Journal of Intelligent & Fuzzy Systems 16 (2005) 107–118
IOS Press

Genetic programming of fuzzy aggregation operations

Petr Musilek*, Rhea Guanlao and Guillermo Barreiro
Department of Electrical and Computer Engineering, University of Alberta, ECERF W2-030, Edmonton, Alberta, T6G 2V4, Canada

Abstract. Aggregation operations play an important role in decision-making problems where a weighted combination of several criteria is used to select an alternative with the strongest support. In fuzzy set theory, aggregation operations are usually modeled as intersection, union, or as combination of both. The particular form and algebraic properties of these operations vary according to requirements for compensation among the criteria and other characteristics of the given decision-making situation. Traditionally, only algebraically well-behaved operations have been considered for this purpose. By relaxing some algebraic constraints, more realistic operations can be obtained that closely capture certain features of human decision-making, such as preferences and a limited level of detail.

This paper proposes a method to generate fuzzy aggregation operations using genetic programming. It is shown that an evolutionary process, facilitated by genetic programming, has the capacity to generate new valid fuzzy aggregation operations and to reproduce existing ones. By varying process conditions, encoded in a fitness function, it is possible to obtain operations with different logical and algebraic properties. This approach, based solely on the axioms which define the desired class of operations, explores the space of possible functions and often leads to discovery of new operations. However, the proposed system can also be used to generate aggregation operations that fit a collected data set. This application is very important as it provides a powerful new tool for modeling and processing empirical data.

1. Introduction

Aggregation operations combine several criteria in order to evaluate the overall support to various alternatives in decision-making problems [16]. They serve as connectives of facts and conditions entering such process [19]. In fuzzy set theory, aggregation operations are usually modeled as intersection, union, or their combination. The particular form and algebraic properties of these operations vary according to requirements for compensation among the criteria and other characteristics of given decision-making situation. There are close to one hundred different families of fuzzy aggregation operators used in various applications [2].

In this paper we propose to use genetic programming for generation of fuzzy aggregation operations in an automated manner. It is shown that the evolutionary process is able to reproduce some of the existing aggregation operations as well as to generate new operations with a broad range of algebraic and semantic properties. In addition, it is possible to constrain the process to evolve functions conforming to a set of empirical data.

This paper is organized in six sections. Section 2 provides an overview of fuzzy aggregation operations and identifies some of their important classes. The information contained in this section also serves as a resource for choosing various fitness functions driving the evolutionary process of genetic programming. The proposed genetic programming system is described in Section 3. In Section 4, results obtained using the proposed approach are presented. Properties of the resulting operations are analyzed and plausible interpretation of their semantics is proposed. This section also illustrates possible application of this approach to modeling of empirical data. Finally Section 5 draws main conclusions and presents plans for future work.

*Corresponding author. E-mail: musilek@ece.ualberta.ca.

2. Fuzzy aggregation operations

In fuzzy set theory, aggregation operations f_A are usually formalized using the following system of conditions:

Commutativity: $f_\mathsf{A}(x,y) = f_\mathsf{A}(y,x)$,
Associativity: $f_\mathsf{A}(x, f_\mathsf{A}(y,z)) = f_\mathsf{A}(f_\mathsf{A}(x,y),z)$,
Monotonicity: $(x \leqslant u) \wedge (y \leqslant v) \Rightarrow$
$f_\mathsf{A}(x,y) \leqslant f_\mathsf{A}(u,v)$,

and boundary conditions, either in the form

$$f_\mathsf{A}(1,x) = x \wedge f_\mathsf{A}(0,x) = 0, \tag{1}$$

for norms, or

$$f_\mathsf{A}(1,x) = 1 \wedge f_\mathsf{A}(0,x) = x \tag{2}$$

for co-norms. By selecting appropriate boundary conditions or relaxing some of the other conditions, a variety of alternative aggregation operations can be obtained.

2.1. *Classes of fuzzy aggregation operations*

2.1.1. *Triangular norms*

Boundary conditions Eqs (1) and (2) lead to two important classes of aggregation operations: *T-norms* (f_T) corresponding to conditions in (1), and *T-conorms* (f_S) specified by conditions in (2). The boundary conditions can be also described in terms of a *unit* element e, for which $f_\mathsf{A}(e,x) = x$; and an *absorbing* element g, for which $f_\mathsf{A}(g,x) = g$. The absorbing element, g, determines the level of compensation, γ, among the criteria entering the aggregation proces.

The absorbing element for T-norms is $g = 0$. This means that both conditions x, y are absolutely necessary: if one of the conditions is not satisfied, the corresponding alternative is completely rejected [9]. In other words, T-norms provide *no compensation*, $\gamma = 0$.

The absorbing element of T-conorms is $g = 1$. Therefore, if one condition is fully satisfied, the given alternative is accepted without regard to the second condition: T-conorms provide *full compensation*, $\gamma = 1$.

2.1.2. *Compensatory operations*

The two extremes described above are not appropriate in many practical situations where aggregation usually shows some degree of compensation. This fact was noticed two decades ago by Zimmerman and Zysno [19]. They proposed to use a combination of a T-norm and a T-conorm,

$$(1-\gamma) f_\mathsf{T}(x,y) + \gamma f_\mathsf{S}(x,y), \tag{3}$$

to achieve an arbitrary compensation, $\gamma \in [0,1]$. This could be interpreted as a generalization of the dual concept of aggregation as an exclusive ‘and/or’ operation to a more general ‘and-or’ operation. However, the resulting operations are non–associative.

2.1.3. *Nullnorms*

Calvo et al. have proposed a new class of binary aggregation operations called *nullnorms* [4]. Nullnorms (f_N) are continuous associative monotonic functions with boundary conditions relaxed to only

$$f_\mathsf{N}(1,1) = 1 \wedge f_\mathsf{N}(0,0) = 0, \tag{4}$$

with absorbing element $g \in [0,1]$. Indeed, these operations can provide an arbitrary degree of compensation. Unlike compensatory operations Eq. (3), nullnorms hold all the conditions defined for aggregation operations, above, including associativity. Goodman et al. [9] introduced *generalized nullnorms* that do not have to satisfy the condition of commutativity. The dismissal of commutativity causes asymmetry of aggregation that has a very interesting semantic interpretation. For example, when combining existing knowledge with new evidence, more emphasis could be put on either the old or the new knowledge. In other words, non-commutativity allows aggregated values to be assigned with *priority*.

2.1.4. *Non-associative operations*

Martinez et al. [13] recently proposed several non-associative aggregation operations in addition to the compensatory operations described in [19]. They proved that such operations are *semi-associative* in the sense that $x \leqslant y \leqslant z$ implies that

$$f_\mathsf{A}(x, f_\mathsf{A}(y,z)) \geqslant f_\mathsf{A}(y, f_\mathsf{A}(x,z)) \geqslant$$
$$f_\mathsf{A}(z, f_\mathsf{A}(x,y)).$$

More importantly, they showed that these operations are in accord with the so called “7 plus minus 2” law [14]. This rule states that a person can normally distinguish between no more than 7 ± 2 classes. The upper boundary of this rule, nine, corresponds to the fact that the maximal difference between two ternary expressions $f_\mathsf{A}(f_\mathsf{A}(x,y),z) - f_\mathsf{A}(x, f_\mathsf{A}(y,z))$ due to non-associativity is equal to $1/9$. This implies that the maximal level of *granularity* of a system based on such operations is 9.

2.2. Choice of an appropriate aggregation operation

Choice of an appropriate aggregation operator for a particular application is a difficult task with very little guidance provided by conventional approaches [2]. Choosing an operator on the basis of its theoretical properties is not feasible as these properties define a very large class of operators rather than a particular expression [1]. Zimmermann [18] offers several criteria for selecting aggregation operators. The criterion based on empirical fit is probably the most important, having a direct quantitative interpretation using a mean square error of approximation.

The problem of selecting an aggregation operator to fit empirical data has been addressed by several authors [8,11,19]. However, the proposed solutions are limited to fitting the parameters of aggregation operators without changing their structure. A number of similar but more general methods based on monotonic univariate splines and additive generators have been proposed by Beliakov and are summarized in [1].

Although used in fields such as computer vision [7, 10], and other problems within the area of fuzzy systems [6], evolutionary computing has not been applied to the problem of fitting fuzzy aggregation operators to data. In this paper, a new method for generating fuzzy aggregation operators is proposed. Based on genetic programming, this approach is not limited to parametric optimization of known families of operators but allows structural optimization of the solution to the data-fitting problem.

3. Genetic programming of fuzzy aggregation operations

3.1. Genetic programming

Genetic programming (GP) applies the genetic model of learning to the space of functions or programs [5]. In a GP system, there is a population of programs that describe candidate solutions to a problem at hand. This population of programs is iteratively transformed into successive new generations by applying principles and processes observed in evolution of natural systems and genetics.

GP starts with an initial *population* of randomly generated *individuals* composed of functions and terminals suitable for the problem at hand [12]. In the case of aggregation operators, functions may be arithmetic and logic operations, such as multiplication, addition, minimum, maximum, etc. The individuals that constitute the population are programs that, when executed, are the candidate solutions to the problem (e.g. potential fuzzy aggregation operators). Each program in the population is evaluated using a *fitness measure* that measures how well it performs in solving the problem. In many cases, fitness can be measured in terms of the error a program produces when run over a set of test cases. For fuzzy aggregation operators, the test cases can be derived from the axioms that must be satisfied by a desired family of operators. In the case of fitting an operator to empirical data, additional test cases may be derived from the data.

After the fitness of all individuals in the current population is evaluated, a new population is formed using the principles of *selection*, and the genetic operators, *crossover* and *mutation*. Due to the nonlinear nature of these processes and operators, a GP system is capable of exploring a large portion of possible solutions to a given problem in a function space. Because individuals are selected for genetic manipulation based on their fitness, the population is likely to improve its overall fitness over time, increasing the chance of finding a high-fitness individual that can be designated as the result of the GP process. The populations continue to reproduce and evolve until a stopping criteria is met (e.g. a perfect-fit individual is found, or a predefined number of generations has been reached). The overall scheme of the genetic programming system is illustrated in Algorithm 1. Details of the individual steps are described in the following subsections. More information on genetic programming can be found in [12, 17].

Algorithm 1 Genetic Programming System

```
Generate a population of computer programs                ▷ Initialization
Execute each program and evaluate how well                ▷ Fitness evaluation
        it solves the problem
Create a new population of computer programs              ▷ New generation
    1. select existing programs with high fitness         ▷ Selection
    2. Alter existing programs by random modifications    ▷ Mutation
    3. Combine existing programs by recombination         ▷ Crossover
Repeat Reproduction Until stopping criteria is satisfied  ▷ Iteration
Result is designated as the best-so-far solution          ▷ Solution
```

3.1.1. Initialization

As mentioned earlier, GP starts with an initial population of randomly generated individuals. The individ-

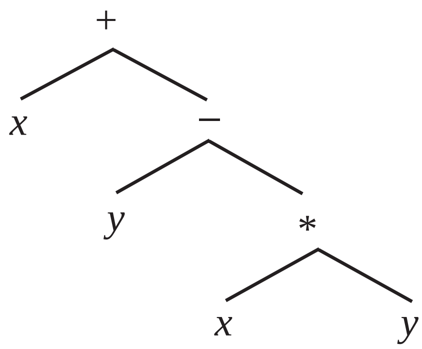


Fig. 1. Tree representation of algebraic sum $x + y - xy$.

Table 1
Function and terminal sets

Function Set	
$+, -, *, /$	basic arithmetic operations
m, M	minimum and maximum operations
e, r	power and root operations
Terminal Set	
$\mathbf{x}, \mathbf{y}$	vectors of discrete values $[0, 1]$
p	positive integer constant
$0, 1$	constants

uals in a GP system are computer programs that must be represented in a form suitable for their execution, evaluation, and for application of the genetic operators. Programs are usually expressed in GP as binary parse trees. For example, a simple function describing the T-conorm algebraic sum $x + y - xy$ can be represented as shown in Fig. 1.

The individuals (programs) in the population are composed of elements from the *function set* and the *terminal set*. The elements selected for the purpose of this study are listed in Table 1. These symbols are chosen because they reflect the typical set of operators and operands found in most fuzzy aggregation operators [16].

Although the individuals can be represented by binary parse trees, the trees are usually encoded into character arrays called *chromosmes*, in which each character represents one of the elements found in the function and terminal sets. Such a character encoding scheme is used for its simplicity and efficiency in processing and manipulating the individuals. The first entry in the array is always a function since it represents the parent node of the entire tree. Its corresponding leaf terminals or functions are located in the following two entries. In general, the corresponding two leafs of any function located at position k are located at positions $2k$ and $2k + 1$. As an example, the array representation of the parse tree from Fig. 1 is shown in Table 2.

3.2. Fitness

The fitness of each individual is evaluated using a fitness function. In the case of GP system described in this paper, this function encapsulates the axioms that classify an operator as a fuzzy aggregation operator and incorporates semantics of a particular family of such operators. Thus, the fitness function evaluates how strongly each operator conforms to the axioms and requirements for a given class of operators (e.g. triangular norms, nullnorms, etc.).

The test cases evaluate the individuals using vectors of N discrete values in the range $[0, 1]$, producing a set of resultant vectors, $\mathbf{R}_i$. Each resultant vector is compared to a vector $\mathbf{S}_i$ that contains the desired values derived from the axioms of a given family of fuzzy aggregation operators. For each pair of resultant and desired vectors representing a single condition, an average sum of the squared error can be determined using the following formula

$$E_i = \frac{\sum_{l=1}^{N} \left(\mathbf{R}_i(l) - \mathbf{S}_i(l)\right)^2}{N}, \tag{5}$$

where E_i is the error value for condition i, $\mathbf{R}_i$ is the vector resulting from running the i-th set of test cases, and $\mathbf{S}_i$ is the i-th vector of expected values. The conditions that can be used in the fitness function to represent specific properties desired from fuzzy aggregation operators are described in the following paragraphs.

3.2.1. Commutativity

Commutativity is tested by checking reciprocal boundary conditions. The following must hold true

$$f_{\mathsf{A}}(\mathbf{x}, \mathbf{y}) = f_{\mathsf{A}}(\mathbf{y}, \mathbf{x}).$$

To test for this property, two input vectors **x** and **y** are first submitted to the operator (in order) yielding vector $\mathbf{R}_c$. The two input vectors are then executed in the opposite order yielding the vector $\mathbf{S}_c$. The vector $\mathbf{R}_c$ is expected to hold exactly the same values as vector $\mathbf{S}_c$. Any deviations from this assumption are captured by error component E_c determined using Eq. (5).

3.2.2. Associativity

In order to be associative, an operator must satisfy the following equality

$$f_{\mathsf{A}}(f_{\mathsf{A}}(\mathbf{x}, \mathbf{y}), \mathbf{z}) = f_{\mathsf{A}}(\mathbf{x}, f_{\mathsf{A}}(\mathbf{y}, \mathbf{z})).$$

On the left hand of the equation, the operator is first applied to **x** and **y**, and secondly to this result and **z** yield-

Table 2
$x + y - xy$ in its character encoding

T-conorm $x + y - xy$															
Element	+	X	–			Y	*							X	Y
Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ing vector $\mathbf{R}_a$. On the right hand side of the equation, the operator is first applied to **y** and **z**, and secondly to this result and **x** yielding vector $\mathbf{S}_a$. Associativity requires that these two vectors are equal, $\mathbf{R}_a = \mathbf{S}_a$. Any departure from this requirement will result in a nonzero value of the error component E_a determined using Eq. (5).

3.2.3. Monotonicity

For an aggregation operation f_A to be monotonous, the following implication must hold

$$\mathbf{x} \leqslant \mathbf{y} \wedge \mathbf{y} \leqslant \mathbf{z} \Rightarrow f_\mathsf{A}(\mathbf{x}, \mathbf{y}) \leqslant f_\mathsf{A}(\mathbf{y}, \mathbf{z}).$$

Because of the logical condition 'less or equal than', this property is evaluated in a different way than Eq. (5). Three vectors are chosen that satisfy the condition, $x_i < y_i < z_i | \forall i \in [1, N]$. The operator is first applied to **x** and **y**, then to **y** and **z** yielding two resulting vectors $\mathbf{R}_m$ and $\mathbf{S}_m$, respectively. Unlike commutativity and associativity, it is not necessary for the two results to be equal. Instead, it is necessary that the values in $\mathbf{R}_m$ are less than or equal to the values of the corresponding elements in $\mathbf{S}_m$. Thus, in the error equation the square of the difference between the resultant and desired values is only considered when this condition is not satisfied

$$E_m = \frac{\sum_{l=1}^{N} \left(\mathbf{R}_m(l) - \mathbf{S}_m(l)\right)^2 |_{\mathbf{R}_m(l) > \mathbf{S}_m(l)}}{N} \qquad (6)$$

3.2.4. Boundary conditions

The boundary conditions have adifferent form depending on which class of aggregation operations is considered. For T-norms, the boundary conditions have the following form

$$\mathbf{x}\ f_\mathsf{T}\ \mathbf{1} = \mathbf{x}, \mathbf{1}\ f_\mathsf{T}\ \mathbf{x} = \mathbf{x}, \qquad (7)$$

$$\mathbf{x}\ f_\mathsf{T}\ \mathbf{0} = \mathbf{0}, \mathbf{0}\ f_\mathsf{T}\ \mathbf{x} = \mathbf{0}. \qquad (8)$$

The operator is applied to **x** and **1** (a unit vector) to give a resultant vector $\mathbf{R}_{bt1}$ for the condition Eq. (7). The expected vector, $\mathbf{S}_{bt1}$ is equal to **x**. To evaluate condition Eq. (8), the operator is applied to **x** and **0** (a null vector) to give $\mathbf{R}_{bt2}$. The expected result, $\mathbf{S}_{bt2}$, is equal to the null vector.

Similarly, the boundary conditions for T-conorms are

$$\mathbf{x} f_\mathsf{S} \mathbf{1} = \mathbf{1}, \mathbf{1}\ f_\mathsf{S}\ \mathbf{x} = \mathbf{1}, \qquad (9)$$

$$\mathbf{x} f_\mathsf{S} \mathbf{0} = \mathbf{x}, \mathbf{0}\ f_\mathsf{S}\ \mathbf{x} = \mathbf{x}. \qquad (10)$$

The operator is applied to **x** and **1** to obtain the resultant vector $\mathbf{R}_{bs1}$ for boundary condition Eq. (9). The expected vector, $\mathbf{S}_{bs1}$, is equal to the unit vector. The operator is then applied to **x** and **0** to give resultant vector, $\mathbf{R}_{bs2}$ for the second boundary condition Eq. (10). The expected result, $\mathbf{S}_{bs2}$, is equal to **x**.

Test cases for other families of aggregation operators can be derived in a similar way, based on the relaxed axioms given by Eq. (4).

3.2.5. Empirical data

To incorporate desired semantics of aggregation operators implied by empirical data, vectors **x**, **y**, and $\mathbf{S}_d$ must be obtained from the data set. The operator is then applied to vectors **x** and **y** yielding resultant vector $\mathbf{R}_d$. This vector is then compared to the vector of expected results $\mathbf{S}_d$ and the data error component, E_d is determined using Eq. (5).

3.2.6. Penalization

To improve convergence of the evolutionary process, the fitness function can be augmented to include more restrictions on the individuals generated. For example, some individuals may contain only one operand while (at least) two operands are required for any aggregation operator. To enforce this restriction, a penalty E_{p1} is applied if both x and y are not found in the individual's parse tree. Another potential problem stems from the simplicity of some existing operators, such as standard and algebraic triangular norms. Because the form of these operators is very simple, there is a significant chance that such operators will be generated as new individuals or obtained by genetic manipulation. This could lead to so called *genetic drift* – a change in frequencies of individuals in a population, resulting from chance rather than selection. Genetic drift reduces the genetic variability in the population, effectively preventing GP from discovering new programs (e.g. new fuzzy aggregation operators). To avoid this situation, individuals corresponding to known but undesirable aggregation operation are given a penalty E_{p2}.

3.2.7. *Calculation of fitness*

Error components calculated for each condition (associativity, commutativity, monotonicity, boundary conditions, etc.) are combined to obtain the overall fitness of given operator

$$F = \frac{\sum_i w_i E_i}{n}, i \in \{c, a, m, b, d, p\} \tag{11}$$

where E_i is the error value of component i, and n is the total number of conditions considered. The weighting coefficient, w_i, allows assignment of a different importance to each error component E_i which may be useful for certain classes of aggregation operators.

3.3. *Creating a new population*

The process of creating a new population of individuals (i.e. candidate fuzzy aggregation operators) is driven by selection and genetic manipulation of the individuals.

3.3.1. *Selection*

Selection takes place between two generations: some individuals from the current generation are selected and passed to the next. The criteria for selection is the fitness value of the individual. The genetic operators (crossover, mutation) are then applied to the individuals that have been selected.

The manner of selecting individuals to undergo genetic manipulation is a major factor in the convergence rate of a genetic program, and it is largely determined by *selective pressure* – the probability of the best individual being selected as compared to the average probability of selection of all individuals. Strong selective pressure can lead to premature convergence of the population towards globally optimal solutions (e.g. towards aggregation operators resembling the $\min$ or $\max$ operators). These solutions would then exploit themselves by replicating repeatedly and thus the population would inevitably exhibit genetic drift. On the other hand, weak selective pressures can make the genetic search ineffective by lack of convergence. Therefore, it is important to strike a balance between strong and weak selective pressures that works well for a particular problem [15]. In the case described in this study it is important to search a diverse space of solutions while still keeping an overall gradual convergence.

To implement fitness-based selection, fitness values first have to be normalized. According to the fitness function, Eq. (11), individuals with a lower numerical values of fitness are stronger. To simplify this inverse relationship, a probability is assigned to each individual based solely on its rank with respect to the rest of the population

$$P_s = q * (1 - q)^r, \tag{12}$$

where P_s is the probability that individual s is selected, r is the rank of the individual in the population, and q is a constant between 0 and 1 determining selective pressure. This selection method provides a greater degree of diversity since it is irrelevant how much difference occurs from consecutive individuals, ranked according to fitness. Figure 2 illustrates convergence of a typical GP run in terms of average and best fitnesses.

3.3.2. *Recombination and mutation*

Genetic recombination, also called crossover, is the process of creating a new individual from parts of its parents' representations. In this study single-point crossover is used in which a point in the chromosome (representing a program) of each of the selected parents is chosen randomly. The substrings of these two points are then exchanged to create two different individuals.

Mutation is a random change to an individual's representation and takes place at a mutation rate. Each individual is assigned a random number, `RAND`. If `RAND` is less than or equal to the mutation rate, P_m, mutation will take place. If an individual is chosen for mutation, a point in its parse tree is randomly chosen and changed along with the values included in its subtree.

Mutation and crossover rates also affect the convergence of the population. Similar to parent selection pressures, it is important to strike a balance in crossover and mutation rates. For example, high crossover rates can lead to genetic drift. When individuals in the population resemble each other, the crossover of the these individuals often produce very similar children. Mutation, however, increases the population diversity by providing an element of randomness. Diversity is particularly important when the population has already significantly converged as it provides the means to avoid local minima. The mutation rate can be varied to provide an optimal degree of variation suited for a given problem.

3.3.3. *Elitism*

Individuals selected for crossover and mutation are not themselves replicated in the following generation: only their children or mutants appear in the next population. Thus, parents with strong fitness values would not survive regardless of their high performance. To avoid this problem, elitism has been incorporated into

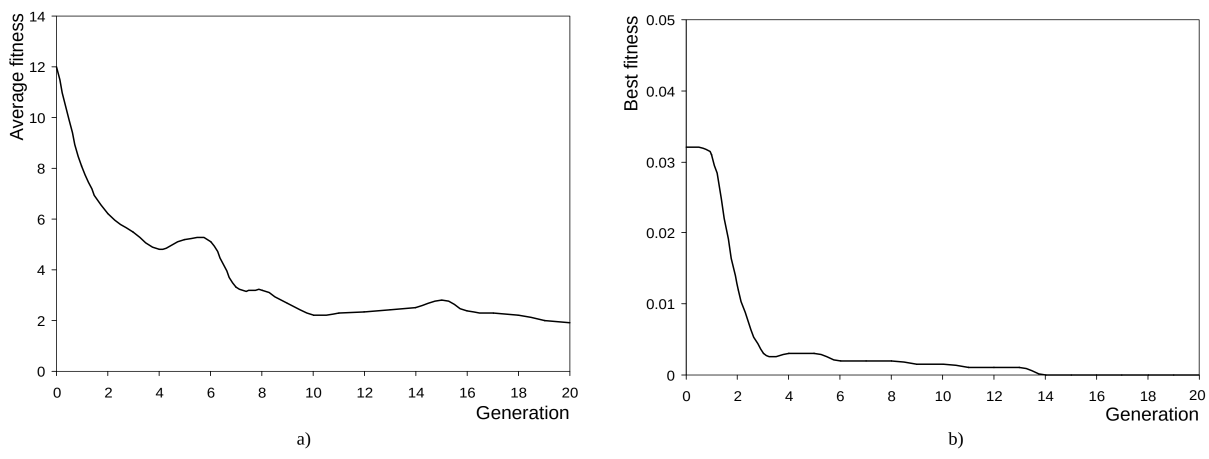


Fig. 2. a) Average fitness of individuals per generation, b) Fitness of the best individual in each generation.

the GP algorithm to copy the best individuals into the next generation. The size of the elite population is set small enough to provide enough vacancies in the population for new individuals yet large enough to provide a sufficient representation of the fittest individuals in the population.

Table 3
Generated aggregation operations

Operator	Fitness
$f^1_{\mathsf{A}} = \frac{x}{y^{x-2}}$	*F* = 0.000038
$f^2_{\mathsf{A}} = min(y, \sqrt{(x)})$	*F* = 0.008049
$f^3_{\mathsf{A}} = x^2 y$	*F* = 0.000473
$f^4_{\mathsf{A}} = x \min(x, y)$	*F* = 0.001157
$f^5_{\mathsf{A}} = xy^{(\frac{1}{1+y})}$	*F* = 0.001686
$f^6_{\mathsf{A}} = (y^{y^x})^{y^x}$	*F* = 0.005434
$f^7_{\mathsf{A}} = \max(y \max(x, y), x^2)$	*F* = 0.001141
$f^8_{\mathsf{A}} = y^{1-x}$	*F* = 0.039150
$f^9_{\mathsf{A}} = x + y^{\frac{1}{(1-x)\sqrt[x]{x}\sqrt[x]{x}}}$	*F* = 0.008026
$f^{10}_{\mathsf{A}} = \max(x, \sqrt{y})$	*F* = 0.009493

4. Experimental results

The GP system described in the previous section has been used to perform a number of experiments aimed at generation of fuzzy aggregation operators. The experiments can be divided into two groups: explorative generation of universal operators based solely on a set of formal conditions; and generation of operators particular to a set of empirical data. The parameters of the GP system were set as follows: population size $n = 10000$, function and terminal sets as in Table 1, selective pressure $q = 0.1$, number of test cases $N = 11$, probability of mutation $P_m = 0.1$, elitisms 10%, penalties $E_{p1} = 1$ and $E_{p2} = 100$, and stopping criteria $G_{\max} = 100$.

4.1. Explorative generation of fuzzy aggregation operations

Initial experiments were conducted with a fitness function based only on the formal axioms of associativity, commutativity, monotonicity, and boundary conditions. The system converged to operations resembling the simplest cases of intersection and union operations: $\min(x, y)$, xy, and $\max(x, y)$. These results were obtained either directly in their basic form or in a more complicated form that could be simplified to obtain a basic one. Examples of such results are

$$x \min(y, \min(y, 1)) = xy$$

$$\min(1, \min(y, \min(1, x))) = \min(x, y)$$

$$\max(x, \max(x, \max(y, y))) = \max(x, y).$$

Subsequently, penalization terms were incorporated into the fitness evaluation to increase the selective pressures toward non-standard aggregation operations. A variety of other aggregation operations were found that exhibit small nonzero error caused by minor violation of some traditional axioms. However, as discussed in Section 2.1, relaxation of certain axioms may lead to operations that are more realistic than fuzzy aggregation operations defined in the strict sense. Several examples of such operations are provided in Table 3.

Aggregation operations f^1_{A} through f^5_{A} exhibit intersection-like behavior while operations f^6_{A} through f^{10}_{A} can be regarded as models of a fuzzy union. The differ-

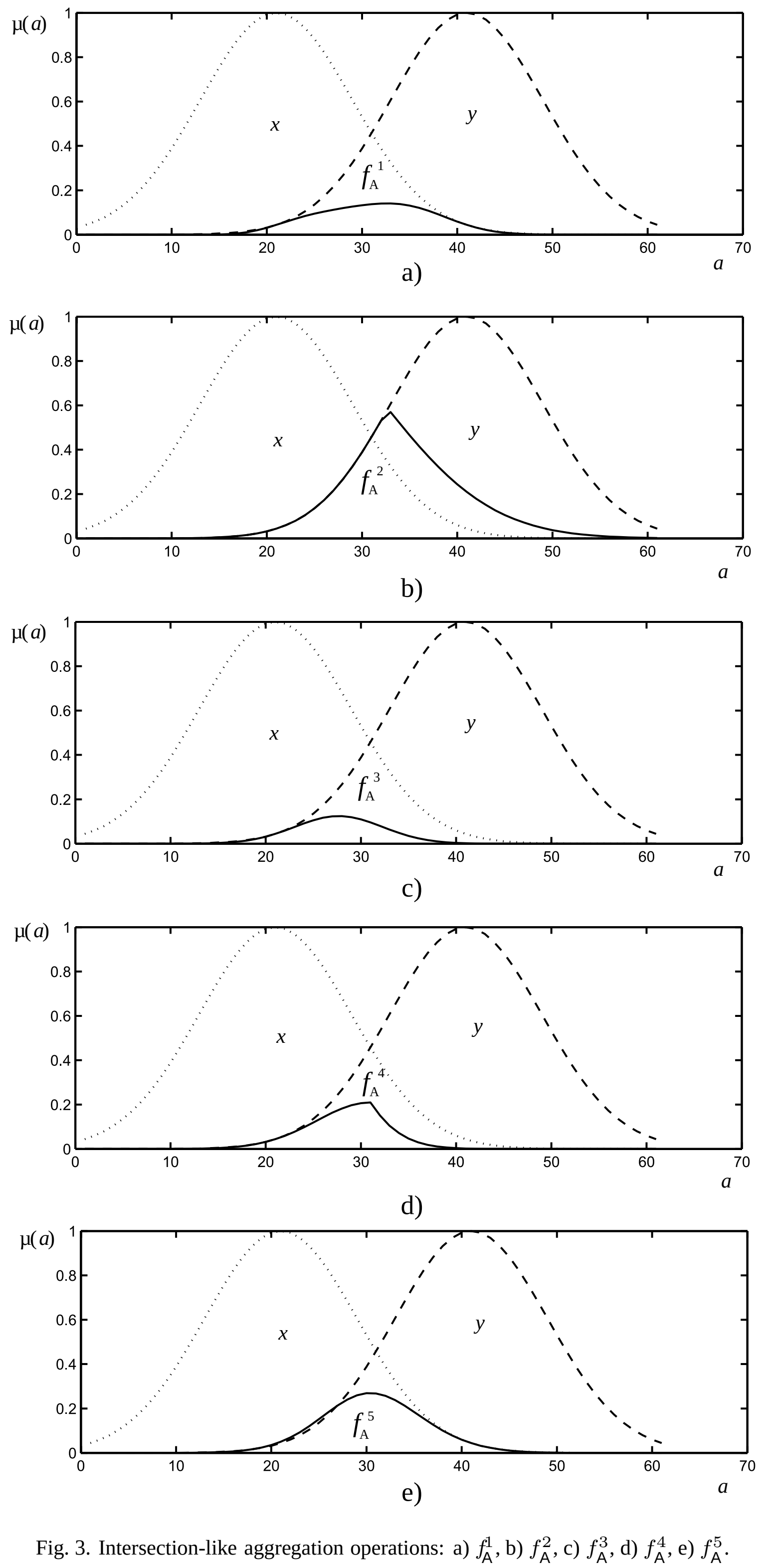
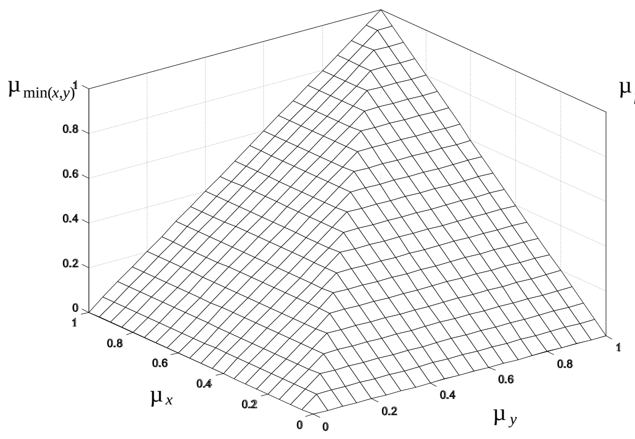
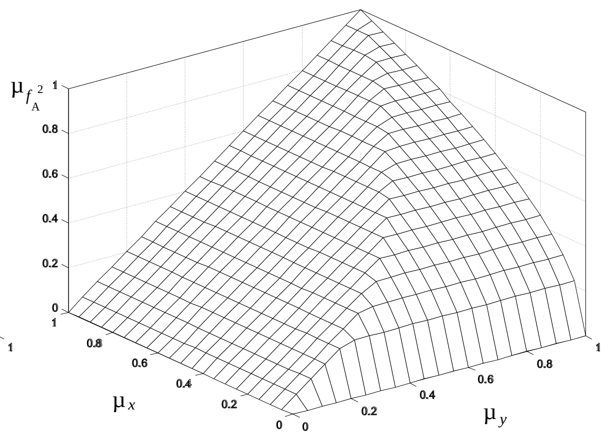


Fig. 3. Intersection-like aggregation operations: a) f_{A}^1, b) f_{A}^2, c) f_{A}^3, d) f_{A}^4, e) f_{A}^5.



a)



b)

Fig. 4. Intersection-like aggregation operations: a) 3d representation of $\min(x, y)$ b) 3d representation of f_A^2.

Table 4
Properties of obtained aggregation operations (Legend: Y satisfied, N not satisfied, $--$ strong preference of first operand, $-$ weak preference of first operand, 0 $-$ no preference, $+$ weak preference of second operand, $++$ strong preference of second operand)

Operation	Commutativity	Preference
f_A^1	N	$+$
f_A^2	N	$+$
f_A^3	N	$-$
f_A^4	N	$-$
f_A^5	N	$-$
f_A^6	N	$--$
f_A^7	Y	0
f_A^8	N	$++$
f_A^9	N	$+$
f_A^{10}	N	$-$

ent operations have algebraic properties (such as commutativity) that provide a different degree of preference to the first or the second operand. Properties of the aggregation operations are summarized in Table 4 and are illustrated in Figs 3 and 5. In addition, three-dimensional plots of selected functions are compared to standard T-norm and T-conorm in Figs 4 and 6, respectively. It is evident that the generated functions offer a wide variety of properties that can be used for various decision-making situations.

4.2. Generation of fuzzy aggregation operations to fit empirical data

This second type of experiment involved a GP system with a fitness function augmented to contain a description of empirical data. The data used in this study are based on a classical model problem provided by Zimmerman and Zysno in [19]. The problem is concerned with evaluation of the overall quality of dovetailing tiles based on separate assessment of two quality components. To combine fuzzy sets describing these two aspects, a suitable aggregation operation has to be chosen. The fit of a given operation can be checked against known (observed) values of the overall quality reported in [19]. The authors proposed a compensatory aggregation operator

$$f_C(x, y) = \min(x, y)^{(1-\gamma)} \max(x, y)^{\gamma}, \qquad (13)$$

and found the optimal value of the compensation factor, $\gamma = 0.562$. Such a compensatory operation is neither a strong T-norm nor T-conorm but a combination of these two extremes.

First, all operations described in the previous section (f_A^1 through f_A^{10}) were applied to the tile quality problem. The results were compared to the observed values to obtain a sum of squared error (SSE) for each operator. The best results were obtained with the aggregation operation $f_A^2 = \min(y, \sqrt{x})$ yielding an SSE of 0.3731 which is of the same order as the optimal compensatory operator (13) with an SSE of 0.1058. For comparison, $\max$ and $\min$ operators used alone yield an SSE of 2.5835 and 0.6442, respectively.

Table 5
Comparison of Sum of Squared Errors (SSE) for dovetailing tile quality problem [19]

Aggregation operation	SSE
$\min(x, y)$	0.6442
$\max(x, y)$	2.5835
$f_A^2 = \min(y, \sqrt{x})$	0.3731
$f_A^{11} = \sqrt{xy}$	0.1242
$\min(x, y)^{.468} \max(x, y)^{.562}$	0.1058

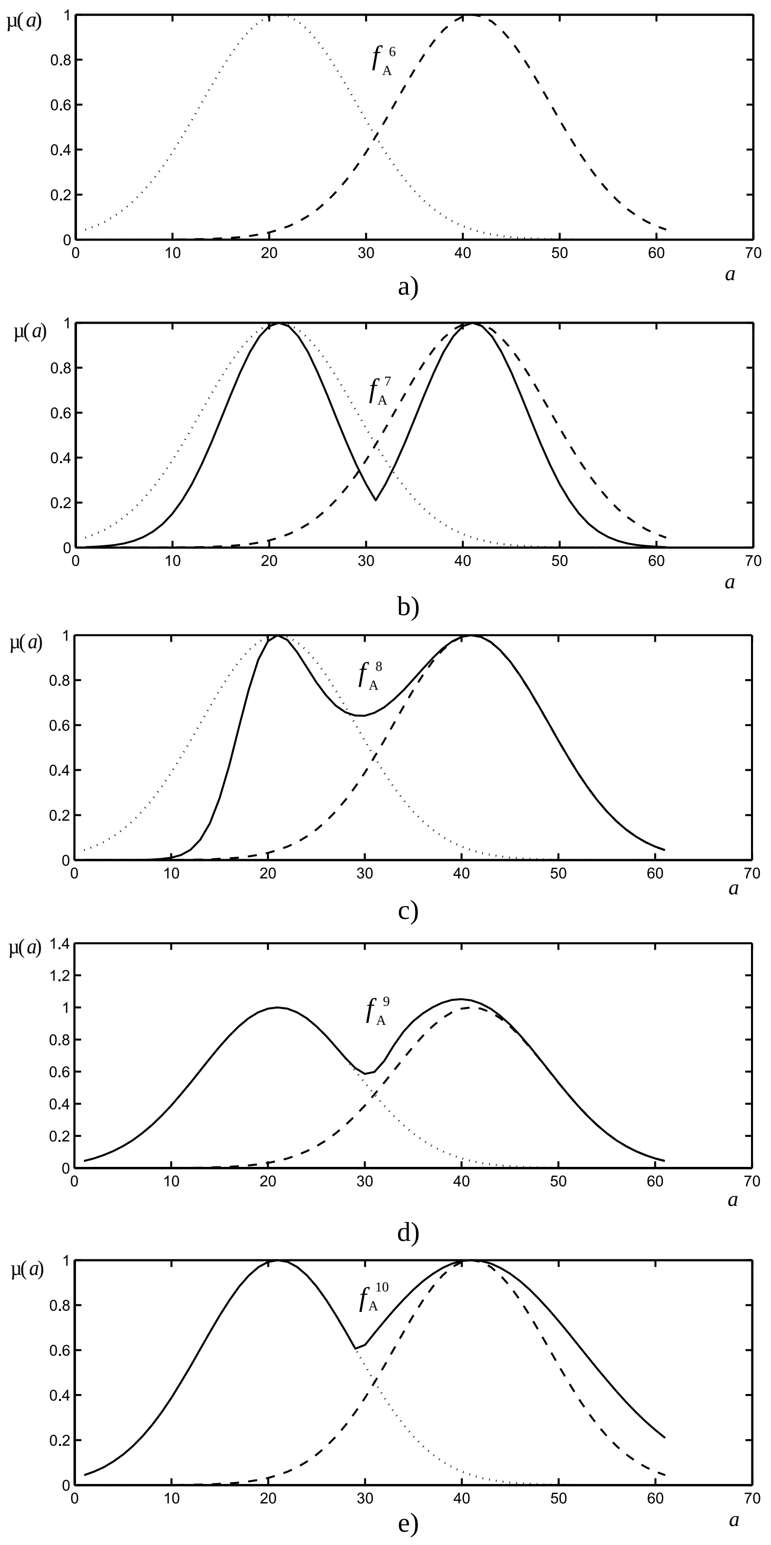


Fig. 5. Union-like aggregation operations: a) f_A^6, b) f_A^7, c) f_A^8, d) f_A^9, e) f_A^{10}.

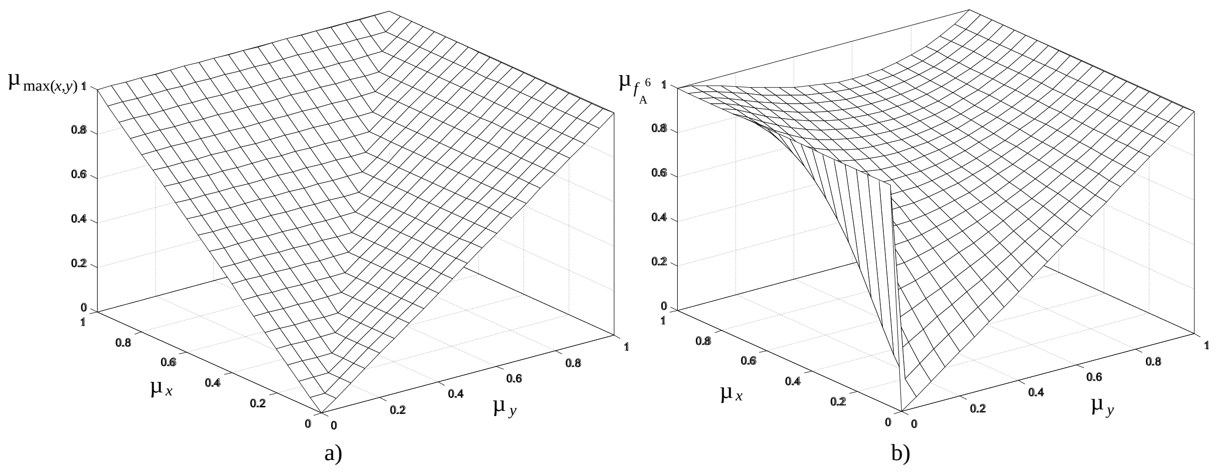


Fig. 6. Union-like aggregation operations: a) 3d representation of $\max(x, y)$ b) 3d representation of f_{A}^6.

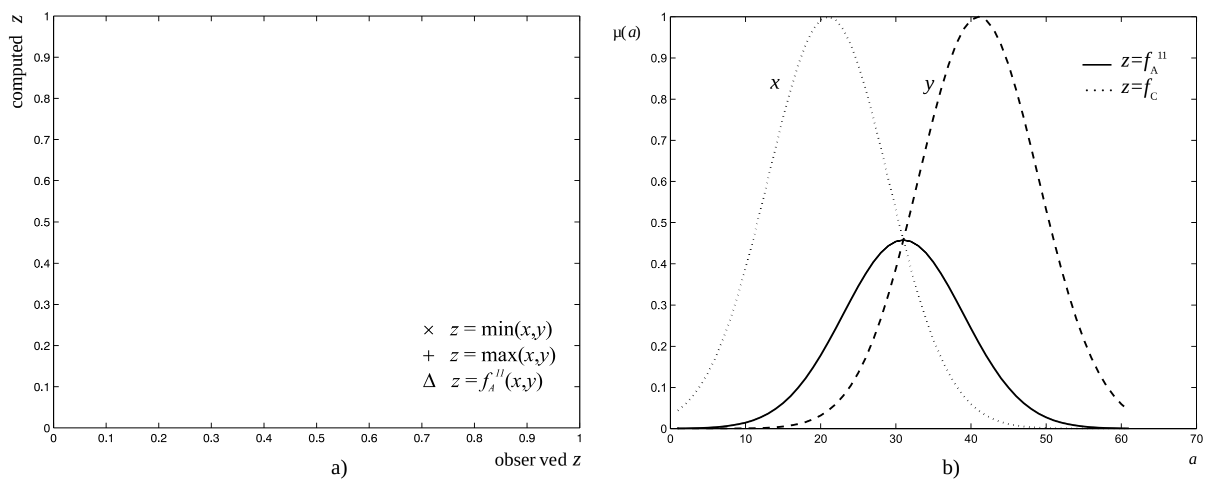


Fig. 7. a) Observed versus computed values for problem dovetailing tile quality problem for $\min(x, y)$, $\max(x, y)$ and f_{A}^{11}, b) f_{A}^{11} and the compensatory operator $\min(x, y)^{.468} + \max(x, y)^{.562}$.

Though surpassing the performance of standard operations, f_{A}^2 does not compare well to the optimal aggregation operator found by Zimmerman and Zysno. Since this optimal aggregation operation is a compensatory function, an aggregation operator discovered using a fitness function based on the union or intersection conditions (such as f_{A}^1 through f_{A}^{10}) may not yield extraordinary results when applied to this problem. For this reason, the fitness function was altered to better suit the problem. Instead of using the boundary conditions stated in Section 3.2.4, the model data provided by Zysno and Zimmerman in the tile quality problem was used as the inputs **x** and **y**. Applying the operator (individual in the GP system) to the two input vectors yielded the vector $\mathbf{R}_d$ while the known values of quality were used as the desired output vector, $\mathbf{S}_d$. By replacing the boundary conditions in the fitness function with the condition based on the empirical data, some new fuzzy aggregation operators were found. The operation $f_{\mathsf{A}}^{11} = \sqrt{(xy)}$, has the best fitness value and yields an SSE of 0.1242. As shown in Fig. 7(b) , operation f_{A}^{11} does not resemble a T-norm or T-conorm operation. Rather, it performs more closely to a compensatory function. Performance of various aggregation operations applied to the tile quality problem is summrized in Table 5.

Although the newly discovered operation f_{A}^{11} does not outperform the compensatory operation from [19], it clearly surpasses the two standard aggregation operations. To illustrate performance of this operation with respect to the standard fuzzy intersection ($\min$) and union ($\max$), Fig. 7(a) plots the observed vs. computed values of the overall quality for these three cases. The diagonal line indicates the optimal relation (equality) of these pairs of values. In addition, the form of f_{A}^{11} is much simpler and computationally less expensive than that of the compensatory operator f_{C}.

As seen from this example, the discovery of fuzzy aggregation operators is not solely applicable to discovering triangular norms and other general aggregation operations. Rather, model data or other specific conditions can easily be incorporated into the fitness function yielding a fuzzy aggregation operator that conforms to the conditions of a given problem.

5. Conclusions and future work

Selection of an appropriate aggregation operation for a given set of empirical data is a challenging problem. Its solution has led to relaxation of some strong algebraic conditions traditionally considered as important properties of such operations. This fact opens wide the possibility of applying semi-heuristic methods, such as genetic programming, to automatically generate aggregation operations with custom algebraic and semantic properties. The major application of this methodology is in the area of decision support systems when little is known about how some criteria and conditions should be formally connected and, at the same time, empirical data are available.

In this paper, a genetic programming system capable of discovering general classes of fuzzy aggregation operations has been proposed. Its practical application in aggregating experimental data has been demonstrated. The evolutionary process can be adapted towards operations with different properties by simply modifying the fitness function. The fitness functions proposed here need to be further refined to make the process more efficient. In addition, appropriate ways of incorporating domain knowledge into fitness functions could be devised to allow more direct search for aggregation operations suitable for particular experimental data or specific decision situations.

Acknowledgements

Support provided by the Canada Foundation for Innovation and the Natural Sciences and Engineering Research Council of Canada is greatly appreciated.

References

[1] G. Beliakov, How to build aggregation operators from data, *Int. J. Intelligent Systems* **18** (2003), 903–923.

[2] G. Beliakov and J. Warren, Appropriate choice of aggregation operators in fuzzy decision support systems, *IEEE Tran. Fuzzy Systems* **9**(6), 773–784.

[3] R.E. Bellman and L.A. Zadeh, Decision-making in a fuzzy environment, *Management Science* (17) (1970), 141–164.

[4] T. Calvo, B. de Baets and J. Fodor, The functional equations of Frank and Alsina for uninorms and nullnorms, *Fuzzy Sets and Systems* **120**(3) (2001), 385–394.

[5] D.A. Coley, *An introduction to genetic algorithms for scientists and engineers*, World Scientific, 1999.

[6] O. Cordón, F.A.C. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, *Fuzzy Sets and Systems* **141**(1) (2004), 5–31.

[7] M. Ebner, On the evolution of interest operators using genetic programming, in: *Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic Programming*, R. Poli, ed., Paris, 14–15 April, 1998, pp. 6–10.

[8] D. Filev and R. Yager, On the issue of obtaining OWA operator weights, *Fuzzy Set Syst* **94**(4) 157–169.

[9] I.R. Goodman, R.A. Trejo, V. Kreinovich, J. Martinez and R. Gonzalez, *An even more realistic* (*non-associative*) *interval logic and its relation to psychology of human reasoning*, In Proceedings of IFSA/NAFIPS 2001, Vancouver, Canada, 25–28 July, 2001, pp. 1586–1591.

[10] C. Harris and B. Buxton, *Evolving edge detectors with genetic programming*, In Proc. of Genetic Programming 1996, July 28–31, Stanford, CA, 1996, pp. 309–315.

[11] K. Hirota and W. Pedrycz, OR/AND neuron in modeling fuzzy set connectives, *IEEE Trans Fuzzy Syst* **2** (1994), 151–161.

[12] J. Koza, *Genetic programming: On the programming of computers by means of natural selection*, MIT Press, 1992.

[13] J. Martinez, L. Macias, A. Esper, J. Chaparro, V. Alvarado, S.A. Starks and V. Kreinovich, Towards more realistic (e.g., non-associative) AND- and OR-operations in fuzzy logic, In Proc. of the 2001 IEEE SMC Conf., *IEEE Press* **5** (2001), 2187–2192.

[14] G.A. Millner, The magical number seven plus or minus two: some limits on our capacity to process information, *Psychological Review* **63** (1956), 81–97.

[15] Z. Michalewicz, *Genetic Alorithms $+$ Data Structures $=$ Evolution Programs*, 3rd Revised and Extended Edition, Springer-Verlag, 1999.

[16] W. Pedrycz and F. Gomide, *An Introduction to fuzzy sets: Analysis and design*, MIT Press, 1998.

[17] M. Tomassini, Parallel and distributed evolutionary algorithms: A review, in: *Evolutionary Algorithms in Engineering and Computer Science*, K. Miettinen, P. Neittaanmaki, M.M. Mäkelä and J. Périaux, eds, Wiley, 1999, pp. 113–133.

[18] H. Zimmermann, *Fuzzy set theory and its applications*, Kluwer, Boston, 1996.

[19] H. Zimmerman and P. Zysno, Latent connectives in human decision making, *Fuzzy Sets and Systems* **4**(1) (1980), 37–51.