A survey of multiobjective optimization in engineering design

Johan Andersson
Department of Mechanical Engineering
Linköping University
581 83 Linköping, Sweden
johan@ikp.liu.se

Technical Report: LiTH-IKP-R-1097

Abstract

Real world engineering design problems are usually characterized by the presence of many conflicting objectives. Therefore, it is natural to look at the engineering design problem as a multiobjective optimization problem. This report summarizes a survey of techniques to conduct multiobjective optimization in an engineering design context.

The report starts with discussing some of the difficulties of expressing the value of a deign and how to characterize different design variables. Thereafter we look more closely on the design problem in order to reformulate the design problem as a multiobjective optimization problem.

As engineering design problems often consist of a mixture of numerical simulations, analytical calculations and catalog selections, there is no easy way of calculating derivatives of the objectives function. Therefore, non-gradient optimization methods are better suited for these types of problems. Different types of non-gradient method are discussed in the report and different ways of developing hybrid methods are presented as well.

As most optimization problems are multiobjective to there nature, there are many methods available to tackle these kind of problems. Generally, a multiobjective optimization problem can be handled in four different ways depending on when the decision-maker articulates his or her preference on the different objectives; never, before, during or after the actual optimization procedure. The most common way is to aggregate the different objectives to one figure of merit by using a weighted sum and the conduct the actual optimization. There is however an abundance of other ways in which multiobjective optimization can be conducted, some of them are presented in this report.

Contents

1 Introduction

This survey focuses on different techniques to conduct multiobjective optimization in an engineering design context. The usage of optimization in engineering is getting larger every day as the computational capabilities of the computers are increasing. Today calculations could be performed in a fraction of the time it took just a couple of years ago. Therefore, the applications for numerical optimization have increased dramatically. A great part of the design process is and will always be intuitive, however analytical techniques as well as numerical optimization could be of great value and can permit vast improvement in designs.

Real engineering design problems are generally characterized by the presence of many often conflicting and incommensurable objectives. This raises the issue about how different objectives should be combined to yield a final solution. There is also the question on how to search for an optimal solution to the design problem. This paper presents a survey of some methods for optimization of engineering problems and different ways of developing hybrids among them. Another part of the paper focuses on how to handle the different objectives. Should the objectives first be aggregated to an overall figure of merit before we search for an optimal solution? Alternatively, shall we first search for a set of optimal solution before we articulate our preferences? There are many methods developed with different answers to these questions; some of them are discussed here.

First however we start with discussing some of the difficulties of expressing the value of a deign and how to characterize different design variables, using many ideas and definitions from Siddall [58]. Thereafter we look more closely on the design problem in order to formulate it as a multiobjective optimization problem.

1.1 The Concept of value

The concept of value is central to decision theory- the measure about what is good or desirable about a design. At a first glance, one would say that it is no problem. If two designs are comparable simply chose the cheapest one. However, consider the designing of a car; it must not only be cheap but safe, have a long life, be both quite and fast. How shall we then choose? Which characteristics contribute the most to the overall value of the design? This is very crucial to decision-making, and in general also to design.

For any given design, the designer has to give the different characteristics such as low initial cost, long life and good performance a weighting value. This is usually not done explicitly, but intuitively the designer does that. However, he might not be aware of it. During the design process, the designer much tradeoff characteristics against each other. How much is longer life worth in terms of higher manufacturing costs. One purpose of conducting multiobjective optimization is to make these tradeoffs visible. It would indeed be an interesting task to estimate what different ratings gave the final design.

Value is an inherit property of the design, which could be defined as that which satisfies desire. It remains however to be determined whose desires we should try to satisfy, and how we could articulate them to the designer. It might be hard to value a design even

when one has the physical artifact; it is even harder to do it in the earlier phases of the design process. However, in order to employ optimization to support the designer this is exactly what we have to do. Usually the designer employs a set of modeling and simulation tools in order to predict the properties of a design.

Often when we say value of a design we refer to the utility value which relates to the function or usefulness of the design. There is however many other values that the designer must take into account. Here however, we are just focusing on the function and usefulness of a design. This is without saying that the others are not important.

1.2 Design variables

Design variables are parameters that the designer might "adjust" in order to modify the artifact he is designing. There are many types of design variables.

Independent design variables are the actual quantities the designer deals with directly, such as geometry, material properties, production volume, surface finish, configuration of components, lubrication properties and many more. Independent design variables are usually called just design variables or design parameters. Here the term *design parameters* will be used.

Dependent variables are variables the designer can not directly assign values to but he works with them trough the design parameters. The dependent variables are usually named *characteristics* or *attributes* of the design. The value of a design is largely a function of the characteristics of the design. In optimization, the term objective function value corresponds to the value of a particular characteristic. An objective function is then the relation between the design parameters and the value of a particular characteristic. For a general design problem, it might be very difficult or even impossible to represent this relation analytically, as the characteristic might be the outcome of a complex simulation.

State variables are an intermediate type of design variables between dependent and independent design variables.

Operating variables are variables that can be changed after the design has been actually built. The *environmental variables* or the external variables are the environmental factors that affect the design when used. The designer has to determine the working conditions of the design in order to assess both the environmental and the operational variables.

The designer problem could be formulated as to assign values to the different design parameters in order to ensure that the state variables and the characteristics are as good as possible during a wide range of operating and environmental variables. This is indeed an intricate multiobjective optimization problem.

1.3 The general multi-objective optimization problem

A general multi-objective design problem is expressed by equations (1) and (2).

$$\min \mathbf{F}(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\right)^T \tag{1}$$

$$s.t.\ \mathbf{x} \in S$$

$$\mathbf{x} = (x_1, x_2, ..., x_n)^T \tag{2}$$

where $f_1(x), f_2(x), ..., f_k(x)$ are the k objectives functions, $(x_1, x_2, ..., x_n)$ are the n optimization parameters, and $S \in R^n$ is the solution or parameter space. Obtainable objective vectors, $\{\mathbf{F}(\mathbf{x}) | x \in S\}$ are denoted by Y, so $\mathbf{F} : S \mapsto Y$, S is mapped by $\mathbf{F}$ onto Y.

$Y \in R^k$ is usually referred to as the attribute space, where ∂Y is the boundary of Y. For a general design problem, $\mathbf{F}$ is non-linear and multi-modal, and S might be defined by non-linear constraints containing both continuous and discrete member variables.

$f_1^*, f_2^*, ..., f_k^*$ will be used to denote the individual minima of each respective objective function, and the utopian solution is defined as $\mathbf{F}^* = (f_1^*, f_2^*, ..., f_k^*)^T$. As $\mathbf{F}^*$ simultaneously minimizes all objectives, it is an ideal solution that is rarely feasible. Figure 1 provides a visualization of the nomenclature.

In this formulation, minimize $\mathbf{F}(\mathbf{x})$, lacks clear meaning as the set $\{\mathbf{F}(\mathbf{x})\}$ for all feasible $\mathbf{x}$ lacks a natural ordering, whenever $\mathbf{F}(\mathbf{x})$ is vector-valued. In order to determine whether $\mathbf{F}(\mathbf{x_1})$ is better then $\mathbf{F}(\mathbf{x_2})$, and thereby order the set $\{\mathbf{F}(\mathbf{x})\}$, the subjective judgment from a decision-maker is needed.

One property commonly considered as necessary for any candidate solution to the multiobjective problem is that the solution is not dominated. The Pareto set consists of solutions that are not dominated by any other solutions. A solution $\mathbf{x}$ is said to dominate $\mathbf{y}$ if $\mathbf{x}$ is better or equal to $\mathbf{y}$ in all attributes, and strictly better in at least one attribute. Considering a minimization problem and two solution vectors $\mathbf{x}$, $\mathbf{y} \in S$. $\mathbf{x}$ is said to dominate $\mathbf{y}$, denoted $\mathbf{x} \succ \mathbf{y}$, if:

$$\forall i \in \{1,2,...,k\}: f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \quad and \quad \exists j \in \{1,2,...,k\}: f_j(\mathbf{x}) < f_j(\mathbf{y}) \tag{3}$$

The space in R^k formed by the objective vectors of Pareto optimal solutions is known as the Pareto optimal frontier, $\mathcal{P}$. It is clear that any final design solution should preferably be a member of the Pareto optimal set. Pareto optimal solutions are also known as non-dominated or efficient solutions.

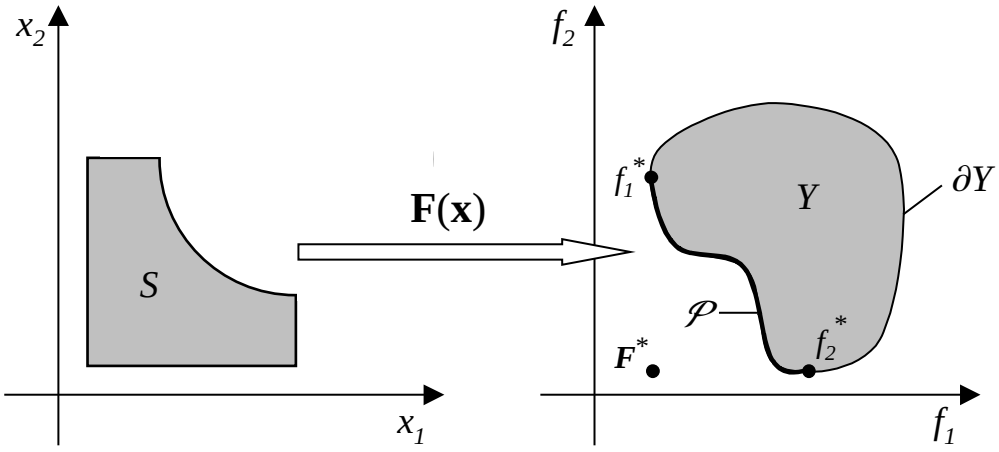

Figure 1: Parameter/solution and attribute space nomenclature for a two dimensional problem with two objectives.

If the final solution is selected from the set of Pareto optimal solutions, there would not exist any solutions that are better in all attributes. It is clear that any final design solution should preferably be a member of the Pareto optimal set. If the solution is not in the Pareto optimal set, it could be improved without degeneration in any of the objectives, and thus it is not a rational choice.

2 General optimization methods

Optimization methods could be divided into derivative and non-derivative methods, se Figure 2. This survey focuses on non-derivative methods, as they are more suitable for general engineering design problems. One reason is that non-derivative methods do not require any derivatives of the objective function in order to calculate the optimum. Therefore, they are also known as black box methods. Another advantages of these methods are that they are more likely to find a global optima, and not be stuck on local optima as gradient methods might do.

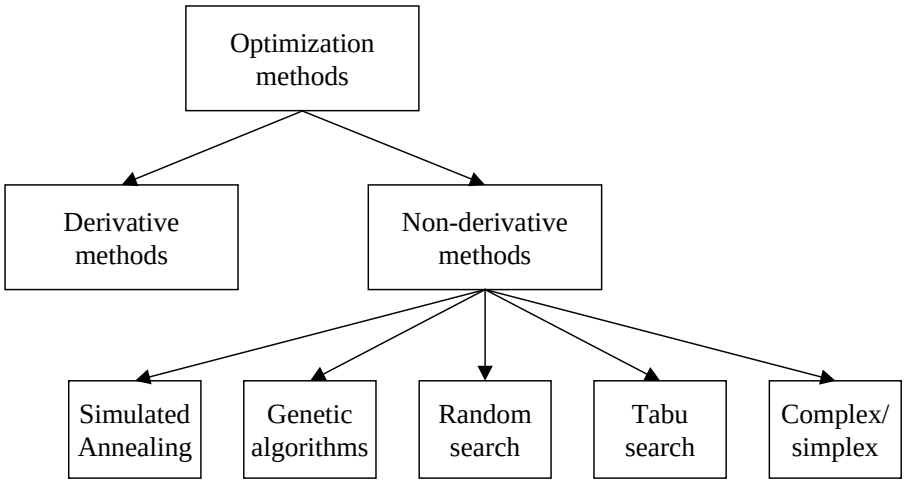

Figure 2: A classification of optimization methods in derivative and non-derivative methods with examples of some common non-derivative methods.

For a general design problem, it is hard to express your objective functions in terms of the design variables directly, as already mentioned. A variety of computer-aided engineering tools is available today, which allow the designer to estimate the performance of the design in an early stage of the design process. Such design tools might include CAD packages, FEM software, CFD solvers, self write simulation codes, commercial multi purpose modeling and simulation packages such as Matlab, spread sheet programs such as Excel, and many more. Typically, the objective function constitutes of a mixture of numerical simulations, analytical calculations as well as catalog selections among other things. Therefore, there is no straightforward way of calculating the derivatives of the different objective functions.

Apart from the methods mentioned in Figure 2, there are also other promising techniques to conduct engineering optimization, for instance neural network optimization and response surface approximations, as well as Taguchi methods and other statistical methods. However, here just the methods in Figure 2 would be discussed further.

2.1 Genetic algorithms

Genetic algorithms (GAs) and the closely related evolutionary strategies are a class of non-gradient methods which has grown in popularity ever since Rechenberg [52] and Holland [33] first published their work on the subject in the early 70's. For a more comprehensive study of genetic algorithms, see Goldbergs splendid book [25] on the subject.

The basic idea of GAs is the mechanics of natural selection. Each optimization parameter, (x_n), is coded into a gene as for example a real number or string of bits. The corresponding genes for all parameters, x_1,..x_n, form a chromosome, which describes each individual. A chromosome could be an array of real numbers, a binary string, a list of components in a database, all depending on the specific problem. Each individual represents a possible solution, and a set of individuals form a population. In a population, the fittest are selected for mating. Mating is performed by combining genes from different parents to produce a child, called a crossover. Finally the children are inserted into the population and the procedure starts over again, thus representing an artificial Darwinian environment, depicted in Figure 3 below. The optimization continues until the population has converged or the maximum number of generations has been reached.

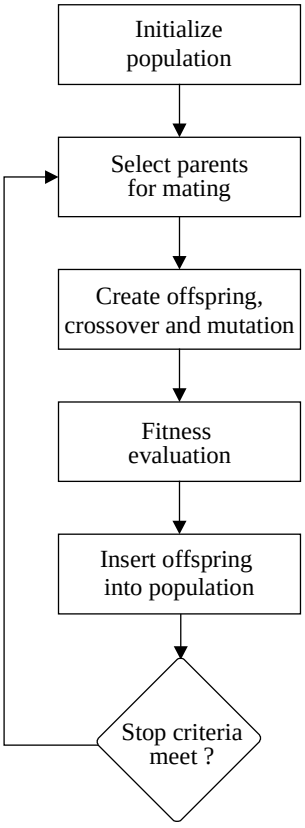

Figure 3: An algorithm for a simple genetic algorithm.

The popularity of genetic algorithms has grown tremendously under recent years and they have been applied to a wide range of engineering problems, see for example [6, 20, 37, 41, 42, 56]. There are also an abundance of different type of genetic algorithms such as simple GA's, steady state GA's, GA's with multiple populations, GA with crowding and sharing techniques, and many, many more, see [3] for a complete set of references**.** The different GA's all have different features in order to solve different type of problems. There are also a number of multiobjective genetic algorithms which aims at converging the population on the Pareto optimal front instead of on just one single optimal point. Different types of multiobjective genetic algorithms will be discussed further on.

GA's are very robust and can handle all type of fitness landscapes and mixture of real and discrete parameters as well as catalog selections, see Senin et al. [56].

2.2 Simulate annealing

Simulated annealing (SA) was first presented by Kirkpatrick [40] in the early 80's. SA simulates the natural phenomena of annealing of solids in order to optimize complex systems. Annealing of solids are accomplished by heating up a solid and allowing it to cold down slowly so that thermal equilibrium is maintained. This ensures that the atoms are obtaining a minimum energy state.

The algorithm starts with an initial design. New designs are then randomly generated in the neighborhood of the current design according to some algorithm. The change of objective function value, (ΔE), between the new and the current design is calculated as a

measure of the energy change of the system. If the new design is superior to the current design ($\Delta E<0$) it replaces it, and the procedure starts over again. If the new design is worse than the current, it might still be accepted according to the Boltzmann probability function (4).

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{T}\right) \tag{4}$$

P is the Boltzmann probability of accepting the new design and T is the current "temperature" of the system. A random number in the range {0..1} is generated, and if it is less than P the new design replaces the current design. The temperature is simply a control parameter in the same units as the objective function. As the annealing processes proceeds the temperature is decreased according to a cooling scheme. The temperature controls the probability that a worse design is accepted. This allows the algorithm to avoid local optima in the beginning of the search when the temperature is high. At the end of the search, when the temperature is low the probability of accepting worse designs is very low. Thus, the search converges to an optimal solution.

An advantage of SA is that it can handle mixed discrete and continues problems. The parameters settings for a SA algorithm determines how new solutions should be generate, the initial temperature and what the cooling scheme should be. Recent applications of SA include [10, 11, 57].

2.3 The Complex method

The Complex method was first presented by Box [7], and later improved by Guin [27]. The method is a constraint simplex method developed from the Simplex method by Spendley et al [59] and Nelder Mead [50]. Similar related methods goes under names such as Nelder-Mead Simplex and flexible polyhedron search. These methods also have similar properties.

In the Complex method, a complex consisting of several possible problem solutions (sets of design parameters) is manipulated. Each set of parameters represents one single point in the solution space. Typically, the complex constitutes of twice as many points as the number of optimization parameters. The main idea of this algorithm is to replace the worst point by a new and better point. The new point is calculated as the reflection of the worst point through the centeroid of the remaining points in the complex. By varying the reflection distance from the centeroid it is possible for the complex to expand and contract depending on the topology of the objective function. The starting points are generated randomly and it is checked that both the implicit and the explicit constraints are fulfilled. The optimal solution is found when all points in the complex have converged.

An example of the complex method is shown in Figure 4 below for a two dimensional parameter space. The circles in the graph indicate the objective function value for different solutions, with the best value in the middle.

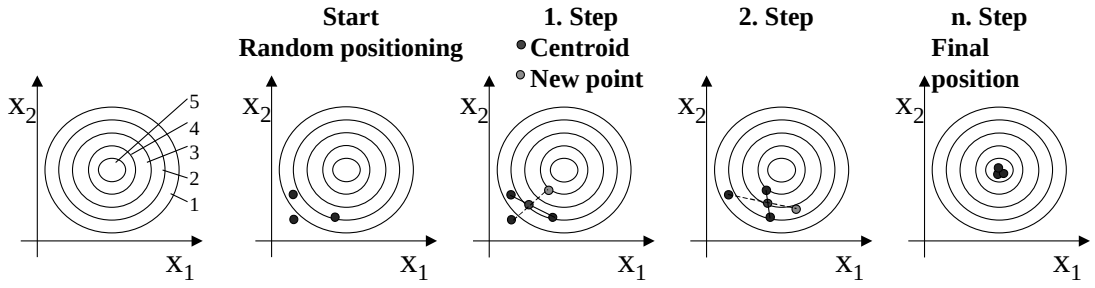

Figure 4: The progress of the Complex method for a two dimensional example, with the optimum located in the middle of the circles.

The complex method has been applied to a wide range of problems such as physics [54], structural engineering [30], fluid power system design [1, 37], aerospace engineering [42], and many others [23, 45, 46]. The Complex method was originally developed for problems with continues variables but Haque [30] has shown that the complex method could also be applied to mixed continues and discrete variable problems.

2.4 Random search

Random search method is a generic term for methods that rely on random numbers to explore the search space. Random search methods are generally easy to implement, and depending on the implementation they can handle mixed continues and discrete problems. However, they usually shows quit slow convergence. Here just one among many methods is described.

2.4.1 Random walk

Random walk is an iterative procedure where the next point $\mathbf{x}^{t+1}$ is calculated as

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} + \alpha \cdot \mathbf{S} \tag{5}$$

where α is a step size parameter and $\mathbf{S}$ is a random generated unit vector in witch to search. If $\mathbf{x}^{t+1}$ has a better function value than $\mathbf{x}^{t}$, $\mathbf{x}^{t+1}$ replaces $\mathbf{x}^{t}$. Otherwise a new search direction $\mathbf{S}$ is generated until a better solution is found. If no better solution can be found the step size is reduced and new search directions is generated. The search is considered to be converged when no better solutions can be found and the step size is reduced beyond a prescribed value.

2.5 Tabu search

Tabu search (TS) is a recent addition to non-derivative optimization algorithms developed by Glover [24] in 1989. TS is an adaptive heuristic strategy that was primarily design for combinatorial optimization. It has been applied to a wide range of problems however, mostly of combinatorial nature.

In the Tabu search method is flexible memory cycles (tabu lists) are used to control the search. At each iteration we take the best move possible that is not tabu, even if it means an increase in objective function value. The idea is that when we reach a local minimum we wish to escape via a different path.

A short term memory is implemented as a list of previously visited solutions that are classed as tabu. Whilst a solution is contained within thetabu-list it cannot be returned to.

The tabu restrictions stop the search from returning to the local optima and the new search trajectory ensures that new regions of the search space are explored and the global optimum located. Intermediate and long term memory cycles are used to intensify and diversify the search to ensure that the entire solution space is adequately explored before the search is terminated. The direction and length of each movement are calculated by employing local hill climbing techniques.

Tabu search can handled both continues and discrete parameters. Recent applications of TS include [5, 14, 32].

2.6 Hybrid methods

Clearly the different methods have different advantages and it is therefor attractive to produce hybrid methods. Yen et al. [70] classified hybrid genetic algorithms in four categories, which are suitable for classifying other hybrids as well.

1. Pipelining hybrids
2. Asynchronous hybrids
3. Hierarchical hybrids
4. Additional operators

2.6.1 Pipelining hybrids

The simplest and most straightforward way of implementing hybrids is to do in sequentially. First, one starts with exploring the whole search space with a method that is like to identify global optima but perhaps with slow convergence. After identifying promising regions, one could switch to a method with higher convergence rate in order to speed up the search. This could be accomplished by combining for instance a GA with a gradient-based algorithm, see [69].

2.6.2 Asynchronous hybrids

In asynchronous hybrids different methods work on different subsets of the solution space. The different methods might work on a subset of a shared population for some iterations. If the individuals from such a subset outperform the ones in the shared population, they are allowed to immigrate into it. With this approach, a method that converges slowly could be combined with one that converges faster, or a method that performs well on one subset of the search space could be combined with method that performs well on another subset. The same idea is employed in genetic algorithms with multiple populations, then in order to find multiple solutions in multi modal spaces.

2.6.3 Hierarchical hybrids

In hierarchical hybrids, different optimization methods are applied at different levels of the optimization. On an overall level, you can apply a robust optimization strategy to find

an optimal layout. At lower levels, where the system might be less complex and sensitive it might more appropriate to employ for instance pure analytical methods.

2.6.4 Additional operators

In the literature the are many examples of hybrids between different optimization methods where operators from one method are added to or even replacing the standard operators of the other method. For instance, Yen et al. [70] has introduced a simplex method as a new way of generating children in a genetic algorithm. At each iteration a certain percentage of the population is generated by employing a simplex method on group of promising individuals. The rest of the population is generated using the usual genetic operators. This has improved both the convergence speed as well as the ability to find the absolute optima.

There are many additional hybrids of SA and GA algorithms for instance [71] where among other things the replacement is done according to a simulated annealing scheme. The thermodynamic genetic algorithm [48] is also an example of such a hybrid.

Gunel and Yazgan [28] have presented a method where they combine a random search strategy with the complex method. The basic idea of this method is to modify the reflection strategy of the complex method. If the new reflected point is still the worst point it is not moved against the centeroid as in the normal Complex, but a random strategy is employed in order to generate the new point.

2.7 Short notes on comparisons of the different methods

There is no simple answer to which optimization methods is the best for any given problem. It is all a matter of opinion; very much depending on the nature of the problem and the availability of different optimization software that fits the problem statement.

In most comparison studies different methods come out on top depending on the problem and how well the different methods have been tuned to fit that particular problem. Comparative studies of different types of non-derivative methods could be found in for instance [6, 29, 37, 47]. An interesting question that one should keep in mind when comparing different methods are the time spent on optimizing the different methods before they are compared. If a method is five percent faster then another one, but takes three times as long to implement and parameterize, it might not be worth the effort.

GA's seems to be most suitable to handle multi modal function landscapes and to identify multiple optima in a robust manner. GA:s are however associated with a high computational cost. Moreover, GA:s are more complicated and harder to implement and parameterize then the other methods. This is however compensated for by the huge number of GA software that are available in almost any programming language. As GA's have be around for such a long time they also have the broadest field of applications.

Simulated annealing and Tabu are growing in popularity and gaining ground on GA's mostly on combinatorial optimization problems. SA could actually be seen as a subset of GA's with a population of one individual and a changing mutation rate. Both SA and Tabu search are robust methods slightly less computational expensive then genetic algorithms.

In order shorten the computation time the methods could be implemented on parallel CPU:s. This have been successfully implemented for both genetic algorithms as well as simulated annealing.

The Complex method and other related methods are very fast, but not as robust as the other methods. Robust then referring to that they are more likely to get stuck in local optima. Moreover, they can just find one optimal solution. However, the complex method is easy to implement and understand and to parameterize, which makes it very user-friendly. Studies like [1, 6, 37] shows that these types of methods could be very promising for engineering optimization. In [6] the flexible polyhedron search comes out on top in a comparison with the other methods.

3 Different ways to perform multiobjective optimization

As mentioned earlier real engineering design problems are usually characterized by the presence of many conflicting objectives that the design has to fulfill. Therefore, it is natural to look at the engineering design problem as a multiobjective optimization problem (MOOP). References to multiobjective optimization could be found in [36, 61] and with engineering applications in [17, 51].

As most optimization problems are multi objective to there nature, there are many methods available to tackle these kind of problems. Generally, the MOOP can be handled in four different ways depending on when the decision-maker articulates his or her preference on the different objectives, never, before, during or after the actual optimization procedure. These different possibilities are shown in Figure 5.

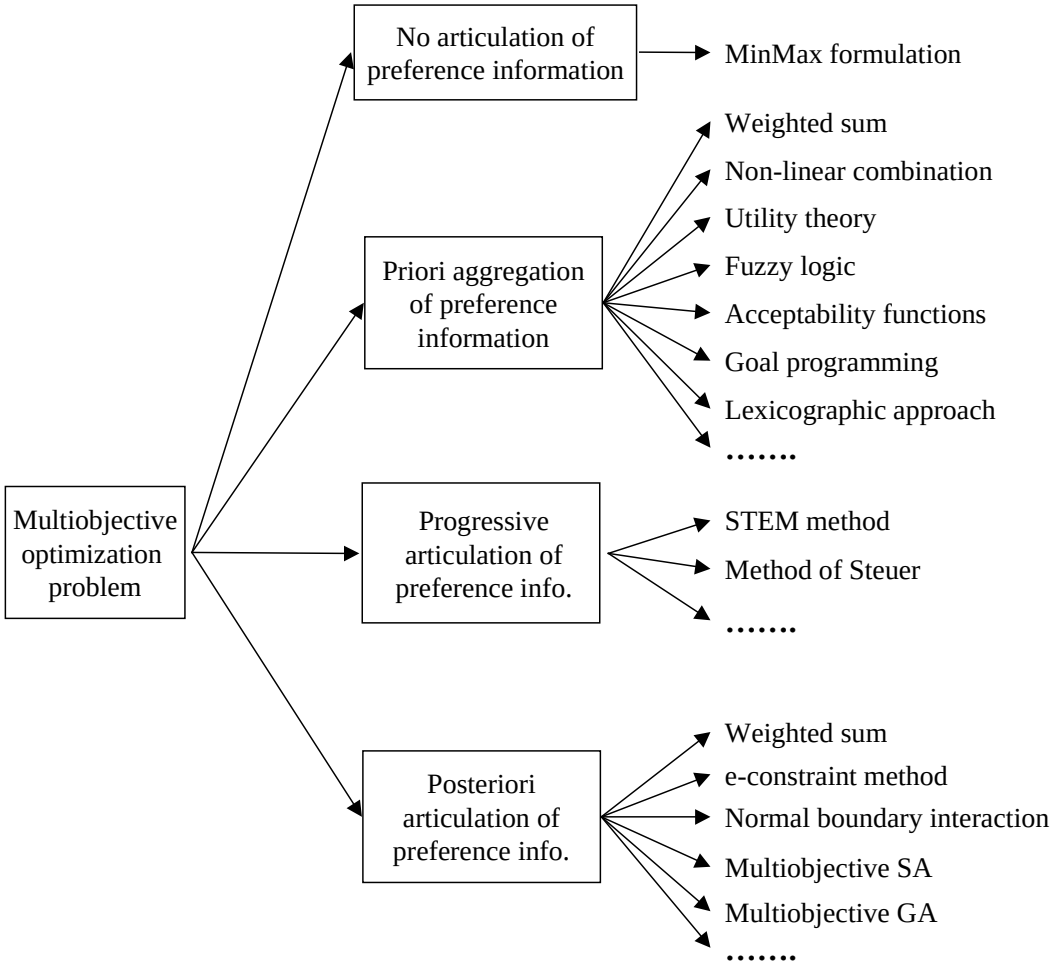

Figure 5: A classification of some methods for multiobjective optimization.

Although the classification gives a far from complete description of all available optimization techniques it constitutes a good frame work for discussing the most common methods suitable for engineering optimization.

3.1 No preference articulation

These types of methods do not use any preference information. Examples are the Min-Max formulation and global criterion method [36, 51, 61].

The Min-Max formulation is based on minimization of the relative distance from a candidate solution to the utopian solution $\mathbf{F}^*$, see Figure 1. The distance between a solution vector and the utopian vector is typically expressed as a $\mathcal{L}_p$-norm. Thus, the optimization problem is formulated according to equation (6).

$$\min\left[\sum_{j=1}^{k}\left(\frac{f_j(\mathbf{x})-f_j^*}{f_j^*}\right)^p\right]^{\frac{1}{p}} \qquad (6)$$

$$s.t.\quad \mathbf{x}\in S$$

$$1\le p\le\infty$$

The exponent p gives different ways of calculating the distance. The most frequently used values for p are 1 for the simples formulation, 2 for the Euclidean distance, and ∞ for Tchebycheff norm.

In the min-max formulation, no preference information from the decision-maker is necessary. However, the output is just one point on the Pareto front, which the DM has to accept as the final solution. By changing the exponent in the distance formulation and by giving the single objectives different weightings, different points on the Pareto front could be found. However, then preference information from the decision-maker is needed.

This formulation is not widely used in engineering design. For interactive methods however, Min-Max formation might be used together with other techniques to find multiple points on the Pareto front.

3.2 Priori articulation of preference information

The most common way of conducting multiobjective optimization is by priori articulation of the decision makers preferences. This means that before the actual optimization is conducted the different objectives are some how aggregated to one single figure of merit. This can be done in many ways; some of which are described here.

3.2.1 Weighted-sum approaches

The most easy and perhaps most widely used method is the weighted-sum approach. The objective function is formulated as a weighted $\mathcal{L}_1$-metric, see equation (7). For an interesting discussion on weighted sum approaches, see Steuer [61].

$$\min\sum_{j=1}^{k}\lambda_j f_j(\mathbf{x}) \qquad (7)$$

$$s.t.\quad \mathbf{x}\in S$$

$$\lambda\in R^k\,\big|\,\lambda_i>0,\sum\lambda_i=1$$

By choosing different weightings, λ_i, for the different objectives, the preference of the decision-maker is taken into account. As the objective functions are generally of different magnitudes, they might have to be normalized first. Although the formulation is simple, the method is somewhat ad-hoc, as there is no clear relation between the weightings and the obtained solution. How to determine the weightings from the decision-makers preferences is also an ad-hoc procedure. Another drawback for this method is that with convex combinations of the different objectives, as in (7), it is not possible to locate

solutions at non-convex part of the Pareto-front. The weighted-sum approaches are discussed later when this approach is employed in order to sample multiple points on the Pareto optimal front.

3.2.2 Non-linear approaches

Although many methods might be referred to as non-linear, hence all the ones mentioned hereafter, we here refer to higher order of $\mathcal{L}$p-metrics formulations and their equals. Equation (8) below represents one such approach used by for instance [2, 43].

$$\min \sum_{j=1}^{k} \left(\frac{f_j(\mathbf{x})}{f_{j0}} \right)^p \qquad (8)$$

$$s.t. \quad \mathbf{x} \in S$$

In this formulation each objective are normalized by f_{j0}, which represent the value of *j:th* objective for the best known solution so far. The expression could be further normalized by dividing with *k*, yielding unity as the value for the present best solution. For consistence the whole expression could be raised to the power of *1/p*. The exponent *p* expresses how much an improvement in f_i is worth and how much a poorer value penalizes the over all objective function. The graph in Figure 6 below depicts $(f_i / f_{i0})^p$ as a function of f_i. The graph could be looked upon as an inverted utility function.

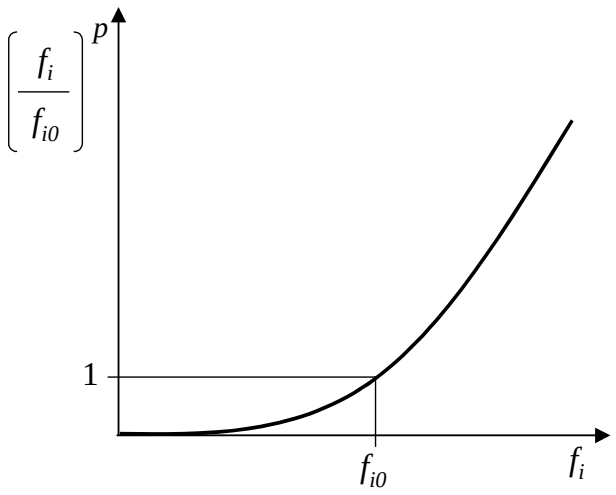

Figure 6: The form $(f_i / f_{i0})^p$ *as a function of* f_i for *p=3.*

Anderson et al [2] employed this formulation as well as the House of Quality method in order to capture the preferences of a set of decision-makers.

3.2.3 Fuzzy logic approaches

The concept of fuzzy sets is based on a multi-valued logic where a statement could be simultaneously partly true and partly false. In fuzzy logic, a membership function μ expresses the degree of truthfulness of a statement, in the range from $\mu=0$, indicating that

the statement is false to $\mu=1$ for truth. This is in opposite to binary logic where a statement can be only false or true.

In an optimization problem the membership function enables us to associate a normalized value to each objective $\mu_i(f_i(\mathbf{x}))$, which expresses the degree of satisfaction of the considered objective i. The value of $f_i(\mathbf{x})$ is fuzzified by μ_i to yield a value in the rang {0,1}, which quantifies how well a solution satisfies the requirements. Examples of two membership functions are depicted in Figure 7 below.

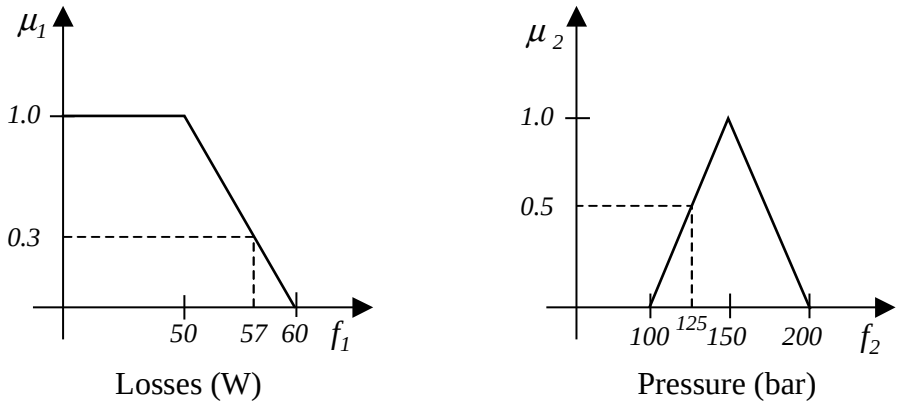

Figure 7: Typical member functions for two characteristics of a fluid power system.

In Figure 7 we consider a fluid power system design, where we want to keep the losses low, say 50W is acceptable whereas 60W is not. We also want to have a constant system pressure of 150 bar. The corresponding membership functions could then look like in Figure 7. A membership function can have any shape. However, in the literature piecewise linear functions are the most common.

Ones the fuzzification has been performed the actual value of each objectives have been transformed into logical values. These values have to be aggregated to one in order to get an overall value for the design. In binary logic this is accomplished by the AND operator. However, in fuzzy logic the AND operator could be implemented by several different rules. The most common ones are the min operator and the product operator. The min operator returns as an output the minimum value of the μ_i on which is operates. As in binary logic, if one μ_i equals zero the output is zero. The product operator returns the product of all individual operators. This formulation is also compatible with the one of binary logic. The overall objective function could consequently be expressed as:

$$F_{fuzzy}(\mathbf{x}) = \min(\mu_1(f_1(\mathbf{x})), \mu_2(f_2(\mathbf{x})), \ldots, \mu_k(f_k(\mathbf{x}))) \text{ or} \tag{9}$$

$$F_{fuzzy}(\mathbf{x}) = \prod_{i=1}^{k} \mu_i(f_i(\mathbf{x})) \tag{10}$$

The overall fuzzy optimization problem is formulated according to equation (11).

$$\max \; F_{fuzzy}(\mathbf{x}) \tag{11}$$

$$s.t. \; \mathbf{x} \in S$$

Examples of fuzzy approaches to multiobjective optimization could be found in [11, 12, 72].

3.2.4 Utility theory

Utility theory forms the basics of decision making and dates back to von Neumann and Morgenstern (1947), although the basic reference can be found in [38]. The utility function $U(f_1(\mathbf{x}), f_2(\mathbf{x}),..., f_i(\mathbf{x})) = U(\mathbf{F}(\mathbf{x}))$ is a function that maps the value of the design as a scalar, i.e. $U : R^k \mapsto R$. $U_i(f_i(\mathbf{x}))$ expresses the value of the design as a function of the attribute expressed by $f_i(\mathbf{x})$ as shown in Figure 8. To the left the utility, and thereby the value, of the design increases as the characteristic increases.

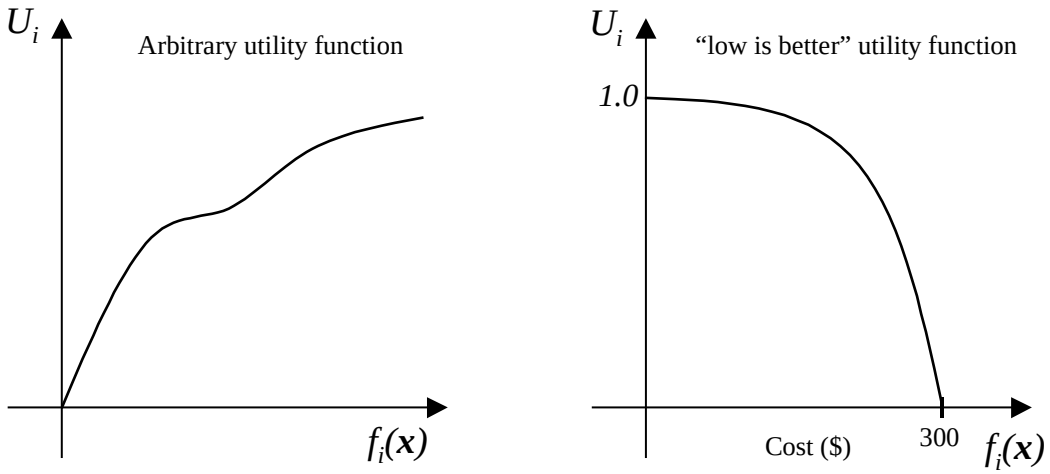

Figure 8: Typical utility functions.

Utility functions are usually expressed as exponential functions, such as $U_j(f_j(\mathbf{x})) = a - be^{cf_j(\mathbf{x})}$. The constants a and b are usually chosen so that $U_j(f_{jmin})=1$ and $U_j(f_{jmax})=0$, see Figure 8 right.

An extensive interaction with the decision-maker is needed in order to determine the utility function for each attribute. A formal method that allows this is presented in [66].

However, in order achieve an overall utility function a set of assumptions has to be made. For instance, it is usually assumed that the different utilities are mutually independent. In order to aggregate the different objectives it is usually assumed the utility functions are either additive or multiplicative. Simplified this could be expressed as $U(\mathbf{F}(\mathbf{x})) = \sum_{i=1}^{k} U_i(f_i(\mathbf{x}))$ or $U(\mathbf{F}(\mathbf{x})) = \prod_{i=1}^{k} U_i(f_i(\mathbf{x}))$. Ones the overall utility function is formulated the optimization problem is formulated as to maximize that utility. If $U(\mathbf{F}(\mathbf{x}))$ really captures how the decision-maker values a design, a maximization of the overall utility would yield the best solution according to the decision-maker.

Utility Theory is mathematically very rigorous. Ones the "right" utility functions has been determined and the assumption holds, it could be guaranteed that the solution found is the one with highest value to the decision-maker. However, deriving the individual

utility functions and aggregating the overall utility function is hard for a simple problem, and for a complex multi-attribute problem, it might even be impossible. There are however examples on how utility theory is employed to solve design problems, [67].

Many of the other methods are related to utility theory, as they try to find easier ways of expressing the utility functions. For instance, the weights in the weighted-sum approach could be seen as the local gradients of the utility function. The non-linear formulation constitutes a simple utility function, and acceptability functions are a convenient way of expressing utility type functions adopted to suit engineering design problems.

3.2.5 Acceptability functions

The method of acceptability functions has been developed by Wallace et al [68] and Kim and Wallace [39]. The method is a goal-oriented design evaluation model that employees the same goals and targets that are commonly used in engineering design to evaluate the performance of each solution.

The acceptability function represents the subjective probability that a designer will accept a design based upon each objective. This is explained in Figure 9, where the acceptability function for a car engine is expressed as function of the fuel consumption.

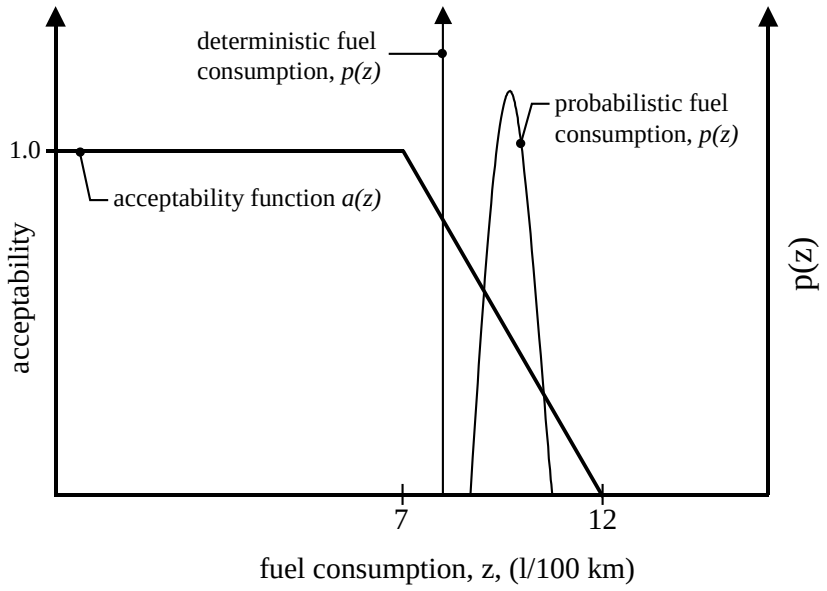

Figure 9: Acceptability function as well as probability density functions for an engine design problem.

a(z) is the acceptability function which defines the probability that different levels of the performance characteristic *z* will be acceptable. A car engine with a fuel consumption less than 6 liter/100 km have a probability of 1.0 of being accepted, whereas a design with a fuel consumption of 12 liter/100 km is surely rejected. The function *p(z)* is a probability density function of unit area which quantifies the design's performance for the characteristic *z*. This formulation allows the designer to quantify a design's performance either deterministically or probabilistically. In the deterministic case the probability density function is an infinite spike, with the area equals to one. The probability, P_i, of

accepting the design based upon the i:th characteristic is expressed in (12), and the overall probability of accepting the design based on all objectives are calculated by multiplying the individual probabilities (13).

$$P_i = \int_x a(z)\,p(z)dz \tag{12}$$

$$P_{acc} = \prod_{i=1}^{k} P_i \tag{13}$$

Based on these equations the optimization problem is formulated according to equation (14) below.

$$\max \; P_{acc}(\mathbf{x}) \tag{14}$$
$$s.t. \;\; \mathbf{x} \in S$$

This formulation has been successfully employed in a number of studies.

The outcome of utility function, fuzzy logic and acceptability function formulations may look very much alike. However, they have different theoretical backgrounds. Utility functions are based on a very rigorous utility theory, acceptability functions are based on utility theory and probability in contradiction to the fuzzy logic approach. It seems like acceptability functions more closely relate to the way design problems are usually formulated and it offers a structured way of handle uncertainties in design.

3.2.6 Goal programming

Goal programming dates back to the early sixties [8, 9], however more recent references could be found in [8, 9, 61, 65]. In goal programming (GP) the objectives are formulated as goal criteria that the decision maker wants each objective to possess. The criteria could be formulated in the following ways. We want the objective to be:

1. Greater than or equal to
2. Less than or equal to
3. Equal to
4. In the range of

Usually a point that satisfies all goals is not a feasible solution. The goal programming problem is to find the point in *S* whose criterion vector “best” compares with the utopian set, i.e. has the smallest deviation from the utopian solution. Different types of goal programming methods uses different ways of determine witch point in *S* compares best with the utopian solution set.

The Archimedean GP uses a weighted $\mathcal{L}_1$-metric to determine the “best” solution. The objective function is thus reformulated as a weighted sum of undesired deviations from the utopian solution.

In lexicographic GP or preemptive GP, the goals are grouped according to priorities. In the first stage a set of solutions that minimize the deviation to the goal with highest priority is obtained. Then in the second stage this set of solution is searched to find a

subset that minimizes the deviation from the second most important goal. The process continues until only one single point is left as the final solution.

Although the idea behind goal programming is very attractive, to minimize the deviation from an utopian solution set, there are a set of problems marred to GP. If the initial set of weights in the Archimedean approach does not yield a satisfying solution, they weights has to be changed in order to yield a better solution. There is however no clear relation between the weights and the solution found as you might end up in new vertices in both the solution space *S,* as well as in the utopian set. In lexicographic GP lower priority goals might have no influence of the generated solution, as a single point might be found without having to consider all goal criteria.

3.2.7 Lexicographic approaches

In lexicographic approaches, the decision-maker determines an order in which the objectives have to be optimized. Like in a dictionary where A precedes B, the decision-maker determines that objective *i* precede objective *j*. This implies that solutions are ordered by first evaluating them based on the foremost objective. If a set of solutions have comparable values in the foremost objective, the comparison continues on lower level objectives until the solutions can be distinguished. The disadvantage with lexicographic approaches is that not all objectives might be considered. Lexicographic methods are not so commonly used by them self in engineering design, but jointly with other techniques, such as in goal programming or as a part of a selection mechanism in genetic algorithms.

3.3 Progressive articulation of preference information

This class of methods is generally referred to as interactive methods. They rely on progressive information about the decision-makers (DM) preferences simultaneously as they search through the solution space. Interactive methods are very common within the field of operations research.

These methods works according to the hypothesis that the DM is unable indicate ‘a priori’ preferences information because the complexity of the problem. However, the DM is able to give some preference information as the search moves on. The DM then learns about the problem as he/she faces different possible problem solutions. Advantages of these types of methods are:

- there is no need for ‘a priori’ preference information,
- only local preference information is needed,
- it is a learning process where the DM gets a better understanding of the problem,
- as the DM takes an active part in the search it is more likely that he/she accepts the final solution.

The disadvantages are:

- The solutions are depending on how well the DM can articulate his preferences
- A high effort is required from the DM during the whole search process.

- The solution is depending on the preferences of one DM. If the DM changes his preferences or if there is a change of DM, the process has to be restarted.
- The required computational effort is higher then in the previous methods.

These methods usually progresses by changing weights in different type of $\mathcal{L}_p$-norm formulations, by progressively reducing the search space or by changing the search direction based on input from the decision-maker. Many of these methods are unsuitable as they build on assumptions on linearity and differentiability of the objective and constraint functions. Here two examples are discussed in some more detail.

3.3.1 STEM Method

The STEM-Method or STEP-method was first presented by Benayoun et al [4]. In STEM and related methods, preference information from the DM is used to reduce the solution space successively. The general optimization problem is reformulated as a $\mathcal{L}_p$-norm problem (min-max formulation), with bounded and weighted objectives.

$$\begin{aligned} &\min\left\{\sum_{i=1}^{k}\left(w_i^h\left(f_i(\mathbf{x})-f_i^*\right)\right)^p\right\}^{\frac{1}{p}} \\ &s.t.\quad \mathbf{x}\in S^h \\ &\qquad w_i^h>0,\ \sum_{i=1}^{k} w_i^1=1 \end{aligned} \tag{15}$$

h is the iteration counter, and p is the parameter in the $\mathcal{L}_p$-norm, usually equaling 1 or ∞. The weights are needed in order to solve the min-max formulation and to equalize the magnitude of the different objectives. The weights are not crucial to the outcome of the optimization as the final solution is obtained by means of bounds on the objective rather then variation of weightings. In the literature methods of calculating the weights are given. The problem is solved resulting in an objective vector $\tilde{f}$. $\tilde{f}$ is compared with the ideal solution $\mathbf{F}^*$. If some components of $\tilde{f}$ are acceptable but some are not, the decision-maker must decide on a relaxation on at least on of the objectives. This means that the upper bound for the j:th objective are adjusted to $\tilde{f}_j + \Delta f_j$. The solution space S^{h+1} is reduced by the new constraint $f_j \le \tilde{f}_j + \Delta f_j$. The weighting of the j:th objective is set to zero and the optimization problem of is solved again, this time in the reduced solution space. After the second iteration, the decision-maker might be satisfied with the obtained solution, or he/she has to relax the boundaries of another function and start over again. Thus, the algorithm proceeds through progressively reducing the solution space by introducing new constraints on the different objectives.

3.3.2 Steuer method

There are a set of methods that sample a progressively smaller subset of the non-dominated set by employing progressively changing weights in weighted sum approaches. Here Steuer and Choo [62] exemplifies these types of methods.

The algorithm converges in a predetermined number of iterations *t*. In each iteration the decision-maker has to chose between *P* alternative non-dominated solutions. Typically, the number of iterations *t* equals to the number of objectives *k*, and $P \gtrsim k$. First we obtain the ideal solution $\mathbf{F}^*$. The algorithm progresses by minimizing a weighted Tchebycheff metric of the distance between a set of proposed solutions and the ideal solution. For the weightings, a dispersed set of weighting vectors $\lambda \in \Lambda$, are employed.

$$\Lambda = \left\{ \lambda \in R^k \middle| \lambda_i \in \left[l_i^h, u_i^h \right], \sum_{i=1}^{k} \lambda_i = 1 \right\} \tag{16}$$

During the progress of the algorithm the iteration counter *h* increases, and the range $\left[l_i^h, u_i^h \right]$ in which the weighting vector λ is allowed to vary is decreased. This leads the algorithm to focus in on a subset of the non-dominated set. The decision-maker interacts with the algorithm to guide it towards a portion of non-dominated set that he/she prefers. The steps below outlines the algorithm.

Step 1: Find the ideal solution **F***, set *h=0*.

Step 2: Generate $50 \cdot k$ λ vectors, and filter them to obtain the $2 \cdot P$ most diverse vectors.

Step 3: Solve the $2 \cdot P$ weighted Tchebycheff problems to obtain *2P* non-dominant solutions.

Step 4: Filter them to achieve the *P* most diverse solutions.

Step 5: Present the decision-maker with the *P* solutions and let him/her decide on one.

Step 6: Contract the intervals for the weighting vector components, based on the preferred solution.

Step 7: Repeat step 2-6 until *h=t*.

3.4 Posteriori articulation of preference information

There are a number of techniques which enables to first search the solution space for a set of Pareto optimal solutions and present them to the decision-maker. The big advantages with these type of methods is that the solution is independent of the DM's preferences. The analysis has only to be performed ones, as the Pareto set would not change as long as the problem description are unchanged. However, some of these methods suffer from a large computational burden. Another disadvantage might be that the DM has too many solutions to choose from. There is however methods that supports in screening the Pareto set in order to cluster optimal solutions, see [49, 53].

In the following, a set of approaches is presented which are conceivable for engineering design problems.

3.4.1 Multiple run approaches

This section discusses the most common approaches to obtain a sample set of points of the Pareto-optimal front. By sampling a set of discrete points on the Pareto front the decision maker could get a feeling for the form of the front and thereby the possible trade-off between objectives. Excluded are methods that require the objective function to be differentiable.

3.4.1.1 Weighted sum approaches

In the weighted sum approaches the optimization problem is formulated according to equation(17).

$$\begin{aligned} &\min \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \\ &s.t. \quad \mathbf{x} \in S \\ &\qquad \lambda \in \Lambda \\ &\Lambda = \left\{ \lambda \in R^k \middle| \lambda_i \geq 0 , \sum_{i=1}^{k} \lambda_i = 1 \right\} \end{aligned} \tag{17}$$

Multiple optimization runs are conducted with different weighting vectors λ in order to locate multiple points on the Pareto front. This method is the simplest and most straightforward way of obtaining multiple points on the Pareto-optimal front. However, this method is associated with some mayor drawbacks. Depending on the scaling of the different objectives and the shape of the Pareto front, it is hard to select the weightings to ensure that the points are spread evenly on the Pareto front [16]. Another problem occurs when the solution space is non-convex. In that case can not all the Pareto-optimal solutions be obtained by solving equation (17) for any $\lambda \in \Lambda$. For a more detailed description of this problem se Steuer [61].

In order to be able to locate points on non-convex parts of the Pareto front the weighted $\mathcal{L}_p$-norm problem could be solve instead. The weighted $\mathcal{L}_p$-norm problem is actually a generalization of the weighted sum formulation, see equation (18) below.

$$\begin{aligned} &\min \left\{ \sum_{i=1}^{k} (\lambda_i f_i(\mathbf{x}))^p \right\}^{\frac{1}{p}} \\ &s.t. \quad \mathbf{x} \in S \\ &\qquad \lambda \in \Lambda \end{aligned} \tag{18}$$

p is an integer satisfying $1 \leq p \leq \infty$. With an appropriate value on p all Pareto optimal points could be obtained. However, such a value for p is unknown in advance. Moreover, a high value of p increases the difficulty of solving the optimization problem. In the extreme case where $p = \infty$ the problem is known as the weighted minmax formulation.

3.4.1.2 e-constraint approach

In the e-constraint method one objective i, is selected for optimization and the others are reformulated as constraints, i.e.

$$\begin{aligned} &min \left\{ f_i(\mathbf{x}) \right\} \\ &s.t. \quad f_j \leq e_j , \ j \neq i , \ j = 1...k \\ &\mathbf{x} \in S \end{aligned} \tag{19}$$

By progressively changing the constraint values, e_j different points on the Pareto-front could be sampled. By calculating the extremes of the Pareto-front the range of different

objective functions could be calculated and constraint values selected accordingly. The method enables an even spread on the Pareto-front as long as the Pareto-front is continues, i.e. that there exist a Pareto-optimal solution with $f_j=e_j$.

3.4.1.3 Normal boundary interaction

Normal-boundary interaction (NBI) is presented in [15]. It is assumed that the global minima of the objectives, F*, is known. First, the convex hull of the individual minima (CHIM) is established. In the two-dimensional case, this is the line connecting the individual minima. The basic idea of the algorithm is to find the intersection of the boundary of the feasible set and the normal to the CHIM. Figure 10 makes this statement more clear.

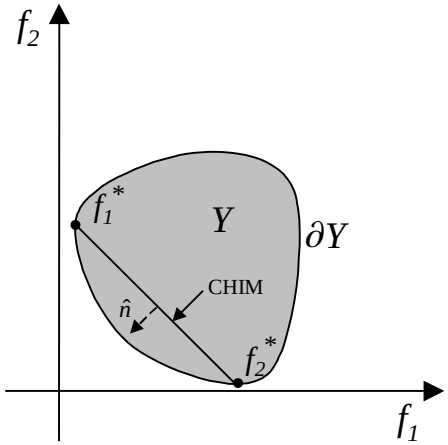

Figure 10: The basic idea behind normal-boundary interaction.

The algorithm enables an easy way of establishing an evenly spread set of points on the Pareto front, by starting the search from points evenly spread on the CHIM. However, if the Pareto front has a very complex shape the method might identify non-Pareto optimal solutions as well as local Pareto optimal solutions.

3.4.1.4 Multiobjective simulated annealing

There have been some work done in multiobjective simulated annealing [63] where the non-dominated solutions found so far in search are stored in a separate array. For each new point that are examined the set of non-dominated points is updated to include the new point if it is non-dominated and to exclude the points that the new one dominates. However, this is quit a new field where lot of research still has to be done. The same principle might be applicable to other methods as well, as for instance Tabu-search.

3.4.2 Multiobjective genetic algorithms

Lately there has been a large development of different types of multiobjective genetic algorithms, which is also reflected in the literature. The big advantage of genetic algorithms over other methods is that a GA manipulates a population of individuals. It is therefore tempting to develop a strategy in which the population captures the whole Pareto front in one single optimization run. For an overview on genetic algorithms in

multiobjective optimization, see Fonseca and Fleming [18]. Literature surveys and comparative studies on multiobjective genetic algorithms are also given in [13, 34, 64, 73].

Fonseca and Fleming have divided multiobjective genetic algorithms in non-Pareto and Pareto based approaches.

3.4.2.1 Non-Pareto based approaches

The first multi-objective genetic algorithm was VEGA (Vector Evaluating Genetic Algorithm) developed by Schaffer [55]. VEGA uses the selection mechanism of the GA to produce non-dominated individuals. Each individual objective is designated as the selection metric for a portion of the population. However, it is reported that the method tends to crowd results at extremes of the solution space, often yielding poor coverage of the Pareto frontier.

Fourman [22] presents a genetic algorithm using binary tournaments, randomly choosing one objective to decide each tournament. Kurasawe [44] further developed this scheme by allowing the objective selection to be random, fixed by the user, or to evolve with the optimization process. He also added crowding techniques, dominance, and diploidy to maintain diversity in the population.

All of these Non-Pareto techniques tend to converge to a subset of the Pareto-optimal frontier, leaving a large part of the Pareto set unexplored. Preferably, one wants to maintain diversity so that the entire Pareto frontier is elicited. Additionally, maintaining diversity will tend to improve robustness in multi-objective problems by ensuring that there is a genetic variety for mating mechanisms to operate upon [26, 31].

3.4.2.2 Pareto based approaches

Goldberg [25] introduced non-dominated sorting to rank a search population according to Pareto optimality. First, non-dominated individuals in the population are identified. They are given the rank 1 and are removed from the population. Then the non-dominated individuals in the reduced population are identified, given the rank 2, and then they are also removed from the population. This procedure of identifying non-dominated sets of individuals is repeated until the whole population has been ranked, as depicted in Figure 11. Goldberg also discusses using niching methods and speciation to promote diversity so that the entire Pareto frontier is covered.

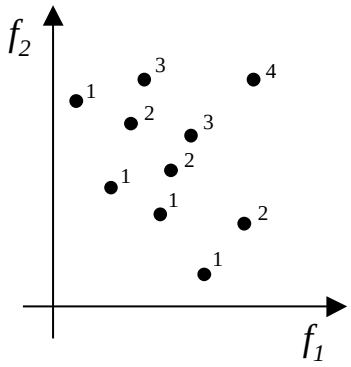

Figure 11: Population ranking based upon non-dominated sorting.

The non-dominated sorting GA (NSGA) of Srinivas and Deb [60] implements Goldberg's thoughts about the application of niching methods. In NSGA, non-dominated individuals in the population are identified, given a high initial individual score and are then removed from the population. These individuals are considered to be of the same rank. The score is then reduced using sharing techniques between individuals with the same ranking. Thereafter, the non-dominated individuals in the remaining population are identified and scored lower than the lowest one of the previously ranked individuals. Sharing is then applied to this second set of non-dominated individuals and the procedure continues until the whole population is ranked.

Sharing is performed in the parameter space rather than in the attribute space. This means that the score of an individual is reduced according to how many individuals there are with similar parameters, regardless of how different or similar they might be based on objective attributes.

In the multi-objective GA(MOGA) presented by Foseca and Fleming [19, 21] each individual is ranked according to their degree of dominance. The more population members that dominate an individual, the higher ranking the individual is given. An individual's ranking equals the number of individuals that it is dominated by plus one (see Figure 12). Individuals on the Pareto front have a rank of 1 as they are non-dominated. The rankings are then scaled to score individuals in the population. In MOGA both sharing and mating restrictions are employed in order to maintain population diversity. Fonseca and Fleming also include preference information and goal levels to reduce the Pareto solution set to those that simultaneously meet certain attribute values.

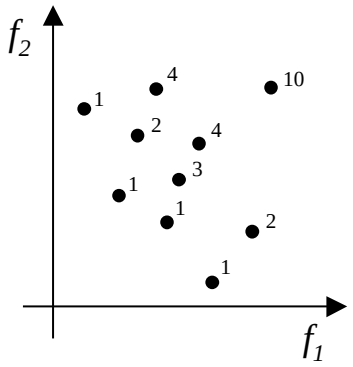

Figure 12: Population ranking according to Fonseca and Fleming.

The niched Pareto GA (NPGA) by Horn et al. [35] is Pareto-based but does not use ranking methods. Rather, Pareto domination tournaments are used to select individuals for the next generation. For binary tournaments, a subset of the population is used as a basis to assess the dominance of the two contestants. If one of the contestants is dominated by a member in the subset but the other is not, the non-dominated one is selected to survive. If both or neither are dominated, selection is based on the niche count of similar individuals in the attribute space. An individual with a low niche count is preferred to an individual with a high count to help maintain population diversity.

Zitzler and Thiele [73] developed a multi-objective genetic algorithm called the strengthen Pareto evolutionary algorithm (SPEA). SPEA uses two populations, *P* and *P'*. Throughout the process copies of all non-dominated individuals are stored in P'. Each individual is given a fitness value, f_i, based on Pareto dominance. The fitness of the members of *P'* is calculated as a function of how many individuals in *P* they dominate (20).

$$f_i = \frac{number\ of\ individuals\ dominated\ by\ i}{size(P)+1} \tag{20}$$

The individuals in *P* are assigned their fitness according to the sum of the fitness values for each individual in *P'* that dominate them plus one (see Figure 13). Lower scores are better and ensure that the individual spawns a larger number of offspring in the next generation. Selection is performed using binary tournaments from both populations until the mating pool is filled. In this algorithm, fitness assignment has a built-in sharing mechanism. The fitness formulation ensures that non-dominated individuals always get the best fitness values and that fitness reflects the crowdedness of the surroundings.

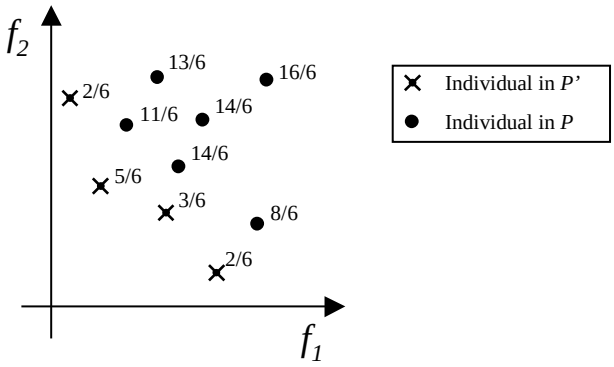

Figure 13: Population ranking according to Zitzler and Thiele.

Other methods such as the one presented by Tamaki [64] builds on the methods described above and adds features such as elitism together with the usage of multiple populations.

4 Discussion and conclusions

Engineering design is clearly about making many decisions often under uncertainty and with multiple conflicting criteria. In this paper, methods have been discussed where the design problem is reformulated as a multiobjective optimization problem. In order to solve such a problem a variety of different optimization methods are available. As we can not assume that derivatives of the objective functions to be known in the general case, we have to trust non-derivative optimization methods. Ones we have explored the objective space and identified promising regions, we might change to other optimization techniques in order to refine the search. As each optimization method has different properties suited for different type of problems, there is no simple answer to which method to use for a particular problem. However, if the problem is very complex with a mixture of continues and discrete variables and simultaneously their is no knowledge about the objective space, a robust method such as genetic algorithm might be the best choice.

As the design problem is multiobjective these objectives somehow has to be combined in order to yield one final solution. It is not likely that there exist a solution, which simultaneously minimizes all the objectives. The paper discusses pros and cons of different ways of articulating the decision-makers preferences in order to obtain the final solution.

By priori articulation of the decision-maker's preferences, an objective function is formulated which yields a scalar value that expresses the value of a candidate solution. These methods span from quick and dirty methods to methods that are more rigorous. Depending on the problem different approaches might be suitable. If the optimization is very time consuming,it might be a good idea to invest more time in the objective function formulation. For other problems, it might be adequate to start with a "quick and dirty" formulation to get a feeling for the problem before conducting the "real" optimization. Naturally, the decision-makers ability to articulate his or her preferences has to be taken into account as well. Depending on the problem and the decision-maker, a formal iterative approach might be more suitable.

However, in order to avoid the subjective judgment of the decision-maker the optimization could be performed with out any preference information. In the posteori articulation methods the search for an optimal set of solutions are based on Pareto optimality. This means to find the set of solutions where there for each solution does not exist any other solution which is better in all other attributes. I.e. solutions for which improvement in one attribute always leads to degeneration in another attribute. The Pareto-optimal set consists of all solutions according to any rational decision-maker. Here the search for an optimal set of solutions is separated from the final decision. The decision-maker is presented with a set of solutions from which he has to choose, and the hypothesis is that when the trade of between the objectives is visible it would be easier to chose. However, this might not hold as the number of objectives increases and visualization becomes harder. This is an interesting field for further research.

5 References

[1] Andersson J., *On Engineering System Design - A simulation optimization approach*, Licentiate thesis, Department of Mechanical Engineering, Linköping University, 1999.

[2] Andersson J., Pohl J., and Krus P., "Design of objective functions for optimization of multi-domain systems," presented at ASME Annual Winter meeting, FPST Division, Anaheim, California, USA, 1998.

[3] Bäck T., "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, pp. 1-23, 1993.

[4] Benayoun R., de Montgolfier J., Tergny J., and Laritchev O., "Linear programming with multiple objective functions: Step Method (STEM)," *Mathematical Programming*, vol. 1, pp. 366-375, 1971.

[5] Bland J. A., "Structural design optimization with reliability constraints using tabu search," *Engineering Optimization*, vol. 30, pp. 55-74, 1998.

[6] Borup L. and Parkinson A., "Comparision of four non-derivative optimization methods on two problem contaning heuristic and analythic knowledge," presented at ASME Advances in Design Automation, Scottsdale, Arizona, 1992.

[7] Box M. J., "A new method of constraint optimization and a comparison with other methods," *Computer Journal*, vol. 8, pp. 42-52, 1965.

[8] Charnes A. and Cooper W. W., *Management models and industrial applications of linear programing*. New York, John Wiley & Sons, 1961.

[9] Charnes A., Cooper W. W., and Ferguson R. O., "Optimal Estimation of executive compensation by linear programing," *Management Science*, vol. 1, pp. 138-151, 1955.

[10] Chattopadhyay A. and Seeley C. E., "Simulated annealing technique for multiobjective optimization of intelligent structures," *Smart Materials & Structures*, vol. 3, pp. 98-106., 1994.

[11] Chiampi M., G F., Ch M., C R., and M. R., "Multi-objective optimization with stochastic algorithms and fuzzy definition of objective function," *International Journal of Applied Electromagnetics in Materials*, vol. 9, pp. 381-389, 1998.

[12] Chiampi M., Ragusa C., and Repetto M., "Fuzzy approach for multiobjective optimization in magnetics," *IEEE transaction on magnetics*, vol. 32, pp. 1234-1237, 1996.

[13] Coello C., *An empirical study of evolutionary techniques for multiobjective optimization in engineering design*, Dissertation, Department of Computer Science, Tulane University, 1996.

[14] Connor A. M. and Tilley D. G., "A tabu search method for the optimization of fluid power circuits," *Journal of Systems & control engineering*, vol. 212, pp. 373-381, 1998.

[15] Das I. and Dennis J., "Normal-boundary interaction: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal of Optimization*, vol. 8, pp. 631-657, 1998.

[16] Das I. and Dennis J. E., "Closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Structural Optimization*, vol. 14, pp. 63-69, 1997.

[17] Eschenauer H., Koski J., and Osyczka A., "Multicriteria Design Optimization," . Berlin: Springer-Verlag, 1990.

[18] Fonseca C. and Fleming P., "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, pp. 1-18, 1995.

[19] Fonseca C. M. and Fleming P. J., "Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: a unified formulation," *IEEE Transactions on Systems, Man, & Cybernetics Part A: Systems & Humans*, vol. 28, pp. 26-37, 1998.

[20] Fonseca C. M. and Fleming P. J., "Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part II: application example," *IEEE Transactions on Systems, Man, & Cybernetics Part A: Systems & Humans*, vol. 28, pp. 38-47, 1998.

[21] Fonseca C. M. and J. F. P., "Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction," presented at 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems, Sheffield, England, 1995.

[22] Fourman M. P., "Compaction of symbolic layout using genetic algorithms," presented at 1st Int. Conference on Genetic Algorithms, Pittsburgh, 1985.

[23] Fu K.-C. and Levey G. E., "Discrete frame optimization by Complex-Simplex procedure," *Computers & Structures*, vol. 9, pp. 207-217, 1978.

[24] Glover F., "Tabu Search - Part I," *ORSA Journal on Computing*, vol. 1, pp. 190-206, 1989.

[25] Goldberg D., *Genetic Algorithms in Search and Machine Learning*. Reading, Addison Wesley, 1989.

[26] Grueninger T. and Wallace D., "Multi-modal optimization using genetic algorithms", Technical Report 96.02, CADlab, Massachusetts Institute of Technology, Cambridge, 1996.

[27] Guin J. A., "Modification of the Complex method of constraint optimization," *Computer Jornal*, vol. 10, pp. 416-417, 1968.

[28] Gunel T. and Yazgan B., “New global hybrid optimization algorithm,” presented at 2nd Biennial European Joint Conference on Engineering Systems Design and Analysis, London, England, 1994.
[29] Hajela P., “Nongradient methods in multidisciplinary design optimization - status and potential,” *Journal of Aircraft*, vol. 36, pp. 255-265, 1999.
[30] Haque M. I., “Optimal frame design with discrete members using the complex method,” *Computers & Structures*, vol. 59, pp. 847-858, 1996.
[31] Harik G., “Finding multimodal solutions using restricted tournament selection,” presented at Sixth International Conference on Genetic ALgorithms, 1995.
[32] He D. W. and Kusiak A., “Design of assembly systems for modular products,” *IEEE Transactions on Robotics & Automation*, vol. 13, pp. 646-655, 1997.
[33] Holland H. J., *Adaptation in Natural and Artificial Systems, an introductory analysis with application to biology, control and artificial intelligence*. Ann Arbor, The university of Michigan Press, 1975.
[34] Horn J., “Multicriterion decision making,” in *Handbook of evolutionary computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds., IOP Publishing Ltd and Oxford University Press, 1997.
[35] Horn J. and Nafpliotis N., “Multiobjective Optimization Using the Niched Pareto Genetic Algorithm”, Technical Report 93005, Illinois Genetic Algorithm Laboratory, Dept. of General Engineering, University of Illinois at Urbana-Champaign, Urbana, USA, 1993.
[36] Hwang C., Paidy S., and Yoon K., “Mathematical programming with multiple objectives: a tutorial,” *Computers & Operations Research*, vol. 7, pp. 5-31, 1980.
[37] Jansson A., *Fluid Power System Design - A Simulation Approach*, Dissertation, Department of mechanical engineering, Linköping University, 1994.
[38] Keeney R. and Raiffa H., *Decisions with multiple objectives - preferences and value tradeoffs*. New York, USA, John Wiley & Sons, 1976.
[39] Kim J.-b. and Wallace D., “A Goal-oriented Design Evaluation Model,” presented at ASME Design Theory and Methodology conference, Sacramento, CA, USA, 1997.
[40] Kirkpatrick S., Gelatt C. D., and Vecchi M. P., “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.
[41] Kroo I., McMasters J., and Smith S., “Highly Nonplanar Lifting Systems,” presented at Transportation Beyond 2000: Technologies Needed for Engineering Design, NASA Langley Research Center, 1995.
[42] Krus P., Jansson A., Berry P., Hansson E., and Ovrebo K., “ A Generic Model Concept for Optimisation in Preliminary Design,” presented at World Aviation Congress and exposition, Los Angeles, California, USA, 1996.
[43] Krus P., Palmberg J.-O., Löhr F., and Backlund G., “The Impact of Computational Performance on Optimisation in Aircraft Design,” presented at I MECH E, 'AEROTECH 95', Birmingham, UK, 1995.
[44] Kurasawe F., “A variant of evolution strategies for vector optimization,” in *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, Berlin, Springer Verlag, pp. 193-7, 1991.

[45] Manetsch T. J., “Toward efficient global optimization in large dynamic systems - The adaptive complex method,” *IEEE Transactions on Systems, Man & Cybernetics*, vol. 20, pp. 257-261, 1990.

[46] Manetsch T. J. and Cabrera A. D., “Use of concurrent processing with the adaptive complex method for global optimization of large dynamic systems,” *IEEE Transactions on Systems, Man & Cybernetics*, vol. 21, pp. 442-445, 1991.

[47] Mongeau M., Karsenty H., Rouzé V., and Hiriart-Urruty J.-B., “Comparision of public-domain software for black-box global optimization”, Tecnical report LAO 98-01, Laboratoire Approximation et Optimisation, Université Paul Sabatier Toulouse, Toulouse, France, 1998.

[48] Mori N., Yoshida J., Tamaki H., Kita H., and Nishikawa Y., “Thermodynamical selection rule for the genetic algorithm,” presented at IEEE Conference on Evolutionary Computation, Perth, Australia, 1995.

[49] Morse J. N., “Reducing the size of the nondominated set: pruning by clustering,” *Computers & Operations Research*, vol. 7, pp. 55-66, 1980.

[50] Nelder J. A. and Mead R., “A simplex method for function minimization,” *Computer Journal*, vol. 7, pp. 308-313, 1965.

[51] Osyczka A., *Multicriterion Optimization in Engineering - with FORTRAN Programs*. Chichester, Ellis Horwood, 1984.

[52] Rechenberg I., *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Friedrich Frommann Verlag, 1973.

[53] Rosenman M. A. and Gero J. S., “Redusing the pareto opimal set in multicriteria optimization (With application to Pareto optimal dynamic programing),” *Engineering Optimization*, vol. 8, pp. 189-206, 1985.

[54] Sahoo N. K. and Apparao K. V. S. R., “Modified complex method for constrained design and optimization of optical multilayer thin-film devices,” *Applied Physics A-Solids & Surfaces*, vol. 59, pp. 317-326, 1994.

[55] Schaffer J., “Multiple objective optimization with vector evaluated genetic algorithms,” presented at 1st Int. Conf. on Genetic Algorithms, Pittsburgh, 1985.

[56] Senin N., Wallace D. R., and Borland N., “Mixed continuous and discrete catalog-based design modeling and optimization,” presented at 1999 CIRP International Design Seminar, University of Twente, Enschede, The Netherlands, 1999.

[57] Shea K., “Applying shape annealing to full-scale transmission tower re-design,,” presented at ASME Design Automation Conferece, Las Vegas, Nevada, 1999.

[58] Siddall J., *Analytical decision-making in engineering design*. Englewood Cliffs, New Jersey, Prentice-Hall, 1972.

[59] Spendley W., Hext G. R., and Himsworth F. R., “Sequencial application of Simplex designs in optimisation and evolutionary operation,” *Technometrics*, vol. 4, pp. 441-462, 1962.

[60] Srinivas N. and Deb K., “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, pp. 221-248, 1995.

[61] Steuer R., *Multiple criteria optimization: theory, computation and application*. New York, John Wiley & Sons, Inc., 1986.

[62] Steuer R. E. and Choo R.-U., “An interactive weighted Tchebycheff procedure fo multiple onjective programming,” *Mathematical Programming*, vol. 26, pp. 326-344, 1983.

[63] Suppapitnarm A., Seffen K. A., Parks G. T., Clarkson P. J., and Liu J. S., "Design by multiobjective optimization using simulated annealing," presented at International conference on engineering design ICED 99, Munich, Germany, 1999.

[64] Tamaki H., Kita H., and Kobayashi S., "Multi-objective optimization by genetic algorithms: a review," presented at 1996 IEEE International Conference on Evolutionary Computation, ICEC'96, Nagoya, Japan, 1996.

[65] Tamiz M., Jones D., and Romero C., "Goal programming for decision making: An overview of the current state-of-the-art," *European Journal of Operational Research*, vol. 111, pp. 569-581, 1998.

[66] Thurston D., "A formal method for subjective design evaluation with multiple attributes," *Research in engineering design*, vol. 3, pp. 105-122, 1991.

[67] Thurston D. and Liu T., "Design evaluation of multiple attributes under uncertainty," *International journal of systems automation and applications*, vol. 1, pp. 143-159, 1991.

[68] Wallace D. R., Jakiela M. J., and Flowers W. C., "Multiple criteria optimization under probabilistic design specifications using genetic algorithms," *Computer-aided Design*, vol. 28, pp. 405-421, 1996.

[69] Yang S. Y., Park L.-J., Park C. H., and Ra J. W., "Hybrid algorithm using genetic algorithm and gradient-based algorithm for iterative microwave inverse scattering," presented at IEEE Conference on Evolutionary Computation, Perth, Australia, 1995.

[70] Yen J., Liao J., Lee B., and Randolph D., "A hybrid approch to modeling metabolic systems using a genetic algorithm and simplex method," *IEEE Transaction on systems, man and cybernetics*, vol. 28, pp. 173-191, 1998.

[71] Yip P. and Pao Y.-H., "A guided evolutionary simulated annealing approach to the quadratic asignment problem," *IEEE Transaction on systems, man and cybernetics*, vol. 24, pp. 1383-1387, 1994.

[72] Zimmermann H.-J. and H.-J. S., "Intelligent system design support by fuzzy-multi-criteria decision making and/or evolutionary algorithms," presented at IEEE International Conference on Fuzzy Systems, Yokohama, Japan, 1995.

[73] Zitzler E. and Thiele L., "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transaction on evolutionary computation*, vol. 3, pp. 257-271, 1999.