A Comparison of Active Set Method and Genetic Algorithm Approaches for Learning Weighting Vectors in Some Aggregation Operators

David Nettleton,[1,]* Vicenc Torra[2,]†
[1] *IBM Global Services, Av. Diagonal, 571, 08029 Barcelona, Spain*
[2] *Institute d'Investigació en Intel · ligència Artificial — CSIC, Campus UAB, 08193, Bellaterra, Catalunya, Spain*

In this article we compare two contrasting methods, active set method (ASM) and genetic algorithms, for learning the weights in aggregation operators, such as weighted mean (WM), ordered weighted average (OWA), and weighted ordered weighted average (WOWA). We give the formal definitions for each of the aggregation operators, explain the two learning methods, give results of processing for each of the methods and operators with simple test datasets, and contrast the approaches and results. © 2001 John Wiley & Sons, Inc.

I. INTRODUCTION

Data fusion and aggregation operators are used in different fields, and in artificial intelligence, to fuse information supplied by different information sources. In recent years, several methods and techniques have been developed to deal with information represented under various forms: numerical, ordinal, etc. For example, in the particular case of the numerical setting, there exist, among others, the weighted mean (WM) (characterized in Ref. 1), the ordered weighted average (OWA) operator,[2] the weighted OWA (WOWA),[3] the Choquet integral (definitions and properties are given in Ref. 4), and the Fuzzy integral.[5]

Usually, these aggregation operators are parametric with respect to a set of parameters. For example, one or two weighting vectors in the case of the weighted mean, OWA, and WOWA operators, a weighting vector and a fuzzy quantifier in the case of the OWA with importances,[6] or a fuzzy measure in the case of the Choquet and fuzzy integral.

* e-mail: dnettlet@es.ibm.com.
† Author to whom correspondence should be addressed; e-mail: vtorra@iiia.csic.es.

One of the open problems in this area is the determination of the parameters for these functions. This need is of great relevance in order to ease the application of these operators, to extend their use to new problems, and to embed them in new systems. Several approaches have been studied in the literature, some of them, such as Saaty's Analytic Hierarchy Process[7] for the weighted mean or the method defined in Ref. 8 for the OWA operator, are based on a user—or a set of users/experts—that supplies critical information that is afterwards used in a certain way to extract the weights. This approach can be applied when there is some background knowledge about the system we want to model or about the decision process. To deal with the case when this background knowledge does not exist, a different approach similar to a machine learning technique has been developed. The main idea is to learn the parameters from a set of examples. This is the case in Ref. 9 for learning the weighting vector for OWA operators, in Ref. 10 for weighted means and OWA operators, and in Ref. 11 for Choquet integrals. In all these works, a set of examples (defined as a set of input parameters and their corresponding output) are given and, from them, the weighting vectors or the fuzzy measures are inferred.

In addition to the advantages of this approach (not needing background knowledge and automating the whole process through the use of examples), learning the weights allows the extension of these operators to larger problems where the number of parameters is great. This is not possible when all the information has to be supplied by experts as a vast amount of knowledge is then required.

In this paper we study the approach based on a set of examples, and we compare the use of genetic algorithms and an algorithm based on active set methods to learn the parameters for the weighted mean, the OWA operator, and the WOWA operator.

The article is structured as follows: in Section II we give the formal definitions of the OWA, WM, and WOWA operators: Section III describes two forms of learning the weights, active set methods (ASM) based approach and genetic algorithms (GA); Section IV describes a reformulation for examples with a different number of variables; Section V explains advantages and inconveniences of the ASM and the GA approaches; Section VI summarizes and gives some conclusions.

II. PRELIMINARIES

We review here some of the operators that are used later on in this work. Yager[2] introduced the OWA operators and Fodor et al.[12] characterized them. Torra[3] introduced the WOWA operator and its relationship with the Choquet integral is given by Torra.[13]

DEFINITION 1. *A vector* $\mathbf{v} = [v_1 \; v_2 \; \cdots \; v_n]$ *is a* weighting vector *of dimension n if and only if*

$$v_i \in [0,1] \qquad \Sigma_i v_i = 1$$

Definition 2. *Let* **p** *be a weighting vector of dimension* n, *then a mapping* WM: $\mathbb{R}^n \to \mathbb{R}$ *is a* weighted mean *of dimension* n *if* $WM_p(a_1, \dots, a_n) = \Sigma_i p_i a_i$.

Definition 3 (*Yager*, *Ref.* 2). *Let* **w** *be a weighting vector of dimension* n, *then a mapping* OWA_w: $\mathbb{R}^n \to \mathbb{R}$ *is an* ordered weighted averaging (OWA) operator *of dimension* n *if*

$$\mathrm{OWA}_w(a_1, \dots, a_n) = \Sigma_i w_i a_{\sigma(i)}$$

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

Definition 4 (*Torra*, *Ref.* 3). *Let* **p** *and* **w** *be two weighting vectors of dimension* n, *then a mapping* $WOWA$: $\mathbb{R}^n \to \mathbb{R}$ *is a* weighted ordered weighted averaging (WOWA) operator *of dimension* n *if*

$$\mathrm{WOWA}_{p,w}(a_1, \dots, a_n) = \Sigma_i \omega_i a_{\sigma(i)}$$

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

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

with w^* *a monotone increasing function that interpolates the points* $(i/n, \sigma_{j \leq i} w_j)$ *together with the point* $(0, 0)$. *The function* w^* *is required to be a straight line when the points can be interpolated in this way.*

Proposition 1 (*Torra*, *Ref.* 3). *The WOWA operator satisfies the following properties*:

(1) *It is an aggregation operator which remains between the minimum and the maximum.*
(2) *It satisfies idempotency* (*unanimity*).
(3) *It is commutative if and only if* $p_i = 1/n$ *for all* $i = 1, \dots, n$ *such that* $w_i \neq 0$.
(4) *It is monotone in relation to the input values* a_i.
(5) *It leads to dictatorship of the ith value when* $p_i = 1$ *and* $p_j = 0$ *for all* $j = 1, \dots, n$ *but* $j \neq i$.
(6) *It leads to the arithmetic mean when* $p_i = 1/n$ *and* $w_i = 1/n$ *for all* $i = 1, \dots, n$.
(7) *It leads to the weighted mean when* $w_i = 1/n$.
(8) *It leads to the OWA operator when* $p_i = 1/n$.

III. TWO APPROACHES FOR LEARNING THE WEIGHTS

Our approach to learning weights is based on using a set of examples. We assume that each consists of a set of values to aggregate and the corresponding outcome. Let the set of examples be those in Table I. That is, a set of *M*

Table I. Data examples.

a_1^1	a_2^1	a_3^1	$\cdots$	a_N^1	|	m^1
a_1^2	a_2^2	a_3^2	$\cdots$	a_N^2	|	m^2
$\cdots$						
a_1^M	a_2^M	a_3^M	$\cdots$	a_N^M	|	m^M

examples, each one with N input values (the dimension of the aggregation operator is, therefore, N). In this table, a_j^i corresponds to the value supplied by the jth source in the ith example; and m^i corresponds to the outcome of the ith example.

The goal of our approach is to learn the weights that, when used in conjunction with the aggregation operator, return m^j (or a similar value if no exact solution exists) when the input vector is $(a_1^i \; a_2^i \; a_3^i \; \cdots \; a_N^i)$. We compare two approaches, one based on the active set methods and the other based on genetic algorithms. We review them below. In both cases, what we need is a way to determine a good solution. This is equivalent to defining a measure of the goodness of a solution. In our case, this will be done through the accumulation, for each example $(a_1^i \; a_2^i \; a_3^i \; \cdots \; a_N^i \,|\, m^j)$, of the distance between the ideal outcome m^j and the outcome given by the aggregation operator. The distance for each example is computed through the squared difference. Therefore, in our case, the more suitable weighting vectors are those that minimize this expression:

$$D(p) = \sum_{j=1}^{M} \left(\mathbb{C}_p\left(a_1^j, \ldots, a_N^j\right) - m^j\right)^2 \tag{1}$$

where p represents the parameters of the aggregation operator $\mathbb{C}$ (i.e., either the weighting vector $\mathbf{p}$, the weighting vector $\mathbf{w}$, or both in the case of the WOWA operator).

Together with this objective function, a set of restrictions has to be considered when the aggregation function $\mathbb{C}$ is either the weighted mean, the OWA operator, or the WOWA operator. This restriction is that the parameters (the weights) have to define a weighting vector. That is, weights have to add to one and be positive. Therefore, the general problem can be formalized in the following way:

$$\text{Minimize } \sum_{j=1}^{M} \left(\mathbb{C}_p\left(a_1^j, \ldots, a_N^j\right) - m^j\right)^2$$

such that

$$\sum_{i=1}^{N} p_i = 1 \qquad \sum_{i=1}^{N} w_i = 1$$

$$p_i \geq 0 \quad \text{for all } i \qquad w_i \geq 0 \quad \text{for all } i$$

where, in the case of $\mathbb{C}$ being the weighted mean, only restrictions over p apply; in the case of the OWA operator only those over w apply and, in the case of the WOWA, both apply.

A. Active Set Method Based Approach

The problem formulated in this way is a typical optimization problem and there exist different techniques to tackle with it. In the particular case of using a weighted mean or an OWA operator, the problem reduces to a quadratic program and there exist several algorithms that solve the problem with accuracy (see, for example, Refs. 14 and 15); for example, the ones based on active set methods.

Active set methods rely on the simplicity of computing the solution of quadratic problems with linear equality constraints. Based on this, algorithms iterate so that in each step inequality constraints are split into two groups: the first with those that will be treated as active and considered as equality constraints; and the second with those that are ignored. Then, the algorithm moves to an improved point moving on the surface defined by the set of active constraints. At this point constraints can be added to, or removed from, the active set. This process is repeated until the minimum is reached. A detailed analysis of such a method, applied to the learning of weighting vectors for the weighted mean and the OWA operator, is presented in Ref. 10.

The application of these methods to the example introduced by Filev and Yager[9] for the OWA operator resulted in a good solution. The data matrix corresponding to this example is given in Table II. Using the active set method approach, the resulting weighting vector for the OWA operator was

$$w = (0.1031, 0.0, 0.2293, 0.6676)$$

while the solution in Ref. 9, after 150 iterations (based on the use of the gradient technique), was

$$w = (0.08, 0.11, 0.14, 0.67)$$

The error using active set methods was: 0.001256, while, with the gradient technique due to the slow convergence, was 0.002156.

Table II. Data matrix H and solution vector d (taken from Ref. 9).

0.4	0.1	0.3	0.8	|	0.24
0.1	0.7	0.4	0.1	|	0.16
1.0	0.0	0.3	0.5	|	0.15
0.2	0.2	0.1	0.4	|	0.17
0.6	0.3	0.2	0.1	|	0.18

B. Genetic Algorithm Based Approach

An evolutive procedure is a probabilistic algorithm which maintains a population of individuals $\mathbf{P(t)} = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t\}$ for iteration $\mathbf{t}$. Each individual represents a potential solution to the problem being considered, and is normally implemented as a data structure **S**. Each solution $\mathbf{x}_i^t$ is evaluated to give a measure of its "aptitude." Then, a new population is formed (iteration $\mathbf{t+1}$) by selection of the most apt individuals (selection step). Some members of the new population undergo transformations (modification step) using "genetic" operators, which form new solutions. There are unary transformations $\mathbf{m}_i$ (mutation type) which create new individuals by a small change in just one individual ($\mathbf{m}_i$: $\mathbf{S} \rightarrow \mathbf{S}$), and transformations of a higher order $\mathbf{c}_j$ (crossover type), which create new individuals by the combination of parts of various (two or more) individuals ($\mathbf{c}_j$: $\mathbf{S} \times, \ldots, \times \mathbf{S} \rightarrow \mathbf{S}$). After a given number of generations the program converges—with the objective that the best individual represents a solution close to the optimum.

Within the extensive literature in this field, we can highlight the following: two key authors for parameter optimization problems are Rechenberg[16] and Schwefel[17]; Fogel's evolutionary programming[18] is a technique for searching through a space of small finite-state machines; Glover's scatter search techniques[19] maintain a population of reference points and generate offspring by weighted linear combinations. A matrix representation for the chromosome was introduced by Vignaux,[20] and Koza[21] and Michalewicz and Janikow[22] are examples of specific genetic operators to accommodate the problem to be solved. The incorporation of problem specific knowledge has been tackled by authors such as Antonisse and Keller,[23] Forrest,[24] and Fox and McMahon.[25] Other relevant results are given in Refs. 30 and 31.

We can use genetic algorithm techniques to learn the weighting factors for aggregation operators such as WM, OWA, and WOWA from historical data. In the case of WOWA, we could also look for interpolation functions which give the best results, with an adequate parametric representation for the function in the chromosome. A genetic algorithm has, as input, a set of input cases and their respective outcomes (examples), a set of modifiable values (in this case the weighting factors), a set of constraints (in this case the sum of the weighting factors must be equal to 1), and an objective function, which we have defined as the minimum difference between the predicted outcome $m^{j'}$ and the real outcome m^j. We wish to find the weighting factors which best approximate the input and output data, while minimizing the objective function.

The routine in Figure 1 simply goes through all the individuals in the current population and assigns a "fitness" score to each. The fitness score for each individual is calculated by executing the function which calculates the output (in this case the WOWA aggregator) for each of the data cases $(1 \cdots j)$ and with the w and p weight vectors contained in the chromosome of individual **i**.

The chromosome consists of a single vector data structure which holds the w weights and the p weights. Another approach would have been to separate

```
Procedure evaluate
begin
    for all (genetic) individuals do
    begin
        read weights ω and p from current individual i's chromosome
        total_distance ← 0
        for all data cases do
        begin
            read (data input row j)
            real_output ← read (real_output for this case)
            wowa_output ← wowa (weights, data)
            local_distance ← real_output – wowa_output
            total_distance ← total_distance + (local_distance)^2
        end
        individual(i).aptitud ← total_distance
    end
end
```

Figure 1. The basic structure of the evaluation routine.

the *w* and *p* weights into two separate vectors, and maintain them as separate populations. This would give the possibility of converging more rapidly because the different weight types are not mixed by the crossover and mutation operations.

```
Struct chromosome
{
  int gene_vector[1..num_weights];
}
```

In the case of WOWA, gene_vector holds the *w* and *p* weights and num_weights is equal to the number of variables × 2. Likewise, in the case of OWA, gene_vector holds the *w* weights and num_weights is equal to the number of variables. Finally, in the case of WM, gene_vector holds the *p* weights and num_weights is also equal to the number of variables.

Table III. Data matrix H and solution vector d (taken from Ref. 3).

0.7	0.6	0.4	0.3	|	0.50
0.9	0.7	0.5	0.3	|	0.60

If we normalize the ω and ρ values, we guarantee that consistent WOWA's are generated:

$$\omega_i' = \omega_i / \Sigma \omega_i$$
$$\rho_i' = \rho_i / \Sigma \rho_i$$

In Table III the data matrix, used to learn the weights for the WOWA operator using the GA method, is given. Using the GA method approach, the resulting weighting vectors for the WOWA operator were

$$w = (0.47, 0.05, 0.11, 0.37)$$
$$p = (0.15, 0.19, 0.35, 0.31)$$

This was obtained using crossover in 1 point, with random and uniform mutation. The population was 150, the possible gene values ranging between 1 and 10; the crossover rate at 0.85 and the mutation rate at 0.01. The best aptitude was 0.000, run over 100 generations, and the best individual found after 3 generations.

If we increase the range of possible gene values to be between 1 and 100, and increase the population to 300, the following result was found, also with best aptitude 0.000, and taking 9 generations to reach it.

$$w = (0.07, 0.42, 0.07, 0.44)$$
$$p = (0.01, 0.51, 0.31, 0.17)$$

while the solution for the w and p vectors given in Ref. 3 was

$$w = (0.13, 0.37, 0.37, 0.13)$$
$$p = (0.25, 0.25, 0.25, 0.25)$$

The data matrix used to learn the weights for the OWA operator using the GA method is that of Table II. Using the GA method approach, the resulting weighting vectors for the OWA operator were

$$w = (0.07, 0.07, 0.33, 0.53)$$

This was obtained using crossover in 1 point, with random and uniform mutation. The population was 150, the possible gene values ranging between 1 and 10; the crossover rate at 0.85 and the mutation rate at 0.01. The best aptitude was 0.067, run over 100 generations, and the best individual was found after 9 generations.

If we increase the range of possible gene values to be between 1 and 100, and increase the population to 350, the following result was found, with best aptitude 0.061, and convergence in 27 generations:

$$w = (0.09, 0.01, 0.31, 0.59)$$

This result approximates more closely to the ASM method result (given in Sect. III A) than to the result given by Filev and Yager.[9]

In Table IV the data matrix used to learn the weights for the WM operator using the GA method is given. Using the GA method approach, the resulting weighting vectors for the WM operator were

$$p = (0.08, 0.21, 0.29, 0.38, 0.04)$$

This was obtained using crossover in 1 point, with random and uniform mutation. The population was 150, the possible gene values ranging between 1 and 10; the crossover rate at 0.85 and the mutation rate at 0.01. The best aptitude was 0.031, run over 100 generations, and the best individual was found after 7 generations.

If we increase the range of possible gene values to be between 1 and 100, and increase the population to 300, the following result was found, with best aptitude 0.004, and convergence in 18 generations:

$$p = (0.1, 0.2, 0.3, 0.4, 0.0)$$

The values of the resulting p vector are identical to those given in Ref. 10.

The weighted mean was the only operator which showed a significant improvement in the best solution found by varying the parameters to the GA. In

Table IV. Data matrix H and solution vector d (taken from Ref. 10).

0.3	0.4	0.5	0.1	0.2	|	0.30
0.2	0.1	0.4	0.1	0.5	|	0.20
0.2	0.5	0.8	0.0	0.1	|	0.36
1.0	0.5	0.3	0.6	0.7	|	0.53
0.2	0.1	0.1	0.1	0.1	|	0.11
0.7	0.7	0.7	0.7	0.7	|	0.70
0.4	0.8	0.2	0.8	0.6	|	0.58
0.3	0.2	0.1	0.4	0.3	|	0.26
0.6	0.8	0.7	0.2	0.5	|	0.51
0.1	0.5	0.2	0.6	0.4	|	0.41

the case of WOWA and OWA, diverse combinations were tried for mutation rate and crossover rate, but without improvement of the best solution found. Notwithstanding, by changing the population size and allowable values for the gene, we were able to find different weight values for the best solutions with the same aptitude.

We also tried a mutation function with Gaussian distribution instead of random and uniform, and a crossover function with a 2 point crossover instead of 1. In the case of the WM, this reached the same precision as the previous solution (best aptitude 0.004) with the same weights. In the case of OWA, the solutions found were slightly worse (best aptitude 0.068 compared to 0.061). The 2 point crossover can be used to preserve the order of subgroups within the chromosome, when this is significant to the solution to the problem, as in the TSP (travelling salesman problem),[26,27] or in this case, where the weights are ordered with respect to the data values. We think the lack of improvement for OWA in this case was due to the insufficient length of the chromosome. Although the type of problem is appropriate to show an improvement with this method, the total number of genes was insufficient to create significant sub-chains of genes within the chromosomes.

IV. REFORMULATION FOR EXAMPLES WITH A DIFFERENT NUMBER OF VARIABLES

In order to use an aggregation operator to process data with missing values, we have several options open to us. The first option may be to fill in the missing values with a new value. This could be a flag which indicates that the value is missing, or it could be the mean of the rest of the values for that variable in the case of numeric values, or the mode in the case of categorical values. This method works better when there are a large number of cases and thus the mean and mode values are good approximations. In our case, we wish to work with a small number of cases and we cannot guarantee that the mean and mode will be accurate. Thus, we choose to eliminate the missing values (not the whole case). If the cases have $\mathbf{N}$ variables, and, for case $\mathbf{j}$, two variables are missing, then we eliminate those variables for case $\mathbf{j}$ and we pass the $\mathbf{p}$ and $\mathbf{a}$ vectors with $\mathbf{N-2}$ variables to the WOWA operator. The $\mathbf{w}$ vector remains unchanged, and we pass $\mathbf{N}$ of its weights to WOWA. This is because the $\mathbf{p}$ and $\mathbf{a}$ vectors must have the same length: for each $\mathbf{a}$ value there must exist a corresponding $\mathbf{p}$ weight. In contrast, the $\mathbf{w}$ vector is not directly related to the $\mathbf{p}$ and the $\mathbf{a}$ vectors, as it is used to construct the quantifier function. The different lengths of the $\mathbf{a}$ and $\mathbf{p}$ vectors with respect to the $\mathbf{w}$ vector make it necessary to maintain two length counters and pass these to WOWA to be used in the appropriate points of the function code. Of course, the aggregation operator performs the aggregation based only on the $\mathbf{N-2}$ variables received as its input. Thus, a reformulation of the aggregation problem is needed so that it can be applied to a number of variables less than $\mathbf{N}$. This approach has its cost/benefit: on the one hand, we have the possible benefit due to a reduction in noise due to erroneous substitute values. On the other hand, we have the possible cost due to information loss due to the absence to the omitted variables' value. The final decision on whether to choose one option or the other depends on the application and the nature of the data being processed, as well as on the number of cases.

In the case of the WM, the reformulation is

$$D(\mathbf{p}) = \sum_{j=1}^{M} \left(\frac{\sum_{i \in I_j} a_i^j p_i}{\sum_{i \in I_j} p_i} m^j \right)^2$$

where p is the weighting vector $\mathbf{p}$, where I_j is a subset of $\{1, \ldots, N\}$ with the index variables of example j. In the reformulation, we divide by Σp_i in order to renormalize the weights and assure that their sum is equal to 1. The approach for the OWA is similar and we have not detailed it here. In the case of the WOWA, the reformulation is

$$D(p) = \sum_{j=1}^{M} \left(\sum_{i \in I_j} \left(W^* \left(\frac{\sum_{j \leq i;\, j \in I_j} p_s(j)}{\sum_{i \in I_j} p_i} \right) - W^* \left(\frac{\sum_{j < i;\, j \in I_j} p_s(j)}{\sum_{i \in I_j} p_i} \right) \right) a_i^j - m^j \right)^2$$

where p is the weighting vectors $\mathbf{p}$ and $\mathbf{w}$ [i.e., $p = (\mathbf{p} \mid \mathbf{w})$], where I_j is a subset of $\{1, \ldots, N\}$ with the index variables of example j. In the reformulation, we divide by Σp_i in the same manner as for WM and OWA, and the ordering $(\Sigma_{j \leq i} p_{\sigma(j)})$ is now restricted to the subset $i \in I_j$. Note that the interpolation function W^* is the same for all examples $j = 1, \ldots, M$ (it is computed using all weights in $\mathbf{w}$).

In Table V the data matrix with one missing value is given. A new version of WOWA, modified to process missing values as described above, was executed with this data set and with the same parameters used in the tests in Section III B.

This resulted in a best aptitude of 0.001, with convergence in 6 generations. The weight vectors were as follows:

$$p = (0.17, 0.39, 0.34, 0.11)$$
$$w = (0.02, 0.41, 0.29, 0.29)$$

Table V. Data matrix H and solution vector d, with missing values indicated by "M."

0.7	0.6	0.4	M	|	0.50
0.9	0.74	0.5	0.3	|	0.60

If we increase the number of missing values to two, that is, in Table V we also make the value corresponding to row 2, column 4 equal to “M,” and rerun the WOWA modified to process missing values, the best aptitude is also 0.001, but convergence is slower in 26 generations. If we increase the number of missing values to four, that is, in Table V we also make row 2, column 4 and row 1 column 1 equal to “M,” and rerun the modified WOWA, the best aptitude is 0.009, achieved at the maximum (cutoff) of 100 generations. Finally, if we increase the number of missing values to six, that is, in Table V we also make row 1, column 2 and row 2, column 3 equal to “M,” and rerun the modified WOWA, we see a significant degradation in the performance: best aptitude is 0.300 achieved in 100 generations; the output values are 0.50 and 0.90 for rows 1 and 2, respectively; the weight vectors in this last case settle to the following:

$$p = (0.42, 0.04, 0.12, 0.42)$$
$$w = (0.05, 0.36, 0.41, 0.19)$$

In conclusion, we see that with the data set of Table V, there is a robust handling of missing values, in which there is only a significant degradation in the overall precision when there exists a high percentage of missing values. We have to take into account that the datum in Table V is a small artificial data set and it is relatively easy for the aggregation to find good alternative solutions for the different subsets of the values.

V. ADVANTAGES AND INCONVENIENCES OF THE TWO APPROACHES

Both active set methods and genetic algorithms have been applied to learn the weights for the aggregation operators described in Section II. We have used them in medium sized real problems (see Refs. 10, 28, and 29 for details) and from our experience the two approaches are complementary. We describe below the advantages and inconveniences of both approaches.

ASM based methods are appropriate for learning the weights for the WM and the OWA operators because, in this case, the minimization problem is a quadratic one and almost exact solutions can be found (except for cases of numerical instability, e.g., linear dependency between two information sources). For quadratic problems, the formulation of the problem and their resolution is simple. Instead, for these two cases, genetic algorithms are not quite appropriate because, usually, the best solution they find is only a suboptimal one. Thus, ASMs are more precise. Moreover, the computation costs in terms of memory and CPU usage are greater in the case of the GA based approach than in the case of the ASM. This is due to the fact that genetic algorithms need to compute the fitness function for each of the individuals in each of the populations and this is calculated applying the aggregation function to each example. Instead, the costs of the ASM are mainly proportional to the number of variables and not to the number of examples. This is so because ASM only uses the examples once—in the initial step—to compute an N^*N matrix (where N is, as above, the number of examples) and the iterative method uses this matrix but not the initial examples. According to this, the greatest difference between the costs is when the set of examples is large and the number of variables is small. In relation to the implementation, neither ASM based methods nor GA based ones are difficult to implement. To implement the former, we need the general ASM algorithm for selecting or removing active constraints (described in Ref. 10) and an algorithm to solve linear equations. The implementation of the latter has to follow the indications given in Section III B. According to the better approximation, the computational cost, and the implementation difficulties, it seems to us that when the aggregation operator is either WM or OWA then ASM based methods are recommended.

This is not the case when weights are learned for the WOWA operator. In this case, the complexity of ASMs increases because the function to minimize is not quadratic. This is due to the existence of the interpolation function w^* (built from the weighting vector w—one of the weighting vectors to learn) and due to the fact that this function is applied to additions of some of the p's (the other weighting vector to learn). Although there exist some optimization techniques to find approximate solutions for nonquadratic problems, their implementation is a difficult and nontrivial task (see, e.g., Ref. 14). In this case, complexity of genetic algorithms does not increase and the main change is to use the WOWA operator in the fitness function instead of using the OWA or WM. Thus, genetic algorithms can be used to obtain suboptimal solutions with sufficient precision more easily. Besides these advantages, the genetic algorithm is also suitable because it can find several suboptimal solutions. This is especially adequate because, at present, we do not know whether the distance for the WOWA operator is convex or not [this is not the case for the WM and OWA where $D(p)$ is convex].

The use of genetic algorithms presents an additional advantage for either the WM, OWA, or WOWA operators. This is the case when data files include missing values or the number of variables is different in each example. Section IV presents an alternative definition of the distance function that takes this fact into account. Using this function is easy when genetic algorithms are applied because we only need to adapt the fitness function and the rest of the program can be kept as it was. However, this is not the case for the ASM based approach, as, for example, in the case of the weighted mean or the OWA operator, the minimization problem is then nonquadratic. That is, the distance cannot be expressed as

$$xQx + dx$$

for an appropriate matrix Q and vector d. Thus, the general ASM method cannot be applied. Therefore, when the number of variables is not the same for all examples (as is sometimes the case in real applications due to missing or nonapplicable values), the genetic algorithm approach is also more appropriate.

VI. CONCLUSIONS

In this article we have detailed two different approaches to learning the weight values used by the WM, OWA, and WOWA aggregation operators. We have seen some advantages and disadvantages of each of the approaches, and results of data processing with simple data sets. The methods provide viable options when considering alternatives for learning weights, whose choice may finally depend on the nature of the data sets being processed, the observed results, and the desired precision and repeatability of the outcome.

Vicenç Torra acknowledges partial support of the CICYT project SMASH: TIC-96-1038-C04-04.

References

1. Aczél J. On weighted synthesis of judgements. Aequationes Math 1984;27:288–307.
2. Yager RR. On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Trans SMC 1988;18:183–190.
3. Torra V. The weighted OWA operator. Int J Intelligent Systems 1997;12:153–166.
4. Fodor J, Roubens M. (1994). Fuzzy preference modelling and multicriteria decision support. Dordrecht: Kluwer Academic; 1994.
5. Murofushi T, Sugeno M. Fuzzy t-conorm integral with respect to fuzzy measures: generalization of Sugeno integral and Choquet integral. Fuzzy Sets and Systems 1991;42:57–71.
6. Yager RR. Quantifier guided aggregation using OWA operators. Int J Int Systems 1996;11:49–73.
7. Saaty TL. The analytic hierarchy process. New York: McGraw-Hill; 1980.
8. O'Hagan M. Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic. In: Proc 22nd Annual IEEE Asilomar Conf Signals, Systems and Computers, Pacific Grove, CA, 1988, p 681–689.
9. Filev D, Yager RR. On the issue of obtaining OWA operator weights. Fuzzy Sets and Systems 1998;94:157–169.
10. Torra V. On the learning of weights in some aggregation operators: the weighted mean and OWA operators. Mathware and Soft Computing, 1999;6:249–265.
11. Marichal J-L, Roubens M. Determination of weights of interacting criteria from a reference set, Papiers de Recherche, Faculté d'Economie de Gestion et de Sciences Sociales, Groupe d'Etude des Mathématiques du Management et de l'Economie, N. 9909. 1999.
12. Fodor J, Marichal J-L, Roubens M. Characterization of the ordered weighted averaging operators. IEEE Trans Fuzzy Systems 1995;3:236–240.
13. Torra V, On some relationships between the WOWA operator and the Choquet integral. In: Proc Seventh Conf Inform Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'98) (ISBN 2-84254-013-1) Paris, France, p 818–824.
14. Luenberger DG. Introduction to linear and nonlinear programming. Menlo Park, CA: Addison-Wesley; 1973.
15. Gill PE, Murray W, Wright MH. Practical optimization. San Diego: Academic Press; 1981.
16. Rechenberg I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. Stuttgart: Frommann-Holzboog; 1973.
17. Schwefel HP. Numerical optimisation for computer models. Chichester, UK: Wiley; 1981.

18. Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution. Chichester, UK: John Wiley; 1966.
19. Glover F. Heuristics for integer programming using surrogate constraints. Decision Sciences 1997;8:156–166.
20. Vignaux GA, Michalewicz Z. A genetic algorithm for the linear transportation problem. IEEE Trans Systems Man Cybernet 1991;21:445–452.
21. Koza JR. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Report No. STAN-CS-90-1314, Stanford University, 1990.
22. Michalewicz Z, Janikow C. GENOCOP: A genetic algorithm for numerical optimization problems with linear constraints. Communications of the ACM 1996;39:175–201.
23. Antonisse HJ, Keller KS. Genetic operators for high level knowledge representation. In: Proc Second Internat Conf on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum Associates; 1987. p 69–76.
24. Forrest S. Implementing semantic networks structures using the classifier system. In: Proc First Internat Conf Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum Associates, 1985. p 24–44.
25. Fox BR, McMahon MB. Genetic operators for sequencing problems. In: Rawlins G, editor. First workshop on the foundations of genetic algorithms and classifier systems. San Mateo, CA: Morgan Kaufmann; 1991. p 284–300.
26. The travelling salesman problem. In: Lawler EL, Lenstra JK, Rinnooy AHG, Shmoys DB, editors. A guided tour of combinatorial optimization. New York: John Wiley & Sons; 1985.
27. Michalewicz Z. Genetic algorithms + data structures = evolution programs, 3rd Edition. Berlin: Springer; 1996.
28. Nettleton DF, Hernandez L. Evaluating reliability and relevance for WOWA aggregation of sleep apnea case data. In: Proc 1999 Eusflat-Estylf Joint Conf, Palma de Mallorca, Spain, 1999.
29. Nettleton DF, Muñoz J. Processing and representation of meta-data for sleep apnea diagnosis with an artificial intelligence approach. Internat J Medical Informatics—special issue, Elsevier, 2000. (Accepted for publication)
30. Bowen J, Dozier G. Solving constraint satisfaction problems using a genetic systematic search hybrid that realizes when to quit. In: Proc Sixth Internat Conf Genetic Algorithms. San Mateo, CA: Morgan Kaufmann; 1995. p 122–129.
31. Cartright HM, Mott GF. Looking around: Using clues from the data space to guide genetic algorithm searches. In: Proc Fourth Internat Conf Genetic Algorithms. San Mateo, CA: Morgan Kaufmann; 1991. p 108–114.