This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

A Comprehensive Study of the Efficiency of Type-Reduction Algorithms

Chao Chen, *Member, IEEE,* Dongrui Wu, *Senior Member, IEEE,* Jonathan M. Garibaldi, *Senior Member, IEEE,* Robert I. John, *Senior Member, IEEE,* Jamie Twycross, and Jerry M. Mendel, *Life Fellow, IEEE*

***Abstract*—Improving the efficiency of type-reduction algorithms continues to attract research interest. Recently, there have been some new type-reduction approaches claiming that they are more efficient than the well-known algorithms such as the enhanced Karnik-Mendel (EKM) and the enhanced iterative algorithm with stopping condition (EIASC). In a previous paper, we found that the computational efficiency of an algorithm is closely related to the platform, and how it is implemented. In computer science, the dependence on languages is usually avoided by focusing on the complexity of algorithms (using big O notation). In this paper, the main contribution is the proposal of two novel type-reduction algorithms. Also, for the first time, a comprehensive study on both existing and new type-reduction approaches is made based on both algorithm complexity and practical computational time under a variety of programming languages. Based on the results, suggestions are given for the preferred algorithms in different scenarios depending on implementation platform and application context.**

***Index Terms*—centroid, type-reduction, interval type-2 (IT2) fuzzy set, Karnik-Mendel (KM) algorithm, enhanced KM (EKM) algorithm, enhanced iterative algorithm with stop condition (EIASC), direct approach (DA), non-derivative based direct approach (DAND), COSTRWSR, simplified COSTRWSR (SC).**

I. Introduction

DURING the past few years, there has been a steady increase of interest in developing type-2 fuzzy logic systems, and in particular interval type-2 fuzzy logic systems [1]. Interval type-2 fuzzy logic systems have been demonstrated to have better abilities to handle uncertainties than their type-1 counterparts in many applications [2, 3, 4, 5, 6, 7, 8]. However, the high computational cost of type-reduction algorithms makes it more expensive to deploy interval type-2 systems, especially for certain cost-sensitive real-world applications.

C. Chen, J. M. Garibaldi, R. John and J. Twycross are with the Laboratory for Uncertainty in Data and Decision Making (LUCID), the Intelligent Modelling and Analysis (IMA) and the Automated Scheduling Optimisation and Planning (ASAP) Research Groups, School of Computer Science, University of Nottingham, Nottingham, Jubilee Campus, NG8 1BB UK e-mail: {chao.chen, jon.garibaldi, robert.john, jamie.twycross}@nottingham.ac.uk.

D. Wu is with the Ministry of Education Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China. Email: drwu@hust.edu.cn.

J.M. Mendel is with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA. He is also with the College of Artificial Intelligence, Tianjin Normal University, Tianjin, China. Email: mendel@sipi.usc.edu.

This research was partly supported by the University of Nottingham, the National Natural Science Foundation of China (61873321), and the 111 Project on Computational Intelligence and Intelligent Control under Grant B18024.

Manuscript received *** **, 2019; revised *** **, 2019; accepted *** **, 2019. Date of publication *** **, 2019; date of current version*** **, 2019.

There have been a lot of type-reduction approaches proposed in the literature [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. While some of the recent work on type-reduction approaches is based on continuous algorithms or general type-2 fuzzy systems [22, 23], this paper focuses on discrete type-reduction approaches which are based on computing the centroid of an interval type-2 fuzzy set. The Karnik-Mendel (KM) algorithm is an iterative approach to determine the switch points when computing the centroids of IT2 fuzzy sets [9]. Type-reduction based on the KM algorithm is usually computationally intensive. Many attempts have been made to improve the efficiency of the KM algorithm. For example, the enhanced KM (EKM) algorithms have better initialisations, which "on average ... can save about two iterations" [11]. They have been the most well-known algorithms for type-reductions, and are still being widely used. Another well-known algorithm is the enhanced iterative algorithm with stopping condition (EIASC) which was proposed in [13]. The EIASC algorithm was reported to be superior to the KM and EKM algorithms when N, the number of discrete points in the universe of discourse for an IT2 fuzzy set, is small (e.g. $N < 100$).

Both EKM and EIASC are iterative based algorithms. A direct approach (DA) for determining the switch points in the KM Algorithm was introduced in [16]. It was shown by simulations in the R programming language that DA clearly outperformed other algorithms regardless of the shapes of fuzzy sets. An optimised version of the DA algorithm, termed the DA* algorithm, was later introduced in [20]. A recent algorithm called center of sets type reducer without sorting requirement (COSTRWSR) was proposed in [17]. As highlighted in the name, this algorithm does not utilise sorting which is required by the other above algorithms. It was illustrated in [17] that COSTRWSR is more efficient than six other enhanced variants of the Karnik–Mendel algorithm.

Almost all of the above algorithms were proposed by claiming better performance based on only time comparisons. It has been mentioned in [20] that the computational efficiency of an algorithm is closely related to the platform, and how it is implemented. In computer science, the dependence on languages is usually avoided by focusing on the complexity of algorithms (using big O notation).

In this paper, as a continuation of our previous work in [16] and [20], two novel type-reduction approaches are proposed. Also, a comprehensive study is made based on both algorithm complexity and practical computational time in order to give explicit recommendations on type-reduction algorithms. The rest of the paper is organised as follows: Sec-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

tion II summarises four existing related algorithms; Section III presents two new algorithms; and Section IV compares the algorithm complexity and the practical running time efficiency of the well-known algorithms described in this paper; After a brief discussion in Section V, we draw our conclusions in Section VI.

II. Existing Related Algorithms

In this section, we briefly summarise related existing algorithms to establish terminology and notation.

Let an IT2 fuzzy set $\tilde{A}$ be based on

$$\begin{aligned} x_i &\in X, \qquad i = 1, 2, ..., N \\ J_i &\equiv [\underline{u}_i, \bar{u}_i], \qquad 0 \leqslant \underline{u}_i \leqslant \bar{u}_i \leqslant 1 \end{aligned}$$

where x_i is the primary variable in the discrete universe of discourse X (note that x_i is in ascending order for i from 1 to N), J_i represents the membership grade interval for the primary variable x_i, and N is the number of discrete points in the universe of discourse of the IT2 fuzzy set.

For any given embedded type-1 fuzzy set[1], with membership grades $u_i \in J_i$ for all i, of such an IT2 fuzzy set $\tilde{A}$, the centroid is defined as:

$$c = \frac{\sum_{i=1}^{N} x_i u_i}{\sum_{i=1}^{N} u_i}. \tag{1}$$

The centroid interval of $\tilde{A}$ is defined to be $[c_l, c_r]$, where c_l and c_r are the minimum and maximum possible values of c respectively. The EKM [11], EIASC [13] and DA [16] algorithms are used to compute such a centroid interval as

$$c_l = \frac{\sum_{i=1}^{L} x_i \bar{u}_i + \sum_{i=L+1}^{N} x_i \underline{u}_i}{\sum_{i=1}^{L} \bar{u}_i + \sum_{i=L+1}^{N} \underline{u}_i} \tag{2}$$

$$c_r = \frac{\sum_{i=1}^{R} x_i \underline{u}_i + \sum_{i=R+1}^{N} x_i \bar{u}_i}{\sum_{i=1}^{R} \underline{u}_i + \sum_{i=R+1}^{N} \bar{u}_i} \tag{3}$$

L and R, which are integer indices in the range of $[1, N-1]$, are known to be the switch points to minimise and maximise c_l and c_r respectively.

EKM and EIASC are iterative algorithms for determining the switch points. In contrast, DA (or the optimised version DA*) is a direct approach based on derivatives. Due to the space limitation, these algorithms are only briefly reviewed below. Detailed summarisation can be found in the supplemental materials of this paper.

Note that in the commonly used center-of-sets (COS) type-reduction which concerns more general interval weighted average, x_i may also be an interval denoted by $[\underline{x}_i, \bar{x}_i]$. In such cases, $\underline{x}_i$ and $\bar{x}_i$ should be used for computing c_l and c_r respectively. However, in this paper, we do not distinguish between $\underline{x}_i$ and $\bar{x}_i$ for the sake of simplicity.

[1]The definition of an embedded type-1 fuzzy set can be found in [24].

A. The EKM algorithm

As mentioned in [13], there could be numerical issues or potential infinite loops for the EKM algorithm. It was clarified that these issues can be prevented by preprocessing steps or extra checks (see Appendix A in [13]). However, it should be noted that these extra steps, especially when they are not well implemented, may make the EKM significantly slower. In this paper, we have taken out the inefficient checks that were added in the implementation of EKM in [16].

B. The EIASC Algorithm

A key difference between EIASC and EKM is that of how to select the next potential solution to the switch point for each new iteration. EKM requires a search which obviously costs more time. In contrast, EIASC is a brute force method which iterates all the solutions one by one. Note that such strategy of EIASC is a ‘double-edged sword’. It reduces the complexity in finding the next solution, but makes EIASC an algorithm which heavily relies on loops. In fact, the use of loops is commonly not the first choice for efficient programming. As clarified in [20], loops are much less efficient in R than they are in Matlab. This is the key reason for the unsatisfactory performance of EISAC in [16].

C. The DA* Algorithm

It was found in [16] that the partial derivatives of c with respect to u_j are in ascending order with j from 1 to N. As illustrated in [16], the switch points are located at the indices where the sign of partial derivatives changes. Based on this, DA is a direct approach to find the switch points for type-reduction. In [20], the implementation of DA is optimised by eliminating some unnecessary computations and more efficient vectorisations. The optimised implementation is called DA*.

D. The COSTRWSR algorithm

COSTRWSR, which was proposed in [17], has a different basis to the above algorithms, not being based on the switch points. A table which summarises the COSTRWSR algorithm can be found in the supplemental material of this paper. Note that a key property of this algorithm is that there is no need to sort x_i in any case. This can save a lot of computations compared to other algorithms (e.g. EKM) for which sorting is required in some cases. It has to be mentioned that the original COSTRWSR algorithm in [17] can be easily and clearly enhanced (e.g. by moving Step 3 out of the loop declared in Step 6). Given that another more efficient but simplified algorithm will be proposed in this paper below, details of the enhanced COSTRWSR (ECOSTRWSR) algorithm will only be presented in the supplemental materials.

III. New Algorithms

In this section, we propose two new algorithms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

A. A Simplified COSTRWSR Algorithm

This section introduces a simplified COSTRWSR algorithm (SC). Note that, for the COSTRWSR proposed in [17], an extra parameter $\lambda_i \in [0,1]$ was added to c. The algorithm COSTRWSR is based on a property of the derivatives. That is, for example, when the derivative of c with respect to λ_j is positive, λ_j must be 1 in order to get c_r, or 0 to get c_l.

In fact, such a property of the derivatives, as described above, can be used to obtain c_l and c_r without the need to add the extra parameter λ_i. Specifically, the derivative of c with respect to u_j,

$$\frac{\partial c}{\partial u_j} = \frac{x_j - c}{\sum_{i=1}^{N} u_i}$$

can be used directly. Note that the denominator $\sum_{i=1}^{N} u_i$ is always positive, and hence it will not affect the sign of $\frac{\partial c}{\partial u_j}$. Let $A_j = x_j - c$, then when A_j is positive, u_j must be 1 in order to get c_r, or 0 to get c_l; when A_j is negative, u_j must be 0 in order to get c_r, or 1 to get c_l.

A_j and c can be rewritten as,

$$A_j = x_j - \frac{\delta_2}{\delta_1}$$
$$c = \frac{\delta_2}{\delta_1}$$

where

$$\delta_1 = \sum_{i=1}^{N} u_i$$
$$\delta_2 = \sum_{i=1}^{N} x_i u_i$$

The SC algorithm is summarised in Table I.

B. A Non-derivative based DA Algorithm

Recall that the DA algorithm is a direct approach to find the switch points for obtaining c_l and c_r based on the sign change of derivatives. This section introduces a new direct approach which is not based on derivatives (DAND) for obtaining c_l and c_r.

Essentially, DAND is a brute force method to get c_l and c_r. There is no need to find the switch points. For example, by Equation (2), we use all possible values of the switch point L from 1 to N to calculate and find the minimum value of c_l. Similarly, by Equation (3), c_r can be found with all possible values of the switch point R from 1 to N. Note that Equations (2) and (3) can be rewritten as (4) and (5),

$$c_l = \frac{\sum_{i=1}^{N} x_i \underline{u}_i + \sum_{i=1}^{L} x_i(\bar{u}_i - \underline{u}_i)}{\sum_{i=1}^{N} \underline{u}_i + \sum_{i=1}^{L}(\bar{u}_i - \underline{u}_i)} \quad (4)$$

$$c_r = \frac{\sum_{i=1}^{N} x_i \bar{u}_i - \sum_{i=1}^{R} x_i(\bar{u}_i - \underline{u}_i)}{\sum_{i=1}^{N} \bar{u}_i - \sum_{i=1}^{R}(\bar{u}_i - \underline{u}_i)} \quad (5)$$

By using cumulative summation for some of the above terms (e.g. $\sum_{i=1}^{L} x_i(\bar{u}_i - \underline{u}_i)$ and $\sum_{i=1}^{L}(\bar{u}_i - \underline{u}_i)$), the computational cost to obtain c_l and c_r can be reduced.

The DAND algorithm is summarised in Table II.

IV. Comparative Study

In this section, we compare the algorithms described above based on both algorithm complexity and practical[2] computational time.

A. Algorithm Complexity

Four existing algorithms (EKM, EIASC, DA* and COSTRWSR) and two new algorithms (DAND and SC) are compared based on the computational complexity. In this paper, the complexity of each algorithm is determined by the number of calculations and comparisons for obtaining c_l. The results are presented in Tables III to VI, and summarised in Table IX.

Note that Step 1 of COSTRWSR and SC is not considered since it is also used by other algorithms. Also note that the total numbers are approximations. Constant values are omitted when calculating the totals. For example, $2N-1$ is considered to be $2N$.

As can be observed in Table IX, the results can be summarised as follows: i) all these algorithms have a similar number of calculations and comparisons, except that SC seems clearly better than COSTRWSR (SC only needs approximately one sixth of the calculations of COSTRWSR); ii) regardless of the difference in the coefficients of N, the asymptotic time complexity of all these algorithms is linear $O(N)$ in terms of the big O complexity; iii) EKM and EIASC are also associated with L and m (the number of iterations), while other algorithms such as DAND and SC only depend on N (the number of discrete points).

B. Experimental Comparison

As discussed in [20], regardless of the algorithm used, the computational time difference between programming languages is very large. Results in one programming language cannot be simply extended to all languages. Hence, computational time comparisons were made under five commonly used programming languages (R, Matlab, C, Java and Python). Two example fuzzy sets from [16], and one control surface example from [25] are used for comparisons.

The test platform was a Macbook Pro (13-inch, 2017) with 3.10GHz Intel Core i5 processor and 16GB 2133 MHz LPDDR3 memory, running macOS High Sierra version 10.13.6. The programming languages and software environment are R x64 version 3.6.1, Matlab R2017b, Python 3.7, Apple LLVM version 10.0.0 (clang-1000.11.45.5) for C (compiled with options -O3 and -std=c99), and Java SE Development Kit 8, Update 202. Computational costs were measured by the user time returned by the built-in function(s) *proc.time* in R, *tic* and *toc* in Matlab, *clock* in C, *System.currentTimeMillis* in Java, and *time.process_time* in Python.

In our experiments, we start with the six algorithms described above in the complexity analysis. It was found that DAND is always more efficient than DA*, which is supported by the complexity analysis. Similarly, SC is always more

[2]By practical we mean the user time taken for the comparisons.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

TABLE I
THE SC ALGORITHM FOR COMPUTING THE CENTROID END POINTS (c_l AND c_r) OF AN IT2 FUZZY SET.

Step	The SC algorithm for computing c_l	The SC algorithm for computing c_r
1	If $\underline{u}_i = 0, \forall i \in [1, N]$, then	
	$c_l = \min(x_j)$,	$c_r = \max(x_j)$,
	$\forall j \in [1, N]$ with $\bar{u}_j \neq 0$. Stop.	
2	Initialise $\delta_i = 1, \Delta u_i = \underline{u}_i - \bar{u}_i, \forall i \in [1, N]$.	
3	Calculate $\left\{ \delta_1 = \sum_{i=1}^{N} \bar{u}_i, \quad \delta_2 = \sum_{i=1}^{N} x_i \bar{u}_i, \right\}$	
4	*flag* $= 0$	
5	For j from 1 to N, repeat the following operations of this Step.	
	$A_j = x_j \delta_1 - \delta_2$	
	If $A_j < 0$,	If $A_j > 0$,
	$\delta'_j = 1$, else $\delta'_j = 0$.	
	If $\delta'_j \neq \delta_j$, then	
	If $\delta_j = 1$, $\left\{ \begin{matrix} \mathit{flag} = 1, & \delta_1 = \delta_1 + \Delta u_j, \\ \delta_j = \delta'_j, & \delta_2 = \delta_2 + x_j \Delta u_j. \end{matrix} \right\}$	
	else $\left\{ \begin{matrix} \mathit{flag} = 1, & \delta_1 = \delta_1 - \Delta u_j, \\ \delta_j = \delta'_j, & \delta_2 = \delta_2 - x_j \Delta u_j. \end{matrix} \right\}$	
6	If *flag* $\neq 0$, go to Step 4; else	
	$c_l = \frac{\delta_2}{\delta_1}$	$c_r = \frac{\delta_2}{\delta_1}$

Note that there is no need to sort x_i, in any case, for the SC algorithm. Also note that Step 1 is included in the pre-processing steps for all the other algorithms in this paper. Compared to COSTRWSR in Table S-IV of the supplementary material, it is clear that Steps 3 and 5 in this Table need less computations.

TABLE II
THE DAND ALGORITHM FOR COMPUTING THE CENTROID END POINTS (c_l AND c_r) OF AN IT2 FUZZY SET.

Step	The DAND algorithm for computing c_l	The DAND algorithm for computing c_r
1	Sort x_i $(i = 1, 2, ..., N)$ in ascending order and match $\underline{u}_i$ and $\bar{u}_i$ accordingly with their respective x_i.	
2	Calculate	
	$a = \sum_{i=1}^{N} x_i \underline{u}_i$	$a = \sum_{i=1}^{N} x_i \bar{u}_i$
	$b = \sum_{i=1}^{N} \underline{u}_i$	$b = \sum_{i=1}^{N} \bar{u}_i$
3	Calculate the vector U, that is	
	$U = \{\bar{u}_i - \underline{u}_i\}$ for i from 1 to N	$U = \{\underline{u}_i - \bar{u}_i\}$ for i from 1 to N
4	Calculate vectors A, B and C, that are,	
	$A = a + \mathrm{cumsum}(XU)$	
	$B = b + \mathrm{cumsum}(U)$	
	$C = A/B$	
	, where X is the vector of x_i for i from 1 to N.	
5	$c_l = \min(C)$	$c_r = \max(C)$

Note that for the case in Section II, Step 1 is not necessary since x_i has already been defined in ascending order.

efficient than COSTRWSR. Hence, to make the results more concise, only four algorithms (EKM, EIASC, DAND and SC) are included in the time comparisons in this section. The results including the original COSTRWSR and DA* algorithms are presented in the Supplemental Material in Figs. S-1 to S-3.

1) Generalised bell-shaped IT2 fuzzy sets: The fuzzy sets used in this comparison are the same as those used in [16]. The vector X, containing x_i, is uniformly distributed from 0 to 10. $\bar{u}_i$ and $\underline{u}_i$ are defined by generalised bell-shaped function[3]:

$$\bar{u}_i = \frac{1}{1 + \left(\left(\frac{x_i - c}{\bar{a}}\right)^2\right)^b}$$

$$\underline{u}_i = \frac{1}{1 + \left(\left(\frac{x_i - c}{\underline{a}}\right)^2\right)^b}$$

where $\underline{a}$ and b are randomly selected between 1 and 2; $\bar{a}$ is the multiplication of $\underline{a}$ with a random number between 1 and

[3]Sometimes Gaussian MFs are also referred to as bell MFs in the literature. However, we use the definition used in the Matlab fuzzy logic toolbox, which is not a Gaussian MF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

TABLE III
THE COMPUTATIONAL COMPLEXITY OF EKM FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1	$k = [N/2.4]$	1	0	1
2	$a = \sum_{i=1}^{k} x_i \bar{u}_i + \sum_{i=k+1}^{N} x_i \underline{u}_i$	$2N$	0	1
3	$b = \sum_{i=1}^{k} \bar{u}_i + \sum_{i=k+1}^{N} \underline{u}_i$	N	0	1
4	$c = a/b$	1	0	1
5	Find $k' \in [1, N-1]$ such that $x_{k'} < c \leqslant x_{k'+1}$	0	N	$m+1$
6	If $k' = k$, set $c_l = c$ and stop;	0	1	$m+1$
7	$s = sign(k' - k)$	1	0	m
8	$a' = a + s \sum_{i=min(k,k')+1}^{max(k,k')} x_i(\bar{u}_i - \underline{u}_i)$	$3\|k - k'\| + 1$	0	m
9	$b' = b + s \sum_{i=min(k,k')+1}^{max(k,k')} (\bar{u}_i - \underline{u}_i)$	$\|k - k'\| + 1$	0	m
10	$c' = a'/b'$	1	0	m
11	Set $c = c'$, $a = a'$, $b = b'$ and $k = k'$. Go to Step 5;	0	0	m

Note that the total number of calculations from Steps 6 to 9 is approximately $4\,|[N/2.4] - L|$, regardless of m. Hence, the total number of calculations for EKM is $3N + 4\,|[N/2.4] - L|$, and the total number of comparisons is $(1 + m)N$.

TABLE IV
THE COMPUTATIONAL COMPLEXITY OF EIASC FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1	$k = 0$	0	0	1
2	$a = \sum_{i=1}^{N} x_i \underline{u}_i$	$2N$	0	1
3	$b = \sum_{i=1}^{N} \underline{u}_i$	N	0	1
4	$k = k + 1$	1	0	L
5	$u_k = \bar{u}_k - \underline{u}_k$	1	0	L
6	$a = a + x_k u_k$	2	0	L
7	$b = b + u_k$	1	0	L
8	$c = a/b$	1	0	L
9	If $c \leqslant x_{k+1}$, set $c_l = c$, $L = k$ and stop; Otherwise, go to Step 4;	0	1	L

The total number of calculations is $3N + 6L$, and the total number of comparisons is L.

2; c is a random number between 0 and 10.

To investigate the performance of algorithms under fuzzy sets of different size, N (the length of discretised X) is set to be 4, 16, 36, 64, 100, 144, 196, 256, 324 and 400 (10 different values). For each value of N, 5000 Monte Carlo simulations were made and the computational time costs were aggregated to be compared for each algorithm.

2) Generalised randomly-shaped IT2 fuzzy sets: This experimental comparison is designed to be similar to the first comparison in [13]. As mentioned in Chapter 8 of Mendel's book [26], all kinds of type-reductions are related to computing the interval-weighted average, which requires solutions to two optimization problems, one of which leads to c_l and the other to c_r. Hence, this experiment is associated with centre of sets type-reduction.

It is assumed that vectors X and $\bar{U}$, containing x_i and $\bar{u}_i$ respectively, are uniformly distributed from 0 to 1. $\underline{u}_i$ is the multiplication of $\bar{u}_i$ with a random number between 0 and 1. Similar to the above for generalised bell-shaped IT2 fuzzy sets, N (the length of discretised X) is set to be 4, 16, 36, 64, 100, 144, 196, 256, 324 and 400 (10 different values). For each value of N, 5000 Monte Carlo simulations were made and the computational time costs were aggregated to be compared for each algorithm.

3) Control Surface Computation: In this comparison, two-input and single-output IT2 fuzzy logic controllers (FLCs) using Gaussian membership functions (MFs) are considered [25]. Note that these FLCs are based on Takagi–Sugeno–Kang (TSK) models. Each input domain of a fuzzy logic controller has n MFs, and hence there are n^2 rules, where $n = 2, 4, 6, ..., 20$. Each of the n^2 rule consequents is represented by a crisp number. Then, a type-reduction algorithm is required to compute the output of the IT2 FLC. The set of crisp consequents of all the rules can be considered as X. Hence,

TABLE V
THE COMPUTATIONAL COMPLEXITY OF DA* FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1	$X' \leftarrow \{x_i - x_{i-1}, 0 \mid i = 2, 3, ..., N\}$	N	0	1
2	$S^1 \leftarrow \{\sum_{i=1}^{j} \bar{u}_i \mid j = 1, 2, ..., N\}$	N	0	1
3	$S^{2a} \leftarrow \{\sum_{i=1}^{j} \underline{u}_i \mid j = 1, 2, ..., N\}$	N	0	1
	$S^2 \leftarrow \{s_N^{2a} - s_i^{2a} \mid i = 1, 2, ..., N\}$	N	0	1
4	$T^P \leftarrow \{x_i' \cdot s_i^1 \mid i = 1, 2, ..., N\}$	N	0	1
5	$T^N \leftarrow \{x_i' \cdot s_i^2 \mid i = 1, 2, ..., N\}$	N	0	1
6				
7	$d^N \leftarrow \sum_{i=1}^{N} t_i^N$	N	0	1
8	$D \leftarrow \{\sum_{i=1}^{j} \left(t_i^P + t_i^N\right) \mid j = 1, 2, ..., N\}$	$2N$	0	1
9	Find the smallest $k \in 1, 2, ..., N-1$ such that $d_k \geqslant d^N$	0	N	1
10	**if** k ***exists*** **then** $L \leftarrow k$ **else** $L \leftarrow N-1$;	1	1	1
11	**if** $L \neq 1$ **then** $\frac{\partial c}{\partial u_L} \leftarrow d_{L-1} - d^N$ **else** $\frac{\partial c}{\partial u_L} \leftarrow -d^N$;	1	1	1
	$c_l = x_L - \frac{\partial c}{\partial u_L}\left(\sum_{i=1}^{L} \bar{u}_i + \sum_{i=L+1}^{N} \underline{u}_i\right)$	N	0	1

Pseudo code is taken from [20], where more details can be found. The total number of calculations and comparisons are $10N$ and N respectively. Note that Steps 2 to 9 here, which are optimised for better performance in R, add an extra N calculations (in Step 3) compared to original steps of DA. This is not necessary if DA* is implemented in C or Java, where the total number of calculations can be reduced to $9N$ by just using the original Steps 2 to 9 of DA.

TABLE VI
THE COMPUTATIONAL COMPLEXITY OF DAND FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1	$a = \sum_{i=1}^{N} x_i \underline{u}_i$	$2N$	0	1
2	$b = \sum_{i=1}^{N} \underline{u}_i$	N	0	1
3	$U \leftarrow \{\bar{u}_i - \underline{u}_i \mid i = 1, 2, ..., N\}$	N	0	1
4	$A = a + \mathrm{cumsum}(XU)$	$3N$	0	1
5	$B = b + \mathrm{cumsum}(U)$	$2N$	0	1
6	$C = A/B$	N	0	1
7	$c_l = \min(C)$	0	N	1

The number of calculations for Steps 4 and 5 can be reduced to $2N$ and N respectively. This can be achieved by adding a and b to the first element of XU and U respectively before calculating the cumulative sum. Hence, the total number of calculations is $10N$, or can be reduced to $8N$. The total number of comparisons is N.

for the defuzzification of the fuzzy controller, N (the length of discretised X) is equal to n^2.

The parameter settings of the fuzzy controller described above are as follows. The centre of each MF is given by a random number uniformly distributed from -1 to 1. The uncertain standard deviations of each MF are uniformly distributed from 0.1 to 0.5. The crisp consequent values of each rule for the fuzzy logic controller are uniformly distributed from -2 to 2.

To generate the control surface, each input domain is discretised into 10 points from -1 to 1. Hence computing a complete control surface for one fuzzy logic controller requires 100 (10×10) defuzzifications (computations of centroids). For each N, 50 fuzzy logic controllers based on the settings described above are generated for the comparison of algorithms. This means, for each N, there are 5000 (100×50) type-reductions to be performed by each algorithm in the comparisons.

C. Experimental Results

The results of the above three experimental comparisons are shown in Figs. 1 to 3 respectively. The comparisons for specified value of N are also presented in Tables X to XII. Here, one algorithm is considered to be more efficient than another if its computational time is smaller. Below, we briefly summarise the results observed.

For all the three cases: i), in Matlab, EIASC performs mostly the best. We use the word ‘mostly’ here since the computational time of EIASC is quite close to DAND and SC. And for some values of N, EIASC does not give the shortest computational time. ii), in R and Python, DAND is shown to be the most efficient, while EIASC and SC are much worse than other algorithms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

TABLE VII
THE COMPUTATIONAL COMPLEXITY OF COSTRWSR FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1				
2	Initialise $\lambda_i = 0.5, \forall i \in [1, N]$.	0	0	1
3	$\left\{\begin{matrix} \delta_1 = \sum_{i=1}^{N} \bar{u}_i, & \delta_3 = \sum_{i=1}^{N} x_i(\bar{u}_i - \underline{u}_i)(1-\lambda_i), \\ \delta_2 = \sum_{i=1}^{N} x_i\bar{u}_i, & \delta_4 = \sum_{i=1}^{N} (\bar{u}_i - \underline{u}_i)(1-\lambda_i). \end{matrix}\right\}$	$12N$	0	p
4	$flag = 0$	0	0	p
5	For j from 1 to N, repeat the following operations of this Step.			
	$A_j = x_j - \frac{\delta_2}{\delta_1} + \frac{\delta_3}{\delta_1} - \frac{\delta_4}{\delta_1} x_j$	7	0	pN
	if $A_j < 0$ **then** $\lambda'_j = 1$ **else** $\lambda'_j = 0$;	0	1	pN
	if $\lambda'_j \neq \lambda_j$ **then**	0	1	pN
	$\left\{\begin{matrix} flag = 1, & \delta_3 = \delta_3 + x_j(\bar{u}_i - \underline{u}_i)(\lambda_j - \lambda'_j), \\ \lambda_j = \lambda'_j, & \delta_4 = \delta_4 + (\bar{u}_i - \underline{u}_i)(\lambda_j - \lambda'_j). \end{matrix}\right\}$	9	0	qN
6	**if** $flag \neq 0$ **then** go to Step 3 **else**	0	1	p
	$c_l = \frac{\delta_2 - \delta_3}{\delta_1 - \delta_4}$	3	0	1

The total number of calculations and comparisons are $(19p + 9q)N$ and $(2p)N$ respectively. According to experiments, p is on average 3 with a maximum of 4, and q is on average less than 1.1. Hence, the total number of calculations and comparisons are on average less than $86N$ and $8N$ respectively.

TABLE VIII
THE COMPUTATIONAL COMPLEXITY OF SC FOR OBTAINING c_l.

Step	Pseudo Code	Calculations per Iteration	Comparisons per Iteration	Iterations
1				
2	Initialise $\delta_i = 1, \Delta u_i = \underline{u}_i - \bar{u}_i, \forall i \in [1, N]$.	N	0	1
3	$\left\{\delta_1 = \sum_{i=1}^{N} \bar{u}_i, \quad \delta_2 = \sum_{i=1}^{N} x_i\bar{u}_i,\right\}$	$3N$	0	1
4	$flag = 0$	0	0	p
5	For j from 1 to N, repeat the following operations of this Step.			
	$A_j = x_j\delta_1 - \delta_2$	2	0	pN
	if $A_j < 0$ **then** $\lambda'_j = 1$ **else** $\lambda'_j = 0$;	0	1	pN
	if $\delta'_j \neq \delta_j$ **then**	0	1	pN
	if $\delta_j = 1$, **then** $\left\{\begin{matrix} flag = 1, & \delta_1 = \delta_1 + \Delta u_j, \\ \delta_j = \delta'_j, & \delta_2 = \delta_2 + x_j\Delta u_j. \end{matrix}\right\}$	3	1	rN
	else $\left\{\begin{matrix} flag = 1, & \delta_1 = \delta_1 - \Delta u_j, \\ \delta_j = \delta'_j, & \delta_2 = \delta_2 - x_j\Delta u_j. \end{matrix}\right\}$	3	1	sN
6	**if** $flag \neq 0$ **then** go to Step 4 **else**	0	1	p
	$c_l = \frac{\delta_2}{\delta_1}$	1	0	1

The total number of calculations and comparisons are $(2p + 3r + 3s + 4)N$ and $(2p + r + s)N$ respectively. According to experiments, p is on average 3 with a maximum of 4, and $(r + s)$ is on average less than 0.6. Hence, the total number of calculations and comparisons are on average less than $14N$ and $9N$ respectively.

For the first case where sort is not needed for all the four algorithms (see comparisons in Fig. 1): i), in C, EIASC is the best except for small N (less than 200) where SC is the most efficient. ii), in Java, without considering the anomaly when N is 200, DAND is the quickest for most values of N; iii), in Matlab, SC performs as good as EIASC and they are both more efficient than other algorithms.

For the other two cases where sort is needed for all algorithms except SC, the results are shown in Figs. 2 and 3: i), in C and Java, SC performs remarkably better than other

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

TABLE IX
A SUMMARY OF THE COMPUTATIONAL COMPLEXITY OF DIFFERENT ALGORITHMS BASED ON THE NUMBER OF CALCULATIONS AND COMPARISONS FOR THE CORE PART OF THEIR IMPLEMENTATIONS FOR c_l

	Calculations	Comparisons
EKM	$3N + 4\,\lvert[N/2.4] - L\rvert$	$(1+m)N$
EIASC	$3N + 6L$	L
DA*	$10N$ or $9N$	$1N$
DAND	$10N$ or $8N$	$1N$
COSTRWSR	$86N$	$8N$
SC	$14N$	$9N$

L ($1 \leq L \leq N$) is the index of the switch point for c_l, and m (normally 2 to 6) is the number of iterations for the EKM algorithm. Note that the calculation complexity for DA* and DAND can be simplified to $9N$ and $8N$ respectively. However, it does not reduce the computational time clearly since operations with indices are required for the simplification.

algorithms; ii), in Matlab, SC is more efficient than other algorithms when N is small (e.g. $N < 100$ as shown in Figs. 2 and 3).

V. Discussion

Fundamentally, all algorithms are $O(N)$ whilst EKM and EIASC are technically $O(N+L)$. As summarised in Table IX, the number of calculations for DA* and DAND can be simplified to $9N$ and $8N$ respectively. However, such simplifications do not reduce the practical computational time clearly since operations with indices are required for the simplifications. Further, in practice, the computational complexity is not necessarily the only factor affecting the efficiency of algorithms, especially when they have the same computational complexity, which is $O(N)$. For example, the practical efficiency of an algorithm is closely related to its implementation. Also, the runtime environment is also important for the efficiency of algorithms. As can be observed from the results above, EIASC performs the best in Matlab, but much worse in R and Python. Results could also be different when comparisons are made under various operating systems or hardware (i.e. the compute infrastructure on which the codes run). Also note that for real-time applications results in C are more crucial as C/C++ are normally used for such applications.

As illustrated in Tables III to IX, the number of calculations and comparisons for DAND only depends on N. For other algorithms, they also depend on the number of iterations. For example, EKM and $EIASC$ also depends on L and m. Note that L and m vary for each specific case given a fixed number of N. As discussed in [20], DAND is more desirable for real-time control problems when the computational time of the algorithm needs to be known in advance.

In our experiments, the comparison based on generalised bell-shaped IT2 fuzzy sets is representative of the type-reduction on ordinary IT2 fuzzy sets. For such type-reductions, sort is not need for X. The experiments based on generalised randomly-shaped IT2 fuzzy sets and the control surface computation can be considered as the same scenarios. For type-reductions in these scenarios, the sorting process for X is required. It has to be mentioned that sort only needs to be done one time in many cases. For example, for applications based on TSK fuzzy models where the rule consequents are fixed values, sorting only has to be done once and it can be normally achieved offline (e.g. during the design process).

Note that a key property of SC is that there is no need to sort x_i in any case. This makes a clear difference for the comparisons made in C and Java. For example, as can be observed in Fig. 1, SC performs worse than other algorithms when sort is also not required for other algorithms. But, it is clearly more efficient than other algorithms which need sort in Fig. 2. However, the sort process does not make too much difference for SC in Matlab, R and Python.

In summary, though there are some differences among the number of calculations and comparisons, the asymptotic time complexity of all algorithms is $O(N)$. The practical time efficiency of algorithms varies under different programming languages. There is no single best algorithm for all cases. An appropriate algorithm should be selected based on specific needs (e.g. for which application and on which platform). Based on our comparisons, it is suggested that: i) EIASC is in general the best choice in Matlab; ii), DAND is the best to use in R and Python; iii) In C and Java, SC should be the best choice when sort is needed for x_i (e.g. the type-reduction and defuzzification process of fuzzy logic controllers), otherwise, EIASC is preferential (e.g. sort is not needed for the type-reduction of interval type-2 fuzzy sets); iv) DAND performs generally as good as EIASC in Matlab, C and Java; v) Given that the complexity of DAND only depends on N, DAND is more desirable for real-time control problems when the computational time of the algorithm needs to be known in advance.

VI. Conclusion

In this paper, two novel type-reduction algorithms (DAND and SC) have been proposed. A comprehensive comparison has been made with other existing algorithms. The comparisons were based on both algorithm complexity and practical time efficiency. Results showed that all the compared algorithms have the same asymptotic time complexity $O(N)$. On the other hand, the practical time efficiency of algorithms varies under different programming languages. All algorithm code, and experiments are available online [27]. The results showed that there is no single algorithm which is best for all cases. Suggestions for the algorithms to be used in different scenarios have been given based on our comparisons. For example, the algorithm to be used may be different depending on whether a sort for X is required. Generally, sorting is not needed for type-reductions on ordinary fuzzy sets. For type-reduction in computing the outputs of IT2 TSK fuzzy models, sorting is required. Note that for many applications (e.g. IT2 TSK fuzzy models where rule consequents are fixed values), sorting only needs to be done once offline. For such cases, algorithms can be selected based on the suggestions that are given for applications without sorting.

Though comparisons have been made under five commonly used programming languages, future work could be done with more languages. It has to be mentioned that current suggestions are given mainly based on programming languages. It

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

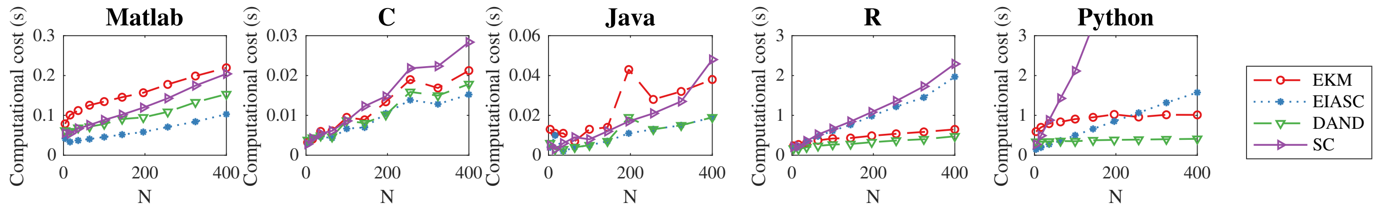

Fig. 1. Practical computational cost comparisons based on bell-shaped fuzzy sets.

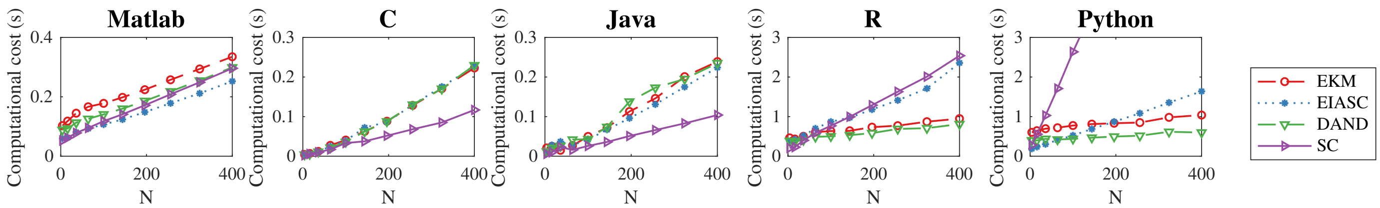

Fig. 2. Practical computational cost comparisons based on random-shaped fuzzy sets.

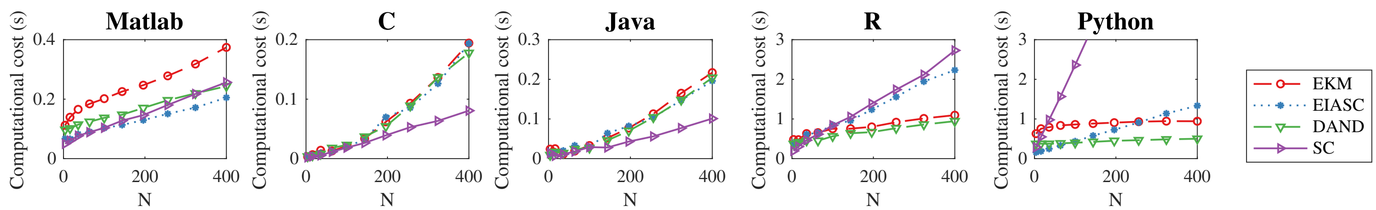

Fig. 3. Practical computational cost comparisons based on computing control surfaces.

TABLE X
PRACTICAL COMPUTATIONAL COST COMPARISONS (MIN / AVERAGE / MAX) BASED ON BELL-SHAPED FUZZY SETS WHEN $N = 100$

	Matlab	C	Java	R	Python
EKM	0.019 / 0.030 / 0.144	0.001 / ***0.003*** / 0.027	0.002 / 0.003 / 2.632	0.057 / 0.112 / 6.310	0.099 / 0.183 / 0.692
EIASC	0.009 / ***0.010*** / 0.045	0.001 / ***0.003*** / 0.011	0.001 / 0.003 / 0.014	0.099 / 0.147 / 5.523	0.092 / 0.105 / 0.403
DAND	0.015 / 0.019 / 0.896	0.001 / ***0.003*** / 0.007	0.001 / ***0.002*** / 0.019	0.042 / ***0.076*** / 5.343	0.070 / ***0.081*** / 0.245
SC	0.016 / 0.022 / 1.658	0.002 / 0.004 / 0.023	0.001 / ***0.002*** / 0.018	0.105 / 0.158 / 5.698	0.343 / 0.463 / 1.720

TABLE XI
PRACTICAL COMPUTATIONAL COST COMPARISONS (MIN / AVERAGE / MAX) BASED ON RANDOM-SHAPED FUZZY SETS WHEN $N = 100$

	Matlab	C	Java	R	Python
EKM	0.030 / 0.040 / 0.195	0.007 / 0.009 / 0.031	0.008 / 0.010 / 0.046	0.091 / 0.150 / 4.991	0.080 / 0.159 / 1.688
EIASC	0.020 / ***0.022*** / 0.151	0.007 / 0.009 / 0.065	0.009 / 0.014 / 0.083	0.136 / 0.191 / 4.517	0.097 / 0.116 / 0.740
DAND	0.027 / 0.034 / 0.320	0.007 / 0.013 / 0.139	0.008 / 0.009 / 0.026	0.083 / ***0.129*** / 22.84	0.084 / ***0.100*** / 1.468
SC	0.021 / 0.029 / 0.531	0.005 / ***0.008*** / 0.069	0.004 / ***0.005*** / 0.035	0.128 / 0.179 / 6.111	0.431 / 0.646 / 1.721

TABLE XII
PRACTICAL COMPUTATIONAL COST COMPARISONS (MIN / AVERAGE / MAX) BASED ON COMPUTING CONTROL SURFACES WHEN $n = 10$, WHICH IS EQUIVALENT TO $N = 100$

	Matlab	C	Java	R	Python
EKM	0.030 / 0.046 / 0.189	0.004 / 0.006 / 0.020	0.004 / 0.016 / 0.615	0.096 / 0.158 / 6.505	0.070 / 0.168 / 0.332
EIASC	0.020 / ***0.022*** / 0.102	0.004 / 0.006 / 0.046	0.005 / 0.012 / 0.239	0.101 / 0.195 / 23.41	0.039 / 0.118 / 0.201
DAND	0.026 / 0.030 / 0.207	0.004 / 0.006 / 0.043	0.003 / 0.006 / 0.052	0.088 / ***0.127*** / 4.390	0.074 / ***0.086*** / 0.240
SC	0.017 / 0.023 / 0.118	0.002 / ***0.005*** / 0.025	0.001 / ***0.005*** / 0.209	0.121 / 0.218 / 5.148	0.354 / 0.663 / 1.695

may worth a further exploration of the efficiency of these algorithms in different types of real-world applications or scenarios in future work. Also, this paper focuses on discrete type-reduction approaches for interval type-2 fuzzy sets. In the future, study could be done with other approaches for general type-2 fuzzy systems.

Acknowledgement

The authors dedicate this paper to our dear friend and colleague, Professor Robert I. (Bob) John, who passed away whilst this paper was being peer reviewed.

References

[1] J. Mendel, H. Hagras, W.-W. Tan, W. W. Melek, and H. Ying, *Introduction To Type-2 Fuzzy Logic Control: Theory and Applications*, 1st ed. Wiley-IEEE Press, 2014.

[2] Q. Liang, N. N. Karnik, and J. M. Mendel, "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 3, pp. 329–339, aug 2000.

[3] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 8, pp. 829–841, 2006.

[4] D. Wu and W. W. Tan, "A simplified type-2 fuzzy logic controller for real-time control," *ISA Transactions*, vol. 45, no. 4, pp. 503–516, 2006.

[5] Z. Liu, Y. Zhang, and Y. Wang, "A Type-2 Fuzzy Switching Control System for Biped Robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1202–1213, nov 2007.

[6] H. Hagras, "Type-2 FLCs: A New Generation of Fuzzy Controllers," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30–43, feb 2007.

[7] O. Castillo and P. Melin, *Type-2 Fuzzy Logic: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[8] E. A. Jammeh, M. Fleury, C. Wagner, H. Hagras, and M. Ghanbari, "Interval Type-2 Fuzzy Logic Congestion Control for Video Streaming Across IP Networks," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 5, pp. 1123–1142, oct 2009.

[9] N. Karnik and J. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, no. 1–4, pp. 195–220, 2001.

[10] M. Nie and W. W. Tan, "Towards an efficient type-reduction method for interval type-2 fuzzy logic systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2008, pp. 1425–1432.

[11] D. Wu and J. M. Mendel, "Enhanced Karnik–Mendel algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 923–934, 2009.

[12] S. Greenfield, F. Chiclana, S. Coupland, and R. John, "The collapsing method of defuzzification for discretised interval type-2 fuzzy sets," *Information Sciences*, vol. 179, no. 13, pp. 2055–2069, jun 2009.

[13] D. Wu and M. Nie, "Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2011, pp. 2131–2138.

[14] C. Y. Yeh, W. H. R. Jeng, and S. J. Lee, "An Enhanced Type-Reduction Algorithm for Type-2 Fuzzy Sets," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 227–240, 2011.

[15] S. Greenfield, F. Chiclana, R. John, and S. Coupland, "The sampling method of defuzzification for type-2 fuzzy sets: Experimental evaluation," *Information Sciences*, vol. 189, pp. 77–92, apr 2012.

[16] C. Chen, R. John, J. Twycross, and J. M. Garibaldi, "A Direct Approach for Determining the Switch Points in the Karnik–Mendel Algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 1079–1085, apr 2018.

[17] M. A. Khanesar, A. J. Khakshour, O. Kaynak, and H. Gao, "Improving the Speed of Center of Sets Type Reduction in Interval Type-2 Fuzzy Systems by Eliminating the Need for Sorting," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1193–1206, 2017.

[18] J. Li, R. John, S. Coupland, and G. Kendall, "On Nie-Tan operator and type-reduction of interval type-2 fuzzy sets," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, p. 1, 2017.

[19] T. A. Runkler, S. Coupland, R. John, and C. Chen, "Interval type-2 defuzzification using uncertainty weights," in *Frontiers in Computational Intelligence*, S. Mostaghim, A. Nürnberger, and C. Borgelt, Eds. Springer International Publishing, 2017.

[20] C. Chen, D. Wu, J. M. Garibaldi, R. John, J. Twycross, and J. M. Mendel, "A Comment on "A Direct Approach for Determining the Switch Points in the Karnik-Mendel Algorithm"," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 6, pp. 3905 – 3907, 2018.

[21] T. A. Runkler, C. Chen, and R. John, "Type reduction operators for interval type–2 defuzzification," *Information Sciences*, vol. 467, pp. 464–476, 2018.

[22] E. Ontiveros-Robles, P. Melin, and O. Castillo, "New Methodology to Approximate Type-Reduction Based on a Continuous Root-Finding Karnik Mendel Algorithm," *Algorithms*, vol. 10, no. 3, 2017.

[23] E. Ontiveros, P. Melin, and O. Castillo, "High order α-planes integration: A new approach to computational cost reduction of General Type-2 Fuzzy Systems," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 186–197, 2018.

[24] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," pp. 117–127, 2002.

[25] D. Wu, "Approaches for Reducing the Computational Cost of Interval Type-2 Fuzzy Logic Systems: Overview and Comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2013.

[26] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions, 2nd Edition*. Springer International Publishing, 2017.

[27] C. Chen and D. Wu, "Source Code for 'A Comprehensive Study of the Efficiency of Type-Reduction Algorithms'," https://www.codeocean.com/, 2019.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2020.2981002, IEEE Transactions on Fuzzy Systems

Chao Chen received the B.Eng. degree in Electronic and Information Engineering from Tianjin University of Technology, Tianjin, China, in 2003, the M.Sc. degree (Distinction) in Management of Information Technology and the Ph.D. degree (with the Vice-Chancellor's Scholarship for Research Excellence) in Computer Science from the University of Nottingham, Nottingham, UK, in 2012 and 2017, respectively. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad in 2017. Dr. Chen is currently a research fellow with the School of Computer Science at the University of Nottingham. He is also a member of the Laboratory for Uncertainty in Data and Decision Making (LUCID) and the Intelligent Modelling and Analysis (IMA) Research Group. His main research interests include the modelling of fuzzy logic systems for different types of applications such as time series forecasting and medical image analysis. He has a particular interest in the optimisation of fuzzy inference systems with different techniques. One of his recent publications, *A new accuracy measure based on bounded relative error for time series forecasting*, has been recommended by leading researchers on forecasting methods and applications, and is receiving more and more attention.

Dongrui Wu (S'05-M'09-SM'14) received the BE degree in automatic control from the University of Science and Technology of China in 2003, the ME degree in electrical engineering from the National University of Singapore in 2005, and the PhD degree in electrical engineering from the University of Southern California in 2009. He is now Professor in the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, and Deputy Director of the Key Laboratory of Image Processing and Intelligent Control, Ministry of Education. His research interests include affective computing, brain computer interfaces, computational intelligence, and machine learning. He has more than 140 publications.

Dr. Wu received the IEEE Computational Intelligence Society Outstanding PhD Dissertation Award in 2012, the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2014, the NAFIPS Early Career Award in 2014, the IEEE Systems, Man and Cybernetics (SMC) Society Early Career Award in 2017, and the IEEE SMC Society Best Associate Editor Award in 2018. He was also a finalist of another three Best Paper Awards. He was/is an Associate Editor of the IEEE Transactions on Fuzzy Systems (2011-2018), the IEEE Transactions on Human-Machine Systems (2014-), the IEEE Computational Intelligence Magazine (2017-), and the IEEE Transactions on Neural Systems and Rehabilitation Engineering (2019-).

Jonathan M. Garibaldi received the B.Sc (Hons) degree in Physics from Bristol University, UK in 1984, and the M.Sc. degree in Intelligent Systems and the Ph.D. degree in Uncertainty Handling in Immediate Neonatal Assessment from the University of Plymouth, UK in 1990 and 1997, respectively. He is Head of School of Computer Science at the University of Nottingham, UK, and leads the Intelligent Modelling and Analysis (IMA) Research Group. The IMA research group undertakes research into intelligent modelling, utilising data analysis and transformation techniques to enable deeper and clearer understanding of complex problems. His main research interests are modelling uncertainty and variation in human reasoning, and in modelling and interpreting complex data to enable better decision making, particularly in medical domains. He has made many theoretical and practical contributions in fuzzy sets and systems, and in a wide range of generic machine learning techniques in real-world applications. Prof. Garibaldi has published over 300 papers on fuzzy systems and intelligent data analysis, and is the Editor-in-Chief of IEEE Transactions on Fuzzy Systems (2017-). He has served regularly in the organising committees and programme committees of a range of leading international conferences and workshops, such as FUZZ-IEEE, WCCI, EURO and PPSN. He is a Senior Mmber of the IEEE.

Robert I. (Bob) John received the B.Sc. (Hons.) degree in mathematics from Leicester Polytechnic, Leicester, U.K., the M.Sc. degree in statistics from UMIST, Manchester, U.K., and the Ph.D. degree in Fuzzy Logic from De Montfort University, Leicester, U.K., in 1979, 1981, and 2000, respectively. He worked in industry for 10 years as a mathematician and knowledge engineer, developing AI systems for British Gas and the financial services industry. Bob spent 24 years at De Montfort University, before joining the University of Nottingham in 2013. He headed up the COL (formerly ASAP) research group in the School of Computer Science, and was also a member of LUCID. Bob published over 250 papers, including co-authoring the seminal papers "Type-2 fuzzy sets made simple" and "Interval type-2 fuzzy logic systems made simple". Bob passed away in February 2020, following a short illness. This paper is dedicated to his memory.

Jamie Twycross is an Assistant Professor in Computer Science at the University of Nottingham. He has a B.Sc. (Hons) in Mathematical Physics from Imperial College, London, an M.Sc. in Evolutionary and Adaptive Systems from the University of Sussex, and a Ph.D. in Computer Science from the University of Nottingham. His main research interest is in Computational Biology, where he works at the interface of computer science and biology to develop and apply computational and mathematical approaches to address biological and digital problems. He has expertise in computational and mathematical modelling, data analytics, machine learning, and software engineering. He is a member of the Intelligent Modelling and Analysis Group, and leads the Modelling Group in the Synthetic Biology Research Centre at the University of Nottingham.

Jerry M. Mendel (LF'04) received the Ph.D. degree in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, NY. Currently, he is Emeritus Professor of Electrical Engineering at the University of Southern California in Los Angeles, where he has been since 1974. He is also a Tianjin 1000-Talents Foreign Experts Plan Endowed Professor, and Honorary Dean of the College of Artificial Intelligence, Tianjin Normal University, Tianjin, China. He has published over 580 technical papers and is author and/or co-author of 13 books, including *Uncertain Rule-based Fuzzy Systems: Introduction and New Directions, 2nd ed.* (Springer 2017), *Perceptual Computing: Aiding People in Making Subjective Judgments* (Wiley & IEEE Press, 2010), and *Introduction to Type-2 Fuzzy Logic Control: Theory and Application* (Wiley & IEEE Press, 2014). He is a Life Fellow of the IEEE, a Distinguished Member of the IEEE Control Systems Society, and a Fellow of the International Fuzzy Systems Association. He was President of the IEEE Control Systems Society in 1986, a member of the Administrative Committee of the IEEE Computational Intelligence Society for nine years, and Chairman of its Fuzzy Systems Technical Committee and the Computing With Words Task Force of that TC. Among his awards are the 1983 Best Transactions Paper Award of the IEEE Geoscience and Remote Sensing Society, the 1992 Signal Processing Society Paper Award, the 2002 and 2014 *IEEE Transactions on Fuzzy Systems* Outstanding Paper Awards, a 1984 IEEE Centennial Medal, an IEEE Third Millenium Medal, a Fuzzy Systems Pioneer Award (2008) from the IEEE Computational Intelligence Society for fundamental theoretical contributions and seminal results in fuzzy systems". His present research interests include: type-2 fuzzy logic systems and computing with words.