Comparing Apples and Oranges: The Weighted OWA Function

Gleb Beliakov*
School of Information Technology, Deakin University, Burwood, Australia

This paper advocates the use of weighted ordered weighted averaging (WOWA) functions in decision-making processes, where the alternatives are not directly comparable. In particular, WOWA allows one to compare the strongest points of each alternative, also weighted by the importance of each criterion. Four different approaches to applying weights in OWA functions are reviewed. Torra's method based on interpolating regular increasing monotone quantifier and the pruned n-ary tree are compared to the WOWA obtained from recently proposed implicit averaging. Computationally efficient algorithms are outlined. The use of WOWA is illustrated in several examples. © 2017 Wiley Periodicals, Inc.

1. INTRODUCTION

Aggregation of degrees of satisfaction is one of the fundamental problems in decision making. Aggregation functions[1–3] combine the individual inputs into an overall degree of satisfaction, which is then used for ranking the alternatives among other purposes. In their most general form, aggregation functions are monotone increasing functions $f : [0, 1]^n \to [0, 1]$ with the boundary conditions $f(0, \ldots, 0) = 0$ and $f(1, \ldots, 1) = 1$, so they ensure matching the classical logical operations in the limiting cases.

When there is a clear prioritization of the criteria, given, for example, by importance weights, standard weighted functions such as arithmetic and quasi-arithmetic means (QAMs) are the method of choice. However, decision makers often face a situation where the priorities are not that clear. Consider a typical situation in an academic promotions committee. The standard criteria for promotion are the achievements in research, teaching and service, and satisfaction of two out of three criteria to a high standard warrant a promotion. Here we are not willing to prioritize teaching over research and vice versa, which would discourage a particular category of applicants, and at the same time we would like to promote high achievers in one or two areas as opposed to all-round yet mediocre candidates. In other words, we do not want a low score in one area (or even absence of a score in case of, say, research-only positions) to pull the overall score down.

*Author to whom all correspondence should be addressed; e-mail: gleb@deakin.edu.au.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 33, 1089–1108 (2018)

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.21913

Situations like this are numerous in decision making, where the alternatives may not satisfy the whole set of decision criteria but only a subset of these. How can we compare seemingly incomparable alternatives, such as apples and oranges. Each has its own strengths and weaknesses, and we aim at comparing the strongest points of each alternative with the others. The strongest points are not known at the design stage, and hence the procedure cannot be modeled by the standard weighted means.

The concept of the ordered weighted averaging (OWA) introduced by Yager[4] addresses this particular point. In OWA, the weights are assigned not to particular criteria but are applied according to the strength of satisfaction, that is, the input's position in the ordered list of inputs. OWA functions allow one to model decision-making processes in which one, two or, k out of n criteria are satisfied, discard the strongest or the weakest inputs, or to the contrary, focus only on the strongest and weakest inputs, model linguistic requirements such as the "majority," "almost all," "at least half" of the inputs, and so on.[5–10]

In the context of our example, decision-making problem, two out of three equally weighted criteria can be modeled by an OWA function with the weighting vector $\mathbf{w} = (\frac{1}{2}, \frac{1}{2}, 0)$, which equally weights the two strongest inputs and discards the third, although the latter input can also be assigned a small weight to ensure it is counted.

Next we would like to weight the decision criteria unequally. Suppose that a faculty needs to improve its teaching performance, and hence weight the teaching criterion higher than research and service. On the other hand, service, although important, is usually dealt through workload allocation and to a lesser degree through promotion, and hence both teaching and research criteria are prioritied over service. It brings us to the introduction of weights into the OWA function, the weighted OWA (WOWA). The WOWA function was considered by Torra[11,12] where the weights were introduced through an auxiliary interpolation function. It was later shown that such a WOWA function is a discrete Choquet integral with respect to a distorted probability measure.[13,14] It allows one to operate with two weighting vectors, one related to the inputs magnitude and another related to the inputs themselves. Another work in this direction was done by Yager.[5,15,16] In Ref. 15, a transformation function that combines the inputs of the OWA with their importance was built using fuzzy modeling techniques from partial knowledge about that function in the prototypical examples.

In this article, we intend to summarize and compare various techniques used to construct WOWA functions, including some very recent ones.[17,18] While serving the same purpose, the WOWA functions constructed by different methods exhibit different numerical behavior and may or may not be suitable for specific tasks. We illustrate the use of WOWA on several examples and point to an implementation of the mentioned WOWA functions.

The article is structured as follows. Section 2 presents the preliminary notions. In Section 3, we present the method of WOWA construction by Torra,[11] which is based on computing the modified OWA weights by solving an interpolation problem. Section 4 presents an alternative method that is based on interpolation of a regular increasing monotone (RIM) quantifier and repetition of inputs a suitable number

of times following the construction in Calvo et al.[19] Section 5 presents a different approach recently discussed in Refs. 20,21. This is a general approach to introducing weights into symmetric bivariate functions that was later extended to the n-variate symmetric functions, and specifically OWA, in Ref. 17. It is based on an n-ary tree construction and recursive application of the base OWA function. In Section 6, we present a new method of introducing weights in symmetric functions such as OWA from Ref. 18. As opposed to other WOWA methods, this technique does not result in a function that is a discrete Choquet integral, yet it satisfies the remaining properties sought in WOWA functions. Section 7 presents an illustrative case study and compares the behavior of the WOWA functions considered. The conclusions are presented in Section 8.

2. PRELIMINARIES

Consider now the following definitions adopted from Refs. 1,3. Let $\mathbb{I} = [0, 1]$, although other intervals can be accommodated easily.

DEFINITION 1. *A function* $f : \mathbb{I}^n \to \mathbb{R}$ *is monotone (increasing) if* $\forall\, \mathbf{x}, \mathbf{y} \in \mathbb{I}^n$, $\mathbf{x} \leq \mathbf{y}$ *then* $f(\mathbf{x}) \leq f(\mathbf{y})$*, with the vector inequality understood componentwise.*

DEFINITION 2. *A function* $f : \mathbb{I}^n \to \mathbb{I}$ *is idempotent if for every input* $\mathbf{x} = (t, t, \ldots, t)$*,* $t \in \mathbb{I}$*, the output is* $f(\mathbf{x}) = t$.

DEFINITION 3. *A function* $f : \mathbb{I}^n \to \mathbb{I}$ *is a mean (or is averaging) if for every* $\mathbf{x}$ *it is bounded by* $\min(\mathbf{x}) \leq f(\mathbf{x}) \leq \max(\mathbf{x})$.

Averaging functions are idempotent, and monotone increasing idempotent functions are averaging. We consider weighting vectors $\mathbf{w}$ such that $w_i \geq 0$ and $\sum w_i = 1$ of appropriate dimensions.

DEFINITION 4. *A function* $f : \mathbb{I}^n \to \mathbb{I}$ *is shift-invariant (stable for translations) if* $f(\mathbf{x} + a\mathbf{1}) = f(\mathbf{x}) + a$ *whenever* $\mathbf{x}, \mathbf{x} + a\mathbf{1} \in \mathbb{I}^n$*. A function* $f : \mathbb{I}^n \to \mathbb{I}$ *is homogeneous (of degree 1) if* $f(a\mathbf{x}) = af(\mathbf{x})$ *whenever* $\mathbf{x}, a\mathbf{x} \in \mathbb{I}^n$.

DEFINITION 5. *For a given generating function* $g : \mathbb{I} \to [-\infty, \infty]$*, and a weighting vector* $\mathbf{w}$*, the weighted quasi-arithmetic mean (QAM) is the function*

$$M_{\mathbf{w},g}(\mathbf{x}) = g^{-1}\left(\sum_{i=1}^{n} w_i g(x_i)\right). \tag{1}$$

In case $g = Id$, we have the weighted arithmetic mean WAM.

DEFINITION 6. *Let* $\varphi : \mathbb{I} \to \mathbb{I}$ *be a bijection. The* φ*-transform of a function* $f : \mathbb{I}^n \to \mathbb{I}$ *is the function* $f_\varphi(\mathbf{x}) = \varphi^{-1}(f(\varphi(x_1), \varphi(x_2), \ldots, \varphi(x_n)))$.

The weighted QAM is a φ-transform of the WAM with $\varphi = g$.

DEFINITION 7. *For a given weighting vector* $\mathbf{w}$, $w_i \geq 0$, $\sum w_i = 1$, *the OWA function is given by*

$$OWA_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{(i)}, \tag{2}$$

where $x_{(i)}$ *denotes the* i*th largest value of* $\mathbf{x}$.

The main properties of OWA are summarized in the following:

- OWA are increasing functions (strictly increasing if all the weights are positive) and idempotent;
- OWA functions are continuous, symmetric, homogeneous and shift-invariant;
- OWA functions are piecewise linear, and the linear pieces are joined together where two or more arguments are equal in value;
- OWA functions are special cases of the Choquet integral with respect to symmetric fuzzy measures;
- the special cases of OWA include the arithmetic mean, the median, and the minimum and maximum operators among others;
- the dual of an OWA with respect to the standard negation is the OWA with the weights reversed.

The orness measure allows one to qualify an OWA function as OR-like or AND-like based on whether it behaves more disjunctively or more conjunctively than the arithmetic mean. The expression for the orness measure is given by the following simple formula

$$orness(OWA_{\mathbf{w}}) = \sum_{i=1}^{n} w_i \frac{n-i}{n-1} = OWA_{\mathbf{w}}\left(1, \frac{n-2}{n-1}, \ldots, \frac{1}{n-1}, 0\right). \tag{3}$$

The OWA functions are OR-like if $orness(OWA_{\mathbf{w}}) \geq \frac{1}{2}$ and AND-like if $orness(OWA_{\mathbf{w}}) \leq \frac{1}{2}$. If the weighting vector is decreasing, that is, $w_i \geq w_j$ whenever $i < j$, OWA is OR-like and is in fact a convex function. The respective (symmetric) fuzzy measure in this case is submodular.[22] The OWA functions with increasing weights are AND-like, concave functions that correspond to the Choquet integral with respect to a super-modular fuzzy measure. OWA with decreasing weighting vectors can be used to define norms.[22,23]

DEFINITION 8.[8,24] *Regular increasing monotone (RIM) quantifier (also referred to as basic unit-interval monotone (BUM) function) is a monotone increasing continuous function* $Q : [0, 1] \to [0, 1]$, $Q(0) = 0$, $Q(1) = 1$ *that generates OWA weights for any* $n > 1$ *using*

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right). \tag{4}$$

The weights in weighted means and in OWA functions represent different things. In weighted means, w_i reflects the importance of the ith input, whereas in OWA, w_i reflects the importance of the ith largest input. In Ref. 11, Torra proposed a generalization of both weighted means and OWA, called WOWA. This aggregation function has two sets of weights $\mathbf{w}$, $\mathbf{p}$. Vector $\mathbf{p}$ plays the same role as the weighting vector in weighted means, and $\mathbf{w}$ plays the role of the weighting vector in OWA functions.

Consider the following motivation. A robot needs to combine information coming from n different sensors, which provide distances to the obstacles. The reliability of the sensors is known (i.e., we have weights $\mathbf{p}$). However, independent of their reliability, the distances to the nearest obstacles are more important, so irrespective of the reliability of each sensor, their inputs are also weighted according to their numerical value, hence we have another weighting vector $\mathbf{w}$. Thus, both factors, the size of the inputs and the reliability of the inputs, need to be taken into account. WOWA provides exactly this type of aggregation function.

Another illustration comes from academic selection problem.

Example 1. An academic department wishes to hire faculty that can contribute significantly to research, teaching, industry engagement and service and leadership. The department advertised for a junior to mid-level position, and therefore does not anticipate the candidates with significant past roles in service and leadership. The research is given marginal priority over teaching, and industry engagement is also welcome. The selection committee decides to assess the applicants on two out of four strongest points. The weightings given to research, teaching, industry engagement and service will be denoted by $\mathbf{p} = (p_r, p_t, p_i, p_s) = (0.3, 0.25, 0.3, 0.15)$. The OWA weighing vector is given by $\mathbf{w} = (0.4, 0.35, 0.2, 0.05)$, so that the two strongest inputs are the most important while the weakest input is only of marginal importance.

The following sections present four alternative approaches to combining the two weighting vectors in a WOWA function.

3. WOWA APPROACH BY TORRA

In this section, we follow the approach by Torra in Ref. 11.

DEFINITION 9. *Let* $\mathbf{w}, \mathbf{p}$ *be two weighting vectors,* $w_i, p_i \geq 0$, $\sum w_i = \sum p_i = 1$. *The following function is called WOWA function*

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = \sum_{i=1}^{n} u_i x_{(i)},$$

where $x_{(i)}$ is the ith largest component of $\mathbf{x}$*, and the weights u_i are defined as*

$$u_i = g\left(\sum_{j\in H_i} p_j\right) - g\left(\sum_{j\in H_{i-1}} p_j\right),$$

where the set $H_i = \{j | x_j \geq x_i\}$ is the set of indices of i largest elements of $\mathbf{x}$ *and g is a monotone nondecreasing function with two properties:*

1. *$g(i/n) = \sum_{j\leq i} w_j$, $i = 0, \ldots, n$ (of course $g(0) = 0$);*
2. *g is linear if all w_i are equal.*

WOWA function becomes the weighted arithmetic mean if $w_i = \frac{1}{n}$, $i = 1, \ldots, n$, and becomes the usual OWA if $p_i = \frac{1}{n}$, $i = 1, \ldots, n$.

The computation of WOWA involves a very similar procedure to that of OWA (i.e., sorting the components of $\mathbf{x}$ and then computing their weighted sum), but the weights u_i are defined by using both vectors $\mathbf{w}$, $\mathbf{p}$, a special monotone function g, and depend on the components of $\mathbf{x}$ as well. One can see WOWA as an OWA function with the alternative weights $\mathbf{u}$.

The generating function g plays a similar role to the RIM quantifier and is computed from the given weights w_i and an additional condition 2, which ensures that $u_i = p_i$ whenever all $w_i = \frac{1}{n}$. Of course, the weights $\mathbf{u}$ also depend on the generating function g. This function can be chosen as a linear spline (i.e., a broken line interpolant), interpolating the points $(i/n, \sum_{j\leq i} w_j)$ (in which case it automatically becomes a linear function if these points are on a straight line), or as a monotone quadratic spline, as was suggested in Ref. 11,12. We suggested in Ref. 25 to use an alternative monotone quadratic spline algorithm due to Schumaker, which has a benefit of automatic satisfaction of the straight line condition when needed. This method allows an efficient parallel implementation for Graphics processing units.[26]

It turns out that WOWA belongs to a more general class of Choquet integral based aggregation functions.[14] It is a piecewise linear function whose linear segments are defined on the simplicial partition of the unit cube $[0, 1]^n$: $\mathcal{S}_i = \{\mathbf{x} \in [0, 1]^n | x_{p(j)} \geq x_{p(j+1)}\}$, where p is a permutation of the set $\{1, \ldots, n\}$. Note that there are exactly $n!$ possible permutations, the union of all $\mathcal{S}_i$ is $[0, 1]^n$, and the intersection of their interiors is empty: $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $i \neq j$.

4. INTERPOLATION OF THE RIM QUANTIFIER FUNCTION

Let us now consider an alternative approach based on interpolating a RIM quantifier function. Here we use a method from,[19] in which the weights of a function are computed by repeating the inputs a suitable number of times. Consider an auxiliary vector of arguments $\mathbf{X} = (x_1, \ldots, x_1, x_2, \ldots, x_2, \ldots, x_n, \ldots, x_n)$, so that x_1 is taken k_1 times and x_2 is taken k_2 times, and so on, so that $\frac{k_1}{M} = p_1$, $\frac{k_2}{M} = p_2, \ldots$, and $k_1 + k_2 + \cdots + k_n = M$. We assume the weights p_i are rational numbers, which

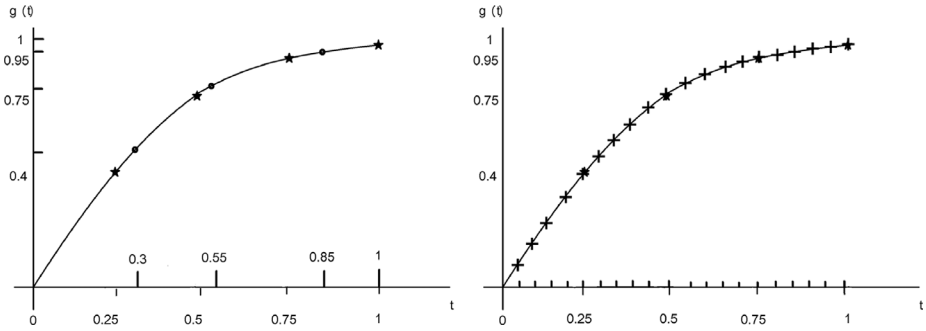

Figure 1. Quantifier function g interpolated from the OWA weights **w** (represented by stars). The weights u_i for Torra's method are found from the points marked with circles (left), and the weights for the vector X ($M = 20$ in this example) are found from the points marked with crosses (right). The abscissae of those points are marked above the horizontal axis.

is not a strong restriction if we look at a computer implementation of the method in finite precision arithmetics.

The approach from Ref. 19 consists of using the auxiliary vector **X** in a strongly idempotent symmetric function (such as OWA induced by a quantifier) whose output will be a weighted function of the inputs **x**.

In the case of OWA, in order to apply it to a larger dimensional auxiliary input vector **X** we need to produce the weighting vector **w** of the corresponding dimension M, denoted here by **u**. We apply a similar approach to Torra's construction, that is, we construct a generating function g by interpolating the data $g(i/n) = \sum_{j \le i} w_j, i = 0, \ldots, n$ and $g(0) = 0$, and the straight line condition, that is, using the two conditions in Definition 9. The latter is necessary to obtain the standard weighted mean in case all $w_i = \frac{1}{n}$.

Hence, we can use a piecewise linear of piecewise quadratic interpolation as in Refs. 11,12,25 to construct the RIM quantifier g. Now, in difference to Torra's approach, we calculate the WOWA as

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = OWA_{\mathbf{u}}(\mathbf{X}) = \sum_{i=1}^{n} u_i X_{(i)},$$

where the weights u_i are defined as

$$u_i = g\left(\frac{i}{M}\right) - g\left(\frac{i-1}{M}\right), \quad i = 1, \ldots, M.$$

Figure 1 illustrates the two approaches.

If we compare both methods based on the generating RIM quantifier function, we can see that both produce exactly the same function WOWA. Indeed, since the vector **X** has groups of repeated arguments, we can break the expression for WOWA

into individual sums

$$OWA_{\mathbf{u}}(\mathbf{X}) = \sum_{i=1}^{n} u_i X_{(i)} = \sum_{j=1}^{n} \sum_{i=1}^{k_j} x_j \left(g\left(\frac{i + k_{j-1}}{M}\right) - g\left(\frac{i + k_{j-1} - 1}{M}\right)\right),$$

with $k_0 = 0$. The inner sums collapse and we have

$$OWA_{\mathbf{u}}(\mathbf{X}) = \sum_{j=1}^{n} x_j \left(g\left(\frac{k_j + k_{j-1}}{M}\right) - g\left(\frac{k_{j-1}}{M}\right)\right)$$

with $g(0) = 0$ as usual. From the definition of $k_j = Mp_j$, we conclude that the above expression matches the one in the Definition 9. We conclude that Torra's formula for the weights of WOWA can be seen as an instance of the approach from Ref. 19 based on replicating the inputs, although it obviously predates the work.[19]

Section 5 introduces an alternative and generic construction to incorporate weights into any symmetric averaging function. In particular, it will work for OWA and will not have a somewhat unclear issue of selecting the function g in Torra's WOWA.

5. *n*-ARY TREE CONSTRUCTION FOR OWA BY DUJMOVIC AND BELIAKOV

Consider first a method of incorporating weights into any symmetric *bivariate* idempotent function f, presented in Refs. 20,21. To introduce the weights we use the same approach from Ref. 19, where each argument x_i is replicated a suitable number of times. We consider an auxiliary vector of arguments $\mathbf{X} = (x_1, \ldots, x_1, x_2, \ldots, x_2)$, so that x_1 is taken k_1 times and x_2 is taken k_2 times, so that $\frac{k_1}{2^L} \approx p_1$, $\frac{k_2}{2^L} \approx p_2$, and $k_1 + k_2 = M = 2^L$. Here M is a power of two and $L \geq 1$ is a specified number of levels of the binary tree shown in Figure 2. One way of doing so is to take $k_1 = \lfloor p_1 2^L + \frac{1}{2} \rfloor$ and $k_2 = 2^L - k_1$. The vector $\mathbf{X}$ needs to be sorted in the increasing or decreasing order.

Next, let us build a binary tree presented in Figure 2, where at each node a value is produced by aggregating the values of two children nodes with the given bivariate symmetric averaging function f (denoted by B on the plot and with weights equal to $\frac{1}{2}$). We start from the leaves of the tree that contain the elements of the vector $\mathbf{X}$. In this example, we took $p_1 = \frac{5}{8}$ and $p_2 = \frac{3}{8}$. The value y at the root node will be the desired output of the n-variate weighted function.

A straightforward binary tree traversal algorithm for doing so, which starts from the vector $\mathbf{X}$, is presented in Refs. 20,21 under the name ABL, but its runtime is $O(2^L)$, which can make its use prohibitive even for moderate L. Fortunately an efficient algorithm based on pruning the binary tree was also presented in Ref. 21.

The pruning of the binary tree is done by using the idempotency of f (see Figure 2, right). Indeed, no invocation of f is necessary if both of its arguments are equal.

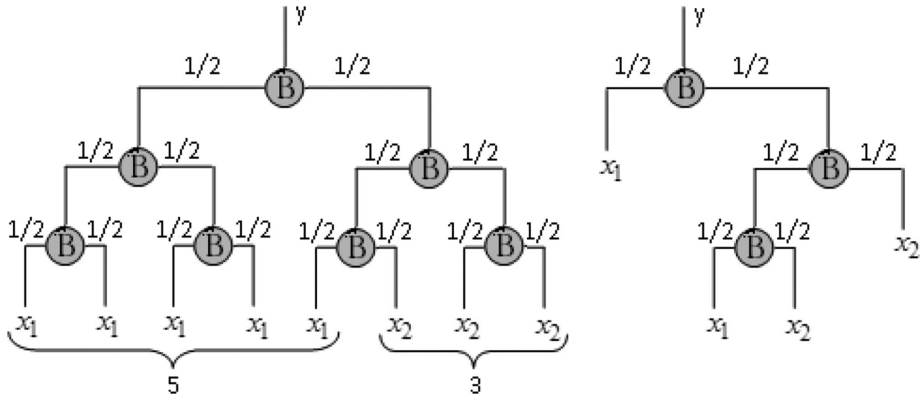

Figure 2. Representation of a weighted arithmetic mean in a binary tree construction. The tree on the right is pruned by using idempotency.

Below we present the pruned tree algorithm (PTA) whose worst case complexity is $O(L)$, which makes it practically applicable for larger L. The algorithm is recursive depth-first traversing of the binary tree. A branch is pruned if it is clear that all its leaves have exactly the same value, and by idempotency this is the value of the root node of that branch.

PTA algorithm
function $node(m, N, K, x)$

(1) If $N[K] \geq 2^m$ then do:
  (a) $N[K] := N[K] - 2^m$;
  (b) $y := x[K]$;
  (c) If $N[K] = 0$ then $K := K + 1$;
  (d) return y;
  else
(2) return $f(node(m-1, N, K, x), node(m-1, N, K, x))$.
  function $f\_n(p, x, L)$
  1. create the array $N := (k_1, k_2)$ by using $k_1 := \lfloor p_1 2^L + \frac{1}{2} \rfloor$, and $k_2 := 2^L - k_1$;
  2. $K := 1$;
  3. return $node(L, N, K, x)$.

In this algorithm, the array N serves as a counter of how many copies of each of $x[K]$ remains. If there are more than 2^m copies, they belong to a branch that can be pruned, so the function $node$ just returns $x[K]$ and never visits the nodes of that branch. If $N[K] = 1$, then the last remaining copy of $x[K]$ is returned and the value of K is incremented. Every time a branch whose leaves contain identical arguments is encountered (which is detected by the counter $N[K] \geq 2^m$), this branch is pruned.

To see the complexity of this algorithm note that f is never executed (nor the corresponding node of the tree is visited) if its arguments are the same. There is exactly one node at each level of the tree where the child nodes contain distinct arguments, hence f is executed exactly L times. Also note that both N and K are input–output parameters, so that the two arguments of f at step 2 are different as

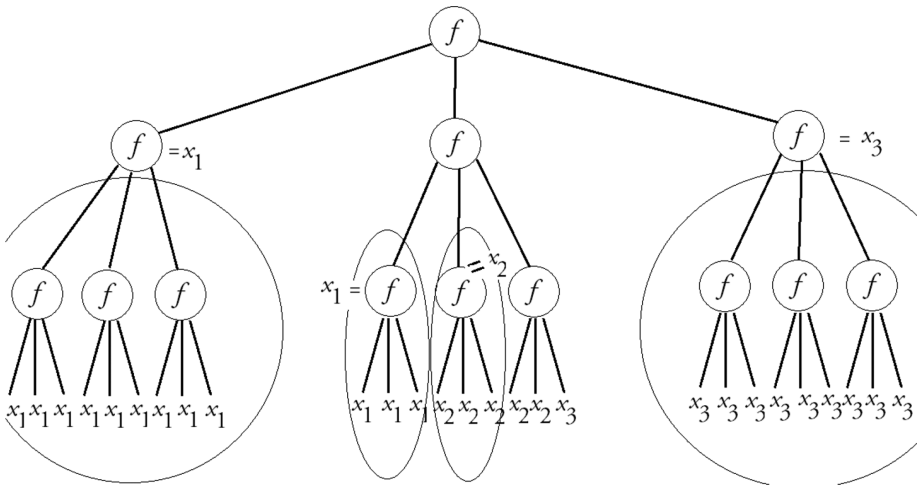

Figure 3. Representation of a weighted tri-variate function f in a ternary tree construction. The weights are chosen as $\mathbf{p} = (\frac{12}{27}, \frac{5}{27}, \frac{10}{27})$ and $L = 3$. The circled branches are pruned by the algorithm.

N and K change from one invocation of the function *node* to another, however the order of execution of the calls to *node* does not matter as the lists of formal parameters are identical.

Several useful properties of the binary tree construction were presented in Ref. 21. In particular, the weighted function f_p inherits many properties of the base aggregator f, such as idempotency, monotonicity, continuity, convexity (concavity), homogeneity and shift-invariance, due to preservation of these properties in function composition. Furthermore, when the weights are given in a finite binary representation (as is always the case in machine arithmetic), the sequence of the outputs of the PTA algorithm with increasing $L = 2, 3, \ldots$ converges to a weighted mean with the specified weights, and in fact L needs not exceed the number of bits in the mantissa of the weights p_i to match these weights exactly. Finally, when f is a QAM, f_p is a weighted QAM with the same generator.

The big advantage of the binary tree construction is its universality and transparency. It is applicable to any bivariate idempotent function f without modification, and the role of the weights as the respective multiplicities of the arguments as argued in Ref. 19 is very clear. The availability of a fast and uncomplicated algorithm for computing the output makes this method immediately applicable.

Once we established the main ideas behind the binary tree construction, we can extend the algorithm to n-variate OWA function following Ref. 17. Our goal here is to incorporate a vector $\mathbf{p}$ of nonnegative weights (which add to one) into a symmetric n-variate function by replicating the arguments a suitable number of times. As in the binary tree construction, we build an n-ary tree with L levels, as shown in Figure 3. As the base symmetric aggregator f we take an OWA function $OWA_{\mathbf{w}}$ with specified weights $\mathbf{w}$ (although the origins of f are not important for the algorithm).

Let us create an auxiliary vector $\mathbf{X} = (x_1, \ldots, x_1, x_2, \ldots, x_2, \ldots, x_n, \ldots, x_n)$, so that x_1 is taken k_1 times, x_2 is taken k_2 times, and so on, and $\frac{k_1}{n^L} \approx p_1$, $\frac{k_2}{n^L} \approx p_2$, ..., and $\sum k_i = n^L$, where $L \geq 1$ is a specified number of levels of the tree shown in Figure 3. One way of doing so is to take $k_i = \lfloor p_i n^L + \frac{1}{n} \rfloor$, $i = 1, \ldots, n-1$ and $k_n = n^L - k_1 - k_2 - \cdots - k_{n-1}$.

The pruned n-tree aggregation (PnTA) algorithm works in the same way as the PTA algorithm for binary trees. The vector of counters N helps determine whether there are more than n^m identical elements of the auxiliary array X, in which case they are the leaves of a branch of the tree with m levels. This branch is pruned. The function f is executed only when some of its arguments are distinct, and since the elements of X are ordered, there are at most $n-1$ such possibilities at each level of the tree, hence the complexity of the algorithm is $O((n-1)L)$.

Note that the complexity is linear in terms of L, as that of the PTA algorithm, which means that the dimension of the base aggregator f does not matter in this respect. Of course, nominally the n-ary tree is larger than the binary tree, but since we only track the multiplicities of the arguments, never creating the array $\mathbf{X}$ explicitly, memorywise the complexity of the PnTA algorithm is the same as that of PTA.

PnTA algorithm

function $node(n, m, N, K, x)$

(1) If $N[K] \geq n^m$ then do:
  (a) $N[K] := N[K] - n^m$;
  (b) $y := x[K]$;
  (c) If $N[K] = 0$ then $K := K + 1$;
  (d) return y;
  else
(2) for $i := 1, \ldots, n$ do $z[i] := node(n, m-1, N, K, x)$;
(3) return $f(\mathbf{z})$.

function $f\_n(n, x, p, L)$
1. create the array $N := (k_1, k_2, \ldots, k_n)$ by using $k_i := \lfloor p_i n^L + \frac{1}{n} \rfloor$, $i = 1, \ldots, n-1$, and $k_n := n^L - k_1 - \cdots - k_{n-1}$;
2. $K := 1$;
3. return $node(n, L, N, K, x)$.

The vector $\mathbf{X}$ needs to be sorted, which is equivalent to sorting the inputs $\mathbf{x}$ jointly with the multiplicities of the inputs N (i.e., using the components of $\mathbf{x}$ as the key), so the complexity of the sort operation is the same $O(n \log n)$ as for OWA functions.

Let us list some useful properties of the function f_p generated by the PnTA algorithm established in Ref.17

THEOREM 1 (The Inheritance Theorem). *The weighted extension f_p of a function f by the PnTA algorithm preserves the intrinsic properties of the parent function f as follows:*

(1) f_p idempotent since f is idempotent;
(2) if f is monotone increasing, then f_p is monotone increasing;

(3) if f is continuous, then f_p is continuous;
(4) if f is convex (respectively concave), then f_p is convex (respectively concave);
(5) if f is homogeneous, then f_p is homogeneous;
(6) if f is shift-invariant, then f_p is shift-invariant;
(7) f_p has the same absorbing element as f (if any);
(8) if f generates f_p, then a φ-transform of f generates the corresponding φ-transform of f_p.

Proof. The proof easily follows from the properties of composition of the respective functions and idempotency of f. For the φ-transform notice that at each inner level of the tree the composition $\varphi^{-1} \circ \varphi = Id$, while φ is applied to the leaves of the tree and φ^{-1} is applied to the root. □

The next results are applicable when an OWA function is taken as the base aggregator f. For proofs, see Ref. 17.

THEOREM 2. *Let $f = OWA_w$. Then, the algorithm PnTA generates the weighted function f_p, which is the discrete Choquet integral (and is hence homogeneous and shift-invariant).*

As the special cases of Choquet integral, we have the following.

THEOREM 3. *Let $f = OWA_w$. Then, the algorithm PnTA generates the weighted function f_p with the following properties:*

(1) for the weights $w_i = \frac{1}{n}$ f_p is the weighted arithmetic mean with the weights $\mathbf{p}$*;*
(2) for the weights $p_i = \frac{1}{n}$ f_p is OWA_w;
(3) when $f = OWA_w = \min$ (or $= \max$), and $p_i > 0$ for all i, f_p is also min *(respectively,* max*);*
(4) when $f = OWA_w = median$ and n is odd, f_p is the weighted median;
(5) if OWA_w generates f_p, then the dual OWA_w^d generates the dual f_p^d, and in particular an OWA with the reverse weights generates the respective WOWA with the reverse weights.

THEOREM 4. *Let $f = OWA_w$ and let the weighting vector be decreasing (increasing). Then, the algorithm PnTA generates a Choquet integral with respect to a submodular (supermodular) fuzzy measure.*

This last result is useful when constructing weighted norms from OWA with decreasing weights (see Refs. 22,23).

On the technical side, we note that we do not need to sort the arguments in each OWA function in the n-ary tree, as the vector $\mathbf{x}$ is already sorted, hence only one sort operation for the inputs is required. Another note is that when the weights $\mathbf{p}$ are specified to m digits in base n, $L = m$ levels of the n-ary tree is sufficient to match these weighs exactly. For example if $\mathbf{p}$ are specified to 3 decimal places and $n = 10$, we only need to take $L = 3$. Therefore to match the weights to machine precision (e.g., 53 bits for data type `double`) n^L need not exceed the largest 64-bit integer, and hence the algorithm PnTA can be implemented with 64-bit data types. The source code in C++ is presented in Figure 4 and the software can be downloaded from Ref. 27.

Finally, we can introduce weights into generalized OWA functions in the same way as for OWA functions by using the n-ary tree construction. This can be done

```
 double OWA(int n, double x[],double w[])
 { /* no sorting is needed when used in the tree */
    double z=0;
    for(int i=0;i<n;i++) z+=x[i]*w[i];
    return z;
 }
 double node(int n, double x[], long int N[], long int C, int & k,
     double w[], double(*F)(int, double [],double[]), double* z)
 {
   /* recursive function in the n-ary tree processing
   Parameters: x - input vector, N vector of multiplicities of  x
   m current level of recursion counted from L (root node)  to 0
   k - input-output parameter, the index of x being processed */
   if(N[k]==0) k++;
   if(N[k]>= C) {  /* we use idempotency to prune the tree */
       N[k] -= C;
       if(N[k]<=0) return x[k++]; else return x[k];
   }
    C /= n;
   /* tree not pruned, process the children nodes */
   for(int i=0;i<n;i++) z[i]=node(n,x,N,C,k,w,F,z+n);
   return F(n,z,w);
}

double weightedf(double x[], double p[], double w[], int n,
         double(*F)(int, double[],double[]), int L)
/*
 Function F is the symmetric base aggregator.
 p[] = array of weights of inputs x[],
 w[] = array of weights for OWA, n = the dimension of x, p, w.
 the weights must add to one and be non-negative.
 L = number of binary tree levels. Run time = O[(n-1)L]  */
{
   long int t=0, C=1;
   int k=0;
   for(int i=0;i<L;i++) C*=n;  /* C=n^L */
   sortpairs(x, x+n, p);
   long int N[n]; /* multiplicities of x based on the weights */
   for(int i=0;i<n-1;i++)  { N[i]=p[i]*C+1./n;   t+=N[i]; }
   N[n-1]=C-t;
   double z[n*L]; /* working memory */
   return node(n,x,N,C,k,w,F,z);
}
/* example: calling the function */
int n=4, L=4;
double x[4]={0.2,0.2,0.4,0.8};
double w[4]={0.1,0.2,0.3,0.4};
double p[4]={0.3,0.2,0.1,0.4};
double y=weightedf(x,p,w,n,&OWA,L);
```

Figure 4. A C++ implementation of the pruned n-ary tree algorithm PnTA. The function `sortpairs` (not shown) implements sorting of an array of pairs (x_i, p_i) in the order of decreasing x_i.

by using a φ-transform of a WOWA with $\varphi = g$, that is, by applying g and g^{-1} only to the leaves and to the root node of the tree, relying on the preservation of φ-transforms. This method is computationally efficient as functions g and g^{-1} need not be used in obtain the special cases of weighted geometric and ordered weighted power-based generalized OWA functions.

6. WOWA BASED ON IMPLICIT AVERAGING

A different approach to introducing weights into averaging function was recently presented in Ref. 18 under the name of implicit averaging. Here, following an analogy with weighted arithmetic means, which can be written in this form

$$y \cdot \frac{\sum_{j=1}^{n} p_j}{n} = \frac{\sum_{i=1}^{n} p_i x_i}{n}, \tag{5}$$

with $y = WAM(x_1, \ldots, x_n)$, we use the following equation to compute the values of a weighted function f_p from a symmetric mean M,

$$C(M(p_1, \ldots, p_n), f_p(\mathbf{x})) = M(C(p_1, x_1), \ldots, C(p_n, x_n)). \tag{6}$$

Here M is an mean and C is a suitable bivariate operation such as a t-norm. The motivation behind the study in Ref. 18 is to produce alternative ways of incorporating weights $\mathbf{p}$ by replacing the product with another suitable operation and replacing the arithmetic mean with an arbitrary mean M. The function f_p is given implicitly through solution to the algebraic equation (6). This equation can be written in a compact form as

$$C(\bar{p}, \bar{x}_p) = \overline{C(p_i, x_i)},$$

where $\bar{p}$ denotes the (unweighted) average weight, $\overline{C(p_i, x_i)}$ denotes the average value of C, and $y = \bar{f}_p(\mathbf{x}) = \bar{x}_p$ is the weighted average of x_i.

The work[18] established a number of useful theoretical properties of the implicit averages, which also apply to the case of $M = OWA$. Instantiating equation (6) with C being the product and M being an OWA function with weights $\mathbf{w}$, we can resolve it explicitly and obtain

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = \frac{OWA_w(p_1x_1, \ldots, p_nx_n)}{OWA_w(p_1, \ldots, p_n)} = \frac{OWA_w(\mathbf{px})}{OWA_w(\mathbf{p})}. \tag{7}$$

Note that the weights p_i are otherwise unrestricted (i.e., they need not add to one) as the denominator in (7) will produce the required normalizing factor to ensure idempotency.

Special care should be taken when the denominator vanishes, as this WOWA may be discontinuous or not well defined if we allow 0 weights. For strictly positive

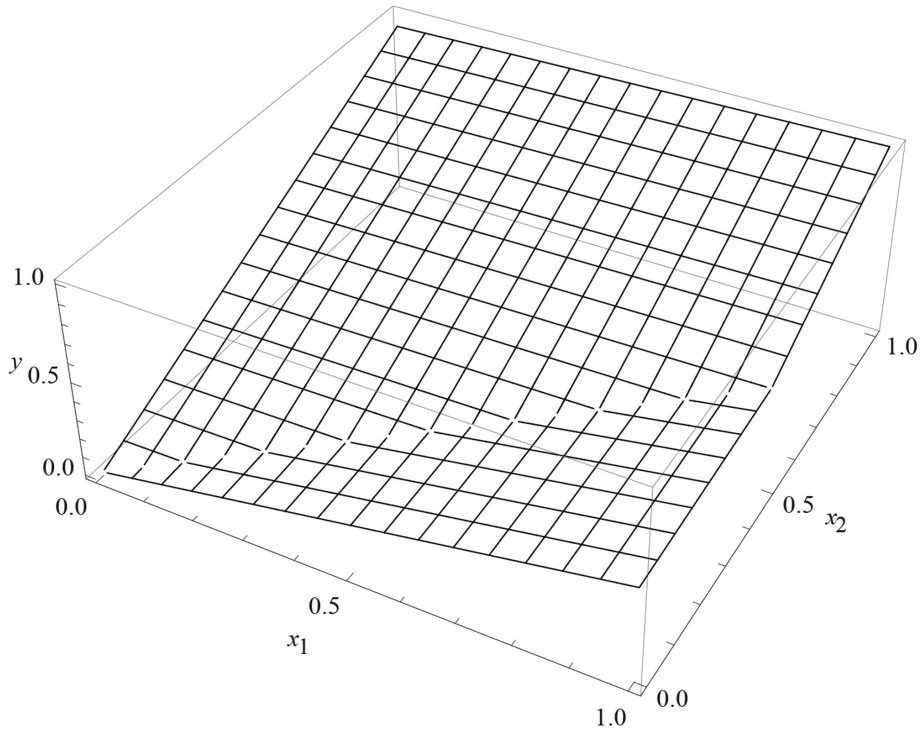

Figure 5. Function *WOWA* in Example 2 for $C = \prod$ and M the Hurwitz operator H.

weighting vectors $\mathbf{w}$, the proposed WOWA is well defined and continuous. It is a piecewise linear, increasing, idempotent, and homogeneous function. However, unlike the other mentioned WOWA, this function *is not* a discrete Choquet integral. This is evident from Figure 5 (note the set of nondifferentiability).

It is not difficult to see that the special case of equal weights $p_i = \frac{1}{n}$ corresponds to the unweighted OWA and $w_i = \frac{1}{n}$ corresponds to the WAM. However, reversing the weights of OWA does not produce the dual of the original function. The implicit WOWA in (7) is a valid alternative to the existing WOWA where the arguments are weighted both according their position and magnitude.

Example 2. Consider Hurwitz operator $M = H(\mathbf{x}) = a\max(\mathbf{x}) + (1-a)\min(\mathbf{x})$, $a \in]0, 1]$ and $C = \prod$. We have

$$WOWA_{(a,1-a)}(\mathbf{x}) = \frac{a\max(\mathbf{px}) + (1-a)\min(\mathbf{px})}{a\max(\mathbf{p}) + (1-a)\min(\mathbf{p})}.$$

Let us consider a particular case where $\mathbf{x} = (x_1, x_2)$, $a = \frac{9}{10}$, $\mathbf{p} = (p_1, p_2) = (\frac{1}{3}, \frac{2}{3})$,

then

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = \frac{\frac{9}{10}\max\left(\frac{1}{3}x_1, \frac{2}{3}x_2\right) + \frac{1}{10}\min\left(\frac{1}{3}x_1, \frac{2}{3}x_2\right)}{\frac{9}{10}\max\left(\frac{1}{3}, \frac{2}{3}\right) + \frac{1}{10}\min\left(\frac{1}{3}, \frac{2}{3}\right)} = \frac{9\max(x_1, 2x_2) + \min(x_1, 2x_2)}{19}.$$

F is plotted in Figure 5.

If we replace the products $p_i x_i$ with a more general function C strictly increasing on $\mathbb{R}_{++}\times]0, 1]$ we get a generalization of WOWA.

Example 3. C is replaced by the square of the product function in Example 2. We have

$$GenWOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = \left[\frac{\frac{9}{10}\max\left(\frac{1}{9}x_1^2, \frac{4}{9}x_2^2\right) + \frac{1}{10}\min\left(\frac{1}{9}x_1^2, \frac{4}{9}x_2^2\right)}{\frac{9}{10}\max\left(\frac{1}{9}, \frac{4}{9}\right) + \frac{1}{10}\min\left(\frac{1}{9}, \frac{4}{9}\right)}\right]^{1/2} = \left[\frac{9\max\left(x_1^2, 4x_2^2\right) + \min\left(x_1^2, 4x_2^2\right)}{37}\right]^{1/2}.$$

Therefore,

$$GenWOWA_{\mathbf{w},\mathbf{p}}(\mathbf{x}) = \left[\frac{9\max\left(x_1^2, 4x_2^2\right) + \min\left(x_1^2, 4x_2^2\right)}{37}\right]^{1/2}.$$

In difference to Example 2 *GenWOWA* is not piecewise linear, yet it is strictly increasing and idempotent.

7. ILLUSTRATIVE EXAMPLE AND DISCUSSION

We now illustrate the behavior of the three WOWA functions using Example 1. We consider a number of model applicants presented in Table I. For simplicity we take perfect satisfaction scores with respect to our criteria.

Table I. The applicants specified by their input vectors (R, T, I, S) in Example 1.

Candidate	Satisfaction scores	PnTA	Torra	Implicit	WAM
A	(1,0,1,1)	0.95	0.95	0.90	0.75
B	(1,1,0,1)	0.94	0.93	0.84	0.70
C	(0,1,1,1)	0.94	0.93	0.84	0.70
D	(1,1,1,0)	0.99	0.98	0.97	0.85
E	(1,0,1,0)	0.89	0.84	0.79	0.60
F	(1,0,0,1)	0.74	0.69	0.61	0.45

The overall aggregated score computed using three types of WOWA and the WAM.

In this case, all the methods including the WAM give reasonable ranking for the first six candidates $D \succ A \succ B \sim C \succ E \succ F$, although the outputs of the methods differ in numerical values. For instance, the PnTA and Torra's method give very close scores to the candidates A, B, C, D, which somewhat agrees with our intuition.

If we now change the OWA weights to $\mathbf{w} = (0.45, 0.45, 0.05, 0.05)$ we get different rankings, namely $D \succ A \succ B \sim C \sim E \succ F$ for PnTA but $D \succ A \succ B \sim C \succ E \succ F$ for Torra's and WAM, and $D \succ A \succ E \succ B \sim C \succ F$ for implicit OWA. We see here that the implicit OWA placed E over B, C, discounting the third largest criterion (service) but favoring higher weighted research and industry engagement.

If we bring another candidate $G = (0.8, 0.5, 0, 0.8)$, all WOWA methods will place him last (as they discounted the third largest input 0.5), but the WAM will rank $G \succ F$.

Another useful illustration is to compare the preferences generated by the WOWA and the weighted arithmetic mean. Consider two applicants $A = (0.8, 0.8, 0, 0.2)$ and $B = (0.5, 0.5, 0.5, 0.5)$, so that A shows excellency in research and teaching and some service capacity, whereas B is all-round but mediocre in all components. The WAM with the same weighting vector $\mathbf{p}$ will give the scores $WAM(A) = 0.47$, $WAM(B) = 0.5$, so that B is preferred to A. In contrast, $WOWA(A) = 0.68$ and $WOWA(B) = 0.5$, so that A is preferred to B, and despite lack of industry experience and little service, A would be considered stronger than B based on her strongest points. Similarly, the candidates with strong research and industry experience and no teaching experience, or excellency in teaching and industry engagement and no research would still be preferred to the mediocre B. We believe that WOWA better reflects the desire of the department as it is unreasonable to expect excellency in all the selection criteria from junior academics. WOWA function allows one to model "k out of n" criteria, whereas the WAM discriminates against the candidates who may not have opportunity to excel in one or two areas.

Very similar situation happens at academic promotion committees, where the assessment is done on the four mentioned criteria. As some applicants come from research-only or teaching-only positions, they obviously lack experience in one or another criterion. While the WAM pulls their overall score down due to zero score in that missing criterion, the WOWA ignores the zero score and allows one to compare candidates with very different skills set based on their strongest points, which matches the desires of decision makers.

Of course, OWA weights could be adjusted to emphasize more the strongest points and disregard the weaknesses. And to the contrary, OWA weights can be chosen to discard the biggest inputs and emphasize the smallest or middle-sized inputs, which is useful when filtering outliers.

Example 4. Consider now a selection panel that has three professors from distinct research fields, two other academics, and the head of department as the chair. In addition to the standard selection criteria (R, T, I, S), there is another one, the applicant's research should match at least one of the field of research conducted in the department. The panel members supply their overall scores ($P1$, $P2$, $P3$, $A1$, $A2$, H), which need to be aggregated into an overall score for ranking of the applicants.

Table II. The applicants specified by the individual evaluations by the panel members ($P1$, $P2$, $P3$, $A1$, $A2$, H) in Example 4.

Candidate	Evaluations	PnTA	Torra	Implicit	WAM
A	(1,**0**, 1, 1,1,1)	1	0.98	0.85	0.81
B	(**1**, 0.5, 0.5, 0.5, 0.5, 0.5)	0.5	0.5	0.57	0.59
C	(0.8, 0.8,**0**, 0.8, 0.8, 0.8)	0.8	0.78	0.68	0.65
D	(0.8, 0.8, 0.8,**0**, 0.8, 0.8)	0.8	0.8	0.8	0.72
E	(0.8, 0.8, 0.8, 0.8, 0.8,**0**)	0.65	0.67	0.68	0.58

The value in boldface indicates outliers. The overall aggregated score computed using three types of WOWA and the WAM.

Now, we assume the professors are biased in the sense that they would give extremely high scores if the applicant's research falls in their area in order to strengthen their research team. Hence, we need to counterbalance this bias. We aggregate the scores using the OWA function with these weights $\mathbf{w} = (0, 0.25, 0.25, 0.25, 0.25, 0)$, so that the highest and the lowest scores are disregarded. This OWA is called the trimmed mean in statistics. Thus, even if professor $P1$ supplied the perfect score to "his" applicant, and zero score to the one outside, this score will not be used in the calculations.

We also weight the opinions of the professors and the head of department higher using this weighting vector $\mathbf{p} = \frac{1}{11}(2, 2, 2, 1, 1, 3)$.

Table II presents prototypical candidates with their evaluations given by the panel members, together with the aggregated score. For candidate A the outlying score 0 was disregarded by the first two WOWA methods while it affected the WAM. The implicit OWA was affected less. Similarly, the outlying score 1 for candidate B was also disregarded by the first two WOWA and accounted to a lesser degree by implicit WOWA than by WAM.

The candidates C, D, E all have one outlying evaluation performed by different panel members, which have different importance weights. In the case D, the WOWA filter out the outlying value completely. In the case of C, the outlier is filtered completely or almost completely by the first two WOWA methods but not so by the remaining methods. In the case of E, all the WOWA methods give similar scores. In this case, even though there is one outlying value, because it is given by the chair of the panel, it is not filtered out. However, the weight of that value is reduced as compared to the WAM, which is heavily biased by that outlier. So we see that the weight of the panel member plays an important role here and determines how aggressively the outliers are downweighted. This functionality of WOWA appears to be in line with our expectations.

Let us now compare the three mentioned types of WOWA functions. All three methods produce averaging piecewise linear continuous functions and match the OWA and WAM functions in the limiting cases. WOWA based on interpolating the quantifier function and WOWA based on n-nary tree are both computationally efficient and preserve several useful properties of the original OWA function. Both result in a discrete Choquet integral. However, both methods are computer oriented:

while the algorithms employed are not too complicated, it is cumbersome to use these methods for quick pen and paper calculations. In contrast, the WOWA based on implicit averaging is quite simple for manual calculations, or to be used in a spreadsheet formula. Care should be taken to ensure the weights are positive in this case to avoid division by zero.

When the OWA function is max or min, the method based on the quantifier function also produce the max and min function irrespective of the weights $\mathbf{p}$, as long as $p_i > 0$. However, the third method produces a different function, similar to the weighted maximum and minimum functions. This may be advantageous as in this case small weights p_i ensure that the aggregated value depends on the corresponding variables to a lesser degree (unlike the max and min functions that are symmetric).

From numerical evidence, it appears that the implicit WOWA provides scores somewhat in between the other two WOWA and the WAM, although this behavior is not universal (e.g., candidate D in Table I).

Finally, the implementation of all the mentioned WOWA methods is available from Ref. 27.

8. CONCLUSIONS

We have advocated the use of WOWA functions for modeling decision-making problems in which the alternatives are not directly comparable as they present distinct strong points. The comparison can be based on partial satisfaction of k out of n criteria, or when removing one or few strongest and weakest inputs among other choices. This is achieved by associating the weights with the inputs magnitude in the standard OWA functions. WOWA combine the weights associated with the inputs and their magnitude, so that the decision-making criteria are also weighted jointly with the inputs magnitude.

We presented and compared three alternative constructions of WOWA functions. Two of those use implicit repetition of the inputs a suitable number of times as to match the desired weights and produce instances of the discrete Choquet integral. Both methods are computer oriented and numerically efficient. The remaining construction involves a simple analytic formula suitable for manual calculations, but differs from the Choquet integral. The presented WOWA methods are illustrated in several examples.

References

1. Beliakov G, Pradera A, Calvo T. Aggregation functions: a guide for practitioners, Studies in fuzziness and soft computing, vol. 221. Berlin, Germany: Springer-Verlag; 2007.
2. Beliakov G, Bustince H, Calvo T. A practical guide to averaging functions. Berlin, Germany: Springer; 2016.
3. Grabisch M, Marichal J-L, Mesiar R, Pap E. Aggregation functions. Encyclopedia of mathematics and its foundations. Cambridge, UK: Cambridge University Press; 2009.
4. Yager RR. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans Syst Man Cybern 1988;18:183–190.

5. Yager RR, Alajlan N. A generalized framework for mean aggregation: toward the modeling of cognitive aspects. Inf Fusion 2014;17:65–73.
6. Yager RR, Beliakov G. OWA operators in regression problems. IEEE Trans Fuzzy Syst 2010;18:106–113.
7. Yager RR, Kacprzyk J, Beliakov G, editors. Recent developments in the ordered weighted averaging operators: theory and practice. Berlin, Germany: Springer; 2011.
8. Yager RR. Quantifier guided aggregation using OWA operators. Int J Intell Syst 1996;11:49–73.
9. Yager RR. Families of OWA operators. Fuzzy Sets Syst 1993;59:125–148.
10. Yager RR. Using stress functions to obtain OWA operators. IEEE Trans Fuzzy Syst 2007;15:1122–1129.
11. Torra V. The weighted OWA operator. Int J Intell Syst 1997;12:153–166.
12. Torra V. The WOWA operator and the interpolation function W*: Chen and Otto's interpolation revisited. Fuzzy Sets Syst 2000;113:389–396.
13. Torra V. On some relationships between WOWA operator and the Choquet integral. 8th Int. Conf. on Information Processing and Management of Uncertainty, Paris, France; 1998. pp 818–824.
14. Narukawa Y, Torra V. Fuzzy measure and probability distributions: distorted probabilities. IEEE Trans Fuzzy Syst 2005;13:617–629.
15. Yager RR. Including importances in OWA aggregations using fuzzy systems modeling. IEEE Trans Fuzzy Syst 1998;6:286–294.
16. Yager RR, Alajlan N. Some issues on the OWA aggregation with importance weighted arguments. Knowl-Based Syst 2016;100:89–96.
17. Beliakov G. A method of introducing weights into OWA operators and other symmetric functions. In: Kreinovich V, editor. Uncertainty modeling. Dedicated to B. Kovalerchuk. Cham, Switzerland: Springer; 2017. pp 37–52.
18. Beliakov G, Calvo T, Fuster P. Implicit averaging functions. Under review 2016.
19. Calvo T, Mesiar R, Yager RR. Quantitative weights and aggregation. IEEE Trans Fuzzy Syst 2004;12:62–69.
20. Dujmovic JJ, Beliakov G. Idempotent weighted aggregation based on binary aggregation trees. Int J Intell Syst 2017;32:31–50.
21. Beliakov G, Dujmovic JJ. Extension of bivariate means to weighted means of several arguments by using binary trees. Inf Sci 2016;331:137–147.
22. Beliakov G, James S, Li G. Learning Choquet-integral-based metrics for semisupervised clustering. IEEE Trans Fuzzy Syst 2011;19:562–574.
23. Yager RR. Norms induced from OWA operators. IEEE Trans Fuzzy Syst 2010;18(1):57–66.
24. Yager RR. Connectives and quantifiers in fuzzy sets. Fuzzy Sets Syst 1991;40:39–76.
25. Beliakov G. Shape preserving splines in constructing WOWA operators: comment on paper by V. Torra in Fuzzy Sets and Systems 113 (2000) 389-396. Fuzzy Sets Syst 2001;121:549–550.
26. Beliakov G, Liu S. Parallel monotone spline interpolation and approximation on GPUs. In: R. Couturier, Ed. Designing scientific applications on GPUs. Boca Raton, FL: CRC Press; 2014. pp 295–310.
27. Beliakov G. Weighted OWA functions implementation in C++. Available at https://github.com/gbeliako/wowa, 2017.