Integrating fuzzy case-based reasoning and particle swarm optimization to support decision making

Nesrine BEN YAHIA[1], Narjès BELLAMINE[1] and Henda BEN GHEZALA[1]

[1] RIADI Laboratory, National School of Computer Sciences, University Campus Manouba, 2010, Tunisia

Abstract

Case-based reasoning (CBR) is a useful technique to support decision making (DM) by learning from past experiences. It solves a new problem by retrieving, reusing, and adapting past solutions to old problems that are closely similar to the current problem. In this paper, we combine fuzzy logic with case-based reasoning to identify useful cases that can support the DM. At the beginning, a fuzzy CBR based on both problems and actors' similarities is advanced to measure usefulness of past cases. For efficiency, we need an optimal design of membership functions of fuzzy sets. Then, we rely on a meta-heuristic optimization technique i.e. Particle Swarm Optimization to adjust the parameters of the inputs and outputs fuzzy membership functions.

Keywords: *Decision Making Support, Case-Based Reasoning Fuzzy Logic, Particle Swarm Optimization.*

1. Introduction

This paper presents a framework that combines case-based reasoning (CBR), fuzzy logic and particle swarm optimization (PSO) to build an intelligent decision support model.

Decision making (DM) whenever and wherever it is happening is crucial to organizations success. DM represents process of problem resolution based on four phases as it is proposed by [1] and revisited by [2]: intelligence, design, choice and review. In the intelligence phase, the problem is identified. In the design phase, the proposed alternatives or solutions are generated. In the choice phase a solution is selected. Finally, in the revision phase the choice is revised and an intelligent feedback permits to correct errors.

In order to make correct decision, individuals and teams within organizations need support to make that more effective and efficient. In this paper, we use the CBR technique to support DM. CBR is a means of solving a new problem by reusing or adapting solutions of old similar problems [3]. It solves a new problem by retrieving, reusing, and adapting past solutions to old problems that are closely similar to the current problem [4].

The most important part of a case-based reasoning system is the selection of the similarity measurement, because, if the one selected is not the right, the system will generate erroneous outputs. To address this problem, Main et al. in [5] explain how fuzzy logic applies to CBR.

Fuzzy logic is similar to the process of human reasoning and it lets people compute with words [6]. The fuzzy approach is useful when the information in hand is too imprecise to justify, and second, when there is a tolerance for imprecision which can be exploited to achieve high performance, low solution cost, and better rapport with reality [6].

Thus, in this paper we combine the use of fuzzy logic and case-based reasoning. Then, in order to make more efficiency, an optimal design of membership functions of fuzzy sets is desired [7]. Therefore, we rely on a meta-heuristic optimization technique i.e. Particle Swarm Optimization to adjust the parameters of the inputs and outputs fuzzy membership functions. We select this technique thanks to its ability to provide solutions efficiently with only minimal implementation effort and its fast convergence [8].

This paper proposes an integrated framework for intelligent decision support that combines case-based reasoning, fuzzy logic and particle swarm optimization. The outline of this paper is as follows. In section two, we begin with a brief background to illustrate the crucial concepts involved in case-based reasoning and fuzzy logic followed by a discussion of the usefulness of combining these two techniques to support decision making. In section three, we present our fuzzy case-based reasoning process. In section four, PSO is used to optimize the proposed process by optimizing the membership functions of fuzzy sets.

2. Background

2.1 Overview of Case based reasoning (CBR)

With Case-based reasoning (CBR), we can learn from past experience and avoid repeating mistakes made in the past. CBR process aims to select past cases based on their degree of match and usefulness to the current situation and it regroups four key issues: (a) identifying key features, (b) retrieving similar cases in the case base, (c) measuring case similarity to select the best match, and (d) modifying and adapting the existing solution to fit the new problem [9].
The retrieval stage, where the system must find in a case base the best matching case or cases, is the most important phase of a case-based reasoning system. In this stage, the similarity measurement is selected; a high degree of similarity measurement implicates a high usefulness and an excellent reason for adaptation. The performance of the system is based on this selection, because, if the one selected is not the appropriate, the system will produce erroneous results. To address this problem, Main et al. in [5] explain how fuzzy logic applies to CBR.

2.2 Overview of fuzzy logic

According to classic or crisp logic each proposition must either be true or false, however fuzzy logic [6] is a solution to represent and treat the uncertainty and imprecision using linguistic terms represented by membership functions. For example, we can consider length as a linguistic variable with three possible linguistic values {short, medium, tall}.
Fuzzy logic process manipulates input and output variables and is based on three phases: fuzzification, fuzzy inference, and defuzzification. Fuzzification transforms the crisp values into fuzzy values, maps actual input values into fuzzy membership functions and evaluates each input's grade of membership in each membership function. The membership function of a fuzzy set A is defined by $\mu_A(x)$ where $\mu_A(x) \in [0, 1]$ and it represents the degree of membership of x to the fuzzy set A.
Each membership function is illustrated by a graphical representation of the degree of participation of each input in a set. Triangular membership functions feature, which is used in this paper, is characterized by a mathematical simplicity. It is specified by three parameters {a, b, c} where for each value x the membership function $\mu_A(x)$ is described as shown in figure 1. These parameters values are obtained from experts' knowledge.

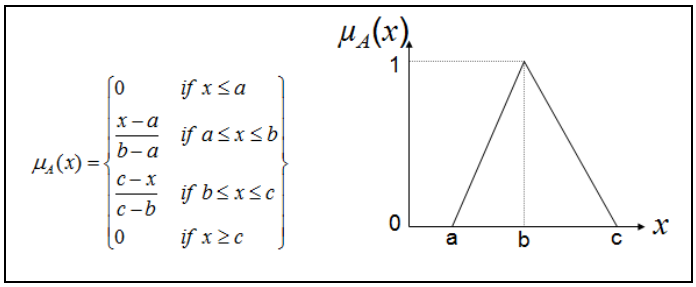

Fig. 1 Triangular membership functions representation.

The second phase in the fuzzy logic process involves the inference of the input values. The fuzzy inference system generates conclusions from the knowledge-based fuzzy rule set of IF-THEN linguistic statements. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. The final phase is the defuzzification where fuzzy results are converted into crisp values.

2.3 Using fuzzy logic to support CBR

There are several advantages of using fuzzy logic techniques to support the case-based reasoning in the retrieval stage [10]. First, it simplifies comparison by converting numerical features into fuzzy terms. For example, we can convert the length of a person into a categorical degree (e.g., short, medium, or tall). Second, fuzzy sets allow simultaneous multiple indexing of a case on a single feature in different sets with different degrees of membership which increases the flexibility of case matching. For example, a 1m50 length person may be classified as short with 0.25 degree, medium with 0.75 degree and tall with 0 degree. This allows the case to be considered as a candidate when we are looking for either a short or a medium length person.

3. Proposal of a Fuzzy Case-Based Reasoning (FCBR) to support DM

Our work consists in proposing a fuzzy case-based reasoning to support decision making. The DM support consists in evaluating cases and identifying useful ones.
In this section, we show how fuzzy logic is applied in our case based reasoning systems.

3.1 Similarity measurement in CBR

In this paper, we consider a case i as a set of (A_i, P_i, Alt_i ,S_i) where A_i represents the actor involved in the case i (who lived the experience), P_i represents the problem, Alt_i

represents the different proposed alternatives to solve P_i and S_i represents the choice among alternatives so $S_i \in Alt_i$.

Then, to measure similarity, we distinguish between two types of similarity: one between current situation problem and problems saved in past cases and another between current situation actor and actors saved in past cases.
Similarity between two elements i and j (two problems or two actors) is computed using the function **Calcul_similarity(element i, element j)** described as follows:

Float function Calcul_similarity(element i, element j) {
W(i, j) = 0;
For each element attribute a do
If a is nominal, mono-valued and i.a = j.a then
W(i, j) = W(i, j) +1;
Else if a is nominal, multi-valued then
For each value i.a.v = 1…k do
For each value j.a.w = 1…m do
If i.a.v = j.a.w then W(i, j) = W(i, j) +1;
End if
Break;
End for
End for
Else if a is continuous then
W(i, j) = W(i, j) +1– α |i.a – j.a|;
End if
End for
Return W(i, j);
}

We distinguish between two types of an attribute: nominal (as topics of interest) and continuous (with continuous values as the attribute age). For nominal ones we distinguish here between mono-valued and multi-valued. Mono-valued attribute is an attribute with one possible value such as the nationality. Multi-valued attribute is an attribute with several possible values such as the preferences or topics of interest.

For example, we will apply this function to measure similarities between actors' attributes (an element consists in an actor) then we consider that each actor A_i, is characterized by the set of attributes <languages, nationality, topics of interest, age>. For the nominal attributes (Languages, Topics of interest and Nationality) the weight is incremented by 1 for each similar attributes but for the continuous attribute (Age), we set α to 0.035 (α =1/28 where 28 is the difference between the minor age 25 and the greater one 53).
Then, we suppose given a cases base containing for each case, the actor, the problem, alternatives and solution. In addition, we suppose that an actor $a_{current}$ characterized by a set of attributes as follows << Arabic, English, French>, < Tunisian>, < CSCW, KM, DM>, <25>> needs support to make decision in a problem so he relies on case-based reasoning to find useful past cases. In table 1, we illustrate the attributes of past actors cases then for each one we compute his/her similarity with $a_{current}$.

Table 1: calculus of similarities between $a_{current}$ and actors saved in past cases

Actors attributes of past cases				***Similarity with $a_{current}$***
Languages	***Nationality***	***Topics of interest***	***Age***	
Arabic, French	Tunisian	IR, DB, KM	30	0.53
Arabic, French	French	AI, KM	28	0.4
English, French	French	CSCW, BPM	42	0.32
Arabic, English, Spanish	Spanish	SOA, SMA	35	0.21
Arabic, English, French	Tunisian	KM, DM, KM	25	1
English, French	French	DB, SGBD	53	0.12
French, Spanish	Spanish	SE, OS	29	0
French, Spanish	French	WWW	45	0.1
Arabic, English, French	English	Sociology, CSCW	35	0.51

Problems similarities are calculated with the same manner. The global similarity between current and past case is then the sum of the similarity between current and past problem with the similarity between current and past actor.
These values are crisp so each case is classified as useful or not useful for the current situation. In fact, if we propose a threshold t then each case with the degree of similarity exceeds t is classified as useful else it is not useful.
In next section, we rely on fuzzy logic to fuzzy these values and represent uncertain cases.

3.2 Fuzzification

In the proposed fuzzy logic process, two inputs are considered which are likeness and closeness and one output which consist in usefulness.

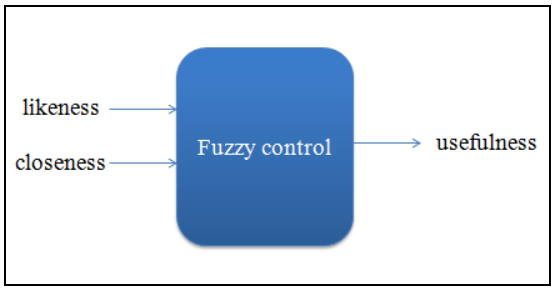

Fig. 2 The proposed fuzzy control.

Likeness and closeness variables represent respectively the values of problems similarities and actors' similarities. As it is shown in figure 3, each variable has three linguistic values: likeness has the values {low, medium, high}, closeness has the values {minor, average, major} and usefulness has the values {poor, good, excellent}.
The inputs membership functions parameters, in our case, are a_{ij}, b_{ij}, c_{ij} where i $\in$ [1..2] represents i^{th} linguistic variable and j $\in$ [1..3] represents j^{th} linguistic value of i^{th} linguistic variable and the outputs membership functions parameters aj, bj, cj where j $\in$ [1..3].

Table 2: Input and output parameters with an example

Linguistic variables	***Linguistic values***	***Parameters***	***Example***
likeness	low	a_{11}, b_{11}, c_{11}	0, 2.5, 5
	medium	a_{12}, b_{12}, c_{12}	2.5, 5, 7.5
	high	a_{13}, b_{13}, c_{13}	5, 7.5, 10
closeness	minor	a_{21}, b_{21}, c_{21}	0, 2.5, 5
	average	a_{22}, b_{22}, c_{22}	2.5, 5, 7.5
	major	a_{23}, b_{23}, c_{23}	5, 7.5, 10
usefulness	poor	a_1, b_1, c_1	0, 2.5, 5
	good	a_2, b_2, c_2	2.5, 5, 7.5
	excellent	a_3, b_3, c_3	5, 7.5, 10

The membership functions of the three variables with the exposed values in the example are given in figure 3 using triangular membership functions feature. Then, figure 4 illustrates the code of the example proposed in table 2 using FCL (Fuzzy Control Language), which is a standard for Fuzzy Control Programming.

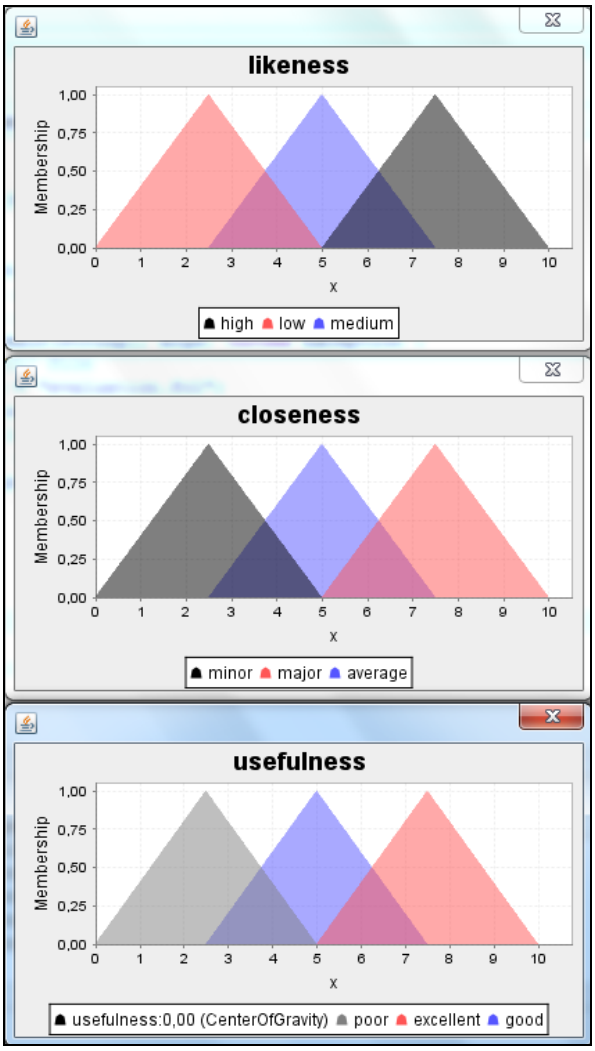

Fig. 3 Membership functions representation.

```
// Fuzzify input variable 'likeness'
FUZZIFY likeness
    TERM low := trian 0 2.5 5 ;
    TERM medium := trian 2.5 5 7.5 ;
    TERM high := trian 5 7.5 10 ;
END_FUZZIFY

// Fuzzify input variable 'closeness'
FUZZIFY closeness
    TERM minor := trian 0 2.5 5 ;
    TERM average := trian 2.5 5 7.5;
    TERM major := trian 5 7.5 10;
END_FUZZIFY
```

Fig. 4 FCL fuzzy inputs code of the proposed example.

3.3 Inference

The fuzzy inference system is the responsible for drawing conclusions from the knowledge-based fuzzy rule set of IF-THEN linguistic statements. In this paper, the MIN-MAX inference method is used. It consists in using the operator min for AND and the operator max for OR. Figure 5 illustrates FCL rules used in our fuzzy control system.

```
RULE 1 : IF likeness IS low AND closeness IS minor
         THEN usefulness IS poor;

RULE 2 : IF likeness IS medium AND closeness IS minor
         THEN usefulness IS poor;

RULE 3 : IF likeness IS high AND closeness IS major
         THEN usefulness is excellent;

RULE 4 : IF likeness IS high AND closeness IS minor
         THEN usefulness IS excellent;

RULE 5 : IF likeness IS low AND closeness IS average
         THEN usefulness IS poor;

RULE 6 : IF likeness IS medium AND closeness IS average
         THEN usefulness IS good;

RULE 7 : IF likeness IS high AND closeness IS average
         THEN usefulness IS excellent;

RULE 8 : IF likeness IS low AND closeness IS major
         THEN usefulness IS good;

RULE 9 : IF likeness IS medium AND closeness IS major
         THEN usefulness IS excellent;
```

Fig. 5 FCL rules used in our fuzzy control system.

3.4 Defuzzification

Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. Figure 6 presents FCL code of defuzzification of the output variable usefulness using Center Of Gravity defuzzification method.

```
// Defzzzify output variable 'usefulness'
DEFUZZIFY usefulness
    TERM poor := trian 0 2.5 5 ;
    TERM good := trian 2.5 5 7.5 ;
    TERM excellent := trian 5 7.5 10 ;
    // Use 'Center Of Gravity' defuzzification method
    METHOD : COG;
    DEFAULT := 0;
END_DEFUZZIFY
```

Fig. 6 FCL fuzzy output code of the proposed example.

3.5 Tests

In this section, we aim to test the proposed fuzzy logic process with setting different values of input variables (likeness and closeness) and getting values of output variable (usefulness) given in table 3. The graphical results are exposed in figure 7 where we display for each test the value of usefulness which consists in the centre of gravity of the result shape.

Table 3: data tests

Input variables		***Output variable***
likeness	closeness	Usefulness
2.0	5.0	2.500000000000005
5.5	5.5	5.603448275862117
5.2	2.1	3.180850532598447
2.5	4.6	2.500000000000006
1.5	8.3	5.000000000000018
3.9	1.8	2.500000000000021

Insert figure 7 here

4. PSO based FCBR

The success of fuzzy logic process depends essentially on its parameters such as the fuzzy membership functions.
In order to get more efficiency, we point to improve the proposed fuzzy case-based reasoning by optimally adjusting the input and output parameters. We propose to use particle swarm optimization which consists of a meta-heuristic technique optimization. This adaptation of PSO for adjusting parameters is inspired from [11] where PSO is used in to optimize multiple faults fuzzy detection.

4.1 Overview of Particle Swarm Optimization

Particle Swarm Optimization (PSO) is introduced by Kennedy and Eberhart in [12] as a new evolutionary computation technique inspired by social behavior simulation of bird flocking or fish schooling. We select this technique thanks to its ability to provide solutions efficiently with only minimal implementation effort and its fast convergence [8]. PSO has attracted much attention and been widely used in complex function optimization, neural network training and data mining.
The population in PSO is called a swarm where the individuals, the particles, are candidate solutions to the optimization problem in the multidimensional search space (D dimensional). Each dimension of this space represents a parameter of the problem to be optimized.
For each iteration t, every particle i is characterized by its position $x_i(t)$ and its velocity $v_i(t)$ which are usually updated synchronously in each iteration of the algorithm.

A particle adjusts its velocity according to its own flight experience and the flight experience of other particles in the swarm in such a way that it accelerates towards positions that have had high objective function or the fitness function values in previous iterations.
There are two kinds of position towards which a particle is accelerated in common use. The first one, a particle's personal best position achieved up to the current iteration, is called $P_{best}i$. The other is the global best position obtained so far by all particles, called G_{best} [13].
There have been several versions of Particle Swarm Optimization. We choose in this paper one of the basic versions as it focuses on cooperation rather than competition and is characterized by no selection. In this context, we choose the PSO version with constriction factor for its speed of convergence [14]. In this case, the modified position $x_i(t+1)$ and velocity $v_i(t+1)$ at the iteration t+1 for particle i can be calculated using the current velocity, $P_{best}i$ and G_{best} as shown in the following formulas:

$$x_i(t+1) = x_i(t) + v_i(t) \quad (1)$$

$$V_i(t+1) = K * (v_i(t) + c_1 * r_1 * (P_{best}i - x_i(t+1)) + c_2 * r_2 * (G_{best} - x_i(t+1)) \quad (2)$$

Where K: the constriction factor given by
$K = 2/ | 2 - c - \sqrt{c2 - 4c} |$ With c=c1+c2 and c>4.
r_1 and r_2: random numbers between 0 and 1.
c_1: self confidence factor.
c_2: swarm confidence factor.

4.2 PSO to support FCBR

In this approach, we rely on PSO to optimally adjust the inputs' parameters: a_{ij}, b_{ij}, c_{ij} where $i \in [1..2]$ represents i^{th} linguistic variable and $j \in [1..3]$ represents j^{th} linguistic value of i^{th} linguistic variable and the output's membership functions parameters aj, bj, cj where $j \in [1..3]$.
For parameters initialization of PSO algorithm, we propose the following values:

- Initial swarm population is composed of 30 particles, Number of Iterations is 1000, c_1= 2.8 and c_2=1.3. The choice of the population size, the values of c1 and c2 is based on the study done in [15].
- Initial positions and velocities of particles are generated by using random values.
- The initial fitness value of each particle is initialized to a big positive constant.
- The local best of each particle $P_{best}i$ is set to its current fitness value.
- The global best value of the position G_{best} is set to the best value of all $P_{best}i$.

The fitness function consists in the objective function and implicates the performance index of a population. For the present problem the aim is to have the proper cases retrieval. Thus, the used fitness function fixes the rate the cases retrieval error which ought to be minimized.

The evolution procedure of PSO Algorithm can be summed up in the following pseudo code:

- First of all, the problem dimension are fixed (PSO parameters)
- Then, the parameters initializations are defined.
- The fitness of each particle is calculated.
- Position and velocity of each particle are updated using (1) and (2).
- If the maximum iteration number is reached, the final solutions are given.
- Otherwise, we loop to calculating membership functions parameters and so on.

5. Conclusions

In this paper, we present a mixed framework that combines case-based reasoning (CBR), fuzzy logic and particle swarm optimization to build an intelligent decision support model. CBR is used to search, retrieve and adapt past useful cases that can help in the current problem resolution. The similarity measurement is based on both problems and actors similarities. Then in order to support CBR, fuzzy logic is used to avoid instable cases retrieval in case of uncertainty and imprecision. In order to make more efficiency, we rely on Particle Swarm Optimization to adjust the parameters of the inputs and outputs fuzzy membership functions.

References

[1] H. Simon, 1977: The New science of management decision. Prentice hall, Englewood-Cliffs.
[2] P. Zaraté, "Des Systèmes Interactifs d'Aide à la Décision Aux Systèmes Coopératifs d'Aide à la Décision : Contributions conceptuelles et fonctionnelles.", HDR dissertation, INP Toulouse, 2005.
[3] C. K. Riesbeck, and R. C. Schank, (1989) Inside Case-Based Reasoning. Lawrence Erlbaum Associates, New Jersey, USA.
[4] A. Aamodt, and E. Plaza, (1994) Case-based reasoning: foundational issues, methodological variations and system approaches, AI Communications, 7, pp. 39-59.
[5] J. Main., T. S. Dillon and R. Khosla. (1996). Use of fuzzy feature vectors and neural vectors for case retrieval in case based systems, NAFIPS 1996 Biennial Conference of the North American Fuzzy Information Processing Society, IEEE, New York, NY, pp. 438–443.

[6] L.A. Zadeh. "Fuzzy Logic = Computing with Words", IEEE Transactions on Fuzzy Systems, Vol. 4, No. 2, May 1996, pp. 103 – 111.
[7] Y. ShengZhou, L. Y. Lai, Optimal design for fuzzy controllers by genetic algorithms, IEEE Transactions on Industry Applications , Volume:36 Issue:1, 2000, pp. 93-97.
[8] K. E. Parsopoulos and M. N. Vrahatis, 2010. Particle Swarm Optimization and Intelligence: Advances and Applications. Information Science Reference (an imprint of IGI Global), United States of America.
[9] J. Kolodner, (1993) Case-Based Reasoning, Morgan Kaufmann, California, USA, 1993.
[10] B. C. Jeng, and T. P. Liang, (1995) Fuzzy indexing and retrieval in case-based systems, Expert Systems With Applications, Vol. 8., No. 1, 1995. Elsevier Science Ltd., 135–142.
[11] I. Fliss and M. Tagina, Exploiting Particle Swarm Optimization in Multiple Faults Fuzzy Detection, journal of computing, volume 4, issue 2, February 2012, pp. 80-91.
[12] R. C. Eberhart and J. Kennedy, " New optimizer using particle swarm theory," in Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, October 1995, pp. 39–43.
[13] J. Yisu, J. Knowles, L. Hongmei, L. Yizeng and D.B. Kell, 2008. The Landscape Adaptive Particle Swarm Optimizer. Applied Soft Computing, 8, pp.295- 304.
[14] M. Clerc, 1999. "The Swarm and the Queen: Towards A Deterministic and Adaptive Particle Swarm Optimization" in Proceedings of the Congress of Evolutionary Computation, Washington, Dc, pp. 1951–1957.
[15] A. Carlisle and G. Dozier, 2001. "An Off-The-Shelf PSO" in Proceedings of the Particle Swarm Optimization Workshop, Indianapolis, Ind, Usa, pp. 1–6.

Nesrine BEN YAHIA is currently preparing her PhD thesis and teaching at the National School of Computer Sciences (Tunisia). She studied computer sciences at National School of Computer Sciences (Tunisia) where she obtained the Engineering degree in 2008 and the Master degree in 2009. Her research interests include decision making, computer supported collaborative work and knowledge management.

Dr. Narjès BELLAMINE is an associate professor of computer science at the University of Manouba, Tunisia, and a researcher in RIADI-GDL laboratory of the national school of computer sciences. Dr. Bellamine received her engineering diploma and master's degree from ENSEEIHT Toulouse and her PhD from University of Toulouse 1, France. Her main research include complexity theory, agent-based modeling and simulation of cooperative systems, computer supported collaborative work and groupware development.

Prof. Henda Ben Ghezala is a Professor of Computer Science at the University of Manouba, Tunisia. She is Director of the RIADI - GDL laboratory of the National School of Computer Sciences. Her research interests lie in the areas of information modeling, databases, temporal data modeling, object-oriented analysis and design, requirements engineering and specially change engineering, method engineering.

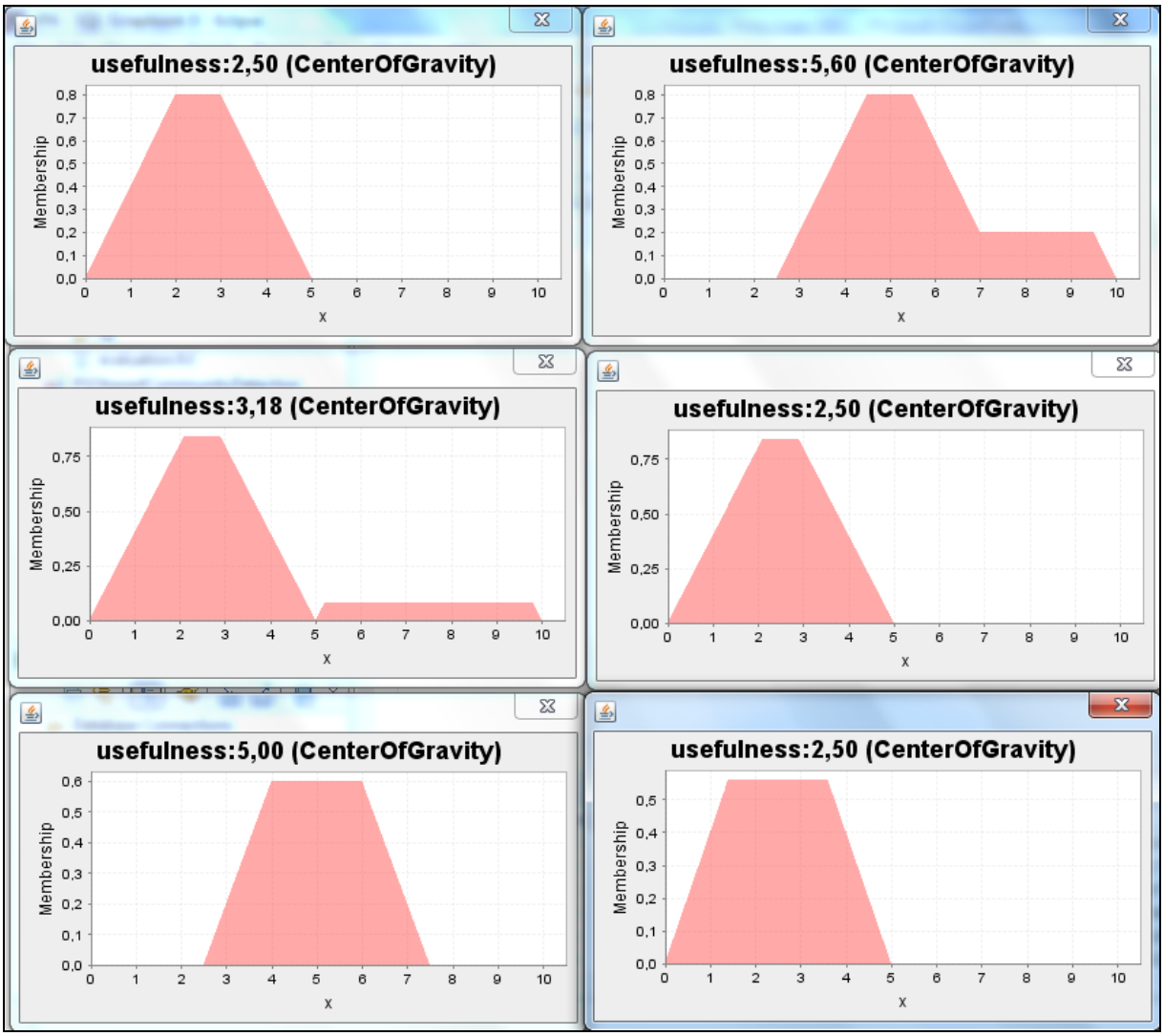

Fig. 7 defuzzification results using center of gravity method.