

DETC99/DTM-8757

GENETIC ALGORITHMS IN FUZZY ENGINEERING DESIGN

Ralf Schleiffer
German Aerospace Center
Department of Transport
Research
Linder Höhe, 51147
Cologne, Germany
Email: Ralf.Schleiffer@dlr

Prof. Dr. Hans-Jürgen Sebastian
Institute for Operations
Research
RWTH Aachen
Templergraben 64
52062 Aachen
Germany

Prof. Erik K. Antonsson, Ph.D.*
Engineering Design Research
Laboratory
California Institute of
Technology
1200 E. California Blvd.
Pasadena, California 91125

ABSTRACT

Problems in the field of engineering design represent an important class of real world problems that typically require a fuzzy and imprecise representation. This article presents and discusses a new approach to model this type of problem, by incorporating linguistic descriptions together with a variety of user-defined trade-off strategies. An interactive computer application is introduced, using stochastic optimization to solve the design task by producing a specially desired output under the given environmental conditions which are partly caused by the personal preferences of the engineer and by the expectations of the customer. It utilizes a randomized evolutionary technique, made suitable for the class of problems at hand, to generate and to optimize design solutions that are later identified by a clustering algorithm. Moreover test problems that were solved by the application are considered. In all cases the good solutions were obtained by evaluating only an extremely small fraction of all possible designs.

KEYWORDS

Genetic Algorithms, Fuzzy Engineering Design, Method of Imprecision (MoI), Preference Modelling, Mathematical Model of Linguistic Descriptions, Population Based Optimization, Fuzzy-C Means Algorithm (FCMA), Computer Application

1 INTRODUCTION

There is an increasing interest in developing tools to determine which among several design variants best fulfill the preferences of an engineer (expert knowledge) and of a client (product requirements). This interest is particularly strong for the preliminary phase of the engineering design process when the design is only imprecisely known. A methodology that formalizes this phase is the Method of Imprecision (MoI), whose concept and algorithms are presented and discussed by [Antonsson], [Law], [Otto], [Scott] and [Wood]. Its goal is to transform, starting from a known design concept, a set of requirements into a vector of parameter values in order to construct an object that fulfills engineers' and customers' demands.

Requirements and constraints are always present in engineering design: they are commonly a mixture of legal regulations, finance plans, physical laws, traditions, culture and habits. The engineer has to consider all of these items to create variants according to customers' requirements and desires. Design can be understood as the generation of recipes for creating objects that meet a collection of requirements and constraints. Commonly, a large portion of an engineer's work is to combine components that are well-known, in an attempt to meet the requirements and constraints. Evaluation of the large number of possible combinations of these components is a

* corresponding author

search or exploration guided by the engineer's experience and judgement.

This paper presents an engineering design methodology that combines imprecise representation of early design information with a genetic algorithm to guide the search for combinations of components that best satisfy the requirements and constraints. This methodology has been implemented in a computer program, and is illustrated by an example design problem taken from the literature.

This paper is organized as follows. Section 2 describes the basic definitions that are utilized to model the design task, following the MoI. The formalization and handling of uncertain and fuzzy data in real world problems is the subject of Section 3. In Section 4 an appropriate mathematical representation is stated. Since this model can not be solved analytically but only handled by suitable numerical methods the powerful non-linear search capabilities of a particular evolutionary technique combined with a clustering strategy are described in Section 5. Subsequently a computer application is illustrated in Section 6. Its advantage is its rapid identification of a range of good design solutions. Section 7 briefly describes some design problems in the field of mechanical engineering design, in which complex designs were worked out based on large numbers of imprecisely described parameters as well as on precise ones. Finally, Section 8 summarizes the results of the previous sections.

2 DEFINITIONS

The following definitions characterize the design problem that is examined throughout the rest of this paper.

Definition 1 (*design problem*)

A design problem is the specification of parameter settings in technical construction environments.

Definition 2 (*design parameter space, design variable*)

The design parameter space (*DPS*) is the set of all possible solutions of a design. Its elements are denoted $d := [d_1, \dots, d_m]$ with $d_j \in X_j$, $j \in J \subset \mathbf{N}$, $|J| = m \in \mathbf{N}$, being an attribute that specifies a variable of a possible design. d_j is called a design variable (DV). X_j denotes any possible set.

Typically (but not necessarily) X_j is a set of points in an interval in \mathbf{R}^1 . In real world problems, however, there is always some uncertainty in measurement. Thus the *DPS* can be treated as a discrete set of rational numbers.

Definition 3 (*performance parameter space, performance variable*)

The performance parameter space (*PPS*) is the set of all considered objectives that a possible design can achieve. Its elements are denoted $p = [p_1, \dots, p_n]$ with $p_i \in Y_i$, $i \in I \subset \mathbf{N}$, $|I| = n \in \mathbf{N}$ being a particular considered objective that a possible design can achieve. p_i is called a performance variable (PV).

Again Y_i notes any possible set.

It is assumed that there exists a mapping $f_i: DPS \rightarrow PPS$, $d \mapsto p_i \quad \forall i \in I$ and furthermore it is postulated that every DV is utilized in at least one of the mappings f_i , $i \in I$. Unlike in other approaches, the mappings here are not restricted. They can be any calculation, computational algorithm, procedure or function that is able to determine the approximate or exact relationship and interaction between DVs and PVs.

3 MODELING IMPRECISE PREFERENCES

The strategy of the approach presented here is to utilize and combine the subjective preferences of the decision makers, known as DMs, on several characteristics of a design in order to formalize, to structure and to describe the real world problem, to build a moderate size model of it, and to employ this model for decision making. It is assumed that the DMs will more or less consciously incorporate aspects of the design that are not explicitly calculated into their preferences on DVs and on PVs, with the engineer judging about DVs and the customer articulating his preferences on PVs. Of course, the engineer may have preferences on PVs, too (see also [Antonsson], [Sebastian]). Dealing with real world problems typically requires the processing of uncertain and imprecise data, and thus the DMs' preferences are typically uncertain and vague. Sources of this uncertainty include the complexity of the problem, unreliable data and insufficient and incomplete information. A promising approach to handle it is offered by the theory of fuzzy sets that can represent preferences by allowing a

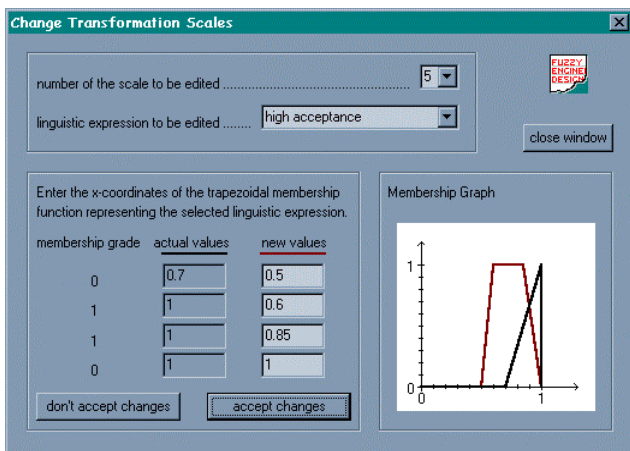
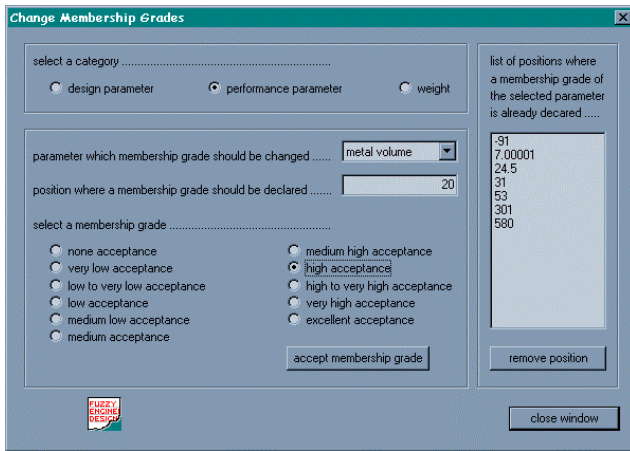


Figure 1. DIALOGS TO DECLARE PREFERENCES LINGUISTICALLY AND TO DEFINE TRANSFORMATION SCALES.

variable's value to have partial membership in more than one set.

These fuzzy preferences are denoted μ_{d_j} , $j \in J$ and μ_{p_i} , $i \in I$ respectively. They can be regarded as constraints or as objectives (functional requirements), that should be optimized by maximizing the preference functions. In the following discussion a design will only be accepted as a possible solution when none of the DVs fails, i.e. when μ_{d_j} is different from *not acceptable* $\forall j \in J$. Hence the preference statement on a DV defines its support as well.

Modeling real world test problems with engineers from different disciplines showed that it is generally a difficult task to select any one particular value as a representation of a grade of preference. Most of the engineers interviewed believed that the preferences of the participating DMs are best expressed linguistically, such as: *This interval is definitely excellent.*, *This point is more or less bad.*, etc.

The only crisp preferences that could be determined by each of them were an aspiration level and a reservation level.

We have found it preferable in handling the preference representation to model the vagueness of the linguistic statements itself by fuzzy sets and therefore to assign a whole interval to each of them, with some points of the interval belonging more to the statement and others less. Taking the ideas presented by [Bonissone] and others, the method converts linguistic terms into fuzzy numbers. This is performed by utilizing a similar numerical approximation system, consisting of a limited number of conversion scales, as was suggested by [Chen] in 1992. In order to make the subjective nature of any preference statement explicit, the scales used by the application can be defined and manipulated by the DMs themselves. A detailed discussion of the scales is available in [Schleiffer]. The computer application's natural language user interface is shown in Figure 1.

The DMs' preferences are reasonably represented by fuzzy sets of type-two [Zimmermann]. For an example see Figure 2. The top graphic shows the (trapezoidal) fuzzy numbers that are obtained after transforming the linguistic preference descriptions for DV d_j . If the DV is defined on a cardinal scale as is assumed in the example (bottom graphic), the application determines the membership grades of those points where it is not explicitly declared, by linearly interpolating between the closest left point and the closest right point on the d_j -axis. That leads to the following definition:

Definition 4 (grade of membership where not explicitly stated)

Let $x \in X$ and $x_L, x_R \in X$ be the closest left and right point to x where a grade of membership is explicitly declared. Let further $M_{x_L} := [x_1^L, x_2^L, x_3^L, x_4^L]$ and $M_{x_R} := [x_1^R, x_2^R, x_3^R, x_4^R]$ be two trapezoidal fuzzy numbers describing the membership grade of x_L and x_R respectively. The fuzzy number representing the grade of membership of the value x is then defined as $M_x := [x_1^M, x_2^M, x_3^M, x_4^M]$ where

$$x_{\vartheta} := \begin{cases} x_{\vartheta}^R, & \text{if } x_L \text{ does not exist} \\ \frac{x_{\vartheta}^R - x_{\vartheta}^L}{x_R - x_L} \cdot x + \frac{x_{\vartheta}^L \cdot x_R - x_{\vartheta}^R \cdot x_L}{x_R - x_L}, & \text{if } x \in [x_L, x_R] \\ x_{\vartheta}^L, & \text{if } x_R \text{ does not exist} \end{cases}$$

$$\vartheta \in \{1, 2, 3, 4\}.$$

It should be admitted that the assumption of linear approximation in the above definition as well as the translation of linguistic terms to trapezoidal fuzzy numbers may be problematic, since there is neither empirical nor formal justification that the obtained model is a true model of reality. They are used for their simplicity. If the DMs know about non-linear approximations or about a non-trapezoidal shape of the fuzzy numbers they should incorporate this knowledge into the model in order to achieve a more realistic representation.

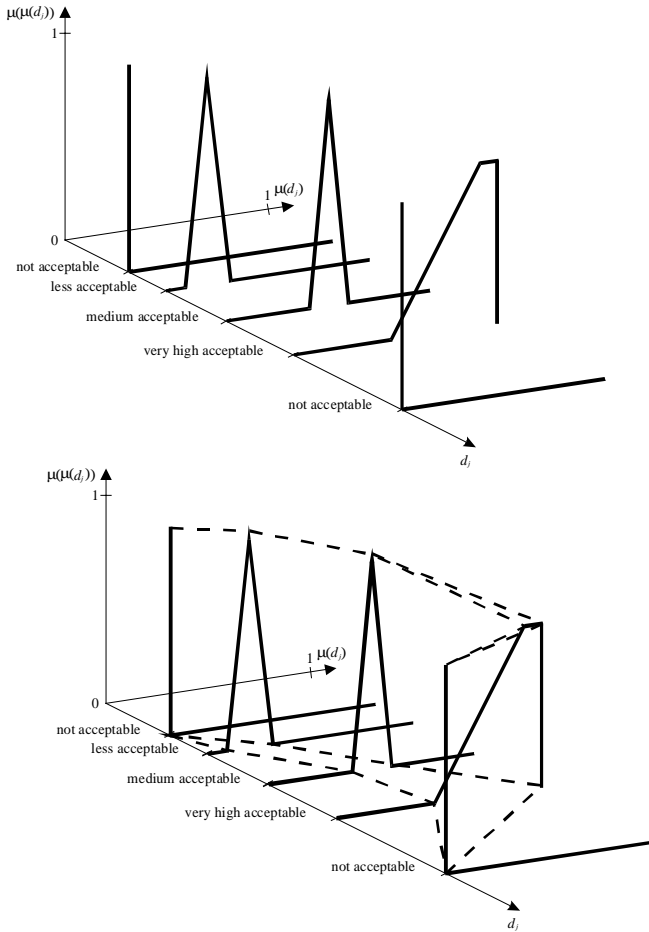


Figure 2. MEMBERSHIP INFORMATION ON DV d_j .

4 THE MATHEMATICAL PROBLEM

The objective of the design is to maximize the fulfillment of the DMs' preferences on multiple, possibly competing, DVs, and PVs. The corresponding mathematical model can be formulated as a multiobjective problem based on the concept of dominance. Unfortunately an attempt to find a maximal solution at which all components of a vector $s^* \in DPS$ are simultaneously maximized is not adequate, since such vectors seldom exist. A way to handle this might be to search for Pareto optimal designs. Then, in the absence of any further information, all Pareto optimal options must be shown to the DMs in order to let them select one option. This can be done by heuristic strategies such as computing their median. But usually the choice of a single best solution is based on subjective criteria the DMs cannot express explicitly.

In many design problems, additional information is available to join the single dimensions, such as grades of importance of the participating variables. In the application these are expressed linguistically and then modeled as fuzzy weights. For details refer to [Schleiffer]. By including this knowledge the multiobjective problem can be formulated as a one-dimensional task with every design's performance being aggregated into a single fuzzy number. Then the problem to be solved is:

Find all solutions $s^* \in DPS$ such that

$$c \cdot \left[\bigcirc_{j \in J, i \in I} \left(\mu_{d_j}(s_j^*) \circ \mu_{p_i}(f_i(s^*)) \right) \right] \geq \bigcirc_{j \in J, i \in I} \left(\mu_{d_j}(s_j) \circ \mu_{p_i}(f_i(s)) \right) \forall s \in DPS$$

where "O" and "o" present any suitable operator and $c \in \mathbf{R}^{>0}$.

This is a problem with a single objective, a supercriterion function, expressed in terms of the DVs. Strong attention must be paid when building this overall objective function to make sure it is acceptable to the DMs. It cannot be expected that it always represents their desires exactly. Therefore the optimization should not merely lead to a single best solution. Instead it should search for a set of good solutions that might be ranked by the DMs themselves whereby the meaning of *good* is coded in the variable c . Of course $c \in \mathbf{R}^{>0}$ makes sense, too. But if $c \in (0, 1) \subset \mathbf{R}^1$, it does allow solutions that are worse than the best ones.

It should be noted that the object of optimization is the preference declarations of the DMs directly.

If the DMs are not satisfied with the determined solutions, they may decide to run the optimization again, perhaps with changed preferences, weights or trade-off strategies. These modifications would be supported if the previous optimization process explored the space of possible solutions and was able to give information about its structure. So another important task of an appropriate tool is to provide the DMs with further insights into the set of all possible designs by identifying regions where design solutions are worse and those where they are better, since this information enhances the understanding of the interactive behavior of included variables.

Apart from the mappings from DPS to PPS , other interdependencies among the different variables are often important for solving a particular problem. The application considers three stages of possible trade-off between them.

The first one is among the DVs themselves formalizing their interaction. Following the ideas developed in the MoI (see [Otto]), there are two alternatives of interest. One is to design without any trade-off between the DVs and the other is to design with compensation among them. In general the strategy will be a mixture of both. There will be some DVs that cannot be traded-off against others and there are other groups of DVs with trade-off among them. The result is an overall preference $P_D(s)$ on a particular design solution $s \in DPS$ expressed in terms of DVs. As in the MoI, these strategies are modeled by employing a minimum operator or a product operator respectively. If $s := [s_1, \dots, s_m]$ is any possible design and if $\mu_{d_j}(s_j) := M_{s_j} := [x_1^{M_{s_j}}, x_2^{M_{s_j}}, x_3^{M_{s_j}}, x_4^{M_{s_j}}]$ notes the grade of attractiveness of the value s_j and $\mu_{\omega}(d_j) := M_{\omega_{d_j}} := [x_1^{M_{\omega_{d_j}}}, x_2^{M_{\omega_{d_j}}}, x_3^{M_{\omega_{d_j}}}, x_4^{M_{\omega_{d_j}}}]$ represents the importance of DV $d_j \forall j \in J$, then $P_D(s)$ is defined as:

$$P_D(s) := \begin{cases} \left[\min_{j \in J} (x_1^{M_{s_j}} \cdot x_1^{M_{\omega_{d_j}}}), \dots, \min_{j \in J} (x_4^{M_{s_j}} \cdot x_4^{M_{\omega_{d_j}}}) \right], & \text{no trade-off} \\ \left[\prod_{j \in J} (x_1^{M_{s_j}} \cdot x_1^{M_{\omega_{d_j}}}), \dots, \prod_{j \in J} (x_4^{M_{s_j}} \cdot x_4^{M_{\omega_{d_j}}}) \right], & \text{trade-off} \end{cases}$$

depending on the trade-off strategy chosen. Notice that when using the minimum operator, the value $j \in J$, where

the minimum is determined, can be different on distinct components. An alternative approach to avoid this is to multiply $M_{s_j} \cdot M_{\omega_{d_j}} \forall j \in J$ using the approximation formulas developed by [Dubois and Prade] and to rank these fuzzy numbers and specify the lowest one. As this ranking is computationally expensive, the simpler operator has been implemented. By using this minimum operator one is always on the safe side because the combined preference cannot be higher than the one of the worst component of s .

The second stage where a trade-off strategy is necessary is when the DVs mapped to PPS are combined with the customer's preferences. The function P_{DP_i} combining these two membership grades should be able to design more to the customer's or more to the engineer's preferences. A convex combination of the preferences is chosen to model this phenomenon. It is

$$P_{DP_i}(s) := \theta \cdot P_D(s) + (1 - \theta) \cdot \mu_{p_i}(f_i(s)),$$

where $\theta \in [0, 1] \subset \mathbf{R}^1$.

The third stage of possible trade-off is when the combined customer and engineer preferences on each PV are joined together. The customer might accept a solution that fulfills some of his requirements only to a low degree but others to a high degree. This trade-off here is similar to combining the variables with the logical *and* or with the logical *or*, or maybe with an operator between them. A version of the *Gamma-Operator* developed by [Zimmermann] will be used. It offers the possibility to specify where between logical *and* and logical *or* the actual operator is located.

With that additional information available the model of the design problem can be written as:

$$\text{Find all solutions } s^* := [s_1^*, \dots, s_m^*] \in DPS \text{ such that} \\ c \cdot \left[\prod_{i \in I} (\mu_{\omega}(p_i) \cdot P_{DP_i}(s^*)) \right]^{1-\gamma} \cdot \left[1 - \prod_{i \in I} (1 - \mu_{\omega}(p_i) \cdot P_{DP_i}(s^*)) \right]^{\gamma} \geq$$

$$\left[\prod_{i \in I} (\mu_{\omega}(p_i) \cdot P_{DP_i}(s)) \right]^{1-\gamma} \cdot \left[1 - \prod_{i \in I} (1 - \mu_{\omega}(p_i) \cdot P_{DP_i}(s)) \right]^{\gamma}$$

$$\forall s = [s_1, \dots, s_m] \in DPS \quad \text{with} \quad \gamma \in [0, 1] \subset \mathbf{R}^1,$$

$$\mathbf{1} := [1, 1, 1, 1] \text{ and } c \in \mathbf{R}^{\geq 1}.$$

Strategies that introduce static and dynamic fuzzy extrema in order to perform the " \geq "-comparisons are developed

and discussed in [Schleiffer]. In the same paper proofs for the following six characteristics of a design modeled by type-two-fuzzy sets and utilizing the above trade-off strategies and weights are presented. The properties are:

- The worst possible design has a zero quality.
- The quality of any design cannot increase higher than one.
- If a design completely fails on one DV, the design solution as a whole fails as well.
- The overall quality of a design does not depend on the order of the preferences in combination with their weights, but only on their values.
- If the preference on a single variable or its grade of importance increases or decreases by utilizing a different linguistic term from one of the presented transformation scales, the total quality of the design increases or decreases respectively.
- Two designs that differ slightly on one variable but that are equal on the others and on weights, have a similar quality, except in the case that one design has zero quality because of a failing DV.

5 THE ALGORITHM

In most practical cases the above design problem cannot be solved analytically but can only be handled by suitable numerical methods. Moreover as the design process proceeds it would be best to present more and more acceptable and reasonable designs to the engineer and to the customer to enable them to develop their preferences in order to find innovative and competitive designs of high performance. In a large design space, an unsupported human doing traditional heuristic design might miss these designs.

Genetic Algorithms (GA) (see [Davis], [Goldberg]), a family of stochastic optimization techniques, provide multiple, acceptable and near optimum design candidates in large, possibly unstructured design spaces. This means that the engineer and the customer can examine these candidates, and their judgment on these gives them more information to develop their preference statements.

It should be emphasized that there is no proof that a GA will find the global optimal solution of an optimization problem. Yet many applications of GAs to a wide range of

optimization problems have shown in the past that they are an effective tool, especially in non-smooth domains where calculus-based methods are ineffective.

In particular, a GA exhibits several major characteristics that make it particularly attractive:

- easily applicable to changes in a stated problem.
- finds many acceptable solutions.
- escapes from local optima.
- there is no need for the computation or even the existence of derivatives.
- continuity of the functions is not required.
- easily exploited to parallelism.

When working with GAs a solution alternative is called an *individual*, denoted $A_u(t)$, where $t \in \mathbf{N}_0$ is a variable representing time. It will be assumed that the number of individuals is the same at every discrete time t . So it is $u \in U(t) \subset \mathbf{N}$ with $|U(t)| = v$ and $v \in \mathbf{N}$ being constant for all $t \in \mathbf{N}_0$. The set of all individuals that exist at a specific time t is referred to as a *generation*, $\mathbf{A}(t) := \{A_u(t) \mid u \in U(t)\}$, $t \in \mathbf{N}_0$. Every individual decodes a point in *DPS*, a possible design. They are identified by their genetic representation analogous to one or a combination of more chromosomes that form the total genetic prescription in biological systems.

As no high order coding shows itself superior to the binary case, except concerning difficulties with Hamming-cliffs and short step mutation that might be solved by running bit-wise as well as variable-wise operators (for a detailed discussion see [Schleiffer]) the implemented algorithm employs a binary alphabet for coding the genes. With such a binary alphabet only pseudo-boolean optimization problems can be handled in a straightforward manner. All other problems require coding functions.

In common engineering design problems there are DVs defined on cardinal scales usually modeled as real-valued continuous variables (e.g. radius, length) or as integer variables (e.g. number of gears), and there occur others that can be defined on a nominal scale (e.g. color, material). Accordingly different coding functions are exploited to deal with them. The strategy to code cardinal

phenotype:

length (in the range 0.18 m
to 0.49 m)

color (red,
blue, green)

Height (in the range 0 m to 63 m)

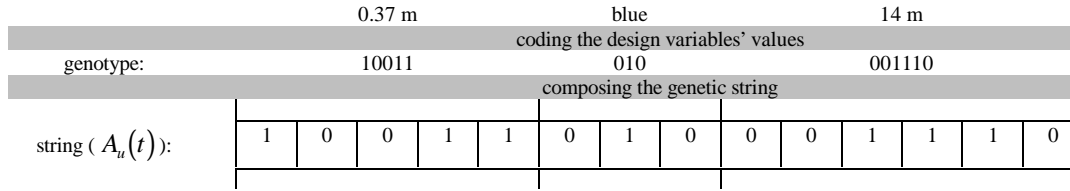


Figure 3 EXAMPLE OF BUILDING THE GENETIC STRING FROM DVs.

variables is based on the principle presented by [Srinivas and Patnaik], which is illustrated in Figure 3.

GAs explore a search space in a way analogous to the basic biological evolution theory. The first phase of a GA consists of the construction of an initial population of individuals, typically created at random to produce a large diversity among the individuals. Alternatively, an initial set of solution alternatives can be created using past designs from the *DPS* or by letting the user choose them heuristically.

In the approach here a guided random initialization of the GA will be carried out. As there is *a priori* knowledge about good points in *DPS* available in the form of the engineer's preferences, this information will be used to generate the first population. Individuals will be initialized with only these values of the single DVs where the engineer expects the solution, i.e. his preference declaration attains a high value ($d_j \in [\hat{l}_{d_j}, \hat{u}_{d_j}] \subset \mathbf{Z}$, $j \in J$). The choice among these points is made randomly. Note that this is different from choosing a population of members that are thought to have good fitness values heuristically by the user, since in the guided random initialization the probability to select one of them is equally distributed and consequently the expected fitness value of each initial individual is the same. That reduces the chance of premature convergence when the first generation, possibly with many similarities among the individuals, is explicitly defined by the user. The engineer's experience is used to guide the search to explore regions that are expected to produce improving solutions of well known designs with a higher probability than driving the search towards new and innovative design solutions.

Given a particular genetic string, the fitness of the related individual will be computed from the chromosomes using a fitness function that returns a

single numerical value, which is proportional to the individual's ability or utility.

For many problems using GAs the fitness function is simply the value of the function that should be optimized. However, in the case at hand, there are many performances to be optimized. In the tank example presented in Section 6 it is a combination of *length*, *radius*, *type*, *metal volume*, *tank capacity*, *outer length* and *outer radius*. The real world test problems required many more variables.

Within this presentation the fitness of an individual is the quality of the design solution that it represents, measured by the DMS' preferences, using the concepts outlined in Section 3. As a result, the engineer's and the customer's preferences on DVs and on PVs, that provide the mechanisms for evaluating each chromosome, lead to a fuzzy number for each individual, describing its ability to thrive in the target environment.

This objective function considers fuzzy goals and fuzzy constraints simultaneously as they are expressed in terms of preference for the DVs' values and for the PVs' values. Thus assigning a low preference value may be viewed as a way of penalizing the solution alternative at hand.

Now these fuzzy numbers will be defuzzified; they will be mapped to numerical values. The choice of a defuzzification method will depend on the algorithm's phase. Within a *fight* phase a dynamic fuzzy minimum and a dynamic fuzzy maximum are introduced similar to the ideas presented by [Chen] in 1985, and the total quality of each fuzzy quality is computed according to these two extremes. In the selection phase, static fuzzy extremes as they were introduced by [Chen] in 1992 are employed. A detailed discussion why these particular defuzzification schemes were used is available in [Schleiffer].

Once the first generation is initialized, individuals are selected from this generation to form a new one. During the selection phase the individuals are selected from the population $\mathbf{A}(t)$, $t \in \mathbf{N}_0$ and then recombined to form a new fitter one $\mathbf{A}(t+1)$, $t \in \mathbf{N}_0$.

The generally utilized method in the selection phase is known as *fitness proportionate reproduction* and is

analogous to selection in natural systems. It can be regarded as a roulette wheel in which the width of the slots is in proportion to the individuals' fitness values. This type of reproduction might result in a population that does not contain the best individuals of the previous generation. Therefore it is a common practice to use an elitist strategy by copying a pre-defined percentage of the best solutions from the actual generation to the next one and to perform random reproduction only on the rest of the individuals forming the following generation, with all members of the previous one included in the roulette wheel.

Unfortunately it is not obvious that this selection procedure can be regarded as a natural one since the GA software has total control of the form of the solution, the fitness function, and the genetic operators. In this case the individuals interact exclusively with the fitness function and not with each other, which is an important characteristic of natural selection. In our approach one component of the fitness function is not included in the system specification so that the individuals interchange with each other as well as they interact with the fitness function. This leads to a weighty influence of natural selection and moreover it enables the DMs to obtain deeper insights in the topology of the problem by determining a whole set of local optima additionally to the global one.

The idea is to model Darwin finches (compare with [Goldberg and Richardson]) artificially. The underlying thought is that individuals have to partition their fitness value with others with whom they share a region in the search space, similar to dividing up resources in real nature, where the behavior of competitors alters an individual's fitness. This is also comparable to game theory where pay-off depends on the strategies of opponents. The path followed here is similar to that presented by [Goldberg and Richardson] in 1987. The difference is that the neighborhood is not defined depending on a grid in the search space but according to an individual's phenotype. To do so the individuals of every generation are clustered by a Fuzzy-C Means Algorithm (FCMA) and subject to the maximum grade to which one belongs to one of the clusters it is assigned to one specific subpopulation. Then in the selection phase each individual is examined and in its neighborhood that is equal to the subpopulation to which it belongs, one individual is, conditionally to the fitness-distribution in the examined neighborhood, chosen to be reproduced when forming the next generation.

The other operators that are used by the application are mutation, crossover and fight, made suitable to support the underlying neighborhood strategy.

5 THE COMPUTER APPLICATION

The implementation of the approach is programmed in C++ under *Microsoft Windows*. If the design problem is limited to approximately 30 DVs and 30 PVs the application provides the DMs with sufficient results in reasonable time on an ordinary personal computer with a 486-processor. The program itself is based on threads that work in parallel and enable the modification of the setting of variables while the optimization is running. Hence DMs can examine the results and the search behavior obtained by different adjustments of the variables defining the trade-off strategy and also by different preference articulations.

The application incorporates menus and dialogs with default parameter setting where entries can be made by menu selection and form-filling, as well as a graphical user interface for visualization of the performance of the algorithm. Explanations are available on request and when wrong entries occur.

The program can easily be extended for further development and modifications. If a multi-processor workstation or a parallel computer is available, one thread can effortlessly be assigned to one processor handling the initialization of the first population as well as the formation of any other generation, while other threads evaluate design alternatives and perform operations on subpopulations on the additional processors.

A graphical user interface provides information about the behavior of the search process in order to allow the DMs to view the progress of the search and to support them to control it by adjusting strategic variables (SVs) such as mutation probability, crossover probability, crossover strategy, and fight strategy, as well as by deciding when to stop execution.

The selection of parameter settings for SVs is of fundamental concern for the performance of the search algorithm as it influences the balance between exploring the search space for new design alternatives, and exploiting the best design alternatives that are already obtained.

Starting with De Jong in 1975, Grefenstette, Schaffer and other researchers claimed to have found optimal settings for the SVs after doing long empirical tests. Unfortunately, the optimal values depend strongly on the

characteristics of the unknown search, and on the stage of the search process. It follows that it is best to modify their values dynamically. Therefore the application offers the ability to manually modify the setting of SVs. For a detailed discussion of parameter settings refer to [Schleiffer].

Because the total amount of stored information can become high, and to avoid reading information from the storage taking too much time, the path length to reach any stored data should be kept as short as possible. Linear lists do not seem to be efficient enough to comply with this goal. A better approach is it to employ trees since these can be modeled to apply shorter path lengths to attain the stored information. An important contribution to handle this task by designing a special binary tree was presented by [Adel'son-Vel'skii and Landis] in 1962. They proved that a search in one of their *avl-balanced trees* never needs more than $O(\log(\zeta))$ operations, where $\zeta \in \mathbf{N}$ is the number of nodes, in our case the number of previously stored function evaluations.

Values concerning the quality of the actual design solutions, the duration of the search, and the computational expenditure are continuously displayed in the status bar of the main window while the GA proceeds and also while design solutions are shown. The values measuring quality are *Online Performance* (the average quality of all design solutions that have been evaluated) and *Offline Performance* (the average quality of the best design solutions, one per generation). For an interpretation of these measures when comparing GAs refer to [Schleiffer]. Now, in the implemented version three quality measures are used to manipulate the population. They are the *minimum* (the quality of the worst design alternative in the present generation), the *maximum* (the quality of the best design alternative in the present generation) and the *average quality* over all designs in the present population. The implementation offers one more value, a reference value with which the others should be compared. It is called *reachable quality* and it is evaluated under the assumption that a design achieves an engineer's preference and a customer's preference of [1, 1, 1, 1] on each DV and on each PV.

Of course not one of these values alone, but only their interplay with each other and with other sources of information should conclude the search or to adjust the SVs to different values.

It would be best to show the distribution function on the search space as a whole on the screen, but this is

difficult in the case of a three dimensional *DPS* and impossible in more dimensions. The path followed here is to show the current state of the individuals' distribution on every single DV in form of graphs. This is done by drawing lines vertical to an axis representing a DV's support, with the length of such a line being proportional to the number of individuals having the value that is marked by the line's position on the horizontal axis. The PVs are presented in the same way. Consider Figure 4, which uses the example of the design of a pressurized air tank, originally taken from [Papalambros and Wilde] in the version presented by [Antonsson, Otto and Wood].

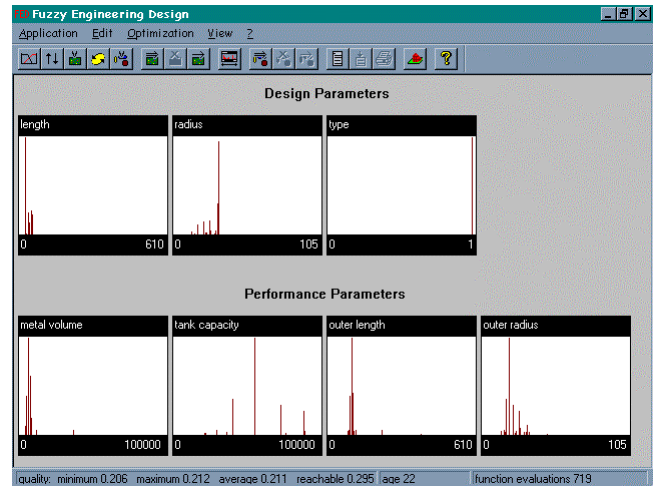


Figure 4. THE MAIN WINDOW AFTER 22 GENERATIONS.

After pressing the button *Show Results* the main window changes to show a user-defined number ($\xi \in \mathbf{N}$) of partitions $R_\chi \subset [0, 1] \subset \mathbf{R}^1$, $\chi \in \{0, \dots, \xi - 1\} \subset \mathbf{N}$ of equal length, defined as

$$R_\chi := \begin{bmatrix} quality_{min} + \frac{c}{x} \cdot (quality_{max} - quality_{min}), \\ quality_{min} + \frac{c+1}{x} \cdot (quality_{max} - quality_{min}) \end{bmatrix}$$

Each individual of the current population is assigned to that partition for which its overall quality is in R_χ . Presented on the screen are then those ranges of the DVs and of the PVs that are covered by the individuals belonging to particular partitions.

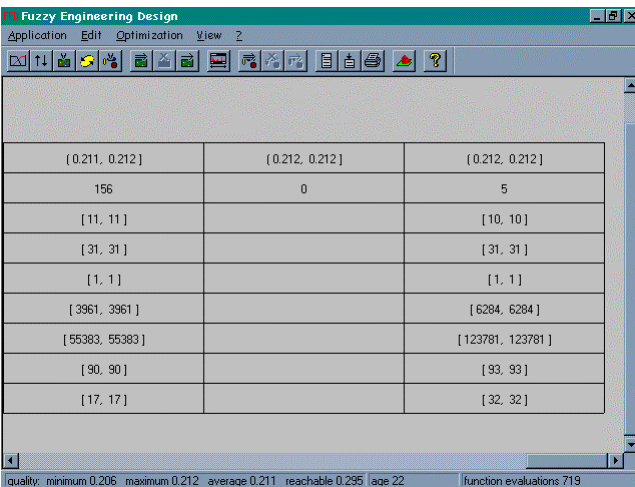
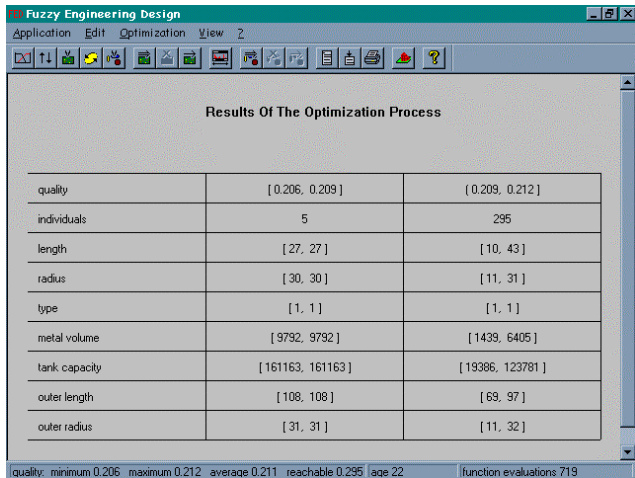


Figure 5. THE MAIN WINDOW SHOWING THE RESULTS OF THE SEARCH PROCESS.

The top screen image in Figure 5 shows two partitions after stopping the execution of the example. Observe that type “0” is no longer represented in the population. The ranges where hopeful solutions should be expected are in the case of the variable *length* limited to less than six percent of the original range. In the case of *radius* it is reduced to less than 20 percent. The utility of these partitions increases with their number, as the bottom screen image in Figure 5 exhibits. Now ξ is set to 100. The displayed table illustrates that more than 50 percent of all individuals (300 exist in the generation) are situated on only two points in the search space. And now it is the DMs’ choice to take the vector $[length = 10, radius = 31, type = hemispherical]$ as the solution of the design problem or to continue the search. Note that a solution is not always so easily derived from these partitions. It is possible that there are

conglomerations of different design solutions at several high quality positions in the search space. A possible indicator for such a situation is that the quality intervals presented as result of the optimization do not include only one solution as displayed in Figure 5, but that they contain multiple potential design solutions. To avoid letting the DMs examine and rank each of them, a Fuzzy-C Means Algorithm (FCMA) to identify representatives for groups of solutions has been implemented.

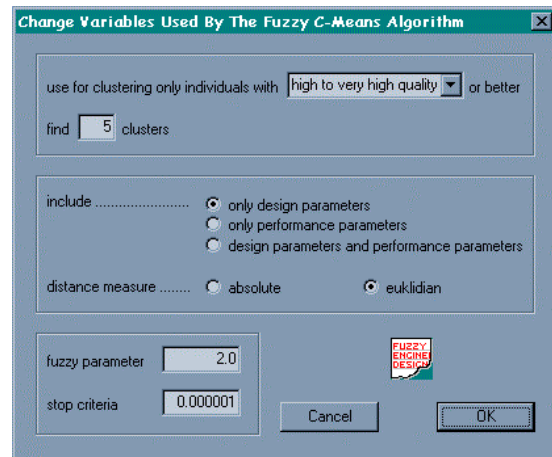


Figure 6. DIALOG TO ADJUST VARIABLES USED BY THE FCMA.

The corresponding dialog is shown in Figure 6. In its top box the DMs can modify a solution acceptance parameter. Only those design solutions with a quality higher than the minimum quality selected in the pop-up menu will be incorporated in the clustering procedure. In the same box the number of clusters to be specified has to be entered. This number should depend on the number of design solutions the DMs are willing to classify, and should be close to the optimal number of clusters derived from values in the main window while the FCMA is executed.

As soon as the clustering process is started the main window changes to what is presented as the top screen image in Figure 7. Updated after each iteration it offers several types of feedback from the FCMA’s performance. For every design solution, represented by a red point, the maximum grade to which this solution belongs to any possible cluster is portrayed. After initializing the FCMA these points are close to the broken horizontal line that represents the lower bound of the highest grade to which a solution can belong to any cluster. If a point is close to this line, the design solution that it characterizes belongs to

every cluster to nearly equal degree. The higher the points move on the screen, the closer they are to one (1), the more sharp is the clustering result and therefore the more expressive is the solution obtained by the clustering process (compare with the bottom screen image in Figure 7).

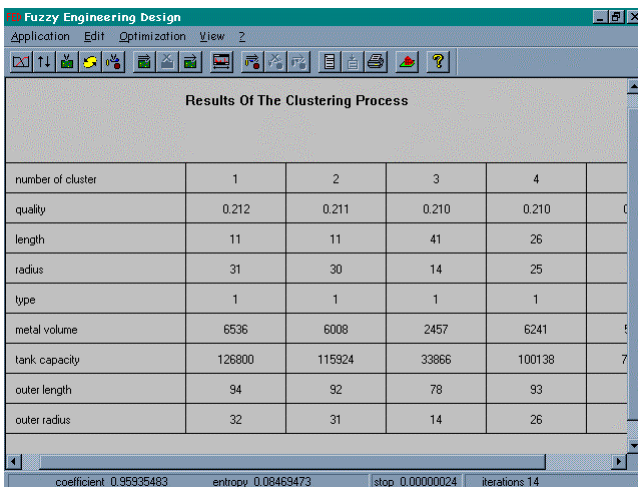
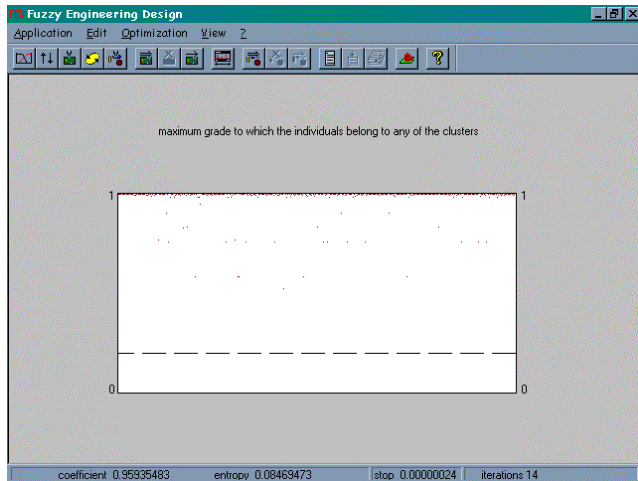


Figure 7. INFORMATION DISPLAYED ABOUT THE PROGRESS OF THE FCMA ALGORITHM AND RESULTS OF THE CLUSTERING PROCESS.

Further information about the performance of the clustering process as well as about the quality of the actual partition is presented in the status bar. It displays the number of executed iterations, the difference between the previous and the actual fuzzy partition matrix, the partition coefficient, measuring the average grade to which any object belongs to any cluster, and the partition entropy as a measure for the degree of fuzziness in the actual partition.

After the FCMA is stopped, either because of fulfillment of the stop criteria or because the user presses the *Stop* button, the main window changes to show the determined centers of the clusters. The results obtained in the tank example are displayed in Figure 7. It shows clustering centers with a quality of 0.210 and higher. This is not surprising since only these design solutions achieving a *high to very high quality* were incorporated in the process.

Note that a solution close to optimal was obtained after 719 function evaluations, or less than 0.6 percent of the efforts of an enumerated search. Depending on the results obtained by utilizing a FCMA the optimum design could be determined by two more function evaluations.

A great advantage of the implemented application is that the DMs can see directly what can be developed and where there is a risk of decision-making errors. They do not take the part of the inventor or the developer, but the much easier one of the critic. Both the engineer and the client simply have to examine the variants and to accept or to refuse them. A refusal should always lead to a modification of the preferences, or should raise the realization that the intention desired cannot be transformed into action with the basic conditions provided. Their distance from the process of design itself enables them to reason about designs in a plausible way and it makes it possible for them to concentrate on what is essential for the product development: the specification of their preferred parameter adjustment.

7 THE TEST PROBLEMS

In addition to the academic example shown in Section 6, the design tool was tested with other, more complex examples. Results of these tests are presented in [Schleiffer]. Among them were the design of a single speed power transmission for a conventional clothes washing machine (for details on the problem see [Wood and Antonsson]), the preliminary design of a ground plan of a detached family house, including a model of the laws and directives of the local authority, and the design of filter equipment as used in the pharmaceutical industry where a complex design incorporated imprecisely described parameters like diameter, cost and stability with some of the variables defined on cardinal and others on nominal scales.

8 CONCLUSION

The MoI [Antonsson] formalizes the preliminary phase in the process of engineering design using customer and designer preferences. Following the MoI, a new approach to model and combine preferences using type-two membership functions, in cases where there is not enough expertise available to develop more specific membership functions, had been presented. It was seen that the engineering design process has parallels to the mode of operation of a population based algorithm. A specific evolutionary algorithm, combined with a Fuzzy-C Means Algorithm (FCMA) to identify neighborhoods, was implemented to determine a set of good solutions within a few generations of parameter combinations and by evaluating on average only a small fraction of all possible designs. These solutions included options that a human designer might not have discovered within the time and budget constraints of the projects. If the evolutionary process is allowed to continue the algorithm may eventually converge upon a global optimum design. Depending on these solutions clusters are created and representatives of each cluster are displayed to the DMs.

A thread-based software tool, based on this methodology and equipped with a graphical user interface and several dialogs to manually monitor the progress of the GA, was demonstrated.

Finally, the application's contribution to solving real world problems was set forth, and previous work on several examples was described. The model of each example was supported by experts in the particular fields, who expect that their problem solving process can be improved by utilizing the method described.

REFERENCES

- Adel'son-Vel'skii G. M., E. M. Landis, *Doclady Akademii Nauk SSSR*, 146, English translation: *Soviet Mathematics*, 3², 1962, p. 1259 - 1263.
- Antonsson E. K., K. N. Otto, Imprecision in Engineering Design, *ASME Journal of Mechanical Design*, Special Combined Issue of the Transactions of the ASME Journal of Mechanical Design and Journal of Vibration and Acoustics, commemorating the 50th anniversary of the Design Engineering Division of the ASME, 1994, p. 25 - 32.
- Antonsson, E. K., K. N. Otto, Improving Engineering Design with Fuzzy Sets, *Fuzzy Information Engineering: A Guided Tour of Applications*, Chapter 38, John Wiley & Sons, 1997, pp. 633-654.
- Antonsson, E. K., K. N. Otto, K. L. Wood, Modeling Imprecision in Product Design, *Proceedings of the Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '94)*, invited paper, 1994, pp. 346-351.
- Bonissone, P. P., A Fuzzy Sets Based Linguistic Approach: Theory and Applications. in: Gupta M. M.; Sanchez, E.. *Approximate Reasoning in Decision Analysis*, North-Holland Publishing Company, 1982, pp. 329 - 339.
- Chen, S. J., Ranking fuzzy numbers with maximizing set and minimizing set, *Fuzzy Sets and Systems*, Vol. 17, No. 2, 1985, p. 113 - 129.
- Chen, S. J., Hwang C. L., *Fuzzy Multiple Attribute Decision Making: Methods and Applications*, Lecture Notes in Economics and Mathematical Systems Vol. 375. Berlin/Heidelberg, Germany, 1992.
- Davis L., *Handbook of Genetic Algorithms*, van Nostrand Reinhold, New York, 1991.
- Dubois, D., H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.
- Fulcher A. J., P. Hills, Towards a Strategic Framework for Design Research, *Journal of Engineering Design*, Vol. 7, No. 2, p. 183 - 193.
- Goldberg D. E., J. Richardson, Genetic Algorithms with Sharing for Multimodal Function optimization, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 1987, p. 41 - 49.
- Goldberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Menlo Park, CA, 1989.
- Goldberg D. E., Zen and the Art of Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, Schaffer J. (ed.), Morgan Kaufman Publishers, Los Altos, CA, 1989, p. 80 - 85.
- Law W. S., E. K. Antonsson, Implementing the Method of Imprecision: An Engineering Design Example, *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, Vol. 1, Orlando, Florida, 1994, p. 358 - 363.
- Law W. S., E. K. Antonsson, Including Imprecision in Engineering Design Calculations, *Proceedings of the Sixth ASME Conference on Design Theory and Methodology*, Minneapolis, 1994.
- Law, W. S., *Evaluating Imprecision in Engineering Design*, Ph.D. Thesis, California Institute of Technology, 1996.
- Law W. S., E. K. Antonsson, Multi-dimensional mapping of design imprecision, *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, California, 1996.
- Otto, K. N., E. K. Antonsson, Trade-Off Strategies in Engineering Design, *Research in Engineering Design*, Springer-Verlag, New York, 1991, p. 87 - 102.
- Otto K. N., E. K. Antonsson, Tuning Parameters in Engineering Design, *Journal of Mechanical Design*, Vol. 115, 1993, p. 14 - 19.
- Otto, K. N., E. K. Antonsson, Design Parameter Selection in the Presence of Noise, *Research in Engineering Design*, Springer-Verlag, London, 1994, p. 234 - 246.
- Papalambros, P., D. Wilde, *Principles of Optimal Design*, Cambridge University Press, New York, 1988.
- Schleiffer, R., *Methode von Chen und Hwang für Fuzzy MADM Probleme*, Referat, RWTH Aachen, Germany, 1995.
- Schleiffer, R., *Fuzzy Engineering Design*, Final Report. RWTH Aachen, Germany; CalTech, USA, 1996.
- Schleiffer, R., *Development of an Intelligent Tool in Fuzzy Engineering Design*, Diploma Thesis. RWTH Aachen, Germany, 1998.
- Sebastian, H.-J., E. K. Antonsson, *Fuzzy Sets in Engineering Design and Configuration*, Kluwer Academic Publishers, Boston, London, 1996.
- Srinivas M., L. M. Patnaik, Genetic Algorithms: A Survey, *IEEE Computer*, Vol. 27, No. 6, 1994, p. 17 - 26.
- Scott, M. J., E. K. Antonsson, Aggregation Functions for Engineering Design Trade-Offs, *Design Theory and Methodology - DTM'95*, ASME, 1995.
- Wood K. L., E. K. Antonsson, Computations with Imprecise Parameters in Engineering Design: Background and Theory, *Transactions of the ASME*, Vol. 111, 1989, p. 616 - 625.
- Wood K. L., E. K. Antonsson, Modeling Imprecision and Uncertainty in Preliminary Engineering Design, *Mechanism and Machine Theory*, Vol. 25, No. 3, 1990, p. 305 - 324.
- Wood K. L., E. K. Antonsson, J. L. Beck, Representing Imprecision in Engineering Design: Comparing Fuzzy and Probability Calculus, *Research in Engineering Design*, Springer-Verlag New York Inc., 1990, p. 187 - 203.
- Wood, K. L., K. N. Otto, E. K. Antonsson, Engineering Design Calculations with Imprecise Parameters, *Fuzzy Sets and Systems*, Vol. 52, North Holland, 1992, p. 1 - 20.
- Zimmermann, H.-J., *Fuzzy Set Theory and Its Applications*, Kluwer-Nijhoff Publishing, Boston Dordrecht Lancaster, 1988.