

**This is the author version of an article published as:**

**Zhang, Jinglan and Pham, Binh L. and Chen, Yi-Ping Phoebe (2001)  
Fuzzy Genetic Algorithms with Fuzzy Solutions. The Australian  
Journal of Intelligent Information Processing Systems 7(1-2):pp. 1-  
10.**

**Copyright 2001 (please consult author)**

**Accessed from <http://eprints.qut.edu.au>**

# Fuzzy Genetic Algorithms with Fuzzy Solutions

*Jinglan Zhang, Binh Pham and Phoebe Chen*

*Faculty of Information Technology  
Queensland University of Technology  
GPO Box 2434 Brisbane QLD 4001 Australia  
([jinglan.zhang, b.pham, p.chen@qut.edu.au](mailto:jinglan.zhang, b.pham, p.chen@qut.edu.au))*

**Abstract:** *Many decisions need to be made based on imprecise or incomplete initial information. In such cases, decision makers are generally more interested in some sets of the most promising solutions rather than the best single solution. Therefore, in contrast to conventional optimisation approaches that aim to find exact optimal points, we aim to find optimal ranges with variable satisfaction degrees. This paper presents a fuzzy-set-based approach for representation and optimisation of practical problems with imprecise properties where evolutionary computation is used to obtain fuzzy solutions through guided searching. The representation of fuzzy sets, their initialisation, crossover, mutation, and validation, the ranking approach for fuzzy objective values, and the propagation method of fuzzy information are discussed. Several examples for illustrating the fuzzy evolutionary optimisation approach are provided.*

**Keywords:** fuzzy optimisation, fuzzy genetic algorithms, propagation of imprecise information.

## 1 Introduction

Many problem-solving activities, such as product design or operational research, can be considered as an optimisation process that aims to find the best solutions to fulfil certain constraints. Since human perception is intrinsically imprecise and subjective, and in some cases only incomplete information is available when making a decision, many decisions need to be made based on imprecise or incomplete initial information. In addition, when exploring possible solutions, decision-makers are generally more interested in some sets of the most promising solutions rather than the best single solution. Hence, it is desirable to seek for a solution to contain set-based information. A set-based solution is more natural, efficient and robust than a crisp solution owing to its multiple choices, especially when imprecise data are involved.

The fuzzy set approach that was introduced by Zadeh [29] is particularly suitable for handling imprecise information by providing a set of solutions with different degrees of preference. We therefore choose this approach to

address the imprecise and set-based optimisation problem. A fuzzy set of solutions is not an arbitrary group of elements because different members in a fuzzy set have similarities to some extent and the differences between members are measurable. Since fuzzy sets may be viewed as the generalization of intervals and crisp values [23], fuzzy-set-based optimisation is a generalization of interval-based optimisation and conventional crisp-value-based optimisation.

Many papers have discussed fuzzy optimisation problems using fuzzy coefficients, fuzzy constraints, or fuzzy objective functions. However, almost all existing fuzzy optimisation methods use crisp solutions [10, 20, 22, 25]. Only a few attempts consider fuzzy solutions [2, 5, 6, 28]. Buckley et al. [5] employed a fully fuzzified genetic algorithm to perform operational research. This approach uses the arbitrary-shaped fuzzy sets to get approximate solutions to a fuzzy optimisation problem. This approach is simple, but the shape of the resultant fuzzy set is not reasonable because it is not convex. Buckley et al. [6] implemented an evolutionary algorithm to solve linear programming problems where only the triangular-shaped fuzzy sets are used and good approximate solutions are produced. In both approaches, the traditional fuzzy arithmetic based on the extension principle is used to propagate fuzzy information. This approach utilises natural set-based information and is suitable for general fuzzy optimisation problems. However, it is computationally expensive and hence is impractical in optimisation problems of large dimensions. Antonsson [2] performed a calculus-based optimisation, in particular, a *one-at-a-time* search which explores all directions one by one [1], while using preference specifications as fuzzy constraints and the Level Interval Algorithm for fuzzy information mapping. This approach relies on gradient information and is suitable only for uni-modal problems that have only one minimum/maximum within a certain range. Scott [27] followed Antonsson's direction but used *pattern search* [1] to replace the one-at-a-time search in order to avoid the calculation of gradient information. Sebastian and Schleiffer [28] used a crisp genetic algorithm to search the most promising solutions and a fuzzy clustering algorithm to group the nearby crisp solutions into fuzzy solutions. This

approach can solve multi-modal problems but it has the difficulties in defining an appropriate number of groups.

Random search methods, such as genetic algorithms [7, 13], can solve both uni-modal and multi-modal functions since the setting of search points is based on genetic and evolutionary principles, rather than on the uni-modal hypothesis. Hence, we choose to use Genetic Algorithms (GAs) to perform the optimisation task.

The combination of GAs and Fuzzy Systems has been explored for many years [11], but most cases deal with simply coupled systems. They either use GAs to design fuzzy systems, such as optimising membership functions or extracting fuzzy inference rules, or use a fuzzy system to tune GAs parameters such as population size, crossover rate and mutation rate. Systems that integrate these two concepts are rare [14]. The fuzzy genetic algorithms with fuzzy solutions are integrated systems because the representation, crossover, mutation and selection are all based on the concept of fuzzy sets. We call these algorithms Fuzzy Genetic Algorithms (FGAs) because each individual (chromosome) is composed of a set of membership functions whose values are between 0 and 1. They are essentially fuzzy-set-coded genetic algorithms. The plural form of the name is used here because some components of FGAs have multiple choices and each option corresponds to a single fuzzy genetic algorithm.

This paper aims to investigate techniques for implementing fuzzy genetic algorithms to perform fuzzy optimisation with or without fuzzy constraints. In particular, we concentrate on the fuzzy representation, fuzzy genetic operations, fuzzy evaluations and orderings of fuzzy objects that are described by fuzzy variables. Section 2 discusses the features of all components of FGAs. Section 3 then provides several possible application cases where the FGAs are applied to perform fuzzy optimisation. Section 4 provides implementation details of the FGAs. Finally, the conclusions and future work are discussed in section 5.

## 2 Fuzzy Genetic Algorithms

To use conventional GAs to solve an optimisation problem, we need to consider six fundamental issues: chromosome representation, genetic operators (crossover and mutation), selection strategy, initialisation scheme, termination criteria and the evaluation function [16]. In the FGAs, we also need to consider the validation of the generated fuzzy sets, the evaluation of fuzzy functions and the ordering of fuzzy sets. These issues are discussed in the following subsections.

### 2.1 Representation strategies of fuzzy sets

We assume that all variables in a fuzzy optimisation problem have fuzzy set values whose membership degrees are real numbers between 0 and 1. The  $\alpha$ -cut of a fuzzy set is the subset in which the membership grades of all elements are equal to or greater than the value  $\alpha$ . If the membership grades of all elements in the subset is greater than the value

$\alpha$ , it is called a strong  $\alpha$ -cut. In fuzzy set theory, a family of  $\alpha$ -cuts or strong  $\alpha$ -cuts can uniquely define a fuzzy set [19].

A fuzzy set is represented by a membership function that may be continuous or discrete, in which each element has a value and a corresponding membership grade. There are basically two ways to represent a membership function: the discrete representation which is the union of all singletons in a fuzzy set and the continuous functional form in calculus [3]. For example, a fuzzy set  $A$  (Figure 1(a)) can be represented by  $A = \bigcup_{x_i \in X} \mathbf{m}_A(x_i) / x_i = \{0.0/0.0, 1.0/1.0,$

$0.0/2.0\}$ . The same triangular-shaped membership function can also be represented in a continuous calculus form,

$$A = \{(x, \mathbf{m}_A(x))\}, x \in X,$$

$$\mathbf{m}_A(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 2 - x & 1 \leq x \leq 2 \\ 0 & x > 2 \end{cases}$$

where  $X$  is the universe of discourse,  $x$  is the value of an element, and  $\mathbf{m}_A(x)$  is the membership degree to which  $x$  belongs to the fuzzy set  $A$ . Since the functional representation cannot represent an arbitrary-shaped membership function that will be used in the fuzzy genetic algorithm, we employ the discrete representation of fuzzy sets. The discrete representation approaches of real number fuzzy subsets can be further classified into the following three classes: piecewise linear distribution, equal interval distribution and family of  $\alpha$ -cuts.

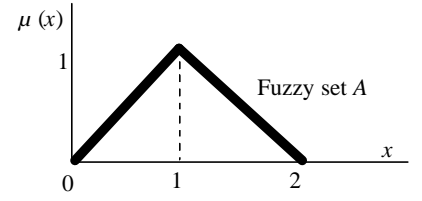
The piecewise linear representation uses a few corner points to represent piecewise linear membership functions such as triangles or trapezoids. The membership values for points that are not vertices can be obtained by interpolation. In this representation, a fuzzy set is represented by pair-wise coordinates representing the locations and membership grades of points of the fuzzy set within a certain universe of discourse. For example, the triangular-shaped fuzzy set  $A$  can be represented as  $A = \{0.0/0.0, 1.0/1.0, 0.0/2.0\}$  (Figure 1 (a)). This approach can exactly represent a piecewise linear-shaped fuzzy set using a few data points, but both of the locations and membership degrees of all corner points must be explicitly represented. Hence, it is not suitable for representing an arbitrary-shaped fuzzy set, where a large number of data points are needed. In the fuzzy genetic algorithm, this representation is used for specifying fuzzy sets in defining the initial population or fuzzy constraints because this is the simplest way to input rough data.

The equal interval distribution approach represents a continuous fuzzy set using equally distributed discrete points (Figure 1 (b)). For example, if the domain of a variable is [0, 10] and the number of interval is 5, then each interval is 2, this fuzzy set can then be represented by the piece wise-linear representation  $B = \{0.0/0, 0.4/2, 0.3/4, 1.0/6, 0.5/8, 0.0/10\}$ . When the universe of discourse and the number of intervals

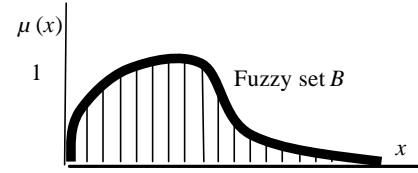
of a fuzzy set are fixed during the optimisation process, the location of each point is fixed. Hence, the location information need not be represented and a fuzzy set can be represented by a vector of membership grades. Therefore, the previous fuzzy set is represented in equal interval representation as  $B = \{0.0, 0.4, 0.3, 1.0, 0.5, 0.0\}$ . The equal interval representation is suitable for representing an arbitrary-shaped fuzzy set because only the membership grades are represented. So, the equal interval representation is used for randomly generating new fuzzy sets and encoding fuzzy sets into genetic algorithms.

We call the horizontal line, whose distance to the horizontal axis is  $a$ , an *a-level line* and the intersection points of an  $a$ -level line with a fuzzy set *a-cut end points* (Figure 1 (c)). The family of  $a$ -cuts ( $a = 0$ ) or family of strong  $a$ -cuts ( $a > 0$ ) represents fuzzy sets using  $a$ -cut endpoints. When the  $a$ -cut values are predefined and fixed, they need not be represented explicitly in a fuzzy set. Hence, a fuzzy set is represented by a set of pairs of  $a$ -cut end points. For example, a fuzzy set  $C$  (Figure 1 (c)) has a closed support, i.e. the membership degrees for all elements that are out of the range of the support are zero. If the fuzzy set  $C$  is represented by five  $a$ -levels,  $a = \{0.001, 0.2, 0.4, 0.6, 0.8, 1.0\}$ , then the fuzzy set  $C$  can be represented by the two endpoints (boundary points) at all  $a$ -levels. In the cases that the two endpoints converge to one point such as the point with the highest membership degree in fuzzy set  $C$ , the value of this single point is repeated to provide a uniform representation of arbitrary-shaped fuzzy sets. Fuzzy set  $C$  is then represented by five pairs of endpoints representing five  $a$ -levels,  $B = \{(1.1, 3.8); (1.2, 3.5); (1.4, 3.0); (1.8, 2.6); (2.1, 2.1)\}$ . The major advantage of  $a$ -cuts or strong  $a$ -cuts is that they bridge between fuzzy sets and crisp sets. Some properties of crisp sets can be extended to fuzzy sets using  $a$ -cuts (or strong  $a$ -cuts). The strong  $a$ -cuts of fuzzy sets can define the support of a fuzzy set which is very important in fuzzy arithmetic, but the  $a$ -cuts have no such property. The strong  $a$ -cuts are therefore used for propagating fuzzy information in arithmetic calculation based on interval arithmetic [18]. The strong  $a$ -cuts are also used for smoothing an arbitrary-shaped fuzzy set to make it  $a$ -convex, so that the result of any strong  $a$ -cuts of the fuzzy set are connected sets. This smoothing process for fuzzy set validation will be discussed in a later section. This family of  $a$ -cuts representation scheme is suitable for representing fuzzy sets with closed-interval supports. Since a fuzzy optimisation problem deals with fuzzy subsets in the universe of discourse (i.e. the value ranges of variables) of real numbers whose support must fall in a certain range, this assumption is reasonable.

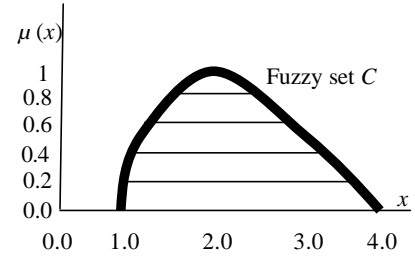
These three discrete representations have a common characteristic in that they represent fuzzy sets using discrete points. The only difference between them lies in the arbitrary or regular relationship between these discrete points. Hence, they can be transformed from one form to another. We can therefore employ these three representation approaches in one genetic algorithm to suit different purposes.



(a) Piece-wise linear function.



(b) Equal interval.



(c) Family of  $a$ -cuts.

**Figure 1. Fuzzy set representation.**

## 2.2 Encoding of individuals in FGAs

Since fuzzy genetic algorithms aim to deal with fuzzy optimisation problems with fuzzy solutions in the universe of discourse of real numbers, each fuzzy set is encoded using an equal interval representation. Each gene is a fuzzy set represented by a vector of membership grades that are real numbers between 0 and 1. Each individual (chromosome) is a concatenated string of fuzzy sets which are the fuzzy values of a set of variables. For example, we consider an optimisation problem which has two parameters with fuzzy values to be optimised. If the two fuzzy values are two fuzzy sets  $A$  and  $B$  and each of them is represented by an equal interval possibility distribution that has five elements,  $A = \{0.1, 0.4, 1.0, 0.3, 0.0\}$  and  $B = \{0.2, 0.9, 1.0, 0.6, 0.0\}$ , then the chromosome composed of these two fuzzy sets is represented by  $I = \{0.1, 0.4, 1.0, 0.3, 0.0, 0.2, 0.9, 1.0, 0.6, 0.0\}$ . The number of partition points of each fuzzy set can be the same or different depending on the precision requirements and the universe of discourse of each fuzzy set. We use ten partition points for each fuzzy set for efficiency, but without loss of generality.

We choose to use a string of real numbers for encoding objects in genetic algorithms because they are natural and

concise for continuous real values. However, the crossover and mutation operation will be modified to suit the features of fuzzy set representation. Relevant issues will be discussed later.

## 2.3 Initialisation scheme

As GAs use random searching, except for the definition of the problem, they do not need much knowledge from humans. However, the incorporation of human knowledge or of existing rough optimum results from other optimisation approaches, can definitely improve and speed up the evolution process in GAs. Thus, existing knowledge is often applied to the initialisation step to add more deterministic knowledge into this random optimisation process.

Initialisation approaches can be classified into three categories: random initialisation, deterministic initialisation and a combination of random and deterministic initialisation. In random initialisation, all individuals are sampled randomly within the valid domain. This is the most popular initialisation approach which utilizes the power of automatic search to the extreme extent. However, the searching process may be slow, especially when the search space is very large. The deterministic (or heuristic) initialisation approach assigns all individuals to preferred solutions, and usually all good schemata have at least one representative in the initial population. This approach can incorporate expert knowledge or good results from other optimisation methods into the random searching process of GAs. It can speed up the searching process, but easily leads to premature convergence. In the combined initialisation, also called random sampling with dopes [12], one part of the individuals are generated according to existing preferences while the other part of individuals are generated randomly. It therefore combines and moderates the features of the other two initialisation approaches.

We combine these three initialisation schemes into one unified representation. The initial population consists of two parts: a deterministic subpopulation and a random subpopulation, while the population size remains constant. The size of the random subpopulation is the difference between the population size and the size of the deterministic subpopulation. It is a random initialisation when the size of deterministic subpopulation is zero, whereas, when the size of deterministic subpopulation is equal to the population size, it becomes a deterministic initialisation. In all other cases, it is a combined initialisation.

## 2.4 Crossover and mutation strategies

The crossover and mutation represent the change in both the location and shape of the membership functions of fuzzy sets in certain domains. Since fuzzy sets are encoded into a string of real numbers between 0 and 1, the FGAs share both the features of the Binary-Coded Genetic Algorithms (BCGAs) and the Real-Coded Genetic Algorithms (RCGAs).

In the BCGAs, the values of basic units are 0 or 1 (digits) which are not related to the domains of the real value they represent, so the crossover can be an arbitrary shuffling of all digits in one chromosome. However, in the RCGAs, every basic unit is a real number which is associated with a specific domain, hence we cannot shuffle a chromosome arbitrarily without consideration of the physical meanings of each value. The crossover in the RCGAs can only exchange the values in the same position in the two parents. Two popular crossover approaches for the RCGAs are discrete crossover that exchanges the two values at the same position of the two parents and intermediate crossover that interpolate between the two values in the same position of the two parents. In the FGAs, the value of each element in a chromosome is a real number, but it can only vary between 0 and 1 because it represents a membership degree to which an element belongs to a fuzzy set. On the other hand, each separate element has no specific meaning in terms of a fuzzy set. An element has meaning only when it is associated with a group of other elements that belongs to the same fuzzy set for the same variable. Hence, in the FGAs, in addition to the real value-based crossover, we can also perform crossover in the same way as that used in the BCGAs, such as single point or multiple points crossover. Therefore, we provide both arithmetic crossover and multiple-points crossover for the FGAs.

The mutation operation changes an individual by introducing new values to some genes. In the BCGAs, mutation simply flips some bits from 0 to 1 or 1 to 0 in a chromosome. In the RCGAs, mutation changes a value to another valid value randomly (random mutation) or changes the original value through adding or subtracting a certain amount gradually (non-uniform mutation). Other researchers' work shows that the former approach is simple and quick but the latter approach behaves better in terms of searching result [15]. We implement the random mutation to perform simple and quick search. We also implement a mutation method similar to the flip operation in the BCGAs. If the current value for a digit is  $m$  the new value is  $1-m$ . When  $m$  is 0 or 1, this mutation degenerates to flip operation. More information on different mutation approaches can be found in [15, 21].

All crossover and mutation approaches mentioned above generate fuzzy sets with arbitrary-shaped membership functions which are not meaningful in common sense. Hence, we need an approach to make the newly generated fuzzy sets meaningful. The process of validation checking which will be discussed in the next section performs this task.

## 2.5 Validation checking of generated fuzzy sets

Since the shape of a membership function characterises the uncertainty of the corresponding fuzzy variable, it should not be arbitrary and must not be defined in detail [3]. In the fuzzy genetic algorithm, we assume that a fuzzy set is valid if it is normal, monotonic, and a-convex. For example, the

fuzzy set shown in Figure 2 is a valid fuzzy set while the fuzzy set in

Figure 3 is invalid. By a normal fuzzy set, we mean its highest membership grade is 1 and lowest membership grade is 0. If the fuzzy set is not normal, it may cause problems in the calculation of  $\alpha$ -cuts. Fuzzy sets can be used to group data that share some similarities. When we define a fuzzy set, we wish at least one element in the universe of discourse totally belongs to this set while some elements do not belong to it. Hence, this assumption is practical. By monotonic, we mean for each element, only one membership grade exists, otherwise it is not meaningful. By  $\alpha$ -convex, we mean each of the  $\alpha$ -cuts of a fuzzy set is connected (every  $\alpha$ -level line has at most two endpoints).

If the arbitrarily generated fuzzy set is not normal, we need first to normalize it by making the lowest membership value to be zero and scale all elements to make the highest membership value to be 1. If it is not monotonic, we select the highest membership grade for each element to make the fuzzy set monotonic. If the fuzzy set is not  $\alpha$ -cut convex, such as the set shown in Figure 3 we consider only the minimum and maximum  $\alpha$ -cut end points for each  $\alpha$  level. We call this treatment the “ $\alpha$ -level smoothening” of the fuzzy set and the whole procedure of processing fuzzy sets “validation checking”.

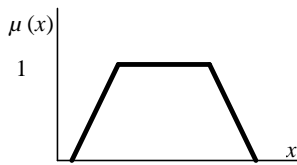


Figure 2 Fuzzy set with normal and  $\alpha$ -convex shape.

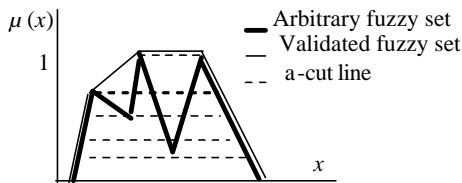


Figure 3. Fuzzy set with arbitrary shape and its validation.

## 2.6 Evaluation of individuals

In genetic algorithms, the evaluation of individuals is usually performed by the evaluation of a fitness function which measures how good each individual is. Human evaluation is necessary when the fitness function is very hard to define and relies on interactive techniques [26]. We use the fitness function as the main evaluation approach, but we also allow users to evaluate the individuals and control the evolution process through a Graphical User Interface. Best

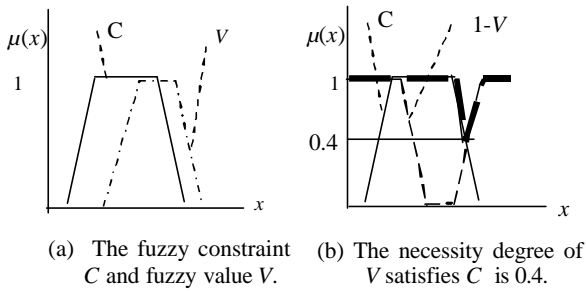
solutions in each generation and statistical data during evolution such as best and average fitness values are displayed on the screen, so users can examine their feel to the data. A user can stop the searching process and the current results are saved in a log file.

We now discuss the approaches for evaluating fuzzy fitness functions, because it plays the main role in individual evaluation in the fuzzy genetic algorithms. In a constrained multiple objectives optimisation problem, the evaluation of an individual consists of three parts: the evaluation of a set of objectives, the evaluation of a set of constraints and the aggregation of these evaluation results to form an overall fitness value.

We first discuss the evaluation of a fuzzy function which has fuzzy variable values. The evaluation of a crisp function has been well developed but the evaluation of a fuzzy function still requires further research. There are two basic approaches for doing fuzzy arithmetic. One is based on the interval arithmetic principle through the  $\alpha$ -cut approach, while the other is based on the extension principle in fuzzy logic through convolution [18]. We choose the LIA [2] which uses the simple  $\alpha$ -cut approach to calculate fuzzy function values because it is simpler and quicker than the conventional approach which is based on the extension principle [18]. In the LIA, multiple dimensional fuzzy information mapping is based on the idea that the function surface is determined by the endpoints of  $\alpha$ -cut in each dimension at each  $\alpha$ -level, where the final function value is obtained by searching the super cube composed of all  $\alpha$ -cut endpoints. When the surface function has local extrema between the  $\alpha$ -cut endpoints of one or more directions, the local extrema must be found first through a crisp optimisation method. The final range values for the fuzzy function at an  $\alpha$ -level are the minimum/maximum of the local extrema and the  $\alpha$ -cut endpoints. Since the LIA uses a predefined number of  $\alpha$ -cuts to compute the fuzzy function value, it is suitable only for normal, monotonic, and  $\alpha$ -convex membership functions. The LIA is suitable only for normal, monotonic and  $\alpha$ -convex membership functions. In GAs however, after the genetic operations (crossover and mutation), the newly generated membership functions have arbitrary shapes, so we need to perform *validation checking* (which was discussed previously) to ensure the generated fuzzy sets are valid. More details on the LIA can be found in [2].

We now discuss the evaluation measure for fuzzy constraints. Assume a variable has a real datum and a constraint exerted upon it. When the datum is a crisp value and the constraint is a crisp or an interval value, the answer for whether the datum satisfies the constraint is yes (1) or no (0). When the datum is a crisp value while the constraint is a fuzzy value, the result for judging whether the datum satisfies the constraint is a degree value which varies between 0 and 1. When both the datum and the constraint are fuzzy values, the degree to which the datum satisfies the constraint varies for different elements in the datum set. In order to obtain a definite solution of whether the datum satisfies the constraint, a special measure needs to be

defined. As the fuzzy set value of a variable can be interpreted as the values it may take with different possibilities, we use the general *possibility degree* and *necessity degree* in possibility theory to measure the degree to which a datum satisfies a constraint [4, 23, 24]. The possibility degree is the degree to which a datum possibly satisfies a constraint and the necessity degree is the degree to which a datum necessarily satisfies a constraint. Since something that necessarily happens must be possible, the necessity degree is always less than or equal to the possibility degree. The necessity degree is therefore chosen to measure the extent to which a datum satisfies a constraint. When the necessity degree is 1, we can ensure that all elements in the datum set are within the fuzzy set of the constraint with satisfaction degree 1. The necessity degree of the constraint  $C$  and the datum set  $V$  is shown in Figure 4 where the membership functions of both fuzzy sets take trapezoidal shapes. The equation for calculating the necessity degree can be found in [4, 23].



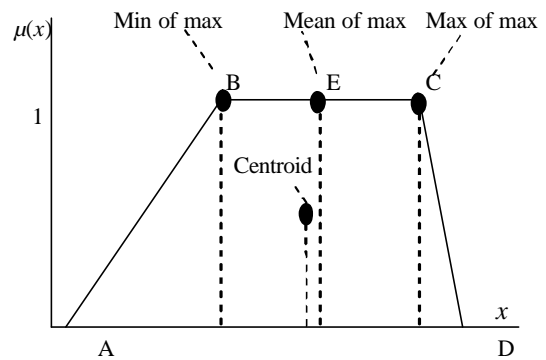
**Figure 4. Fuzzy constraint and satisfaction degree.**

A constrained optimisation problem can usually be converted into unconstrained problems using penalty functions. If the necessity degree to which a value satisfies a constraint is  $\mu$ , then the penalty for the value violating the constraint is  $1-\mu$ . If there are multiple constraints exerted on the same variable, the maximum of all separate penalties can be used as the overall penalty for the corresponding variable. Multiple penalties for different variables can be aggregated together and integrated into the fitness function by the *simple additive weighting* approach [10].

## 2.7 Fuzzy ranking, selection and reproduction of individuals

The fuzzy optimisation problem aims to find a fuzzy value to maximize or minimize the value of a fuzzy function. Since the fuzzy function value is a fuzzy set, we cannot optimise it directly because fuzzy sets have only partial order. The minimum, maximum, and the distance between two fuzzy sets are also fuzzy sets [18, 19]. This is a major difference between fuzzy data that are partially ordered and crisp data that are totally ordered. Hence, to get a total ordering of fuzzy sets, a crisp measure must be defined and

then the optimisation of a fuzzy set value is transformed to the optimisation of crisp measure [5]. The crisp measure for ordering fuzzy sets can be defined in many different ways [10]. It is usually a scalar representative of the fuzzy set to be measured. For example, the commonly-used scalar representatives of a fuzzy set include its centroid (centroid), the minimum value of all values with maximum membership degree (min of max), the maximum value of all values with maximum membership degree (max of max) and the mean value of all values with maximum membership degree (mean of max) (Figure 5). These scalar measures differ on their locations within a fuzzy set. The selection of a crisp measure to defuzzify or to order fuzzy sets is usually based on the requirements of the applications.



**Figure 5. Popular measures for ordering fuzzy sets.**

Since the centroid of gravity of a fuzzy set proposed by Yager [10] is the most simple and popular measure to defuzzify a fuzzy set, we use this measure to perform a quick test of the fuzzy genetic algorithms. As the centroid of gravity uses only one point of a fuzzy set to measure the whole set, it sometimes gives irrational results. To make the comparison of two fuzzy sets more accurate, we combine both the spread of a fuzzy set and its centroid to define the measure for ordering two fuzzy sets using the simple additive weighting approach [10]. The spread of a fuzzy set is the distance between the two  $\alpha$ -level endpoints when  $\alpha = 0$  and describes the specificity (or fuzziness) of a fuzzy set. The underlying assumption of this ordering approach is that the fuzzy set with a higher centroid value and reasonable specificity is better. The equations for calculating the centroid of a fuzzy set can be found in [3]. Since this approach considers both the location of a fuzzy set and its spread, it is more logical than the single point measure such as the centroid method.

Once the fuzzy ranking approach is defined, the selection of individuals from current population and the reproduction of new population is the same as that in a conventional genetic algorithm. For simplicity, we choose the proportional selection scheme which chooses parent individuals with a probability proportional to its fitness value in order to perform cross over and mutation. The commonly

used Elitism reproduction scheme which copies the best members of a generation to the next generation is used to produce the new population.

## 2.8 Stopping criteria

The evolution process will stop when a predefined maximum number of generations is reached or the average fitness value is not improved any more. The average fitness can be calculated by the Level Interval Algorithm if the fuzzy set value of fuzzy objective functions are considered. However, in this case the difference of the average fitness values between two generations is also a fuzzy distribution. A special measure to calculate the difference of two fuzzy sets needs to be defined. This approach keeps the fuzzy information as long as possible but it is time consuming. The average fitness can also be calculated according to the crisp fitness values which are the defuzzified result of each fuzzy fitness value. We implement the latter method for simplicity and for consistency with the selection and reproduction process.

## 2.9 Pseudo code

We call the genetic algorithms composed of the above discussed components and operations Fuzzy Genetic Algorithms because each individual (chromosome) is composed of a set of membership functions and all operations are adapted to this representation. They are essentially real-coded genetic algorithms. The differences between fuzzy genetic algorithms and conventional genetic algorithms are: (i) the validation (adjustment) of the newly generated membership functions; (ii) the evaluation of the objective function using LIA rather than a crisp function calculation; and (iii) the ordering of fuzzy sets based on a predefined criterion.

The general procedure of fuzzy genetic algorithms is outlined in the following pseudo-code. Assume that  $t$  denotes a generation (iteration) counter and  $IP(t) \in I^m$  is a population of  $\mu$  individuals with  $I$  as the searching space.  $P(t) \in I^l$  is a population of  $\nu$  individuals selected from  $IP(t)$ . The value of  $\nu$  can be equal to or less than  $\mu$ , i.e., the size of the selected population can be smaller. After this selection, the population size remains constant during the evolution process.  $C'(t)$ ,  $C''(t)$ , and  $C'''(t)$  indicate the intermediate populations. The evolution process is a loop of generation, evaluation and selection which is iterated until some termination criteria are satisfied Figure 6.

```

// Procedure of Fuzzy Genetic Algorithms.
// start counting the number of generations
t = 0;
// generate a population of individuals composed of
// strings of real values between 0 and 1.
Initialise (IP(t) ∈ Im);
// calculate the fitness value using fuzzy
// propagation and defuzzify the fuzzy objective
// values for ranking.
evaluate(IP(t));
// reproduction of the next generation
// according to a scale measure.
P(t) = selection (IP(t)), P(t) ∈ Il;
// calculate the average fitness value.
averageFitness(P(t));
// start the evolution loop.
while not terminate (P(t)) do
    // arithmetic or simple crossover.:
    C'(t) = crossOver (P(t));
    // arbitrary shuffling or Non-uniform or
    // real number creep mutation.
    C''(t) = mutation (C'(t));
    // validation checking to ensure the
    // generated set normal, monotonic,
    // and alpha-convex.
    C'''(t) = validation (C''(t));
    // fuzzy evaluation through Level Interval
    // Algorithm and defuzzification of fuzzy
    // objective values for ranking.
    evaluate(C'''(t));
    // reproduction of the next generation
    // according to a scale measure.
    P(t+1) = selection (C'''(t), P(t));
    // P(t+1) has μ individuals.
    // calculate the average fitness using fuzzy
    // arithmetic for checking the termination
    // condition.
    averageFitness(P(t+1));
    t = t+1;
end
// return the best solutions which have fuzzy set
// values.
return best (P(t));

```

Figure 6. Procedure of Fuzzy Genetic Algorithms.

## 2.10 Summary

In summary, a fuzzy problem is represented by a set of parameters that have fuzzy set values denoted by equal interval possibility distributions. All these fuzzy sets are concatenated together to form an individual in the fuzzy real-coded genetic algorithms. The searching process starts from some predefined individuals or randomly generated initial population or the combination of both. The FGAs exploit the whole valid domain through guided searching by crossover and mutation operations. The shapes of membership functions are arbitrarily generated but validated by a checking process to comply with the general meaning of a fuzzy set. Since the variables in the objective function have fuzzy set values, the individuals are evaluated through a fuzzy information propagation algorithm called the Level Interval Algorithm. The ordering of individuals is based on a predefined scale that measures the largeness of a fuzzy set. The selection of parents for mating and the reproduction of the next generation are also based on the selected ranking approach.

## 3 Implementation

Existing research shows that the real-coded genetic algorithms are more natural, concise and precise than binary-coded genetic algorithms in the domain of continuous real values [15]. Since each fuzzy set is represented as a vector of real values indicating its membership grades, we implement the fuzzy genetic algorithms within the environment of real-coded genetic algorithms. The encoding and decoding approach is modified to cater for the representation of fuzzy sets. The crossover operator and the mutation operator are implemented based on the property of fuzzy set values which are always between 0 and 1. In particular, the crossover and mutation operators for both binary and real-coded genetic algorithms are integrated together. A fuzzy information propagation module based on the Level Interval Algorithm is constructed. A crisp measure for ordering fuzzy sets is designed and embedded in the evaluation process. The implementation is performed within MATLAB because of its powerful numerical computation ability and advanced graphics functions. Some source codes of several free software packages, including GAOT (the Genetic Algorithm Optimisation Toolbox) developed by Houck et al [16, 17] and ICPT (the Imprecise Constraints Propagation Tool) developed by Chen [8, 9], are used and modified to suit the special purpose of fuzzy-set-based optimisation. A Graphical User Interface is developed to interact with users and draw the membership functions of fuzzy sets.

The next section provides several applications of the fuzzy genetic algorithms to perform fuzzy optimisation.

## 4 Case Studies

The advantage of the fuzzy genetic algorithms presented is that they provide fuzzy solutions with variable degrees, hence they are suitable for solving fuzzy optimisation problems with fuzzy variables, with or without fuzzy constraints. We first use a simple mathematical example to show how it works. Assume the optimisation problem is to maximize  $Y$ , and  $Y = X * (1-X)$ , where  $X$  is a fuzzy variable within the range  $[0, 1]$ ,  $Y$  is a fuzzy function value. Since the value for  $X$  is a fuzzy set, the value for  $Y$  is also a fuzzy set, so we cannot maximize  $Y$  directly. We use the centroid of  $Y$  to measure its largeness. The ideal crisp solution for this problem is  $x = 0.5$  and  $y = 0.25$ , so the fuzzy solution should be near this value. Figure 7 (a) and (b) show the evolution process when the population size is 100, the number of individuals performing crossover and mutation is 30 and the crossover rate and mutation rate are 0.9 and 0.1. The best fitness value is near 1.22 which is the sum of defuzzified function value 0.22 and the compensation of the spread of a fuzzy set (1 for fuzzy sets with reasonable spreads and 0 for others). In the fuzzy solution, elements near the location 0.5 have higher membership degrees.

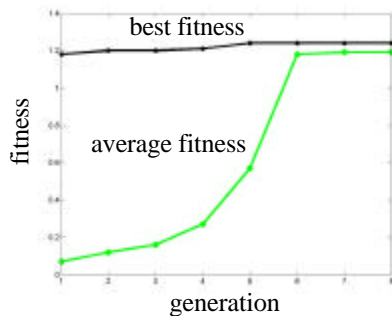
We use the objective function  $\max(\text{SurfaceArea})$  to search a solid shape with maximum 2 units in  $X, Y, Z$  direction and centred in the origin of the coordination system. A solid shape is represented by several parameters but we change only two shape parameters while the others are fixed. When these two shape parameters change from 0 to 3, the solid shapes created can be a cube, sphere, cylinder, double pyramid and any shapes in between. Intuitively, within a 2 units box the surface area of a cube is largest, which is about 24 and the surface area of a sphere is smallest, which is about 18. The fuzzy solutions found by the fuzzy genetic algorithm is shown in Figure 7 (c) where the approximate values for the two parameters are near 0 which means a shape similar to a cube is obtained.

## 5 Conclusions and Future Work

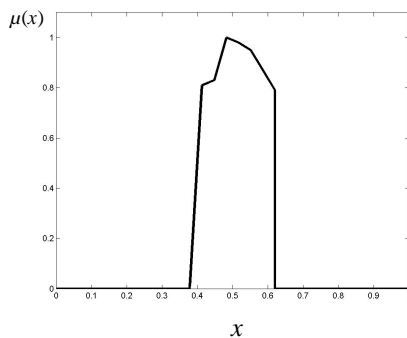
Fuzzy genetic algorithms with fuzzy solutions are essentially fuzzy-set-coded genetic algorithms. In contrast to conventional GAs that use crisp solutions, FGAs produce fuzzy solutions to a fuzzy optimisation problem through range-by-range searching. FGAs can be implemented within the scheme of a real-coded genetic algorithm. The crossover and mutation in the FGAs share both the features of that in the binary-coded GAs and real-coded GAs. The experiment results show that FGAs can produce good approximate solutions. However, we also notice that a crisp or an interval solution may be obtained depending on the definition of the fitness function and the ordering measure of fuzzy sets. The incorporation of the spread of fuzzy sets into the fitness function is important for maintaining reasonable fuzziness. The validation checking process of fuzzy sets that are generated randomly in FGAs is necessary and improves the searching process but it is computationally expensive. When

the searching dimension is fixed, the evaluation time and precision of the LIA are dependent only on the quantisation of the interval  $[0, 1]$ , which is the valid range for all membership functions. Therefore, the LIA is quicker than the conventional approach where the quantisation of all variable ranges must be considered. The LIA for fuzzy information propagation is feasible but is suitable only for low dimension searching problems. When the searching dimension is very high, the LIA is too time-consuming to be practical.

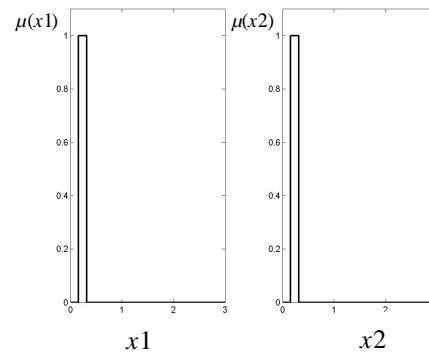
Future work includes investigating effective methods for generating valid fuzzy sets within the genetic algorithms to reduce computation cost during the validation checking process. More suitable approaches for fuzzy set ordering need further investigation, as none of the existing approaches are perfect. The multiple additive weighting approach, which integrates multiple scalar measures of a fuzzy set to form a crisp measure, is feasible, but the adjustment of the weights of each component is time-consuming. More effective approaches for fuzzy information propagation will be further explored, since the current LIA approach is time-consuming when the searching dimension is high.



(a) Best and average fitness over generations.



(b) Membership function for fuzzy set solution.



(c) Membership functions for optimal shape parameters.

Figure 7. Fuzzy optimisation examples.

## 6 References

- [1] P. R. Aday and M. A. H. Dempster, *Introduction to Optimization Methods*. London New York: Chapman and Hall ; Halsted Press, 1974.
- [2] E. K. Antonsson and K. N. Otto, "Improving Engineering Design with Fuzzy Sets," in *Fuzzy Sets in Engineering Design and Configuration*, H.-J. Sebastian and E. K. Antonsson, Eds.: Kluwer, 1996, pp. 1-52.
- [3] R. C. Berkan and S. L. Trubatch, *Fuzzy System Design Principles*. NY: IEEE Press, 1997.
- [4] P. Bosc and M. Galibourg, "Indexing Principles for a Fuzzy Data Base," *Information Systems*, vol. 14, pp. 493-499, 1989.
- [5] J. J. Buckley and Y. Hayashi, "Fuzzy Genetic Algorithm and Applications," *Fuzzy Sets and Systems*, vol. 61, pp. 129-136, 1994.
- [6] J. J. Buckley and T. Feuring, "Evolutionary Algorithm Solution to Fuzzy Problems: Fuzzy Linear Programming," *Fuzzy Sets and Systems*, vol. 109, pp. 35-53, 2000.
- [7] L. Chambers, *Practical handbook of genetic algorithms*. Boca Raton: CRC Press, 1995.
- [8] J. E. Chen, "ICPT: Imprecise Constraint Propagation Tool," 1.0 ed. Cambridge, MA. Available from <http://design.mit.edu/Software/>; Otto, K. N., 1995.
- [9] J. E. Chen and K. N. Otto, "A Tool for Imprecise Calculations in Engineering Systems," in *Proceedings of the 1995 FUZZ-IEEE Conference, Yokohama, JP.*, 1995.
- [10] S. J. Chen and C. L. Hwang, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*. New York: Springer-Verlag, 1992.
- [11] O. Cordon, F. Herrera, and M. Lozano, "A Classified Review on the Combination Fuzzy Logic-Genetic Algorithms Bibliography: 1989-

- 1995," in *Genetic Algorithms and Fuzzy Logic Systems : Soft Computing Perspectives*, vol. 7, *Advances in Fuzzy Systems*, E. Sanchez, T. Shibata, and L. A. Zadeh, Eds. Singapore ; River Edge, NJ: World Scientific Pub, 1997, pp. 209-240.
- [12] A. Geyer-Schulz, *Fuzzy Rule-based Expert Systems and Genetic Machine Learning*, 2nd rev. and enl. ed. Heidelberg: Physica-Verlag, 1997.
- [13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Mass.: Addison-Wesley Pub. Co., 1989.
- [14] F. Herrera and M. Lozano, "Fuzzy Genetic Algorithms: Issues and Models," Dept. of Computer Science and A.I., University of Granada, Granada, Spain, Technical Report DECSAI-98116, 1998.
- [15] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review*, vol. 12, pp. 265-319, 1998.
- [16] C. R. Houck, J. A. Joines, and M. G. Kay, "A Genetic Algorithm for Function Optimization: A Matlab Implementation," Department of Industrial Engineering, College of Engineering, North Carolina State University, North Carolina, Technical Report NCSU-IE TR 95-09, 1995.
- [17] C. R. Houck, J. A. Joines, and M. G. Kay, "GAOT : Genetic Algorithms Optimization Toolbox," 1.0 ed. North Carolina. Available from [http://www.eos.ncsu.edu/eos/service/ie/research/kay\\_res/GAToolBox/gaot/](http://www.eos.ncsu.edu/eos/service/ie/research/kay_res/GAToolBox/gaot/): Houck, C. R., Joines, J. A., Kay, M. G., 1995.
- [18] A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic : Theory and Applications*. New York, N.Y: Van Nostrand Reinhold Co., 1985.
- [19] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*. New Jersey: Prentice Hall, 1995.
- [20] Y.-J. Lai and C.-L. Hwang, *Fuzzy Multiple Objective Decision Making: Methods and Applications*. Berlin Heidelberg: Springer-Verlag, 1996.
- [21] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd rev. and extended ed. Berlin ; New York: Springer-Verlag, 1996.
- [22] S. Petrovic-Lazarevic and Z. Prasevic, *Fuzzy Multiple Objective Decision Making in the Construction Industry*. [Caulfield East, Vic.]: Monash University Faculty of Business and Economics, 1998.
- [23] F. E. Petry and P. Bosc, *Fuzzy Databases: Principles and Applications*. USA: Kluwer Academic Publishers, 1996.
- [24] H. Prade and C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries," *Information Sciences*, vol. 34, pp. 115-143, 1984.
- [25] K. C. Sarma and H. Adeli, "Fuzzy Genetic Algorithm for Optimization of Steel Structures," *Journal of Structural Engineering*, vol. 126, pp. 596-604, 2000.
- [26] T. Sato and M. Hagiwara, "IDSET: Interactive Design System Using Evolutionary Techniques," *Computer Aided Design*, vol. 33, pp. 367-377, 2001.
- [27] M. J. Scott, "Formalizing Negotiation in Engineering Design," in *Engineering Design Research Laboratory, Division of Engineering and Applied Science*. Pasadena, California: California Institute of Technology, 1999.
- [28] H.-J. Sebastian and R. Schleiffer, "Using Computational Intelligence in Fuzzy Engineering Design," *Cybernetics and Systems*, vol. 31, 2000.
- [29] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.