An Enhanced Type-Reduction Algorithm for Type-2 Fuzzy Sets

Chi-Yuan Yeh, Wen-Hau Roger Jeng, and Shie-Jue Lee, *Member, IEEE*

***Abstract*—Karnik and Mendel proposed an algorithm to compute the centroid of an interval type-2 fuzzy set efficiently. Based on this algorithm, Liu developed a centroid type-reduction strategy to carry out type reduction for type-2 fuzzy sets. A type-2 fuzzy set is decomposed into a collection of interval type-2 fuzzy sets by α-cuts. Then, the Karnik–Mendel algorithm is called for each interval type-2 fuzzy set iteratively. However, the initialization of the switch point in each application of the Karnik–Mendel algorithm is not a good one. In this paper, we present an improvement to Liu's algorithm. We employ the previously obtained result to construct the starting values in the current application of the Karnik–Mendel algorithm. Convergence in each iteration, except the first one, can then speed up, and type reduction for type-2 fuzzy sets can be carried out faster. The efficiency of the improved algorithm is analyzed mathematically and demonstrated by experimental results.**

***Index Terms*—α-Cut, α-plane, centroid type reduction, fuzzy inference, Karnik–Mendel algorithm, membership function, type-1 fuzzy set, type-2 fuzzy system.**

I. Introduction

Type-1 fuzzy sets, which represent uncertainties by numbers in the range [0, 1], have been widely applied in fuzzy systems in different areas of applications [1]. However, the membership functions of type-1 fuzzy sets are often overly precise. They require that each element of the universal set be assigned a precise real number [2]. Type-2 fuzzy sets, instead, have been proposed for which the associated membership degrees are allowed to be uncertain and denoted as type-1 fuzzy sets [3]–[9]. Some successful applications of fuzzy systems using type-2 fuzzy sets have been published, such as automatic control [10]–[12], function approximation [13], [14], data classification [15]–[17], and medical applications [18], [19], but most of them use interval type-2 fuzzy sets which are special type-2 fuzzy sets. This may be due to the reason that the inference involving type-2 fuzzy sets is more complex and less efficient than that involving interval type-2 fuzzy sets.

Type reduction is one of the major steps involved in type-2 fuzzy inference. It does the work of reducing a type-2 fuzzy set to a type-1 one. A type-2 fuzzy system contains a set of type-2 fuzzy rules in which variables are expressed in terms of type-2 fuzzy sets. When providing desired inputs, one would like to obtain a crisp-valued output deduced from the type-2 fuzzy sets contained in the system. Usually, the aggregated result combined from the deduced conclusions of individual rules is a type-2 fuzzy set. One then has to apply type reduction to reduce the obtained type-2 fuzzy set to a type-1 set. Following this, defuzzification is applied to convert the obtained type-1 fuzzy set into a crisp number. Usually, type reduction is much more time-consuming than defuzzification. Therefore, making type reduction more efficient can do good to the growing interest in using type-2 fuzzy systems [20]–[25]. Recently, several type-reduction methods for type-2 fuzzy sets have been proposed. Liu [26] proposed a centroid type-reduction strategy using α-cuts to decompose a type-2 fuzzy set into a collection of interval type-2 fuzzy sets and then applying the Karnik–Mendel algorithm [27] to do type reduction for each interval type-2 fuzzy set. Coupland and John [28] proposed a geometric-based defuzzification method for type-2 fuzzy sets. It was claimed to be faster than type-reduction-based method. However, it has a limitation on the form of fuzuy sets being used. Rotationally symmetrical membership functions are to be avoided in a practical system. Lucas *et al.* [29] also applied type reduction to land-cover classification [17]. In this study, a centroid is calculated for each vertical slice, which is a type-1 fuzzy set. The calculated centroids are then combined to form a type-reduced set. There are also other type-reduction methods. For example, Tan and Wu [30] introduced the concept of equivalent type-1 fuzzy sets. By replacing a type-2 fuzzy set with a collection of equivalent type-1 fuzzy sets, type reduction can be simplified to deciding which equivalent type-1 fuzzy set to employ in a particular situation. A genetic algorithm (GA) was developed to select the equivalent type-1 fuzzy set in the evolution process.

In this paper, we propose an improvement to Liu's algorithm for type-2 fuzzy sets. In Liu's algorithm, the initialization of the switch point in each application of the Karnik–Mendel algorithm is not a good one. We employ the previously obtained result to construct the starting values in the current application of the Karnik–Mendel algorithm. As a result, unnecessary computations and comparisons are avoided. Convergence in each iteration, except the first one, can speed up and type reduction can be done faster. The rest of this paper is organized as follows. Section II introduces some basic fuzzy concepts. Section III describes type reduction for type-2 fuzzy sets. Section IV presents the improved type-reduction algorithm. Mathematical work on efficiency analysis is also provided in this section. Section V gives an example for illustration.

Manuscript received December 12, 2009; revised July 27, 2010; accepted July 27, 2010. Date of publication November 18, 2010; date of current version April 4, 2011. This work was supported by the National Science Council (NSC) under Grant NSC-97-2221-E-110-048-MY3 and Grant NSC-98-2221-E-110-052.

The authors are with the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 804, Taiwan (e-mail: leesj@mail.ee.nsysu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2010.2093148

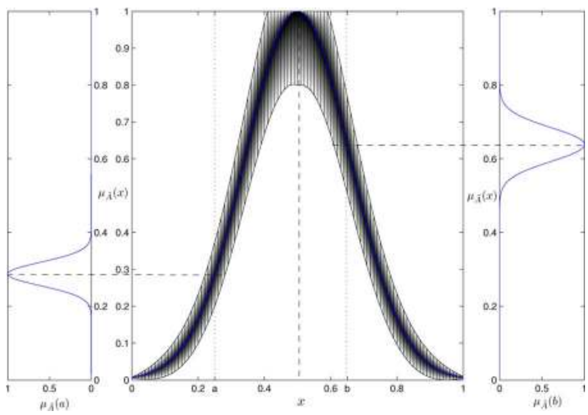


Fig. 1. Type-2 fuzzy set.

Experimental results are presented in Section VI. Finally, a conclusion is given in Section VII.

II. Basic Fuzzy Concepts

A type-2 fuzzy set $\tilde{A}$ on a given universal set X is characterized by the membership function $\mu_{\tilde{A}}(x, u)$, where $x \in X$, and $u \in J_x \subseteq [0, 1]$ and can be represented as [27]

$$\tilde{A} = \{((x, u), \mu_{\tilde{A}}(x, u)) \mid \forall x \in X, \forall u \in J_x\} \tag{1}$$

in which $0 \le \mu_{\tilde{A}}(x, u) \le 1$. We refer to $\mu_{\tilde{A}}(x) = \int_{u \in J_x} \mu_{\tilde{A}}(x, u)/u$ as a secondary membership function which is a type-1 fuzzy set. Obviously, $\tilde{A}$ can also be represented as $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid \forall x \in X\}$. Fig. 1 shows a type-2 fuzzy set where $\mu_{\tilde{A}}(a)$ and $\mu_{\tilde{A}}(b)$ are explicitly shown.

When X is discretized into n points $x_1, x_2, \ldots, x_n$, $\tilde{A}$ becomes

$$\tilde{A} = \sum_{i=1}^{n} \left[\int_{u \in J_{x_i}} \mu_{\tilde{A}}(x_i, u)/u \right] \Big/ x_i. \tag{2}$$

The centroid of $\tilde{A}$ can then be defined as follows [27]:

$$C(\tilde{A}) = \int_{u_1 \in J_{x_1}} \cdots \int_{u_n \in J_{x_n}} [\mu_{\tilde{A}}(x_1, u_1) \star \cdots \star \mu_{\tilde{A}}(x_n, u_n)] \Big/ \frac{\sum_{i=1}^{n} x_i u_i}{\sum_{i=1}^{n} u_i} \tag{3}$$

where $\star$ is the minimum t-norm operator. Note that if $\tilde{A}$ is a type-1 fuzzy set, $C(\tilde{A})$ is a scalar [1]. If $\tilde{A}$ is an interval type-2 fuzzy set, $C(\tilde{A})$ is an interval set [27], [31]–[33]. If $\tilde{A}$ is a type-2 fuzzy set, $C(\tilde{A})$ is a type-1 fuzzy set [26]–[28].

III. Type Reduction

The work of type reduction is to find the centroid type-reduced set, i.e., (4), for a given type-2 fuzzy set $\tilde{A}$. Karnik and Mendel [27] developed an efficient algorithm for the case when $\tilde{A}$ is an interval type-2 fuzzy set. Based on the Karnik–Mendel (KM) algorithm, Liu [26] proposed a method for the case when $\tilde{A}$ is a general type-2 fuzzy set. Both algorithms assume that the universal set is discrete. If the universal set X is continuous, sampling on X is performed before using them.

A. Karnik–Mendel Algorithm

Given an interval type-2 fuzzy set $\tilde{A}$ on a universal set $X = \{x_1, x_2, \ldots, x_n\}$, where $x_1 < x_2 < \cdots < x_n$, let $\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2), \ldots, \mu_{\tilde{A}}(x_n)$ be intervals $[\underline{I}_1, \overline{I}_1], [\underline{I}_2, \overline{I}_2], \ldots, [\underline{I}_n, \overline{I}_n]$, respectively. The KM algorithm [27] can find the centroid type-reduced set, i.e., (4), which is an interval $[\underline{b}, \overline{b}]$, as follows. Initially, compute

$$t = \frac{\sum_{j=1}^{n} x_j((\underline{I}_j + \overline{I}_j)/2)}{\sum_{j=1}^{n} (\underline{I}_j + \overline{I}_j)/2}. \tag{4}$$

Then, set t to $\underline{b}$, which is called the *left initial value*. We locate $\underline{b}$ in X. Let $x_k \le \underline{b} < x_{k+1}$. We set $\underline{L} = k$, which is called the *left switch point*, and update $\underline{b}$ as follows:

$$\underline{b} = \frac{\sum_{j=1}^{\underline{L}} x_j \overline{I}_j + \sum_{j=\underline{L}+1}^{n} x_j \underline{I}_j}{\sum_{j=1}^{\underline{L}} \overline{I}_j + \sum_{j=\underline{L}+1}^{n} \underline{I}_j}. \tag{5}$$

We locate $\underline{b}$ in X again. Let $x_k \le \underline{b} < x_{k+1}$. Then, we update $\underline{L}$ also by $\underline{L} = k$. If $\underline{L}$ has a different value than before, we continue to update $\underline{b}$ by (5). Otherwise, we are done with the computation of $\underline{b}$. Next, we set t to $\overline{b}$, which is called the *right initial value* and locate $\overline{b}$ in X. Let $x_k \le \overline{b} < x_{k+1}$. We set $\overline{L} = k$, which is called the *right switch point*, and update $\overline{b}$ by

$$\overline{b} = \frac{\sum_{j=1}^{\overline{L}} x_j \underline{I}_j + \sum_{j=\overline{L}+1}^{n} x_j \overline{I}_j}{\sum_{j=1}^{\overline{L}} \underline{I}_j + \sum_{j=\overline{L}+1}^{n} \overline{I}_j}. \tag{6}$$

We locate $\overline{b}$ in X again. Let $x_k \le \overline{b} < x_{k+1}$. Then, we update $\overline{L}$ also by $\overline{L} = k$. If $\overline{L}$ has a different value than before, we continue to update $\overline{b}$ by (6). Otherwise, we are done with the computation of $\overline{b}$. The KM algorithm can be summarized below.

procedure Karnik–Mendel($\tilde{A}$, X)
  Initialize $\underline{b}$ and $\overline{b}$ to t computed by (4);
  Determine $\underline{L}$ and $\overline{L}$;
  call KM-In($\tilde{A}$, X, $\underline{b}$, $\overline{b}$, $\underline{L}$, $\overline{L}$);
  return $[\underline{b}, \overline{b}]$, $\underline{L}$, and $\overline{L}$;
endprocedure

procedure KM-In($\tilde{A}$, X, $\underline{b}$, $\overline{b}$, $\underline{L}$, $\overline{L}$)
  do
    $\underline{L}^o$ = $\underline{L}$;
    Update $\underline{b}$ by (5) and $\underline{L}$ accordingly;
  while($\underline{L} \neq \underline{L}^o$);
  do
    $\overline{L}^o$ = $\overline{L}$;
    Update $\overline{b}$ by (6) and $\overline{L}$ accordingly;
  while ($\overline{L} \neq \overline{L}^o$);
endprocedure

Wu and Mendel [31] proposed an enhanced KM (EKM) algorithm in which the initial values of $\underline{L}$ and $\overline{L}$ are estimated as follows:

$$\underline{L} = \frac{n}{2.4}, \qquad \overline{L} = \frac{n}{1.7}. \tag{7}$$

Besides, a better initialization is used to reduce the number of iterations. The termination condition of the iterations is changed to remove one unnecessary iteration and a subtle computing technique is used to reduce the computational cost of each iteration.

B. Liu's Algorithm

Liu [26] proposed a method for type reduction for general type-2 fuzzy sets for the case that the secondary membership functions are convex type-1 fuzzy sets. A type-1 fuzzy set is convex if its every α-cut, i.e., $\alpha > 0$, is a continuous interval. Suppose that we are given such a type-2 fuzzy set $\tilde{A}$ on a universal set $X = \{x_1, x_2, \ldots, x_n\}$, where $x_1 < x_2 < \cdots < x_n$. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be k real numbers, and $\alpha_i \in [0, 1]$, $1 \le i \le k$. For each α_i, we take α_i-cuts of the secondary membership functions of $\tilde{A}$ at $x_1, x_2, \ldots, x_n$ and let them be intervals $[\underline{I}(x_1|\alpha_i), \overline{I}(x_1|\alpha_i)]$, $[\underline{I}(x_2|\alpha_i), \overline{I}(x_2|\alpha_i)], \ldots, [\underline{I}(x_n|\alpha_i), \overline{I}(x_n|\alpha_i)]$. Group these n intervals and we have the α-plane ${}^{\alpha_i}\tilde{A}$ for $\tilde{A}$, which is an interval type-2 fuzzy set on X. By applying the KM algorithm on ${}^{\alpha_i}\tilde{A}$, we get a centroid type reduced interval $[\underline{b}_i, \overline{b}_i]$ for ${}^{\alpha_i}\tilde{A}$. The centroid type-reduced set, i.e., O_y, for $\tilde{A}$ can then be derived as

$$O_y \approx \bigcup_{i=1}^{k} \alpha_i / [\underline{b}_i, \overline{b}_i] \tag{8}$$

by the first decomposition theorem of type-1 fuzzy sets [1]. Let $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_k]$. Liu's algorithm can be summarized below.

procedure Liu($\tilde{A}$, X, $\boldsymbol{\alpha}$)
  for i = 1 **to** k
    Take the α-plane ${}^{\alpha_i}\tilde{A}$ for $\tilde{A}$ on X;
    call Karnik–Mendel(${}^{\alpha_i}\tilde{A}$, X) and obtain the
      centroid type-reduced interval $[\underline{b}_i, \overline{b}_i]$ for ${}^{\alpha_i}\tilde{A}$;
  endfor;
  Obtain the type-reduced set O_y for $\tilde{A}$ by (8);
endprocedure

Some properties for the centroid, including a nesting property, were stated in [34]. Note that we can adopt the EKM algorithm [31] to replace the KM algorithm in the above code. To get a crisp, defuzzified value y for $\tilde{A}$ as output, we can calculate the COG of (8).

IV. Improved Type Reduction Algorithm

In Liu's algorithm, the calculation of the type-reduced interval of each α-plane is done independently. The information obtained in the previous iteration does not help the calculation of the current iteration. We exploit the obtained results of the previous iteration to set the initial values in the current iteration. As a result, efficiency is improved.

A. Improved Algorithm

Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be k real numbers, and $0 \le \alpha_1 < \alpha_2 < \cdots < \alpha_k \le 1$. For simplicity, we denote $\underline{I}(x_j|\alpha_i)$ as $\underline{I}_{j,i}$, and $\overline{I}(x_j|\alpha_i)$ as $\overline{I}_{j,i}$ for $1 \le j \le n$ and $1 \le i \le k$. For example, the α-plane ${}^{\alpha_i}\tilde{A}$ consists of the intervals $[\underline{I}_{1,i}, \overline{I}_{1,i}]$, $[\underline{I}_{2,i}, \overline{I}_{2,i}], \ldots, [\underline{I}_{n,i}, \overline{I}_{n,i}]$. For α_k, we apply the KM algorithm on ${}^{\alpha_k}\tilde{A}$ and obtain the centroid type-reduced interval $[\underline{b}_k, \overline{b}_k]$, the left switch point $\underline{L}_k$, and the right switch point $\overline{L}_k$. For any $\alpha_i, i = k-1, k-2, \ldots, 2, 1$, we exploit $\underline{L}_{i+1}$ and $\overline{L}_{i+1}$, which are the left switch point and the right switch point obtained from the previous iteration for α_{i+1}, to do initial settings for α_i as follows:

$$\underline{L}_i = \underline{L}_{i+1}, \qquad \overline{L}_i = \overline{L}_{i+1} \tag{9}$$

$$\underline{b}_i = \underline{t}_i = \frac{\sum_{j=1}^{\underline{L}_i} x_j \overline{I}_{j,i} + \sum_{j=\underline{L}_i+1}^{n} x_j \underline{I}_{j,i}}{\sum_{j=1}^{\underline{L}_i} \overline{I}_{j,i} + \sum_{j=\underline{L}_i+1}^{n} \underline{I}_{j,i}} \tag{10}$$

$$\overline{b}_i = \overline{t}_i = \frac{\sum_{j=1}^{\overline{L}_i} x_j \underline{I}_{j,i} + \sum_{j=\overline{L}_i+1}^{n} x_j \overline{I}_{j,i}}{\sum_{j=1}^{\overline{L}_i} \underline{I}_{j,i} + \sum_{j=\overline{L}_i+1}^{n} \overline{I}_{j,i}}. \tag{11}$$

Note that, unlike Liu's algorithm, the initial values for $\underline{b}_i$ and $\overline{b}_i$ are set differently and are related to the information derived earlier. Then, we call KM-In(${}^{\alpha_i}\tilde{A}, X, \underline{b}_i, \overline{b}_i, \underline{L}_i, \overline{L}_i$) to obtain the centroid type-reduced interval $[\underline{b}_i, \overline{b}_i]$, the left switch point $\underline{L}_i$, and the right switch point $\overline{L}_i$ for α_i. Let $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_k]$. The improved algorithm can be described below.

procedure Improved($\tilde{A}, X, \boldsymbol{\alpha}$)
  for $i = k$ **down to** 1
    Take α_i-cuts of the secondary membership
    functions of $\tilde{A}$ at $x_1, x_2, \ldots, x_n$;
    Let these α_i-cuts be $[\underline{I}_{1,i}, \overline{I}_{1,i}]$, $[\underline{I}_{2,i}, \overline{I}_{2,i}]$,
    $[\underline{I}_{n,i}, \overline{I}_{n,i}]$;
    Form these α_i-cuts as the α-plane ${}^{\alpha_i}\tilde{A}$ for $\tilde{A}$
    on X;
    if($i = k$)
      call Karnik–Mendel(${}^{\alpha_k}\tilde{A}, X$) and obtain $[\underline{b}_k, \overline{b}_k]$,
        $\underline{L}_k, \overline{L}_k$ for ${}^{\alpha_k}\tilde{A}$;
    else
      Initialize $\underline{L}_i$ and $\overline{L}_i$ by (9);
      Initialize $\underline{b}_i$ and $\overline{b}_i$ by (10) and (11),
      respectively;
      call KM-In(${}^{\alpha_i}\tilde{A}, X, \underline{b}_i, \overline{b}_i, \underline{L}_i, \overline{L}_i$)
        to obtain $[\underline{b}_i, \overline{b}_i], \underline{L}_i, \overline{L}_i$ for ${}^{\alpha_i}\tilde{A}$;
    endif;
  endfor;
    Obtain the type-reduced set O_y for $\tilde{A}$ by (8);
endprocedure

Note that the α-cuts with larger values are processed first.

B. Some Observations

Our proposed improved algorithm has a better computational complexity than Liu's algorithm. We present some observations. The proofs for them can be found in the Appendix at the end of the paper.

Lemma 1: Given a set of data $X = \{x_1, x_2, \ldots, x_n\}$ and nonnegative weights $w_1, w_2, \ldots, w_n$, let c be the centroid of X

defined by

$$c = \frac{\sum_{j=1}^{n} x_j w_j}{\sum_{j=1}^{n} w_j} \tag{12}$$

where $\sum_{j=1}^{n} w_j > 0$. Then, we have $\sum_{j=1}^{n}(x_j - c)w_j = 0$.

Lemma 2: Given a set of data $X = \{x_1, x_2, \ldots, x_n\}$, and nonnegative weights $w_1, w_2, \ldots, w_n$ with $\sum_{j=1}^{n} w_j > 0$, let c be the centroid of X and c' be another number. Then, $\sum_{j=1}^{n}(x_j - c')w_j < 0$ if and only if $c < c'$.

Proposition 1: Given an interval type-2 fuzzy set $\tilde{A}$ on $X = \{x_1, x_2, \ldots, x_n\}$, where $x_1 < x_2 < \cdots < x_n$, with the membership degrees being intervals $[\underline{I}_1, \overline{I}_1], [\underline{I}_2, \overline{I}_2], \ldots, [\underline{I}_n, \overline{I}_n]$, respectively, let the centroid type-reduced interval, the left switch point, and the right switch point obtained by the KM algorithm be $[\underline{b}, \overline{b}]$, $\underline{L}$, and $\overline{L}$, respectively. Then, we have

$$\sum_{j=1}^{\underline{L}}(x_j - \underline{b})\overline{I}_j + \sum_{j=\underline{L}+1}^{n}(x_j - \underline{b})\underline{I}_j = 0 \tag{13}$$

$$\sum_{j=1}^{\overline{L}}(x_j - \overline{b})\underline{I}_j + \sum_{j=\overline{L}+1}^{n}(x_j - \overline{b})\overline{I}_j = 0. \tag{14}$$

Lemma 3: Suppose $\tilde{A}$ is a type-2 fuzzy set on $X = \{x_1, x_2, \ldots, x_n\}$, as described in Section III-B. Let α_1 and α_2 be two real numbers with $0 \le \alpha_1 < \alpha_2 \le 1$. Let the α_i-cuts, $i = 1$ and 2, of the secondary membership functions of $\tilde{A}$ at $x_1, x_2, \ldots, x_n$ be $[\underline{I}_{1,i}, \overline{I}_{1,i}], [\underline{I}_{2,i}, \overline{I}_{2,i}], \ldots, [\underline{I}_{n,i}, \overline{I}_{n,i}]$. Form these two collections of α-cuts as two α-planes ${}^{\alpha_1}\tilde{A}$ and ${}^{\alpha_2}\tilde{A}$, respectively. Let $\underline{L}_1$ and $\overline{L}_1$ be the left and right switch points for ${}^{\alpha_1}\tilde{A}$, and $\underline{L}_2$ and $\overline{L}_2$ for ${}^{\alpha_2}\tilde{A}$, which are obtained by the KM algorithm. Then, we have $\underline{L}_2 \ge \underline{L}_1$, and $\overline{L}_2 \le \overline{L}_1$.

Theorem 1: Suppose $\tilde{A}$ is a type-2 fuzzy set on $X = \{x_1, x_2, \ldots, x_n\}$, as described in Section III-B, and α_1 and α_2 are two real numbers with $0 \le \alpha_1 < \alpha_2 \le 1$. Let the α_i-cuts, $i = 1$ and 2, of the secondary membership functions of $\tilde{A}$ at $x_1, x_2, \ldots, x_n$ be $[\underline{I}_{1,i}, \overline{I}_{1,i}], [\underline{I}_{2,i}, \overline{I}_{2,i}], \ldots, [\underline{I}_{n,i}, \overline{I}_{n,i}]$. Form these two collections of α-cuts as two α-planes ${}^{\alpha_1}\tilde{A}$ and ${}^{\alpha_2}\tilde{A}$, respectively. Let $[\underline{b}_i, \overline{b}_i]$, $\underline{L}_i, \overline{L}_i$ be the centroid type-reduced interval, the left switch point, and the right switch point obtained from the application of the KM algorithm on ${}^{\alpha_i}\tilde{A}, i = 1, 2$. Then, we have

$$\underline{b}_2 \ge \frac{\sum_{j=1}^{\underline{L}_2} x_j \overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} x_j \underline{I}_{j,1}}{\sum_{j=1}^{\underline{L}_2} \overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} \underline{I}_{j,1}} \ge \underline{b}_1 \tag{15}$$

$$\overline{b}_2 \le \frac{\sum_{j=1}^{\overline{L}_2} x_j \underline{I}_{j,1} + \sum_{j=\overline{L}_2+1}^{n} x_j \overline{I}_{j,1}}{\sum_{j=1}^{\overline{L}_2} \underline{I}_{j,1} + \sum_{j=\overline{L}_2+1}^{n} \overline{I}_{j,1}} \le \overline{b}_1. \tag{16}$$

From Lemma 3 and Theorem 1, we can easily obtain the following order relationships involved in our improved algorithm:

$$\underline{b}_1 \le \underline{t}_2 \le \underline{b}_2 \le \underline{t}_3 \le \ldots \le \underline{b}_{k-1} \le \underline{t}_k \le \underline{b}_k \tag{17}$$

$$\overline{b}_1 \ge \overline{t}_2 \ge \overline{b}_2 \ge \overline{t}_3 \ge \ldots \ge \overline{b}_{k-1} \ge \overline{t}_k \ge \overline{b}_k. \tag{18}$$

Note that $\underline{t}_i$ is the left initial value of (10) and that $\overline{t}_i$ is the right initial value of (11) for $i = k-1, k-2, \ldots, 2, 1$.

C. Complexity Comparison

Now, we are ready to give a complexity comparison between Liu's algorithm and our improved algorithm. We are concerned about two numbers, average count and compare count. The average count is the number of initialization, i.e., (4), (10), or (11), and updates, i.e., (5) or (6), to be performed. These five equations can be regarded as doing the average of $x_j, 1 \le j \le n$. For (4), the x_j's are averaged with the weights $(\underline{I}_j + \overline{I}_j)/2, 1 \le j \le n$; for (5), the weights are $\overline{I}_1, \overline{I}_2, \ldots, \overline{I}_{\underline{L}}, \underline{I}_{\underline{L}+1}, \ldots, \underline{I}_n$, respectively, etc. The compare count is the number of comparisons with $x_i, 1 \le i \le n$, to be done. Furthermore, we consider worst cases. Note that in a call to the KM algorithm, one needs to compare with x_1 up to find the initial switch point. Then, search is done up and down from this point. Liu's algorithm has to do this procedure k times. Instead, in our algorithm, we search for $\underline{L}_i$ down from $\underline{L}_{i+1}$, and for $\overline{L}_i$ up from $\overline{L}_{i+1}$.

For Liu's algorithm, one call to the KM algorithm is performed for each $\alpha_i, 1 \le i \le k$. In each call of the KM algorithm, we have to apply (4) once, and advance down iteratively to find $\underline{b}_i$ by applying (5), possibly traversing through each interval of the lower part of the universe set, and then advance up iteratively to find $\overline{b}_i$ by applying (6), possibly traversing through each interval of the upper part of the universe set. Note that (5) and (6) need to be executed twice at the final switch point. Therefore, the average count for one application of the KM algorithm is $n - 1 + 2 = n + 1$ and the total average count involved in the Liu's algorithm in the worst case is $k(n + 1)$, which is of order $O(kn)$. In each call of the KM algorithm, at most n comparisons with $x_j, 1 \le j \le n$, may be done in finding the initial value of $\underline{L}_i$. Then at most $2n$ comparisons are required to obtain the left switch point $\underline{L}_i$ and the right switch point $\overline{L}_i$. The factor 2 comes from the possibility of two comparisons done with each x_j. Therefore, one call to the KM algorithm requires at most $n + 2n = 3n$ in compare count. The total compare count involved in the Liu's algorithm is bounded by $3kn$, which is also of order $O(kn)$.

For our improved algorithm, we do the same procedure as Liu's for α_k. Therefore, the average count and compare count for α_k are $n + 1$ and $3n$, respectively. Then, we use $\underline{L}_{i+1}$ and $\overline{L}_{i+1}$ to derive $[\underline{b}_i, \overline{b}_i], i = k-1, k-2, \ldots, 2, 1$. For initialization, we apply (10) and (11). This takes $2(k-1)$ in average count. As α decreases from α_{k-1} to α_1, the updates go all the way down or up from the switch points obtained from the previous iteration. The worst case occurs when updates traverse through each interval of the universal set. This leads to $n - 1$ computations. Also, both (5) and (6) need to be executed twice at the final switch point. This leads to another $2k$ computations. Therefore, the total average count involved in our improved algorithm in the

TABLE I
SUMMARY ON AVERAGE AND COMPARE COUNTS IN WORST CASES

	Liu's	Ours
average count	$k(n+1)$	$4k+2n-2$
compare count	$3kn$	$5n$

worst case is $n+1+2(k-1)+n-1+2k = 4k+2n-2$, which is of order $O(k+n)$. Regarding comparisons, we need at most $2n$ comparisons for $\alpha_i, i = k-1, k-2, \ldots, 2, 1$. Again, the factor 2 comes from the possibility of two comparisons done with each x_j. Therefore, the compare count involved in our improved algorithm is bounded by $3n + 2n = 5n$, which is of order $O(n)$. Note that the complexity of Liu's algorithm is of quadratic order, while the complexity of our improved algorithm is of linear order. A summary about the average and compare counts in worst cases is listed in Table I.

V. Example

We give an example here to illustrate how the improved algorithm differs from Liu's algorithm. Let $\tilde{A}$ be a type-2 fuzzy set. Let us assume that after sampling, we have $X = \{x_1, x_2, \ldots, x_6\}$, where $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4, x_5 = 5$, and $x_6 = 6$. The secondary membership functions of $\tilde{A}$ at these sampled points are shown in Fig 2. Suppose that we take $k = 3$, and let $\alpha_1 = 0.1, \alpha_2 = 0.5$, and $\alpha_3 = 1.0$. We take α_3-cuts of the secondary membership functions and have $[\underline{I}_{1,3}, \overline{I}_{1,3}] = [0.3, 0.7]$, $[\underline{I}_{2,3}, \overline{I}_{2,3}] = [0.4, 0.8]$, $[\underline{I}_{3,3}, \overline{I}_{3,3}] = [0.4, 0.8]$, $[\underline{I}_{4,3}, \overline{I}_{4,3}] = [0.5, 0.9]$, $[\underline{I}_{5,3}, \overline{I}_{5,3}] = [0.4, 0.8]$, and $[\underline{I}_{6,3}, \overline{I}_{6,3}] = [0.3, 0.7]$. These six intervals form the interval type-2 fuzzy set ${}^{\alpha_3}\tilde{A}$. Similarly, we have ${}^{\alpha_2}\tilde{A}$ containing the six intervals $[\underline{I}_{1,2}, \overline{I}_{1,2}] = [0.1, 0.9]$, $[\underline{I}_{2,2}, \overline{I}_{2,2}] = [0.2, 0.8]$, $[\underline{I}_{3,2}, \overline{I}_{3,2}] = [0.3, 1.0]$, $[\underline{I}_{4,2}, \overline{I}_{4,2}] = [0.4, 1.0]$, $[\underline{I}_{5,2}, \overline{I}_{5,2}] = [0.3, 0.9]$, $[\underline{I}_{6,2}, \overline{I}_{6,2}] = [0.1, 0.8]$, and ${}^{\alpha_1}\tilde{A}$ containing the six intervals $[\underline{I}_{1,1}, \overline{I}_{1,1}] = [0.0, 1.0]$, $[\underline{I}_{2,1}, \overline{I}_{2,1}] = [0.0, 1.0]$, $[\underline{I}_{3,1}, \overline{I}_{3,1}] = [0.1, 1.0]$, $[\underline{I}_{4,1}, \overline{I}_{4,1}] = [0.2, 1.0]$, $[\underline{I}_{5,1}, \overline{I}_{5,1}] = [0.0, 1.0]$, $[\underline{I}_{6,1}, \overline{I}_{6,1}] = [0.0, 1.0]$.

Liu's algorithm works as follows.

1) For $\alpha_3 = 1.0$, we apply the KM algorithm on ${}^{\alpha_3}\tilde{A}$. First, we initialize $\underline{b}_3$ and $\overline{b}_3$ by (4) and have $\underline{b}_3 = \overline{b}_3 = t = 3.514$. Since $x_3 \leq 3.514 \leq x_4$, we have $\underline{L}_3 = 3$. For this, the average count is 1, and the compare count is 3. Note that we compared 3.514 against x_1, x_2, and x_3. Then, we update $\underline{b}_3$ by (5) and have $\underline{b}_3 = 3.000$, from which we have $\underline{L}_3 = 3$. This time the average count is 1, and the compare count is 1 by comparing 3.000 against x_3. Note that the value of $\underline{L}_3$ does not change. Therefore, we are done with $\underline{b}_3$. The average count and compare count for obtaining $\underline{b}_3$ are $1 + 1 = 2$ and $3 + 1 = 4$, respectively. Now we proceed with $\overline{b}_3$. Obviously, the initial values of $\overline{b}_3$ and $\overline{L}_3$ are 3.514 and 3, respectively. We update $\overline{b}_3$ by (6), and have $\overline{b}_3 = 4.029$. Since $x_4 \leq 4.029 \leq x_5$, we have $\overline{L}_3 = 4$. For this, the average count is 1 and the compare count is 3. Note that we compared 4.029 against x_3 and x_4. The value of $\overline{L}_3$ changes; therefore, we update it again by (6) and have $\overline{b}_3 = 4.032$, from which we have $\overline{L}_3 = 4$. This time the average count is 1, and the compare count is 1 by comparing 4.032 against x_4. Note that the value of $\overline{L}_3$ does not change. Therefore, we are done with $\overline{b}_3$. The average count and compare count for obtaining $\overline{b}_3$ are $1 + 1 = 2$ and $2 + 1 = 3$, respectively. This completes the iteration for α_3. In summary, we obtain $[\underline{b}_3, \overline{b}_3] = [3.000, 4.032], \underline{L}_3 = 2, \overline{L}_3 = 4$, average count $= 2 + 2 = 4$, and compare count $= 4 + 3 = 7$.
2) For $\alpha_2 = 0.5$, we apply the KM algorithm on ${}^{\alpha_2}\tilde{A}$. By repeating the same procedure as above, we obtain $[\underline{b}_2, \overline{b}_2] = [2.536, 4.556], \underline{L}_2 = 2, \overline{L}_2 = 4$, average count $= 5$, and compare count $= 9$.
3) For $\alpha_1 = 0.1$, we apply the KM algorithm on ${}^{\alpha_1}\tilde{A}$. By repeating the same procedure as above, we obtain $[\underline{b}_1, \overline{b}_1] = [1.615, 5.462], \underline{L}_1 = 1, \overline{L}_1 = 5$, average count $= 7$, and compare count $= 13$.

The results obtained from Liu's algorithm are shown in Table II. The centroid type-reduced set is depicted in Fig. 3. The corresponding crisp, defuzzified output is 3.535.

Our improved algorithm works as follows.

1) For $\alpha_3 = 1.0$, our algorithm runs exactly as Liu's algorithm. Therefore, $[\underline{b}_3, \overline{b}_3] = [3.000, 4.032], \underline{L}_3 = 3, \overline{L}_3 = 4$, average count $= 4$, and compare count $= 7$.
2) For $\alpha_2 = 0.5$, we exploit $\underline{L}_3$ to obtain $\underline{b}_2$. First, we set $\underline{L}_2 = \underline{L}_3 = 3$, and calculate the initial value of $\underline{b}_2$ by (10), and have $\underline{b}_2 = \underline{t}_2 = 2.629$. Since $x_2 \leq 2.629 \leq x_3$, we have $\underline{L}_3 = 2$. For this, the average count is 1 and the compare count is 2. Note that we compared 2.629 against x_3 and x_2. The value of $\underline{L}_3$ changes, and therefore, we update it by (5) and have $\underline{b}_2 = 2.536$, from which we have $\underline{L}_2 = 2$. For this, the average count is 1 and the compare count is 1. Note that we compared 2.536 against x_2. Since the value of $\underline{L}_2$ does not change, we are done with $\underline{b}_2$. The average count and compare count for obtaining $\underline{b}_2$ are 2 and 3, respectively. Now, we proceed with $\overline{b}_2$. Again, we exploit $\overline{L}_3$ to obtain $\overline{b}_2$. First, we set $\overline{L}_2 = \overline{L}_3 = 4$, and calculate the initial value of $\overline{b}_2$ by (11) and have $\overline{b}_2 = 4.556$, from which we have $\overline{L}_2 = 4$. For this, the average count is 1 and the compare count is 1. Note that we compared 4.556 against x_4. Since the value of $\overline{L}_2$ does not change, we are done with $\overline{b}_2$. The average count and compare count for obtaining $\overline{b}_2$ are 1 and 1, respectively. In summary, we obtain $[\underline{b}_2, \overline{b}_2] = [2.536, 4.556], \underline{L}_2 = 2, \overline{L}_2 = 4$, average count $= 2 + 1 = 3$, and compare count $= 3 + 1 = 4$.
3) For $\alpha_1 = 0.1$, we exploit $[\underline{L}_2, \overline{L}_2]$ to obtain $[\underline{b}_1, \underline{b}_1]$. Repeating the same procedure as above, we obtain $[\underline{b}_1, \overline{b}_1] = [1.615, 5.462], \underline{L}_1 = 1, \overline{L}_1 = 5$, average count $= 4$, and compare count $= 6$.

The results obtained from our algorithm are also shown in Table II. Clearly, our algorithm runs more efficiently than Liu's algorithm. Both average count and compare count of our algorithm are smaller than those of Liu's algorithm.

VI. Experimental Results

In this section, we present experimental results to show the effectiveness of our improved algorithm. We compare our

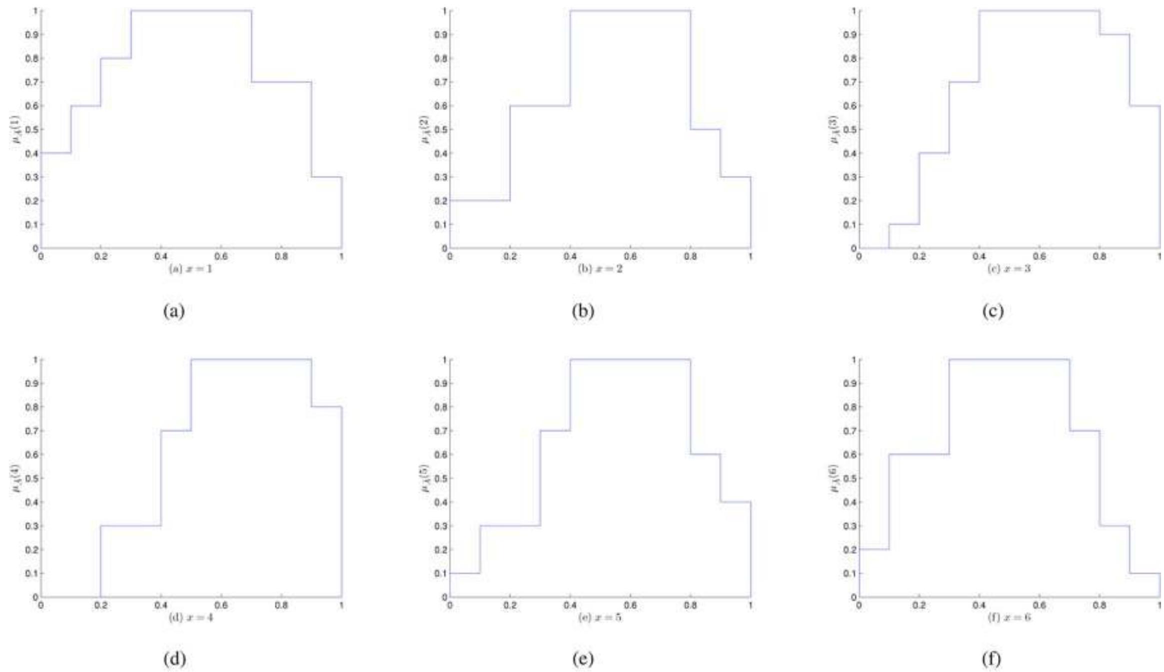


Fig. 2. Secondary membership functions of $\tilde{A}$ at sampled points.

TABLE II
RESULTS OBTAINED FOR THE EXAMPLE

	Liu's			Ours		
	$[\underline{b}_i,\ \overline{b}_i]$	average count	compare count	$[\underline{b}_i,\ \overline{b}_i]$	average count	compare count
$\alpha_3 = 1.0$	$[3.000, 4.032]$	4	7	$[3.000, 4.032]$	4	7
$\alpha_2 = 0.5$	$[2.536.4.556]$	5	9	$[2.536.4.556]$	3	4
$\alpha_1 = 0.1$	$[1.615, 5.462]$	7	13	$[1.615, 5.462]$	4	6
total		16	29		11	17

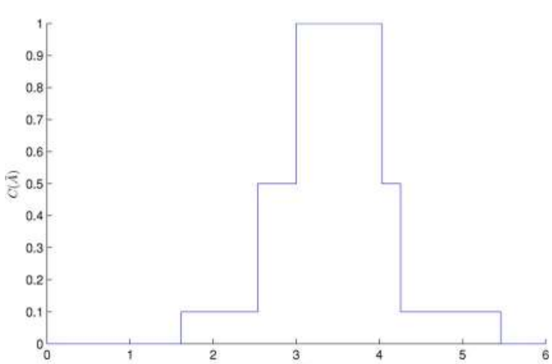


Fig. 3. Centroid type-reduced set of the example.

algorithm with four versions of Liu's algorithm described in Section III-B. In the first version, which was denoted as Liu-original-KM, and was originally published in [26], locating the appropriate range of $[x_k, x_{k+1}]$ for each update of $\underline{b}_i$ or $\overline{b}_i$ is always done from x_1 up, i.e., comparing with x_1, x_2, etc. In the second version, which was denoted as Liu-KM, we modified the version of Liu-original-KM such that in each call of the KM algorithm one compares with x_1 up to find the initial switch point, and then search up for $\overline{b}_i$ or down for $\underline{b}_i$ from this initial point, as described in Section IV-C. The third version, which is denoted as Liu-original-EKM, is the same as the version of Liu-original-KM except the EKM algorithm [31], estimating initial switch points the way as in (7), was adopted. The fourth version, which is denoted as Liu-EKM, is the same as the version of Liu-KM except the EKM algorithm was adopted. As can be seen later, Liu-original-KM and Liu-KM require the same numbers in average count, and Liu-original-EKM and Liu-EKM require the same numbers in average count. However, Liu-KM and Liu-EKM save a lot of comparisons, and thus, can run faster than Liu-original-KM and Liu-original-EKM. Our improved algorithm requires fewer average and compare counts, and runs faster than all these four versions of Liu's algorithm. In the following experiments, we use a computer with Intel(R) Core(TM)2 Quad CPU Q6600 2.40 GHz, 4 GB of RAM to conduct the experiments. The programming language used is MATLAB7.0.

A. Experiment I

In this experiment, we do type reduction for a type-2 fuzzy set $\tilde{A}$ with the following secondary membership function [26]:

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{z - d(x)}{p(x) - d(x)}, & d(x) \le z \le p(x) \\ \dfrac{u(x) - z}{u(x) - p(x)}, & p(x) \le z \le u(x) \\ 0, & \text{otherwise} \end{cases}, \quad 0 \le z \le 1$$

$$u(x) = \max(f_1(x), f_2(x))$$

$$d(x) = \max(g_1(x), g_2(x))$$

$$f_1(x) = \exp\left(-\frac{(x-3)^2}{8}\right)$$

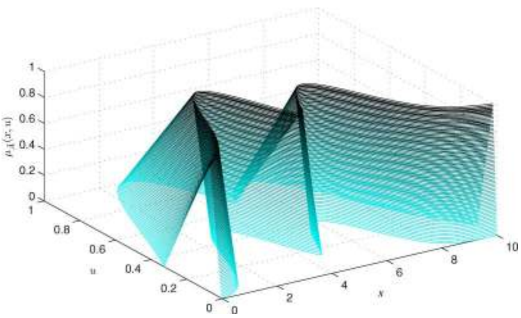


Fig. 4. $\tilde{A}$ for experiment I.

$$f_2(x) = 0.8 \exp\left(-\frac{(x-6)^2}{8}\right)$$

$$g_1(x) = 0.5 \exp\left(-\frac{(x-3)^2}{2}\right)$$

$$g_2(x) = 0.4 \exp\left(-\frac{(x-6)^2}{2}\right)$$

$$p(x) = d(x) + w(x)(u(x) - d(x))$$

where $x \in R$, and $w(x)$ is a randomly selected number between 0 and 1. A 3-D figure of this fuzzy set is shown in Fig. 4.

Fig. 5 shows comparisons on average count, compare count, and execution time taken by each of the five algorithms in deriving the centroid type-reduced set for $\tilde{A}$. In this figure, the number of samplings is fixed at 200. The samplings are taken evenly in the range $[0, 10]$, i.e., $x_1 = 0.05, x_2 = 0.10, \ldots, x_{200} = 10.00$. The horizontal axis indicates the number of α-cuts, k, which varies from 5 to 100. For each $k, 5 \leq k \leq 100$, the α-cuts are taken evenly. For example, for $k = 10$, we have $\alpha_1 = 0.1, \alpha_2 = 0.2, \ldots, \alpha_{10} = 1.0$. Note that in Fig. 5(a), the vertical axis is plotted in log scale with base 10. In Fig. 5(b), we only have three curves since Liu-original-KM and Liu-KM have identical values in average count and Liu-original-EKM and Liu-EKM have identical values in average count. From Fig. 5, we can see that the compare count involved in our algorithm is around 100 to 300. However, the four Liu's algorithms have a larger compare count of about 500–40 000. Note that Liu-EKM requires more compare counts than Liu-KM. For example, Liu-EKM requires about 4000 in compare count for $k = 25$, but Liu-KM only requires about 2000 in this case. The reason is that the EKM algorithm estimates two different initial switch points in each iteration. A comparison with x_1 up has to be done for both switch points. For the KM algorithm, identical initial switch points are estimated and one such comparison can be saved in each iteration. The average count involved in our algorithm is around 20–200. However, the four Liu's algorithms have a larger average count, Liu-KM around 30–500 and Liu-EKM around 20–400. Our algorithm runs about three to five times faster than Liu-KM and Liu-EKM and about six–15 times faster than Liu-original-KM and Liu-original-EKM. Liu-original-EKM runs faster than Liu-original-KM, and Liu-EKM runs faster than Liu-KM. Note that an average count is more computation-demanding than a compare count. Therefore, execution time heavily depends on

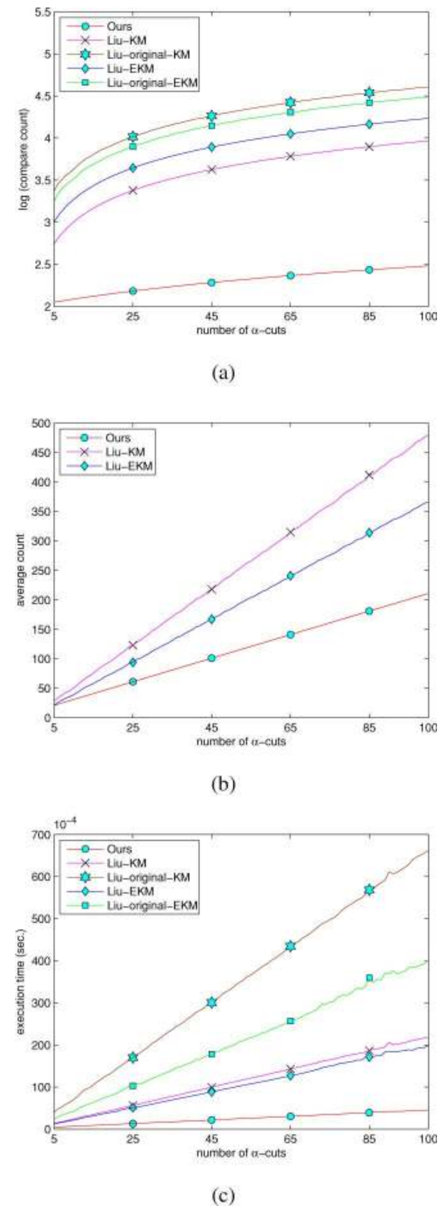


Fig. 5. Comparisons on (a) compare count, (b) average count, and (c) execution time for experiment I, with the number of samplings being 200.

average count. The values at certain snapshots of Fig. 5 are shown in Table III. The defuzzified values obtained for different α-cuts are shown in the last row in this table. To rule out random effects in comparisons, paired samples t-tests were conducted for two pairs of competitors, Ours versus Liu-KM and Ours versus Liu-EKM. The p-values of the tests are shown in Table IV. Note that the lower the p-value, the more significant the result in the sense of statistical significance. In this case, if the p-value is less than 0.05, then there is a significant difference between the two competitors. Apparently, our algorithm is significantly better than Liu-KM and Liu-EKM in terms of compare count, average count, and execution time.

Fig. 6 shows comparison results with k, the number of α-cuts, being fixed at 20, i.e., we have $\alpha_1 = 0.05, \alpha_2 = 0.10, \ldots, \alpha_{20} = 1.00$. The horizontal axis indicates the number

TABLE III
VALUES AT CERTAIN SNAPSHOTS FOR FIG. 5

number of samplings = 200						
number of α-cuts		5	25	50	75	100
compare count	Ours	112	152	202	252	302
	Liu-KM	551	2386	4679	6973	9268
	Liu-original-KM	2354	10346	20352	30356	40366
	Liu-EKM	1018	4417	8663	12908	17155
	Liu-original-EKM	1760	7892	15528	23162	30715
average count	Ours	21	61	111	161	211
	Liu-KM	28	123	242	361	480
	Liu-EKM	21	94	185	276	366
execution time	Ours	4.565	12.996	23.920	34.706	45.311
	Liu-KM	13.501	56.261	110.521	164.587	218.228
	Liu-original-KM	39.896	170.022	333.706	497.884	662.103
	Liu-EKM	12.037	51.276	98.022	150.573	199.820
	Liu-original-EKM	23.086	102.733	197.395	302.153	401.891
defuzzified value		4.392	4.395	4.395	4.395	4.395

TABLE IV
p-VALUES OBTAINED FOR EXPERIMENT I, WITH THE NUMBER OF SAMPLINGS BEING 200

	Ours vs. Liu-KM	Ours vs. Liu-EKM
compare count	4.013×10^{-33}	1.909×10^{-33}
average count	1.107×10^{-31}	2.003×10^{-31}
execution time	2.334×10^{-33}	2.967×10^{-33}

of samplings, varying from 5 to 200. Samplings are done evenly in the range $[0, 10]$. From this figure, we can see that the compare count involved in our algorithm is around 50–150. However, the four Liu's algorithms have a larger compare count, which is about 100–8000. The average count involved in our algorithm is around 40–50. However, the four Liu's algorithms have a larger average count, Liu-KM around 70–100 and Liu-EKM around 50–80. Our algorithm runs about two to four times faster than Liu-KM and Liu-EKM, and about two to 12 times faster than Liu-original-KM and Liu-original-EKM. The values at certain snapshots of Fig. 6 are shown in Table V. The defuzzified values obtained for different samplings are shown in the last row in this table. The p-values of paired samples t-tests for the two pairs, i.e., Ours versus Liu-KM and Ours versus Liu-EKM, are shown in Table VI. Apparently, our algorithm is significantly better than Liu-KM and Liu-EKM in terms of compare count, average count, and execution time.

Two centroid type-reduced sets, i.e., one for $k = 20$, and the other for $k = 80$, which are derived for $\tilde{A}$ are shown in Fig. 7, with the number of samplings being 200. Note that with $k = 20$, we have a zigzag curve for the set. However, when the resolution is set higher with $k = 80$, the derived set is much smoother.

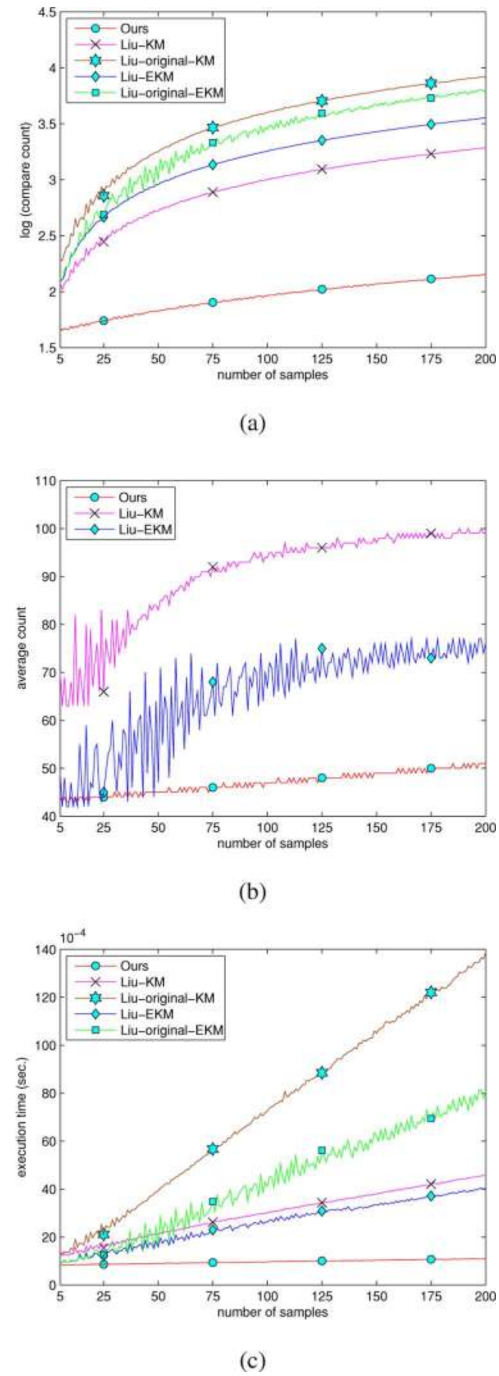


Fig. 6. Comparisons on (a) compare count, (b) average count, and (c) execution time for experiment I, with k being 20.

B. Experiment II

In this experiment, we do type-reduction for another type-2 fuzzy set $\tilde{A}$ with the following secondary membership function [26]:

$$\mu_{\tilde{A}}(x) = \begin{cases} 1, & d(x) \le z \le p(x) \\ \dfrac{u(x) - z}{u(x) - p(x)}, & p(x) \le z \le u(x) \\ 0, & \text{otherwise} \end{cases}, \qquad 0 \le z \le 1$$

$$u(x) = \max(f_1(x), f_2(x))$$

$$d(x) = \max(g_1(x), g_2(x))$$

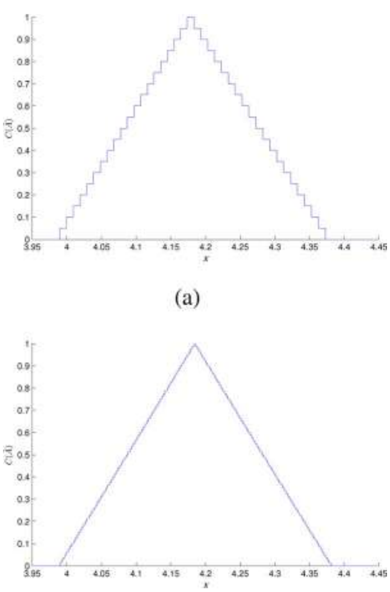
$$f_1(x) = \begin{cases} \dfrac{x-1}{2}, & 1 \le x \le 3 \\ \dfrac{7-x}{4}, & 3 \le x \le 7 \\ 0, & \text{otherwise} \end{cases}$$

TABLE V
VALUES AT CERTAIN SNAPSHOTS FOR FIG. 6

number of α-cuts = 20						
number of samplings		5	50	100	150	200
compare count	Ours	47	67	93	117	142
	Liu-KM	119	523	1020	1461	1927
	Liu-original-KM	196	1775	4095	6127	8326
	Liu-EKM	126	903	1840	2686	3567
	Liu-original-EKM	133	1001	3049	4663	6379
average count	Ours	44	45	47	49	51
	Liu-KM	70	83	95	97	99
	Liu-EKM	49	47	71	74	76
execution time	Ours	8.501	9.023	9.730	10.352	11.011
	Liu-KM	13.285	21.238	30.606	37.951	45.742
	Liu-original-KM	13.915	39.067	74.328	105.013	138.614
	Liu-EKM	10.183	15.809	27.285	33.499	40.398
	Liu-original-EKM	10.291	17.212	44.940	61.893	80.771
defuzzified value		4.363	4.389	4.392	4.394	4.394

TABLE VI
p-VALUES OBTAINED FOR EXPERIMENT I, WITH k BEING 20

	Ours vs. Liu-KM	Ours vs. Liu-EKM
compare count	5.480×10^{-65}	1.348×10^{-63}
average count	3.786×10^{-146}	2.295×10^{-142}
execution time	5.780×10^{-140}	5.234×10^{-103}



(a)

(b)

Fig. 7. Centroid type-reduced set derived with (a) $k = 20$ and (b) $k = 80$ for experiment I.

$$f_2(x) = \begin{cases} \dfrac{x-2}{5}, & 2 \le x \le 6 \\ \dfrac{16-2x}{5}, & 6 \le x \le 8 \\ 0, & \text{otherwise} \end{cases}$$

$$g_1(x) = \begin{cases} \dfrac{x-1}{6}, & 1 \le x \le 4 \\ \dfrac{7-x}{6}, & 4 \le x \le 7 \\ 0, & \text{otherwise} \end{cases}$$

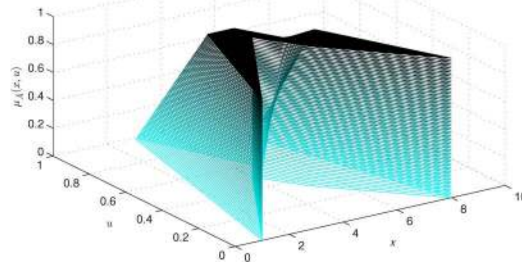


Fig. 8. $\tilde{A}$ for experiment II.

TABLE VII
VALUES AT CERTAIN SNAPSHOTS FOR FIG. 9

number of samplings = 200						
number of α-cuts		5	25	50	75	100
compare count	Ours	136	179	229	279	329
	Liu-KM	668	2908	5714	8509	11308
	Liu-original-KM	2701	11950	23932	35210	46823
	Liu-EKM	1055	4571	8979	13371	17772
	Liu-original-EKM	2180	9464	19013	28073	37253
average count	Ours	25	68	118	168	218
	Liu-KM	31	138	276	406	540
	Liu-EKM	25	109	219	323	429
execution time	Ours	5.419	14.586	25.330	36.238	47.925
	Liu-KM	14.914	64.188	126.700	188.445	250.385
	Liu-original-KM	44.324	194.739	390.360	575.055	763.224
	Liu-EKM	12.289	51.834	101.821	154.675	204.320
	Liu-original-EKM	28.669	122.353	245.413	373.266	489.256
defuzzified value		4.311	4.314	4.314	4.314	4.314

$$g_2(x) = \begin{cases} \dfrac{x-3}{6}, & 3 \le x \le 5 \\ \dfrac{8-x}{9}, & 5 \le x \le 8 \\ 0, & \text{otherwise} \end{cases}$$

$$p(x) = u(x) - 0.6(u(x) - d(x))$$

where $x \in R$. A 3-D figure of this fuzzy set is shown in Fig. 8. Fig. 9 shows comparisons on average count, compare count, and execution time taken by each of the five algorithms in deriving the centroid type-reduced set for $\tilde{A}$. In this figure, the number of samplings is fixed at 200. The samplings are taken evenly in the range $[0, 10]$. The horizontal axis indicates the number of α-cuts, i.e., k, which varies from 5 to 100. For each $k, 5 \le k \le 100$, the α-cuts are taken evenly. From this figure, we can see that the compare count involved in our algorithm is around 150–350. However, the four Liu's algorithms have a larger compare count, which is about 700–50 000. The average count involved in our algorithm is around 20–200. However, the four Liu's algorithms have a larger average count, Liu-KM around 30–550 and Liu-EKM around 30–450. Our algorithm runs about three to five times faster than Liu-KM and Liu-EKM, and about six–15 times faster than Liu-original-KM and Liu-original-EKM. The values at certain snapshots of Fig. 9 are shown in Table VII. The defuzzified values obtained for different α-cuts are shown in the last row in this table. The p-values of paired samples t-tests for the two pairs, Ours versus Liu-KM and Ours versus Liu-EKM, are shown in Table VIII. Apparently, our algorithm

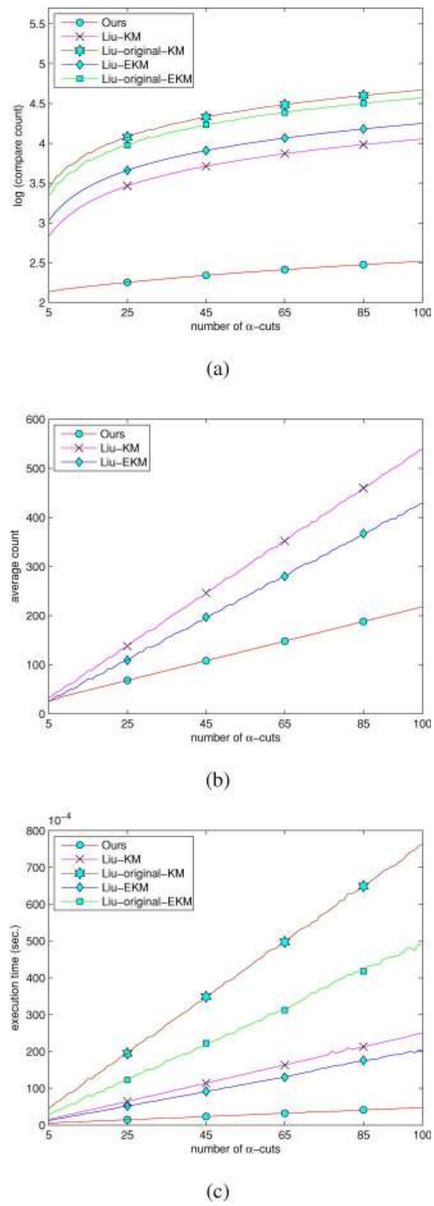


Fig. 9. Comparisons on (a) compare count, (b) average count, and (c) execution time for experiment II, with the number of samplings being 200.

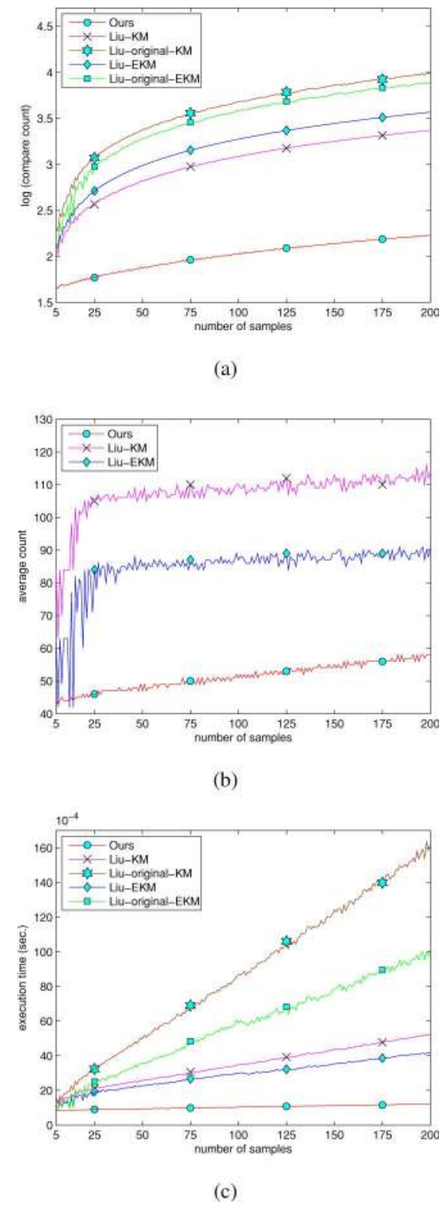


Fig. 10. Comparisons on (a) compare count, (b) average count, and (c) execution time for experiment II, with k being 20.

TABLE VIII
p-VALUES OBTAINED FOR EXPERIMENT II, WITH THE NUMBER OF SAMPLINGS BEING 200

	Ours vs. Liu-KM	Ours vs. Liu-EKM
compare count	4.352×10^{-33}	2.302×10^{-33}
average count	8.131×10^{-31}	1.297×10^{-30}
execution time	7.734×10^{-33}	7.832×10^{-33}

is significantly better than Liu-KM and Liu-EKM in terms of compare count, average count, and execution time.

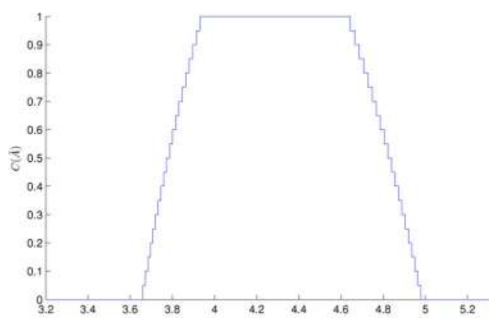
Fig. 10 shows comparison results with k, the number of α-cuts, being fixed at 20. The horizontal axis indicates the number of samplings, varying from 5 to 200. Samplings are done evenly in the range $[0, 10]$. From this figure, we can see that the compare count involved in our algorithm is around 50–200. However, the four Liu's algorithm have a larger compare count, about 150–10 000. The average count involved in our algorithm is around 40–60. However, the four Liu's algorithms have a larger average count, Liu-KM around 80–120 and Liu-EKM around 60–90. Our algorithm runs about two to four times faster than Liu-KM and Liu-EKM and about two to 14 times faster than Liu-original-KM and Liu-original-EKM. The values at certain snapshots of Fig. 10 are shown in Table IX. The defuzzified values obtained for different samplings are shown in the last row in this table. The p-values of paired samples t-tests for the two pairs, i.e., Ours versus Liu-KM and Ours versus Liu-EKM, are shown in Table X. Apparently, our algorithm is significantly better than Liu-KM and Liu-EKM in terms of compare count, average count, and execution time.

TABLE IX
VALUES AT CERTAIN SNAPSHOTS FOR FIG. 10

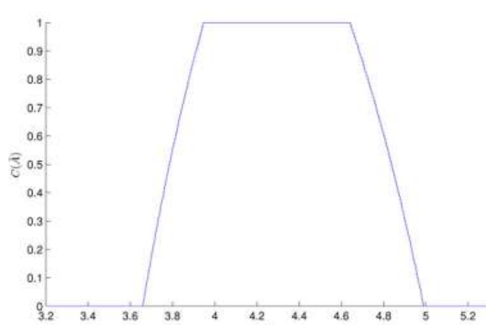
number of α-cuts = 20						
number of samplings		5	50	100	150	200
compare count	Ours	47	74	106	139	169
	Liu-KM	147	655	1228	1784	2352
	Liu-original-KM	210	2338	4798	7347	9883
	Liu-EKM	126	969	1879	2788	3698
	Liu-original-EKM	147	1876	3799	5806	7905
average count	Ours	44	47	51	55	58
	Liu-KM	84	106	110	113	114
	Liu-EKM	63	85	87	89	91
execution time	Ours	8.440	9.285	10.399	11.331	12.259
	Liu-KM	15.099	25.658	34.965	43.833	52.605
	Liu-original-KM	15.649	49.806	86.686	124.363	162.225
	Liu-EKM	12.035	22.719	29.931	35.387	42.051
	Liu-original-EKM	12.274	35.141	58.988	78.582	101.944
defuzzified value		4.196	4.314	4.314	4.314	4.314

TABLE X
p-VALUES OBTAINED FOR EXPERIMENT II, WITH k BEING 20

	Ours vs. Liu-KM	Ours vs. Liu-EKM
compare count	3.960×10^{-66}	4.375×10^{-64}
average count	1.463×10^{-206}	8.995×10^{-160}
execution time	2.554×10^{-153}	3.233×10^{-109}



(a)



(b)

Fig. 11. Derived type-reduced set with (a) $k = 20$ and (b) $k = 80$ for experiment II.

Two centroid type-reduced sets, i.e., one for $k = 20$ and the other for $k = 80$, derived for $\tilde{A}$ are shown in Fig. 11, with the number of samplings being 200. Again, the curve with $k = 80$ is much smoother than the curve with $k = 20$.

VII. CONCLUSION

We have presented an improvement to Liu's algorithm [26] to derive the centroid type-reduced set for a type-2 fuzzy set. In Liu's algorithm, a type-2 fuzzy set is decomposed, by α-cuts, into a collection of interval type-2 fuzzy sets, and then the, KM algorithm [27] is applied to do type reduction for each interval type-2 fuzzy set. However, the initialization of the switch point in each application of the KM algorithm is not good, thereby leading to unnecessary computations and comparisons. In our improved algorithm, the result previously obtained is employed to construct the starting values in the current application of the KM algorithm. As a result, average and compare counts are reduced, and convergence in each iteration except the first one speeds up. Through mathematical analysis and experiments, we have concluded the superiority of our improved algorithm over Liu's algorithm.

APPENDIX

Proof of Lemma 1: Note that

$$c = \frac{\sum_{j=1}^{n} x_j w_j}{\sum_{j=1}^{n} w_j}$$

$$\Rightarrow \sum_{j=1}^{n} x_j w_j = c \sum_{j=1}^{n} w_j$$

$$\Rightarrow \sum_{j=1}^{n} (x_j - c) w_j = 0$$

which is the desired result.

Proof of Lemma 2: Note that from Lemma 1, we have

$$\begin{aligned}\sum_{j=1}^{n}(x_j - c')w_j &= \sum_{j=1}^{n}(x_j - c')w_j - \sum_{j=1}^{n}(x_j - c)w_j \\ &= \sum_{j=1}^{n}[(x_j - c') - (x_j - c)]w_j \\ &= \sum_{j=1}^{n}(c - c')w_j \\ &= (c - c')\sum_{j=1}^{n} w_j.\end{aligned}$$

Since $\sum_{j=1}^{n} w_j > 0$, we conclude that $\sum_{j=1}^{n}(x_j - c')w_j < 0$ if and only if $c < c'$.

Proof of Proposition 1: Let us refer to the KM algorithm described in Section III. For $\underline{b}$, when convergence occurs, we have

$$\underline{b} = \frac{\sum_{j=1}^{\underline{L}} x_j \overline{I}_j + \sum_{j=\underline{L}+1}^{n} x_j \underline{I}_j}{\sum_{j=1}^{\underline{L}} \overline{I}_j + \sum_{j=\underline{L}+1}^{n} \underline{I}_j}$$

from (5). Let $w_1 = \overline{I}_1, \ldots, w_{\underline{L}} = \overline{I}_{\underline{L}}, w_{\underline{L}+1} = \underline{I}_{\underline{L}+1}, \ldots, w_n = \underline{I}_n$. From Lemma 1, we get (13). For $\overline{b}$, when convergence occurs, by (6), we have

$$\overline{b} = \frac{\sum_{j=1}^{\overline{L}} x_j \underline{I}_j + \sum_{j=\overline{L}+1}^{n} x_j \overline{I}_j}{\sum_{j=1}^{\overline{L}} \underline{I}_j + \sum_{j=\overline{L}+1}^{n} \overline{I}_j}$$

which can be shown to be identical to (14) in a similar way.

Proof of Lemma 3: Let $[\underline{b}_1, \overline{b}_1]$ and $[\underline{b}_2, \overline{b}_2]$ be the centroid type-reduced intervals obtained for ${}^{\alpha_1}\tilde{A}$ and ${}^{\alpha_2}\tilde{A}$, respectively. We prove $\underline{L}_2 \geq \underline{L}_1$ here. The proof for $\overline{L}_2 \leq \overline{L}_1$ can be done similarly. By the KM algorithm, we have

$$x_{\underline{L}_2} \leq \underline{b}_2 < x_{\underline{L}_2+1} \tag{19}$$

$$x_{\underline{L}_1} \leq \underline{b}_1 < x_{\underline{L}_1+1}. \tag{20}$$

From Proposition 1, we have

$$\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)\overline{I}_{j,2} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)\underline{I}_{j,2} = 0. \tag{21}$$

Let us assume that $\underline{L}_2 < \underline{L}_1$. Then, we have $\underline{L}_2 + 1 \leq \underline{L}_1$, and $x_{\underline{L}_2+1} \leq x_{\underline{L}_1}$. By (19), we have

$$\underline{b}_2 < x_{\underline{L}_1}. \tag{22}$$

From (21), (22), and Lemma 2, we have

$$\sum_{j=1}^{\underline{L}_2} (x_j - x_{\underline{L}_1})\overline{I}_{j,2} + \sum_{j=\underline{L}_2+1}^{n} (x_j - x_{\underline{L}_1})\underline{I}_{j,2} < 0. \tag{23}$$

Since $x_j < x_{\underline{L}_2+1}$ for $1 \leq j \leq \underline{L}_2, x_j \geq x_{\underline{L}_2+1}$ for $\underline{L}_2 + 1 \leq j \leq n, \underline{I}_{j,2} \geq \underline{I}_{j,1}$, and $\overline{I}_{j,2} \leq \overline{I}_{j,1}$, (23) can lead to the following inequality:

$$\sum_{j=1}^{\underline{L}_2} (x_j - x_{\underline{L}_1})\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - x_{\underline{L}_1})\underline{I}_{j,1} < 0. \tag{24}$$

By the assumption of $\underline{L}_2 < \underline{L}_1$, we have

$$\begin{aligned}
&\sum_{j=1}^{\underline{L}_2} (x_j - x_{\underline{L}_1})\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - x_{\underline{L}_1})\underline{I}_{j,1} < 0 \\
&\Rightarrow \sum_{j=1}^{\underline{L}_1} (x_j - x_{\underline{L}_1})\overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} (x_j - x_{\underline{L}_1})\underline{I}_{j,1} < 0 \\
&\Rightarrow \sum_{j=1}^{\underline{L}_1} x_j\overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} x_j\underline{I}_{j,1} \qquad (25)\\
&\quad < x_{\underline{L}_1}\left(\sum_{j=1}^{\underline{L}_1} \overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} \underline{I}_{j,1}\right) \\
&\Rightarrow x_{\underline{L}_1} > \frac{\sum_{j=1}^{\underline{L}_1} x_j\overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} x_j\underline{I}_{j,1}}{\sum_{j=1}^{\underline{L}_1} \overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} \underline{I}_{j,1}} = \underline{b}_1 \qquad (26)
\end{aligned}$$

which contradicts (20). Therefore, we conclude $\underline{L}_2 \geq \underline{L}_1$.

Proof of Theorem 1: We prove (16) here. Equation (11) can be proved in a similar way. From Proposition 1, we get

$$\sum_{j=1}^{\underline{L}_i} (x_j - \underline{b}_i)\overline{I}_{j,i} + \sum_{j=\underline{L}_i+1}^{n} (x_j - \underline{b}_i)\underline{I}_{j,i} = 0 \tag{27}$$

for $i = 1$ and 2. Then, we have

$$\begin{aligned}
&\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)\underline{I}_{j,1} \qquad (28)\\
&= \sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)\underline{I}_{j,1} \\
&\quad - \left[\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)\overline{I}_{j,2} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)\underline{I}_{j,2}\right] \qquad (29)\\
&= \sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)(\overline{I}_{j,1} - \overline{I}_{j,2}) \\
&\quad + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)(\underline{I}_{j,1} - \underline{I}_{j,2}). \qquad (30)
\end{aligned}$$

Note that (29) is equivalent to (28), since the bracketed expression is zero when i is set to 2 in (27). By the inclusion property of α-cuts, we have $\underline{I}_{j,1} \leq \underline{I}_{j,2}$, and $\overline{I}_{j,1} \geq \overline{I}_{j,2}$ for $1 \leq j \leq n$. Furthermore, we have $x_j \leq \underline{b}_2$ for $j \leq \underline{L}_2$, and $x_j > \underline{b}_2$ for $j \geq \underline{L}_2 + 1$. Therefore, (30) is less than or equal to 0. Then, (28) is also less than or equal to 0, i.e.,

$$\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_2)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_2)\underline{I}_{j,1} \leq 0$$

which results in

$$\frac{\sum_{j=1}^{\underline{L}_2} x_j\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} x_j\underline{I}_{j,1}}{\sum_{j=1}^{\underline{L}_2} \overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} \underline{I}_{j,1}} \leq \underline{b}_2 \tag{31}$$

after rearrangements. Also, we have

$$\begin{aligned}
&\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_1)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1} \qquad (32)\\
&= \sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_1)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1} \\
&\quad - \left[\sum_{j=1}^{\underline{L}_1} (x_j - \underline{b}_1)\overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1}\right] \qquad (33)\\
&= \sum_{j=1}^{\underline{L}_1} (x_j - \underline{b}_1)\overline{I}_{j,1} + \sum_{j=\underline{L}_1+1}^{\underline{L}_2} (x_j - \underline{b}_1)\overline{I}_{j,1} \\
&\quad + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1} - \left[\sum_{j=1}^{\underline{L}_1} (x_j - \underline{b}_1)\overline{I}_{j,1}\right.
\end{aligned}$$

$$+ \sum_{j=\underline{L}_1+1}^{\underline{L}_2} (x_j - \underline{b}_1)\underline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1} \Bigg] \quad (34)$$

$$= \sum_{j=\underline{L}_1+1}^{\underline{L}_2} (x_j - \underline{b}_1)(\overline{I}_{j,1} - \underline{I}_{j,1}). \quad (35)$$

Note that (32) is equivalent to (33), since the bracketed expression is zero when i is set to 1 in (27). The collapse of one summation into two summations in (34) is possible because $\underline{L}_1 \le \underline{L}_2$ due to Lemma 3. Since $\overline{I}_{j,1} \ge \underline{I}_{j,1}$ for $1 \le j \le n$ and $x_j > \underline{b}_1$ for $j \ge \underline{L}_1 + 1$, (35) is greater than or equal to 0. Then, (32) is also less than or equal to 0, i.e.,

$$\sum_{j=1}^{\underline{L}_2} (x_j - \underline{b}_1)\overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} (x_j - \underline{b}_1)\underline{I}_{j,1} \ge 0$$

which leads to

$$\underline{b}_1 \le \frac{\sum_{j=1}^{\underline{L}_2} x_j \overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} x_j \underline{I}_{j,1}}{\sum_{j=1}^{\underline{L}_2} \overline{I}_{j,1} + \sum_{j=\underline{L}_2+1}^{n} \underline{I}_{j,1}} \quad (36)$$

after rearrangements. By combining (31) and (36), we complete the proof for (15).

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their comments, which were very helpful in improving the quality and presentation of the paper.

REFERENCES

[1] G. J. Klir and B. Yuan, *Fuzzy Set and Fuzzy Logic*. Englewood Cliffs, NJ: Prentice-Hall PTR, May 1995.
[2] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-1," *Inform. Sci.*, vol. 8, pp. 199–249, Jan. 1975.
[3] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 535–550, Oct. 2000.
[4] J. M. Mendel, *UNCERTAIN Rule-Based Fuzzy Logic Systems*. Englewood Cliffs, NJ: Prentice-Hall PTR, Jan. 2001.
[5] J. M. Mendel and R. I. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 117–127, Apr. 2002.
[6] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic system," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, Feb. 2004.
[7] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 808–821, Dec. 2006.
[8] J. M. Mendel, "Type-2 fuzzy sets and systems: an overview," *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 20–29, Feb. 2007.
[9] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," *Inform. Sci.*, vol. 177, no. 1, pp. 84–110, Jan. 2007.
[10] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, Aug. 2004.
[11] R. Sepúlveda, O. Castillo, P. Melin, A. Rodríguez-Díaz, and O. Montiel, "Experimental study of intelligent controllers under uncertainty using type-1 and type-2 fuzzy logic," *Inform. Sci.*, vol. 177, no. 10, pp. 2023–2048, May 2007.
[12] F.-J. Lin and P.-H. Chou, "Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 178–193, Jan. 2009.
[13] O. Castillo and P. Melin, "Comparison of hybrid intelligent systems, neural networks, and interval type-2 fuzzy logic for time series prediction," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2007, pp. 3086–3091.
[14] M. H. F. Zarandi, B. Rezaee, I. B. Turksen, and E. Neshat, "A type-2 fuzzy rule-based expert system model for stock price analysis," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 139–154, Jan. 2009.
[15] H. B. Mitchell, "Pattern recognition using type-II fuzzy sets," *Inform. Sci.*, vol. 170, no. 2–4, pp. 409–418, Feb. 2005.
[16] J. Zeng and Z.-Q. Liu, "Type-2 fuzzy Markov random fields and their application to handwritten chinese character recognition," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 3, pp. 747–760, Jun. 2008.
[17] L. A. Lucas, T. M. Centeno, and M. R. Delgado, "Land cover classification based on general type-2 fuzzy classifiers," *Int. J. Fuzzy Syst.*, vol. 10, no. 3, pp. 207–216, Sep. 2008.
[18] R. I. John, P. R. Innocent, and M. R. Barnes, "Neuro-fuzzy clustering of radiographic tibia image data using type-2 fuzzy sets," *Inform. Sci.*, vol. 125, no. 1–4, pp. 65–82, Jun. 2000.
[19] L. D. Lascio, A. Gisolfi, and A. Nappi, "Medical differential diagnosis through type-2 fuzzy sets," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 2005, pp. 371–376.
[20] O. Castillo and P. Melin, "A new approach for plant monitoring using type-2 fuzzy logic and fractal theory," *Int. J. General Syst.*, vol. 33, no. 2/3, pp. 305–319, Apr. 2004.
[21] P. Melin and O. Castillo, "An intelligent hybrid approach for industrial quality control combining neural networks, fuzzy logic and fractal theory," *Inform. Sci.*, vol. 177, no. 7, pp. 1543–1557, Apr. 2007.
[22] N. R. Cazarez-Castro, L. T. Aguilar, and O. Castillo, "Hybrid genetic-fuzzy optimization of a type-2 fuzzy logic controller," in *Proc. 8th Int. Conf. Hybrid Intell. Syst.*, Sep. 2009, pp. 718–725.
[23] N. R. Cazares-Castro, L. T. Aguilar, and O. Castillo, "Designing type-2 fuzzy logic system controllers via fuzzy Lyapunov synthesis for the output regulator of a servomechanism with nonlinear backlash," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 268–273.
[24] J. R. Castro, O. Castillo, P. Melin, and A. Rodríguez-Díaz, "A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks," *Inform. Sci.*, vol. 179, no. 13, pp. 2175–2193, Jun. 2009.
[25] R. Martínez, O. Castillo, and L. T. Aguilar, "Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms," *Inform. Sci.*, vol. 179, no. 13, pp. 2158–2174, Jun. 2009.
[26] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Inform. Sci.*, vol. 178, no. 9, pp. 2224–2236, Apr. 2008.
[27] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Inform. Sci.*, vol. 132, no. 1-4, pp. 195–220, Feb. 2001.
[28] S. Coupland and R. John, "A fast geometric method for defuzzification of type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 4, pp. 929–941, Aug. 2008.
[29] L. A. Lucas, T. M. Centeno, and M. R. Delgado, "General type-2 fuzzy inference systems: Analysis, design and computational aspects," in *Proc. Int. Conf. Fuzzy Syst.*, Jul. 2007, pp. 1–6.
[30] W.-W. Tan and D. Wu, "Design of type-reduction strategies for type-2 fuzzy logic systems using genetic algorithms," in *Adv. Evol. Comput. Syst. Des.*, 2007, vol. 6, pp. 169–187.
[31] D. Wu and J. M. Mendel, "Enhanced Karnik–Mendel algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 4, pp. 923–934, Aug. 2009.
[32] S. Greenfield, F. Chiclana, S. Coupland, and R. I. John, "The collapsing method of defuzzification for discretised interval type-2 fuzzy sets," *Inform. Sci.*, vol. 179, no. 13, pp. 2055–2069, Jun. 2009.
[33] S. Greenfield, F. Chiclana, and R. I. John, "Type-reduction of the discretised interval type-2 fuzzy set," in *Proc. 18th IEEE Int. Conf. Fuzzy Syst.*, Aug. 2009, pp. 738–743.
[34] J. M. Mendel, F. Liu, and D. Zhai, "α-plane representation for type-2 fuzzy sets: Theory and applications," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1189–1207, Oct. 2009.

Chi-Yuan Yeh was born in Tainan, Taiwan, in 1971. He received the B.S. and M.S. degrees in business administration in 2002 and 2004, respectively, from Shu-Te University, Kaohsiung, Taiwan. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung.

His current research interests include machine learning, data mining, and soft computing.

Mr. Yeh received the Best Paper Award at the 2008 International Conference on Machine Learning and Cybernetics.

Wen-Hau Roger Jeng was born in Taipei County, Taiwan, in 1978. He received the B.S. degree from National Chengchi University, Taipei, in 2001 and the M.S.E.E. degree from National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2009.

He is currently with the Department of Electrical Engineering, National Sun Yat-Sen University. His current research interests include machine learning and data mining.

Shie-Jue Lee (M'90) was born in Kin-Men, Taiwan, on August 15, 1955. He received the B.S. and M.S. degrees in electrical engineering in 1977 and 1979, respectively, from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in computer science from the University of North Carolina, Chapel Hill, in 1990.

He joined the faculty of the Department of Electrical Engineering, National Sun Yat-Sen University (NSYSU), Kaohsiung, Taiwan, in 1983, where he has been a Professor in the department since 1994. He has been the Director of the Center for Telecommunications Research and Development, NSYSU (1997–2000), the Director of the Southern Telecommunications Research Center, National Science Council (1998–1999), and the Chair of the Department of Electrical Engineering, NSYSU (2000–2003). He is currently the Deputy Dean of the Academic Affairs and the Director of the NSYSU-III Research Center. His current research interests include artificial intelligence, machine learning, data mining, information retrieval, and soft computing.

Dr. Lee received the Distinguished Teachers Award from the Ministry of Education, Taiwan, in 1993. He was awarded by the Chinese Institute of Electrical Engineering for Outstanding M.S. Thesis Supervision (1997). He received the Distinguished Paper Award from the Computer Society of the Republic of China (1998), the Best Paper Award from the Seventh Conference on Artificial Intelligence and Applications (2002), the Best Paper Award from the International Conference on Machine Learning and Cybernetics (2008), and the Best Paper Award of Cloud Computing and Virtualization (2010). He received the Distinguished Research Award from National Sun Yat-Sen University (1998) and the Distinguished Teaching Award from National Sun Yat-Sen University in 1993 and 2008, respectively. He also received the Distinguished Mentor Award from National Sun Yat-Sen University in 2008. He was the Program Chair for the International Conference on Artificial Intelligence, Kaohsiung, Taiwan, in December 1996, the International Computer Symposium—Workshop on Artificial Intelligence, Tainan, Taiwan, in December 1998, and the Sixth Conference on Artificial Intelligence and Applications, Kaohsiung, in November 2001. He is a member of the IEEE Systems, Man, and Cybernetics Society, the IEEE Computational Intelligence Society, the Association for Automated Reasoning, the Institute of Information and Computing Machinery, the Taiwan Fuzzy Systems Association, and the Taiwanese Association of Artificial Intelligence.