A Method of Introducing Weights into OWA Operators and Other Symmetric Functions

Gleb Beliakov

Abstract This paper proposes a new way of introducing weights into OWA functions which are popular in fuzzy systems modelling. The proposed method is based on replicating the inputs of OWA the desired number of times (which reflect the importances of the inputs), and then using a pruned n-ary tree construction to calculate the weighted OWA. It is shown that this tree-based construction preserves many useful properties of the OWA, and in fact produces the discrete Choquet integral. A computationally efficient algorithm is provided. The tree-based construction is universal in its applicability to arbitrary symmetric idempotent n-ary functions such as OWA, and transparent in its handling the weighting vectors. It will be a valuable tool for decision making systems in the presence of uncertainty and for weighted compensative logic.

1 Introduction

One of the fundamental aspects in dealing with various forms of uncertainty is operations with uncertainty degrees, for example in the *if-then-else* rule statements. Theories that extend classical logic require extensions of the logical operations, such as conjunction, disjunction, negation and implication. In fuzzy logic in particular, membership degrees from the unit interval are combined by using aggregation functions [4, 14], which in the most general form are monotone increasing functions f 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.

The first class of prominent aggregation functions are the triangular norms and their duals, the triangular conorms [17]. The product was the first t-norm to appear alongside with the minimum in L. Zadeh's original paper on fuzzy sets [30]. When trying to mimic human decision making empirically, however, it became clear that

G. Beliakov (✉)
Deakin University, 221 Burwood Hwy, Burwood, Australia
e-mail: gleb@deakin.edu.au

V. Kreinovich (ed.), *Uncertainty Modeling*, Studies in Computational Intelligence 683, DOI 10.1007/978-3-319-51052-1_3

it is a combination of the t-norm and t-conorm, which itself is neither conjunctive nor disjunctive operation, that fits the data best [19, 31]. These functions expressed compensation properties, so that an increase in one input could be compensated by a decrease in another.

Another prominent class of aggregation functions that are in use since the early expert systems MYCIN and PROSPECTOR are the uninorms [4, 7, 26]. These are associative mixed operations that behave as conjunction on one part of their domain and as a disjunction on another. These functions, when scaled to the interval $[-1, 1]$, are useful bi-polar operations, where the information can be interpreted as "positive" or "negative", supporting or negating the conclusion of a logical statement or a hypothesis.

The class of averaging functions is the richest in terms of the number of interesting families. Averaging functions, whose prototypical examples are the arithmetic mean and the median, allow compensation between low values of some inputs and high values of the others. Such functions are also important for building decision models in weighted compensative logic [8], where the concept of Generalized Conjunction/Disjunction (GCD) play a role [10, 12].

Along with many classical means [5], the class of averaging functions contains such constructions as ordered weighted averages (OWA) [24] and fuzzy integrals [15]. The OWA functions in particular became very popular in fuzzy systems community [13, 28, 29]. These are symmetric functions which associate the weights with the magnitude of the inputs rather than with their sources.

The inability of OWA functions to associate weights with the specific arguments in order to model the concepts of importance and reliability of information sources was a stumbling block for their usage in some areas. However the concept of weighted OWA (WOWA) [20] resolved this issue. In WOWA functions, two vectors of weights are used. One vector has the weights associated with the arguments, thus modelling importance of the inputs, whereas the second vector associates weights with the magnitude of the inputs. It was later shown that WOWA are a special class of the discrete Choquet integral [18].

A different way of introducing weights into OWA was presented in [25]. In this paper, the OWA function was applied to the modified arguments, and the modifying function was found by using fuzzy modelling techniques. One such function was a linear combination of the argument and the orness (see Definition 3) of the respective OWA function. This method did not produce idempotent functions except in a few special cases.

In this contribution we present a different approach to introducing weights into OWA, based on n-ary tree construction and recursive application of the base OWA function. Such an approach based in binary trees was recently introduced by Dujmovic [9] in order to incorporate weights into bivariate symmetric means, as well as to extend bivariate means to n variables. More detailed analysis of the properties of this method is in [2, 11].

This paper is structured as follows. After presenting preliminaries in Sect. 2, we describe the construction of WOWA functions by Torra [20] in Sect. 3. In Sect. 4 we describe the method of introducing weights into arbitrary bivariate averaging

functions from [9]. Our main contribution is in Sect. 5, where we introduce the n-ary tree construction, outline some of its theoretical properties, and present an efficient computational algorithm. The conclusions are presented in Sect. 6.

2 Preliminaries

Consider now the following definitions adopted from [4, 14]. 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 $\mathbf{x} = (t, t, ..., t)$, $t \in \mathbb{I}$, 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

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

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 $f_\varphi(\mathbf{x}) = \varphi^{-1}\left(f\left(\varphi(x_1), \varphi(x_2), ..., \varphi(x_n)\right)\right)$.

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

Definition 7 For a given weighting vector w, $w_i \geq 0$, $\sum w_i = 1$, the **OWA function** is given by

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

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

The main properties of OWA are summarised below.

- As with all averaging aggregation functions, OWA are increasing (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.
- The OWA functions are special cases of the Choquet integral with respect to symmetric fuzzy measures.
- The special case 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

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

The OWA functions are OR-like if $orness(OWA_w) \geq \frac{1}{2}$ and AND-like if $orness(OWA_w) \leq \frac{1}{2}$. If the weighting vector is decreasing, i.e., $w_i \geq w_j$ whenever $i < j$, OWA is OR-like and is in fact a convex function. The respective (symmetric) fuzzy measure in this case is sub-modular [3]. The OWA functions with increasing weights are AND-like, concave functions which correspond to the Choquet integral with respect to a super-modular fuzzy measure. OWA with decreasing weighting vectors can be used to define norms [3, 23].

Similarly to quasi-arithmetic means, OWA functions have been generalized with the help of generating functions $g : \mathbb{I} \to [-\infty, \infty]$ as follows.

Definition 8 Let $g : \mathbb{I} \to [-\infty, \infty]$ be a continuous strictly monotone function and let w be a weighting vector. The function

$$GenOWA_{w,g}(x) = g^{-1}\left(\sum_{i=1}^{n} w_i g(x_{(i)})\right) \quad (4)$$

is called a **generalized OWA**. As for OWA, $x_{(i)}$ denotes the i-th largest value of x.

The generalized OWA is a φ-transform of the OWA function with $\varphi = g$. One special case is the Ordered Weighted Geometric (OWG) function studied in [16, 22]. It is defined by

Definition 9 For a given weighting vector w, the **OWG function** is

$$OWG_w(x) = \prod_{i=1}^{n} x_{(i)}^{w_i}. \tag{5}$$

Similarly to the weighted geometric mean, OWG is a special case of (4) with the generating function $g(t) = \log(t)$. Another special case is the Ordered Weighted Harmonic (OWH) function, where $g(t) = 1/t$.

A large family of generalized OWA functions is based on power functions, similar to weighted power means [27]. Let g_r denote the family of power functions

$$g_r(t) = \begin{cases} t^r, & \text{if } r \neq 0, \\ \log(t), & \text{if } r = 0. \end{cases}$$

Definition 10 For a given weighting vector w, and a value $r \in \mathbb{R}$, the function

$$GenOWA_{w,[r]}(x) = \left(\sum_{i=1}^{n} w_i x_{(i)}^r \right)^{1/r}, \tag{6}$$

if $r \neq 0$, and $GenOWA_{w,[r]}(x) = OWG_w(x)$ if $r = 0$, is called a **power-based generalized OWA**.

3 Weighted OWA

The weights in weighted means and in OWA functions represent different things. In weighted means w_i reflects the importance of the i-th input, whereas in OWA w_i reflects the importance of the i-th largest input. In [20] Torra proposed a generalization of both weighted means and OWA, called WOWA. This aggregation function has two sets of weights w, p. Vector p plays the same role as the weighting vector in weighted means, and 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 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 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.

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

Definition 11 Let w, p be two weighting vectors, $w_i, p_i \geq 0$, $\sum w_i = \sum p_i = 1$. The following function is called **Weighted OWA** function

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

where $x_{(i)}$ is the i-th largest component of 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 x, and g is a monotone non-decreasing 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.

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

Of course, the weights 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 [20, 21], see also [1] where Schumaker's quadratic spline algorithm was used, which automatically satisfies the straight line condition when needed.

It turns out that WOWA belongs to a more general class of Choquet integral based aggregation functions [18]. It is a piecewise linear function whose linear segments are defined on the simplicial partition of the unit cube $[0, 1]^n$: $\mathscr{S}_i = \{x \in [0, 1]^n | x_{p(j)} \geq x_{p(j+1)}\}$, where p is a permutation of the set $\{1, \ldots, n\}$. Note that there are exactly $n!$ possible permutations, the union of all $\mathscr{S}_i$ is $[0, 1]^n$, and the intersection of the interiors of $\mathscr{S}_i \cap \mathscr{S}_j = \emptyset$, $i \neq j$.

The next two sections introduce 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.

4 Binary Tree Construction

Consider now a method of incorporating weights into a symmetric bivariate idempotent function f, presented [9] and then extended in [2, 11]. To introduce the weights we use the approach from [6], where each argument x_i is replicated a suitable number of times. To be more precise, 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 $\frac{k_1}{2^L} \approx w_1$, $\frac{k_2}{2^L} \approx w_2$, and $k_1 + k_2 = 2^L$. Here w_i are the desired weights, and $L \geq 1$ is a specified number of levels of the binary tree shown in Fig. 1. One way of doing so is to take $k_1 = \lfloor w_1 2^L + \frac{1}{2} \rfloor$ 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 Fig. 1, 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 which contain the elements of the vector $\mathbf{X}$. In this example we took $w_1 = \frac{5}{8}$ and $w_3 = \frac{3}{8}$. 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 as follows:

Aggregation by Levels (ABL) Algorithm

1. Compute $k_1 := \lfloor w_1 2^L + \frac{1}{2} \rfloor$, $k_2 := 2^L - k_1$, and create the array $X := (x_1, \ldots, x_1, x_2, \ldots, x_2)$ by taking k_1 copies of x_1 and k_2 copies of x_2;
2. $N := 2^L$;
3. Repeat L times:
   (a) $N := N/2$;
   (b) For $i := 1 \ldots N$ do $X[i] := f(X[2i-1], X[2i])$;
4. return $X[1]$.

The algorithm is obviously terminating. The runtime of the ABL algorithm 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 presented in [2].

The pruning of the binary tree is done by using the idempotency of f, see Fig. 1, right. Indeed no invocation of f is necessary if both of its arguments are equal. Below

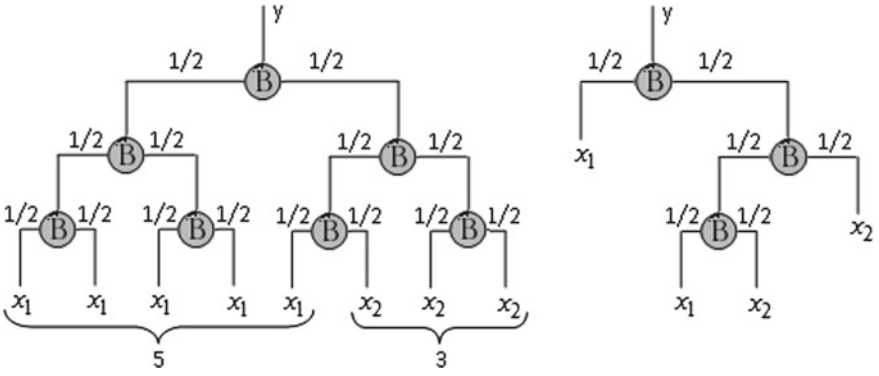

Fig. 1 Representation of a weighted arithmetic mean in a binary tree construction. The tree on the *right* is pruned by using idempotency

we present the pruned tree algorithm 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.

Pruned Tree Aggregation (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 $f\_n(w, x, L)$

1. create the array $N := (k_1, k_2)$ by using
   $k_1 := \lfloor w_1 2^L + \frac{1}{2} \rfloor$, 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.

As an example, consider the binary tree in Fig. 1. Here $L = 3$, $k_1 = 5$ and $k_2 = 3$. In the first call to function *node* instruction passes to step 2 where *node* is called recursively twice. In the first recursive call $N[1] = 5 \geq 2^2$ at step 1, hence x_1 is returned and $N[1]$ is set to 1. In the second call to *node* the instruction goes to step 2, where *node* is called recursively twice. In the first of those calls the recursion continues until $m = 0$, at which point x_1 is returned and K is incremented. In the subsequent call to *node* x_2 is returned and then $f(x_1, x_2)$ is computed and returned (the bottom level in the pruned tree in Fig. 1, right). At this point $N[2]$ becomes 2, and in the subsequent call to *node* step 1 is executed, x_2 is returned and subsequently aggregated in $f(f(x_1, x_2), x_2)$ (middle level of the tree in Fig. 1, right). That last output is aggregated with x_1 at the top level of the tree, and the recursive algorithm terminates, producing the output $y = f(x_1, f(f(x_1, x_2), x_2))$.

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 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.

The ABL and PTL algorithms produce identical outputs but differ in computational complexity. For this reason it may be convenient to formulate (or prove) the results in terms of the complete tree processed by algorithm ABL.

Several useful properties of the binary tree construction were presented in [2]. In particular, the weighted function f_w 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 ABL (and hence 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 w_i to match these weights exactly. Finally, when f is a quasi-arithmetic mean, f_w is a weighted quasi-arithmetic mean with the same generator.

Another contribution made in [2, 11] is the extension of the symmetric bivariate means to weighted n-variate means by using essentially the same approach, i.e., by replicating the n inputs a suitable number of times and constructing a binary tree with the desired numbed of levels L. The ABL and PTA algorithms in fact remain the same, safe the definition of the array of multiplicities N which is now n-dimensional.

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 [6] is very clear. The availability of a fast and uncomplicated algorithm for computing the output makes this method immediately applicable.

However the binary tree construction is not suitable for introducing weights into OWA functions, as here we already start with an n-variate function. The approach presented in the next section is an adaptation of the binary tree approach to n-variate OWA functions.

5 Importance Weights in OWA Functions

Our goal here is to incorporate a vector p of non-negative 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 Fig. 2. As the base symmetric aggregator f we take an OWA function OWA_w with specified weights w (although the origins of f are not important for the algorithm).

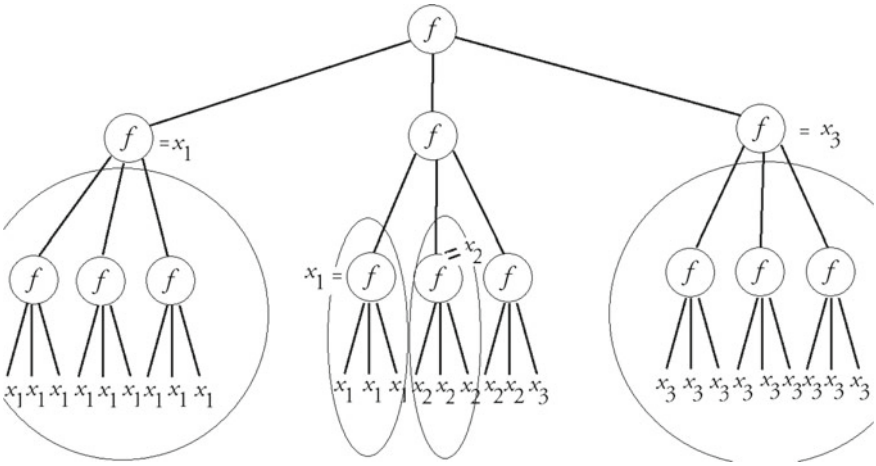

Fig. 2 Representation of a weighted tri-variate function f in a ternary tree construction. The weights are chosen as $\mathbf{p} = (\frac{12}{27}, \frac{5}{27}, \frac{10}{27})$ and $L = 3$. The *circled branches* are pruned by the algorithm

Now, 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 Fig. 2. 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 $k_n = n^L - k_1 - k_2 - \cdots - k_{n-1}$.

Pruned n-Tree Aggregation (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 $f(z)$.

function $f\_n(n, x, w, p, L)$

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 algorithm PnTA 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.

We also reiterate that 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).

Let us list some useful properties of the function f_p generated by the PnTA algorithm.

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 (resp. concave) then f_p is convex (resp. 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. □

Now we focus on the OWA functions as the base aggregator f. Here we can show the following.

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).*

Proof The Choquet integral is a piecewise linear continuous aggregation function where the linear pieces are joined together at the intersections of the canonical simplices $\mathscr{S}_i$, i.e., where two or more components of $\mathbf{x}$ are equal. Since OWA_w is

continuous and piecewise linear, so is f_p, by the properties of function composition. Now, let us show that the function f_p is not differentiable only on the sets where some of the components of the input vector $\mathbf{x}$ are equal, which will imply that the result is the discrete Choquet integral. Indeed at each node in the n-ary tree there is a function (OWA) not differentiable when some of inputs are equal. At the L-th level of the tree these are the points where the components of $\mathbf{x}$ are equal.

At the level $L-1$, the arguments of the OWA function are equal if and only if the arguments of the child nodes are equal, because the smallest argument of the left child node is no smaller than the largest argument of the right node (we recall that $\mathbf{X}$ is sorted). Continuing this recursion, we end up with the root node where the resulting function is not differentiable if and only if some of the arguments of the nodes at the bottom level L are equal, which are exactly the components of $\mathbf{x}$. Hence f_p is a piecewise linear, continuous aggregation function, and the linear pieces are joined together at the intersections of $\mathscr{S}_i$, so f_p is the discrete Choquet integral. □

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

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 weighted OWA with the reverse weights.*

Proof 1. In this case OWA_w is the arithmetic mean, and hence f_p is the weighted arithmetic mean with the respective weights.

2. In this case, each argument x_i is repeated exactly n^L times, hence the inputs to each node of the n-ary tree (except the root node) are all equal, and by idempotency the tree is pruned to just one level, and hence delivers the original OWA_w function.
3. The function at each node in the tree returns the minimum (maximum) of its arguments, hence the result does not depend of the weights if they are strictly positive. However, when the weights p_i could be 0, the result is not true, as the smallest (largest) component of $\mathbf{x}$ can be excluded from the calculation of the minimum (maximum). Note that f_p is not a weighted minimum or weighted maximum functions, as those are the instances of the Sugeno and not Choquet integral.
4. The weighted median can be written as the median of a vector with the components repeated the relevant number of times, i.e., *median*$(\mathbf{X})$ [4], p. 121. While in general the median of medians of subsets of inputs is not the median of the whole set of inputs, for an odd n at each level of the tree the median is the value of the central child node, which in turn is the value of its central child node, and so on until

the bottom level where we get the centralof $\mathbf{X}$; see Fig. 2 for an illustration. This statement does not work for an even n if we consider the lower (respectively, upper) medians, because now we need to take the value of the right middle child node at every level, but the lower median of $\mathbf{X}$ sorted in the decreasing order corresponds to $X_{n^L/2+1}$, which happens to be the left child of its parent node. This is clearly seen in the case of $n = 2$ where the lower median of the bivariate function f is the minimum of its arguments, hence $f_p(\mathbf{x}) = \min(\mathbf{X})$ which clearly does not always coincide with the median of $\mathbf{X}$. For example, in the binary tree in Fig. 1 $median(\mathbf{X}) = X_5 = x_1$ whereas $f_p(x_1, x_2) = X_8 = x_2$.

5. Follows from the preservation of φ-transform. □

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.*

Proof An OWA function with decreasing weights is convex (respectively concave for increasing weights), and hence is a special case of the Choquet integral with respect to a submodular (supermodular) fuzzy measure [3]. Since the convexity (concavity) is preserved in the n-ary tree construction as per Theorem 1, the resulting weighted function f_p is also convex (concave), and hence it is a Choquet integral with respect to a submodular (supermodular) fuzzy measure [3]. □

This result is useful when constructing weighted norms from OWA with decreasing weights, see [3, 23].

On the technical size 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 Fig. 3.

Finally by using Definition 8 we can introduce weights into generalized OWA functions in the same was as for OWA functions, by using the n-ary tree construction. This can be done in two ways: a) by using $GenOWA_{\mathbf{w},g}$ function as f, or b) by using a φ-transform of a weighted OWA 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. The second method is computationally more efficient as functions g and g^{-1} need not be used in the middle of the tree, where they cancel each other.

This way we also obtain the special cases of weighted OWG and weighted power based generalized OWA functions.

```
 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 root node L 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);
```

Fig. 3 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

6 Conclusions

The proposed method of introducing weights into n-ary symmetric functions has several advantages. Firstly, it is a generic method universally applicable to any symmetric idempotent function, in particular to OWA functions. Secondly, the handling of the weights is transparent and intuitive: the weights correspond to the multiplicities of the arguments. Thirdly, many important properties of the base symmetric aggregator are preserved in the n-ary tree construction, which is very useful as these properties need to be verified only for the base aggregator. Finally, the pruned n-ary tree algorithm delivers a numerically efficient way of calculating weighed averages, among them the weighted OWA. This algorithm has complexity linear in n and the number of levels of the tree L, and L is bounded by the desired accuracy of the weights. We believe the n-ary tree algorithm constitutes a very competitive alternative to the existing weighted OWA approaches.

References

1. G. Beliakov. Shape preserving splines in constructing WOWA operators: Comment on paper by V. Torra in Fuzzy Sets and Systems 113 (2000) 389–396. *Fuzzy Sets and Systems*, 121:549–550, 2001.
2. G. Beliakov and J.J. Dujmovic. Extension of bivariate means to weighted means of several arguments by using binary trees. *Information Sciences* 331:137–147, 2016.
3. G. Beliakov, S. James, and G. Li. Learning Choquet-integral-based metrics for semisupervised clustering. *IEEE Trans. on Fuzzy Systems*, 19:562–574, 2011.
4. G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*, volume 221 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, Berlin, 2007.
5. P.S. Bullen. *Handbook of Means and Their Inequalities*. Kluwer, Dordrecht, 2003.
6. T. Calvo, R. Mesiar, and R.R. Yager. Quantitative weights and aggregation. *IEEE Trans. on Fuzzy Systems*, 12:62–69, 2004.
7. B. De Baets and J. Fodor. Van Melle's combining function in MYCIN is a representable uninorm: An alternative proof. *Fuzzy Sets and Systems*, 104:133–136, 1999.
8. J.J. Dujmovic. Continuous preference logic for system evaluation. *IEEE Trans. on Fuzzy Systems*, 15:1082–1099, 2007.
9. J.J. Dujmovic. An efficient algorithm for general weighted aggregation. In *Proc. of the 8th AGOP Summer School*, Katowice, Poland, 2015.
10. J.J. Dujmovic. Weighted compensative logic with adjustable threshold andness and orness. *IEEE Trans. on Fuzzy Systems*, 23:270–290, 2015.
11. J.J. Dujmovic and G. Beliakov. Idempotent weighted aggregation based on binary aggregation trees. *International Journal of Intelligent Systems*, 32:31–50, 2017.
12. J.J. Dujmovic and H.L. Larsen. Generalized conjunction/disjunction. *Int. J. Approx. Reasoning*, 46:423–446, 2007.
13. A. Emrouznejad and M. Marra. Ordered weighted averaging operators 1988-2014: A citation-based literature survey. *Int. J. Intelligent Systems*, 29:994–1014, 2014.
14. M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap. *Aggregation Functions*. Encyclopedia of Mathematics and Its Foundations. Cambridge University Press, 2009.
15. M. Grabisch, T. Murofushi, and M. Sugeno, editors. *Fuzzy Measures and Integrals. Theory and Applications*. Physica – Verlag, Heidelberg, 2000.

16. F. Herrera and E. Herrera-Viedma. A study of the origin and uses of the Ordered Weighted Geometric operator in multicriteria decision making. *Int. J. Intelligent Systems*, 18:689–707, 2003.
17. E.P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer, Dordrecht, 2000.
18. Y. Narukawa and V. Torra. Fuzzy measure and probability distributions: Distorted probabilities. *IEEE Trans. on Fuzzy Systems*, 13:617–629, 2005.
19. U. Thole, H.-J. Zimmermann, and P. Zysno. On the suitability of minimum and product operators for the intersection of fuzzy sets. *Fuzzy Sets and Systems*, 2:167–180, 1979.
20. V. Torra. The weighted OWA operator. *Int. J. Intelligent Systems*, 12:153–166, 1997.
21. V. Torra. The WOWA operator and the interpolation function W*: Chen and Otto's interpolation revisited. *Fuzzy Sets and Systems*, 113:389–396, 2000.
22. Z.S. Xu and Q.L. Da. The ordered weighted geometric averaging operator. *Int. J. Intelligent Systems*, 17:709–716, 2002.
23. R. R. Yager. Norms induced from OWA operators. *IEEE Trans. on Fuzzy Systems*, 18(1):57–66, 2010.
24. R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Trans. on Systems, Man and Cybernetics*, 18:183–190, 1988.
25. R.R. Yager. Including importances in OWA aggregations using fuzzy systems modeling. *IEEE Trans. on Fuzzy Systems*, 6:286–294, 1998.
26. R.R. Yager. Uninorms in fuzzy systems modeling. *Fuzzy Sets and Systems*, 122:167–175, 2001.
27. R.R. Yager. Generalized OWA aggregation operators. *Fuzzy Optimization and Decision Making*, 3:93–107, 2004.
28. R.R. Yager and J. Kacprzyk, editors. *The Ordered Weighted Averaging Operators. Theory and Applications.* Kluwer, Boston, 1997.
29. R.R. Yager, J. Kacprzyk, and G. Beliakov, editors. *Recent Developments in the Ordered Weighted Averaging Operators: Theory and Practice*. Springer, Berlin, New York, 2011.
30. L. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
31. H.-J. Zimmermann and P. Zysno. Latent connectives in human decision making. *Fuzzy Sets and Systems*, 4:37–51, 1980.