The Bees Algorithm to Extract Fuzzy Measures for Sample Data

Xiaojing Wang
Computer Science Department
The University of Texas at El Paso
El Paso, Texas 79968-0518
Email: xwang@utep.edu

Jeremy Cummins
Department of Computer Science and Information Systems
Youngstown State University
Youngstown, Ohio 44555
Email: jjcummins@gmail.com

Martine Ceberio
Computer Science Department
The University of Texas at El Paso
El Paso, Texas 79968-0518
Email: mceberio@utep.edu

***Abstract*—In Multi-Criteria Decision Making (MCDM), decisions are based on several criteria that are usually conflicting and non-homogenously satisfied. Non-additive (fuzzy) measures along with the Choquet integral can model and aggregate the levels of satisfaction of these criteria by considering their relationships. However, in practice, it is difficult to identify such fuzzy measures. An automated process is necessary and can be done when sample data is available. In this article, we propose to use an adapted Bees algorithm to identify fuzzy measures from sample data. Our experimental results show that our Bees algorithm is faster and provides at least similar accuracy as or better than existing algorithms.**

I. Introduction

Very often, decisions are based on several conflicting criteria; e.g., which car to buy that is cheap and energy efficient. This kind of complex decision process is called Multi-Criteria Decision Making (MCDM). In general, in order to reach a decision, we mentally "average / sort" these criteria and their satisfaction levels.

In computer science, averaging corresponds to aggregating values of satisfaction with some weight on each criterion (a.k.a. additive aggregation). Additive aggregation, however, assumes that criteria are independent, which is seldom the case [2].

Non-additive (fuzzy) measures [7] can be used to represent relationships between criteria. Along with the Choquet integral that combines them, such fuzzy measures address the problem of taking dependencies into account. However, to make this happen, fuzzy measures need to be determined: they can either be identified by decision maker/expert or by extracting them from sample data.

Since human expertise might not always be available, we focus here on extracting fuzzy measures from sample data. The sample data that we use is a set of overall (i.e., preference that would otherwise be obtained after being combined through Choquet integral) preference values associated with given inputs (i.e., items that we need to decide on, such as cars). Fuzzy measure extraction seeks to determine the fuzzy measure that, when combined in a Choquet integral, returns the value that is the closest to the corresponding sample data. This problem is therefore an optimization problem.

In this article, we propose to use the Bees algorithm to identify fuzzy measures from sample data. The Bees algorithm has been successfully used in many optimization problems but requires a significant amount of tuning to attain reasonable performance. We adjusted the Bees algorithm to the needs of fuzzy measure extraction problems: in this article, we show that using the Bees algorithm to extract fuzzy measures from sample data provides better performance with results of similar or better accuracy than existing approaches.

The article is organized as follows: Section 2 provides background and recalls necessary definitions on MCDM, fuzzy measures, fuzzy integrals, and fuzzy measure extractions. Section 3 introduces the optimization problems and existing approaches to solving the specific problem of fuzzy measure extraction. We present our approach in Section 4, describe our experimental strategy, report, and analyze the results in Section 5. Finally, we draw conclusions and propose directions for future work in Section 6.

II. Background

A. Multicriteria Decision Making

Multicriteria decision making (MCDM) is the making of decisions based on multiple attributes (or criteria). It is a 3-tuple problem $(X, A, \succeq)$, where:

- X is the set of consequences;
- $A = \{1, \cdots, n\}$ is the (finite) set of n criteria (or attributes);
- $\succeq$ is a preference relation on the set of consequences.

The set of consequences X is a multidimensional space, where $X \subseteq X_1 \times \cdots \times X_n$, and each $X_i \subseteq X$ represents a set of values of attribute i, where $i \in A$. For each $i \in A$, there is a preference relation $\succeq_i$ on each space X_i, such that for $x_i, y_i \in X_i$, $x_i \succeq_i y_i$ means that x_i is preferred to y_i. And there is a global preference relation $\succeq$ on X.

Note: The reason why X can be a proper subset of $X_1 \times \cdots \times X_n$ because all combinations of all criteria values do not necessarily exist: each n-tuple of $X_1 \times \cdots \times X_n$ represents a possible instance / an alternative to pick from, all of which are not necessarily possible. For instance, consider the case of cars: one criterion being the price, another being the year of make. It is unlikely that the lowest value of the price criterion can match any high value of the high criterion; *i.e., there is recent car that is very cheap*.

For example, consider the MCDM case of buying a car. The decision is based on the buyer's preference on several criteria, such as, manufacturer, model, power, cost, mileage per gallon. As a result:

$$A = \{\text{manufacturer, model, power, cost, mileage per gallon}\}$$

Assume the buyer wants to choose one car from four alternatives, then X={car1, car2, car3, car4}. For each attribute $i \in A$, the buyer may order those four alternative cars based on his/her preference, e.g., for attribute A_i, car1 $\succeq$ car3 $\succeq$ car4 $\succeq$ car2; for attribute A_j, car2 $\succeq$ car4 $\succeq$ car1 $\succeq$ car3. Now, the buyer has to combine his/her preferences with respect to all attributes for each alternative (each of the four cars) to obtain a global preference ranking such that the final order of the alternatives is in agreement with the buyer's partial preferences.

We assume that for each attribute $i \in A$, there exists a real valued function $u_i : X_i \to R$ such that for all x_i, $y_i \in X_i$:

$$x_i \succeq_i y_i \Longleftrightarrow u_i(x_i) \geq u_i(y_i)$$

Function u_i is called the i-th monodimensional utility function [8] and scales the values of all attributes onto a common (real) scale. Then an aggregation operator that "combines" the monodimensional utility functions needs to be used to represent the global preference, i.e., a preference over the set of consequences X: $\forall x, y \in X$, $x \succeq y$ or $y \succeq x$.

The common aggregation operator being used is a weighted sum; i.e.,

$$u(x) = \sum_{i=1}^{n} w_i u_i(x_i),$$

where w_i is the weight of each attribute, representing the importance of each attribute, and $\sum_{i=1}^{n} w_i = 1$. The best alternative is the one with the highest value of u. This approach is simple and easy to use with low complexity. However, using an additive aggregation operator assumes that all the attributes are independent. In practice, it is only seldom the case that criteria are indeed independent: often, decisions are based on several conflicting criteria. Therefore, using additive approach is often not good: we have to use non-additive approaches, i.e., fuzzy measures and integrals [2].

B. Fuzzy measures and integrals

Fuzzy measures are non-additive measures. They can be used to represent the degree of interaction of each subset of criteria [3]. In what follows, we consider a finite set of criteria $A = \{1, \cdots, n\}$.

Definition Let A be a finite set and $\mathcal{P}(A)$ the power set of A. A fuzzy measure (or a non-additive measure) defined on A is a set function $\mu : \mathcal{P} \to [0, 1]$ satisfying the following axioms:

(1) $\mu(\emptyset) = 0$
(2) $\mu(A) = 1$
(3) if X, $Y \subseteq A$ and $X \subseteq Y$, then $\mu(X) \leq \mu(Y)$

Note: Fuzzy measures provide a weaker property, called monotonicity, than normal probability measures. The fuzzy measures are used to show the importance of each subset and how each subset of attributes interacts with others. Fuzzy measures are expensive to determine: for a set defined over n attributes, 2^n values of a fuzzy measure are needed because there are 2^n subsets of A. In reality, only $2^n - 2$ coefficients are needed since the values for the empty set and the full set are known (properties (1) and (2) from the above definition).

Two main integrals can be used to combine fuzzy measures.

Definition Let μ be a fuzzy measure on A. The Sugeno integral of a function $f : A \to R$ with respect to μ is defined by:

$$(S) \int f \circ \mu = \bigvee_{i=1}^{n} (f(x_{(i)}) \wedge \mu(A_{(i)}))$$

where $\vee$ is the supremum and $\wedge$ is the infimum. $\cdot_{(i)}$ indicates that the indices have been permuted so that $0 \leq f(x_{(1)}) \leq \cdots \leq f(x_{(n)}) \leq 1$, and $A_{(i)} = \{x_{(i)}, \ldots, x_{(n)}\}$.

Definition Let μ be a fuzzy measure on A. The Choquet integral of a function $f : A \to R$ with respect to μ is defined by:

$$(C) \int_A f d\mu = \sum_{i=1}^{n} (f(\sigma(i)) - f(\sigma(i-1)))\mu(A_{(i)})$$

where σ is a permutation of the indices in order to have $f(\sigma(1)) \leq \cdots \leq f(\sigma(n))$, $A_{(i)} = \{\sigma(i), \ldots, \sigma(n)\}$ and $f(\sigma(0)) = 0$, by convention.

The Sugeno and Choquet integrals are structurally similar, but essentially different in nature [6]: the Sugeno integral is based on non-linear operators and the Choquet integral is usually based on linear operators. The applications of Sugeno and Choquet integrals are also very different [8]: the Choquet integral is generally used in quantitative measurements, and a MCDM problem usually uses a Choquet integral as a representation function.

In this article, we focus on the Choquet integral.

C. Determining Fuzzy Measures

In MCDM, we would expect the decision maker to be more than likely to give the values of the fuzzy measure, but in most circumstances this is not the case. Attempts of making fuzzy measure identification easier for the decision makers have been made in [2], [10], [11].

- In [2], the authors attempt to make this task easier by only requiring the decision maker to give an interval of importance for each interaction.
- In [11], the author suggests a diamond pair-wise comparison, where the decision maker only must identify the interaction of 2 criteria using a labeled diamond. From there, the algorithm evaluates the values of the numeric weights.
- In [10], the author discusses user specified weights mixed with an interaction index denoted λ or ξ. This algorithm is applied using an online aggregation application [15].

However, in most cases, the decision maker still does not understand the interactions well enough to be able to give a good value of each fuzzy measure. This is where expert identification or fuzzy measure extraction comes into play.

In expert identification, an expert is used to give all values of the fuzzy measures. Expert identification in most circumstances is unfeasible since in many cases, the decision maker does not have constant access to an expert. In addition, since there are $2^n - 2$ values of a fuzzy measure for a problem with n criteria expert identification: it would be too time consuming anyway to be practical [5].

As a result, instead of using an expert to provide us with the values of the fuzzy measure, we choose to extract the fuzzy measure.

III. Fuzzy Measure Extraction and Optimization

A. Relation Between our Problem and Optimization

For lack of an expert to provide all values of the fuzzy measure, we need seed data to give us an idea of the preferences: we use sample data. Extracting fuzzy measure is performed starting from such seed data. Fuzzy measure extraction seeks to find the fuzzy measure that returns the closest value to the expert's coalition value: one of the preference values that constitute our seed data.

Let's take a look at the following situation: Our MCDM problem involves n attributes, and we have m sample data. It means that we have access to the following: m preference values $\tilde{y}_j$, $j \in \{1, \cdots, m\}$, corresponding to m alternative items X_j. It means that if we knew the corresponding exact fuzzy measure μ, let's denote it by $\tilde{\mu}$, we would be able to compute $\tilde{y}_j$ as $(C)\int_A f d\tilde{\mu} = \sum_{i=1}^{n}(f(\sigma(i)) - f(\sigma(i-1)))\tilde{\mu}(A_{(i)})$, where f is a utility function defined on X.

Now, with our sample data, we only have access to the preference values of a subset of X. In order to have access to preference values of other alternatives in X, we need to determine μ, which is, the $2^n - 2$ values of the fuzzy measure. We are going to determine all values of μ such that the corresponding computed Choquet integral ideally equals the preference values of the sample data. In practice however, for lack of equaling the sample data preferences, we aim at getting as close to them as possible. As a result, we aim at minimizing the following sum (and getting as close to 0 as possible) [4]:

$$e = \sum_{j=0}^{m} \left(\tilde{y}_j - \sum_{i=1}^{n} (f(\sigma(i)) - f(\sigma(i-1)))\mu(A_{(i)}) \right)^2 \quad (1)$$

When $e = 0$, the identified fuzzy measure μ is the exact solution of the problem: this is the ideal case. In most cases, we put up with "only" reaching an approximate optimal solution, that is, with $e \neq 0$ but close to 0. The reason for such a weaker expectation is that the sample data might not be fully consistent with one fuzzy measure.

In any case, extracting a fuzzy measure is cast down to solving an optimization problem. This optimization problem is actually a constrained optimization problem since the values of μ that the minimization process seeks must satisfy the monotonicity properties that characterize a fuzzy measure (as seen in Section 2).

B. Optimization Techniques used for Fuzzy Measure Extraction

As just pointed out, the problem we address can be solved using an optimization algorithm. Several optimization approaches have been proposed to extract fuzzy measures. In what follows, we review the most important of these techniques.

1) Gradient Descent Approach: In [5], Grabisch proposed a gradient-descent algorithm with constraints for identifying fuzzy measures that takes advantage of the "lattice structure of the coefficients". Using this lattice structure, it is easier to find the path to be used to calculate the Choquet integral and the path used depends on the input values.

This algorithm consists of two steps. The first step modifies only the coefficients on the path in order to decrease the error. The second step modifies unmodified nodes left in the first step if there exist any.

This algorithm can reach a local optimum quickly and accurately if the initial values are properly selected. But, the monotonicity constraints need to be checked at every iteration in both steps. This algorithm was improved on in [1]. According to the experiments reported in [1], the modified gradient-descent algorithm is 385 times faster than a Genetic Algorithm with similar or better accuracy.

However, there is no evidence that a global optimum can ever be reached.

2) Genetic Algorithms: Genetic algorithms attempt to model natural evolution. They have been used successfully to solve a number of optimization problems, including fuzzy measure extraction in [3], [13], and [14].

The genetic algorithm starts with a seed population, and each individual in the population is evaluated by a fitness function which measures how well the individual performs to the problem. The fittest individuals are selected for mating to generate a new population through a three-step process: selection, crossover, and mutation. This is done to make sure that the new generation not only inherits the parents' properties but also has some genetic diversities. The algorithm stops until all individuals are identical or a stopping criteria is met.

The genetic algorithm shows promise for extraction of fuzzy measures, especially since it has been used successfully in the past. The problem with genetic algorithms is that it is possible to fall into a local optimum. While mutation is included in the algorithm to try to avoid it, it does not totally prevent it (especially if there are local optimums that are in distant locations but have values close to the global optimum).

3) Neural Networks: An algorithm using a neural network for fuzzy measure extraction is proposed in [12]. The calculation of the Choquet integral is described by a neural network. The goal of this algorithm is to find a global optimum on the search space. However, the authors realized that using neural network algorithm, the search easily fall in a local minimum. They proposed to use multi-initialized parallel search (MIPS), that is, spread the initial values to the corners, the edges, and the center of the search space in order to increase the chance to find the global optimum. They also proposed to alternatively use genetic algorithm with the neural network algorithm to jump out from a local optimum on the search space.

Because falling into a local optimum is the main drawback of existing algorithms, we explore the use of a global optimization algorithm to solve our fuzzy measure extraction problem.

IV. Identifying Fuzzy Measures Using the Bees Algorithm

The Bees optimization Algorithm, proposed in [9], uses bees' natural food foraging habits as a model for the exploration of the search space. The Bees Algorithm combines a local and "global" search that are both based on bees natural foraging habits.

The Bees Algorithm roughly unwinds as follows:

- First a number of "scout bees" are randomly sent out.
- The patches of "nectar" (elements of the search space / candidate values for the fuzzy measure) are then ranked according to evaluated fitness. More bees are dispatched to look in neighboring areas of good patches of "nectar".
- At each iteration, a number of "scout bees" are kept to explore other areas in hope of better patches of "nectar": this keeps the algorithm searching "globally".
- When a new patch is found, its fitness is evaluated and compared against previously explored patches and a proportional number of "bees" is sent to it. The dispatched "bees" perform a local search by moving in a random direction from the patch of "nectar".
- If a local search "bee" finds a better patch of "nectar", the location from where it was dispatched is moved to the new location [9]. The Bees Algorithm performs local search by sending an amount of "bees" that is proportional to the patch's fitness.

It is believed that the best ranked "patch" when a stopping criterion is met, is the optimal solution. By the work the authors of [9] did applying the algorithm, in most cases, is faster than a number of other algorithms, and returned a solution with in .1% of the perfect solution every time run. They believe that the new algorithm proposed is up to 207 times faster than the Genetic Algorithm. The algorithm also presents an improvement over the simplex method and the stochastic simulated annealing optimization procedure. The functionality of using real life modeled methods is also shown as the Genetic Algorithm, and the Bees Algorithm, were the only algorithm tested to achieve success 100% of all runs. They also state that the Bees Algorithm always finds the global optimum and ignores local optimum [9]. We believe that the Bees Algorithm shows the most promise for use in creating a new algorithm for fuzzy measure extraction of the methods

```
1. Randomly initialize population
2. Evaluate fitness of each population
3. While stopping criteria not met
4. Randomly pick all patches
5. Search around elite & best patches
6. Select the fittest bee from each patch
7. Replace scout patches with random locations
   and evaluate their fitness
8. If met shrink criteria
   a. Shrink patch size used for local search
9. End while
```

Fig. 1. Pseudocode of the Bees algorithm

discussed. Not enough evidence is provided to prove that the algorithm does not fall into a local optimum. In addition, this algorithm's global search method is based totally on random chance. While this seems to work, it could be possible for a more optimal solution to be totally missed. The algorithm also requires a fairly large amount of experimenting as the algorithm has a relatively large number of parameters that must be set.

As described above, the bees algorithm shows its better performance. We proposed to use the bees algorithm to extract fuzzy measures from sample data. The pseudo code of the algorithm is illustrated in Figure 1. To save the running time, we add penalty to patches when monotonicity is violated, and the penalty is big enough to make sure those penalized patches do not have chance to be selected.

V. Experiments and Results

A. Testing Methodology

The goal of our experiments is to extract the fuzzy measure μ that is presented in Table 1. To test the performance of the Bees algorithm, we follow the same procedure as in [5] and [3], as described hereafter. The input-output system contains 81 sample data with $Y = f_c(X) + n$, where Y is the vector of the system outputs, X is a 4-tuple input (x_1, x_2, x_3, x_4) with $x_i \in \{0, 1\}$, and n is a centered gaussian noise. In the same manner as [5] and [3] did, we also check 5 different variances σ^2 = 0.0, 0.00096, 0.00125, 0.00625, and 0.0125, respectively. Our algorithm is executed on an Intel Xeon e5540 @2.53GHz machine. The parameters used for the algorithm are presented in Table 2. In order to find the optimal solution (most fitting fuzzy measure), we are interested in the minimum mean square error E, where $E = e^2 + p = \frac{1}{81}\sum_{i=1}^{81}(y_i - \hat{y_i})^2 + p$, where p is a penalty factor adding to the penalized patch.

B. Quality of Solutions

Table 3 shows our experimental results. We compare them with the results of other algorithms in the same table. The results of other algorithms are taken from [5] and [3], respectively. The final obtained fuzzy measures are in Table 4.

TABLE I
FUZZY MEASURE TO BE IDENTIFIED

A	$\mu(A)$	**A**	$\mu(A)$	**A**	$\mu(A)$
$\{1\}$	0.1	$\{1,2\}$	0.3	$\{1,2,3\}$	0.5
$\{2\}$	0.2105	$\{1,3\}$	0.3235	$\{1,2,4\}$	0.8667
$\{3\}$	0.2353	$\{1,4\}$	0.7333	$\{1,3,4\}$	0.8824
$\{4\}$	0.6667	$\{2,3\}$	0.4211	$\{2,3,4\}$	0.9474
		$\{2,4\}$	0.8070		
		$\{3,4\}$	0.8235		

TABLE II
PARAMETERS

Parameter	**Value**
Number of total iterations	100
Number of scout bees	50
Number of best patches	30
Number of elite patches	10
Number of best bees	50
Number of elite bees	100
Initial patch size	0.001
Number of iterations before patch shrink	10
Monotonicity enforced by penalty/fix	Penalty
Local/Global patch age	Global
Shrink on no change/improvement	No change

We observe that when the variance of the gaussian noise is increased, the mean square error is increased accordingly. However, the mean square error E is always less than the variance, which means our algorithm does not generate negative influence to the data.

Comparing to the gradient descent algorithm in [5], our algorithm performs much better when the variance is greater than zero. Comparing to the genetic algorithm in [3], our algorithm has better results when variance is small, but when the variance keeps increasing (≥ 0.00625), the results of our algorithm becomes worse.

C. Time to Solutions

To get the results in table 3, the execution time of the Bees algorithm is 0.25 seconds for 100 iterations, while the execution time for the genetic algorithm and the gradient descent algorithm are 27 minutes and 1.7 seconds [1], respectively. The results for the genetic algorithm and the gradient descent algorithm in [1] are 3.5E-3 and 2.6E-3, which are much worse than the results for the genetic algorithm and the gradient descent algorithm in table 3. Table 5 shows the CPU benchmarks comparison of two machines we used and [1] used. Although the CPU Mark score is a relative value and real life may not always reflect the CPU Mark, we would say that our machine is about 14.7 times faster than what [1] used. Considering machine performance difference, the Bees algorithm still works lot faster than the genetic algorithm and the gradient descent algorithm. Our experiments also show that when iterations increase from 100 to 1000, the mean square errors of each variance are almost same, although the execution time increases from 0.25 seconds to 2.5 seconds. We can conclude that the bees algorithm converges quickly and is stable after 100 iterations.

TABLE III
COMPARISON WITH OTHER ALGORITHMS

σ^2	Gradient Descent [5]		Genetic Algorithm [3]		Bees algorithm	
	no. iter	E	no. iter	E	no. iter	E
0.0	8	1.4E-7	1000	0.00141472	100	1.47E-06
0.00096	10	0.00083	1000	0.0014723	100	0.000509
0.00125	11	0.0108	1000	0.00141241	100	0.001106
0.00625	11	0.0530	1000	0.00183267	100	0.004566
0.01250	9	0.1054	1000	0.00241865	100	0.009878

TABLE IV
RESULTS OF FUZZY MEASURE

μ	**0.0**	**0.00096**	**0.00125**	**0.00625**	**0.01250**
$\mu(\{1\})$	0.10438	0.0800392	0.0764291	0.05845	0.0725738
$\mu(\{2\})$	0.208948	0.200362	0.272064	0.127179	0.235488
$\mu(\{3\})$	0.239601	0.246264	0.305229	0.408975	0.296089
$\mu(\{4\})$	0.666377	0.637343	0.717156	0.663812	0.671832
$\mu(\{1,2\})$	0.293485	0.347558	0.331216	0.337807	0.235542
$\mu(\{1,3\})$	0.321726	0.279074	0.340944	0.409823	0.457894
$\mu(\{1,4\})$	0.729298	0.759373	0.718045	0.66487	0.673014
$\mu(\{2,3\})$	0.420896	0.417667	0.421213	0.484186	0.296426
$\mu(\{2,4\})$	0.809941	0.816434	0.719934	0.858011	0.764084
$\mu(\{3,4\})$	0.82076	0.797741	0.809397	0.790631	0.763582
$\mu(\{1,2,3\})$	0.499038	0.47359	0.458572	0.491641	0.459702
$\mu(\{1,2,4\})$	0.864815	0.823745	0.881903	0.935657	0.873362
$\mu(\{1,3,4\})$	0.881175	0.881366	0.833616	0.842672	0.937774
$\mu(\{2,3,4\})$	0.946263	0.953104	0.943259	0.996405	0.893088

TABLE V
CPU BENCHMARKS COMPARISON [16]

CPU Name	Passmark CPU Mark (higher is better)
Intel Xeon E5540 @ 2.53GHz	4613
Intel Pentium 4 2.40GHz	314

VI. CONCLUSION AND FUTURE WORK

In this article, we proposed to use an adapted Bees algorithm to extract fuzzy measures from sample data. The advantage of using the Bees algorithm is that it performs a global search on the search space and reduces the risk of falling into a local optimum. In comparison with the genetic algorithm and the gradient descent algorithm, the experimental results show that the Bees algorithm is faster and provides at least similar accuracy as or better than existing algorithms. The experimental results also indicate that the Bees algorithm provides a stable result in a very short time.

In the future, we plan to combine the Bees algorithm with a global constraint solver that could help to reduce the search space quickly. As a result, we expect that optimal results will be reached within a shorter time and the accuracy of the results won't be lost. A next topic would be use this algorithm to some subfamily of fuzzy measures, e.g., 2-additive fuzzy measures and apply to some real applications, such as cancer diagnosis and prognosis, finance, or survey results analysis for mechanical engineering.

Acknowledgment

The work presented here was partially supported by NSF grant IIS0852066 through the involvement of our second author, who was an NSF REU scholar in Summer 2010.

References

[1] S. Alavi, J. Jassbi, P. Serra, and R. Ribeiro. "Defining fuzzy measures: a comparative study with genetic and gradient descent algorithms". *Intelligent Engineering Systems and Computational Cybernetics*, pp. 427-437, 2009.

[2] M. Ceberio and F. Modave. "An interval-valued 2-additive Choquet integral for multicriteria decision making". In *Proceedings of 10th Conference IPMU*. 2004

[3] E.F. Combarro and P.Miranda. "Identification of Fuzzy Measures from Sample Data with Genetic Algorithms", Computers & Operations Research , vol. 33(10), 3046-3066, 2006.

[4] M. Grabisch, H.T. Nguyen and E. A. Walker. Fundamentals of uncertainty calculi with applications to fuzzy inference. Kluwer Academic, Dordrecht, 1995.

[5] M. Grabisch. "A New Algorithm for Identifying Fuzzy Measures and Its Application to Pattern Recognition". *Proc. Seventh IEEE Int'l Conf. Fuzzy Systems*, vol. 1, pp. 145-150, 1995.

[6] M. Grabisch. "The Application of Fuzzy Integrals in Multicriteria Decision Making". *Euro. J. of Operational Research*, 89:445-456, 1996.

[7] M. Grabisch and M.Roubens. "Application of the Choquet integral in multicriteria decision making". in *Fuzzy Measures and Integrals: Theory and Applications*, M. Grabisch, T. Murofushi, and M. Sugeno. Physica Verlag, pp. 348-374, 2000.

[8] F. Modave. "A measurement theory perspective for MCDM".

[9] D. Pham, A. Ghanbarzadeha, E. Koc, S.Otri, S. Rahim, and M. Zaidi. "The bees algorithm-a novel tool for complex optimization problems". In *Intelligent Production Machines and Systems* pp. 454-459, 2006.

[10] E. Takahagi. "λ fuzzy measure identification methods using λ and weights". 2005.

[11] E. Takahagi. "A fuzzy measure identification method by diamond pairwise comparisons and transformation". *Fuzzy Optimization and Decision Making*, vol. 7(3), pp. 219-232, 2008.

[12] J. Wang and Z. Wang. "Using Neural Networks to Determine Sugeno Measures by Statistics". Neural Networks, vol. 10 (1), 183-195, 1997.

[13] W. Wang, Z. Wang and G. J. Klir. "Genetic Algorithms for Determining Fuzzy Measures from Data". Joural of Intelligent and Fuzzy Systems 6 (1998) 171-183.

[14] Z. Wang, K. Leung and J. Wang. A Genetic Algorithm for Determing Nonadditive set functions in information fusion. *Fuzzy Sets and Systems*, 102 (1999) 463-469.

[15] "Usage: Fuzzy measure-Choquet integral calculation system (λ fuzzy measure and sensitivity analysis)". Available: http://www.isc.senshu-u.ac.jp/ thc0456/Efuzzyweb/mant2/mant2.html

[16] "CPU Benchmarks". Available: http://www.cpubenchmark.net/cpu_list.php