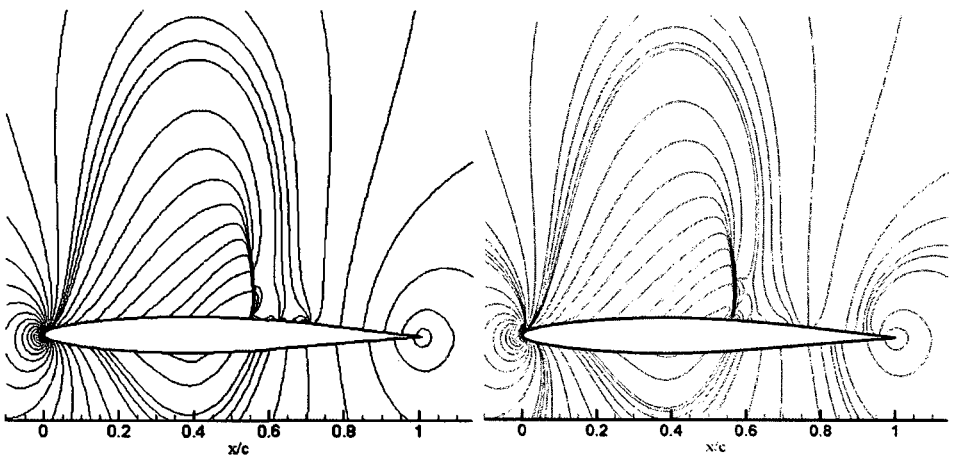

Figure 3.67(a) The effect of location on pressure counters for three-actuator case

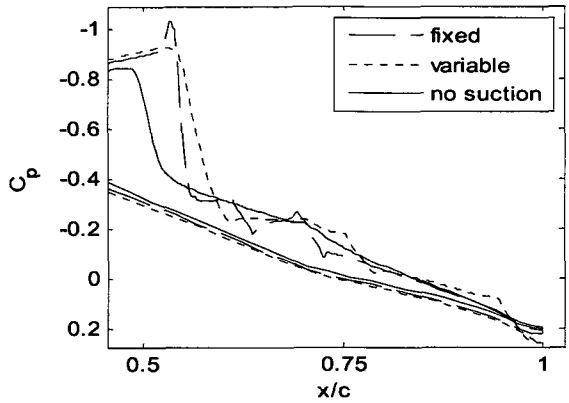

Figure 3.67(b) The effect of location on pressure coefficient for three-actuator case

In all cases forwarding the shock wave toward the trailing edge extends the supersonic region and shortens the subsonic area. Additionally optimization process enforces the suction locations to be unified for better aerodynamic performance.

Conclusions

Some validation cases have been done against the experimental data regarding pressure distribution, lift and drag coefficients. These validation studies guaranteed the prediction capabilities of the flow analyzer software. Additionally the grid sensitivity study was performed to ensure the numerical solution accuracy of the governing equations.

In the first part of the case studies, numerical studies has been carried out to investigate the benefits of AFC to improve the aerodynamic performance of 2D aerofoil, NACA64A010, at transonic speed by using heuristic based optimization technique PSO and the gradient based optimization technique SQP. The suction/blowing angles and mass flow coefficients are taken as design parameters in one-actuator, two-actuator, and three-actuator AFC cases. Navier Stokes solver is coupled with both optimizers to obtain a flow solution for a given parameters and then trying to improve the aerodynamics of the airfoil via the optimizers. Unfortunately, the gradient based optimization method depends heavily on the initial points, so that several optimization runs with different initial values needed to escape from the local minimum points. Therefore, it is generally observed that PSO approach is much more efficient than SQP in terms of time and CFD calls. Using more actuators provide better aerodynamic performance. However, this trend is exponential. Suction operations result in increases in both aerodynamic coefficients in all cases. But increase in C_L is more than increase in C_D resulting better aerodynamic performance. The best result is provided by three-actuator case with 16.46% increase in aerodynamic performance. Additionally the shock

wave location on the upper surface is moved toward the trailing edge resulting in extension of supersonic region in all cases.

In the second part of the case studies, numerical studies has been carried out to investigate the benefits of AFC to improve the aerodynamic performance of the same airfoil under the same flow conditions by using PSO. The suction/blowing angles, mass flow coefficients, and the center locations of actuators are taken as design parameters in one-actuator, two-actuator, and three-actuator AFC cases. Navier Stokes solver is coupled with PSO optimizer to obtain a flow solution for a given parameters and then trying to improve the aerodynamics of the airfoil via the optimizer. Similar to previous cases using more actuators provide better aerodynamic performance. Suction operations result in increases in C_L and decrease or increase in C_D. Especially optimization of one-actuator case gives interesting result because C_D is decreased and C_L is increased in this case. The best result is provided by three-actuator case with 20.7% increase in aerodynamic performance. All cases give better results than fixed location optimization processes. Additionally the shock wave location on the upper surface is moved toward the trailing edge resulting in extension of supersonic region in all cases. The other important point is that the optimization processes enforce the locations to be unified to provide general, not local suction operations.

As a result, based on this study it is concluded that AFC can provide high aerodynamic performance enhancement by optimizing its parameters. However, AFC may not be sufficient to get more efficient designs. Therefore, using of both AFC and passive flow control techniques may be more promising at preliminary design phase.

SUMMARY AND CONCLUSIONS

Optimization is to search the best solution under certain conditions. It is a powerful tool. Engineers who research in different scientific areas if "not must" but absolutely should study one of the optimization methods. Because the real world is not perfect, we always need to find better solutions.

There are different methods of optimization. This variety originates from the real world. Different problems need different solution methodologies. It is a natural phenomenon. As such, however, it is difficult to find the universal solution method or even general trends. However we can point out some guidelines.

Mathematical modeling of a natural phenomenon may be defined by finite increments. Differential equations are constructed with relative increments among the factors related to yield. Therefore, the gradients of these increments are essential to search the yield space. However, the landscape of yield is not simple. If the particular one being considered is simple, let's say convex, we can efficiently find the best solution by using these gradient information. If the landscape is complex, as it is in real engineering problems, the gradient information is not enough to find the best solution. It gives a better solution than the initial solution at hand, but it will probably not be the best solution. To compensate this deficiency, we can use multiple starting points. However, the required number of multiple points may not be cheap if the number of factors and also the number of factor levels are large. The systematical approach to find proper multiple points is to use one of DoE (Design of Experiments) methods. Factorial designs, such as 2^k or 3^k, are basic examples for DoE. Here k is the number of factors, 2 and 3 are the number of factor levels. We can easily observe from these simple expressions, that the increase in the number of factors or the number of levels, results in huge multiple points. Just for an example, if k is 10 for 3-level factorization, the required multiple points are 59049; if k is 11, the required multiple points are 177147. We recall that these multiple points are just the beginning for multiple searches. Another issue is the differentiability. Engineering design problems are usually nonlinear and they sometimes exhibit discontinuous derivatives for objective as well as the constraint functions. Due to these difficulties, non-gradient based algorithms have gathered more interest in recent decades.

Genetic algorithms (GA) and particle swarm optimization (PSO) algorithms are popular, non-gradient based algorithms. Both are population based search algorithms and have multiple points for initiation. The difference of population-based from gradient based methodology is the nature of search methodology. The randomness is essential for this kind

of method. They are also referred to as stochastic optimization methods. These algorithms are simple, robust, and have high fidelity. However, they suffer from similar defects such as premature convergence, less accuracy, or large computational time. The premature convergence is sometimes inevitable due to the lack of diversity. As the generations of particles or individuals in the population evolve, they may lose their diversity and become similar to each other. To overcome this issue, we studied the diversity concept in GA and PSO algorithms.

Diversity is essential for a healthy search, and mutations are the basic operators to provide variety within the population. After having a close scrutiny of the diversity concept based on qualification and quantification studies, we improved new mutation strategies and operators to provide beneficial diversity within the population. We referred to this new approach as multi-frequency vibrational GA or PSO. They were applied to different aeronautical engineering problems in order to verify the efficiency of these new approaches. These implementations were: applications in selected benchmark test functions, inverse design of two-dimensional (2D) airfoil in subsonic flow conditions, optimization of 2D airfoil in transonic flow, path planning problems of autonomous unmanned aerial vehicle (UAV) over 3D terrain environments, 3D radar cross section minimization problem for a 3D air vehicle, and active flow control over a 2D airfoil.

From these test cases we observed that the newly developed algorithms outperform the current algorithms:

- The principal role of this multi-frequency approach was to answer the question of which individuals/particles should be mutated and when they should be mutated.
- The new mutation operators provided local and global diversities during the reproduction phases of the generations.
- The new approach, when combined with an artificial intelligent method, such as, neural network or fuzzy logic process, it also introduced a random diversity and a controlled diversity.
- The first improved mutation operator was applied to the whole population/swarm and this application provided global but random diversity in the population/swarm. The global diversity afforded a chance for the population to escape from all local optima.
- The second improved mutation operator was applied to specifically the elite individual/particle in the population. This operation provided local diversity in the population.
- The control provided by neural network or fuzzy logic offered controlled diversity leading to a fast convergence. With the neural network diversity tool, the elite

individual/particle was stochastically (random based) mutated. However, with the fuzzy logic diversity tool, the elite individual was deterministically mutated.

- Due to still being population-based techniques, these methods were as robust as the plain GA or PSO algorithms.
- Based on the results obtained, it was concluded that the variants of the present multi-frequency vibrational GA and PSO were efficient and fast algorithms since they successfully avoided all local optima within relatively short optimization cycles.

For future work, it is planned to apply the newly developed algorithms to a multi-element airfoil design and the problem of sonic boom mitigation.

REFERENCES

[1] Merriam-Webster, online Dictionary, http://www.merriam-webster.com, 2009.

[2] Weise T., *Global Optimization Algorithms – Theory and Application*, 2nd Edition, e-book, http://www.it-weise.de., 2008, pp. 37-59.

[3] Holland J. H., *Adaptation in Natural and Artificial Systems*, the University of Michigan Press, 1975.

[4] Karaboga, D., and Pham, D. T., *Intelligent Optimization Techniques; Genetic Algorithms, Tabu Search, Simulated Annealing, and Neural Networks*, Springer-Verlag London Limited, 2000, p. 3.

[5] Whitley D., "A Different Genetic Algorithm," Genitor, *In Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, 1989.

[6] Goldberg D., Korb B., and Deb K., "Messy Genetic Algorithms; Motivation, Analysis, and First Results," *Complex Systems*, Vol. 3, 1989, pp. 493-530.

[7] Bäck T., "Selective Pressure in Evolutionary Algorithms; a Characterization of Selection Mechanics", edited by Fogel, D., *Proceeding of the First IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway, 1994, pp. 57-62.

[8] Haupt R.L. and Haupt S.E., *Practical Genetic Algorithms*, 2nd Edition, John Wiley & Sons, Inc., Publication, 2004.

[9] Kampolis, I. C., and Giannakoglou, K. C., "A Multilevel Approach to Single- and Multiobjective Aerodynamic Optimization, *Comput. Methods Appl. Mech. Engrg.*, (2008), doi: 10.1016/ j.cma.2008. 01. 015.

[10] Lim, D., Ong, Y-S., Jin, Y., Sendhoff, B., Lee, B-S., "Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing," *Future Generation Computer Systems*, Vol. 23, No. 4, 2007, pp. 658-670.

[11] Giannakoglou, K. C., Papadimitriou, D. I., Kampolis, I. C., "Aerodynamic Shape Design Using Evolutionary Algorithms and New Gradient-Assisted Metamodels," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195/44-47, 2006, pp. 6312-6329.

[12] Vicini, A., and Quagliarella, D., "Airfoil and Wing Design Through Hybrid Optimization Strategies," AIAA 98-2729, New Mexico, 1998.

[13] Tse, D. C. M., and Chan, Y. Y. L., "Application of Micro Genetic Algorithms and Neural Networks for Airfoil Design Optimization," *RTO-AVT Symposium*, RTO MP-35, Canada, 1999.

[14] Muyl, F., Dumas, L., and Herbert, V., "Hybrid Method for Aerodynamic Shape Optimization in Automotive Industry," *Computers & Fluids*, 33, 2004, pp. 849–858.

[15] Hacioglu, A., "Augmented Genetic Algorithm with Neural Network and Implementation to Airfoil Design," AIAA 2004-4633, 2004.

[16] Jouhaud, J. C., Sagaut, P., Montagnac, M., Laurenceau, J., "A Surrogate-Model Based Multidisciplinary Shape Optimization Method with Application to a 2D Subsonic Airfoil," *Computers & Fluids*, Vol. 36/3, 2007, pp. 520-529.

[17] Khurana M. S., Winarto H., and Sinha A. K., "Application of Swarm Approach and Artificial Neural Networks for Airfoil Shape Optimization", AIAA 2008-5954, *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia, 2008.

[18] Epstein B., "Robust Hybrid Approach to Multiobjective Constrained Optimization in Aerodynamics", *AIAA Journal*, Vol.42, No.8, pp. 1572-1581.

[19] Rechenberg I., *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biotogischen Evolution*, Stuttgart: Frommann-Holzboog, 1973.

[20] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin: Springer, 1996.

[21] De Falco, I., Cioppa, A. D., Balio, R. D., Tarantino, E., "Breeder Genetic Algorithms for Airfoil Design Optimization," *IEEE Int. Conf. on Evolutionary Computing*, Japan, 1996.

[22] De Falco, I., Cioppa, A. D., Lazzetta, A, Tarantino, E.,"Mijn Mutation Operator for Airfoil Design Optimization, *Soft Computing in Engineering Design and Manufacturing*, 1998, pp. 211-220.

[23] Vatandas, E., Hacioglu, A., and Ozkol, I., "Vibrational Genetic Algorithm (VGA) And Dynamic Mesh in the Optimization of 3D Wing Geometries," *Inverse Problems in Science and Engineering*, Vol.15/6, 2007, pp.643-657.

[24] Eberhart, R. C., and Kennedy, J., A New Optimizer Using Particle Swarm Theory, *In Proc. 6th Int. Symp. Micromachine Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[25] Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J.-C., and Harley, R. G., "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 2, 2008, pp. 171-195.

[26] Shi, Y., and Eberhart, R., "Parameter Selection in Particle Swarm Optimization," *Proc. 7th Annual Conf. on Evolutionary Programming*, 1998, Pp. 591-601.

[27] Higashi, H., and Iba, H., "Particle Swarm Optimization with Gaussian Mutation," *In Proc. Of The IEEE Swarm Intelligence Symposium*, 2003, pp. 72 – 79.

[28] Stacey, A., Jancic, M., and Grundy, I., "Particle Swarm Optimization with Mutation," *In Proc. of the IEEE Congress on Evolutionary Computation*, 2003, pp. 1425 – 1430.

[29] Pant, M., Thangaraj, R., Singh, V. P., and Abraham, A., "Particle Swarm Optimization Using Sobol Mutation," *1st Inter. Conf. on Emerging Trends in Engineering and Technology*, IEEE 2008, pp. 367-372.

[30] Andrews, P. S., "An Investigation into Mutation Operators for Particle Swarm Optimization," *IEEE Congress on Evolutionary Computation*, July 2006, pp. 1044-1051.

[31] Zavala, A. E., Aguirre, A. H., Diharce, E. R. V., and Rionda, S. B., "Constrained Optimization with an Improved Particle Swarm Optimization Algorithm," *Inter. Journal of Intelligent Computing and Cybernetics*, Vol. 1, No. 3, 2008, pp. 425-453.

[32] Jia, D., Li, L., Zhang, Y., and Chen, X., "Particle Swarm Optimization Combined with Chaotic and Gaussian Mutation," *Proc. of the 6th World Congress on Intelligent Control and Automation*, June 2006, pp. 3281-3285.

[33] Wu, X., Cheng, B., Cao, J., and Cao, B., "Particle Swarm Optimization with Normal Cloud Mutation," *Proc. on the 7th World Congress on Intelligent Control and Automation*, June 2008, pp.2828-2832.

[34] Chen, J., Ren, Z., and Fan, X., "Particle Swarm Optimization with Adaptive Mutation and its Application Research in Tuning of PID Parameters," *1st Inter. Symp. on Systems and Control in Aerospace and Astronautics*, 2006, pp.990-994.

[35] Yang, M., Huang, H., and Xiao, G., "A Novel Dynamic Particle Swarm Optimization Algorithm Based on Chaotic Mutation," *2nd Inter. Workshop on Knowledge Discovery and Data Mining*, IEEE 2009, pp.656-659.

[36] Liu, J., Fan, X., and Qu, Z., "An Improved Particle Swarm Optimization with Mutation Based on Similarity,"*3rd Inter. Conf. on Natural Computation*, 2007, pp. 824-828.

[37] Biao, C., Zhishu, L., Die, F., Jian, H., Peng, O., and Qing, L., "Mutated Fast Convergent Particle Swarm Optimization and Convergence Analysis," *1st Inter. Conf. on Intelligent Networks and Intelligent Systems*, IEEE 2008, pp. 5-8.

[38] Merriam-Webster, online Dictionary, http://www.merriam-webster.com, 2009.

[39] Reeves C.R., 1993, "Using Genetic Algorithms with Small Populations," *Proc. of the 5th Int. Conf. on Genetic Algorithms*, Edited by Forrest S., pp. 92-99.

[40] Misiti M., Misiti Y., Oppenheim G., and Poggi J-M., *Wavelets and Their Applications*, ISTE Ltd., 2007, pp. 8-12.

[41] Abry, P., *Ondelettes et Turbulence. Multirésolutions, Algorithmes de Décomposition, Invariance D'échelles*, Diderot Editeur, Paris, 1997, p. 268.

[42] Rodrigues D.L., Stanford C.A., "Response Surface Based Optimization with a Cartesian CFD Method," AIAA paper 2003-0465, 2003.

[43] Simpson T.W., Mauery T.M., Korte J.J., Mistree F., 2001, "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization," *AIAA Journal*, Vol. 39, No. 12, pp. 2233-2241.

[44] Chung H.S., Alonso C.C., 2000, "Comparison of Approximation Models with Merit Functions for Design Optimization," *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, CA. AIAA 2000-4754, 2000.

[45] Park J., Sandberg I.W., "Universal Approximation Using Radial Basis Function Networks," *Neural Computation,* Vol. 3, No. 2, 1991, pp. 246-257.

[46] Elanayar S.V.T., Shin Y.C., "Radial Basis Function Neural Network for Approximation and Estimation of Nonlinear Stochastic Dynamic Systems," *IEEE Transactions on Neural Networks*, Vol. 5, No. 4, 1994, pp. 594-603.

[47] Jin, Y., Olhofer, M., and Sendhoff, B., "A Framework for Evolutionary Optimization with Approximate Fitness Function," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 5, 2002, pp 481-494.

[48] Ong, Y. S., Nair, P. B. and Keane, A. J., "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling," *AIAA Journal*, Vol. 41, No. 4, 2003, pp. 687-696.

[49] Ozcan, E., and Mohan, C. K., "Particle Swarm Optimization: Surfing the Waves," *In Proc. 1999 Congr. Evolutionary Computation*, Washington, DC, 1999, pp. 1939–1944.

[50] Clerc, M., and Kennedy, J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Trans. Evol. Comput.*, Vol. 6, No. 1, 2002, pp. 58–73.

[51] Shi, Y., and Eberhart, R., "A Modified Particle Swarm Optimizer," *Proc. of the World Congress on Computational Intelligence*, IEEE 1998, pp. 69–73.

[52] Pant, M., Radha, T., and Singh, V. P., "A New Diversity Based Particle Swarm Optimization Using Gaussian Mutation," *Int. J. of Mathematical Modeling, Simulation and Applications*, 1(1), 2008, pp. 47-60.

[53] Sobieczky H., "Parametric Airfoils and Wings," *Notes on Numerical Fluid Mechanics*, 1998, pp.71-88.

[54] Wu H-Y., Yang S., Liu F., and Tsai H-M, "Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization," AIAA 2003–4095, *16th AIAA Computational Fluid Dynamics Conference*, 2003.

[55] Farin G., *Curves and Surfaces for Computer Aided Geometric Design; A practical Guide*, Academic Press, Inc., 1993, p. XVI.

[56] Rogers F. D., *An Introduction to NURBS with Historical Perspective*, USA, 2001, p. 10.

[57] Klein M., Sobieczky H., "Sensitivity of Aerodynamic Optimization to Parameterized Target Functions," *Proc. Int. Symp. On Inverse Problems an Engineering Mechanics*, Japan, 2001.

[58] von Doenhoff E. A., Abbott H. I., *Theory of Wing Sections*, Dover Publications, Inc., 1959, pp. 111-115.

[59] Nemec M., *Optimal Shape Design Of Aerodynamic Configurations: A Newton-Krylov Approach,* A thesis for the degree of Doctor of Philosophy, Toronto, 2003.

[60] Chang I-C., Torres F. C., and Tung C., *Geometric Analysis of Wing Sections*, NASA Technical Memorandum 110346, 1995.

[61] Hajek J., "Parameterization of Airfoils and Its Application in Aerodynamic Optimization," *WDS'07 Proceedings of Contributed Papers*, Part I, 2007, pp. 233–240.

[62] Wu H-Y., Yang S., Liu F., and Tsai H-M, "Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization," AIAA 2003–4095,*16th AIAA Computational Fluid Dynamics Conference*, 2003.

[63] Mousavi A., Castonguay P., Nadarajah S. K, *Survey Of Shape Parameterization Techniques and its Effect on 3-Dimensional Aerodynamic Shape Optimization*, AIAA document.

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

[65] Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill Book Company, New York, 1984.

[66] Holland J. H., *Adaptation in Natural and Artificial Systems*, the University of Michigan Press, 1975, pp. 21-22.

[67] Shahrokhi A., Jahangirian A., "Airfoil Shape Parameterization for Optimum Navier–Stokes Design with Genetic Algorithm," *Aerospace Science and Technology* 11 [2007] 443–450.

[68] Song W., and Keane A. J., "A Study of Shape Parameterization Methods for Airfoil Optimization," *10th AIAA Multidisciplinary Analysis and Optimization Conference*, Albany, New York, AIAA 2004-4482, 2004.

[69] Haupt, R. L., and Haupt, S. E., *Practical Genetic Algorithms*, 2nd Edition, John Wiley & Sons, Inc., Publication, 2004, pp. 39-40.

[70] Eshelman, L.J., and Schaffer, J. D., *Real Coded Genetic Algorithms and Interval Schemata, Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, 1993, pp.187-202.

[71] Anderson, J. D., *Fundamentals of Aerodynamics*, Mc-Graw Hill, Inc., 1984, pp. 217-222.

[72] Dambrosio, L., Pascazio, G., and Semeraro, S., "Aerodynamic Inverse Design Using Fuzzy Logic, *Inverse Problems in Science and Engineering*, Vol. 16/2, March 2008, pp. 249–268.

[73] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Computational Physics*, Vol. 43, 1981, pp.347-372.

[74] Haykin, S., *Neural Network; A Comprehensive Foundation*, Prentice Hall, 1999.

[75] Hwang, J. Y., Kim, J. S., Lim, S. S., and Park, K. H., "A Fast Path Planning by Path Graph Optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, Vol. 33, 2003, pp. 121-129.

[76] Chen, G., Shen, D., Cruz, J., Kwan, C., Riddle, S., Cox, S., and Matthews, C., "A Novel Cooperative Path Planning for Multiple Aerial Platforms, 2005, AIAA-2005-6948.

[77] Fries, T. P., "Fuzzy Genetic Motion Planning under Diverse Terrain Conditions for Autonomous Robots," *Proc. [449] Circuits, Signals, and Systems*, 2004, pp. 504-508.

[78] Lee, Y., Kim, Y., and Kohou, L. J., "An Intelligent Collision Avoidance System for Autonomous Underwater Vehicles Using Fuzzy Relational Products," *Information Sciences-Informatics and Computer Science: An International Journal*, Vol. 158, Issue 1, 2004, pp. 209-232.

[79] Althoefer, K., *Neuro-Fuzzy Motion Planning for Robotic Manipulators*, Ph. D. Thesis, King's College, London, 1997.

[80] Fan, X., Luo, X., Yi, S., Yang, S., and Zhang, H., "Optimal Path Planning for Mobile Robots Based on Intensified Ant Colony Optimization Algorithm," *IEEE Proc. Int. Conf. Robotics, Intelligent Systems and Signal Processing*, Vol. 1, 2003, pp. 131-136.

[81] Mou, C., Qing-xian, W., and Chang-Sheng, J., "A Modified Ant Optimization Algorithm for Path Planning of UCAV," *Appl. Soft Computing*, Vol. 8, Issue 4, 2007, pp. 1712-1718.

[82] Tan, G., He, H., and Sloman, A., "Global Optimal Path Planning for Mobile Robot Based on Improved Dijkstra's Algorithm and Ant System Algorithm," *Central South University of Technology*, China, Vol. 13, 2006, pp. 80-86.

[83] Park, M. G., and Lee, M. C., "Experimental Evaluation of Robot Path Planning by Artificial Potential Field Approach with Simulated Annealing," *Proc. of the 41st SICE Annual Conf.*, Vol. 4, 2002, pp. 2190-2195.

[84] Sharifi, F. J., and Vinke, D., "Integration of the Artificial Potential Field Approach with Simulated Annealing for Robot Path Planning," *Proc. of IEEE Int. Sym. Intelligent Control*, 1993, pp. 536-541.

[85] Saska, M., Macas, M., Preucil, L., and Lhotska, L., "Robot Path Planning Using Particle Swarm Optimization of Ferguson Splines," *Conf. Emerging Technologies and Factory Automation*, IEEE, 2006, pp. 833-839.

[86] Piao, S., and Hong, B., "Path Planning of Robot Using Genetic Annealing Algorithm," *Proc. of the 4th World Congress on Intelligent Control and Automation*, Vol.1, 2002, pp. 493-495.

[87] Pehlivanoglu, Y. V., Baysal, O., and Hacioglu, A., "Path Planning for Autonomous UAV via VGA," *Aircraft Engineering and Aerospace Technology: An International Journal*, Vol. 79, No. 4, 2007, pp. 352–359.

[88] Yan, X., Wu, Q., Yan, J., and Kang, L., "A Fast Evolutionary Algorithm for Robot Path Planning," *Int. Conf. Control and Automation*, IEEE 2007, pp. 84-87.

[89] Xuanzi, H., and Cunxi, X., "Niche Genetic Algorithm for Robot Path Planning," *3rd Int. Conf. Natural Computation*, IEEE, Vol. 2, 2007, pp. 774-778.

[90] Li, Q., Zhang, W., Yin, Y., Wang, Z., and Liu, G., "An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots," *6th Int. Conf. on Intelligent Systems Design and Applications*, Vol. 2, 2006, pp. 637-642.

[91] Jia, D., and Vagners, J., "Parallel Evolutionary Algorithms for UAV Path Planning," AIAA 2004-6230, *AIAA 1st Intelligent Systems Technical Conf.*, Chicago, Illinois, 2004.

[92] Nikolos, I. K., Valavanis, K. P., Sourveloudis, N. C., and Kostaras, A. N., "Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation," *IEEE Trans. Syst., Man, Cybern. B, Cybern*, Vol. 33, No. 6, 2003, pp. 898-912.

[93] Rathbun, D., Kragelund, S., Pongpunwattana, A., and Capozzi, B., "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments," *21st Conf. on Digital Avionics Systems, Proc.*, Vol. 2, 2002, pp. 8D2-1-8D2-12.

[94] Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K., "Adaptive Evolutionary Planner/Navigator for Mobile Robots," *IEEE Transactions on Evolutionary computation*, Vol. 1, No. 1, 1997, pp. 18-28.

[95] Lee, D. T. and Schachter, B. J., "Two Algorithms for Constructing a Delaunay Triangulation," *International Journal of Computer and Information Sciences*, Vol. 9, No. 3, June 1980, pp. 219-242.

[96] Krozel J., and Andrisani D., "Navigation Path Planning for Autonomous Aircraft: Voronoi Diagram Approach," *Journal of Guidance: Engineering Notes*, Nov-Dec. 1990, pp. 1152-1154.

[97] Jung, D., and Tsiotras, P., "Online Path Generation for Small Unmanned Aerial Vehicles Using B-Spline Path Templates," AIAA 2008-7135.

[98] Bezdec, J.C., Pattern *Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[99] Brown A., "Fundamentals of Low Radar Cross-Sectional Aircraft Design," *Journal of Aircraft*, Vol. 30, No. 3, May-June 1993, pp. 289-290

[100] Raymer D., *Aircraft Design: A Conceptual Approach*, 3rd Ed., AIAA, Washington, 1999, p. 201-203

[101] Johnson, J.M., and Rahmat-Samii, Y., "Genetic Algorithms in Engineering Electromagnetics", *IEEE Antennas and Propagation Magazine*, Vol. 39, No. 4, August 1997, pp. 7-25.

[102] De-ging Y. and Feng-jun G., "Ship Appearance Optimal Design on RCS Reduction Using Response Surface Method and Genetic Algorithms," *J. Shanghai Jiatong Univ. (Sci)*, 2008, 13(3):336-342.

[103] Jing C., Shuo T., "Integrated Optimization Design of Hypersonic Cruise Vehicle", AIAA-2008-142, *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, Jan. 7-10, 2008.

[104] Hong Liu, "Multiobjective Evolutionary Computation for Noncircular Missile Shape Optimization", AIAA-2004-453, *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, Jan. 5-8, 2004.

[105] Bondeson A., Yang Y., and Weinerfelt P., Optimization Of Radar Cross Section by a Gradient Method, *IEEE Transactions On Magnetics*, Vol. 40, No. 2, March 2004, pp. 1260-1263.

[106] Nair, D., and Webb, J. P., "Optimization of microwave devices using 3-D finite elements and the design sensitivity of the frequency response," *IEEE Trans. Magn.*, Vol. 39, 2003, pp. 1325–1328.

[107] Ozturk A. K., *Implementation Of Physical Theory Of Diffraction For Radar Cross Section Calculations*, Master Thesis, Bilkent University, Turkey, July 2002.

[108] Mahahza R.B., Elsherbeni A. Z., *Matlab Simulations for Radar Systems Design*, Chapman & Hall/CRC, Florida, 2004, pp. 503-535.

[109] Chatzigeorgiadis F., *Development of Code for Physical Optics Radar Cross Section Prediction and Analysis Application*, Master's Thesis, Naval Post Graduate School, California, USA, 2004.

[110] Tan, C. M., Ray, T., and Tsai, H. M., "A Comparative Study of Evolutionary Algorithm and Swarm Algorithm for Airfoil Shape Optimization Problems," AIAA 2003-102, *41st Aerospace Sciences Meeting and Exhibit*, Reno 2003.

[111] Ng, K.Y., Tan, C.M., Ray, T., and Tsai, H.M., "Single and Multi-objective Wing Platform and Airfoil Shape Optimization Using a Swarm Algorithm," AIAA 2003-45, *41st Aerospace Sciences Meeting and Exhibit*, Reno, 2003.

[112] Ray, T., and Tsai, H. M., "Swarm Algorithm for Single and Multiobjective Airfoil Design Optimization, *AIAA Journal*, Vol. 42, No. 2, 2004, pp. 366-373.

[113] Venter, G. and Sobieski, J., "Multidisciplinary Optimization of a Transport Aircraft Wing Using Particle Swarm Optimization, AIAA 2002–5644, *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, 2002.

[114] Venter, G. and Sobieski, J., "Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations," *Journal of Aerospace Computing, Information, and Communication*, Vol. 3, March 2006, pp. 123-137.

[115] Khurana1, M. S., Winarto H., and Sinha, A.K., "Application of Swarm Approach and Artificial Neural Networks for Airfoil Shape Optimization," AIAA 2008-5954, *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, 2008.

[116] Gad-el-Hak, M., "Flow Control: The Future", *Journal of Aircraft*, Vol. 38, No. 3, May–June 2001, pp. 402-418.

[117] Kibens, V. and Bower, W. W., "An Overview of Active Flow Control Applications at The Boeing Company", *2nd AIAA Flow Control Conference*, 28 June - 1 July 2004, Portland, Oregon, AIAA 2004-2624.

[118] Chng, T.L., Zhang, J. and Tsai, H.M., "A Novel Method of Flow injection and Suction for Lift Enhancement", *46rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 7-10 Jan 2008, AIAA 2008-335.

[119] Duvigneau, R., and Visonneau, M., "Simulation and Optimization of Aerodynamic Stall Control using a Synthetic Jet", *2nd AIAA Flow Control Conference*, Portland, Oregon, AIAA 2004-2315, 2004.

[120] Huang, L., Huang G., and LeBeau, R., "Optimization of Airfoil Flow Control Using a Genetic Algorithm with Diversity Control," *Journal of Aircraft*, Vol. 44, No. 4, 2007, pp. 1338, 1349

[121] Vadillo, J., Agarwal, R.K, and Hassan, A.A., "Active Control of Shock/Boundary Layer Interaction in Transonic Flow over Airfoils", *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, AIAA 2005-486, 2005.

[122] Smith, D.W. and Walker, J.H., "Test of an Area Suction Flap on a NACA64A010 Airfoil at High Subsonic Speeds", 1960, NASA TN D 310.

[123] Qin, N., Zhu, Y. and Shaw, S.T., "Numerical Study of Active Shock Control for Transonic Aerodynamics," *International Journal of Numerical Methods for Heat & Fluid Flow*, Vol. 14 No. 4, 2004, pp. 444-466.

[124] Qin, N., Zhu, Y., Ashill, P. and Shaw, S.T, "Active Flow Control of Transonic Aerodynamics Using Suction, Blowing. Bumps and Synthetic Jets", AIAA 2000-4329, 2000.

[125] Vadillo, J., Agarwal, R.K., "Numerical Study of Transonic Drag Reduction for Flow Past Airfoils Using Active Flow Control," AIAA 2007-712, 2007.

[126] Meunier , M., "Simulations of Flow Control Strategies for Novel High-Lift Configurations," *AIAA Journal*, Vol. 47, No. 5, 2009, pp. 1145-1157.

[127] Yagiz, B. and Kandi O., "Optimization of Active Flow Control in Transonic Aerodynamics", *27th AIAA Applied Aerodynamics Conference*, San Antonio, Texas, AIAA 2009-3763, 2009.

[128] Krist, S.L., Biedron, R.T., and Rumsey, C.L., *CFL3D User's Manual Version 5.0*, June 1998, NASA/TM-1998208444.

[129] *VisualDOC User Manual*, Vanderplaats Research and Development, Inc., Colorado Springs, 2002.

[130] Qin, N., Zhu, and D. I. A., Poll, "Surface Suction on Aerofoil Aerodynamic Characteristics at the Transonic Speeds", *Proceedings of IMechE Part G J. Aeronautical Engineering,* Vol. 212, 1998, pp. 339-351.

VITA

Y. Volkan Pehlivanoglu was born in Erzurum, Turkey in 1973. He received his Bachelor's degree in Aeronautical Engineering from Istanbul Technical University in 1993. He also received a Bachelor's degree in International Relations from Istanbul University. He worked in different units of Turkish Air Force from 1993 to 2005. In 2006, he received his Master of Science degree in Aerospace Engineering from Old Dominion University in Norfolk, VA.