A decision making module for multi-agent systems architectures

Vicenç Torra
Institut d'Investigació en Intel·ligència Artificial - CSIC
Campus UAB s/n, 08193 Bellaterra (Catalunya, Spain)
vtorra@iiia.csic.es

Abstract

In this work we argue on the benefits of including decision making modules in multi-agent systems architectures. We describe one of such modules used to fuse quantitative information. The description includes the syntaxis of the module, together with their options. Several aggregation operators can be used, each of them with a particular set of parmeters (that take different forms according to the particular options selected in the module). Some of the aggregation operators have been adapted to fuse information from an arbitrary number of parameters.
Keywords: Multi-Agent Systems, decision-making, aggregation operators, data fusion.

1. Introduction

When an agent is implemented in a multi-agent environment it needs decision making procedures for considering data supplied by the other agents present in the system or to evaluate data gathered from the real world in which the agent is embedded. In particular, this need is strenghtened when in order to find a particular result the agent has delegated a similar task to several agents solve it using different methodologies or resources. In this case, the results obtained by these systems have to be combined to lead to a single output.

This is the case, for example, of classifier systems when several distributed agents perform the same task of classifying the same data into the same classes using different methodologies (e.g., fuzzy rules, neural networks) or using the same methodology but with different learning methdos. In these systems the final output is computed as a combination of the output of the particular classification agents. See for example [Ishibuchi et al., 1996], [Cordón et al., 1998] as particular implementations of these systems.

Decision making processes are not restricted to these particular systems but are a common part of several agents. See [Matos et al., 1998], [Velasquez, 1998] for other examples. In these systems, a certain value is computed as a combination of several other values, obtained by the agent (computed internaly or retrieved from the real world) or obtained from other agents in the environment.

To ease the definition of multi-agent systems we propose here the inclusion of a decision making generic module in architectures for such systems. In this way, it is not needed to implement from scratch these procedures each time one is needed, and its reuse is garanteed. Another positive effect is that these modules can disseminate several aggregation procedures that have been extensively studied and used in the literature concerning decision making processes but absent from the literature on multi-agents systems. So, these modules can be a bridge between the two communities, as more elaborated aggregation procedures can be used with small effort.

In this paper we describe the syntaxis of a particular decision making module. We give some general considerations corresponding to the implementation of such systems and present a general module with several possible instantiations. We do not consider in this work how a single task is splitted into several subtasks but only how the final computation of these subtasks are combined in a single output. We are neither concerned about inter-module and inter-agent communication protocols. In the following we assume that the module receives a set of values and computes with them an output one.

There exists nowadays several methods for decision making. We present here only some of them. In particular, we restrict to the case when the information supplied by the other agents is numerical (restricted to the [0,1] interval). The inclusion of other aggregation operators in the module is straightforward.

The structure of the paper is as follows. In Section 2 we describe the syntactic structure of the module. In Section 3 we describe the aggregation procedures that can be used in the module. In Section 4 we review the parameters that can be used in each operator in the module.

2. A decision-making module

The generic definition of the decision making module allows several parameters to be especified. Some of the parameters to be instantiated are related to the fact that the module is embedded in an agent that interacts with some other ones. Some of the implicit assumptions in our approach are given below. They are based to general architectures for multi-agent systems.

• A single agent can perform at the same time different tasks, and several of them could apply decision making procedures. Therefore the module should specify to which task concerns.

• Communication among agents is asyncronous: agents communicate the result of a computation to the agent that has made the corresponding query when this is available. Therefore, a module receives answers along its existence.

• Due to the fact that communication is asyncronous, the knowledge that a module can use increases as time passes. We assume that an agent can revise previous computed results on the basis of new information. Therefore, previously considered inputs can be disregarded when new values arrive and a new output is computed and propagated to whom corresponds. This follows in some sense a data-flow approach and also allows to implement any-time algorithms [Dean et al., 1988] as new and better outputs can be obtained as time passes.

• It is assumed that input values can have some uncertainty. Due to this it is allowed that the decision making module gives as result a value with a measure of its uncertainty. The decision making module can manipulate the certainty values attached to input values to compute the certainty value attached to the output.

• It is assumed that the module knows who (another agent or a particular submodule of the module) has generated a particular input. Therefore, when generating the output it has to be said which module has generated it.

Once all the implicit assumptions of our approach have been given, we consider the general structure of the module. This general structure is given in Figure 1. The syntaxis of the tokens that appear in Figure I are given in Figure II. The meaning of the fields in the decision making module are as follows:

• The field `task:` corresponds to the task that the decision module is performing, and its corresponding identifier is used to send the result to the agent or module that has asked for it. Therefore, in the field `output-as:` the subfield `task: <id-number>` refers to the same task. Instead, the field `subtask:` corresponds to the identified that has been given to the agents to whom some work has been delegated. The identifier attached to this field is used to filter input messages.

• The field `decision-type:` defines in which domain input and output values are expressed. We have restricted to be the unit interval with real values.

• The fields `output:` defines which of the general procedures for computing the output should be used. Three alternatives are considered: `first-input`, `fixed-using-n-input` and `continuous`. The first one corresponds to give as output the first value that arrives to the module. In this case the definition of any aggregation procedure is meaningless. The second one `fixed-using-n-input` corresponds to compute the output from the first n input values arrived to the module. The latter option `continuous` is to be used when the output is computed using the information available. When it is used the `fixed-using-n-input` option, the dimension field is used to settle the value n. This parameter can be smaller than the number of agents that can return values to the module.

• The fields `revision-allowed:` and `revision-overwrite:` correspond to whether the output has to be changed when an input value from a particular agent (or module) is revised. The first field says whether the output has to be revised, while the second one selects among two alternatives. One of the alternatives (when `revision-overwrite` is set to true) is to remove the previous value received by the agent and recompute the new output value from scratch, and the other (when `revision-overwrite` is set to false) is to add the new value without removing the previous one. The second alternative is less accurate but it is simpler in some aggregation operators from a computational point of view. The values of these fields are only left free for `continuous` output. For `first-input` output, revision is not allowed while for `fixed-using-n-input` output revision is always performed using `overwrite` option.

Initials	Meaning
AM	Arithmetic Mean
WM	Weighted Mean
OWA	Ordered Weighted Aggregation Operator
WOWA	Weighted Ordered Weighted Aggregation operator
WOWA++	Modified WOWA operator
WMAX	Weighted Max
WMIN	Weighted Min

Table I. Aggregation operators considered in the decision making module

• The field `method:` selects the aggregation procedure to be used in the decision-making module. Several ones are implemented and they are listed in Table I and defined in Section 3. The field `parameters:` is used to settle the parameters of these aggregation operators. As the number and the type of parameters depend on the operator selected only a general form for them can be given. Table II gives the parameters allowed for the aggregation operators implemented.

• The next two fields correspond to certainty values. If certainty values are to be used, the user has to select their type. We have restricted them to be real numbers in the unit interval although other alternatives are conceivable. For example, values in a multi-valued scale. Three methods to compute the certainty value of the output value are considered: the same aggregation procedure used to compute the output (when the `output-aggregation-method` option is selected), the maximum of the certainty values attached to the inputs (when the `maximum` option is selected), and to assign a constant value of 1 (when the `1` option is selected).

• The last field (that, in fact, has not to be included in the module) corresponds to the structure of the output message. It includes the information computed by the module that is sent to the agent that needs it.

```
(Decision-making-module
        task:                   <id-number>
        subtasks:               <id-number>
        decision-type:          <type>
        output:                 <output-proc.>
        dimension:              <integer>
        revision-allowed:       <boolean>
        revision-overwrite:     <boolean>
        method:                 <predefined-methods>
        parameters:             (<parameter>* )
        certainty:              <certainty-type>
        certainty-method:       <certainty-method>
        ;; output-as:      (send-result task: <id-number> at-time: <time-tag> computed-by: <id-agent>
        ;;                      is: <result> with-certainty: <value>)
)
```

Figure I. Syntaxis of a decision making module

```
<id-number>                 ::= T-{0|1|...|9}*
<type>                      ::= (real restricted_to: [0,1])
<output-proc.>              ::= first-input | fixed-using-n-input | continuous
<boolean>                   ::= true | false
<predefined-methods>        ::= - | AM | WM | OWA | WOWA | WOWA++ | WMAX | WMIN
<parameter>                 ::= LISP S-expression
<certainty-type>            ::= none | (real restricted_to: [0,1])
<certainty-method>          ::= - | output-aggregation-method | maximum | 1
```

Figure II. Tokens corresponding to the syntaxis of a decision making module. The complete name of aggregation methods are given in Table I.

3. The aggregation operators

We review now the aggregation operators presented in Table I. In these definitions we use the weighting vectors **p**=(p_1, ..., p_n), **w**=(w_1, ..., w_n) and **u**=(u_1, ..., u_n). We use the vector **u** to refer to the certainty values received by the module. When no certainty value is received but needed a value of 1 is assumed. Some of the aggregation operators described below are modified versions of standard definitions to allow for aggregation of an arbitrary number of elements.

Definition 1. A mapping AM: K^n -> K is an *arithmetic mean* of dimension n if

$$AM(a_1,...,a_n) = \sum_i a_i / n.$$

Definition 2. Let **p** be a weighting vector of dimension n, then a mapping WM: K^n -> K is a *weighted mean* of dimension n if

$$WM_{\mathbf{p}}(a_1,...,a_n) = \sum_i p_i a_i / \sum_i p_i.$$

Definition 3 [Yager, 1988, 1993]. Let **w** be a weighting vector of dimension n, then a mapping $OWA_{\mathbf{w}}$: K^n -> K is an *Ordered Weighted Averaging (OWA) operator* of dimension n if

$$OWA(a_1,...,a_n) = \sum_i w_i a_{\sigma(i)} / \sum_i w_i$$

where $\{\sigma(1),...,\sigma(n)\}$ is a permutation of {1,..,n} such that $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ for all i=2, ..., n. (i.e., $a_{\sigma(i)}$ is the i-th largest element in the collection a_1,..., a_n).

Definition 4 [Torra, 1997]. Let **p** be a weighting vector of dimension m and **w** a weighting vector of dimension n, then a mapping WOWA:K^k -> K is a *Weighted Ordered Weighted Averaging (WOWA) operator* of dimension k if

$$WOWA(a_1,...,a_k) = \sum_i \omega_i a_{\sigma(i)}$$

where $\{\sigma(1),...,\sigma(k)\}$ is a permutation of {1, .., k} such that $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ for all i=2, ..., k. (i.e., $a_{\sigma(i)}$ is the i-th largest element in the collection a_1,..., a_k), and the weight ω_i is defined as

$$\omega_i = w^*\left(\sum_{j\leq i} p_{\sigma(j)} / \sum_j p_j\right) - w^*\left(\sum_{j<i} p_{\sigma(j)} / \sum_j p_j\right)$$

with w^* a monotone increasing function that interpolates the points $(i/n, \sum_{j\leq i} w_j / \sum_j w_j)$ together with the point (0,0). w^* is required to be a straight line when the points can be interpolated in this way. See [Torra, 1999] for an interpolation method that satisfies these properties.

In this definition we consider k values to combine, while the weighting vectors **p** and **w** have different dimensions. The same idea applies to the OWA operator. In this case, when the dimension of the weighting vector given by the user is not the same of the data to aggregate (n and k, respectively), we proceed as in the WOWA computing first the function W^* from the n weights in **w** and then defining

$$w_i = w^*(i/k) - w^*((i-1)/k)$$

In this way the OWA is defined for an arbitrary number of parameters.

The WOWA++ defined below only differs with the previous operator in that takes into account the certainties supplied by the agents together with their values. Weights **p** are modified according to these certainties **u**.

Definition 5. Let **p** and **w** be two weighting vectors defined as above, and let **u** be a vector of dimension k with the certainty values given by the corresponding sources, then a mapping WOWA++:K^k -> K is a WOWA++ of dimension k if

$$WOWA(a_1,...,a_k) = \sum_i \omega_i a_{\sigma(i)}$$

where $\{\sigma(1),...,\sigma(k)\}$ is defined as before and the weight ω_i is defined as

$$\omega_i = w^*\left(\sum_{j\leq i} (p_{\sigma(j)})^{u_i} / \sum_j (p_j)^{u_i}\right) - w^*\left(\sum_{j<i} (p_{\sigma(j)})^{u_i} / \sum_j (p_j)^{u_i}\right)$$

with w^* defined as in definition 5.

Definition 6 [Dubois et al., 1986]. Let **u** be the vector with dimension n with the certainties attached to the values given by the corresponding sources, then a mapping WMin: K^n -> K is a *weighted min* of dimension n if

$$WMin_{\mathbf{u}}(a_1,...,a_n) = Min_i\ Max(1 - (u_i / max_i\ u_i)\ a_i).$$

Definition 7 [Dubois et al., 1986]. Let **u** be the vector with dimension n with the certainties attached to the values given by the corresponding sources, then a mapping WMax: K^n -> K is a *weighted max* of dimension n if

$$WMax_{\mathbf{u}}(a_1,...,a_n) = Max_i\ Min\ (u_i / max_i\ u_i,\ a_i).$$

4. On the parameters of the aggregation operators

Parameters are context dependent as they are based on the aggregation operator selected in the slot `method:` We have considered parameters being of the form of a LISP s-expression. However, from an abstract point of view each aggregation operator has only a set of admissible parameters. These admissible parameters are given in Table II. We have considered three kind of parameters according to the output procedure being `fixed-using-n-input` (case a in Table II), being `continuous` without `overwrite` option (case b in Table II) or being `continuous` with `overwrite` option (case c in Table II).

In Table II, m-dimensional weighting vectors are defined by a list of pairs of the form (id-agent weight) where id-agent is an identifier for a computation element and weights are real numbers in the unit interval. The dimension of the weighting vector is required to be equal or greater than the dimension fixed by the dimension slot if any.

Definition of weighting vectors does not require normalization. Normalization is performed by the aggregation operator itself.

Initials	Parameters
AM	a, b and c) parameters not allowed
WM	a) m-dimensional weighting vector (defined by means of pairs agent / weight) b) lambda-expression that computes the i-th weight from the pair (i, w) where i is the numbering of the weight and w is the last computed weight (the (i-1)-th weight) c) m-dimensional weighting vector (defined by means of pairs agent / weight)
OWA	a, b and c) n-dimensional weighting vector
WOWA	a, b and c) two classes of weights: one as in the WM and the other as in the OWA
WOWA++	a, b and c) as in the WOWA
WMAX	a) m-dimensional weighting vector (defined by means of pairs agent / weight) b) lambda-expression that computes the i-th weight from the pair (i, w) where i is the numbering of the weight and w is the last computed weight (the (i-1)-th weight) c) m-dimensional weighting vector (defined by means of pairs agent / weight)
WMin	a, b and c) none
WMax	a, b and c) none

Table II. Parameters of the aggregation operators: (a) corresponds to the slot `output:` being `fixed-using-n-input`; (b) corresponds to the slot `output:` being `continuous` without `overwrite` option; (c) corresponds to the slot `output:` being `continuous` with `overwrite` option

5. Conclusions

In this work we have presented a decision making module that includes the definition of a set of aggregation operators. We have argued the suitability of such module and described its syntax together with the operators that can be used.

Acknowledgements

The author acknowledge partial support of the CYCIT project SMASH (TIC96-1138-C04-04).

References

[1] Cordón, O., del Jesús, M. J., Herrera, F., López, E., (1997), Selecting Fuzzy Rule-Based Classification Systems with Specific Reasoning Methods using Genetic Algorithms, *Proc. of the Seventh Int. Fuzzy Systems Association World Congress*, 424-429, Praga.

[2] Dean, T., Boddy, M., (1998), An analysis of time-dependent planning, *Proc. of the Seventh National Conference on Artificial Intelligence*, 49-54, St. Paul, Minnesota, USA.

[3] Dubois, D., Prade, H., (1986), Weighted Minimum and Maximum Operations in Fuzzy Set Theory, *Information Sci.* 39 205-210.

[4] Ishibuchi, H., Morisawa, T., Nakashima, T., (1996), Voting Schemes for Fuzzy-Rule-Based Classification Systems, *Proc. of the Fifth IEEE Int. Conference on Fuzzy Systems*, 614-620, New Orleans, USA.

[5] Matos, N., Sierra, C., Jennings, N. R., (1998), Determining Successful Negotiation Strategies: An Evolutionary Approach, *Proc. of the Third Int. Conf. on Multi-Agent Systems*, 182-189, Paris, France.

[6] Torra, V., (1997), The Weighted OWA operator, *International Journal of Intelligent Systems*, 12 153-166.

[7] Torra, V., (1999), The WOWA operator and the interpolation function W^*: Chen and Otto's interpolation method revisited, *Fuzzy Sets and Systems*, in press.

[8] Velásquez, J. D., (1998), When Robots Weep: Emotional Memories and Decision-Making, *Proc of the Fifteenth Nat. Conference on Artificial Intelligence*, 70-75, Madison, Wisconsin, USA.

[9] Yager, R. R., (1988), On ordered weighted averaging aggregation operators in multi-criteria decision making, *IEEE Trans. on SMC*, 18 (1988) 183-190.

[10] Yager, R. R., (1993), Families of OWA operators", *Fuzzy Sets and Systems*, 59 (1993) 125-148.