Towards an Intelligent Decision Making Support

Nesrine Ben Yahia, Narjès Bellamine, and Henda Ben Ghezala

Abstract. This paper presents an intelligent framework that combines case-based reasoning (CBR), fuzzy logic and particle swarm optimization (PSO) to build an intelligent decision support model. 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. Then, we rely on a meta-heuristic optimization technique i.e. Particle Swarm Optimization to adjust optimally the parameters of the inputs and outputs fuzzy membership functions.

1 Introduction

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 (problem identification), design (solutions generation), choice (solution selection) and review (revision phase).

In order to make correct decisions, we use the Case based reasoning (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

Nesrine Ben Yahia · Narjès Bellamine · Henda Ben Ghezala
RIADI Laboratory, National School of Computer Sciences, University Campus Manouba, 2010, Tunisia
e-mail: {Nesrine.benyahia,Narjes.bellamine}@ensi.rnu.tn,
Henda.benghezala@ensi.rnu.tn

A. Abraham and S.M. Thampi (Eds.): Intelligent Informatics, AISC 182, pp. 81–87.
springerlink.com

is the selection of the similarity measurement, because, if the one selected is inappropriate, 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]. It is useful when the information is too imprecise to justify and when this imprecision can be exploited to realize 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].

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 Fuzzy Case-Based Reasoning Supported Decision Making

2.1 Overview of Case Based Reasoning and Fuzzy Logic

Thanks to Case-based reasoning (CBR), we can learn from past experience and avoid repeating mistakes made in the past. CBR process is based on four steps: (a) identifying key features, (b) retrieving similar cases, (c) measuring case similarity and selecting best ones (d) modifying and adapting the existing solution to resolve the current problem [9]. 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. Fuzzy logic process is based on three phases: fuzzification, fuzzy inference, and defuzzification. Fuzzification transforms the crisp values into fuzzy values and maps actual input values into fuzzy membership functions. The membership function of a fuzzy set A is defined by $\mu_A(x)$ and it represents the degree of membership of x to the fuzzy set A. 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 final phase is the defuzzification where fuzzy results are converted into crisp values.

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. Second, fuzzy sets allow simultaneous

indexing of a case on a single feature in different sets with different degrees of membership which increases the flexibility of case matching.

2.2 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 (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.

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 CalculSimilarity(element i, element j) described in Fig. 1.

```
Float function CalculSimilarity(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);
}
```

Fig. 1 Calculus of weights

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, 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). In addition, we suppose given an actor $A_{current}$ characterized by a set of attributes as follows ((Arabic, English, French), (Tunisian), (CSCW, KM, DM), 25), Table 1 illustrates the compute of the similarities between attributes of past actors cases and attributes of $A_{current}$. Problems' similarities are calculated with the same manner. The global similarity between current and past cases is then the

Table 1 Calculus of similarities between $A_{current}$ and actors saved in past cases

Languages	Nationality	Topics of interest	Age	Similarity with $A_{current}$
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	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	PL, CSCW	35	0.51

sum of the similarity between current and past problems with the similarity between current and past actors. As these values are crisp, 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.

2.3 Using Fuzzy Logic to Support CBR

In the proposed fuzzy logic process, two inputs are considered which are likeness and closeness and one output which consists of usefulness. Likeness and closeness variables represent respectively the values of problems' and actors' similarities. 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). In this paper, we use triangular membership functions feature, which is specified by three parameters, as it is characterized by a mathematical simplicity. 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 a_j, b_j, c_j where j $\in$ [1..3].

The fuzzy inference system makes conclusions using the knowledge-based fuzzy rules which is based on a set of IF-THEN linguistic statements. In this paper, the MIN-MAX inference method is used. It consists in using the operators min and max for respectively AND and OR. Table 2 illustrates the matrix inference of our fuzzy control system.

Once the functions are inferred and combined, they are defuzzified into a crisp output which drives the system. In our case, defuzzification of the output variable usefulness is based on Center Of Gravity defuzzification method.

Table 2 The proposed fuzzy inference system

Likeness/Closeness	minor	average	major
low	poor	poor	good
medium	poor	good	excellent
high	excellent	excellent	excellent

3 Using Particle Swarm Optimization in FCBR

3.1 Overview of Particle Swarm Optimization

Particle Swarm Optimization (PSO) is introduced in [11] as a new evolutionary and metaheuristic computation technique inspired by social behavior simulation of fish schooling. We select this technique thanks to its ability to provide solutions efficiently with only minimal implementation effort and its fast convergence [8]. 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. 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. Each particle converges towards positions that had optimized fitness function values in previous iterations by adjusting its velocity according to its own flight experience and the flight experience of other particles in the swarm. There are two kinds of position, $Pbest_i$ that represents personal best position of particle i and G_{best} that represents the global best position obtained by all particles [12]. In this paper, we choose the PSO version with constriction factor for its speed of convergence [13]. In this case, the position $x_i(t+1)$ and velocity $v_i(t+1)$ at the iteration t+1 for particle i are calculated using following formulas:

$$x_i(t+1) = x_i(t) + v_i(t) \tag{1}$$

$$V_i(t+1) = K * (v_i(t) + c_1 * r_1 * (Pbest_i - x_i(t+1)) + c_2 * r_1 * (G_{best} - x_i(t+1)) \tag{2}$$

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

3.2 Using PSO to Optimize the Proposed FCBR

In this approach, we rely on PSO to optimally adjust the inputs' parameters: a_{ij}, b_{ij}, c_{ij} where $i \in [1..2]$, $j \in [1..3]$ and the output's membership functions parameters a_j, b_j, c_j where $j \in [1..3]$. The fitness function, in our case, consists of the error function

of cases retrieval which must be minimized. The evolution of PSO Algorithm can be summed up in the following pseudo code:

1. 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 c_1 and c_2 are based on the study done in [14] and initial positions and velocities of particles are randomly generated.
2. The fitness of each particle is calculated.
3. The local best of each particle Pbest_i and the global best G_{best} values are updated.
4. If the maximum iteration number is reached, the best set of parameters is given.
5. Otherwise, position and velocity of each particle are updated using (1) and (2) and we loop to 2.

4 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) as it is shown in Table 3.

Table 3 Data tests

Likeness	Closeness	Usefulness
2.0	5.0	2.5000000000000053
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	2.5000000000000213