DISSERTATION

A TABU SEARCH EVALUTIONARY ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION: APPLICATION TO A BI-CRITERION AIRCRAFT STRUCTURAL RELIABILITY PROBLEM

Submitted by

Kim Chenming. Long

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2015

Doctoral Committee:

Advisor: William S. Duff
Co-advisor: John W. Labadie

Mitchell Stansloski
Edwin K.P. Chong
Walajabad S. Sampath

ABSTRACT

A TABU SEARCH EVALUTIONARY ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION: APPLICATION TO A BI-CRITERION AIRCRAFT STRUCTURAL RELIABILITY PROBLEM

Real-world engineering optimization problems often require the consideration of multiple conflicting and noncommensurate objectives, subject to nonconvex constraint regions in a high-dimensional decision space. Further challenges occur for combinatorial multiobjective problems in which the decision variables are not continuous. Traditional multiobjective optimization methods of operations research, such as weighting and epsilon constraint methods, are ill-suited to solving these complex, multiobjective problems. This has given rise to the application of a wide range of metaheuristic optimization algorithms, such as evolutionary, particle swarm, simulated annealing, and ant colony methods, to multiobjective optimization. Several multiobjective evolutionary algorithms have been developed, including the strength Pareto evolutionary algorithm (SPEA) and the non-dominated sorting genetic algorithm (NSGA), for determining the Pareto-optimal set of non-dominated solutions.

Although numerous researchers have developed a wide range of multiobjective optimization algorithms, there is a continuing need to construct computationally efficient algorithms with an improved ability to converge to globally non-dominated solutions along the Pareto-optimal front for complex, large-scale, multiobjective engineering optimization problems. This is particularly important when the multiple objective functions and constraints of the real-world system cannot be expressed in explicit mathematical representations. This research presents a novel metaheuristic evolutionary algorithm for complex multiobjective optimization

problems, which combines the metaheuristic tabu search algorithm with the evolutionary algorithm (TSEA), as embodied in genetic algorithms.

TSEA is successfully applied to bicriteria (i.e., structural reliability and retrofit cost) optimization of the aircraft tail structure fatigue life, which increases its reliability by prolonging fatigue life. A comparison for this application of the proposed algorithm, TSEA, with several state-of-the-art multiobjective optimization algorithms reveals that TSEA outperforms these algorithms by providing retrofit solutions with greater reliability for the same costs (i.e., closer to the Pareto-optimal front) after the algorithms are executed for the same number of generations. This research also demonstrates that TSEA competes with and, in some situations, outperforms state-of-the-art multiobjective optimization algorithms such as NSGA II and SPEA 2 when applied to classic bicriteria test problems in the technical literature and other complex, sizable real-world applications. The successful implementation of TSEA contributes to the safety of aeronautical structures by providing a systematic way to guide aircraft structural retrofitting efforts, as well as a potentially useful algorithm for a wide range of multiobjective optimization problems in engineering and other fields.

ACKNOWLEDGEMENTS

It has been long but a fruitful journey that is full of hard work, enlightens, and growing. First of all, I would like to thank my advisor William Duff for his guidance, understanding, patience and support over the years. Secondly, I would like to thank my co-advisor John Labadie for the valuable discussions and guidance on the paper: "Multiobjective Fatigue Life Optimization using Tabu Genetic Algorithms", which is published in the International Journal of Structural Integrity, as well as on this dissertation.

DEDICATION

This dissertation is dedicated to my husband David Long for his continuous support over the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Real-world optimization problems usually face multiple conflicting objectives and large, complex search spaces. This has led to the emergence and increasing popularity of the field of evolutionary multiobjective optimization (EMOO). The advantage of EMOO lies in its ability to devise many solutions in a single step because EMOO operates in terms of generations. There are challenges associated with it as well; including ensuring that the algorithm converges to the Pareto front without becoming trapped in local maxima and that it covers the entire Pareto front with maximum diversities. Ultimately, all EMOO algorithms serve the same purpose: to help the decision maker (DM) choose the best possible solution in a timely manner. A brief review of the basic concepts of multiobjective optimization algorithms is presented in the following sections.

1.1 MOTIVATION FOR THE RESEARCH

Before a new type of military aircraft can go in to production, full-scale fatigue tests are usually conducted to ensure the aircraft has sufficient fatigue life and structural integrity. An aircraft must sustain the entire test without cracking. If test loading conditions are too severe, an optimized retrofitting plan is needed to reallocate limited resources to ensure the completion of the test. Unlike commercial aircraft, which fly regular missions with minimal turbulence for the sake of public safety and comfort, military aircraft may experience some level of turbulence as a mission requirement. The turbulent air flow passes over the wing and impinges on the horizontal stabilizer, causing it to vibrate. The vibration can be a major fatigue driver for the horizontal stabilizer. It is critical for DMs to have several options that balance the increased reliability by retrofitting the aircraft at fatigue-critical locations on the horizontal stabilizer, though there are cost limitations. In recent decades, improvement has been seen in many evolutionary multiobjective optimization algorithms, such as NSGA II versus NSGA and SPEA2 versus SPEA. Many new hybrid algorithms have emerged to solve large, complex multiobjective

system optimization problems. Nevertheless, there is still a lack of efficient multiobjective evolutionary algorithms (MOEAs) that are effective in many different large complex applications. Many real-world multiobjective system optimization problems are nonconvex, and their objective functions are so complex that they cannot even be explicitly modeled. An example is the reliability model of options for retrofitting the aircraft horizontal tail structure.

The purpose of this study is to provide an innovative evolutionary algorithm that applies the concept of metaheuristic tabu search to an evolutionary algorithm, such as the genetic algorithms, (TSEA). To demonstrate its robustness and efficiency, TSEA is compared with state-of-the-art MOEA algorithms such as NSGA II and SPEA2 on both testing problems and real-world applications. Both convex and nonconvex testing problems are included. The input parameters for the real-world applications from two different industries (supplier chain management and aircraft structure design optimization) range from combinatory to real variables. TSEA has shown superior performance on both real-world applications and is well in line with the state of the art on selected testing problems, as shown in chapters 5, 6, and 7.

1.2 MULTIOBJECTIVE OPTIMIZATION BASICS

Engineering system design usually involves conflicting targets. Thus, multiobjective optimization algorithms are required to achieve a good design. To prolong the reliability of the engineering system under multiple constraints, a multiobjective optimization algorithm is feasible because some of the constraints can be treated as objectives. For example, modern aircraft design must minimize cost and weight and maximize reliability. The mathematical model for this multiobjective optimization problem can be stated as follows:

Maximize $\mathbf{z} = [z_1(\mathbf{x}), z_2(\mathbf{x}), \ldots, z_k(\mathbf{x})]$

subject to $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_1(\mathbf{x}), \ldots, y_m(\mathbf{x})] \leq 0$

where $\mathbf{x} = (x_1, x_1, \ldots, x_{n)} \epsilon \mathbf{X}$; $\mathbf{x}$ is the decision vector and $\mathbf{X}$ is the decision space.

$\mathbf{z} = (z_1, z_1, \ldots, z_k) \in \mathbf{Z}$; **z** is the objective vector and **Z** is the objective space.

k is the number of objectives, *m* is the number of constraints, and *n* is the number of decision parameters.

To clearly and effectively continue the discussion of multiobjective optimization, it is essential to define various terms that will be used extensively in this study.

Def. 1 (dominance relation):

Let $f, g \in \mathbf{Z}$. g is said to dominate f (denoted as $f \prec g$) if

$$\forall i \in \{1, \ldots, m\}: g_i \geq f_i$$

$$\exists j \in \{1, \ldots, m\}: g_j > f_j$$

Def. 2 (Pareto set):

Let $F \subseteq \mathbb{R}^m$ be a set of vectors. The Pareto set F* of F is defined as follows:

F* contains all vectors $g \in F$ that are not dominated by any vector $f \in F$

$F^* = \{ g \in F \mid \nexists f \in F : g \prec f \}$

Usually, after a search through a large and complex search space, the solution of the above multiobjective optimization problem is a Pareto set that is either the exact Pareto set or an approximation of it. A certain type of ranking methodology is applied to present the DM with the best solution. The DM can also interface with the search algorithm to guide the search constantly. In the application section of this dissertation, this interaction between DM and the behind-the-scenes search engine will be illustrated.

1.3 TRADITIONAL APPROACHES

This section introduces several multiobjective optimization methods and a ranking method that can help the DM reach a final decision.

1.3.1 WEIGHTING METHOD

The primary idea of the weighting method (also called the weighted sum method) is to choose the weighting coefficients w_i corresponding to objective functions $f_i(x)$, i =1, k. Therefore, the multicriteria optimization problem is transformed to a single-objective one. Many authors have developed systematic approaches to selecting weights. The traditional weighting method combines multiple objectives into a single objective as follows:

Maximize $z = w_1 * f_1(\boldsymbol{x}) + w_2 * f_2(\boldsymbol{x}) + \ldots + w_k * f_k(\boldsymbol{x})$

subject to $\boldsymbol{g(x)} \leq 0$

One of difficulties with the weighting method is that varying the weights consistently and continuously may not necessarily result in an accurate, complete representation of the Pareto optimal set. Without loss of generality, weights are normalized to satisfy the following equation:

$$\sum \omega_i = 1$$

A set of Pareto-optimal solutions can be obtained through choosing different combinations of weights. The disadvantage of this method is that it cannot reach the complete Pareto set if the objective space is nonconvex.

1.3.2 EPSILON CONSTRAINT METHOD

The epsilon-constraint method overcomes some of the convexity problems of the weighted sum technique. The idea of this method is selecting one objective as ultimate and turning all other objectives into constraints.

Max f_1

Subject to:

$f_i \geq \epsilon_i$ i = 2, . . . , k

$\boldsymbol{g(x)} \leq 0$

Here, all objectives are maximized. The lower bounds εi are used as variables to find multiple Pareto-optimal solutions. Because the objective functions are no longer limited to linear functions, this method is not limited in its weighting method to convex trade-off surfaces. This approach enables the identification of a number of non-dominated solutions on a nonconvex boundary that are not obtainable using the weighted sum technique. A problem with this method, however, lies in finding a suitable selection of i to ensure a feasible solution. A further disadvantage of this approach is that the use of hard constraints is rarely adequate for expressing true design objectives. The optimization proceeds with reference to these priorities and allowable bounds of acceptance. The difficulty here is in expressing such information at early stages of the optimization cycle.

1.3.3 COMPROMISE PROGRAMMING

This ranking method produces compromise solutions by measuring the distances to various "ideal" critical levels. The most popular ways to rank closeness are L^1, L^2, and L^∞.

L^1: $\quad L^1(j) = \sum_{i=1}^{m} w_i \left| \frac{z_i^* - z_i(a_j)}{z_i^* - z_i^{**}} \right|$

Subject to: $x < X$ (constraints on decision variables)

In which L^1 corresponds to the absolute value norm.

Figure 1.1 illustrates what the equivalue contours of the L^1 norm would look like, with the ideal solution assumed to be infeasible, as shown. For the L^2 norm, the problem is formulated as:

L^2: $\quad L^2(j) = \min \sum_{i=1}^{m} w_i^2 \left[\frac{z_i^* - z_i(a_j)}{z_i^* - z_i^{**}} \right]^2$

which corresponds to the L^2 norm.

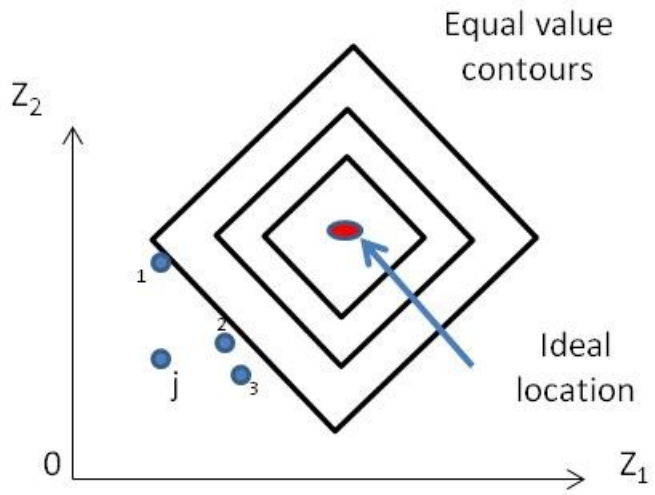

Figure 1.1 Absolute value norm

Figure 1.2 illustrate the equi-valued contours of the L^2 norm, and a different solution is obtained with the L^2 norm, in this case, point 2.

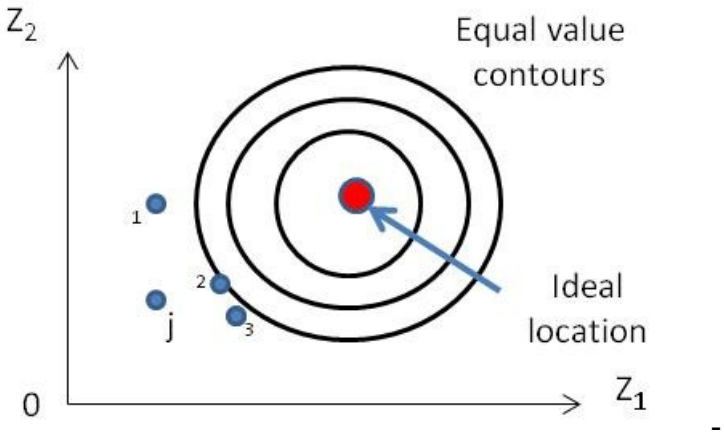

Figure 1.2 Least squares norm

L^{∞}: $L^{\infty}(j) = max_{1 \le i \le m} w_i \left| \frac{z_i^* - z_i(a_j)}{z_i^* - z_i^{**}} \right|$

This value corresponds to the min-max norm, as shown in figure 1.3.

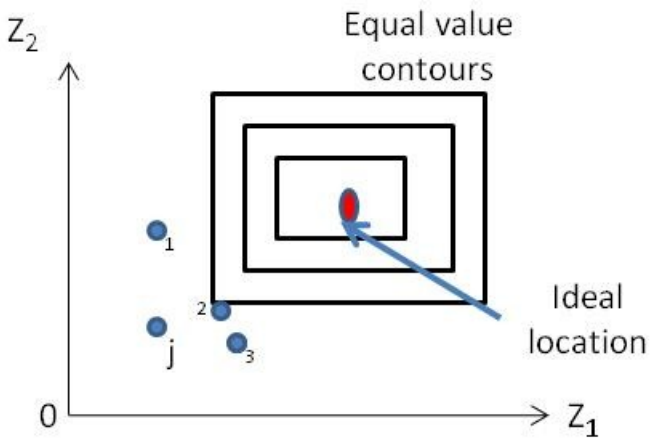

Figure 1.3 Min-max norm

1.4 NONTRADITIONAL APPROACHES

In contrast to the traditional approaches presented in the above section, there are two main branches of nontraditional approaches: evolutionary and heuristic.

1.4.1 EVOLUTIONARY APPROACH

The evolutionary algorithm (EA) represents a collection of optimization algorithms that simulate the process of natural evolution. The advantage of EAs lies in the fact that they operate on a set of candidate solutions, called generations. Multiobjective evolutionary algorithms (MOEAs) search for Pareto-optimal solutions using the processes of selection and variation, which are performed on each generation. In EAs, natural selection is simulated by a stochastic selection process. Each solution is given a chance to reproduce based on its "fitness." The variation process imitates the natural capability to create new generations by means of crossover and mutation. Here, "crossovers" are operated on portions of chromosomes through the process of switching over. Mutations are accomplished by flipping a single, randomly selected gene within a chromosome.

MOEAs date back to 1985, when Shaffer (1985) proposed an extension of the simple genetic algorithm (SGA) to accommodate vector-valued fitness measures, thus creating the

vector-evaluated genetic algorithm (VEGA). VEGA was based on subpopulations developed for each objective function. A new population was generated by selecting individuals from each subpopulation according to their fitness values. In particular, the selection step of the algorithm was modified. At each generation, several subpopulations were created through the process of proportional selection, according to each objective function. Thus, for a problem with p objectives, p subpopulations of size N/p would be generated (assuming a population size of N).

To avoid the necessity of combining objective functions in any way, Fonseca and Fleming (1993) described a rank-based fitness assignment method for multiple-objective genetic algorithms (MOGAs). Fonseca and Fleming thought that the external DM should be actively involved with the MOGA, and that the interaction between the two would lead to the determination of a satisfactory solution for the problem at hand. This study additionally demonstrates the best way to sample regions of the trade-off surface uniformly. Fonseca and Fleming (1993) proposed a Pareto-based ranking procedure assuming all objectives were minimizing objectives and individuals ranked according to the number of decision variables they dominate. Consider an individual x at t generation, which is currently dominated by p individuals. Its rank is shown in figure 1.4 and defined by $\text{Rank}(x, t) = 1 + p$.

The population is sorted according to its rankings, and a raw fitness value is assigned to each individual, as shown in figure 1.5 as dashed lines. The fitness of an individual is linearly interpolated between the best (rank 1) and the worst. As shown in figure 1.5, the final fitness value of each individual is averaged, as shown via the solid lines, and shared among individuals with the same ranking.

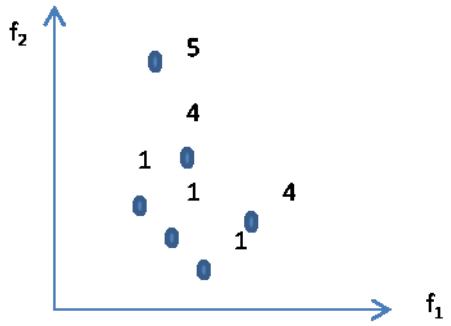

Figure 1.4 Pareto ranking

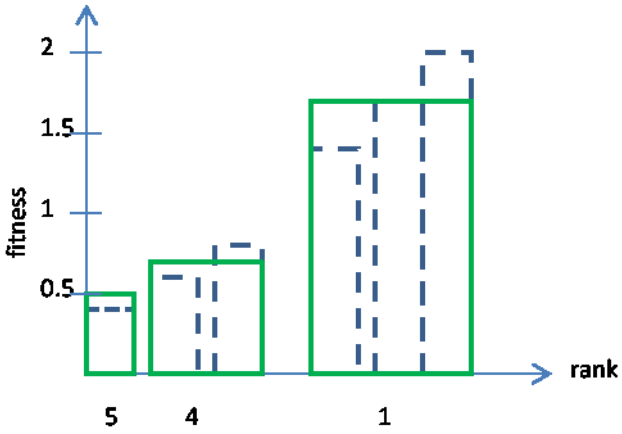

Figure 1.5 Rank-based fitness assignments

Horn (1993) proposed another niche Pareto genetic algorithm (NPGA) and introduced the concepts of partial order optimization, equivalence class sharing, and nested sharing. The following sections will examine the popular NPGA in greater detail.

After a decade of pioneering work in the area of MOGAs, Deb (1999) began to explore the difficulties involved in such algorithms and to construct testing problems for researchers in this area. Deb summarized the difficulties inherent in converging to the Pareto-optimal front as being multimodality, deception, isolated optimum, and collateral noise.

To overcome those difficulties, Zitzler (2000) compared six multiobjective evolutionary algorithms and devised several empirical outcomes, which were performed on six testing problems. Zitzler ranked these algorithms as follows:

1. SPEA (strength Pareto evolutionary algorithm) (Zitzler and Thiele,1999)
2. NSGA (non-dominated sorting genetic algorithm) (Srinivas and Deb,1995)
3. VEGA (vector-evaluated genetic algorithm)(Schaffer's,1984)
4. HLGA (EA proposed by Hajela and Lin (1992), based on aggregation selection and fitness sharing, where an individual is assessed by summing up the weighted objective values)
5. NPGA (niched Pareto genetic algorithm)(Horn1993)
6. FFGA (Fonseca and Fleming's multiobjective genetic algorithm) proposed by Fonseca and Fleming (1993)

Zitzler's (2000) six testing problems have become popular benchmark testing problems for many researchers in the field of MOEAs. Detailed definitions of those testing problems are given in chapter 6.

Regarding the implementations of those six algorithms, one chromosome was used by Zitzler to encode the *m* parameters for the corresponding test problem. All approaches except FFGA were realized using binary tournament selection with replacement to avoid effects caused by different selection schemes. Because FFGA requires a generational selection mechanism, stochastic universal sampling was used in the FFGA algorithm. Tan (2001) also performed MOEA performance assessments and comparisons. The overall simulation results showed that none of the methods was superior, considering all aspects of the performance measures.

1.4.2 META-HEURISTIC ALGORITHMS

Many comparative studies have been performed to identify the best solution technique in the area of metaheuristic methods applied to system optimization problems. By examining four approaches of solving large-scale problems related to maximum expected coverage location, Aytug (2002) found that the genetic algorithm (GA) approach generated high-quality solutions in predictable periods of time. Dorn et al. (1996) examined four heuristic methods for the schedule optimization problem: iterative deepening, random search, tabu search (TS), and genetic algorithms. Iterative deepening is a combination of depth-first backtracking search and breadth-first search. It has the advantages of requiring only linear space and guaranteeing discovery of the shortest path to an improvement.

Moreover, Dorn et al. developed a methodology to gradually satisfy given controversial constraints, which allowed them to reach the results in a timely manner. They applied these techniques on data from a steel-making plant in Linz, Upper Austria. Their application was constrained by a greater variety of antagonistic criteria that were partly contradictory. The constraints included compatibility and due date. The work was developed through the attempted building of reusable scheduling software. The results showed that, for a given application, iterative deepening and tabu search performed better than random search and genetic algorithms. This conclusion was based on an experimental comparison of four iterative improvement techniques for schedule optimization: iterative deepening, random search, tabu search, and genetic algorithms. They compared the performance of these techniques by using the same evaluation function, knowledge representation, and data. The evaluation was conducted by gradually satisfying explicitly represented domain constraints and optimization functions. They weighted and aggregated satisfactions of individual constraints for the whole schedule.

Youssef (2001) examined three general iterative algorithms for combinatorial optimization problems: evolutionary algorithms, simulated annealing (SA), and tabu search. Youssef summarized the following similarities among the three optimization heuristics:

- They are approximation algorithms and do not guarantee the finding of an optimal solution.
- They are blind and do not know when to stop.
- They have a hill-climbing property, such that they accept uphill moves occasionally.
- They are general and can be easily applied to any combinatorial optimization problem, under certain conditions; they asymptotically converge to an optimal solution.

Aytug's (2002) comparative study on facility location allocation optimization problems, focused on GAs with nonlinear objective functions. He coded two types of GAs, GALS and GAs, compared and the results with those from Daskin's heuristic method. Arostegui (2006) compared the relative performance of TS, SA, and GA on various types of facility location problems (FLP). In this comparison, three FLP variations were examined: capacitated FLP, multi-period FLP, and multi-commodity FLP.

1.5 CHAPTER BREAKDOWN

Following this chapter's review of the existing literature on traditional and nontraditional approaches to multiobjective optimization, the rest of this dissertation provides a detailed study of multiobjective system reliability optimization, ranging from algorithm development (comparing it with the state of the art) to industry applications of those algorithms. Chapter 2 briefly reviews multiobjective optimization algorithms. The aircraft structural reliability issues are introduced in chapter 3 to lay the foundation for the creation of the new algorithm TSEA,

which is introduced in chapter 4 and applied to the aircraft structural reliability and cost model in chapter 5. Embedded in a decision support system, TSEA can provide the DM with Pareto-optimal aircraft tail structure retrofitting solutions to maximize reliability and minimize the retrofitting cost. Chapter 6 presents comparisons of TSEA with the state of the art on benchmark testing problems. A supplier chain management application is analyzed using TSEA, and the results are compared with those from state-of-the-art MOEA algorithms (NSGA II and SPEA2) in chapter 7. To demonstrate how TSEA can be used to provide Pareto-optimal solutions to DMs, this new algorithm is also embedded in a decision support system (DSS) developed to provide aircraft tail structure retrofitting options to maximize the structure's reliability and minimize its cost. Chapter 8 delivers the conclusions and future research directions.

1.6 PUBLICATIONS

K. C. Long, W. S. Duff, J. W. Labadie, M. J. Stansloski, W. S. Sampath, and E. K. P. Chong, **"**Multiobjective Fatigue Life Optimization Using Tabu Genetic Algorithms," accepted by the *International Journal of Structural Integrity*.

CHAPTER 2: MULTIOBJECTIVE OPTIMIZATION ALGORITHMS

2.1 OVERVIEW OF MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS

The previous section provided a general overview of evolutionary algorithms. This section will focus on multiobjective evolutionary algorithms. An overview is presented to lay the foundation for their applications in chapters 5 through 7. Schaffer (1984, 1985) presented an MOEA called the vector-evaluated genetic algorithm (VEGA), which selects the optimized outcome by switching objectives. Here, selections are undertaken for each of the k objectives separately, filling equally sized portions of the mating pool. Then the mating pool is shuffled before the crossovers and mutations are performed as usual. Despite having some known serious drawbacks (Fonseca and Fleming 1995b), this algorithm has been a strong point of reference up until now.

Hajela and Lin (1992) advocated another approach that belongs to category aggregation selection with parameter variation and is based on the weighting method. The weights are encoded in each individual's vector. Hence, each individual is evaluated in relation to a potentially different weight combination. This method is inherently biased toward the convex portion of the trade-off front. But due to its simplicity, weighted-sum aggregation is still fairly widespread.

Fonseca and Fleming (1993) proposed a Pareto-based ranking procedure in which individuals are ranked according to the number of dominating decision variables. The population is sorted according to these rankings, and a raw fitness value is assigned to each individual. The final fitness value of each individual is averaged and then shared among those individuals with the same rankings.

The next three algorithms (SPEA, NPGA, and NSGA) are relatively recent and more popular than other evolutionary multi-objective optimization algorithms, and for those reasons

were chosen for more detailed examinations in the following sections. The new algorithm proposed in this dissertation will be compared to these algorithms to demonstrate its superiority. Figure 2.1 shows the relationships among individual space, decision space, and objective space. Individuals are associated with the objective vector Y through the decision vector X. Each individual is mapped to the decision variables by the function m(i), and decision variables are linked with objectives by the function f(X).

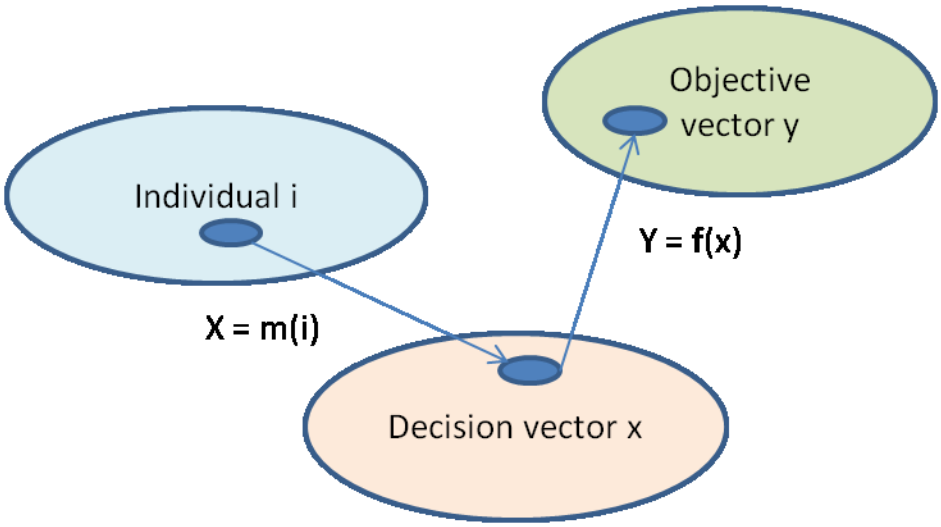

Figure 2.1 Relationship between spaces

2.1.1 SPEA AND SPEA2 ALGORITHMS

Zitzler (1999) proposed the evolutionary algorithm for multiobjective optimization, the strength Pareto evolutionary algorithm (SPEA). This algorithm includes an external set of non-dominated fronts, and the fitness of a population member was determined only from the individuals stored in the external set. All of those individuals participated in the selection for the next generation. A new Pareto-based niching method was used to preserve diversity in the population. If the size of the updated archive exceeded a predefined limit, more archive members were deleted through a clustering technique, which preserved the characteristics of the non-

dominated front. Afterward, fitness values were assigned to members of both the archive and the population:

- Each individual *i* in the archive was assigned a strength value *S(i)*, which represented its fitness value *F(i)*. *S(i)* was the number of population members *j* that were dominated by *i* with respect to the objective values, divided by the population size plus one.
- The fitness *F(j)* of an individual *j* in the population was calculated by summing the strength values *S(i)* of all archive members *i* that dominate *j*, and then adding one at the end.

The mating selection was accomplished by means of binary tournaments. The fitness was minimized. Each individual in the archive had a higher chance of being selected than any population member. Finally, following crossovers and mutations, the old population was replaced by the resulting offspring population.

The details of the SPEA are shown below:

SPEA

Input: N --- population size

N' --- maximum size of external set

T --- maximum number of generations

P_t --- generation t

$\bar{P}t$ --- external set t

P_c --- crossover probability

P_m --- mutation rate

Output: A --- non-dominated set

Step 1: Initiation:

Generate an initial population P_o with probability of 1 (i.e., pick N data points between a and b randomly, which amount to the population size N), create the external set $\bar{P}o = Ø$, and set t = 0.

Step 2: Update of external set:

Set the temporary external set $P' = P_t$.

Copy individuals whose decision vectors are non-dominated regarding $m(P_t)$ to P':

$$P' = P' + \{i |\, i \in P_t \cap m(i) \in p(m(P_t))\}$$

As long as there exists a pair (i, j) with $i, j \in P'$ and $m(i) \geq m(j)$, then $P' = P' - \{j\}$.

Reduce the number of externally stored individuals by means of clustering.

Call the clustering algorithm with parameters P' and $\bar{N}$ and assign the resulting reduced set to $\bar{P}_{t+1}$.

Step 3: Fitness assignment:

Calculate the fitness values of the individuals in P_t and $\bar{P}t$ by the following steps:

Each individual $i \in \bar{P}t$ is assigned a real value $S(i) \in [0,1]$, called its strength; $S(i)$ is proportional to the number of population members $j \in Pt$ for which $m(i) \geq m(j)$:

$$S(i) = \frac{|\{j \,|j \in Pt \ \cap m(i) \geq m(j)\}|}{N+1}$$

The fitness of i is equal to its strength: $F(i) = S(i)$.

The fitness of an individual $j \in P_t$ is calculated by summing the strengths of all externally stored individuals $i \in \bar{P}t$ whose decision vectors weakly dominate $m(j)$. Zitzler added one to the total to guarantee that members of $\bar{P}$t have better fitness than members of P_t (the smaller the fitness, the better):

$$\mathrm{F(j)} = 1 + \sum_{i \in \bar{P}t, m(i) \geq m(j)} S(i) \quad \text{where } \mathrm{F(j)} \in [1, N]$$

Step 4: Selection:

Set $P' = \emptyset$. For $i = 1, \ldots, N$, do:

Select two individuals $i, j \in Pt + \bar{P}t$ at random.

If $F(i) < F(j)$, then $P' = P' + \{i\}$; otherwise, $P' = P' + \{j\}$.

Step 5: Crossover:

Set $P'' = \emptyset$ for $i = 1, \ldots, N/2$, do:

Randomly choose two individuals $i, j \in P'$ and remove them from P'.

Randomly select a cut point and cross over at the selected cut to create children $k, l \in I$.

Add k, l to P'' with probability p_c.

Step 6 Mutation:

Set $P''' = \emptyset$ for each individual $i \in P''$, then do:

Mutate i with mutation probability p_m.

Set $P''' = P''' + \{i\}$.

Step 7 Termination:

Set $P_{t+1} = P'''$ and $t = t+1$. If $t \geq T$ or another stopping criterion is satisfied, then set $O = p(m(\bar{P}t))$; otherwise, go to step 2.

Constraints handling:

Zitzler(1999) implemented a greedy repair method that produced the best outcome among all algorithms applied to the multiobjective 0/1 knapsack problem. The repair was based upon a predefined scheme. Because many coding lead to infeasible solutions, the mapping function m(i) defined a simple repair method that decodes an individual i according to the following scheme: set x = i; then remove step by step items from x as long as any capacity constraints was violated.

The order in which the items were deleted was determined by the maximum profit/weight ratio per item: for item j the maximum profit/weight ratio q_j was determined by:

$$q_j = max_{i=1}^{n} \left\{ \frac{p_{i,j}}{w_{i,j}} \right\}$$

The items were considered in increasing order of the q_j , those items achieving the lowest profit per weight unit were removed first. The mechanism intends to fulfill the capacity constraints while diminishing the overall profit as little as possible.

One drawback of the SPEA is that, when many individuals have the same fitness, the algorithm becomes inefficient. Noticing this possible deficiency, Zitzler (2001) improved the SPEA by providing the following elements:

- An improved fitness assignment scheme
- A nearest-neighbor density estimation technique
- New archive truncation methods that guarantee the preservation of boundary solutions

Unlike SPEA, the newly improved algorithm SPEA2 uses a fine-grained fitness assignment strategy that incorporates density information. Furthermore, the archive size is fixed. When the number of non-dominated individuals is less than the predefined archive size, the archive is filled by dominated individuals; with SPEA, however, archive size may vary over time. Moreover, the clustering technique, which is invoked when the non-dominated front exceeds the archive limit, is replaced by an alternative truncation method, which has similar features but does not lose boundary points. Another difference between SPEA and SPEA2 is that the latter only has members of the archive participate in the mating selection process.

In SPEA2, to avoid the situation in which individuals dominated by the same archive members have identical fitness values, both the dominating and dominated solutions are taken

into account for each individual. Each individual *i* in archive $\overline{P_t}$ and population P_t is assigned a strength value *S(i)*, which represents the number of solutions it dominates:

$$S(i) = |\{j \mid j \in P_t + \overline{P_t} \wedge i \propto j\}|$$

where |.| denotes the cardinality of a set, + stands for a multi-set union, and the symbol $\propto$ corresponds to the Pareto dominance relationship. The raw fitness *R(i)* of an individual *i* is calculated as follows:

$$R(i) = \sum_{j \in P_t + \overline{P_t}} S(j)$$

Here, raw fitness is determined by the strength of the dominators in both the archive and the population, whereas in SPEA only archive members are considered in this content. Fitness is minimized here. Although the raw fitness assignment provides a niching mechanism based on the concept of Pareto dominance, it may fail when most individuals fail to dominate one another. Additional density information is incorporated to allow for discrimination among individuals with the same raw fitness values. This is accomplished by introducing a density parameter called the *k*th nearest-neighbor method (Silverman 1986), in which the density at any point is a function of the distance to the *k*th nearest data point.

Suggested by Silverman, the parameter *k* is chosen as such:

$$k = \sqrt{N + \overline{N}}$$

For each individual *i*, the distances in objective space to all individuals *j* in archive and population are calculated and stored in a matrix. After sorting the matrix in increasing order, the *k*th element gives the distance sought, denoted as σ_i^k. The density *D(i)* corresponding to *i* is defined by

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

Two is added to the denominator to ensure that its value is greater than 0 and that *D(i)* < 1. Finally, *D(i)* is added to the raw fitness value *R(i)* of an individual *i*, which yields its fitness *F(i)*:

$$F(i) = R(i) + D(i)$$

All non-dominated individuals (i.e., those have fitness lower than 1), from archive and population are added to the archive of the next generation:

$$\overline{P_{t+1}} = \{i \mid i \in P_t + \bar{P}_t \cap F(i) < 1\}$$

In SPEA2, this archive update operation (step 3 in the SPEA algorithm) differs in two areas. First, the number of individuals contained in the archive is constant over time. When the number of individuals selected to fill the archive is less than a predefined archive size, the best-dominated individuals in the previous archive and population are copied to the new archive.

If the archive is too large, the truncation method is employed iteratively to prevent the removal of boundary solutions. At the first iteration, a distance matrix is created between every two individuals (sample distance matrix of 12 individuals) in the archive and sorted in ascending order, as shown in table 2.1.

Table 2.1 Sample Distances between 12 Individuals

	1	2	3	4	5	6	7	8	9	10	11	12
1	230	1410	2200	3420	3660	4290	8120	9440	9660	10250	10700	11790
2	3590	6030	8320	9390	14410	18800	21170	22670	25200	26460	28290	30130
3	6620	10900	11140	12180	12290	14560	15570	17160	17550	20430	21610	21840
4	1740	3530	8730	10340	11440	15810	16450	22700	34060	57460	75860	82160
5	2980	5480	8740	8910	11100	12430	15220	15830	19150	20060	20220	23510
6	50	570	1920	3620	4520	4590	5880	7470	9140	10120	10710	13620
7	1040	2790	3360	4700	5300	10360	11900	12710	12810	13000	14030	18490
8	240	1220	3420	3650	4700	4830	6020	6830	7710	8370	13080	13430
9	810	1540	2130	6590	6600	6830	8050	10250	10480	11660	11900	12940
10	900	3050	3570	3620	5520	5540	6500	8210	9500	10000	11090	11740
11	910	4230	8960	14580	17060	17310	20060	20980	22240	23040	24770	26270
12	380	6300	8520	24700	27850	30790	39270	58170	59240	63970	67560	71820

Starting from the first column of table 2.1, the individual that has the smallest distance will be taken out to reduce cluster. When multiple individuals have the same distance, the selection process is re-sorted to the next column, and so on. The same procedure continues until the total number in the next archive equals the predefined value. The constraint handling in SPEA2 was the same as that in SPEA which used a greedy repair method.

2.1.2 NSGA AND NSGA II ALGORITHMS

Srinivas and Deb (1994) developed the most direct implementation of Goldberg's sketch. The different trade-off fronts in the population were peeled off step by step, and fitness sharing was performed separately for each front. The algorithm is detailed as follows:

Input: N --- population size

N_{max} --- maximum number of non-dominated solutions

T --- maximum number of generations

P_c --- crossover rate

P_m --- mutation rate

$\sigma shear$ --- niche radius

t_{dom} --- domination pressure

P_t --- generation t

Output: O --- non-dominated set

Step 1: Initialization:

Set $P_o = \emptyset$ and $t = 0$.

For $i = 1, \ldots, N$, do:

Choose $i \in I$ randomly.

Set $P_o = P_o + \{i\}$.

Step 2: Fitness assignment and selection of mating pool:

Set $P_{remain} = P_t$ and initialize the dummy fitness value F_d with N.

Determine set P_{nondom} of individuals in P_{remain} whose decision vectors are non-dominated with regard to $m(P_{remain})$.

Delete P_{nondom} in the further classification process ($P_{remain} = P_{remain} - P_{nondom}$).

Set the raw fitness of individuals in P_{nondom} to F_d and perform fitness sharing in the decision space (only within P_{nondom}).

Decrease the dummy fitness value F_d, such that it is lower than the smallest fitness in P_{nondom}:

$0 < F_d < \min\{F(i) \mid i \in P_{nondom}\}$

If $P_{remain} \neq \emptyset$, then go to (*b*); otherwise, stop.

Step 3: Crossover:

Set $P_2 = \emptyset$ for $i = 1, \ldots, N/2$

do:

Randomly choose two individuals $i, j \in P_1$ and remove them from P_1.

Randomly select a cut point, then cross over at the selected cut to create children $k, l \in I$.

Add k, l to P_2 with probability p_c.

Step 4: Mutation:

Set $P_3 = \emptyset$, then, for each individual $i \in P_2$, do:

Mutate i with mutation probability *pm*.

Set $P_3 = P_3 + \{i\}$

Step 5: Termination:

Set $P_{t+1} = P_3$ and $t = t + 1$

If $t \geq T$ or the number of non-dominating individuals> N_{max}, then set $O = f(m(P_{t+1}))$; otherwise, go to step 2.

Srinivas and Deb(1995) transformed the testing problem F3 used into an unconstrained optimization problem using an exterior penalty function. This problem was used to test NSGA's ability in optimizing multi-parameter problems as well as handling constrained search spaces.

F3: Minimize $f_{31} = (x_1 - 2)^2 + (x_2 - 1)^2 + 2$, and

$$f_{32} = 9x_1 - (x_2 - 1)^2$$

Subject to $x_1^2 + x_2^2 - 225 \leq 0$

$$x_1 - 3x_2 + 10 \leq 0$$

Both objectives were penalized if any point lies in the infeasible region. The exterior penalty function is shown below:

$$P(x) = \sum_{j=1}^{m} max\left[0, g_j(x)\right]^2 + \sum_{k=1}^{n} h_k(x)^2$$

Where g(x) is inequality constraints, and h(x) is equality constraint.

Deb (2002) introduced a fast and elitist MOGA algorithm, the non-dominated sorting genetic algorithm (NSGA II), which improved on the original non-dominated sorting genetic algorithm (NSGA) in the following ways: finding a diverse set of solutions and converging near the true Pareto-optimal set.

To begin NSGA II, one first randomly generates a population P_1 with a size of N_p solutions and then sorts the solutions in P_1 into several fronts of non-dominated solutions. To preserve the diversity of the non-dominated solutions, a crowding distance is evaluated. The original NSGA used the well-known fitness-sharing approach, which has been found to maintain sustainable diversity in a population, given appropriate settings of the associated parameters. The sharing function method involves the sharing parameter σ, which sets the extent of sharing

desired in a problem. There are two challenges with this sharing function approach: the performance of the algorithm largely depends on the sharing parameter set by the user, and, because each solution must be compared with all other solutions in the population, the overall complexity of the sharing function approach is $O(N^2)$.

In NSGA II, the sharing-function approach is replaced by a crowding-comparison approach that eliminates both of the above difficulties to some extent. This new approach does not require any user-defined parameter to maintain diversity among population members. The crowding-distance computation requires that the population be sorted in ascending order according to each objective function value. Thereafter, for each objective function, the boundary solutions are assigned large values. All other solutions are assigned distance values equal to the absolute normalized difference in the function values of two adjacent solutions, as shown in figure 2.2. The new generation of solutions is selected according to both front values and crowding distances. Figure 2.2 illustrated the components of crowding distance in the case of two objective functions. The crowding distance at point i is the summation of two fractions:

$$C(i)_{distance} = \frac{f1(i+1)-f1(i-1)}{\Delta 1} + \frac{f2(i+1)-f2(i-1)}{\Delta 2}$$

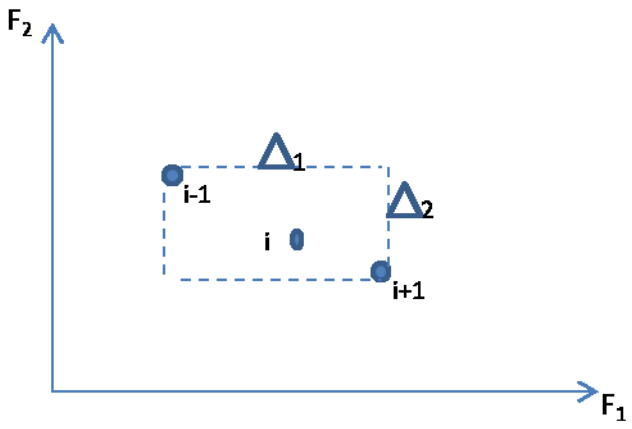

Figure 2.2 Crowding distances (not normalized)

Considering the chromosomes obtained using the tournament selection operator for P_1, the offspring population O_1 is created with respect to the crossover rate (P_c) and the mutation rate (P_m). After merging P_1 and O_1 to form R_t, the algorithm sorts R_t in several non-dominated fronts F_i, where the best F_is form the next population P_{t+1}. This process is illustrated in figure 2.3.

Between two solutions with different non-domination ranks, the solution with the lower (better) rank is selected. Otherwise, if both solutions belong to the same front, the solution that is located in the less crowded region prevails. To maintain a size of N_p for the next generation, not all solutions are kept. The selection process is illustrated in figure 2.4.

In a recent study, Bhattacharya (2010) used NSGA II to solve a conflicting bi-objective facility location problem. With the application of NSGA II, Bhattacharya was confident that more objectives could be introduced into the model in the future, along with their uncertainties or fuzziness. The following naming convention was used in the NSGA II algorithm, as shown in figure 2.3:

N_p --- number of population

I_t --- number of total iterations

P_c --- probability of crossover

P_m – probability of mutation

P_t --- parents' population of generation t

O_t ---- offspring population of generation t

P_{t+1}---parents' population of generation t+1

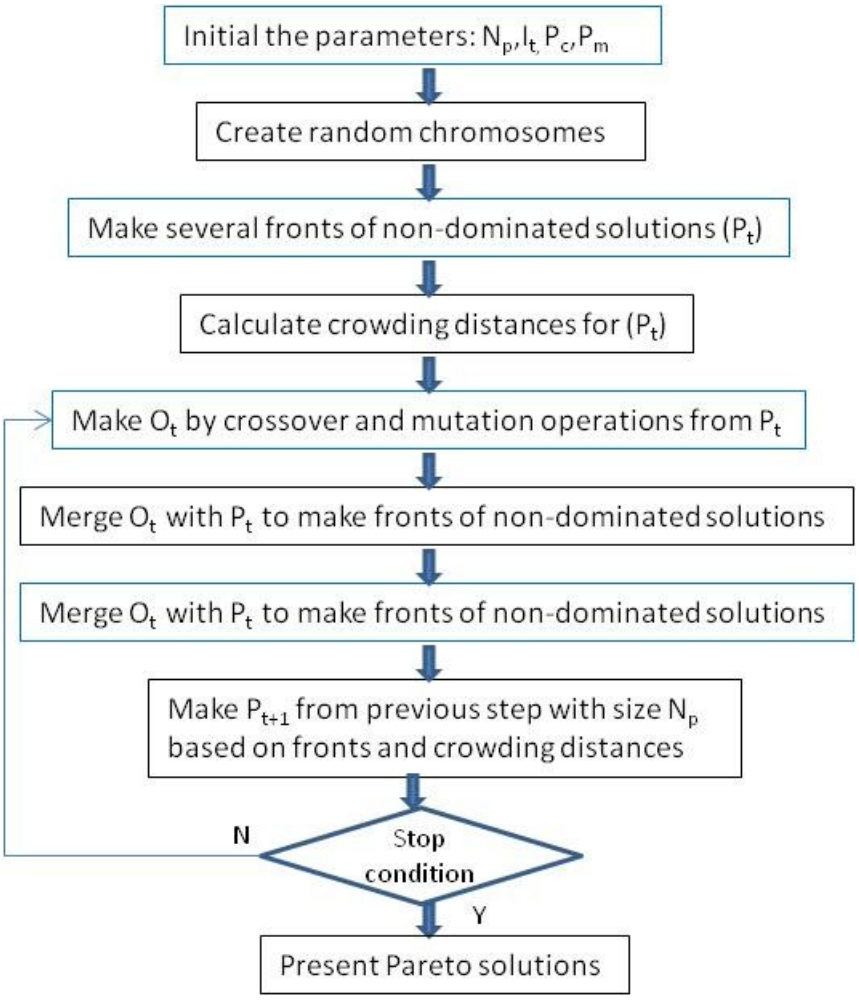

Figure 2.3 NSGA II algorithm

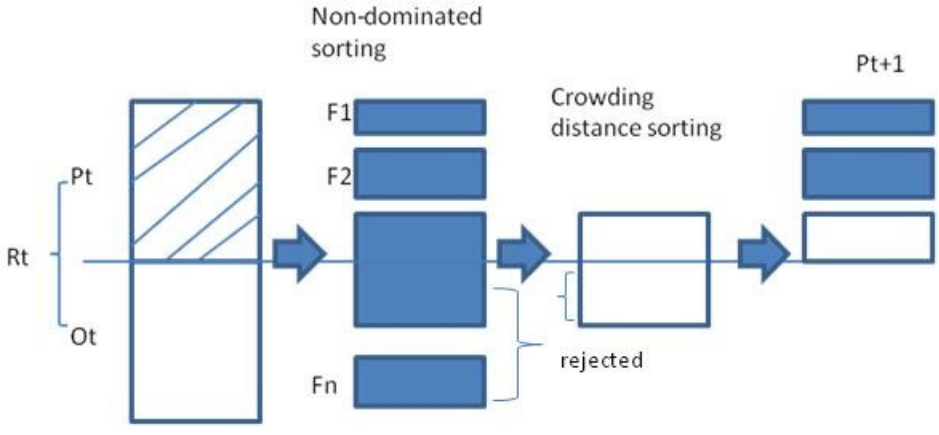

Figure 2.4 NSGA II new generation selection process

Figure 2.4 shows the progression from generation t to generation t+1 in NSGA II. First, a combined population $R_t = P_t \cup O_t$ is formed. Then the population Rt is sorted according to non-domination. Solutions belonging to the best non-dominated set F_1 are the best solutions in the combined population. If the size of F_1 is smaller than N, solutions from the set F_2 are chosen next, followed by solutions from the set F_3, and so on. To choose exactly N population members, the solutions of the last front F_l are sorted by the crowding distance, and the best solutions needed to fill all population slots are chosen as shown in figure 2.4.

2.1.2 CONSTRAINT HANDLING BY NSGA II

Deb (2002) proposed a constraint-handling method using the binary tournament selection, in which two solutions were picked from the population and there may be, at most, three situations:

1. Both solutions are feasible.
2. One is feasible and other is not.
3. Both are infeasible.

Deb gave the definition of constraint dominance such that solution i is said to be a constrained-dominate solution j if:

1) Solution i is feasible and solution j is not;
2) Solutions i and j are both infeasible, but solution i has a smaller overall constraint violation; or
3) Solutions i and j are feasible, and solution i dominates solution j.

The effect of using this constrained-domination principle is that any feasible solution has a better non-domination rank than any infeasible solution. All feasible solutions are ranked according to their non-domination level based on the objective function values. Among two

infeasible solutions, the one with a smaller constrain violation has a better rank. An infeasible solution's having a larger overall constraint-violation was classified as a member of a larger non-domination level.

Sadeghi et al. (2013) implemented NSGA II in the field of supply chain management (SCM) in the context of vendor-managed inventory (VMI). One of the most important issues in SCM is inventory management. Many strategies have been attempted, including quick response (QR), advanced continuous replenishment (ACR), and VMI. Sadeghi extended Zavanella's (2009) VMI model and proposed a bi-objective VMI of a supply chain consisting of a single vendor and multiple retailers. The first objective was the minimization of total chain cost, including production, ordering, holding, and transportation. The second objective was the maximized reliability of the production system using the redundancy allocation problem approach. The goal was to determine the order size, the replenishment frequency of the retailers, the routing tour, and the number of machines of different types in series. Through this approach, the above two objectives were simultaneously optimized.

In addition, there were five constraints. First, the retailer's warehouse had a predetermined, limited space; second, the average inventory level of the vendor was restricted to an upper bound; third, the replenishment frequency was limited; fourth, the space required for machine installation was constrained; and fifth, there was an upper bound for the budget to provide the machines. Moreover, the vendor determined the shortest route to deliver goods in order to reduce transportation costs. A comparison of three different types of algorithms, including the new algorithm introduced in this dissertation, which can be used to solve this multiobjective optimization problem, is shown in chapter 7.

2.1.3 NPGA Algorithm

The niched Pareto genetic algorithm (NPGA), proposed by Horn and Nafpliolis (1993), combines tournament selection and the concept of Pareto dominance in the following way:

Input: N --- population size

$Nmax$ --- maximum number of non-dominated solutions

T --- maximum number of generations

P_c --- crossover rate

P_m --- mutation rate

$\sigma shear$ --- niche radius

t_{dom} --- domination pressure

P_t --- generation t

Output: O --- non-dominated set

Step 1: Initialization:

Set $P_o = \emptyset$ and $t = 0$.

For $i = 1, \ldots, N$, do:

Choose $i \in I$ randomly.

Set $P_o = P_o + \{i\}$.

Step 2: Fitness assignment and selection of mating pool:

Set $i = 1$ and $P_1 = \emptyset$.

Select two competitors $i, j \in P_t$ and a comparison set $P_{dom} \in Pt$ of t_{dom} individuals at random.

If *m(i)* (decision variable) is non-dominated with respect to P_{dom} and *m(j)* is not, then i is the winner of the tournament.

$$P_1 = P_1 + \{i\}$$

Otherwise, if *m(j)* (decision variable) is non-dominated with respect to P_{dom} and *m(i)* is not, then *j* is the winner of the tournament.

$$P_1 = P_1 + \{j\}$$

Otherwise, the tournament is decided by fitness sharing:

Calculate the number of individuals in the partially filled mating pool that are within $\sigma shear$ distance of *i* and *j*:

$$n(i) = \{k \mid k \in P_1 \cap d(i,k) < \sigma shear\}$$

$$n(j) = \{k \mid k \in P_1 \cap d(j,k) < \sigma shear\}$$

If $n(i) < n(j)$, then $P_1 = P_1 + \{i\}$; otherwise, $P_1 = P_1 + \{j\}$.

Step 3: Crossover:

Set $P_2 = \emptyset$ for $i = 1, \ldots, N/2$, then do:

Randomly choose two individuals $i, j \in P_1$ and remove them from P_1.

Randomly select a cut point, then cross over at the selected cut to create children $k, l \in I$.

Add k, l to P_2 with probability p_c.

Step 4: Mutation:

Set $P_3 = \emptyset$ for each individual $i \in P_2$, then do:

Mutate *i* with mutation probability p_m.

Set $P_3 = P_3 + \{i\}$.

Step 5: Termination:

Set $P_{t+1} = P_3$ and $t = t + 1$.

If $t \geq T$ or the number of non-dominating $> N_{max}$, then set $O = f(m(P_{t+1}))$.

Otherwise, go to step 2.

2.2 SHORT SURVEY OF META-HEURISTIC ALGORITHMS IN MULTIOBJECTIVE OPTIMIZATION

Many comparative studies have been performed in metaheuristic methods applied to system optimization to identify the best solution technique. Dorn et al. (1996) examined four heuristic methods in the context of the schedule optimization problem: iterative deepening, random search, TS, and genetic algorithms. They developed a methodology to satisfy the given controversial constraints gradually, allowing them to reach results in a timely manner. Their work was developed through the attempted building of reusable scheduling software. Their results showed that, for the studied application, iterative deepening and TS are better suited than random search and genetic algorithms.

Sexton (1999) conducted a comparative study of genetic algorithms and simulated annealing to optimize neural networks. The application of artificial neural networks (ANNs) has become increasingly popular due to its ability to approximate unknown functions. The author examined test problems and two real-world problems. The ANN was held constant at six hidden nodes for the GA and SA for the purpose of comparison only. Sexton concluded that the GA appeared able to systematically obtain better solutions, regarding optimizing neural networks, than SA could obtain. Youssef (2001) explored three general iterative algorithms for combinatorial optimization problems: EAs, SA, and TS. Youssef summarizes the following similarities among the three optimization heuristics:

- They are approximation algorithms and do not guarantee the finding of an optimal solution.
- They are blind and do not know when to stop.
- They have a hill-climbing property, such that they occasionally accept uphill moves.

- They are general and can be easily applied to any combinatorial optimization problem.
- Under certain conditions, those algorithms asymptotically converge to an optimal solution.

Youssef used two measures, the time complexity of the algorithm and the quality of the solution, to perform a comparison in the context of the floor-planning problem. The objective function is a vector function:

$$\boldsymbol{F}(x) = (\boldsymbol{Area}(x), \boldsymbol{Length}(x), \boldsymbol{Delay}(x))$$

where *x* is a particular floor-plan solution.

To solve this problem, Youssef introduced fuzzy logic into the study. Only one linguistic value was defined for each variable (i.e., area, length, delay). The degree of satisfaction was described by membership functions for fuzzy sets of linguistic values: small area, short length, and low delay. Youssef compared GA, SA, and TS based on

- quality of the best solution
- progress of the search from the initial solution to the point at which the stopping criteria are met
- iteration count
- the number of solutions found at successive intervals of the cost function

With respect to these measures, TS ranked first, GA ranked second, and SA ranked third; however, this result has its limitations. As Youssef (2001) mentioned, the intention was to study the behaviors of the three heuristics in solving a hard engineering problem, not to demonstrate the superiority of one algorithm over another in all problem domains. Aytug (2002) conducted a comparative study in the context of facility location allocation optimization problems. The focus

was on those genetic algorithms with nonlinear objective functions. Two types of Gas—GALS and Gas—were coded, and their results were compared to those found using Daskin's heuristics method. Daskin (1980) developed a single-nod substitution heuristics method to maximize expected coverage with a system-wide probability p.

$$\text{Maximum } \sum_{j=1}^{n} h_j(1 - p^{y_j})$$

In examining four approaches to solving large-scale maximum expected coverage location problems, Aytug found that the GA approach generated high-quality solutions within predictable times. Daskin's heuristic approach produced better-quality solutions than the GA, but it was computationally cumbersome. Ruiz (2005) evaluated several heuristics for the flow shop problem. The contribution was to include more recent available heuristics and five meta-heuristics for this specific optimization problem. Ruiz used the standard test of Taillad (1993), which included 120 benchmark instances. The objective of the flow shop optimization problem was to minimize total completion time. To accomplish this, Ruiz evaluated a total of 25 heuristic methods. The stopping criterion for all of the meta-heuristic methods tested was a maximum of 50,000 makespan evaluations. Ruiz performed a design of experiments and an analysis of variance on the results of 15 different optimization algorithms. Ruiz determined the validity of the experiments by carefully checking three hypotheses: normality, homogeneity of variance, and independence of residuals. By allowing 60 seconds for every 1,000 jobs in the problem, Ruiz found that the feature that affected CPU time the most was the number of jobs. An ANOVA analysis indicates whether or not there are statistically significant differences between algorithms. With respect to the meta-heuristics algorithms, the GAs of Reeves (1993) and Stutzle's (1998) ILSs were better than all of the other methods. With respect to the permutation flow shop maximum makespan problem, the GA was better than the TS method.

Arostegui (2006) compared the relative performances of TS, SA, and GA on various types of facility location problems (FLP). Capacitated FLP, multi-period FLP, and multi-commodity FLP, three variations of facility location problems, were examined to compare the performances of TS, SA, and GA. This FLP study was the first to compare TS, GA, and SA using more than one classical facility location problem type. Arostegui discovered that GA was superior in the solution-limited evaluations. TS showed very good performance for all types of FLPs in the time-limited evaluation.

2.2.1 TABU SEARCH

Tabu search (TS) is a metaheuristic technique that guides a local heuristic search procedure to explore the solution space beyond local optimality. TS enhances the performance of local search methods by using memory structures. Once a potential solution has been determined, it is marked as "taboo" ("tabu" being a different spelling of the same word), and the algorithm does not visit that possibility again. This search method is attributed to Glover (1993). Local search procedures often become stuck in poor-scoring areas (i.e., where the system reliability stays flat). To avoid these pitfalls, TS carefully explores the neighborhood of each solution as the search progresses. The solutions admitted to the new neighborhood, $N^*(x)$ are determined through the use of memory structures. Using these memory structures, the search progresses by iteratively moving from the current solution x to an improved solution x' in $N^*(x)$. These memory structures form what is known as the tabu list, a set of rules and banned solutions used to filter what solutions will be admitted to the neighborhood $N^*(x)$ to be explored by the search. In its simplest form, a tabu list is a short-term set of those solutions that have been visited in the recent past (less than n iterations ago, where n is the number of previous solutions to be stored; n is also called the "tabu tenure").

The memory structures used in TS can be divided into three categories:

- Short-term: the list of solutions recently considered. If a potential solution appears on this list, it cannot be revisited until it reaches an expiration point.
- Intermediate-term: a list of rules intended to bias the search toward promising areas of the search space.
- Long-term: a list of rules that promote diversity in the search process (i.e., with regard to resets when the search becomes stuck in a plateau or a suboptimal dead-end).

2.2.2 SIMULATED ANNEALING

Simulated annealing (SA) is another generic probabilistic metaheuristic search method for the global optimization problem in a large search space. The name and inspiration for SA came from metallurgical annealing, a technique involving the heating and controlled cooling of a material to increase the size of its crystals and reduce the number of defects in the material. In an analogy of this physical process, each step of the SA algorithm replaces the current solution with a random "nearby" solution, chosen with a probability that depends on the differences among corresponding function values and on a global parameter T (simulating the cooling temperature), which is gradually decreased during the process.

CHAPTER 3: AIRCRAFT STRUCTURAL RELIABILITY AND MOEA APPLICATIONS

This chapter provides a short survey on ensuring aircraft structural reliability and applications of multiobjective evolutionary algorithms in real-world problems.

3.1 AIRCRAT STRUCTURAL RELIABILITY

Aircraft structural reliability is maintained by following three regulatory requirements: safe-life, fail-safe, and damage tolerance. In the early time of aircraft manufacturing, all civil aircraft had to be designed safe-life. This philosophy requires that no cracks of a certain size occur during the entire design service life. Later, the concepts of fail-safe and damage tolerance were introduced. Being fail-safe requires that the aircraft structure be able to sustain strength under a limit load. Damage tolerance is the concept that frequent human inspections will allow early detection of cracks developed through slow crack growth. In recent years, initial and repeat inspections have been triggered by structural health monitoring systems. The concept of conditional inspection has been gaining ground, leading to reduced inspection costs and increased mission availability.

The typical design philosophy is to satisfy the safe-life requirement at longitudinal joints of wing panels and longitudinal lap joints of fuselage skin, and to meet the fail-safe and damage tolerance requirements for the rest of the primary structure of the airframe. Landing gears are usually designed for safe life. The use of crack growth damage tolerance as a substantiation methodology for airframe has been received increasing attention as a logical and viable improvement in fatigue reliability and structural integrity.

Metal fatigue still plays an important role in the design of modern optimized aircraft. The ability to design accurately against the probability of fatigue failures is crucial. Ensuring simultaneous reliability, high durability, minimal weight, and low cost constitute the main

challenge. A key element of the airframe design and certification process is the full-scale fatigue test (FSFT), which seeks to:

- Representatively exercise the structure to discover where fatigue cracking may occur;
- Determine the economic life and the safe-life of the major structural components under an operationally representative loads spectrum;
- Validate modifications and repairs required for early cracking; and
- Build an engineering database for life-cycle fleet management.

A full-scale fatigue test applies cyclic loading to the entire airframe for two lifetimes. The cyclic loading is designed to simulate the operational loading environment in a compact form due to a limited testing budget. Actuators apply cyclic loadings on the wing, fuselage, and vertical fin. The horizontal stabilizers are usually tested off of the aircraft as a component test. The applied loads are equivalent to aerodynamic and inertia loadings that the aircraft would experience during normal operations. The applied point loads, driven by actuators, are determined by aerodynamic and finite element analyses. Simulating the aerodynamic pressures, inertial loads, and internal pressures using a system of discrete loading rams or jacks presents a significant challenge in the ground testing of a complete airframe**.** The jack loads required to simulate each of the design fatigue conditions were defined and then applied to a vehicle-level finite element model (FEM) of the test article. Missions predefined by the customers were mixed together randomly to simulate a lifetime operational usage. This spectrum load sequence, to be applied on the airframe during full-scale fatigue testing, is generally obtained from the flight data records of previous flights. Considering fatigue loads in various types of missions for which the aircraft is used and a statistical mix of these missions, a representative flight load spectrum block was derived.

Molent (2006) compared full-scale airframe fatigue test results from several aircraft, including F/A-18s, F-16s, and P-3Cs. These military aircraft were designed according to different requirements and different design regulations; any shared data apparent from those FSFT results would be highly valuable. Molent concluded as a result of this review that a relatively simple crack-growth model can adequately represent typical crack growth and that this model can be used to help optimize fatigue design so that an airframe will survive the certification fatigue test requirements.

Molent discovered that crack growth was basically exponential. Lead cracks start to grow early in the service life. Under the assumption that critical cracks commence growth shortly after the aircraft is introduced into service, whereas lead cracks grow exponentially with time, the simple CG model can be expressed as follows:

$$\mathrm{Ln}(a) = \Psi N + \mathrm{Ln}(a_o)$$

Where N is the "fatigue life", Ψ is a parameter that is geometry, material and load spectrum dependent," a" is the initial crack-like size of the discontinuity from which the crack starts at the start of loading. The presence of cracks found in these tests indicates that, despite the best efforts of designers, fatigue cracking is likely to develop in all airframes in service, and an FSFT is required to reveal these to allow analysis of their importance to the life of the airframes. The worst possible fatigue scenario is the presence of the largest crack-like discontinuity in the airframe coincident with the locations of maximum cyclic stresses. Normally, cracks discovered in FSFT occur due to poor fatigue detail designs that were somehow ignored by routine fatigue analysis.

3.2 MOEA REAL-WORLD APPLICATIONS

Aerospace design optimization usually focuses on the effects of structural performance and aerodynamics on the geometry of an airframe component. In the past decade, there have been many attempts to develop compromised solutions incorporating manufacturing cost and fatigue life models into an integrated system in order to balance the conflicting objectives of minimizing weight and manufacturing cost while maintaining structural integrity and fatigue reliability.

Huque [2012] provided a computational fluid dynamics (CFD) and response surface-based multiobjective design optimization method. The Pareto-optimal front of six different 2-D airfoil profiles was presented. The standard least square method was used to generate response surface. The elitist non-dominated sorting genetic algorithm (NSGA-II) was used to determine the Pareto-optimal set based on the response surfaces. This research was carried out in the following steps:

- Identify several airfoil profiles with their geometric coordinates.
- Perform CFD simulations around the airfoils based on *Re* (Reynolds number) and α (angle of attack).
- Determine response surfaces for lift and drag coefficients as a function of *Re* and α.
- Perform the optimization using NSGA-II which used the following input parameters:
    1. Population size: 100
    2. Generations: 250
    3. Crossover probability: 1.0
    4. Distribution parameter (for crossover): 20
    5. Mutation probability: 0.25

6. Distribution parameter (for mutation): 200

Rao (2007) applied multiobjective cost and weight optimization to the initial design of turbine disks. Rao et al. did a case study on a design of high-pressure turbine disk from an aircraft engine. Rao et al. discovered that optimization with cost as the objective function has been rarely used in the past for a number of reasons:

- Lack of detailed information for accurately determining the cost of manufacture at the design stage
- Difficulty modeling manufacturing cost in terms of geometry parameters and design variables
- Minimizing mass or weight considered analogous to minimizing cost

Rao et al. developed an integrated optimization system including computer-aided design (CAD), finite element analysis, cost estimation, and fatigue life prediction models within a MATLAB script environment. The goals were to demonstrate an MDO process driven by cost and structural performance; and to show that there is a definite difference between minimizing weight and minimizing cost in finding the optimal shape of a high-pressure turbine disk in a civil aircraft engine.

The cost calculations for every process are embedded in an object-oriented environment and treated as black-box models reused interactively by the designer. This is a bicriteria optimization problem in which the goals are to minimize both volume and manufacturing cost. Turbine disk-lifting methods fundamentally affect safe component life, material usage, and maintenance procedures; an optimization of the disk profile should account for those methods' fatigue effects. The material's S-N curve was used, and the modified Goodman relationship was

used to incorporate the mean stress effects because fatigue loading is not fully reversible in disk design.

Many MOEA algorithms have matured over the years, with a newly emergent breed of hybrid algorithms combining the benefits of two different types of optimization methodologies to develop higher-performance new MOEAs. For a tailless delta wing aircraft, unfavorable aeroelastic responses can be suppressed with the help of active control. Maneuver load alleviation (MLA) is the concept of redistribution of forces and moments on airframes through the optimal actuation of control surfaces.

Suresh (2013) presented a hybrid optimization algorithm combining the heuristic-based NSGA-II with calculus-based, goal-programming methods for minimizing the tailless delta wing root bending and twisting movements. The constraints were stability equations and limitations on actuator hinge movements. The primary objective of Suresh's paper was to present a hybrid method to promote superior spread and faster convergence than NSGA-II, with an accurate Pareto front. The optimization was carried out in two phases. In the first phase, which entailed an initial 50 generations, the problem was solved by the NSGA-II method; the second phase used the goal-programming method. The result from the first Pareto-ranked solution at the 50th generation was captured with the goal of identifying the ideal objective vector, and the pseudo weights were assigned according to the position of the population on the Pareto front. For this type of MLA problem, the hybrid method performed much better in terms of spread, accuracy, and speed of convergence than did the conventional NSGA-II and goal-programming methods. The hybrid method led to a maximum 10% savings in computational time.

The next chapter presents an innovative hybrid method combining a metaheuristic tabu search with evolutionary genetic algorithms. After the introduction of TSEA, this hybrid MOEA

algorithm is applied to aircraft structural reliability problems to maximize the aircraft horizontal stabilizer reliability and minimize the retrofitting cost during fatigue testing. Further chapters also compare TSEA with other metaheuristic algorithms for real-world and other applications. TSEA applies the concept of tabu search and uses tabu list memory aid to guide the genetic algorithm to search the Pareto-optimal front. The tabu list serves as a moving boundary to accelerate the search toward the Pareto-optimal front with good spread.

CHAPTER 4: TABU SEARCH EVOLUTIONARY ALGORITHM (TSEA)

Inspired by the efficiency of the metaheuristic tabu search and many state-of-the-art MOEAs, an innovative, hybrid tabu search evolutionary algorithm (TSEA) is provided in this dissertation. TSEA combines the advantages of tabu search memory structure and non-dominated sorting genetic algorithms to improve the performance of multiobjective system optimizations. This new algorithm provides an effective methodology for tackling real-world multiobjective system optimization problems that are nonconvex, wherein the objective functions cannot even be explicitly written. The effectiveness of the innovative algorithm TSEA is demonstrated in its application to commonly accepted testing problems and to large, complex real-world applications. The results obtained from both sets of applications are compared with state-of-the-art MOEAs in chapters 5, 6, and 7.

4.1 MATHEMATICAL MODEL

The proposed hybrid TS and evolutionary algorithm TSEA focuses on solving multiobjective optimization problems (MOP) as shown below:

MOP1:

Max $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (\boldsymbol{f_1}(\boldsymbol{x}), \boldsymbol{f_2}(\boldsymbol{x}), \dots, \boldsymbol{f_k}(\boldsymbol{x}))$

subject to $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{g_1}(\boldsymbol{x}), \boldsymbol{g_2}(\boldsymbol{x}), \dots, \boldsymbol{g_m}(\boldsymbol{x})) \leq 0$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_n) \in \boldsymbol{X}$

$\boldsymbol{y} = (y_1, y_2, \dots, y_k) \in \boldsymbol{Y}$

x is the decision vector, **X** is the decision space,

y is the objective vector, and **Y** is the objective space.

4.2 FITNESS ASSIGNMENT AND FITNESS SHARING

The fitness of the population is based on its Pareto-optimal status. The purpose of the fitness function is to guide the search algorithm toward the Pareto front. Therefore, it is crucial to choose the most effective fitness function for multiobjective optimization problems. This algorithm is built on the NSGA and involves finding the Pareto fronts one layer at a time. The fitness is assigned by peeling off each layer of non-dominated individuals as their rankings. Non-dominated solutions make up the first layer, as shown in figure 4.1, and are selected among all current populations. Then solutions of layer 1 are removed from the pool for selecting non-dominated solution layer 2, and so on. At the end, the number of individuals in the pool is less than the predefined tabu list size and all members in the pool (last layer) go onto the tabu list, as shown in figure 4.1. All individuals in a given layer share the same fitness.

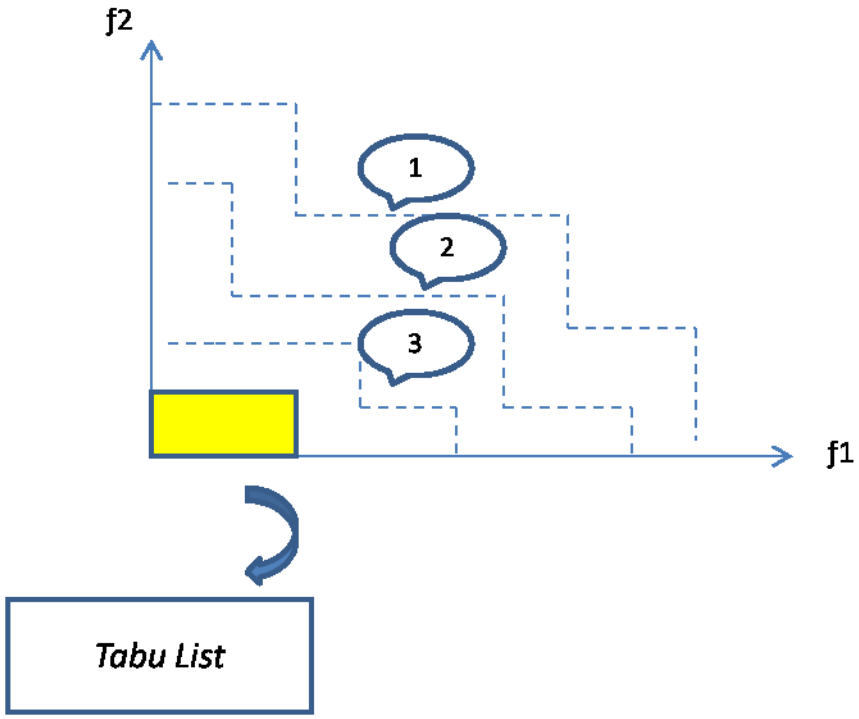

Figure 4.1 Formation of tabu list

4.3 TABU DISTANCE TO REINE FITNESS

The fitness values defined by non-dominating ranks may not be sufficient for the later binary tournament selection process, especially when many individuals are dominated by the same set of non-dominated individuals. The selection for the next generation is performed through crossover and mutation. When selecting the parents, a binary tournament selection with replacement is based on the fitness of the two selected individuals. If they have the same fitness, a tabu distance (TD) is introduced to separate them. TD is determined as the average distance an individual with respect to the entire tabu population, as shown in figure 4.2. This tabu distance is added to the fitness to promote diversity in the solutions. When selecting a new parent, the one with larger tabu distance (i.e., farther away from the tabu list (tabu population) is always selected to increase the spread in the solutions.

$$TD(i) = \frac{\sum_{j=1}^{N} d_{ij}}{N}$$

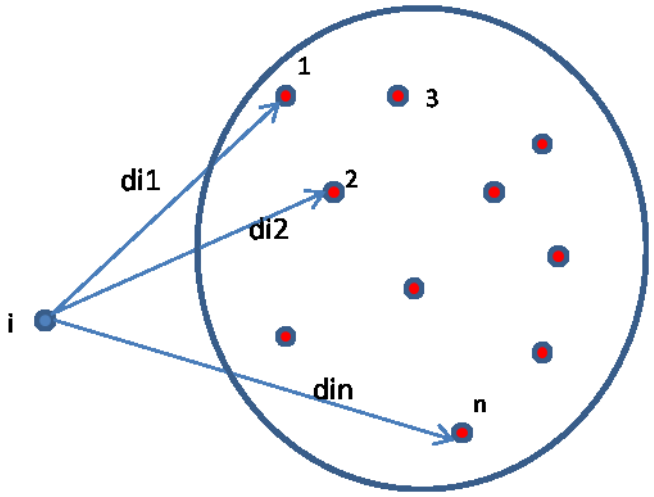

Figure 4.2 Tabu distance

4.4 TABU SEARCH TO GUIDE EVOLUTIONARY SEARCH ALGORITHMS

The purpose of this research is to provide an innovative evolutionary algorithm guided by tabu search to explore new ways for multiobjective GA optimization to converge to the Pareto-optimal front effectively. Following is a detailed introduction of TSEA.

Predetermine the following parameters:

N --- population size

T_{max} --- maximum of number of iterations

Build the initial population P_0, which represents the entire decision space. Build an empty tabu list.

Randomly select a subset of P_0 as the prepopulation P_p of size N and check with the tabu list (if it is not empty).

While iter < Tmax

Step 1: Find all feasible solutions from previous population P_p:

While the population is currently not empty, do:

Find all non-dominated solutions, and assign the fitness of $i = i + 1$ (last batch has the fitness of k).

Remove all non-dominated solutions from the current population.

Step 2: Move the last group of individuals to the tabu list.

Step 3: Build the current population P_t:

Select all individuals with fitness values of 1, . . . (k–1).

For the rest of the slots of the current population:

- Fill with randomly selected feasible solutions from the individual space
- For the entire new population, calculate the tabu distance, which is defined as the average distance from an individual to all individuals in the tabu set.

Step 4: Crossover:

- Select two parents from current population using the binary tournament approach. The parent that has lower fitness (fitness is to be minimized) prevails.
- When two individuals have the same fitness, selection is based on the tabu distance.
- Perform a crossover operation on the current population and accept the offspring based upon the crossover probability p_c.
- If the crossover is given a go-ahead, two parents are selected by the binary tournament approach. If the two individuals have the same fitness value, then the one with the larger tabu distance is selected.
- All selected individuals in the new generation are then checked against the tabu list to make sure none of them are on the tabu list.

Step 5: Mutation:

- Mutate the current population with a mutation rate of p_m.
- Check with the tabu list before accepting any new individuals.

Step 6: Termination:

If iteration $t < T_{max}$, go to step 1.

Otherwise, stop.

End while.

Choose the feasible and non-dominated solutions from the current population as the final solutions.

The uniqueness of TSEA lies in the combination of a non-dominated fitness assignment and a constantly moving tabu front formed from a tabu list (with long-term memory) to guide and accelerate the search for the Pareto-optimal front without trapping in local optima. This new

metaheuristic algorithm offers superior performance when applied to bicriteria real-world applications, as shown in sections 5 and 7. It converges faster than such state-of-the-art algorithms as SPEA2 and NSGA II and provides more evenly spread-out Pareto-optimal solutions than the other algorithms. TSEA is also applied to typical convex (problem number 1) and nonconvex (problem number 2) testing problems as developed by Zitzler (2000) to demonstrate its wide spectrum of effectiveness. The Pareto-optimal solutions are also compared with those of state-of-the-art algorithms such as SPEA2 and NSGA II.

4.5 CONSTRAINT HANDLING

Unlike the SPEA 2, which used a repair method, or the NSGA II, which used a constraint-dominance, TSEA is designed for use in an interactive mode with the DM such that selected constraints can be relaxed based upon the DM's preference. After obtaining non-dominated solutions near the Pareto-optimal front, a compromise programming ranking method is used, and those infeasible non-dominated solutions are deleted at this time. All the rest of the constraints are strictly enforced at every iteration.

CHAPTER 5: APPLICATION OF TSEA IN AIRCRAFT TAIL STRUCTURE RELIABILITY OPTIMIZATION

Before a new type of military aircraft can go into production, full-scale fatigue tests are usually conducted to ensure the aircraft has sufficient fatigue life and structural integrity. The purpose of the fatigue test is to prove that the aircraft can sustain the entire test without cracking. If test loading conditions are too severe, an optimized retrofitting plan is needed to reallocate limited resources to ensure the completion of the test.

A decision support system (DSS) is needed to provide the flexibility and insight to develop the retrofitting plans. The underlying database of fatigue damage rates is predetermined by a reliability model. An innovative multiobjective optimization algorithm combining metaheuristic tabu search with an evolutionary algorithm embedded in the DSS, TSEA helps DMs select the best plan to execute prior to the airplane fatigue test; this way, aircraft can sustain the fatigue test without needing to stop for repairs.

Potential fatigue-critical locations will be at the rear spar of each bay. Doublers or stiffeners can be installed on the rear spar web to increase the strength and stiffness of the horizontal tail structure. For real-world system reliability optimization problems, it is difficult and sometimes unnecessary to obtain exact solutions. Rather, close-to-optimum solutions are considered acceptable, especially under conflicting objectives such as cost. Figure 5.1 shows a typical horizontal aircraft stabilizer.

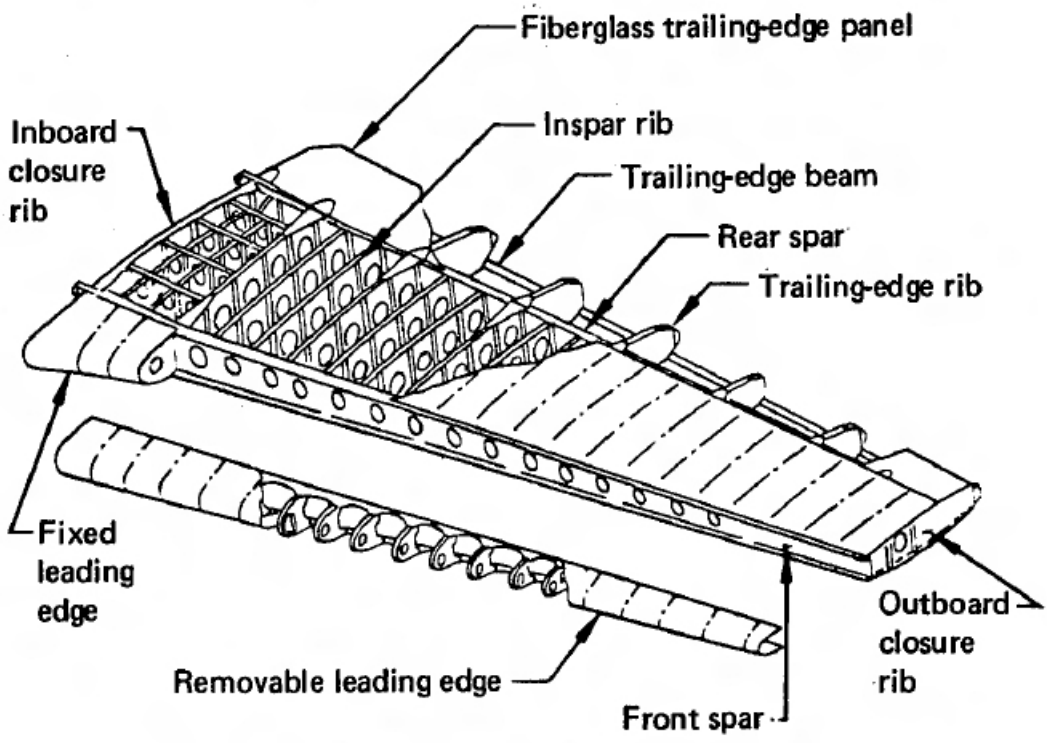

Figure 5.1 Aircraft horizontal stabilizer

5.1 NEED FOR CONTINUED RESEARCH

The few difficulties related to GA multiobjective optimization recognized by Deb (1999) included difficulties converging to a Pareto-optimal front and maintaining diversity in Pareto-optimal fronts. Deb then proposed a simple construction methodology for building the tests for objective functions from single-objective optimization problems. This allows a multiobjective GA to be tested in a controlled manner on various aspects of problem difficulties. Deb suggested the following areas for future research in developing better multiobjective GAs:

- comparisons of existing multiobjective GA implementations
- knowledge of the dynamics of GA populations with generations
- scalability of multiobjective GAs with several objectives
- development of constrained test problems for multiobjective optimization

- convergence to Pareto-optimal fronts
- definitions of appropriate multiobjective GA parameters (such as elitism)
- comparisons of two populations
- hybrid multiobjective GAs
- real-world applications
- multiobjective scheduling and other optimization problems

Janssens (2008) pointed to various future challenges in the area of evolutionary meta-heuristics designed to solve multiobjective problems:

- how to efficiently generate the small set of non-dominant solutions that DMs need
- how to properly approximate when the number of points on the Pareto curve becomes exponentially large
- the need for further exploration of the performance and complexity of some heuristics that have been proposed to construct an approximation of the curve
- the need to develop standards for evaluation and conduct more theoretical work regarding quality evaluation
- an assessment of the computational complexity of maintaining the "archive" of all non-dominated solutions obtained during the search process

5.2 PURPOSE OF THIS REAL-WORLD APPLICATION

Guliashki (2009) conducted a survey of evolutionary algorithms used in multiobjective optimization problems and concluded, "At the next stage, the EMOO researchers concentrated on developing better and computationally faster algorithms by means of scalable test problems and adequate performance metrics to evaluate EMOO algorithms. One of the major aspects of

scientific research is the efficiency, which is regarded at algorithmic level and at a data structure level."

From what has been reviewed in the literature so far, multiobjective reliability optimization based on real engineering problems has mostly been explored in the context of solving typical problems such as floor planning, schedule optimization, "traveling salesman," redundancy allocations for integrated circuit boards, and so on. Because these real-world problems are usually large and complex, metaheuristic methods such as GAs, SAs, and TSs are often used. Many comparative studies have been performed on these metaheuristic methods, however, using simple benchmark problems. These metaheuristics are valuable in their application to solving complex nonlinear multiobjective optimization problems where they can help DMs decide how best to allocate their limited resources and achieve the most economical results.

Of the three meta-heuristic methods, the evolutionary multiobjective optimization (EMOO) approach is a popular and useful field of research. Although the GA method is used in many real-world problems, it still lacks application in the aircraft industry logistics area, where aircraft structural reliability is still facing many optimization challenges. This is due to the substantial problem size and the complexity of its objective functions, which normally—because they are highly nonlinear—cannot be expressed explicitly. One challenge in applying EMOO is determining a small set of Pareto-optimal solutions from which DMs can choose. The number of solutions is exponential for discrete problems, and it is often difficult to construct the full Pareto curve; therefore, approximation is applied.

The focus of this real-world application is determined based on the literature review and the challenges pointed out by the three papers mentioned above. This study will address the ninth

of Deb's proposed future research areas: that is, it seeks to apply GA to a real-world problem. It also addresses the fifth area by exploring new ways for multiobjective GA optimization to converge to Pareto-optimal fronts. This study will try to answer Janssens's (2008) first challenge by exploring ways to reach a small set of Pareto solutions quickly using GA.

The purpose of this research is to address the research gaps with regard to using EMOO to optimize the reliability of aircraft horizontal tail structures. To meet the challenges noted above, this research will attempt to develop an efficient EMOO algorithm to provide DMs with a small set of non-dominant solutions as the output of a DSS. This EMOO algorithm will be a nonelitist algorithm because elitist EMOOs tend to be less efficient.

A new MOEA method that combines the benefit of tabu search with genetic algorithms called TSEA is developed in this dissertation research to achieve the same set of solutions. TSEA will contribute to the state of the art by providing an efficient methodology for achieving multiobjective aircraft design optimization in DSS. This research effort is well in line with Guliashki's (2009) prediction regarding the directions of new findings.

5.3 RELIABILITY AND COST MODELS

Military aircraft must sometimes carry out missions that put the aircraft in stall buffet conditions. A stall buffet condition occurs when the aircraft is flying at a low level with a high angle of attack. The air flow passing over the wings starts to separate, causing a turbulent flow downstream, this then encroaches on the horizontal stabilizers. This turbulent flow excites vibration along the horizontal stabilizers, which has a negative impact on their fatigue life (i.e., their structural reliability). A typical aircraft horizontal stabilizer is shown in figure 5.1. Because the rear spar carries most of the load, fatigue-critical locations usually are located at selected bays of the rear spar. The goal of the reliability optimization is to maximize the minimum fatigue life at fatigue-critical locations.

This is a bi-objective problem where the objectives are to (1) maximize reliability and (2) minimize cost. The objective for optimizing the reliability of the aircraft horizontal stabilizer is defined as follows:

$$R_s = 1 - \max(FL_1, FL_2, \ldots, FL_n) \qquad (1)$$

Here, the reliability of the aircraft horizontal stabilizer corresponds to the reliability of the bay with the worst fatigue prone detail design. An example of fatigue detail is a rivet hole where, if continued cyclic loading (tension, then compression, then tension) exists for a long enough period, a crack might occur. Other examples, such as the corner of a window cutout, are also sensitive to fatigue failure and are therefore labeled as fatigue details. The concept of the "weakest link" is used in this definition, where when the worst fatigue detail uses up its fatigue life ($FL_i = 1$), the reliability of the horizontal stabilizer goes to zero.

The objective for minimizing cost of retrofitting the horizontal stabilizer is defined as follows:

$$C_s = \Sigma Scost_i + \Sigma Dcost_i \qquad (2)$$

Where

n --- the number of bays that are sensitive to stall buffet

FL_i --- fatigue life used up at the critical location of bay, which is the ratio of fatigue damage accumulated so far over the allowed lifetime fatigue damage. FLi is unit less ratio and its maximum value is 1.

$Scost_i$ --- cost of installed stiffener at bay i, which is proportional to the weight of the stiffener

$Dcost_i$ --- cost of installed doublers at bay i, which is proportional to the weight of the doublers

Subject to:

$FL_i <= 1;$ $\quad i=1,\ldots, n$ $\quad (1)$

$x_{di} = 0$ or $1;$ $\quad i= 1, \ldots, n$ $\quad$ (bays that installs doublers when $x_{di} = 1$)

$x_{sj} = 0$ or $1;$ $\quad j=1, \ldots, n$ $\quad$ (bays that installs stiffeners when $x_{si} = 1$)

except when $i = j$, $x_{di} = x_{sj} = 1$ $\quad (2)$

Each of the 14 bays on the horizontal stabilizer can be installed with doublers or a stiffener, but not both, or nothing. This constraint was relaxed to allow each bay to install both doublers and stiffeners based on the DM's opinion. When ranking the Pareto-optimal solutions for DMs, this relaxation was penalized by assigning a large weight, as shown in table 5.9. A binary coding scheme was used in this application. (Each chromosome had a length of 28 binary bits.) The first 14 bits represented the installation of doublers in each bay, with the last 14 representing the installation of stiffeners for each bay. The initial pool of all possible combinations of 28 bit-long chromosomes was generated, and each generation was selected from this pool randomly. The following naming conventions were used throughout this application: where $FLi = 0$ represents full life remaining and $FL_i = 1$ represents no life remaining since a detectable crack has initiated.

FL_i --- FL_{io} or FL_{id} or FL_{is}

FL_{io} --- fatigue life used up at the critical location of bay *i* without doublers or stiffener

FL_{id} --- fatigue life used up at the critical location of bay *i* with doublers installed

FL_{is} --- fatigue life used up at the critical location of bay *i* with stiffener installed

The fatigue life used up at each bay of the horizontal stabilizer is calculated by the following equation:

$$FL_{io} = (Dmg_{si} \times N_{si} + Dmg_{mi} \times N_{mi} + Dmg_{bi} \times N_{bi})/DLT_i \qquad (3)$$

where

Dmg_{si} --- fatigue damage from a severe flight that experienced a deterrent level of stall buffet for 8 seconds, where fatigue damage is relative damage (i.e., the fraction of actual fatigue damage over lifetime total allowable damage).

DLTi --- life time fatigue damage allowable at bay i

When FL_{io} reaches 1, detectable cracks can be found in the structure. The horizontal stabilizer will still be able to carry loads until cracks are found during planned inspections and repaired.

N_{si} --- number of severe flights in a lifetime

Dmg_{mi} --- fatigue damage from a medium severe flight that experienced a deterrent level of stall buffet for 4 seconds

N_{mi} --- number of medium severe flights in a lifetime

Dmg_{bi} --- fatigue damage from a benign flight that experienced a stall buffet for 0.5 seconds

N_{bi} --- number of benign flights in a lifetime

Typical fatigue analysis locations at bay 20 are also shown in figure 5.2. For the purpose of this study, only fatigue-critical location 2 will be examined. In its lifetime, a military aircraft may experience stall buffeting in some of its flights that is severe enough to reach a deterrent level. A deterrent level of buffeting is defined as a severe level of buffeting that constitutes a clear deterrent to any further decrease in airspeed or increase in angle of attack. Table 5.1 summarizes such flights.

Table 5.1 Stall Buffet Mission Summary

Mission Type	Mission name	Duration of buffet per mission (sec.)
Benign	Type I	0.4
Medium	Type II	0.7
Severe	Type III	4.7

To obtain the stress time history at location 2 for an entire flight, a sequence of flight events called a fatigue template is predetermined. Load conditions determined according to this fatigue template are applied to the airframe FEM. Loads to stress transfer functions were applied to obtain the stress time histories. This stress time history was rain flow counted to find all stress reversals. Endo & Matsuishi (1968) developed the Rainflow Counting method by relating stress reversal cycles to streams of rainwater flowing down a Pagoda. Rainflow is an effective way separating small, uninteresting oscillations from the larger ones, without affecting turning points by the smoothing effect of a filter or interrupting a large range before it is actually completed. The rainflow method allows the application of Miner's rule in order to assess the fatigue life of a structure subject to complex loading.

The time history of a typical load, stress, or strain as shown in figure 5.2 is usually variable amplitude. With the load, stress, or strain time history plotted such that the time axis is vertically downward as shown in figure 5.2, the lines going horizontally from a reversal to a succeeding range can be considered as rain flowing down a roof represented by the history of peaks and valleys. The operation of the rainflow method is shown in figure 5.3 for a history consisting of four peaks and four valleys.

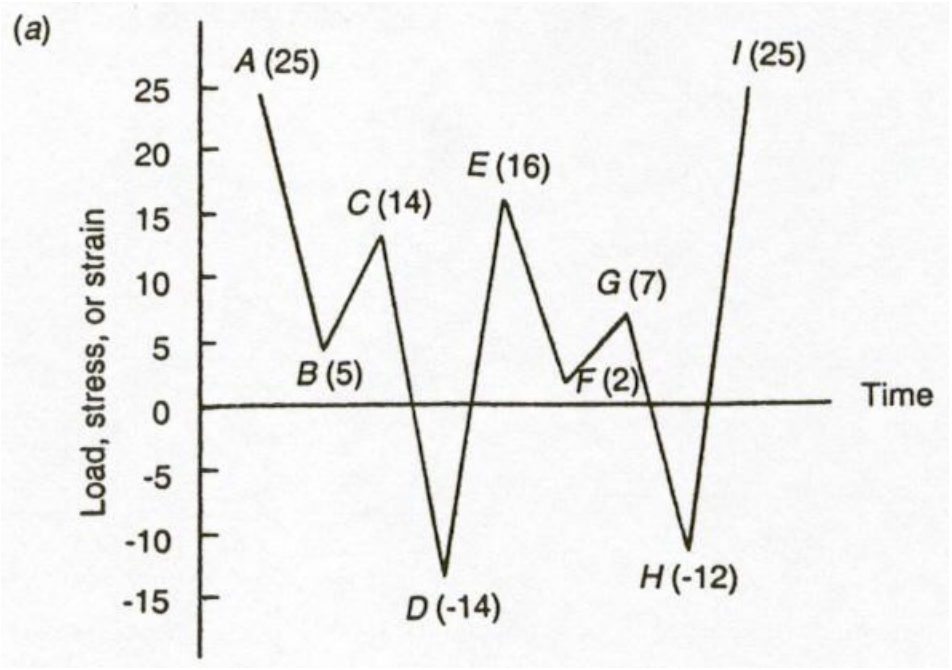

Figure 5.2 Typical load, stress or strain time history

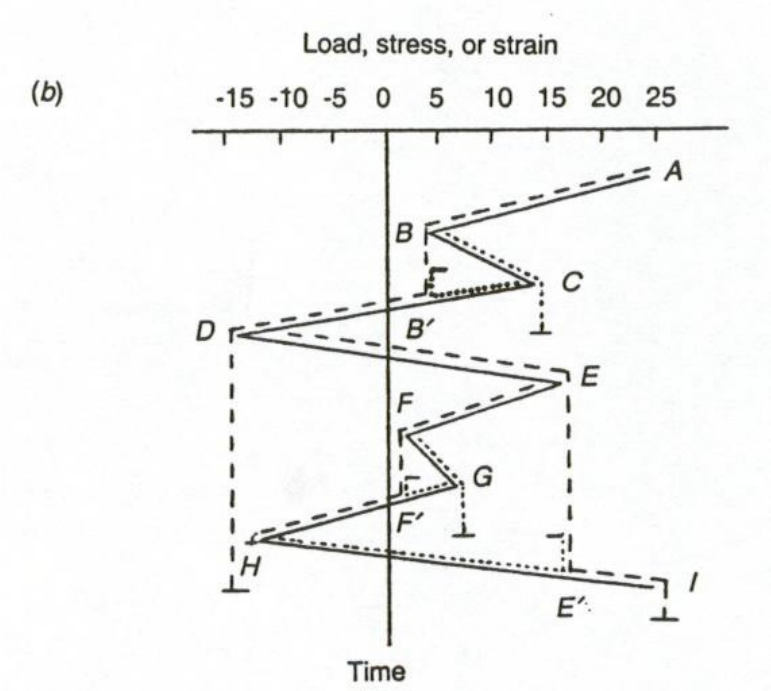

Figure 5.3 Rain flow counting method

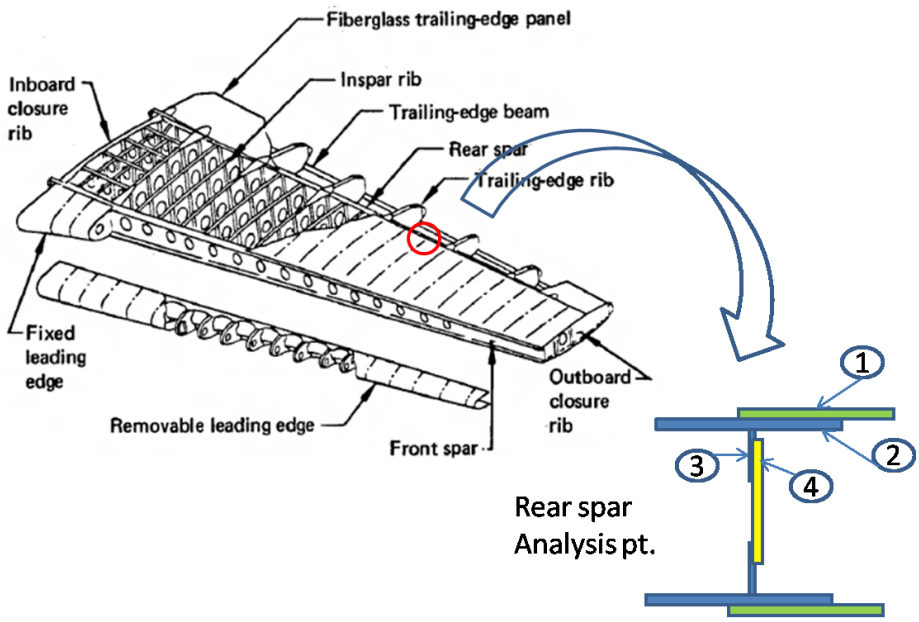

Figure 5.4 Fatigue analysis locations at aircraft horizontal stabilizer

The fatigue damage from each stress reversal cycle is computed according to a test-verified S-N curve. The Palmgren-Miner linear cumulative fatigue damage theory is known as the Miner's rule. According to Miner's rule, failure occurs when:

$$\frac{n_1}{N_1} + \frac{n_2}{N_2} + \ldots + \frac{n_i}{N_i} = 1$$

where

n_i = number of cycles at the *i*th stress level

N_i = number of cycles to failure corresponding to the *i*th stress level

n_i/N_i = damage ratio at the *i*th stress level

The fatigue damage from all cycles of a flight is accumulated according to the Miner's Rule:

$$Dmg = \sum_{i-1}^{p} \frac{ni}{Ni}$$

where

Dmg is fatigue damage, or the ratio of life used up so far (when *Dmg* = 0, a crack is assumed to have initiated).

n_i is the number of cycles of the ith stress applied

N_i is the expected lifetime in cycles at the ith stress level

P is the number of discrete stress levels

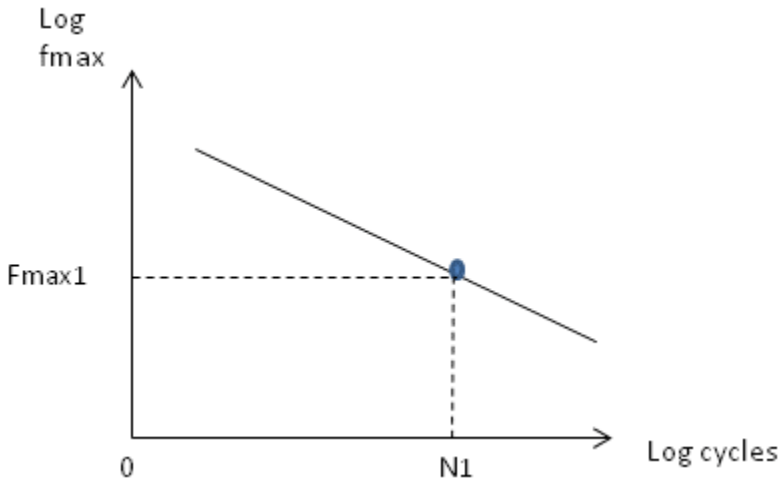

Figure 5.5 Typical aluminum S-N curves

For all fatigue-critical locations on the horizontal stabilizer, the fatigue damage for three types of missions is determined using Miner's rule, which is shown in table 2. Because the stall buffeting mostly affects the outboard portion of the horizontal stabilizer, only bays 15 through 30 were examined. The damage rates shown in table 5.2 are listed by unit of fatigue damage per 100,000 flights. The fatigue damages allowed over the course of a lifetime, based on the buildup at each fatigue-critical location, are listed in table 5.2. Figure 5.4 shows the maximum stresses at those two bays with and without doublers. The magnification factors applied to the stresses with and without the doublers can be calculated by the following equation:

$$Mac\_d = (.705 + .713) / 2 = .72$$

Design loads for the three types of missions—I, II, and III, with varied stall buffet severity—were applied on the horizontal stabilizer. Stresses at fatigue-critical location 2 shown in figure

5.2 were obtained from static analysis. Fatigue damages from each mission were calculated using Miner's rule.

Fatigue damage rates for the three types of missions without any retrofitting were obtained by dividing the mission fatigue damage with mission length, as shown in table 5.3. Since the stall buffet mostly affects the outboard portion of the horizontal stabilizer, only bays 16 through 29 are examined. The last column in table 5.2 shows the allowable lifetime damage for each bay. The damage rates shown in table 5.3 are for retrofitting with doublers.

These fatigue damage rates in tables 5.2, 5.4, and 5.5 are given in units of damage per 100,000 flights. Notice that the damage rates are nonlinear with respect to locations and installation of doublers or stiffeners, which demonstrates the extent of the complexity of the problem. This DSS can be applied to many similar situations due to the generality of this modeling system.

Table 5.2 Damage Rates at Fatigue-Critical Locations

ID	Mission I	Mission II	Mission III	LT Dmg Allowable
UPCHRD15	1.048	14.796	99.133	77483
UPCHRD16	1.348	19.031	127.508	95868
UPCHRD17	1.260	17.782	119.139	89696
UPCHRD18	1.266	17.872	119.744	95219
UPCHRD19	1.767	24.950	167.165	96519
UPCHRD20	4.697	66.307	444.254	99832
UPCHRD21	6.183	87.295	584.875	100000
UPCHRD22	6.128	86.514	579.643	100000
UPCHRD23	7.199	101.638	680.975	100000
UPCHRD24	4.933	69.644	466.613	100000
UPCHRD25	4.137	58.403	391.302	97504
UPCHRD26	2.545	35.924	240.689	97339
UPCHRD27	1.225	17.292	115.853	77620
UPCHRD28	0.526	7.429	49.773	76801
UPCHRD29	1.368	19.318	129.428	96356
UPCHRD30	0.427	6.031	40.406	74915

1) All fatigue damage rates are multiplied by 100000 to avoid computation on small numbers. The unit of fatigue damage rates at any bay is fatigue damage per flight
2) Life time fatigue damage allowable, when total fatigue damage reaches this level at any bay, crack is initiated.

3) Each row provides fatigue damage rates at the upper chord of each bay for three types of missions.

Table 5.3 Maximum Stresses Comparison

Bay	Elem ID	Retrofit	fmax with retrofit	fmax without retrofit	factor
Upr Bay 24	3253271	stiffener	9.961	11.139	0.894
Upr Bay 25	3262271	stiffener	7.947	9.66	0.823
Upr Bay 26	3271271	doubler	5.467	7.756	0.705
Upr Bay 27	3280291	stiffener	3.181	5.517	0.577
Upr Bay 28	3291251	doubler	2.698	3.784	0.713

This factor is applied to stresses at bays 15 through 30 to rerun the fatigue damages per flight using Miner's rule. The results from this analysis are shown in table 5.4. To optimize the reliability of the horizontal stabilizers (i.e., to maximize their fatigue life), a number of doublers or stiffeners can be installed at selected bays to reduce stress levels at fatigue-critical areas. Cost is associated with the materials and manufacturing of the doublers or stiffeners as well as their design and engineering. Therefore, when the system reliability increases with the installation of doublers or stiffeners, the system cost also increases. Because the dimension of each bay varies, the cost associated with the installation of doublers or stiffeners also varies. Typical doublers are shown in figure 5.6, while typical stiffeners are shown in figure 5.7.

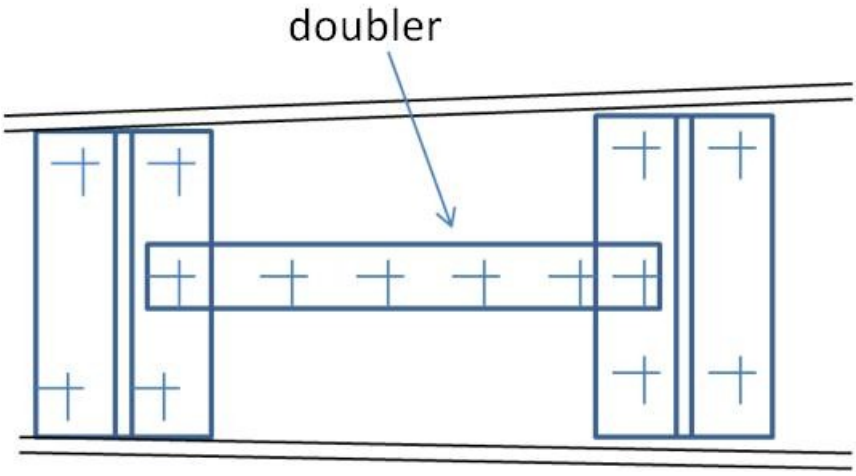

Figure 5.6 Typical doublers in bay 28

Table 5.4 Fatigue Damage Rate with Doublers

ID	Mission I	Mission II	Mission III
UPCHRD15	2.053	28.978	194.151
UPCHRD16	2.172	30.670	205.491
UPCHRD17	1.771	24.999	167.494
UPCHRD18	1.789	25.253	169.196
UPCHRD19	1.729	24.414	163.573
UPCHRD20	2.187	30.869	206.819
UPCHRD21	2.197	31.018	207.820
UPCHRD22	2.201	31.068	208.154
UPCHRD23	2.218	31.318	209.832
UPCHRD24	2.222	31.368	210.169
UPCHRD25	2.272	32.079	214.929
UPCHRD26	2.283	32.233	215.960
UPCHRD27	2.361	33.326	223.284
UPCHRD28	2.312	32.646	218.727
UPCHRD29	1.786	25.211	168.912
UPCHRD30	2.349	33.168	222.226

1) All fatigue damage rates are multiplied by 100000 to avoid computation on tiny numbers. The unit of fatigue damage rates at any bay is fatigue damage per flight
2) Life time fatigue damage allowable, when total fatigue damage reaches this level at any bay, crack is initiated.

3) Each row provides fatigue damage rates at the upper chord of each bay for three types of missions.

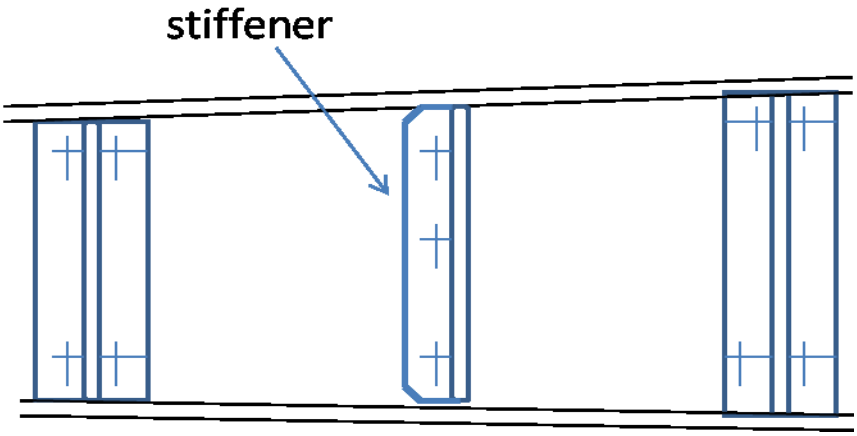

Figure 5.7 Typical stiffeners in bay 27

The magnification factors applied to the stresses with and without the stiffeners are shown in figure 5.5. Because there are no data to calculate the factors for bays before bay 24 or after bay 28, the factors at bay 24 are used for lower-numbered bays, and the factors for bay 28 are used for higher-numbered bays.

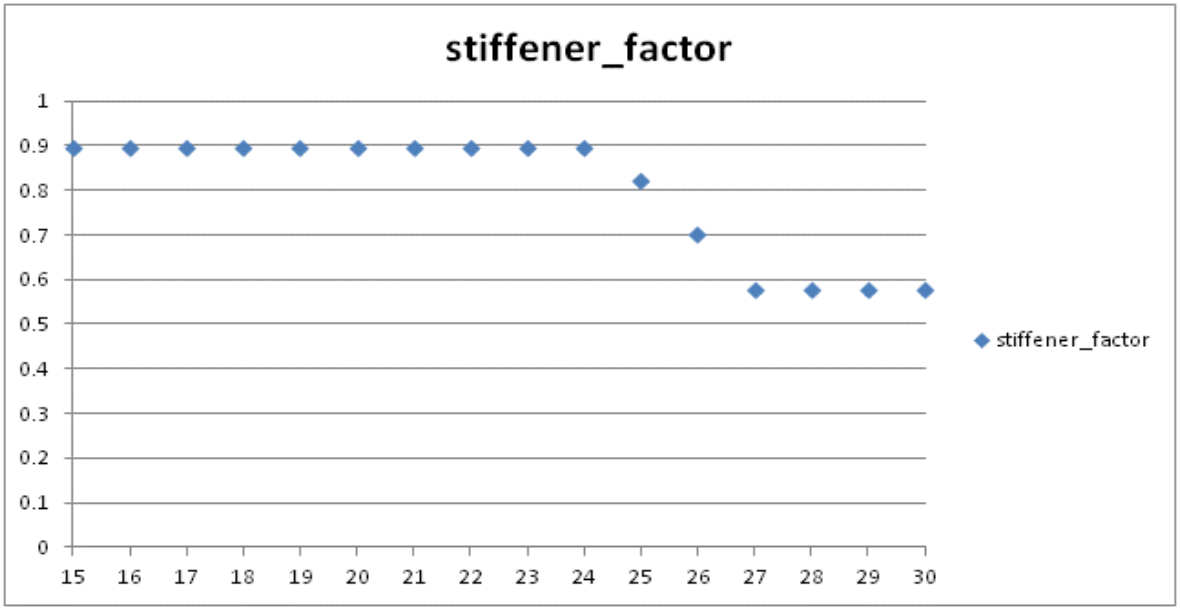

Figure 5.8 Factors on maximum stresses with stiffeners

These factors were applied to stresses at bays 15 through 30, and the fatigue damage per flight were recalculated using the same analysis tool. The results of this analysis are shown in table 5.5.

Table 5.5 Fatigue Damage Rate with Stiffeners

ID	Mission I	Mission II	Mission III
UPCHRD15	4.754	67.114	449.664
UPCHRD16	5.042	71.185	476.939
UPCHRD17	4.079	57.590	385.856
UPCHRD18	4.122	58.197	389.917
UPCHRD19	3.981	56.195	376.509
UPCHRD20	5.076	71.662	480.138
UPCHRD21	5.102	72.022	482.548
UPCHRD22	5.110	72.142	483.354
UPCHRD23	5.153	72.746	487.398
UPCHRD24	5.161	72.867	488.211
UPCHRD25	3.812	53.816	360.568
UPCHRD26	2.050	28.944	193.922
UPCHRD27	1.024	14.449	96.811
UPCHRD28	1.003	14.163	94.894
UPCHRD29	0.780	11.018	73.822
UPCHRD30	1.019	14.383	96.366

1) All fatigue damage rates are multiplied by 100000 to avoid computation on small numbers. The unit of fatigue damage rates at any bay is fatigue damage per flight.
2) Life time fatigue damage allowable, when total fatigue damage reaches this level at any bay, crack is initiated.
3) Each row provides fatigue damage rates at the upper chord of each bay for three types of missions.

5.4 DSS SYSTEM DEFINITION

The DSS contains three components: a graphical user interface (GUI), a database, and several modules to provide the optimized retrofitting solutions. The GUI receives input from the

user (i.e., the DM), who provides information such as the number of stall buffet flights and their durations. After the user pushes the "Run" button, the DSS will run for about several mimutes, at the end of which it will provide several non-dominant solutions and their rankings. The solutions include suggestions for where to install doublers and stiffeners. Figure 5.7 shows the GUI. There are three types of flights and sorties, as shown in table 5.6, from which the user can choose.

Table 5.6 Flight and Sortie Types

Flight Type
initial buffet
moderate buffet
severe buffet
Sortie Type
short
medium
long

The user must press the "OK" button after inputting the number of flights and their duration times. The DSS will prompt the user to provide more input until all combinations of flight type and sortie type have been visited.

A message of "All Done!" will appear after the DSS has run through all of its modules and found the desired results. The user can hit the "Exit" button when done using the DSS. The DSS provides its users with a number of fatigue-critical locations where doublers or stiffeners can be installed to ensure that the fatigue life is maximized for all locations on the horizontal stabilizer studied. By performing these modifications, the user should be able to build a horizontal stabilizer that will survive the fatigue test without premature fatigue failures.

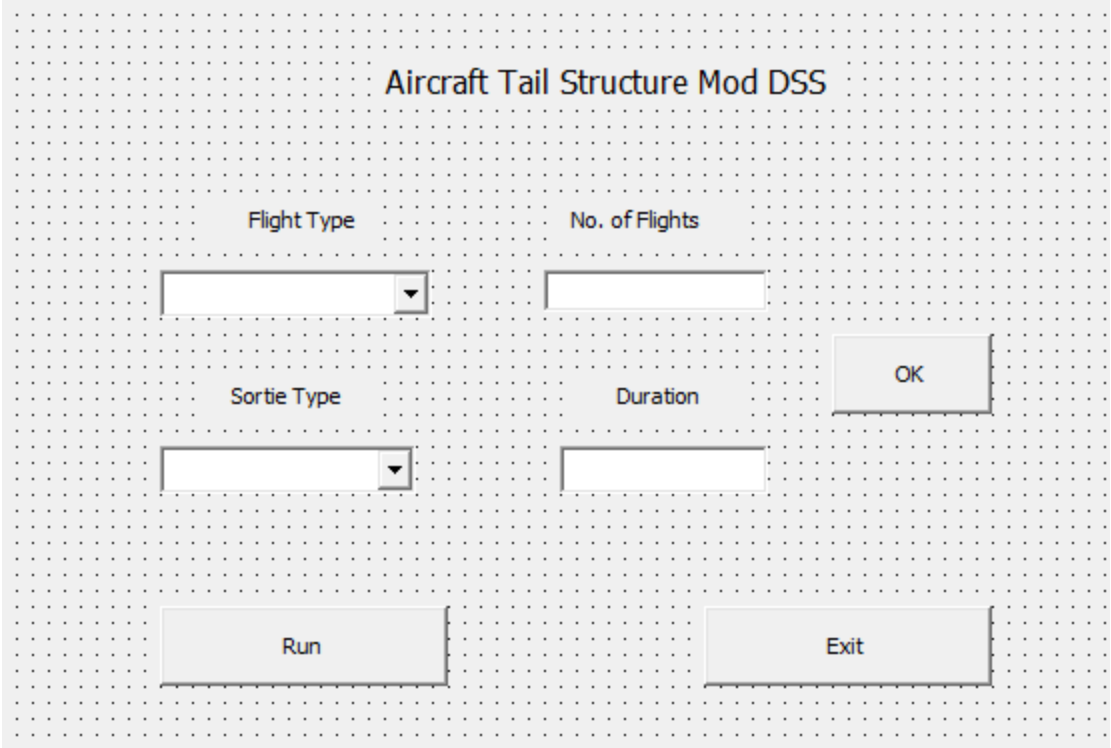

Figure 5.9 GUI for aircraft tail structure, Mod DSS

The entire DSS is developed on a combination of Microsoft Visual Basic.net (front end) and MATLAB (embedded search engine) platforms, with one user input form; one database, which contains all costing and damage rates stored on separate Excel files; and 10 major modules, which make up the algorithm for solving the multiobjective optimization problem. The decision variables for this DSS are:

x_i --- doublers at each fatigue-critical location, where $i = 1$ to n. x_i is a binary variable of 0 (no retrofitting) or 1 (install doublers).

y_i --- stiffeners at each fatigue-critical location. y_i is a binary variable of 0 (no retrofitting) or 1 (install stiffener).

The fatigue life at each fatigue-critical location can be calculated by querying the database tables. The database is made up of fatigue damage rates pre-calculated. Table 5-3 shows

the fatigue damage rates for the initial buffet flight. The damage rates are divided into the following three categories:

No_s_no_d --- damage rates for the case of no doublers and no stiffeners

D_only --- damage rates for the case of doublers only

S_only --- damage rates for the case of stiffeners only

The unit used for the damage rates is damage per 100,000 flights. Notice that the damage rates are nonlinear with respect to their locations and the installation of either doublers or stiffeners, which demonstrates the extent of complexity for this issue. This DSS can be applied to many similar situations due to the generality of the system. The architecture of this DSS is shown in figure 5.10.

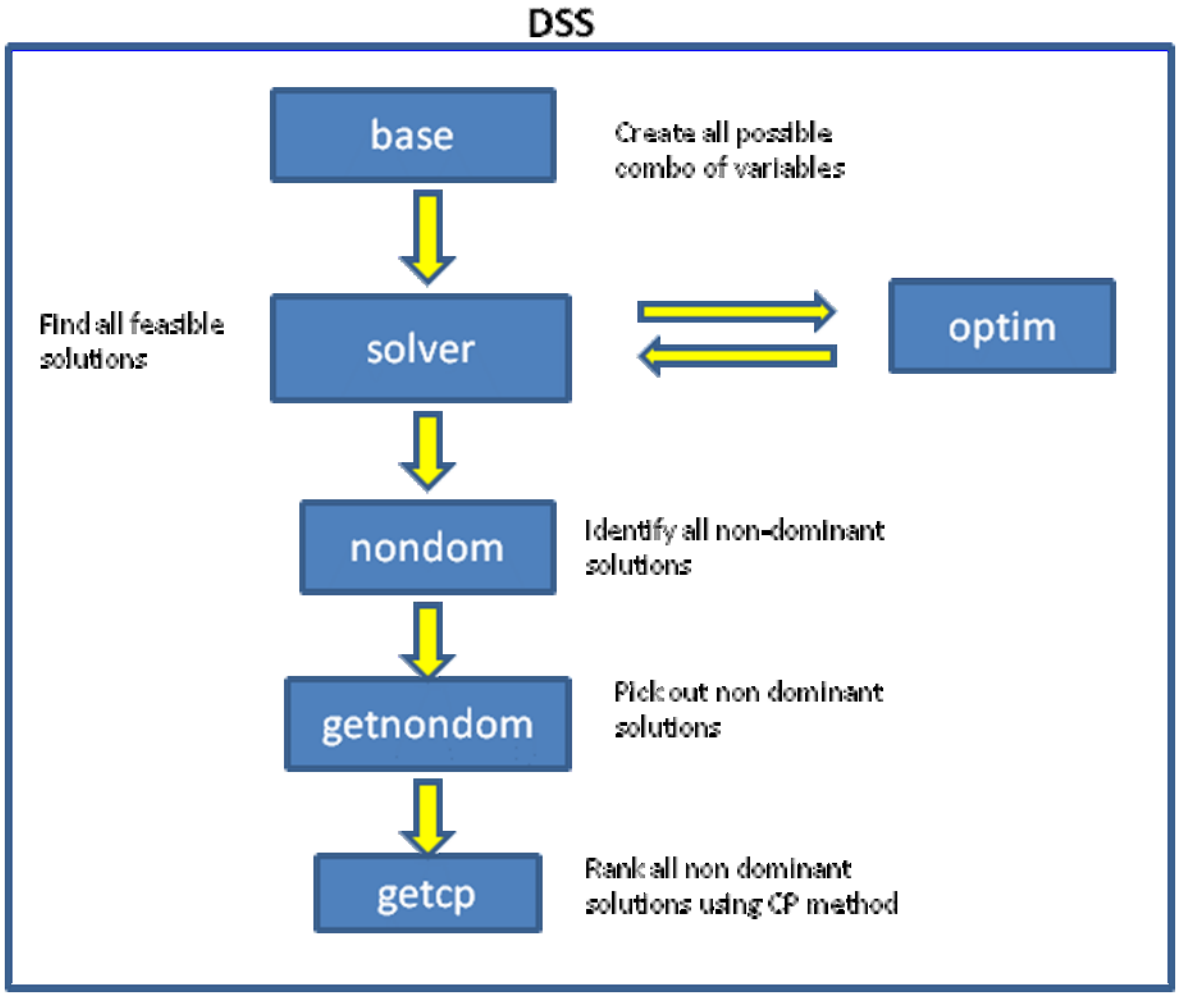

Figure 5.10 DSS architecture

5.5 TSEA IMPLEMENTATION DSS

The architecture of the DSS is shown in figure 5.10. The real optimization is accomplished with the module's "solver" and "optim." For each variable combination, the model "solver" calls the module "optim" to evaluate all hard and soft constraints. The DSS has embedded TSEA as its search engine, as shown in figure 5.11.

In the aircraft tail structure application, the objective space consists of system reliability and cost. Fitness is assigned by extracting each layer according to non-dominance. The new generation is composed of individuals passed on from the current generation by selecting

individuals with higher fitness as well as the tabu distance, crossover from the current generation, and mutation from the current generation. At the end of the fitness assignment, the maximum-fitness individuals are placed on the tabu list to guide the search.

The search starts with an initial generation by randomly selecting 200 from the pool of all individuals. The tabu list is initially empty. All individuals are assigned a fitness value. Individuals with the lowest (worst) level of fitness are put on the tabu list. The current generation is then chosen from the previous generation of higher-fitness individuals as well as new offspring obtained through crossover and mutation. Before the selection of the new generation, an average distance between each individual and all other individuals on the tabu list is calculated.

The parents for new offspring are selected based on fitness and this average distance (tabu distance). Binary tournament selection is used to select the parents. Crossover and mutation are conducted according to predetermined probabilities. The new generation comprises three parts: solutions passed on from the current generation through a picking process that selects individuals with higher fitness, solutions crossed over from the current generation, and solutions mutated from the current generation.

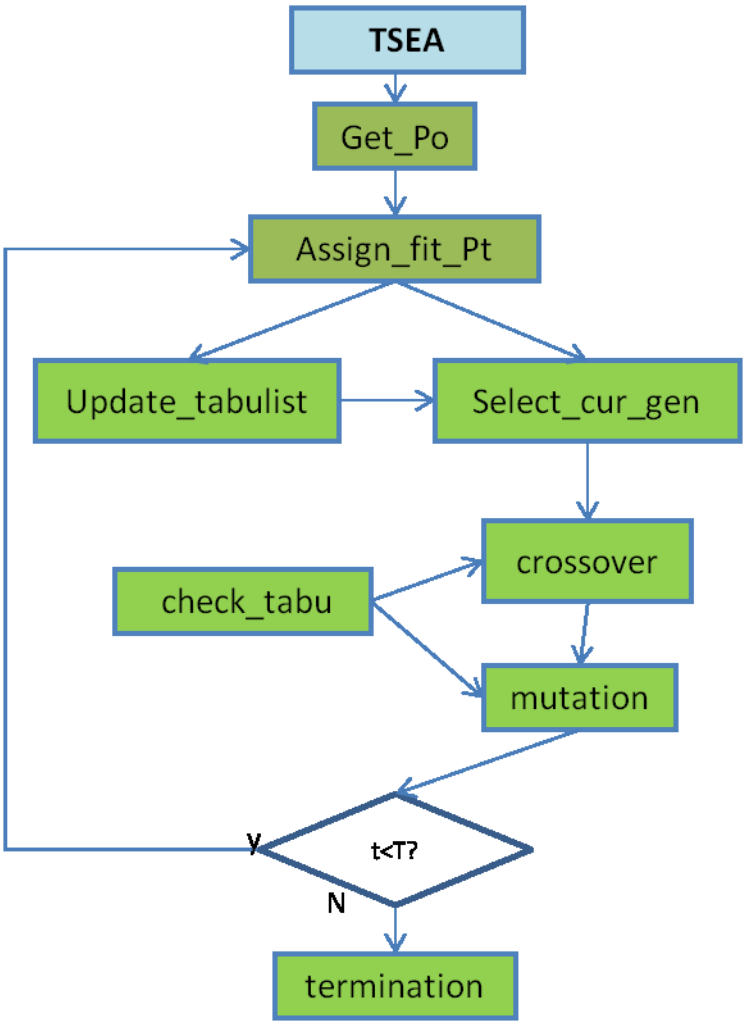

Figure 5.11 DSS search engine architecture

The new generation becomes the input for next iteration. The tabu list is updated by adding the new worst individuals. The search process continues until it reaches the maximum allowable iterations. After all the iterations have been gone through, all non-dominated solutions are summarized in table 5.7. Those non-dominated solutions are ranked by the compromise programming (CP) method as shown in section 5.7.

5.6 RESULTS COMPARISON WITH THE STATE OF THE ART

Provided herein is an innovative GA guided with tabu search to explore new ways for multiobjective GA optimization to converge to the Pareto-optimal front efficiently. The purpose of this research is to apply the EMOO principle in multiobjective optimization for analyzing the trade-offs between maximizing the reliability of aircraft horizontal tail structures and minimizing the cost associated with suggested retrofitting. To meet these challenges, an efficient EMOO algorithm called TSEA provides the DM with a small set of non-dominated solutions as the output of a DSS.

In the aircraft tail structure reliability and cost bi-criteria optimization application, TSEA provided Pareto-optimal solutions closer to the Pareto optimal front than those from SPEA2 and NSGA II, as shown in figure 5.12. At the same cost, TSEA found horizontal stabilizer retrofitting options that gave higher reliability (i.e., with longer fatigue lives). Results from NSGA II are close to those from TSEA, but they are spread out and scattered. SPEA2 provided closer and smoother results than did NSGA II. At a lower cost range, TSEA and SPEA2 reached very similar results. At a higher cost range, their results diverged with higher system reliabilities from TSEA. Notice the limitation of this comparison (bi-criteria): more comparisons of multiobjective system optimizations are needed to accurately rank TSEA with state-of-the-art meta-heuristic search algorithms.

Table 5.7 Non-dominated Solutions

	doublers installations														stiffeners installations														Cost	Reliability
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14		
1	0	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1220	0.90076
2	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1065	0.89995
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1000	0.89982
4	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	815	0.89907
5	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	750	0.89894
6	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	700	0.89781
7	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	635	0.89768
8	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	595	0.89687
9	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	560	0.89438
10	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	495	0.89425
11	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	495	0.89425
12	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	465	0.87559
13	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	400	0.87546
14	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	395	0.86433
15	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	330	0.8642
16	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	310	0.8619
17	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	310	0.8619
18	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	305	0.8535
19	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	230	0.84006
20	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	230	0.84006
21	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	225	0.83166
22	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	155	0.82614
23	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	155	0.82614
24	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0.81774
25	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70	0.80199
26	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	65	0.79359

The number of flights in the three categories of benign, medium, and severe are given as 1,000, 300, and 20, respectively. The crossover rate was set at 0.6, and the mutation rate was set at 0.2. The output from this multiobjective optimization DSS includes the number of doublers or stiffeners needed and their installation locations.

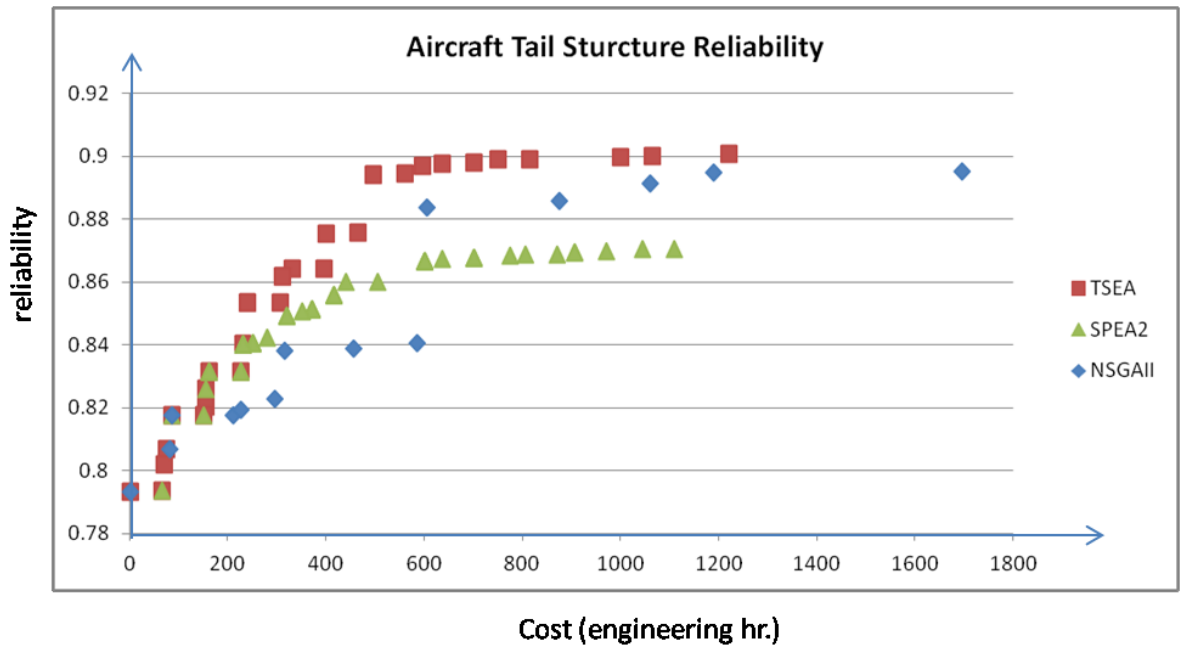

Figure 5.12 Aircraft tail structure reliability non-dominated solutions

Figure 5.13 shows the evaluation of the TSEA search process. It starts from a random set of retrofitting options and reaches its final 23 non-dominated solutions in 200 iterations.

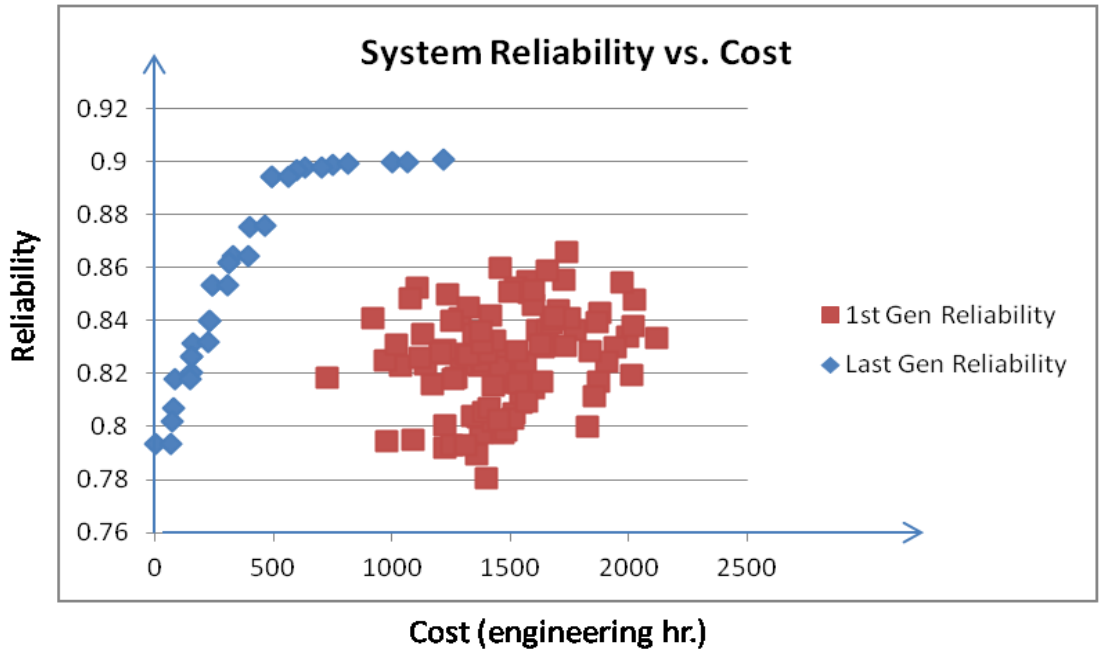

Figure 5.13 DSS results showing the Pareto-optimal front (in blue)

This aircraft tail structural reliability and cost optimization application was executed as a constrained bicriteria optimization problem. All fatigue damage accumulated at any bay cannot exceed the life time fatigue damage allowable for the same fatigue critical location. However, the constraint that in any given bay, there can be only one type of retrofitting installation or none has been relaxed until the final CP ranking phase.

The optimization outcomes from NSGA II and SPEA 2, shown in figure 5.12, resulted from developing three sets of MATLAB codes for all three algorithms. The MATLAB code for NSGA II is developed according to the original definition of those algorithms in the literature (Deb 2002), as shown in figure 5.16. The NSGA II code developed is validated by application to classical testing problems in chapter 6 and comparison of the results with those published in Deb. The SPEA2 code developed, shown in figure 5.15, is validated by applying it to benchmark testing problems in chapter 6 and comparing the results with those published in Zitzler (2001).

The MATLAB codes for all three algorithms are shown in figures 5.14, 15, and 16. The execution of the optimization process was done on a Dell Precision M4800 PC with the Windows 7 operating system.In the TSEA algorithm, the tabu-list is checked after the crossover and the mutation processes to ensure its convergence to the Pareto optimal front. The comparison of Pareto-optimal solutions from TSEA, NSGA-II, and SPEA2 was performed by running each MATLAB code on a Dell Precision M4800 PC with the Windows 7 operating system. The computing time elapsed is summarized in table 5.8.

Figure 5.14 showed the matlab code logic for the aircraft horizontal stabilizer reliability and cost optimization problem (Act).

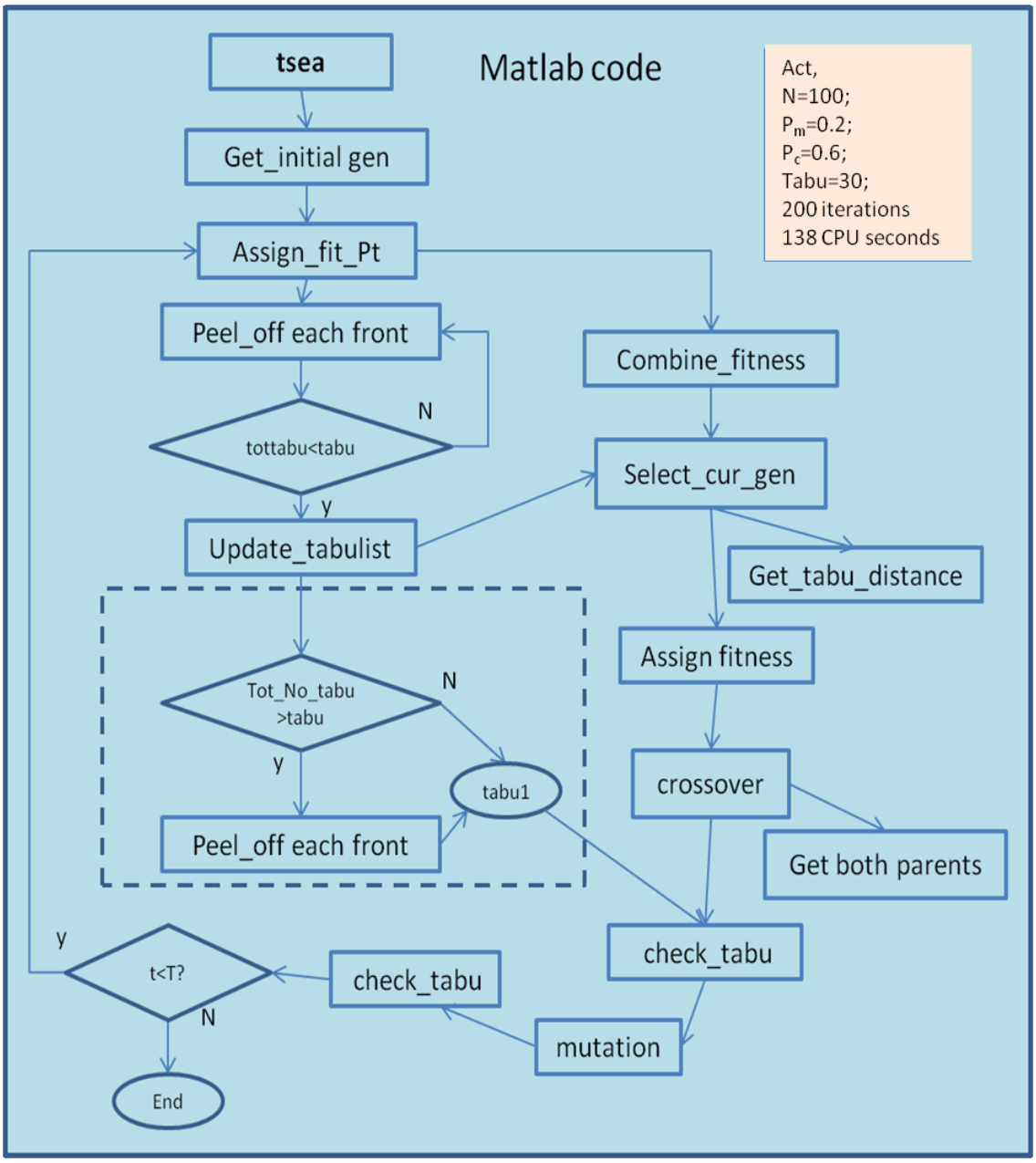

Figure 5.14 TSEA MATLAB code

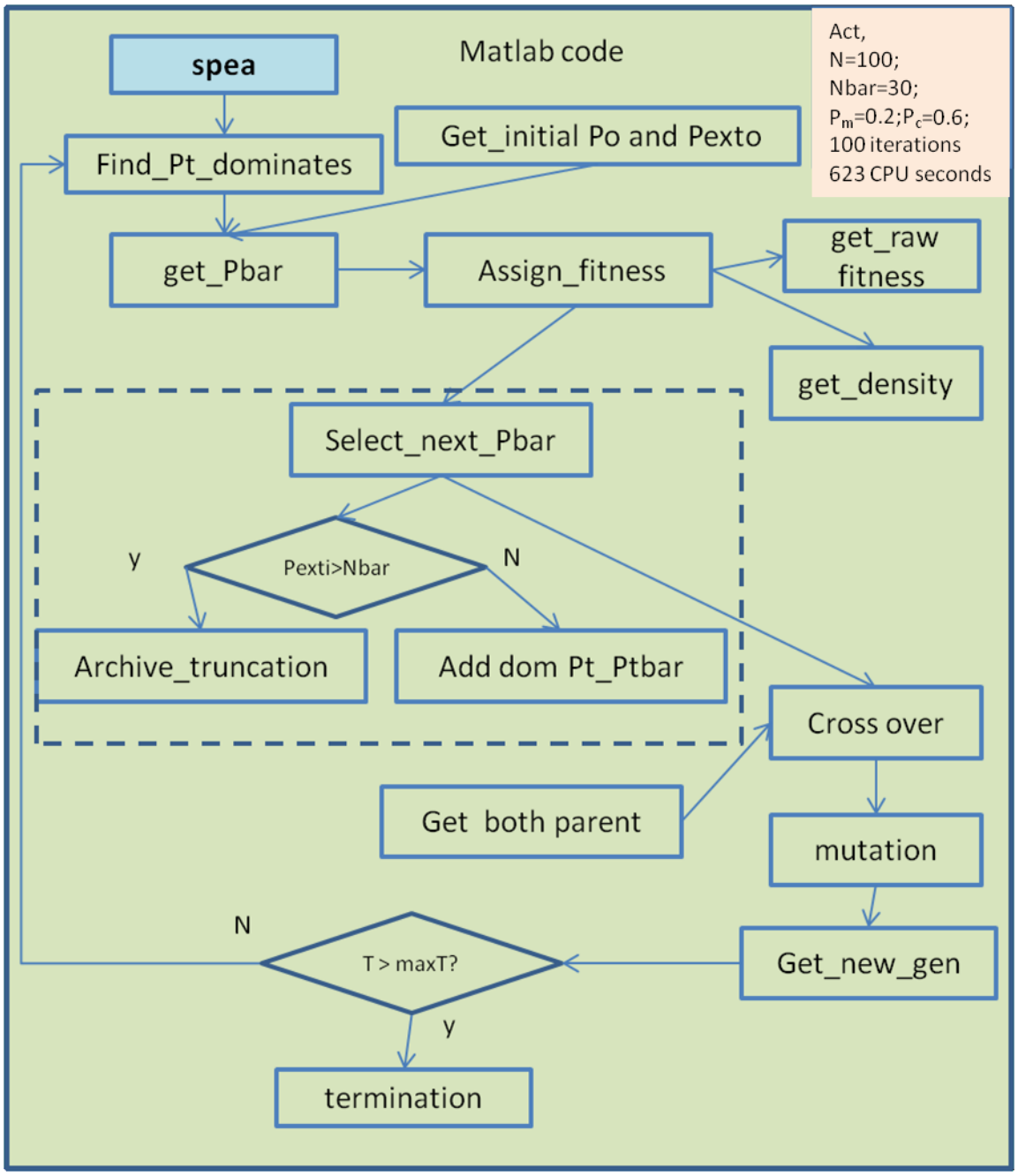

Figure 5.15 SPEA2 MATLAB code

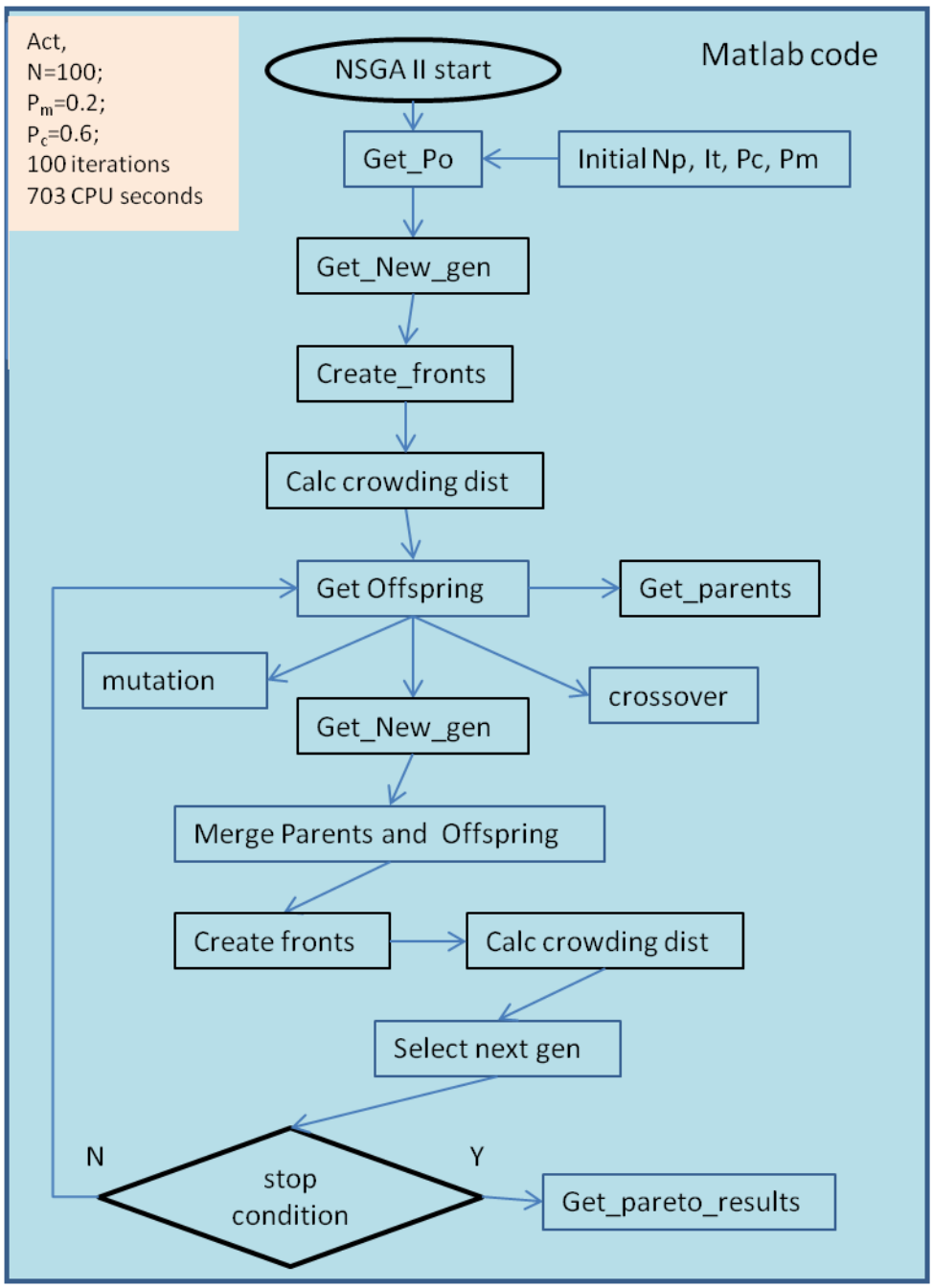

Figure 5.16 NSGA II MATLAB code

Table 5.8 Aircraft Tail Structure Optimization Computing Results

Algorithm	Application	Iteration	Computing time (sec.)	Population	External Population
TSEA	Aircraft horizontal stabilizer	200	138	100	tabu list = 30
NSGA II	Aircraft horizontal stabilizer	200	703	100	N/A
SPEA 2	Aircraft horizontal stabilizer	200	623	100	Nbar = 30

5.7 RANK PARETO RESULTS BY COMPROMISE PROGRAMMING

All three types of norms—the absolute distance, the Euclidean, and the min max norms of each non-dominated solution—were evaluated as shown in table 5.9. All non-dominated solutions were ranked by compromise programming method (Romero 1998). Large weights were assigned to each non-dominated solution according to the difficulty of installation, which is proportional to the number of bays that have to have installed both doublers and stiffeners. The more difficult the installation, the higher the retrofitting cost. The DSS runs for less than 15 seconds (Dell PC with 32-bit operating system, 8 GB RAM) though each generation, with convergence usually occurring in 200 generations.

The outputs of this multiobjective optimization DSS will include the number of doublers or stiffeners needed and their installation locations. The 22 non-dominated solutions shown in table 5.9 are ranked using the CP method. All weights are assumed to be 1. Figure 5.17 shows their rankings according to L^1, L^2, and L^∞. Only the top five non-dominated solutions (highlighted in red) are selected for each closeness norm measure.

Table 5.9 CP Ranking of Top Non-dominated Solutions

options	cost	reliability	L^1	L^2	L^∞
1*	1220	0.90076	1	1	1
2*	1065	0.89995	0.87336	0.74967	0.86580
3*	1000	0.89982	0.81829	0.65541	0.80952
4	815	0.89907	0.66512	0.42190	0.64935
5	750	0.89894	0.61006	0.35202	0.59307
6	700	0.89781	0.57731	0.30302	0.54978
7	635	0.89768	0.52225	0.24437	0.49351
8	595	0.89687	0.49517	0.21188	0.45887
9	560	0.89438	0.48810	0.18722	0.42857
10	495	0.89425	0.43304	0.14229	0.37229
11	465	0.87559	0.58118	0.17510	0.34632
12	400	0.87546	0.52612	0.13986	0.29004
13	395	0.86433	0.62564	0.19718	0.33993
14	330	0.8642	0.57058	0.16902	0.34114
15	310	0.8619	0.57472	0.17648	0.36260
16	305	0.8535	0.64877	0.23764	0.44098
17	230	0.84006	0.70925	0.34121	0.56639
18	225	0.83166	0.78330	0.43492	0.64477
19	155	0.82614	0.77420	0.49087	0.69628
20	150	0.81774	0.84825	0.60551	0.77466
21	70	0.80199	0.92595	0.84940	0.92162
22	65	0.79359	1	1	1

*Infeasible non-dominated solutions where both doublers and stiffener are required in certain bay.

At any given bay, only doublers or stiffener can be installed, but not both. This constraint has been relaxed (based upon DM's preference) to allow search to progress to the Pareto optimal front. The penalty of cost for installing both doublers and stiffeners in a certain bay has lowered the CP rankings. Those solutions (solutions 1, 2 and 3 in table 5.9) would not be recommended to DMs, as shown in figure 5.15. This DSS could serve as a robust tool for DMs attempting to efficiently modify existing aircraft tail structures to prolong the aircraft's fatigue life and thereby increase its reliability. If the goal were to double its reliability, this DSS could provide

retrofitting options that enable the aircraft to sustain two lifetimes of fatigue testing. This DSS runs 200 iterations on a Dell Tech PC in less than 10 minutes.

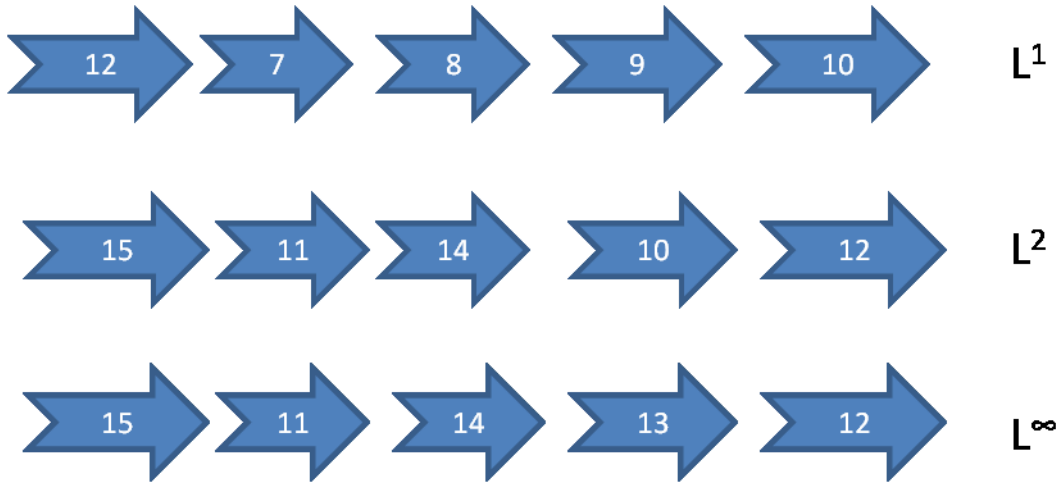

Figure 5.17 Ranking of non-dominated solutions by CP

The most significant contribution of this DSS is that all of its model parameters—such as damage rates, costs, and weights—for each FCL are chosen to be as close to reality as possible without any loss of generality. No real data are used in this research due to security issues. However, the fatigue damage rates are close to the real data for demonstrating the various algorithms. Therefore, this DSS can easily be adopted to solve similar multiobjective optimizations problems for any aircraft structure.

CHAPTER 6: COMPARISON OF THE TSEA WITH THE STATE OF THE ART ON TESTING PROBLEMS

Four problems were chosen to demonstrate TSEA's competitiveness to state-of-the-art algorithms such as NSGA II and SPEA2: two benchmark testing problems and two real-world applications. The two test problems from Zitzler (2000) are widely used by researchers to show how their algorithms fit in with the state-of-the-art algorithms. The first testing problem was selected for its convex nature and the second testing problem for its nonconvex nature to demonstrate the complete spectrum of TSEA's application range. One real-world application is a decision support system (DSS) to guide the aircraft horizontal tail retrofitting process by providing Pareto-optimal solutions under severe turbulent conditions.

The TSEA algorithm is embedded in this DSS to obtain the Pareto-optimal retrofitting solutions. The goal of this DSS is to minimize retrofitting costs and maximize the reliability of the horizontal tail structure. This multiobjective system reliability optimization application demonstrates the robustness of TSEA modeled in a real-world-variable environment, as shown in chapter 5. Another real-world application of TSEA is the vendor-managed inventory of the supplier chain. The purpose of this application of TSEA is to show how well the algorithm works in a combinatorial (binary variable) redundancy allocation type of multiobjective optimization environment, as shown in chapter 7. In this chapter, the focus is on comparing TSEA with state-of-the-art methods using two classical testing problems.

6.1 TESTING PROBLEM 1 STATEMENT

Zitzler (2000) suggested six test problems to compare multiobjective evolutionary algorithms. Two test problems are selected in this research to compare the proposed TSEA method with two state-of-the-art multiobjective evolutionary algorithms: SPEA2 and NSGA II. Test problem T_1 has a convex Pareto-optimal front:

$$f_1(x_1) = x_1$$

$$g(x_2, \ldots, x_m) = 1 + 9 * \sum_{i=2}^{m} \frac{x_i}{(m-1)}$$

$$f_2 = 1 - \sqrt{\frac{f_1}{g}}$$

Under the constraints: $0 < x_i < 1$, where $i = 1, \ldots, 30$

Test problem T_2 has a nonconvex Pareto-optimal front:

$$f_1(x_1) = x_1$$

$$g(x_2, \ldots, x_m) = 12 + 9 * \sum_{i=2}^{m} \frac{x_i}{(m-1)}$$

$$f_2 = 1 - (\frac{f_1}{g})^2$$

where $m = 30$ and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g(\boldsymbol{x}) = 1$.

The constraint $x_i \in [0,1]$ is enforced by only allowing each x_i to be within the range of 0 and 1 when selecting each generation. Therefore, optimization is performed within the feasible decision space.

6.1.1 ENVIRONMENT SETTINGS FOR TESTING PROBLEM 1

The following parameters are chosen to ensure consistency of comparison:

Popsize = 100

Maxiter = 250

Crossover rate = .6

Mutation rate = 0.2

Total variable $n = 30$

All algorithms are coded and executed in MATLAB on a Dell PC. All three algorithms, NSGA II, SPEA2, and TSEA, are run using the same environment settings. The original

population is selected by random. For TSEA, the tabu list size is set at 30, while for the SPEA2 the archive size is fixed at 100.

6.1.2 BENCHMARK TEST PROBLEM #1 RESULTS COMPARISON

For this convex optimization problem, TSEA results fit between that of NSGA II and SPEA2 in terms of both the spread of Pareto-optimal solutions and the magnitude of the solutions. SPEA2 tends to have a few outliers. In general, all three algorithms have fairly consistent results. NSGA II, however, has provided more Pareto-optimal solutions than SPEA2 and TSEA. SPEA2 has the best set of results in terms of minimizing both objectives. TSEA provides consistent results with the other two algorithms without any outliners. The comparison is shown in figure 6.1 for the first testing problem.

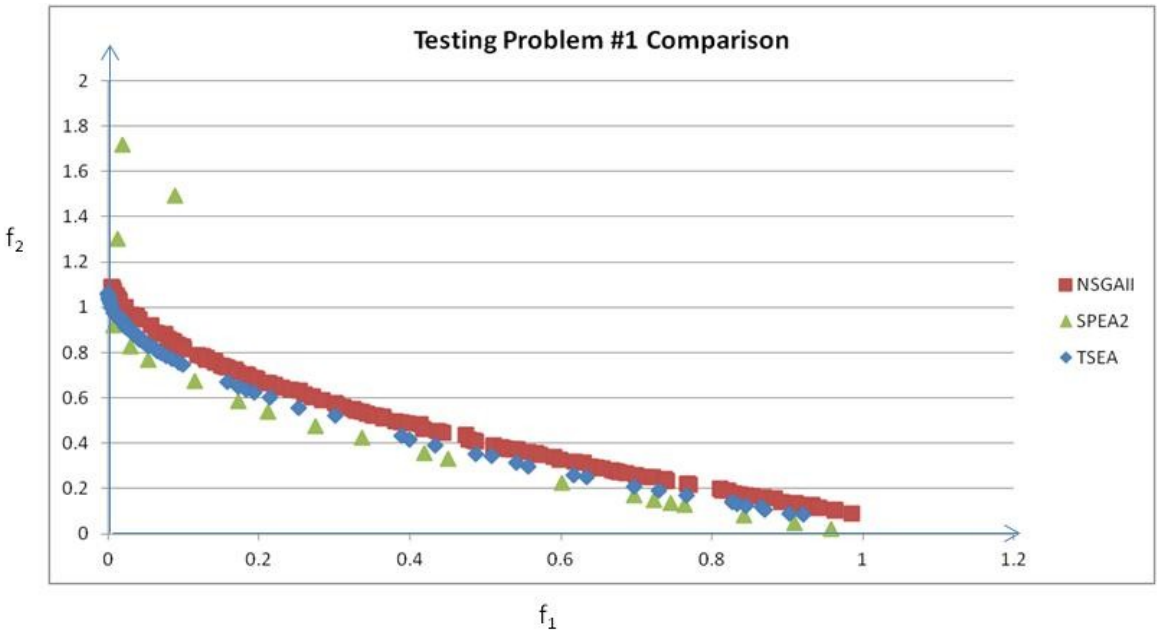

Figure 6.1 Testing problem #1 results comparison

The optimization results from NSGA II and SPEA 2 shown in figure 6.1 were obtained by developing three sets of MATLAB codes for all three algorithms. The MATLAB code for NSGA II was developed according to the original definition of those algorithms in literature

(Deb 2002). The NSGA II code developed was validated by applying it to classical testing problems in this chapter and comparing the results with those published in Deb. The SPEA2 code developed was validated by applying it to benchmark testing problems in this chapter and comparing the results with those published in Zitzler (2001).

The MATLAB codes for all three algorithms are shown in figures 5.14, 5.15, and 5.16. The optimization process was executed on a Dell Precision M4800 PC with the Windows 7 operating system. Different environmental parameters were input to the three algorithms for this classical testing problem.

For testing problem 1, the tsea code executed for 223 seconds to complete 200 iterations with the following input parameters:

N--------- Population size of 100

Tabu----- Tabu list size of 30

P_c-------- crossover rate of 0.60

P_m--------mutation rate of 0.20

For testing problem 1, the spea code executed for 610 seconds to complete 250 iterations with the following input parameters:

N------- Population size of 100

N_{bar}----$\overline{N}$ population of 30

P_c------ crossover rate of 0.60

P_m------mutation rate of 0.20

For testing problem 1, the NSGAII code executed for 232 seconds to complete 250 iterations with the following input parameters:

N_p----- Population size of 100

P_c----- crossover rate of 0.60

P_m-----mutation rate of 0.20

The comparison of Pareto-optimal solutions from TSEA, NSGA-II, and SPEA2 was performed by running each MATLAB code on a Dell Precision M4800 PC with the Windows 7 operating system. The computing time elapsed is summarized in table 6.1.

Table 6.1 Testing Problem #1 Optimization Computing Results

Algorithm	Application	Iteration	Computing time (sec.)	Population	External Population
TSEA	Testing problem #1	200	223	100	tabu list = 30
NSGA II	Testing problem #1	200	232	100	N/A
SPEA 2	Testing problem #1	200	610	100	Nbar = 30

6.2 TESTING PROBLEM 2 STATEMENT

Test problem T_2 represents a nonconvex Pareto-optimal front. This test problem was used by Zitzler (2000), who used six test problems to compare several state-of-the-art evolutionary algorithms. One of these six test problems was chosen as test problem T_2 in this research.

$$f_1(x_1) = x_1$$

$$g(x_2, \ldots, x_n) = 1 + 9 * (\sum_{i=2}^{n} xi)/(n-1)$$

$$h(f_1, g) = 1 - (\frac{f_1}{g})^2$$

$$f_2 = g * h$$

6.2.1 ENVIRONMENT SETTINGS FOR TESTING PROBLEM 2

The following parameters are chosen to ensure consistency of comparison:

Popsize = 100;

Maxiter = 200;

Number of variables = 30;

Crossover rate = .6;

Mutation rate = 0.2;

Total variable n = 30.

6.2.2 BENCHMARK TEST PROBLEM #2 RESULTS COMPARISON

All three algorithms were coded and executed in MATLAB both for benchmark testing problems and to verify the optimization results from the original analysis in literature. All three algorithms were run using the same environment settings. The original population was selected at random. For TSEA, the size for the tabu list was chosen as 30, whereas the archive size was fixed at 100 for SPEA2.

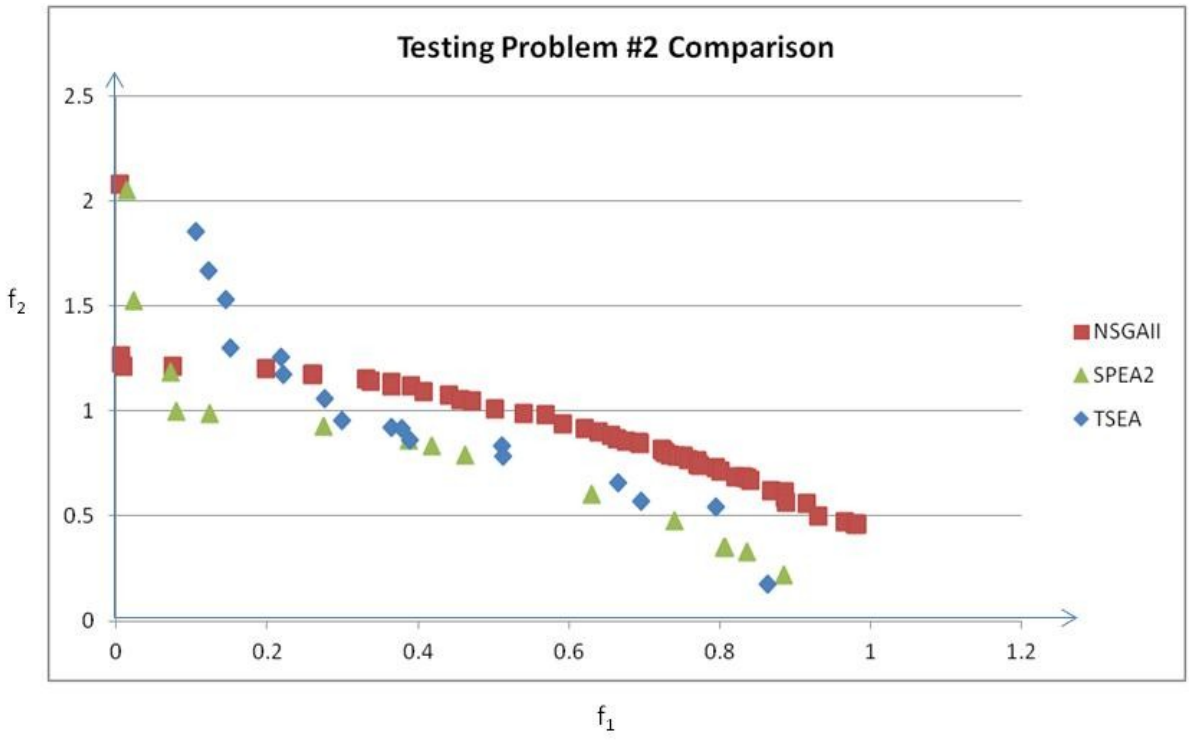

Figure 6.2 Testing problem #2 results comparison

For this nonconvex testing problem, SPEA2 gave the best Pareto-optimal set, while NSGA II provided more non-dominated solutions. Again, TSEA fits right in the middle of NSGA II and SPEA2, as shown in figure 6.2. The optimization results from NSGA II and SPEA

2, shown in figure 6.2, were developed by developing three sets of MATLAB codes for all three algorithms. The MATLAB code for NSGA II was developed according to the original definition of those algorithms in the literature (Deb 2002). The NSGA II code developed was validated by applying it to classical testing problems in chapter 6 and comparing the results with those published in Deb. The SPEA2 code developed was validated by applying it to benchmark testing problems in chapter 6 and comparing the results with those in Zitzler (2001).

The MATLAB codes for all three algorithms are shown in figures 5.14, 5.15, and 5.16. The execution of the optimization process was done on a Dell Precision M4800 PC with the Windows 7 operating system. For the second testing problem, the TSEA code (figure 5.14) executed for 284 seconds to complete 200 iterations with the following input parameters:

N----- Population size of 100

tabu-----tabu lise size of 30

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

For testing problem 2, the spea code (figure 5.15) executed for 510 seconds to complete 200 iterations with the following input parameters:

N----- Population size of 100

N_{bar}----$\overline{N}$ population of 30

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

For testing problem 2, the NSGAII code (figure 5.16) executed for 170 seconds to complete 200 iterations with the following input parameters:

N_p----- Population size of 100

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

The comparison of Pareto-optimal solutions from TSEA, NSGA-II, and SPEA2 was performed by running each MATLAB code on a Dell Precision M4800 PC with the Windows 7 operating system. The computing time elapsed is summarized in table 6.2.

Table 6.2 Testing Problem #2 Optimization Computing Results

Algorithm	Application	Iteration	Computing time (sec.)	Population	External Population
TSEA	Testing problem #2	200	284	200	tabu list = 30
NSGA II	Testing problem #2	200	170	200	N/A
SPEA 2	Testing problem #2	200	510	200	Nbar = 30

CHAPTER 7: COMPARISON OF TSEA WITH STATE OF THE ART ALGORITHMS FOR A SUPPLY CHAIN MANAGEMENT PROBLEM

Supply chain management (SCM) is critical in today's competitive marketplace. With an integrated supply chain, companies can reduce costs and enhance their competitiveness. SCM plays an important role in the coordination among a network of facilities and distribution options to increase the efficiency and responsiveness of all parties in the network.

In this research, a bi-objective vendor-managed inventory model in a supply chain with one vendor and several retailers was developed to minimize the total cost and maximize network reliability. The two objectives conflict. Meanwhile, many constraints have to be satisfied. While the demand rates of the retailers are deterministic and known, the constraints are the total budget, storage space required, vendor's total replenishment frequencies, and average inventory.

In addition to the production and storage costs of the vendor and the retailers, the transportation cost of delivering the item to retailers is also considered part of the total chain cost. The goal is to find the order size, the replenishment frequency of the retailers, the optimal traveling route from the vendor to the retailers, and the number of machines so as to minimize the total chain cost while maximizing the system reliability of producing the item. Because all variables are coded in an arbitrary manner and not all time-related intervals are specifically spelled out, the model used here can be easily adapted for any business situation.

7.1 MATHEMATICAL MODEL

The following notations and assumptions (from Sadeghi [2014]) are used throughout this section:

i an index used for a retailer; $i = 1, 2, \ldots, r$

j an index used for a machine; $j = 1, 2, \ldots, m$

A_i ordering cost for retailer i

A ordering cost for the vendor

h_i holding cost for retailer i

H holding cost for the vendor

P vendor's production rate (item/time unit)

n_i vendor replenishment frequency of retailer i (decision variable)

n_1 vendor replenishment frequency of retailer *1*

d_i demand rate of retailer i (item/time unit)

d_1 demand rate of retailer *1*

D demand rate of the vendor

q_i order quantity for retailer i

q_1 order quantity for retailer *1* (decision variable)

Q_v total vendor order quantity

Z upper bound for the average inventory of the vendor

f space required to store one unit of the product

F total available storage space for retailers

Λ upper bound for replenishment frequencies

TC total cost of the VMI system

R production system reliability

R_j reliability of machine j

N_j number of redundant machines j (decision variable)

C_j purchasing cost of machine j

e total available space to install machines

s space occupied by a machine

B budget available to install machines

S_{kl} cost of transportation from retailer k to retailer l

T production cycle time

x_{kl} 1 if retailer k is reached from retailer l; 0 otherwise (decision variable)

Sadeghi (2014) assumed an equal consumption interval for retailers; the following equation reflected this assumption:

$$\frac{n_i q_i}{d_i} = \frac{n_z q_z}{d_z}$$

where z ε {1, 2, . . . , r} is based on (1). The order quantity of the ith retailer is obtained by

$$q_i = \frac{d_i * n_1 * q_1}{d_1 * n_i}$$

where d_1 was utilized to simplify modeling.

By minimizing the total system inventory cost and maximizing the total system reliability, the multiobjective VMI problem can be modeled by the following equation (Sadeghi, 2014):

$$\text{MinTC(minimize total cost)} = \frac{Ad_1}{n1*q1} + \frac{H*ni*q1}{2*d1*P}\sum_{i=1}^{r}\frac{d_i^2}{n_i} + \frac{d_1}{n_1 q_1}\sum_{i=1}^{r} A_i n_i$$

$$+\frac{n_1 q_1}{2d_1}\sum_{i=1}^{r} h_i d_i\left(1 - \frac{d_i}{P} + \frac{d_i}{n_i P}\right)$$

$$+\sum_{k=1}^{r}\sum_{l=1}^{r} S_{kl}x_{kl} + \sum_{j=1}^{m} C_j N_j$$

$$\text{MaxR(maximize reliability)} = \prod_{j=1}^{m}\left[1 - \left(1 - R_j\right)^{N_j}\right]$$

s.t.

$$\sum_{i=1}^{r}\frac{f d_i n_1 q_1\left(1-\frac{d_i}{P}+\frac{d_i}{n_i P}\right)}{d_1} \leq F$$

The warehouse space for all retailers is limited to F.

$$n_1 q_1 (\sum_{i=1}^{r} \frac{d_i^2}{n_i}) / 2Pd_1 \leq Z$$

Vendor's average inventory is restricted to Z.

$\sum_{i=1}^{r} n_i \leq \lambda$

The replenishment frequency is limited to λ.

$\sum_{j=1}^{m} C_j N_j \leq B + \sum_{j=1}^{m} C_j$ --- the budget constraint

$\sum_{j=1}^{m} N_j \, s \leq e$

The space occupied by redundant machines is restricted to a predetermined value.

$\sum^{r}{}_{k=1} x_{kl} = 1; \; \forall l = 1{:}r$

It is assumed that the vendor delivers the products to all the retailers using a single vehicle. It is also assumed that the optimal traveling route from vendor to retailers is a base for calculating transportation cost. The vehicle is not allowed to pass a retailer's location twice to save money and time.

N_j, n_i, q_1, x_{kl} are nonnegative integers.

All of the above constraints were satisfied by checking the feasibility whenever the optimization algorithms need to select a new individual from the predefined generation pool. The constraints boundary data for four retailers and seven machines are shown in table 7.1.

Table 7.1 Basic Data of the Numerical Example

	Reliability	Cost			demand rate	ordring cost	holding cost
machine 1	0.65	19342		retailer 1	8361	58	6
machine 2	0.25	11293		retailer 2	7376	95	8
machine 3	0.36	33194		retailer 3	1388	27	2
machine 4	0.11	43688		retailer 4	2313	68	7
machine 5	0.73	32276					
machine 6	0.95	43596					
machine 7	0.77	18198					

The transportation costs for retailers are shown in table 7.2.

Table 7.2 Transportation Costs between Retailers

	Ratiler1	Ratiler2	Ratiler3	Ratiler4
Ratiler1	0	473	319	576
Ratiler2	473	0	399	708
Ratiler3	319	399	0	676
Ratiler4	576	708	676	0

All of the boundary data for each constraint are shown in table 7.3.

Table 7.3 Supplier Chain Management Boundary Limitations

ID	Limits	Description
H	12	holding cost of the vendor
P	23974	vendor's production rate (item/time unit)
A	265	ordering cost of the vendor
B	450000	available budget to install machines
s	3	occupied space by a machine
e	100	total available space to install machines
Z	2000	vendor's average inventory
f	3	space required storing one unit of product
F	3000	total available storage space for retailers
λ	120	upper bound for replenishment frequencies

7.2 TSEA COMAPRE TO STATE OF THE ART

All three algorithms: NSGA II, SPEA2, and TSEA were run on the SCM model using the same initial environment settings and number of iterations. Initial environmental conditions are as follows:

$N = 100$

$P_m = 0.2$ (mutation rate)

$P_c = 0.6$ (crossover rate)

$Tabu = 30$ (used in TSEA as the size of its tabu list)

$\bar{N} = 100$ (used in SPEA2 as the size of its archive)

The system reliabilities vs. costs are plotted in figure 7.1. After a set number of iterations, TSEA provided Pareto-optimal solutions superior to those of SPEA2 and NSGA II, as shown in

figure 7.1. For solutions with the same cost, TSEA finds optimized solutions in the SCM system that has higher reliability. Therefore, TSEA is able to reach non-dominated solutions that are closer to the Pareto optimal front than NSGA II and SPEA 2. The objectives are to minimize cost and maximize system reliability. SPEA2 exhibited some divergence from NSGA II and TSEA**.** The best Pareto-optimal results TSEA found are attributed to its long-term memory, which in turn is due to the tabu list, which helped guide the search to zoom in on the Pareto front faster than the other two algorithms did. Current application is in bicriteria optimization of the SCM, which demonstrated the partial superiority of TSEA over SPEA2 and NSGA II. More comparisons of a larger number of objective functions will be conducted in the future to complete this claim.

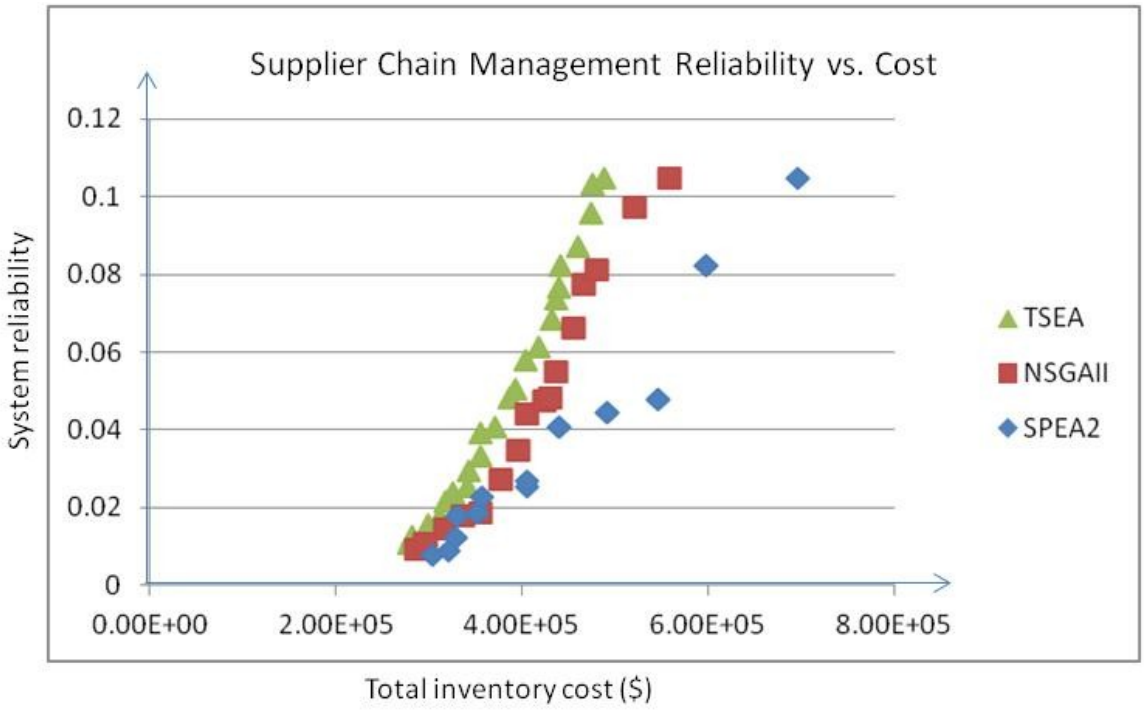

Figure 7.1 SCM system reliability comparisons

The optimization results from NSGA II and SPEA 2, shown in figure 7.1, were derived by developing three sets of MATLAB codes for all three algorithms. The MATLAB code for

NSGA II was developed according to the original definition of those algorithms in the literature (Deb 2002). The NSGA II code developed was validated by applying it to classical testing problems in chapter 6 and comparing the results with those in Deb. The SPEA2 code developed was validated by applying it to benchmark testing problems in chapter 6 and comparing the results with those in Zitzler (2001).

The MATLAB codes for all three algorithms are shown in figures 5.14, 5.15, and 5.16. The execution of the optimization process was done on a Dell Precision M4800 PC with the Windows 7 operating system. For the supplier chain management problem, the TSEA code (figure 5.14) executed for 249 seconds to complete 100 iterations with the following input parameters:

N----- Population size of 100

tabu-----tabu lise size of 30

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

For the supplier chain management problem, the spea code (figure 5.15) executed for 196 seconds to complete 100 iterations with the following input parameters:

N----- Population size of 100

N_{bar}----$\overline{N}$ population of 30

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

For the supplier chain management problem, the NSGAII code (figure 5.16) executed for 688 seconds to complete 100 iterations with the following input parameters:

N_p----- Population size of 100

P_c------ crossover rate of 0.60

P_m-----mutation rate of 0.20

The comparison of Pareto-optimal solutions from TSEA, NSGA-II, and SPEA2 was performed by running each MATLAB code on a Dell Precision M4800 PC with the Windows 7 operating system. The computing time elapsed is summarized in table 7.4.

Table 7.4 Supplier Chain Management Optimization Computing Results

Algorithm	Application	Iteration	Computing time (sec.)	Population	External Population
TSEA	Supplier Chain Management	200	249	100	tabu list = 30
NSGA II	Supplier Chain Management	200	688	100	N/A
SPEA 2	Supplier Chain Management	200	196	100	Nbar = 30

CHAPTER 8: SUMMARY AND CONCLUSIONS

8.1 SUMMARY

Aircraft horizontal stabilizers sometimes have to encounter stall buffet loading conditions, which may have a significant impact on fatigue life. When simulated stall buffet loads are applied to the horizontal stabilizers during full-scale fatigue tests, it is essential that the structure is properly retrofitted to be able to sustain this type of loading throughout the testing. A set of balanced retrofitting options is needed for the DM to have higher structural reliability with minimum cost. An innovative tabu evolutionary algorithm, TSEA, was developed to meet this challenge. This algorithm can also be the engine of a DSS to help the DMs perform trade-off studies, as shown in chapter 5. The purpose of this dissertation is to propose a novel multiobjective optimization evolutionary algorithm (TSEA) created through a combination of the metaheuristic search method TS and the genetic algorithm. This algorithm was compared with evolutionary, state-of-the-art, multiobjective optimization algorithms, NAGA II and SPEA2, on benchmark testing problems and two real-world applications. One of them was for SCM coded in real variables; the other was for an aircraft structure coded in binary variables.

8.2 CONCLUSIONS

The objectives for both real-world applications were to minimize cost and maximize system reliability. By comparing the proposed algorithm (TSEA) with two state-of-the-art multiobjective optimization algorithms, this dissertation showed that TSEA outperformed those algorithms by providing a set of non-dominated solutions closer to the Pareto-optimal front. This innovative multiobjective optimization algorithm was also embedded in a DSS for the aircraft tail structure modification to provide the DM with structural retrofitting Pareto-optimal options to maximize the aircraft's fatigue life and reduce cost at the same time based upon user input. This DSS can serve as a robust tool to help DMs determine how to efficiently modify an existing

structure to be able to sustain two lifetimes of fatigue testing, by which one lifetime of a structure without cracking is warranted. The constraint handling of TSEA partially relies on the interaction between the DM and DSS so that some constraints may be relaxed to find the non-dominated solutions that are close to the Pareto optimal front. The significant contribution of this DSS is that all of its model parameters, such as damage rates and costs for each bay, are chosen to be as close to reality as possible and yet can be modified to represent different aircraft configurations.

This DSS can easily be adopted to solve similar multiobjective optimizations problems for any type of aircraft tail structure. Contribution to the state of the art in multiobjective system reliability optimization is provided by the introduction of the TSEA algorithm. TSEA is much more effective than SPEA 2 thanks to its application of a tabu list to guide the search to the optimum solution and its avoidance of niching, a costly process in terms of CPU time. This research also demonstrates that TSEA competes with and in some situations out performs state-of-the-art multiobjective optimization algorithms such as NSGA II and SPEA 2 when applied to classic bi-criteria bench mark test problems and other complex, real world sizable applications.

8.3 FUTURE RESEARCH DIRECTIONS

The findings suggest several promising areas for future research. The key factor for TSEA to outperform NSGA II and SPEA 2 on some large-scale, complex, real bicriteria optimization applications is the introduction of the tabu list to help guide the search and avoid pitfalls in reaching the Pareto-optimal front. How long this list should be and how long each individual should be kept in this list are questions for future research. Elitism and the tabu list are two opposite approaches to guide multiobjective optimization searches to their Pareto-optimal solutions. A comparison of the effectiveness of both methods is needed to guide future research efforts, especially with regard to studying which method is most effective for which types of applications. Fuzzy logic has been introduced into multiobjective optimization research. More

studies are needed that incorporate fuzziness in multiobjective evolutionary and metaheuristic algorithms to provide more realistic solutions to real-world engineering applications.

This research provided a novel metaheuristic evolutionary algorithm, TSEA, for multiobjective optimization, applying the concept of the metaheuristic tabu search to evolutionary approach of the genetic algorithms. Now that TSEA has been successfully applied to bicriteria structural reliability optimization problems in the area of aircraft structure, it needs to be applied in true multiobjective system reliability problems to further verify its effectiveness. A comparison of TSEA with current options, such as NSGA II and SPEA2, on more large-scale multiobjective real-world applications will be provided in follow-up research to rank the efficiency of all applied metaheuristic optimization algorithms.

BIBLIOGRAPHY

Agarwal M., 2010. Optimal redundancy allocation in complex systems, Journal of Quality in Maintenance Engineering, V 16 No. 4 413-424.

Armentano, V. A. (2004). An application of a multiobjective tabu search algorithm to a Bicriteria flow shop problem, J. Heuristics, 10(5), 463-81.

Arostegui Jr., M. A., 2006. An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems, International Journal of Production Economics, 103 742-754.

Aytug, H., 2002. Solving large-scale maximum expected covering location problems by genetic algorithms: A comparative study. European Journal of Operational Research, 141 480-494.

Back, T. (1996). Evolutionary algorithms in Theory and Practice, New York, etc.: Oxford University Press.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In J. J. Grefenstette (Ed.). Proceedings of an International Conference on Genetic algorithms and Their Applications, pp. 101-111, sponsored by Texas Instruments and U.S. Navy Center for Aplied Research in Artificial Intelligence (NCARAI).

Barichard, V. (2009). Multiobjective programming and goal programming: theroretical results and practical applications. Springer, Berlin.

Bhattacharya, R., 2010. Solving conflicting bi-objective facility location problem by NSGA II evolutionary algorithm. International Journal of Advanced Manu factoring Technology, 51 397-414.

Blickle, T. (1996). Theory of Evolutionary algorithms and Application to System-Synthesis. Ph. D. thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, ETH diss no. 11894.

Blickle, T. J., (1998). System-level synthesis using evolutionary algorithms. Design Automation for Embedded Systems (1), 23-58.

Buck, J. (1994). Ptolemy: A framework for simulating and prototyping heterogeneous systems. International Journal on Computer Simulation 4, 155-182.

Carlos, A. (2002). Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering, 191(11-12), 1245-87.

Cieniawski, S. E. (1995). Using genetic algorithms to solve a multi objective groundwater monitoring problem, Water Resource Research 31(2), 399-409.

Coello CA, (2007). Evolutionary algorithms for solving multiobjective problems, 2nd edn. Springer, Berlin.

Deb, K., 1999. Multiobjective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Evolutionary Computations, Vol 7, No. 3 205-230.

Deb, K., 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evelutionary computation, vol. 6, No. 2, 182-197.

Dorn, J., 1996. Comparison of iterative improvement techniques for schedule optimization. European Journal of Operational Research 94 349-361.

Fonseca, C. M., 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, Proceedings of the Fifth International Conference, San Mateo, CA.

Fonseca, C. M. and Fleming, P.J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. Proceedings of the Fifth International Conference on Genetic Algorithms, pages 416-423, Morgan Kaufmann, San Mateo, California.

Fonseca, C. M. and Fleming, P. J. 1995. An overview of evolutionary algorithms in multiobjective optimization, Evolutionary computation 3(1), 1-16.

Ghezavati VR. (2009). A new heuristic method for distribution networks considering service level constraint and coverage radius. Expert Systems Appl 36:5620-5629.

Guliashki, V., 2009. Survey of evolutionary algorithms used in multiobjective optimization. Problems of Engineering Cybernetics and Robotics, vol. 60, 42-54.

Hajela, P. and Lin, C.-Y. (1992). Genetic search strategies in multicriterion optimal design. Structural Optimization 4, 99-107.

Horn, J., 1993. Multiobjective Optimization Using the Niche Pareto Genetic Algorithm. Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World congress on Computational Intelligence, Vol. 1, 1994.

Jadaan, OA, Rajamani, L., Rao, CR., "Non-dominated ranked genetic algorithm for solving multiobjective optimization problems: NRGA", Journal of Theoretical and Applied Information Technology, Vol. 2, pp. 60-67, 2008.

Janssens, G. K., 2008. Recent challenges in the use of evolutionary algorithms for multiobjective optimization. Proceedings of the International Conference on Information Technologies, 19-20 Sept. 2008, Bulgaria, 51-60.

Kale, A. A., etal. 2008. Tradeoff of weight and inspection cost in reliability based structural optimization. Journal o Aircraft, v45, n 1, p 77-85.

Laumanns, M., 2002. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. Evolutionary Computation 10(3):263-282.

Le Grauffre, P., (2007). A multicriteria decision support methodology for annual rehabilitation programs of water networks, Computer Aided Civil and Infrastructure Engineering, 22(7), 478-88.

Liang, L. Y., 2004. Optimizing the design of sewer networks using genetic algorithms and Tabu Search, Engineering, Construction and Architectural Management, 11 No. 2 101-112.

Liang, Y. H., 2008. Combining neural networks and genetic algorithms for predicting the reliability of repairable systems, International Journal of Quality & Reliability Management, V 25, No. 2 201-210.

Manzini R, (2007). Optimization models for the dynamic facility location and allocation problem. Int J Prod Res 46(8):2061-2086.

Mathakari, S. (2007). Reliability-based optimal desing of electrical transmission towers using multiobjective genetic algorithms, Computer Aided Civil and Infrastructure Engineering, 22(4), 282-92.

Molent, L., Barter, S.A. 2007. A comparison of crack growth behavior in several full-scale airframe fatigue tests. International Journal of Fatigue 29(2007) 1090-1099.

Nakagawa, Y., 1981. An experimental comparison of the heuristic methods for solving reliability optimization problems, IEEE Transactions of reliability, R-30, No. 2, 181-184.

Nesterenko, G. I., Nesterenko, B.G. 2009. Ensuring structural damage tolerance of Russian aircraft. International Journal of Fatigue, 31 (2009) 1054-1061.

Rama Mohan Rao, A., 2010. A meta-heuristic algorithm for multiobjective optimal design of hybrid laminate composite structures, Computer-Aided and Infrastructure engineering, 25 149-170.

Rama Mohan Rao, A. (2008). Development of a hybrid meta-heuristic algorithm for combinatorial optimization and its application for optimal design of laminated composite cylindrical skirt, Computers and Structures, 86, 796-815.

Rao, A. R., et al. 2007. Applying Multiobjective Cost and Weight Optimization to the Initial Design of Turbine Disks. Journal of Mechanical Design, Vol. 129, p. 1303-1310.

Rentizelas AA,(2010). Locating a bio-energy facility using a hybrid optimization method. Int J. Prod Econ 123:196-209.

Romero, C., Tamiz, M. and Jones, DF, "*Goal programming, compromise programming and reference point method formulations: linkages and utility interpretations*". Journal of the operational research society, Vol. 49, No. 9, pp. 986-991,1998

Rostegui Jr., M. A., “An empirical comparison of Tabu search, Simulated Annealing, and Genetic Algorithms for facilities location problems”. International Journal of Production economics, Vol. 103, pp. 742-754, 2006.

Ruiz, R., 2005. A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research, 165 479-494.

Sadeghi, J. et al, “A hybrid vendor managed inventory and redundancy allocation optimization problem in supply chain management: An NSGA-II with tuned parameters”. Computers & Operations Research, Vol 41, pp. 53-64, 2014.

Shaffer, J. D., 1985. Multiple objective optimization with vector evaluated genetic algorithms, Proc. First Int. Conf. on Genetic Algorithms, pp. 93-100. Lawrence Erlbaum.

Rostegui Jr., M. A., 2006. An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems. International Journal of Production economics, 103 742-754.

Schaffer, J. D. (1984). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, Ph.D. thesis, Vanderbilt University.

Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, In J. J. Gerfenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, pp. 93-100.

Sexton, R. S., (1999). Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research, 114 589-601.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms, Evolutionary computation 2(3), 221-248.

Suresh, P. S., et al. 2013. Optimal trends in Maneuver Load Control at subsonic and supersonic flight points for tailless delta wing aircraft. Aerospace Science and Technology, 24 (2013) 128-135.

Tan. K. C., 2001. Evolutionary Algorithms for Multiobjective Optimization: Performance Assessments and Comparisons. IEEE congress on Evolutionary Computation Seoul, Korea. May 27-30, 2001.

Teklu, F. (2007). A genetic algorithm for optimizing traffic control signals considering routing, Computer Aided civil and Infrastructure engineering, 22(1), 31-43.

Youssef, H., 2001. Evolutionary algorithms simulated annealing and search: a comparative study. Engineering Applications of Artificial Intelligence, 14 167-181.

Zhao, J., Adams, D., 2011. Challenges in damage tolerance approach for dynamic loaded rotorcraft components – From risk assessment to optimal inspection planning. ICAF 2011 Structural Integrity: Influence of Efficiency and Green Imperatives – Proceedings of the 26th Symposium of the International Committee on Aeronautical Fatigue, ICAF 2011, June 1, 2011 – June 3, 2011.

Ziaul, H.,et al. 2012. Optimization of Wind Turbine Airfoil using Nondominated Sorting Genetic Algorithm and Pareto Optimal Front. International Journal o Chemical Engineering, Vol 2012, Article ID 193021, 9 pages.

Zizler, E., 1999. Evolutionary algorithms for multiobjective optimization: Methods and applications", Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology, Zurich, Switzerland.

Zizler, E., 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary computation, 8(2):173-184.