大葉大學 九十四 學年度 研究所博士班 招生考試試題紙					
系 所 別	組 別	考 試 科 目 (中文名稱)	考試 日期	節 次	備 註
管理研究所	乙	論文評述	6月20日	第 二節	

註：考生可否攜帶計算機或其他資料作答，請在備註欄註明（如未註明，一律不准攜帶）

請就所提供的兩篇論文中，選擇一篇閱讀並回答下列問題

【請務必於答案卷中，註記作答的論文編號，否則以零分計算】:

1.請說明該篇論文的主要研究目的、所要解決的問題以及所提出的解決問題的方法【請務必自行整理，並以**中文**作答說明】。(30%)

2.請說明該篇論文的優點以及缺點，並寫出如何改進該論文的缺點。如果沒有缺點，則寫出為什麼沒有缺點的理由【請務必自行整理，並以**中文**作答說明】。(40%)

3.請就整篇論文內容訂定合適它的英文題目名稱，並用英文寫出約200-300 字左右的英文摘要。(30%)

論文編號：1.

1. Introduction

In this section we first briefly review the contents of the paper, present our notation, and review the literature on fuzzy hierarchical analysis (FHA). In Section 2 we review the computational details of finding the weights in the analytical hierarchical process. We fuzzify hierarchical analysis in Section 3 by allowing fuzzy numbers for the pairwise comparisons. Direct computation of fuzzy eigenvalues and fuzzy eigenvectors (the fuzzy weights) from a fuzzy, positive, reciprocal matrix is outlined in this section.

Section 4 contains an example having five criteria and three alternatives. In Section 4 we also discuss consistency of fuzzy, positive, reciprocal matrices and how to obtain the final ranking based on fuzzy weights. Section 5 gives the results in the example and Section 6 contains conclusions.

We place a "bar" over a letter to denote a fuzzy set. All our fuzzy sets will be fuzzy subsets of the real numbers. So, $\bar{a}_{ij}, \bar{w}_{ij}, \bar{b}, \bar{c}, \ldots$ are all fuzzy subsets of $\Re$. If $\bar{a}$ is a fuzzy set, then $\bar{a}(x)$ is the value of the membership function at $x \in \Re$. An α-cut of $\bar{a}$, written as $\bar{a}[\alpha]$, is defined as $\{x \mid \bar{a}(x) \geqslant \alpha\}$ for $0 < \alpha \leqslant 1$. The support of $\bar{a}$, written as $\bar{a}[0]$, is the closure of the union of $\bar{a}[\alpha]$, $0 < \alpha \leqslant 1$.

A triangular fuzzy number $\bar{N}$ is defined by three numbers $a < b < c$ where the graph of $y = \bar{N}(x)$ is a triangle with base on the interval $[a, c]$ and vertex at $x = b$. We write $\bar{N} = (a/b/c)$. A triangular-shaped fuzzy

0165-0114/01/$-see front matter
PII: S0165-0114(99)00155-4

number $\bar{M}$ is specified by its α-cuts $\bar{M}[\alpha]=[m_1(\alpha), m_2(\alpha)]$ where $n_1(\alpha_2)\leqslant m_1(\alpha_1), m_2(\alpha_2)\geqslant m_2(\alpha_1)$ for all $0\leqslant\alpha_2<\alpha_1\leqslant 1$. The actual triangular-shaped fuzzy number $\bar{M}$ is then defined by

$$\bar{M}=\bigcup_{0\leqslant\alpha\leqslant 1}\alpha\bar{M}[\alpha]. \tag{1}$$

A trapezoidal fuzzy number $\bar{N}$ is defined by four numbers $a<b<c<d$ where the graph of $y=\bar{N}(x)$ is a trapezoid with base on the interval $[a,d]$ and $\bar{N}(x)=1$ for $b\leqslant x\leqslant c$. A trapezoidal-shaped fuzzy number $\bar{M}$ is specified by its α-cuts $\bar{M}[\alpha]$, just like a triangular-shaped fuzzy numter, except that $m_1(1)\leqslant m_2(1)$.

All our fuzzy numbers will be triangular (trapezoidal) fuzzy numbers, their reciprocals, or triangular (trapezoidal)-shaped fuzzy numbers. Also, all our fuzzy sets will be strictly positive which means that $a_1(0)>0$ where $\bar{A}[\alpha]=[a_1(\alpha), a_2(\alpha)]$.

In FHA, one uses fuzzy numbers for the pairwise comparisons and the main problem is to compute the corresponding fuzzy weights. The direct approach, of finding fuzzy eigenvalues and fuzzy eigenvectors, was considered too computationally difficult [2–5] except for [6,10] to be discussed below, so researchers fuzzified another method. However, all of these methods, except [6,10] and this paper, deviate from the original procedure used by Saaty in HA for finding the weights.

In [15], the authors by using the results in [16] on log least squares extended HA to FHA. They used logarithmic regression to estimate the fuzzy weights (see also [17]). In their model they can have multiple estimates for each pairwise comparison and they can handle the problem of missing data (no estimates for certain comparisons). However, as pointed out in an example in [13], the logarithmic least-squares method can produce different weights, compared to Saaty's original procedure, for crisp data. In [1], the authors pointed out an error in [15] and they showed how to correct the procedure. However, in [11], it is shown that this method can produce fuzzy weights $\bar{w}=(w_1/w_2/w_3)$, triangular fuzzy numbers, with $w_3<w_1$. That is, it is not a fuzzy number. This paper was followed by [12] where the authors define the concept of strong transitivity of a fuzzy, positive, reciprocal matrix (Section 3) and show that if this condition is satisfied, the log least-squares method of [15] produces triangular fuzzy weights with $w_1<w_3$.

The logarithmic least-squares method of obtaining fuzzy weights has been carried on in other papers. In [14], the author presents another solution to the problem using a generalized pseudo-inverse approach but also points out that you can get $w_3<w_1$. The paper [21] uses "step-form" fuzzy numbers in logarithmic least squares to estimate these fuzzy weights, but they use a different objective function to be minimized in logarithmic regression.

There are also other papers in FHA using different procedures to compute fuzzy weights. In [20], they employed "step-form" fuzzy numbers and fuzzified another procedure, which they claim is the same as Saaty's original method for crisp perfectly consistent, positive, reciprocal matrices, tc calculate the fuzzy weights. However, the matrices are usually not perfectly consistent only "reasonably" consistent, so this procedure will produce different weights compared to Saaty's original method, for crisp data. The paper [18] uses fuzzy relational equations to model the FHA problem. The modeling in [18] gives a fuzzy hierarchical process quite different from Saaty's original HA. The author in [25] developed a method for the interactive analysis of fuzzy pairwise comparisons in hierarchical weighting models which appears, in our opinion, far removed from Saaty's original HA.

The series of papers [7–9,19] are also related to FHA. In [8,9,19], they changed a fuzzy, positive, reciprocal matrix into a crisp matrix, using α-cuts and convex combinations, and then computed the eigenvector (weight vector) from the crisp matrix. They do not obtain a fuzzy weight vector. Paper [7] is about speeding up the calculations in [19]. In our opinion, these papers are not about FHA since there are no fuzzy weights.

Paper [10] is in the spirit of Saaty's original HA. They first discuss a way of finding $\lambda_{\max}$ (Section 2), where $\lambda_{\max}$ is the largest, positive eigenvalue of a fuzzy, positive, reciprocal matrix. However, where they run into computational problems is in computing of the fuzzy eigenvector associated with $\lambda_{\max}$.

In [2,5] the author also presents a method of computing the fuzzy weights in FHA. He used the fuzzification of the geometric mean of each row. If the positive, reciprocal matrix is perfectly consistent, then the geometric row mean procedure gives the same weights as the eigenvector method, which was Saaty's original method. However, we do not expect perfect consistency, so the geometric row procedure can give different weights compared to the eigenvector method.

In [6], the authors did not directly find a fuzzy $\lambda_{\max}$ and its corresponding fuzzy eigenvector, but instead fuzzified an equivalent procedure for finding the fuzzy weights. They tested their method on the 3×3 and 4×4 cases, where formulas exist for the fuzzy weights, to show that their procedure finds the correct fuzzy weights.

This paper directly fuzzifies Saaty's $\lambda_{\max}$ method of computing the weights and hence gives an alternative method to that in [6]. In fact, we use the same example as in [6] to compare the two approaches.

2. Hierarchical analysis

In this section, we review the basic computations needed to find the weights in hierarchical analysis (HA). In HA, a person (expert, judge) is asked to give ratios a_{ij} for each pairwise comparison between issues (alternatives, candidates) $A_1, \ldots, A_n$ for each criterion (objective) in a hierarchy, and also between the criteria. For some specific criterion C_k if a person considers A_1 more important than A_5 then a_{15} might equal 3/1, or 5/1, or 7/1. The numbers for the ratios will be taken from the set $S = \{1, 2, 3, \ldots, 9\}$ so a_{15} could be s_1/s_5 with $s_1, s_5 \in S$ and $s_1 > s_5$. The ratios a_{ij} indicate, for this expert, the strength with which A_i dominates A_j. If $a_{15} = 5/1$ then $a_{51} = 1/5$. That is, $a_{ij} = (a_{ji})^{-1}$, all i, j, with $a_{ii} = 1$, $1 \leqslant i \leqslant n$. Let A be the $n \times n$ matrix whose entries are the ratios $(a_{ij} = (a_{ji})^{-1})$. A is called a positive reciprocal matrix. Since A is for criterion C_k we will now write A_k for this matrix.

Assume that there are K criteria $C_1, \ldots, C_K$ with a positive reciprocal matrix A_k for each C_k, $1 \leqslant k \leqslant K$. Also, the judge must give pairwise comparisons of the criteria producing a positive reciprocal matrix E. This hierarchical structure is shown in Fig. 1. Examples, with actual fuzzy numbers in the A_k and E, are presented in Section 4.

Next, one computes weights $w_k^{\mathrm{T}} = (w_{1k}, \ldots, w_{nk})$ for each A_k and $e^{\mathrm{T}} = (e_1, \ldots, e_K)$ for E. Given any positive reciprocal matrix A, let the eigenvalues, counting a root of multiplicity m m-times, be $\lambda_1, \ldots, \lambda_n$. There is a dominant (real, positive) eigenvalue, let us call it $\lambda_{\max}$, so that $|\lambda_i| < \lambda_{\max}$ for all $\lambda_i \neq \lambda_{\max}$. Also, $\lambda_{\max}$ is a root of multiplicity one. Corresponding to $\lambda_{\max}$ there is a unique eigenvector $w^{\mathrm{T}} = (w_1, \ldots, w_n)$ so that

$$Aw = \lambda_{\max} w, \tag{2}$$

where $w_i > 0$ for all i and $\sum_{i=1}^{n} w_i = 1$. This positive, normalized (sum one), vector w gives the weights for A [22–24]. Then w_k is the positive, normalized eigenvector corresponding to $\lambda_{\max}$ for A_k, $1 \leqslant k \leqslant K$, and e is the eigenvector for E.

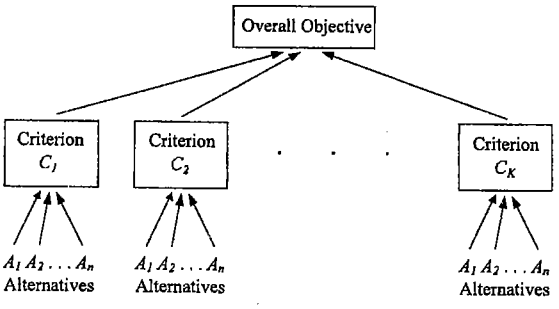


Fig. 1. Hierarchical structure.

The objective of HA is to rank the alternatives across all the criteria. Then, assuming that the reciprocal matrices A_k, $1 \leqslant k \leqslant K$, and E are reasonably consistent [22–24], the final ranking of the alternatives is determined by the vector $r^{\mathrm{T}} = (r_1, \ldots, r_n)$ where

$$r_j = \sum_{k=1}^{K} w_{jk} e_k, \tag{3}$$

$1 \leqslant j \leqslant n$. We will discuss consistency for fuzzy hierarchical analysis in Section 4. The weight for alternative A_j is r_j, $1 \leqslant j \leqslant n$. The alternatives are ranked according to the numbers r_j, $1 \leqslant j \leqslant n$. The hierarchical structure (Fig. 1) can be expanded to more levels, but we shall consider, in this paper, only the three levels shown in Fig. 1.

3. Fuzzy hierarchical analysis

The experts are allowed to use fuzzy ratios in place of exact ratios. The $\bar{a}_{ij}$, $i \neq j$, can now be fuzzy numbers in any positive reciprocal matrix. As before, $\bar{a}_{ii} = 1$ for all i. We will start using only triangular fuzzy numbers, or their reciprocals, for the $\bar{a}_{ij}$, $i \neq j$. Then we show how our procedure easily extends to trapezoidal fuzzy numbers and their reciprocals.

The types of fuzzy numbers that can be used in paired comparisons are described by $\bar{a}_{ij} = (\alpha/\beta/\gamma)$ where $\alpha, \beta, \gamma \in S$, $\alpha < \beta < \gamma$. If $\bar{a}_{ij} = (\alpha/\beta/\gamma)$ then $\bar{a}_{ij} = (\bar{a}_{ij})^{-1} = (\gamma^{-1}/\beta^{-1}/\alpha^{-1})$. However, $\bar{a}_{ij}^{-1}$ is not exactly a triangular fuzzy number, if $\bar{a}_{ij}$ is a triangular fuzzy number, but we will use the same notation $(\gamma^{-1}/\beta^{-1}/\alpha^{-1})$ for $\bar{a}_{ij}^{-1}$.

We will need the α-cuts of all the $\bar{a}_{ij}$, $i \neq j$. If $\bar{a}_{ij} = (\alpha/\beta/\gamma)$, then set $\bar{a}_{ij}[\alpha] = [a_{ijl}(\alpha), a_{iju}(\alpha)]$ and then $\bar{a}_{ij}^{-1}[\alpha] = [a_{iju}^{-1}(\alpha), a_{ijl}^{-1}(\alpha)]$.

Now, we assume that the elements in the fuzzy positive reciprocal matrices $\bar{A}_\kappa$ and $\bar{E}$ are $\bar{a}_{ij} = (\alpha/\beta/\gamma)$, $\bar{a}_{ii} = 1$, $\bar{a}_{ji} = \bar{a}_{ij}^{-1}$.

We now describe how we are going to compute the fuzzy weight vectors $\bar{w}_k$ and $\bar{e}$. There are a number of other issues to be addressed in FHA, like consistency, and how do we obtain the final ranking because now the weight r_j (Eq. (3)) for alternative A_j will be a fuzzy number. These two issues will be considered in Section 4. Right now, we are only concerned with finding the fuzzy weight vector for a fuzzy, positive, reciprocal matrix.

Let $\bar{A}$ be an $n \times n$ fuzzy, positive, reciprocal matrix with elements $\bar{a}_{ij}$. Let $\bar{a}_{ij}[\alpha] = [a_{ijl}(\alpha), a_{iju}(\alpha)]$. If $i = j$, then $a_{iil}(\alpha) = a_{iiu}(\alpha) = 1$. If $\bar{a}_{ji} = \bar{a}_{ij}^{-1}$, then $a_{jil}(\alpha) = \bar{a}_{iju}^{-1}(\alpha)$, $a_{jiu}(\alpha) = \bar{a}_{ijl}^{-1}(\alpha)$. Define $n \times n$ matrices $A_{\alpha l} = [a_{ijl}(\alpha)]$, $A_{\alpha u} = [a_{iju}(\alpha)]$, all α in $[0, 1]$. For all $i \neq j$ there is a unique x_{ijm} so that $\bar{a}_{ij}(x_{ijm}) = 1$. Define $A_m = [x_{ijm}]$ where $x_{iim} = 1$ for all i. Note that $A_{\alpha l}$ and $A_{\alpha u}$ are positive matrices but they are no longer reciprocal matrices.

Now each $A_{\alpha l} > 0$, $A_{\alpha u} > 0$, and $A_m > 0$ so each matrix has a positive, dominant eigenvalue [22,24], called $\lambda_{\max}$. Let $\lambda_{\alpha l} = \lambda_{\max}$ for $A_{\alpha l}$, $\lambda_{\alpha u} = \lambda_{\max}$ for $A_{\alpha u}$ and $\lambda_m = \lambda_{\max}$ for A_m. We know that [22,24]

$$\lambda_{0l} < \lambda_{\alpha l} < \lambda_m < \lambda_{\alpha u} < \lambda_{0u} \tag{4}$$

for $0 < \alpha < 1$. Let $\bar{\lambda}_{\max}$ the triangular-shaped fuzzy number specified by the α-cuts $[\lambda_{\alpha l}, \lambda_{\alpha u}]$. Note that $\bar{\lambda}_{\max}(x) = 1$ for $x = \lambda_m$.

Let w_m be an $n \times 1$ unique, positive, normalized (sum is one) eigenvector corresponding to λ_m [22,24]. Then $A_m w_m = \lambda_m w_m$ and if $w_m^{\mathrm{T}} = (w_{1m}, \ldots, w_{nm})$, then $w_{1m} > 0$, $w_{1m} + \cdots + w_{nm} = 1$.

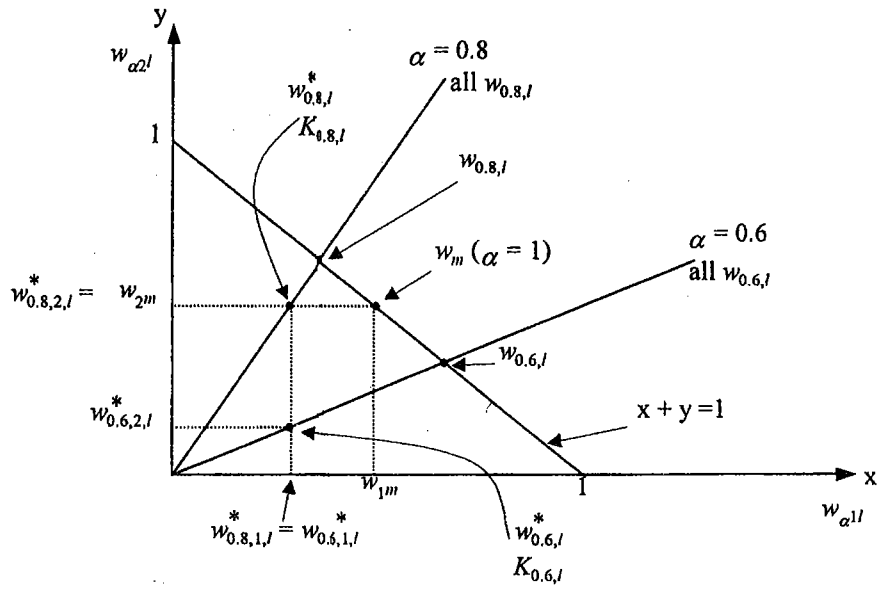


Fig. 2. Selecting $w^*_{\alpha l}$ when $n = 2$.

Now let $w_{\alpha l}(w_{\alpha u})$ be an $n \times 1$ unique, positive, normalized (sum is one) eigenvector corresponding to $\lambda_{\alpha l}(\lambda_{\alpha u})$. We will show how to find constants $K_{\alpha l}, K_{\alpha u}$ so that:

(1) $0 < K_{\alpha l} < 1$, $1 < K_{\alpha u}$; and (2) $K_{\alpha l}, K_{\alpha u}$ depend only on α, $0 \leqslant \alpha < 1$. Then we will define

$$w^*_{\alpha l} = K_{\alpha l} w_{\alpha l}, \tag{5}$$

$$w^*_{\alpha u} = K_{\alpha u} w_{\alpha u} \tag{6}$$

for $0 \leqslant \alpha < 1$. Let $(w^*_{\alpha l})^{\mathrm{T}} = (w^*_{\alpha 1 l}, \ldots, w^*_{\alpha n l})$, $(w^*_{\alpha u})^{\mathrm{T}} = (w^*_{\alpha 1 u}, \ldots, w^*_{\alpha n u})$, $w^{\mathrm{T}}_{\alpha l} = (w_{\alpha 1 l}, \ldots, w_{\alpha n l})$ and $w^{\mathrm{T}}_{\alpha u} = (w_{\alpha 1 u}, \ldots, w_{\alpha n u})$.

We next show that if $0 \leqslant \alpha_2 < \alpha_1 < 1$, then $0 < w^*_{\alpha_2 i l} \leqslant w^*_{\alpha_1 i l} \leqslant w_{im} \leqslant w^*_{\alpha_1 i u} \leqslant w^*_{\alpha_2 i u}$, for all $i = 1, \ldots, n$. This means that the α-cuts $[w^*_{\alpha i l}, w^*_{\alpha i u}]$ specify a triangular-shaped fuzzy number $\bar{w}^*_i$, $1 \leqslant i \leqslant n$. Set $(\bar{w}^*)^{\mathrm{T}} = (\bar{w}^*_1, \ldots, \bar{w}^*_n)$. The fuzzy weights will be the $\bar{w}^*_i$, $1 \leqslant i \leqslant n$.

Finally, we argue that

$$\bar{A}\bar{w}^* = \bar{\lambda}_{\max}\bar{w}^*, \tag{7}$$

using α-cuts and interval arithmetic.

We also choose the $K_{\alpha l}, K_{\alpha u}$ to minimize the "fuzziness" of the $\bar{w}^*_i$. By the "fuzziness" we mean the lengths of the α-cuts. We wish to minimize the fuzziness so that we can "spread out" the alternatives for the final ranking. By "spreading out" the alternatives we mean that the set H_i, in Section 4.2, will have few members.

Our method may be easily explained geometrically in Figs. 2 and 3 for the $n = 2$ case. The reader may then immediately generalize to $n \geqslant 3$. Fig. 2 is for selecting $K_{\alpha l}$. The eigenvectors w_m and $w_{\alpha l}$ are non-negative and normalized, so they will lie on the line $x + y = 1$. The line $\alpha = 0.8$, labeled as "all $w_{0.8,l}$", represents all positive eigenvectors corresponding to $\lambda_{0.8,l}$. So we must select $w^*_{0.8,l}$ on this line. But we also want $w^*_{0.8,1,l} \leqslant w_{1m}$ and $w^*_{0.8,2,l} \leqslant w_{2m}$. Therefore, we must choose $w^*_{0.8,l}$ along the line segment from $(0,0)$ to the point labeled "$w^*_{0.8,l}, K_{0.8,l}$" on the line $\alpha = 0.8$ in Fig. 2. To minimize fuzziness we choose $w^*_{0.8,l}$ to be the point labeled as "$w^*_{0.8,l}, K_{0.8,l}$" and this determines the number $K_{0.8,l}$. As a result, we see that $w^*_{0.8,1,l} < w_{1m}$ but

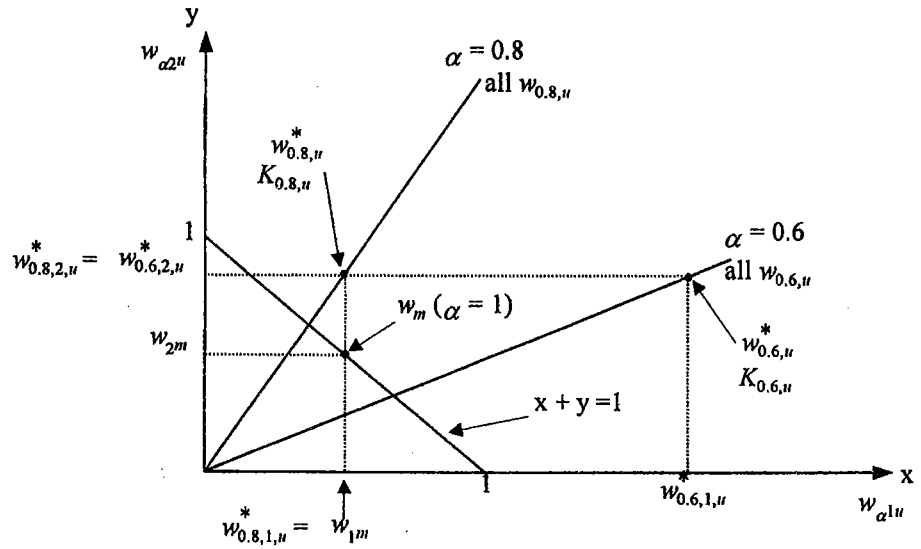


Fig. 3. Selecting $w^*_{\alpha u}$ when $n = 2$.

$w^*_{0.8,2,l}$ will be equal to w_{2m}. Next, we need to choose $w^*_{0.6,l}$ along the line labeled as "all $w_{0.6,l}$" in Fig. 2. Since we require $w^*_{0.6,1,l} \leqslant w^*_{0.8,1,l}$ and $w^*_{0.6,2,l} \leqslant w^*_{0.8,2,l}$ we see that to reduce fuzziness we must choose $w^*_{0.6,l}$ at the point labeled as "$w^*_{0.6,l}, K_{0.6,l}$" which then determines $K_{0.6,l}$.

Fig. 3 is for finding the constants $K_{\alpha u}$. We will pick $w^*_{0.8,u}$ along the line $\alpha = 0.8$ but we must have $w_{1m} \leqslant w^*_{0.8,1,u}$ and $w_{2m} \leqslant w^*_{0.8,2,u}$. So, for minimum fuzziness choose $w^*_{0.8,u}$ at the point labeled as "$w^*_{0.8,u}$, $K_{0.8,u}$" and we get $w^*_{0.8,1,u} = w_{1m}$ but $w_{2m} < w^*_{0.8,2,u}$. Similarly, define $K_{0.6,u}$ so that $K_{0.6,u} w_{0.6,u} = w^*_{0.6,u}$ is at point "$w^*_{0.6,u}, K_{0.6,u}$" in Fig. 3 and now $w^*_{0.6,1,u} > w^*_{0.8,1,u}$, $w^*_{0.6,2,u} = w^*_{0.8,2,u}$. Let us return to the general method for any $n \geqslant 2$.

Actually, the process will be iterative. Choose α_i in $[0, 1]$ so that $0 = \alpha_n < \alpha_{n-1} < \cdots < \alpha_1 < 1$. We first find the $w^*_{\alpha_1 il}$ and $w^*_{\alpha_1 iu}$, $1 \leqslant i \leqslant n$. Then, using these results we determine the $w^*_{\alpha_2 il}$, $w^*_{\alpha_2 iu}$, $1 \leqslant i \leqslant n$. We work our way down to finally obtain w^*_{0il}, w^*_{0iu}, $1 \leqslant i \leqslant n$. So, let us only go through α_1 and α_2. Define

$$K_{\alpha_1 l} = \min\left\{ \frac{w_{im}}{w_{\alpha_1 il}} \;\middle|\; 1 \leqslant i \leqslant n \right\}, \tag{8}$$

$$K_{\alpha_1 u} = \max\left\{ \frac{w_{im}}{w_{\alpha_1 iu}} \;\middle|\; 1 \leqslant i \leqslant n \right\}. \tag{9}$$

Then surely

$$w^*_{\alpha_1 il} \leqslant w_{im} \leqslant w^*_{\alpha_1 iu}, \tag{10}$$

for all i. In fact, we used the largest possible $K_{\alpha l}$ and the smallest possible $K_{\alpha u}$ to minimize the fuzziness.

Next, define

$$K_{\alpha_2 l} = \min\left\{ \frac{w^*_{\alpha_1 il}}{w_{\alpha_2 il}} \;\middle|\; 1 \leqslant i \leqslant n \right\}, \tag{11}$$

$$K_{\alpha_2 u} = \max\left\{ \frac{w^*_{\alpha_1 iu}}{w_{\alpha_2 iu}} \;\middle|\; 1 \leqslant i \leqslant n \right\} \tag{12}$$

and we get

$$0 < w^*_{\alpha_2 il} \leqslant w^*_{\alpha_1 il} \leqslant w_{im} \leqslant w^*_{\alpha_1 iu} \leqslant w^*_{\alpha_2 iu}, \tag{13}$$

for all i.

All we need to show is that $\bar{A}\bar{w}^* = \bar{\lambda}_{\max}\bar{w}^*$. We will use α-cuts and interval arithmetic. Since everything is positive all we need from interval arithmetic is that $[a,b][c,d] = [ac, bd]$. Let $A_\alpha = \lfloor [a_{ijl}(\alpha), a_{iju}(\alpha)] \rfloor$, $\bar{\lambda}_{\max}(\alpha) = [\lambda_{\alpha l}, \lambda_{\alpha u}]$, and $(\bar{w}^*(\alpha))^{\mathrm{T}} = ([\bar{w}^*_{\alpha 1 l}, \bar{w}^*_{\alpha 1 u}], \ldots, [\bar{w}^*_{\alpha n l}, \bar{w}^*_{\alpha n u}])$.

Then $A_\alpha \bar{w}^*(\alpha) = \bar{\lambda}_{\max}(\alpha)\bar{w}^*(\alpha)$ is equivalent to

$$A_{\alpha l} w^*_{\alpha l} = \lambda_{\alpha l} w^*_{\alpha l}, \tag{14}$$

$$A_{\alpha u} w^*_{\alpha u} = \lambda_{\alpha u} w^*_{\alpha u}, \tag{15}$$

which is true since $w^*_{\alpha l}$ ($w^*_{\alpha u}$) is a constant $K_{\alpha l}$ ($K_{\alpha u}$) times $w_{\alpha l}$ ($w_{\alpha u}$). That is, we know that $A_{\alpha l} w_{\alpha l} = \lambda_{\alpha l} w_{\alpha l}$ and $A_{\alpha u} w_{\alpha u} = \lambda_{\alpha u} w_{\alpha u}$.

Now, we will briefly explain how the above method may be extended to cover trapezoidal fuzzy numbers and their reciprocals for the $\bar{a}_{ij}$, $i \neq j$.

$\bar{A}$ is an $n \times n$ fuzzy, positive, reciprocal matrix with elements $\bar{a}_{ij} = (\alpha/\beta, \gamma/\delta)$ for $\alpha, \beta, \gamma, \delta \in S$, $\alpha < \beta \leqslant \gamma < \delta$. We assume that for some $i \neq j$ we have $\beta < \gamma$. Then $\bar{a}^{-1}_{ij} = (\delta^{-1}/\gamma^{-1}, \beta^{-1}/\alpha^{-1})$ and $\bar{a}_{ii} = 1$ for all i. As before, define $A_{\alpha l}$ and $A_{\alpha u}$ and $\lambda_{\alpha l}, \lambda_{\alpha u}$ are their corresponding dominant, positive eigenvalues. $\bar{\lambda}_{\max}$ is the trapezoidal-shaped fuzzy number defined by the α-cuts $[\lambda_{\alpha l}, \lambda_{\alpha u}]$. As before $w_{\alpha l}$ ($w_{\alpha u}$) are the unique, positive, normalized (sum is one) eigenvectors corresponding to $\lambda_{\alpha l}$ ($\lambda_{\alpha u}$).

Define $x_{ijm} = (\beta + \gamma)/2$ if $\bar{a}_{ij} = (\alpha/\beta, \gamma/\delta)$, $x_{ijm} = (\gamma^{-1} + \beta^{-1})/2$ when $\bar{a}_{ij} = (\delta^{-1}/\gamma^{-1}, \beta^{-1}/\alpha^{-1})$, and $x_{iim} = 1$ for all i. Define an $n \times n$ $X = [x_{ijm}]$. Let λ_m be the dominant, positive eigenvalue of X and w_m its corresponding positive, normalized eigenvector.

Choose the α_i in $[0, 1]$ with $0 = \alpha_n < \alpha_{n-1} < \cdots < \alpha_1 = 1$. As before, we first find the w^*_{1il} and w^*_{1iu}, $1 \leqslant i \leqslant n$. Using these values we determine $w^*_{\alpha_2 il}, w^*_{\alpha_2 iu}$, $1 \leqslant i \leqslant n$. Eventually, we work our way down to w^*_{0il}, w^*_{0iu}, $1 \leqslant i \leqslant n$. These values determine the trapezoidal-shaped fuzzy numbers $\bar{w}^*_i$, $1 \leqslant i \leqslant n$, so that $\bar{A}\bar{w}^* = \bar{\lambda}_{\max}\bar{w}^*$.

Let us now only show how to get the w^*_{1il} and w^*_{1iu}, $1 \leqslant i \leqslant n$. We find constants K_{1l}, K_{1u} so that

$$w^*_{1l} = K_{1l} w_{1l}, \tag{16}$$

$$w^*_{1u} = K_{1u} w_{1u}. \tag{17}$$

These constants are

$$K_{1l} = \min\left\{ \frac{w_{im}}{w_{1il}} \;\middle|\; 1 \leqslant i \leqslant n \right\}, \tag{18}$$

$$K_{1u} = \max\left\{ \frac{w_{im}}{w_{1iu}} \;\middle|\; 1 \leqslant i \leqslant n \right\}. \tag{19}$$

Finally, we need to explain what to do if $\alpha = \beta < \gamma < \delta$, or $\alpha = \beta = \gamma < \delta$, etc., in the trapezoidal fuzzy numbers because this will occur in the application in the next section (see [6]). We follow the same procedure as that outlined above to get the fuzzy weights since all that is needed are the α-cuts of the $\bar{a}_{ij}$ in $\bar{A}$. All that is required is for $0 < a_{ijl}(\alpha_2) \leqslant a_{ijl}(\alpha_1) \leqslant 1$ and $1 \leqslant a_{iju}(\alpha_1) \leqslant a_{iju}(\alpha_2)$ if $0 \leqslant \alpha_2 < \alpha_1 \leqslant 1$. The types of fuzzy numbers we are discussing are shown in Fig. 4 with their reciprocals in Fig. 5.

Finally, we need to consider the question of whether or not our method of calculating the fuzzy weights for fuzzy, positive, reciprocal matrices is independent of the set of α's used in the α-cuts. The question is: if we use $\alpha_1 = 0.9$, $\alpha_2 = 0.8$, $\alpha_3 = 0.7, \ldots, \alpha_9 = 0.1$ and $\alpha_{10} = 0$ and someone else uses $\alpha_1 = 0.8$, $\alpha_2 = 0.6, \ldots, \alpha_5 = 0$,

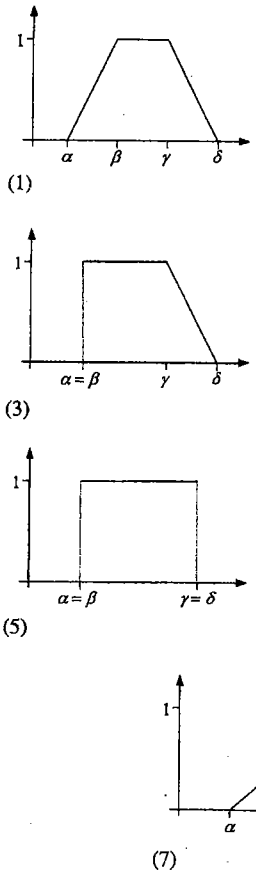


Fig. 4. Fuzzy number in FHA: (1) trapezoidal; (2) triangle; (3) more than α to 1; (4) less than δ to 1; (5) between $\alpha/1$ and $\gamma/1$; (6) at least $\alpha/1$; and (7) at most $\delta/1$.

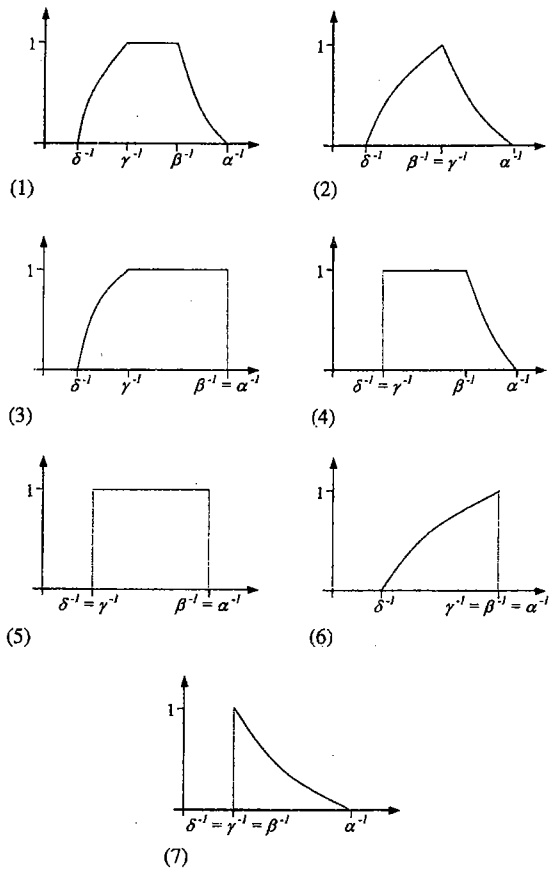


Fig. 5. Reciprocals of fuzzy numbers in Fig. 4: (1) reciprocal of Fig. 4(1); (2) reciprocal of Fig. 4(2); (3) reciprocal of Fig. 4(3); (4) reciprocal of Fig. 4(4); (5) reciprocal of Fig. 4(5); (6) reciprocal of Fig. 4(6); and (7) reciprocal of Fig. 4(7).

are the α-cuts of the fuzzy weights the same for $\alpha = 0.8$, 0.6, 0.4 and 0.2? We will now present a sufficient condition for this to be true.

For simplicity, let $n = 3$. Let us consider $0 \leqslant \alpha_2 < \alpha_1 < 1$ and first get the fuzzy weights for both α_1 and α_2. Then we will calculate the fuzzy weights for only α_2. For example, α_1 could be 0.9 and $\alpha_2 = 0.8$. We give a sufficient condition so that the α-cuts of the fuzzy weights are the same for $\alpha = \alpha_2$ using both procedures. Also, let us only look at the left end points of the α-cuts. From Eq. (8) we get $K_{\alpha_1 l}$ from

$$\min\left\{\frac{w_{1m}}{w_{\alpha_1 1l}}, \frac{w_{2m}}{w_{\alpha_1 2l}}, \frac{w_{3m}}{w_{\alpha_1 3l}}\right\} \tag{20}$$

and assume that the minimum is taken on in the second position. That is, $K_{\alpha_1 l}$ is $w_{2m}/w_{\alpha_1 2l}$. Then we compute $w^*_{\alpha_1 il} = K_{\alpha_1 l} w_{\alpha_1 il}$, $1 \leqslant i \leqslant 3$. Next, we find $K_{\alpha_2 l}$ from

$$\min\left\{\frac{w^*_{\alpha_1 1l}}{w_{\alpha_2 1l}}, \frac{w^*_{\alpha_1 2l}}{w_{\alpha_2 2l}}, \frac{w^*_{\alpha_1 3l}}{w_{\alpha_2 3l}}\right\}. \tag{21}$$

Then $w^*_{\alpha_2 il} = K_{\alpha_2 l} w_{\alpha_2 il}$, $i = 1, 2, 3$.

Now we find $w^*_{\alpha_2 l}$ directly from w_m. First, let $\tilde{K}_{\alpha_2 l}$ be

$$\min\left\{\frac{w_{1m}}{w_{\alpha_2 1l}}, \frac{w_{2m}}{w_{\alpha_2 2l}}, \frac{w_{3m}}{w_{\alpha_2 3l}}\right\}. \tag{22}$$

It follows that $\tilde{w}^*_{\alpha_2 il} = \tilde{K}_{\alpha_2 l} w_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$, by this method. Will $\tilde{w}^*_{\alpha_2 il} = w^*_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$?

A sufficient condition for $\tilde{w}^*_{\alpha_2 il} = w^*_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$, is for the minimum in Eq. (21) to be also in the second position, as in Eq. (20).

So assume that $K_{\alpha_2 l} = w^*_{\alpha_1 2l}/w_{\alpha_2 2l}$. We now show that it follows that the minimum in Eq. (22) must also be in the second position and then $\tilde{w}^*_{\alpha_2 il} = w^*_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$.

From the minimum in Eq. (20) we have

$$K_{\alpha_1 l} = \frac{w_{2m}}{w_{\alpha_1 2l}} \leqslant \frac{w_{im}}{w_{\alpha_1 il}}, \quad i \neq 2 \tag{23}$$

or

$$w_{2m} \leqslant \frac{w_{im} w_{\alpha_1 2l}}{w_{\alpha_1 il}}, \quad i \neq 2. \tag{24}$$

Next, from the minimum in Eq. (21) we see that

$$K_{\alpha_2 l} = \frac{w^*_{\alpha_1 2l}}{w_{\alpha_2 2l}} \leqslant \frac{w^*_{\alpha_1 il}}{w_{\alpha_2 il}}, \quad i \neq 2 \tag{25}$$

or, after substituting $w^*_{\alpha_1 il} = K_{\alpha_1 l} w_{\alpha_1 il}$, $1 \leqslant i \leqslant 3$, and substituting for $K_{\alpha_1 l}$, we obtain

$$\frac{w_{2m}}{w_{\alpha_2 2l}} \leqslant \frac{w_{2m} w_{\alpha_1 il}}{w_{\alpha_1 2l} w_{\alpha_2 il}}, \quad i \neq 2. \tag{26}$$

Now substitute the inequality from Eq. (24) for w_{2m} on the right-hand side of Eq. (26) and we have

$$\frac{w_{2m}}{w_{\alpha_2 2l}} \leqslant \frac{w_{im}}{w_{\alpha_2 il}}, \quad i \neq 2, \tag{27}$$

or the minimum in Eq. (22) is also in the second position.

Now we can show that $w^*_{\alpha_2 il} = \tilde{w}^*_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$. Start with $w^*_{\alpha_2 il} = K_{\alpha_2 l} w_{\alpha_2 il}$, $1 \leqslant i \leqslant 3$, substitute $K_{\alpha_2 l} = w^*_{\alpha_1 2l}/w_{\alpha_2 2l}$, $w^*_{\alpha_1 2l} = K_{\alpha_1 l} w_{\alpha_1 2l}$, $K_{\alpha_1 l} = w_{2m}/w_{\alpha_1 2l}$ and we obtain $(w_{2m}/w_{\alpha_2 2l}) w_{\alpha_2 il}$ which is $\tilde{w}^*_{\alpha_2 il}$.

Note that this means that $w^*_{\alpha_1 2l} = w^*_{\alpha_2 2l} = w_{2m}$. Let us generalize for $n \geqslant 3$ and α-cuts for $1 > \alpha_1 > \alpha_2 > \cdots > \alpha_n = 0$. The sufficient condition implies that the minimum always occurs in the same position, say the fourth position, for a certain fuzzy, positive, reciprocal matrix, which means that $w^*_{\alpha_1 4l} = w^*_{\alpha_2 4l} = \cdots = w^*_{\alpha_n 4l} = w_{4m}$. The sufficient condition also implies that our method is independent of the set of α's used in the α-cuts. Of course, we would also have a sufficient condition for the right end points of the α-cuts of the fuzzy weights. The sufficient condition for the right end points means that their minimum always occurs in the same position, say the sixth position, so that $w^*_{\alpha_1 6u} = w^*_{\alpha_2 6u} = \cdots = w^*_{\alpha_n 6u} = w_{6m}$.

Note that when we obtain equality with w_{im} the sufficient conditions holds and the method is independent of the set of α's used in the α-cuts. That is, if $w^*_{\alpha_1 2l} = w^*_{\alpha_2 2l} = \cdots = w^*_{\alpha_n 2l} = w_{2m}$, then the minimum was always in the second position.

In the application in the next section this is exactly what happens. That is, for each fuzzy, positive, reciprocal matrix there is an i so that $w^*_{\alpha il} = w_{im}$ for all α and there is a j so that $w^*_{\alpha ju} = w_{jm}$ for all α. In all the examples we have used for finding the fuzzy weights the sufficient condition holds.

We have no general proof at this time so we conjecture that for fuzzy, positive, reciprocal matrices the sufficient condition holds and our method is independent of the set of α's used in the α-cuts. This is a topic for future research.

4. Application

This application has been developed from an example in [22,23,6]. A recent graduate has been offered three jobs A_1, A_2, A_3. In order to rank these jobs, he evaluates each job with respect to five criteria:

(1) $C_1 =$ pay; (2) $C_2 =$ benefits; (3) $C_3 =$ location; (4) $C_4 =$ colleagues (fellow workers); and (5) $C_5 =$ potential for advancement. Using FHA, he constructs the following fuzzy reciprocal matrices:

$$\bar{A}_1 = \begin{bmatrix} 1 & (3/3,5/5)^{-1} & 1/2 \\ (3/3,5/5) & 1 & (2/3,3/4) \\ 2 & (2/3,3/4)^{-1} & 1 \end{bmatrix}$$

for $C_1 =$ pay,

$$\bar{A}_2 = \begin{bmatrix} 1 & (2/3,3/4)^{-1} & (2/3,3/4)^{-1} \\ (2/3,3/4) & 1 & 1 \\ (2/3,3/4) & 1 & 1 \end{bmatrix}$$

for $C_2 =$ benefits,

$$\bar{A}_3 = \begin{bmatrix} 1 & 1 & (7/7,8/10) \\ 1 & 1 & (7/8,9/10) \\ (7/7,8/10)^{-1} & (7/8,9/10)^{-1} & 1 \end{bmatrix}$$

for $C_3 =$ location,

$$\bar{A}_4 = \begin{bmatrix} 1 & (1/3,3/3)^{-1} & (2/2,2/5) \\ (1/3,3/3) & 1 & (6/7,7/8) \\ (2/2,2/5)^{-1} & (6/7,7/8)^{-1} & 1 \end{bmatrix}$$

for $C_4 =$ colleagues,

$$\bar{A}_5 = \begin{bmatrix} 1 & (4/4,4/6)^{-1} & (3/4,5/5)^{-1} \\ (4/4,4/6) & 1 & 1 \\ (3/4,5/5) & 1 & 1 \end{bmatrix}$$

for $C_5 =$ potential for advancement, and

$$\bar{E} = \begin{array}{c} \\ P \\ B \\ L \\ Co \\ Av \end{array} \begin{array}{c} \begin{array}{ccccc} P & B & L & Co & Av \end{array} \\ \begin{pmatrix} 1 & (1/2,2/3) & (3/3,5/5)^{-1} & 1 & (4/4,6/6)^{-1} \\ (1/2,2/3)^{-1} & 1 & \frac{1}{6} & (1/2,4/5)^{-1} & \frac{1}{8} \\ (3/3,5/5) & 6 & 1 & 3 & 1 \\ 1 & (1/2,4/5) & \frac{1}{3} & 1 & \frac{1}{4} \\ (4/4,6/6) & 8 & 1 & 4 & 1 \end{pmatrix} \end{array}$$

for the criteria, where $P =$ pay, $B =$ benefits, $L =$ location, $Co =$ colleagues, and $Av =$ advancement. In the $\bar{A}_i$ matrices: (1) the first row/column corresponds to alternative A_1; (2) the second row/column is A_2; and (3) the third row/column is for job A_3. Using our algorithm we compute the fuzzy weight vectors $\bar{w}_k$ for $\bar{A}_k$, $1 \leqslant k \leqslant 5$, and $\bar{e}$ for $\bar{E}$. Then from Eq. (3) we get

$$\bar{r}_j = \sum_{k=1}^{5} \bar{w}_{jk} \bar{e}_k \tag{28}$$

for all j. The fuzzy weight for job A_j is $\bar{r}_j$. However, before showing the results we need to discuss consistency and the ranking of fuzzy numbers.

4.1. Consistency

Let A be a positive, reciprocal matrix. A is said to be consistent when $a_{ik}a_{kj}=a_{ij}$ for all i,j,k. This means that if the judge states that $a_{ik}=2/1$ for A_i versus A_k and gives $a_{kj}=3/1$ for A_k versus A_j, then to be logically consistent this judge should state 6/1 for A_i versus A_j. If A is consistent then $\lambda_{\max}=n$ and in general, $\lambda_{\max}\geqslant n$. So a measure of consistency is built around the difference $(\lambda_{\max}-n)$ (see [22,24]). We would say that A is "reasonably" consistent when $(\lambda_{\max}-n)$ is not too large (may be $\lambda_{\max}-n\leqslant 1$).

To talk about consistency for fuzzy, positive, reciprocal matrices we first need to define what is meant by $\bar{M}\geqslant\bar{N}$, $\bar{M}>\bar{N}$, and $\bar{M}\approx\bar{N}$, for two fuzzy numbers $\bar{M}$ and $\bar{N}$. Define (see [2,5])

$$v(\bar{M}\geqslant\bar{N})=\sup_{x\geqslant y}(\min\bar{M}(x),\bar{N}(y)).$$

We then write $\bar{M}>\bar{N}$ if $v(\bar{M}\geqslant\bar{N})=1$ and $v(\bar{N}\geqslant\bar{M})<\theta$ where θ is some fixed positive fraction less than one. Let us use $\theta=0.8$ in this paper. Next, we write $\bar{M}\approx\bar{N}$ when $\bar{M}$ is not greater than $\bar{N}$ and $\bar{N}$ is not greater than $\bar{M}$. Or, if

$$\min(v(\bar{M}\geqslant\bar{N}),v(\bar{N}\geqslant\bar{M}))\geqslant\theta,$$

then $\bar{M}\approx\bar{N}$. Finally, we say that $\bar{M}\geqslant\bar{N}$ if $\bar{M}>\bar{N}$ or $\bar{M}\approx\bar{N}$.

A fuzzy, positive, reciprocal matrix $\bar{A}=[\bar{a}_{ij}]$ is defined to be consistent when

$$\bar{a}_{ik}\cdot\bar{a}_{kj}\approx\bar{a}_{ij}$$

for all i,j,k. The following theorem was proven in [2].

Theorem 1. *Let $\bar{A}=[\bar{a}_{ij}]$ be a fuzzy, positive, reciprocal matrix with $\bar{a}_{ij}=(\alpha_{ij}/\beta_{ij},\gamma_{ij}/\delta_{ij})$. Choose $a_{ij}\in[\beta_{ij},\gamma_{ij}]$ and form $A=[a_{ij}]$. If A is consistent, then $\bar{A}$ is consistent.*

We shall not demand all $\bar{A}_k$ and $\bar{E}$ to be perfectly consistent. All we shall ask is that they be "reasonably" consistent. What this means is that each has an A, constructed as in Theorem 1, which is reasonably consistent.

If we look at the $\bar{A}_i$, $1\leqslant i\leqslant 5$, and $\bar{E}$ in the application, all are "reasonably" consistent. In fact, $\bar{A}_2$, $\bar{A}_3$, and $\bar{A}_5$ are consistent. Let us look at $\bar{A}_1$ to see how it is "reasonably" consistent. From Theorem 1, for $\bar{A}_1$ to be consistent we need $[\beta_{ij},\gamma_{ij}]\subset[\beta_{ik},\gamma_{ik}]\cdot[\beta_{kj},\gamma_{kj}]$, for all i,j,k. Consider $i=1$, $k=2$ and $j=3$.

We see that $[\beta_{13},\gamma_{13}]=1/2$, $[\beta_{12},\gamma_{12}]=[1/5,1/3]$ and $[\beta_{23},\gamma_{23}]=3$ and $1/2\notin[3/5,1]$. But since 1/2 is "reasonably" close to 3/5 we conclude that $\bar{a}_{12}\cdot\bar{a}_{23}$ is "reasonably" close to $\bar{a}_{13}$. In $\bar{A}_1$ we find that $\bar{a}_{ik}\cdot\bar{a}_{kj}$ is "reasonably" close to $\bar{a}_{ij}$ for all i,j,k and we conclude that $\bar{A}_1$ is "reasonably" consistent. We have no test for reasonable consistency for fuzzy, positive, reciprocal matrices as is used for crisp, positive, reciprocal matrices.

4.2. Ranking fuzzy numbers

We end up (Eq. (28)) with fuzzy numbers $\bar{r}_1,\ldots,\bar{r}_n$ which need to be ranked so that we may obtain the final ranking of alternatives. Let H_1 be all the undominated fuzzy numbers $\bar{r}_i$. We say that $\bar{r}_i$ is undominated if no $\bar{r}_j>\bar{r}_i$, $j\neq i$. Next, define H_2 to be all the undominated $\bar{r}_k$ after deleting all the fuzzy numbers in H_1. Similarly, we construct $H_3,\ldots,H_d$. Then, all the A_i corresponding to an $\bar{r}_i$ in H_1 have the highest ranking, all the A_j having $\bar{r}_j$ in H_2 have the second ranking, etc. Properties of this ranking method are given in [2,5].

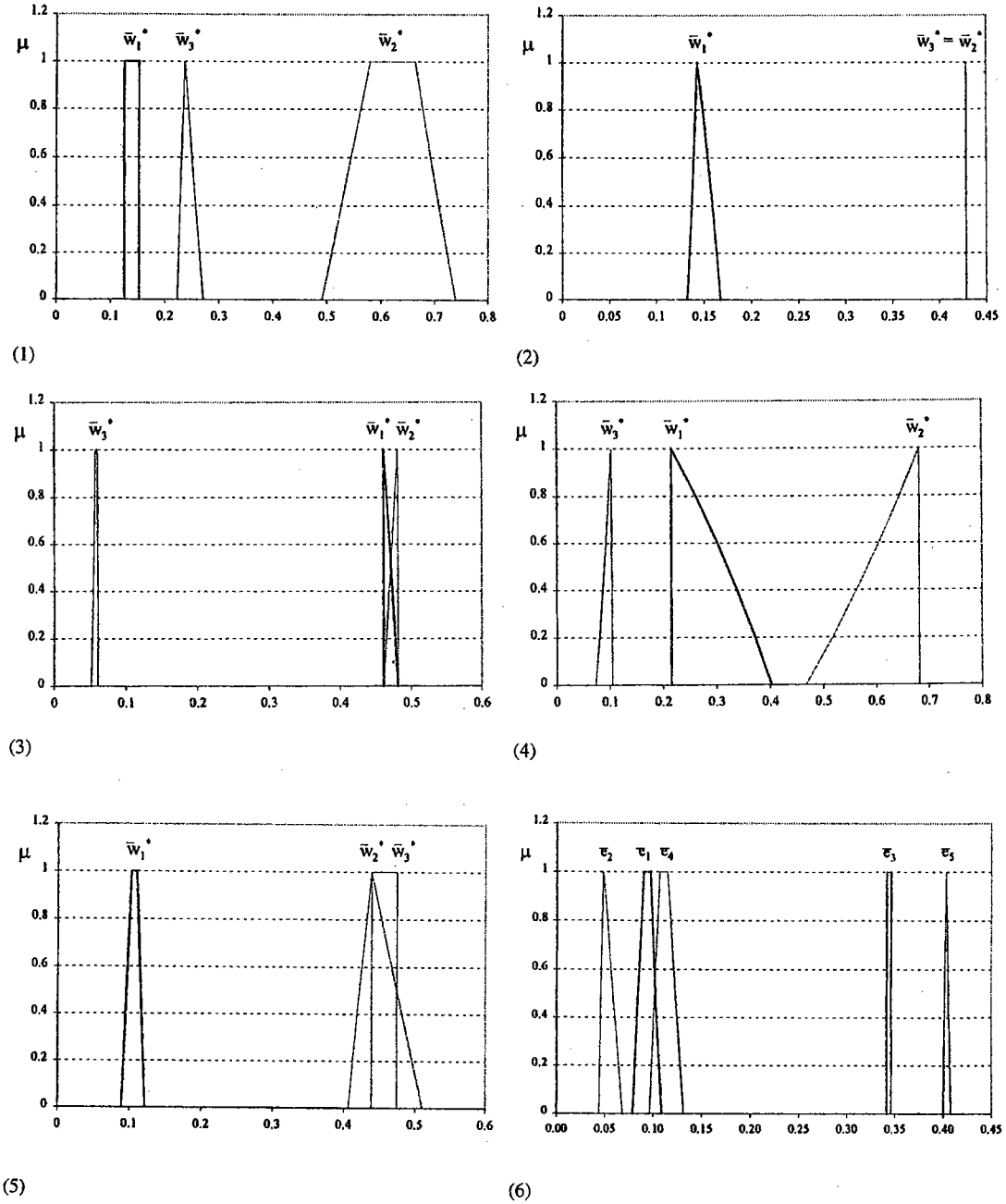


Fig. 6. Fuzzy weights of $\tilde{A}_k$ and $\tilde{E}$: (1) Fuzzy weights of pay; (2) Fuzzy weights of benefits; (3) Fuzzy weights of location; (4) Fuzzy weights of colleagues; (5) Fuzzy weights of advancement; (6) Fuzzy weights of criteria.

Table 1
Comparison of the calculated fuzzy weights between the Lambda-Max method and Ref. [6]

	Lambda-Max method		Method in Ref. [6]	
	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$
(1) *Fuzzy weights for* $\bar{A}_1$ (*pay*)				
$\bar{w}_{11}^*$	[0.1253, 0.1529]	[0.1253, 0.1529]	[0.1158, 0.1630]	[0.1219, 0.1549]
$\bar{w}_{21}^*$	[0.4910, 0.7395]	[0.5805, 0.6639]	[0.5401, 0.6849]	[0.5973, 0.6486]
$\bar{w}_{31}^*$	[0.2230, 0.2705]	[0.2371, 0.2371]	[0.1990, 0.2967]	[0.2296, 0.2478]
(2) *Fuzzy weights for* $\bar{A}_2$ (*benefit*)				
$\bar{w}_{12}^*$	[0.1324, 0.1673]	[0.1429, 0.1429]	[0.1111, 0.1996]	[0.1429, 0.1429]
$\bar{w}_{22}^*$	[0.4286, 0.4286]	[0.4286, 0.4286]	[0.3776, 0.4754]	[0.4286, 0.4286]
$\bar{w}_{32}^*$	[0.4286, 0.4286]	[0.4286, 0.4286]	[0.3776, 0.4753]	[0.4286, 0.4286]
(3) *Fuzzy weights for* $\bar{A}_3$ (*location*)				
$\bar{w}_{13}^*$	[0.4611, 0.4808]	[0.4611, 0.4613]	[0.4443, 0.4999]	[0.4509, 0.4706]
$\bar{w}_{23}^*$	[0.4611, 0.4808]	[0.4808, 0.4808]	[0.4442, 0.4999]	[0.4706, 0.4902]
$\bar{w}_{33}^*$	[0.0517, 0.0607]	[0.0578, 0.0603]	[0.0476, 0.0666]	[0.0556, 0.0625]
(4) *Fuzzy weights for* $\bar{A}_4$ (*colleagues*)				
$\bar{w}_{14}^*$	[0.2158, 0.4020]	[0.2158, 0.2158]	[0.2101, 0.4406]	[0.2158, 0.2158]
$\bar{w}_{24}^*$	[0.4674, 0.6817]	[0.6817, 0.6817]	[0.4723, 0.6945]	[0.6818, 0.6818]
$\bar{w}_{34}^*$	[0.0728, 0.1049]	[0.1025, 0.1025]	[0.0653, 0.1184]	[0.1024, 0.1024]
(5) *Fuzzy weights for* $\bar{A}_5$ (*advancement*)				
$\bar{w}_{15}^*$	[0.0889, 0.1213]	[0.1022, 0.1097]	[0.0834, 0.1255]	[0.1002, 0.1111]
$\bar{w}_{25}^*$	[0.4388, 0.5105]	[0.4388, 0.4388]	[0.4332, 0.5000]	[0.4332, 0.4444]
$\bar{w}_{35}^*$	[0.4063, 0.4744]	[0.4388, 0.4744]	[0.3977, 0.4666]	[0.4444, 0.4660]
(6) *Fuzzy weights for* $\bar{E}$ (*criteria*)				
$\bar{e}_1$	[0.0790, 0.1090]	[0.0901, 0.0971]	[0.0720, 0.1158]	[0.0836, 0.1050]
$\bar{e}_2$	[0.0438, 0.0686]	[0.0478, 0.0485]	[0.0371, 0.0695]	[0.0430, 0.0525]
$\bar{e}_3$	[0.3415, 0.3458]	[0.3415, 0.3458]	[0.3183, 0.3670]	[0.3231, 0.3661]
$\bar{e}_4$	[0.0966, 0.1316]	[0.1067, 0.1151]	[0.0872, 0.1328]	[0.0994, 0.1215]
$\bar{e}_5$	[0.4002, 0.4079]	[0.4030, 0.4030]	[0.3816, 0.4276]	[0.3850, 0.4245]
(7) *Final fuzzy weights*				
$\bar{r}_1$	[0.2296, 0.2968]	[0.2398, 0.2504]	[0.2040, 0.3284]	[0.2220, 0.2694]
$\bar{r}_2$	[0.4357, 0.5743]	[0.4865, 0.5068]	[0.4008, 0.6018]	[0.4550, 0.5416]
$\bar{r}_3$	[0.2236, 0.2872]	[0.2493, 0.2677]	[0.2009, 0.3071]	[0.2369, 0.2817]

5. Results

Using our algorithm, we first found the fuzzy weight vectors $\bar{w}_k^*$ for $\bar{A}_k$ (Figs. 6(1)–(5)), $1 \leqslant k \leqslant 5$, and $\bar{e}$ for $\bar{E}$ (Fig. 6(6)). All fuzzy numbers were calculated for α-cuts of $\alpha = 0, 0.2, 0.4, 0.6, 0.8, 1.0$. Instead of displaying all this data we show, in the following Table 1, only $\alpha = 0$ and $\alpha = 1$. In Table 1, we also show the fuzzy weight vectors for the same two α-cuts calculated (with the help of an evolutionary algorithm) by the direct fuzzification of the original method used by Saaty in the AHP [6]. Note that the supports of the fuzzy weights are about the same for $\bar{A}_1$ and $\bar{A}_5$ but our $\lambda_{\max}$ method produced smaller supports for the fuzzy weights in $\bar{A}_2, \bar{A}_3, \bar{A}_4$ and $\bar{E}$. But what is more important is that the $\lambda_{\max}$ method gave smaller supports for the final fuzzy weights $\bar{r}_1, \bar{r}_2$ and $\bar{r}_3$. The final decision (which alternatives are the best?) is easier when the final fuzzy weights have smaller supports. The final fuzzy weights $\bar{r}_1, \bar{r}_2, \bar{r}_3$ are given in Fig. 7. We see that $H_2 = \{A_1, A_3\}$, $H_1 = \{A_2\}$ and the student selected A_2, the decision which is, in this case, identical with the result of [16].

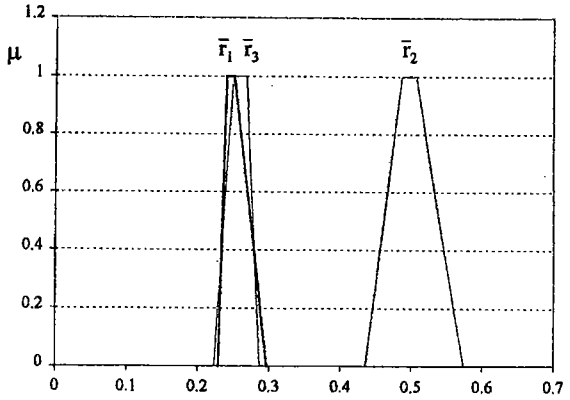


Fig. 7. The final fuzzy weights.

6. Summary and conclusions

In this paper we presented a method of performing fuzzy hierarchical analysis fuzzifying the λ_{max} method of Saaty. Important results of our new method are: (1) it can handle any type of fuzzy number used for pairwise comparisons; (2) it is computationally easy (does not require an evolutionary algorithm as in [6]) since all we need to compute are eigenvalues and eigenvectors of positive matrices; and (3) it reduced the fuzziness in the final fuzzy weights compared to the procedure used in [6]. Relative to the third point made above, our new method did not always reduce the fuzziness in the weights ($\bar{w}_{21}^*$ in Table 1(1), $\bar{w}_{25}^*$ in Table 1(5)), but the final fuzzy weights (Table 1(7)) had less fuzziness. A major topic for future research is to show that the sufficient condition (Section 3) always holds for fuzzy, positive, reciprocal matrices.

References

[1] C.G.E. Boender, J.G. deGraan, F.A. Lootsma, Multi-criteria decision analysis with fuzzy pairwise comparisons, Fuzzy Sets and Systems 29 (1989) 133–143.
[2] J.J. Buckley, Fuzzy hierarchical analysis, Fuzzy Sets and Systems 17 (1985) 233–247.
[3] J.J. Buckley, Fuzzy eigenvalues and input–output analysis, Fuzzy Sets and Systems 34 (1990) 187–195.
[4] J.J. Buckley, Solving fuzzy equations, Fuzzy Sets and Systems 50 (1992) 1–14.
[5] J.J. Buckley, V.R.R. Uppuluri, Fuzzy hierarchical analysis, in: V.T. Covello, L.B. Lave, A. Moghissi, V.R.R. Uppuluri (Eds.), Uncertainty and Risk Assessment, Risk Management and Decision Making, Plenum, New York, 1984, pp. 389–401.
[6] J.J. Buckley, T. Feuring, Y. Hayashi, Fuzzy hierarchical analysis revisited, European J. Oper. Anal., under revision.
[7] S.-M. Chen, Evaluating weapon system using fuzzy arithmetic operations, Fuzzy Sets and Systems 77 (1996) 265–276.
[8] C.-H. Cheng, Evaluating naval tactical missile system by fuzzy AHP based on the grade value of membership function, European J. Oper. Res. 96 (1996) 343–350.
[9] C.-H. Cheng, D.-L. Mon, Evaluating weapon system by analytical hierarchy process based on fuzzy scales, Fuzzy Sets and Systems 63 (1994) 1–10.
[10] M. Fedrizzi, R.A. Marques Pereina, Positive fuzzy matrices, dominant eigenvalues and an extension of Saaty's analytical hierarchy process, Proceedings of IFSA World Congress, Sao Paulo, Vol. II, Brazil, 1995, pp. 245–247.
[11] O. Gogus, T.O. Boucher, A consistency test for rational weights in multi-criterion decision analysis with fuzzy pairwise comparisons, Fuzzy Sets and Systems 86 (1997) 129–138.
[12] O. Gogus, T.O. Boucher, Strong transitivity and weak monotonicity in fuzzy pairwise comparisons, Fuzzy Sets and Systems 94 (1998) 133–144.
[13] C.-L. Hwang, K.-Yoon, Multiple Attribute Decision Making, Lecture Notes in Economics and Mathematical Systems, Vol. 186, Springer, Berlin, 1981.
[14] M. Kwiesielewicz, A note on the fuzzy extension of Saaty's priority theory, Fuzzy Sets and Systems 95 (1998) 161–172.
[15] P.J.M. Laarhoven, W. Pedrycz, A fuzzy extension of Saaty's priority theory, Fuzzy Sets and Systems 11 (1983) 229–241.

[16] F.A. Lootsma, Performance evaluation of nonlinear optimization methods via multi-criteria decision analysis and via linear model analysis, in: M.J.D. Powell (Ed.), Nonlinear Optimization, Vol. 1981, Academic Press, London, 1982, pp. 419–453.
[17] F.A. Lootsma, Rank preservation of propagation of fuzziness in pairwise-comparison methods for multi-criteria decision analysis, in: G. Fandel, M. Grauer, A. Kurzhanski, A.P. Wierzbicki (Eds.), Large-Scale Modeling and Interactive Decision Analysis, Springer, Berlin, 1986, pp. 127–137.
[18] B.K. Mohanty, N. Singh, Fuzzy relational equations in analytical hierarchy process, Fuzzy Sets and Systems 63 (1994) 11–19.
[19] D.-L. Mon, C.-H. Cheng, J.C. Lin, Evaluating weapon system using fuzzy analytic hierarchy process based on entropy weight, Fuzzy Sets and Systems 62 (1994) 127–134.
[20] X. Ruoing, Z. Xiaoyan, Extensions of the analytic hierarchy process in fuzzy environment, Fuzzy Sets and Systems 52 (1992) 251–257.
[21] X. Ruoing, Z. Xiaoyan, Fuzzy logarithmic least squares ranking method in analytical hierarchy process, Fuzzy Sets and Systems 77 (1996) 175–190.
[22] T.L. Saaty, A scaling method for priorities in hierarchical structures, J. Math. Psychol. 15 (1977) 234–281.
[23] T.L. Saaty, Exploring the interface between hierarchies, multiple objectives and fuzzy sets, Fuzzy Sets and Systems 1 (1978) 57–68.
[24] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.
[25] A.A. Salo, On fuzzy ratio comparisions in hierarchical decision models, Fuzzy Sets and Systems 84 (1996) 21–32.

論文編號：2

1 Introduction

Data mining has attracted a significant amount of research attention due to its usefulness in many applications, including selective marketing, decision support, business management, and user profile analysis, to name a few [6], [7], [11], [14]. Among others, data clustering is one of the most active research areas [16], [18]. The data clustering techniques can be used to perform similarity search, pattern recognition, trend analysis, grouping, classification, and so forth [7], [14], [26].

In general, there are two types of attributes associated with input data in clustering algorithms, i.e., numerical attributes [1], [4], [25], [29], [31] and categorical attributes [13], [30]. Numerical attributes are those with a finite or infinite number of ordered values, such as the height of a person or the x-coordinate of a point on a 2D domain. On the other hand, categorical attributes are those with finite unordered values, such as the occupation or the blood type of a person. In this paper, we focus only on the clustering of numerical data.

Many data clustering algorithms have been proposed in the literature. These algorithms can be categorized into nearest-neighbor clustering [22], fuzzy clustering [3], partitional clustering [9], [20], [23], hierarchical clustering [19], [27], artificial neural networks for clustering [15], statistical clustering algorithms [8], [28], density-based clustering algorithm [5], [10], and so on. In theses methods, hierarchical and partitional clustering algorithms are two primary approaches in research communities. Hierarchical clustering algorithms can usually find satisfiable clustering results. A hierarchical clustering algorithm is able to obtain different clustering results for different similarity requirements. However, most of those hierarchical algorithms are very computationally intensive and require much memory space. Both of the two well-known hierarchical algorithms, i.e., *single-link* [27] and *complete-link* [19] algorithms, require time complexity of $O(n^2 \log n)$, where n is the number of input points. Algorithm CURE [12], which is an improvement of the single-link algorithm, though being able to lead to good clustering results, is, in essence, very costly in its computational overhead.

On the other hand, most partitional clustering algorithms run in linear time. With better efficiency, the clustering quality of a partitional algorithm is, however, not as good as that of hierarchical algorithms. The *k-means* clustering algorithm is one of the most famous partitional clustering algorithms [23]. Although widely used, the k-means algorithm suffers from some drawbacks, including: 1) dependency on the input order, 2) the tendency to result in local minimum, and 3) limited applicability to only the data set consisting of isotropic clusters (i.e., a circle in the 2D domain or a sphericity in the 3D domain).

Several clustering methods have been proposed to combine the features of hierarchical and partitional clustering algorithms. In general, these algorithms first partition the input data set into m subclusters. Then, these algorithms construct a hierarchical structure based on these m subclusters. As shown in Fig. 1, the data set is first partitioned into 15 subclusters and these subclusters are next grouped into two clusters.

• *The authors are with the Electrical Engineering Department and Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, ROC. E-mail: {owenlin, mschen}@arbor.ee.ntu.edu.tw.*

Manuscript received 16 Aug. 2002; revised 21 Aug. 2003; accepted 9 Mar. 2004; published online 17 Dec. 2004.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 117145.

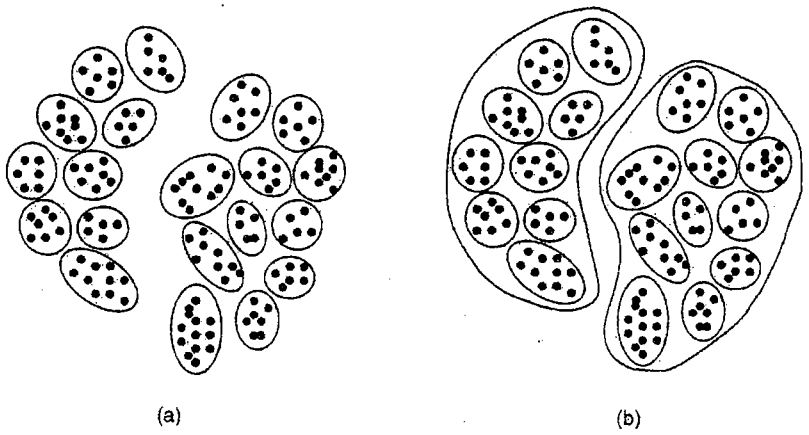


Fig. 1. An illustration of a hybrid clustering algorithm. (a) Step 1: Form several small clusters. (b) Step 2: Merge small subclusters into clusters.

This hybrid idea of clustering is first proposed in [24] where a multilevel algorithm is developed. At the first level, the multilevel algorithm partitions the data set into several partitions and then performs the k-means algorithm on each partition to obtain several subclusters. In subsequent levels, this algorithm uses the centroids of the subclusters identified in the previous level as the new input data points and performs the hierarchical clustering algorithm on those points. This process continues until exactly k clusters are determined. Finally, the algorithm performs a top-down process to reassign all points of each subcluster to the cluster of their centroids. However, representing a subcluster by only one point makes this multilevel algorithm not applicable to some cases, especially when the sizes of those subclusters are similar to that of the merged cluster.

Algorithm BIRCH is one of the most efficient clustering algorithms [31]. In BIRCH, only one scan is needed to obtain good clustering results. One or more additional passes can be used to further improve the clustering qualities. Algorithm BIRCH first partitions the input data set into many small subclusters and then applies a global clustering algorithm on those subclusters to achieve the final results. The main contribution of BIRCH is as an efficient data preprocessor for large input data sets. It does not focus on the design of global clustering algorithm, which can be effectively applied on subclusters. In view of this, a new definition of distances between two data bubbles (subclusters) are proposed in [5]. The algorithm proposed first executes BIRCH to obtain many small data bubbles and then applies a modified algorithm of OPTICS with the new definition of distance between data bubbles.

In most related studies, the dissimilarity between two clusters is defined as the distance between their centroids or the distance between two closest (or farthest) data points. However, these measures are vulnerable to outliers and removing outliers precisely is yet another difficult task. This is the very reason that most prior clustering methods do not perform stably for various types of inputs. In view of this, we propose a new similarity measure, referred to as cohesion, to measure the intercluster distances. Using cohesion, we have designed a two-phase clustering algorithm, called cohesion-based self-merging (abbreviated as CSM), which runs in time linear to the size of input data set. Combining the features of partitional and hierarchical clustering methods, algorithm CSM partitions the input data set into several small subclusters in the first phase and then continuously merges the subclusters based on cohesion in a hierarchical manner in the second phase. The time and the space complexities of algorithm CSM are analyzed. Our performance studies show that the proposed similarity measure, cohesion, is more effective than others. With cohesion, algorithm CSM is not only very robust to the existence of outliers, but also able to lead to better clustering results than most prior methods while incurring much shorter execution times.

The rest of the paper is organized as follows: Section 2 presents the preliminaries. Algorithm CSM is presented in Section 3. Performance studies are conducted in Section 4. This paper concludes with Section 5.

2 Preliminaries

Given a desired number of clusters k, data clustering is the process of partitioning the input data points into k clusters so that the points in each cluster are similar to one another and are different from the points in other clusters. For example, for the data set shown in Fig. 2a, a data clustering algorithm with $k = 2$ may partition these points into two clusters, $\{A, B, C, D\}$ and $\{E, F, G\}$. We review several prior algorithms related to this study in the following.

2.1 Hierarchical Clustering Algorithms

As its name implies, a hierarchical clustering algorithm establishes a hierarchical structure as the clustering result. Consider the example in Fig. 2a. One possible hierarchical structure is shown in Fig. 2b. With the hierarchical structure, we can obtain different clustering results for different similarity requirements. As shown in Fig. 2b, if the similarity requirement is set at *level 1*, the input data set is partitioned into two clusters, i.e., $\{A, B, C, D\}$ and $\{E, F, G\}$. However, if the similarity requirement is set at *level 2*, then the input data set is partitioned into six clusters, i.e., $\{A, B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{F\}$, and $\{G\}$.

Most existing hierarchical clustering algorithms are variations of the single-link and complete-link algorithms. Both algorithms require time complexity of $O(n^2 \log n)$,

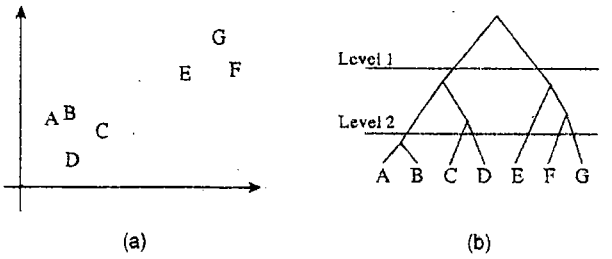


Fig. 2. Illustrative hierarchical clustering results for a data set of seven points. (a) The input data set. (b) A possible hierarchical tree.

where n is the size of the input data set. Owing to their good quality of clustering results, hierarchical algorithms are widely used, especially in document clustering and classification. The outline of a general hierarchical clustering algorithm is given below:

Hierarchical Clustering Algorithm

1. Initially, each data point forms a cluster by itself.
2. The algorithm repetitively merges the two closest clusters.
3. Output the hierarchical structure constructed.

A single-link clustering algorithm differs from a complete-link clustering algorithm in the intercluster distance measure, i.e., Step 2. The single-link algorithm uses the distance between the two closest points of the two clusters as the intercluster distance, i.e.,

$$dist(C_i, C_j) = \min\{dist(o_i, o_j) | o_i \in C_i, o_j \in C_j\},$$

while the complete-link algorithm uses the distance of two farthest points as the intercluster distance, i.e.,

$$dist(C_i, C_j) = \max\{dist(o_i, o_j) | o_i \in C_i, o_j \in C_j\}.$$

As shown in Fig. 3a, the single-link algorithm suffers from the so-called chaining effect and the complete-link clustering algorithm has problems in dealing with particular shapes, such as the circles shown in Fig. 3b.

In the single-link algorithm, we can maintain a pointer to the closest neighboring cluster for each cluster. After merging Cluster C_j into Cluster C_i, the single-link algorithm needs only to update those clusters whose closest neighbor is C_j and change it to C_i. With this implementation, the space complexity required by the single-link clustering algorithm is only $O(n)$. However, the same mechanism cannot be applied to the complete-link algorithm. Instead, the complete-link algorithm needs to keep all the distances of any two points and, thus, is of space complexity $O(n^2)$.

Algorithm CURE [12] is an improvement over the single-link clustering algorithm. CURE selects several scattered data points carefully as the representatives for each cluster and shrinks these representatives toward the centroid in order to eliminate the effects of outliers and avoid the chaining effect. The distance between two clusters in CURE is defined as the minimal distance between the two representatives of each cluster. In each iteration, it merges the two closest clusters.

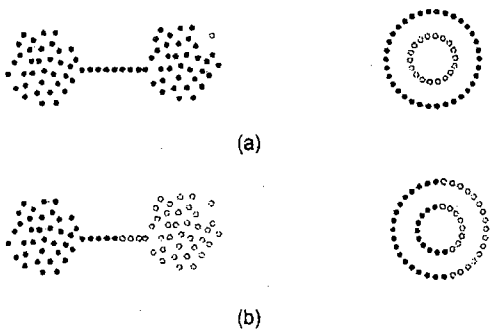


Fig. 3. Examples of the clustering capability of the single-link and complete-link clustering algorithms. (a) Clustering results by the single-link algorithm. (b) Clustering results of the complete-link algorithm.

It is known that CURE is one of the most successful clustering methods for clustering the data of any shape. However, it is very computationally intensive. The time complexity of algorithm CURE is essentially $O(n^2 \log n)$. In addition, algorithm CURE utilizes a spatial searching data structure, kd-tree, to search the nearest representatives. As pointed out in [2], this kind of searching structure does not work well in a high-dimensional data set. This fact limits the applicability of CURE. The sampling and partitioning technique adopted by algorithm CURE may accelerate the computation. However, most sampling techniques are orthogonal to clustering algorithms, i.e., a clustering algorithm can usually be easily integrated with most sampling techniques by no or slight modification. Please also note that, while being widely used, random sampling suffers from the problem of missing small clusters. In view of this, a density biased sampling algorithm is proposed in [10]. In this paper, we will focus on the clustering problem.

2.2 Partitional Clustering Algorithms

The k-means algorithm is one popular partitional algorithm for data clustering. However, the k-means algorithm suffers from the tendency of resulting in local minimum and is likely to obtain different clustering results with different initial states. The clustering results are also dependent on the sequence of the input data. The adoption of Euclidean distance as the similarity measure causes the application of the k-means algorithm to be limited to data sets consisting of only isotropic clusters, thus refraining it from being used in many real applications. Nevertheless, most partitional algorithms have the advantages on the execution time and the space required. The outline of the k-means algorithm is given as follows:

Algorithm K-Means

1. Initially, select k centroids arbitrarily for each cluster C_i, $i \in [1, k]$.
2. Assign each data point to the cluster whose centroid is closest to the data point.
3. Calculate the centroid c_i of cluster C_i, $i \in [1, k]$.
4. Repeat Step 2 and Step 3 until no points change between clusters.

2.3 Expectation Maximization Algorithms

In addition to using distance as the similarity measure, the Expectation Maximization (EM) algorithm uses probabilities to measure the similarities. It is assumed that the points of a cluster follow a certain distribution [8]. By assuming the parameters of the distribution of each cluster, EM utilizes probability to judge which cluster a data point should be assigned to. Algorithm EM then adjusts the parameters of each cluster's distribution according to the data points in that cluster. Next, it reassigns these points according to the new distributions. These iterations continue until the clustering results converge. For example, if the distribution of Cluster C_i follows a given probability density function (abbreviated as *pdf*) $f_{C_i}(v)$, then the probability for a point with position v to belong to this cluster is:

$$P(C_i|v) = \frac{P(v|C_i) \times P(C_i)}{P(v)} = \frac{P(C_i)}{P(v)} f_{C_i}(v).$$

If a point at location v is more likely to belong to Cluster C_i than to Cluster C_j, i.e., $P(C_i|v) > P(C_j|v)$, then this point will be assigned to Cluster C_i.

2.4 Hybrid Clustering Algorithms

Since the complexity of hierarchical data clustering is relatively high, several improved algorithms have been proposed, such as the hybrid clustering algorithm proposed in [24] and algorithm BIRCH [31]. The hybrid algorithm in [24] is a multilevel algorithm. In the first level, the input data set is divided into P_1 partitions equally. Then, the hybrid algorithm performs a k-means algorithm to get C_1 clusters in each partition. In level i, where $i > 1$, the centroids of the clusters obtained in level $i-1$ are taken to this level for merging. These centroids are partitioned into P_i partitions. In each partition, the hybrid algorithm performs a hierarchical clustering algorithm to get C_i clusters. This process continues until exactly k clusters are identified. Finally, the algorithm performs a top-down process to reassign all points of each subcluster to the cluster of their centroid. This algorithm is shown to be very efficient in both aspects of computation and memory space. However, using only one point to represent the whole cluster may easily lose some information about the distributions of clusters, which are important to the similarity of two clusters.

Another hybrid clustering algorithm, BIRCH, is designed to deal with large input data sets. BIRCH uses *cluster features* (CF) to represent a subcluster. Given the CF of a subcluster, one can obtain the *centroid*, *radius*, and *diameter* of that subcluster easily (in constant time). Furthermore, the CF vector of a new cluster formed by merging two subclusters can be directly derived from the CF vectors of the two subclusters by algebra operations. Algorithm BIRCH consists of four phases. In Phase 1, BIRCH partitions the input data set into many subclusters by a CF tree. In Phase 2, it reduces the size of the CF tree (i.e., the number of subclusters) in order to apply a global clustering algorithm in Phase 3 on those generated subclusters. In Phase 4, each point in the data set is redistributed to the closest centroids of the clusters produced in Phase 3. Among these phases, Phase 2 and Phase 4 are used to further improve the clustering quality and are thus optional. Algorithm BIRCH does not specify the global clustering algorithm. However, if a clustering algorithm, which can find clusters in any shape, is adopted in Phase 3, then Phase 4 will need some slight modifications to keep the shape information. More specifically, each cluster produced in Phase 3 should be represented by several points instead of only the centroid. However, the main contribution of BIRCH is as an efficient data preprocessor for a large input data set so that the global clustering algorithm can be executed efficiently.

Algorithm CHAMELEON is also a hybrid clustering algorithm [17]. The outline of the algorithm is as follows:

Algorithm CHAMELEON

1. Construct a k-nearest neighbor graph.
2. Partition the k-nearest neighbor graph into many small subclusters.
3. Merge those subclusters into final clustering results.

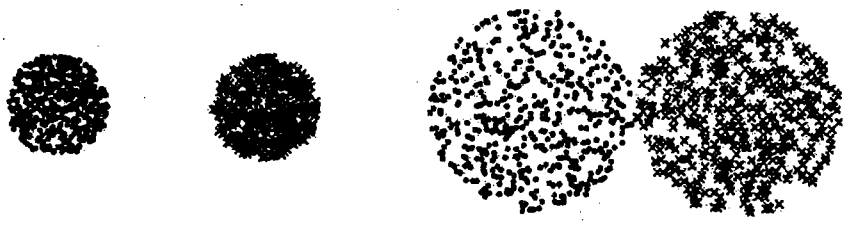
Note that the k-nearest-neighbor graph is built by connecting each node with its k nearest-neighboring nodes. Different from most of clustering algorithms, it considers not only the interconnectivity (the number of links between two clusters) but also the closeness (the length of those links) as the similarity measure of two clusters. Their experiments also show the excellent clustering quality. However, the time complexity of building a k-nearest-neighbor graph of a high-dimensional data set is as high as $O(n^2)$, which makes CHAMELEON infeasible for large data sets.

3 Cohesion-Based Self-Merging Algorithm

In this section, we describe the details of the cohesion-based self-merging algorithm (abbreviated as CSM). In Section 3.1, we propose a new measure for the similarity of two subclusters, *cohesion*, which is intrinsically different from other prior measures. Cohesion is more appropriate for an intercluster similarity measure because it does not judge the similarity of two subclusters by only some data points. Rather, the cohesion measure takes the distributions of the two clusters into account. In Section 3.2, based on cohesion, we devise a clustering algorithm, CSM, which fully utilizes the features of cohesion. We also describe a general outlier removal mechanism in Section 3.3 to enable algorithm CSM to resist outliers. (Note that the colored versions of some figures in this paper can be found at http://arbor.ee.ntu.tw/~owenlin/tkde_csm/.)

3.1 Similarity Measure between Subclusters

As shown in Fig. 4, the distance between the centroids of the two clusters in Fig. 4a and that of the two clusters in Fig. 4b are the same. The two clusters shown in Fig. 4b, however, are more inclined to be merged together. In addition to the distance between centroids, the average complete distance (i.e., $\frac{\sum d(p_i - p_j)}{|C_i| \times |C_j|}$ for each $p_i \in C_i$ and $p_j \in C_j$) is another method to measure the intercluster similarity of two clusters. However, with much more computation, the average complete distance cannot even distinguish between the two cases shown in Fig. 4. The average complete distance is 20.40 in Fig. 4a and 21.57 in Fig. 4b. (In contrast, as can be verified later, the corresponding cohesions are 2.21×10^{-11} in Fig. 4a and 2.17×10^{-4} in



(a) (b)

Fig. 4. Illustration for subclusters that have different cohesion values. (a) Two clusters with smaller cohesion. (b) Two clusters with greater cohesion.

Fig. 4b.) Several alternatives, such as the distance between the closest (or farthest) points of the two clusters, could be employed to redeem this deficiency. However, those measures are very vulnerable to random noises (outliers).

Consequently, we propose a new similarity measure, namely, *cohesion*, based on the *joinability* of two clusters, referring to the existence of a data point. Conceptually, *joinability* is the merging inclination of two clusters according to the existence of a shared data point. Thus, *joinability* is expected to have the following properties: 1) Data points located closer to the boundary of the two clusters are more important and 2) the merging inclination should not be determined by only a few points, i.e., the value of *joinability* should not vary dramatically. Formally, we have the following definition for *joinability*:

Definition 1. *The* joinability *of the two clusters (C_i and C_j) referring to the existence of the point p with location v is defined as:*

$$join(p, C_i, C_j) = \min\big(f_i(v), f_j(v)\big),$$

where f_i and f_j are the probability (density) function of the distributions in Cluster C_i and C_j, respectively.

An illustration of *joinability* is shown in Fig. 5. In Fig. 5, the joinabilities of the data points at v_1 and v_2 are j_1 and j_2, respectively. With the notion of *joinability*, the definition of *cohesion* of two clusters is given below:

Definition 2. *The* cohesion *of two clusters (C_i and C_j) is defined as*

$$chs(C_i, C_j) = \frac{\sum_{p \in C_i, C_j} join(p, C_i, C_j)}{|C_i| + |C_j|},$$

where $|C_i|$ is the size of Cluster C_i.

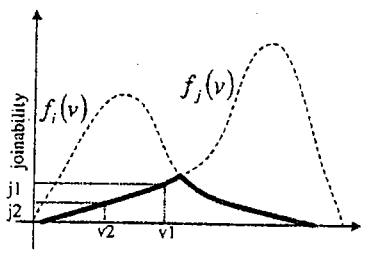


Fig. 5. An illustration for the meaning of *joinability*.

In general, the *pdf* $f(v)$ can be evaluated in constant time. Thus, the time complexity of the computation of cohesion of two clusters (C_i and C_j) is linear to the size of the two clusters, i.e., $O\big(|C_i| + |C_j|\big)$.

In this paper, we assume the location of a point in each cluster follows a multivariate normal distribution, i.e., $V \sim N_d(\mu, \psi)$, where d is the dimension of the space, μ is the mean vector, and ψ is the covariance matrix. The probability density function is

$$f(v) = (2\pi)^{-\frac{d}{2}} (\det \psi)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\Delta^2(v)\right],$$

where

$$\Delta^2(v) = (v - \mu)^T \psi^{-1} (v - \mu).$$

Please note that, in this formula, the location of a point (i.e., v_i) and the mean vector (i.e., μ) are both d-variate vectors and the covariance matrix (i.e., ψ) is a positive, definite $d \times d$ matrix.

Since the values of the mean vector and covariance matrix are unknown in advance, we use the maximum likelihood estimator of (μ, ψ) in practice. Given a cluster of n points with locations $(v_1, v_2, \ldots, v_n)$, the values of (μ, ψ) of the cluster are estimated by the following formulae:

$$\widehat{\mu} = \frac{1}{n} \sum_{i=1}^{n} v_i$$

and

$$\psi = \frac{1}{N} \sum_{i=1}^{n} (v_i - \widehat{\mu})(v_i - \widehat{\mu})^T.$$

This similarity measure of cohesion is robust to the existence of outliers due to the following two reasons: 1) Using the cohesion measure, instead of only a few of points, all points of the two subclusters are considered to evaluate this intercluster similarity and 2) this cohesion measure makes the effect of outliers much smaller than that of other points since outliers are much farther from the centroid of the two clusters. For example, for the four pairs of clusters shown in Fig. 6, cohesions between these pairs of clusters are 2.99×10^{-9} in Fig. 6a, 1.63×10^{-6} in Fig. 6b, 5.23×10^{-8} in Fig. 6c, and 8.52×10^{-9} in Fig. 6d. It can be noted that, comparing with the cohesion value in Fig. 6b, i.e., 1.63×10^{-6}, the cohesions of Fig. 6b and Fig. 6c are similar to the original one, i.e., cohesion is robust to the effects of outliers.

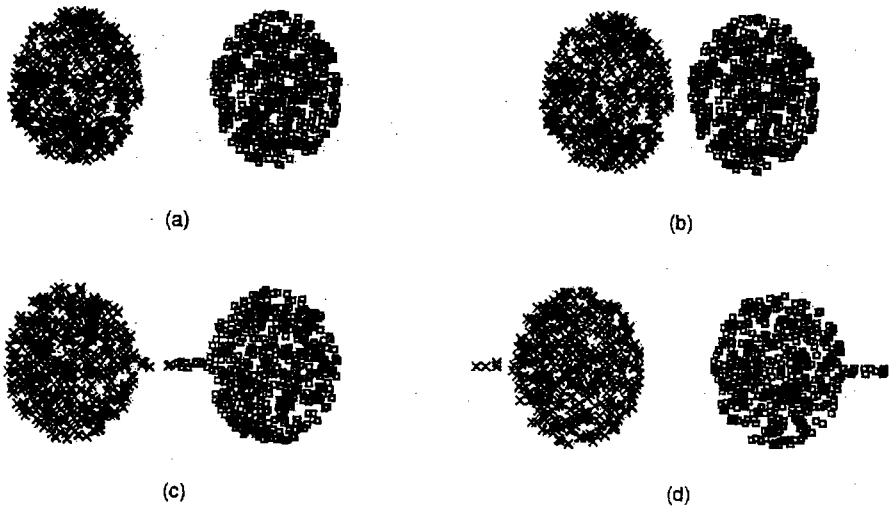


Fig. 6. An illustration of the robustness of the cohesion measure against outliers. (a) Two clusters. (b) Two closer clusters. (c) Two clusters with a link between them. (d) Two clusters with tails.

TABLE 1
All the Cohesions in Example 3.2

	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	Cl_6
Cl_1	–	2.32×10^{-25}	2.05×10^{-60}	4.88×10^{-21}	5.34×10^{-5}	3.74×10^{-5}
Cl_2	2.32×10^{-25}	–	6.31×10^{-5}	3.32×10^{-5}	6.36×10^{-12}	3.91×10^{-11}
Cl_3	2.05×10^{-60}	6.31×10^{-5}	–	4.11×10^{-5}	3.33×10^{-23}	2.51×10^{-19}
Cl_4	4.88×10^{-21}	3.32×10^{-5}	4.11×10^{-5}	–	5.82×10^{-12}	2.92×10^{-13}
Cl_5	5.34×10^{-5}	6.36×10^{-12}	3.33×10^{-23}	5.82×10^{-12}	–	4.08×10^{-5}
Cl_6	3.74×10^{-5}	3.91×10^{-11}	2.51×10^{-19}	2.92×10^{-13}	4.08×10^{-5}	–

3.2 Algorithm CSM

Now, we describe the proposed algorithm based on cohesion self-merging as follows:

Algorithm CSM

//Input: The input data set, the size of the data set, n, the number of subclusters, m, and the desired number of clusters, k.
//Output: The hierarchical structure of the k clusters.

1. Apply k-means on the input data set to obtain m subclusters.
2. Apply the single-link clustering algorithm on the m subclusters produced in Step 1 with cohesion as the similarity measure and stop when k clusters are obtained.

Example 3.2. Consider the data set shown in Fig. 7. We apply algorithm CSM with $k = 2$ and $m = 6$ to it. In the first phase, as shown in Fig. 7, algorithm CSM partitions the data set into six subclusters. In the second phase, algorithm CSM tries to merge the six subclusters into two clusters by their cohesions. The cohesions between these subclusters are shown in Table 1.

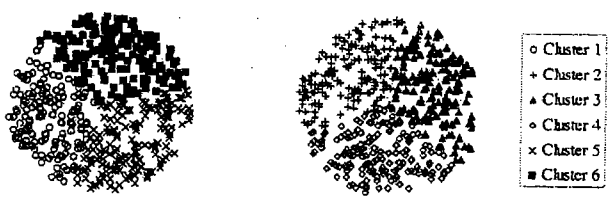


Fig. 7. An illustrative example of algorithm CSM.

Since the largest cohesion is $chs(Cl_2, Cl_3) = 6.31 \times 10^{-5}$, we first merge cluster Cl_2 and cluster Cl_3. Then, we find the next large cohesion, i.e., $chs(Cl_1, Cl_5) = 5.34 \times 10^{-5}$, and merge cluster Cl_1 and cluster Cl_5. Similarly, we merge Cl_3 and Cl_4 and then Cl_5 and Cl_6. Finally, the two clusters reached by CSM are $\{Cl_1, Cl_5, Cl_6\}$ and $\{Cl_2, Cl_3, Cl_4\}$.

Algorithm CSM is a two-phase clustering algorithm. In the first phase, it adopts the k-means algorithm to divide the input data set into m subclusters. At the beginning of Phase 2, it obtains the cohesions of these m subclusters produced in the first phase. Then, algorithm CSM performs a single-link clustering algorithm based on cohesion to obtain the final clusters.

In algorithm CSM, the parameter m, i.e., the number of subclusters, is the only additional parameter. We can obtain desired clustering results by adjusting the value of m. It is clear that the value of parameter m falls in the range of $[k, n)$. When $m = k$, algorithm CSM is degenerated to the k-means clustering algorithm. When m approaches n, this algorithm is reduced to the single-link algorithm. It is known that the k-means algorithm is good for obtaining clusters of isotropic shape, while the single-link algorithm is able to find clusters of any shape. With prior knowledge, we can make the clustering algorithm adapt to various inputs by adjusting the parameter m. As will be validated

by our performance studies later, algorithm CSM, in general, leads to better clustering results than those by most prior methods, showing not only the generality but also the advantage of algorithm CSM over the k-means and single-link methods.

3.3 Complexity Analysis

Theorem 1. *The time complexity of algorithm CSM is $O(mnl + m^2 \log m)$, where l is the number of iterations in the k-means algorithm.*

Proof. In Phase 1, algorithm CSM takes time of complexity $O(nml)$ to apply the k-means algorithm on the input data set to obtain m subclusters. Then, at the beginning of Phase 2, the values of cohesion between any two subclusters are evaluated. Recall that the time complexity of each evaluation is linear to the size of the clusters. Thus, the time complexity is

$$O\left(\sum_i \sum_{j>i} (|C_i| + |C_j|)\right) = O\left(\frac{1}{2}\sum_i \sum_{j\neq i} (|C_i| + |C_j|)\right)$$
$$= O\left(\frac{1}{2}\sum_i \left((m-2)|C_i| + \sum_j |C_j|\right)\right)$$
$$= O\left(\frac{1}{2}n(m-1)\right) = O(nm).$$

Next, algorithm CSM applies the single-link clustering algorithm to those subclusters and has time in order of $O(m^2 \log m)$. The time complexity of algorithm CSM is therefore $O(mnl + nm + m^2 \log m) = O(mnl + m^2 \log m)$. □

Note that the number of main iterations (i.e., l) in algorithm k-means is data dependent and determined empirically. However, experiments reveal that the k-means algorithm takes only a few iterations to converge.

Theorem 2. *The space complexity of the algorithm is $O(n)$.*

Proof. In the first phase, we only need the memory space to store the input data set and the group relationship of subclusters, which requires the memory space of complexity $O(n)$. In the second phase, memory space is required for the single-link clustering algorithm. Thus, the space complexity is $O(m)$. Since m is always smaller than n, the total space complexity of algorithm CSM is $O(n)$. □

3.4 Resilience to Outliers

Outliers are those random points that are very different from others and do not belong to any clusters. In algorithm CSM, we adopt cohesion as our similarity measure to resist the effects of outliers within a subcluster. However, it is possible that some subclusters consist only of outliers. We will get those noise clusters especially when we partition the input data set into too many subclusters. Those noise clusters will affect the correctness of the subsequent merging. Consider an example shown in Fig. 8. (One may note that the data sets in these figures are not identical to one another. This is because the data sets have been properly sampled in order to be clearly displayed.) In this example, algorithm CSM first partitions the input data set into 128 subclusters and then start to merge closest clusters pair by pair. The clustering process works well until those subclusters are merged into 12 clusters (as shown in Fig. 8a). However, in the next step, the upper two clusters are merged together undesirably (as shown in Fig. 8b) due to the existence of those noise clusters. Finally, those subclusters are merged into five clusters, as shown in Fig. 8c. However, after applying the noise removal mechanism introduced in this section, we can correctly identify the five clusters, as shown in Fig. 8d.

As shown in Fig. 9, there are two kinds of noise clusters, i.e., sparse noises and dense noises. Note that those sparse noise clusters are likely to be merged with others and, thus, may become bridges between subclusters which should be separated. For example, the sparse noise cluster shown in Fig. 9 may be merged with Cluster A and Cluster B. It also should be noted that the density of clusters could vary in different regions in a large data set. The noise clusters in one region are possibly even denser than some normal clusters in another region. Thus, it is hard to identify those sparse noise clusters. Instead of trying to identify those noises and remove them immediately, algorithm CSM first makes those sparse noises harder to join into normal clusters and then they can be removed with those dense noises together. (It is assumed that the number of points in a noise cluster is much less than that of a normal cluster.) Algorithm CSM adopts *density impedance* to achieve this. More specifically, given two clusters C_i and C_j, the *density impedance* of the two clusters is defined as

$$impedance(C_i, C_j) = \frac{C_i.density() + C_j.density()}{2\sqrt{C_i.density() \times C_j.density()}}.$$

Algorithm CSM then defines the similarity of two clusters as

$$\frac{cohesion(C_i, C_j)}{impedance(C_i, C_j)^{\alpha}},$$

where α is named the *impedance factor* and is specified by users to control the effect of impedance. Then, when those m subclusters are merged into m' subclusters, where m' is a predefined number such that $k < m' < m$, algorithm CSM will try to remove those sparse and dense noises clusters. The value of density impedance is only related to the density ratio of the two clusters. The relationship between them is shown in Fig. 10.

In order to quantify the meaning of sparseness, we first define the volume and the density of a cluster. Recall that the *pdf* of a multivariate normal distribution ($N_p(\mu, \psi)$) is

$$f(v) = (2\pi)^{-\frac{p}{2}} (\det \psi)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\Delta^2(v)\right],$$

where

$$\Delta^2(v) = (v-\mu)^T \psi^{-1} (v-\mu).$$

Thus, the contours of the *pdf* will satisfy the condition

$$(v-\mu)^T \psi^{-1} (v-\mu) = c.$$

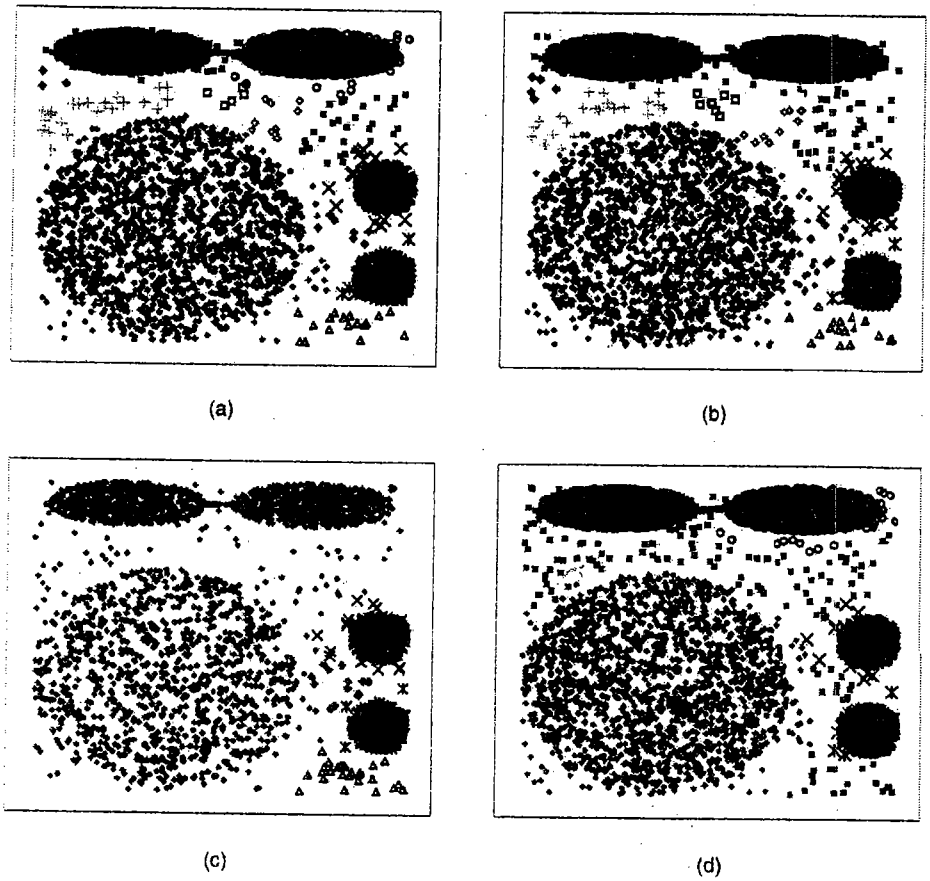


Fig. 8. An illustrative example of the effects of noise clusters. (a) Intermediate clustering state of 12 subclusters (five normal clusters and seven noise clusters). (b) Intermediate clustering state of 11 subclusters. (c) The final clustering result (three clusters are mixed together). (d) The clustering result after removing those noise clusters.

For simplicity, we assume $\mu = 0$ in the following discussion. Since ψ is a $d \times d$ symmetric matrix, there exist d real eigenvalues, i.e., $\lambda_1, \lambda_2, \ldots\ldots, \lambda_d$, and associated eigenvectors, i.e., $\beta_1, \beta_2, \ldots., \beta_d$, of unit length that can be chosen to be mutually orthogonal to one another. Next, making a $d \times d$ matrix $\mathbf{B}$ as $(\beta_1, \beta_2, \ldots., \beta_d)$, we have $\mathbf{B}^T\mathbf{B} = \mathbf{I}_d$, i.e., $\mathbf{B}^T = \mathbf{B}^{-1}$. Then, we change the coordinate in such a way that the new coordinate v' of vector v will satisfy the equation: $v = \mathbf{B}v'$. Note that this is a rigid transformation, i.e., all the angles and lengths will be preserved after the transformation. According to this transformation, we can present the function of the contour as $v'^T(\mathbf{B}^T\psi^{-1}\mathbf{B})v' = c$. Since matrix $\mathbf{B}$ consists of eigenvectors, we have

$$(\mathbf{B}^T\psi^{-1}\mathbf{B}) = \begin{bmatrix} \lambda_1^{-1} & 0 & 0 & 0 \\ 0 & \lambda_2^{-1} & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d^{-1} \end{bmatrix}.$$

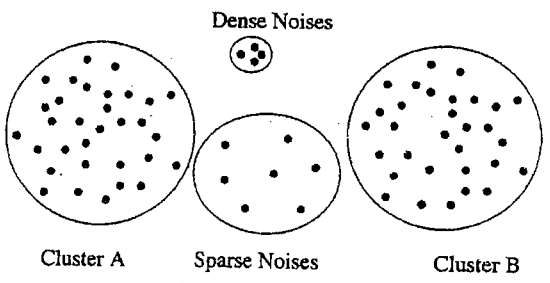


Fig. 9. The two kinds of noise clusters generated in Phase 1 of algorithm CSM.

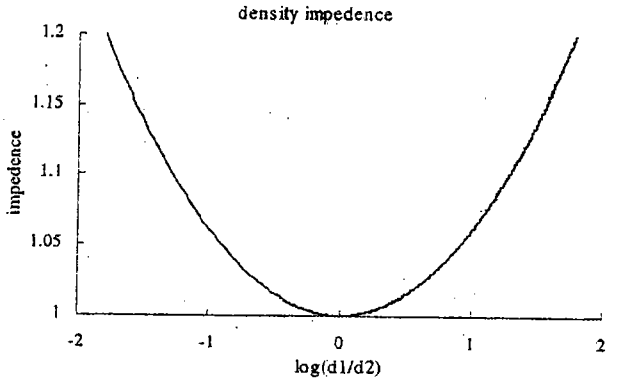


Fig 10. The density impedence of two clusters with density as d_1 and d_2.

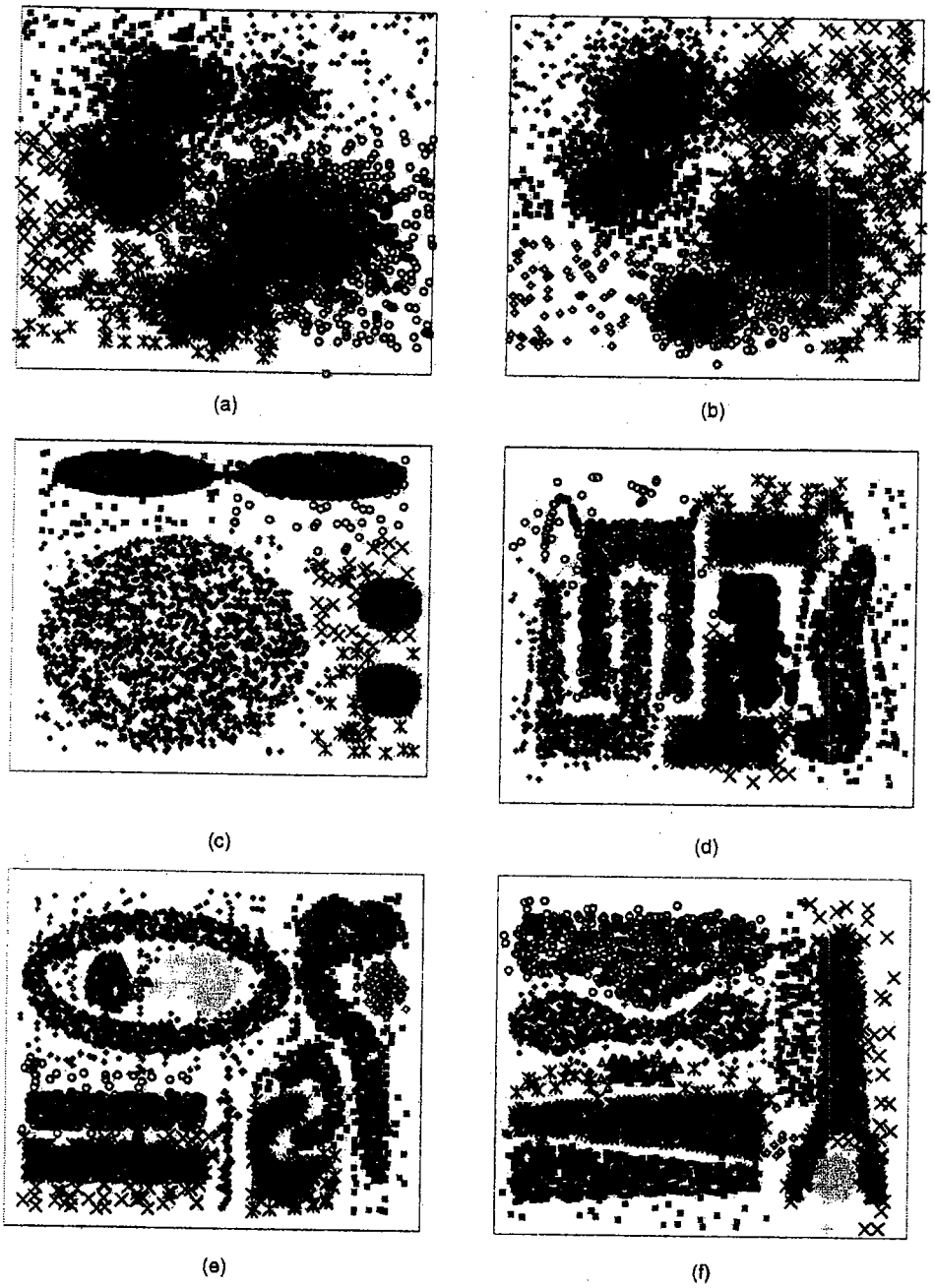


Fig. 11. The clustering results of algorithm CSM on the four input data sets. (a) CSM on Data Set 1. (b) CSM on Data Set 2. (c) CSM on Data Set 3. (d) CSM on Data Set 4. (e) CSM on Data Set 5. (f) CSM on Data Set 6.

As a result, the contour is a hyperellipse with $\sqrt{\lambda_1}$, $\sqrt{\lambda_2}, \ldots \sqrt{\lambda_d}$ as the length of each axis. Therefore, the volume and density of a cluster can be defined as follows:

Definition 3. *The volume of Cluster C_i is defined as*

$$C_i.volume = \sqrt{\lambda_1 \lambda_2 \ldots \lambda_d} = \sqrt{\det(\mathbf{B}^T \psi_i \mathbf{B})}$$
$$= \sqrt{\det \mathbf{B}^T \det(\psi_i) \det \mathbf{B}} = \sqrt{\det(\psi_i)},$$

where ψ_i is the covariance matrix of cluster C_i.

Definition 4. *The density of Cluster C_i is defined as*

$$C_i.density() = \frac{|C_i|}{C_i.volume},$$

where $C_i.volume$ is the volume of C_i and $|C_i|$ is the number of points in Cluster C_i.

The outlier removal mechanism is formally stated as follows. Note that the impedance factor, α, and *size_ratio* are two parameters specified by users.

TABLE 2
Execution Details on Each Data Set by CSM

	n	m	k	Average execution time
Data Set 1	10000	10	5	0.75 sec.
Data Set 2	10000	10	5	0.82 sec.
Data Set 3	10000	16	5	1.82 sec.
Data Set 4	8000	48	6	3.18 sec.
Data Set 5	10000	64	9	5.51 sec.
Data Set 6	8000	96	8	6.13 sec.

Outlier Removal Procedure

1. At Phase 2, algorithm CSM merges those subclusters by the similarity defined as

$$\frac{cohesion(C_i, C_j)}{impedance(C_i, C_j)^{\alpha}}.$$

2. When those m subclusters are merged into m' subclusters, algorithm CSM removes all the subclusters whose sizes are less than the threshold defined as

$$size\_ratio \times \frac{1}{m'} \sum |C_i|.$$

With proper parameter settings, algorithm CSM can precisely remove those noise clusters and result in clustering of better quality.

4 Performance Studies

To assess the performance of algorithm CSM, we have conducted a series of experiments. These experiments are performed on a computer with an 800 Mhz Intel CPU and 1.7G of memory. Several similarity measures are evaluated in Section 4.1. In Section 4.2, we compare the clustering quality of CSM with several prior clustering methods.

4.1 Experiment I: Comparing with Other Measures

We perform our experiments on the data sets shown in Fig. 11. Data Set 1 is generated from normal distribution and the sizes of clusters follow the Zipf distribution. Data Set 2 is in the same layout of Data Set 1. However, the density of clusters in Data Set 2 varies, while the density of clusters in Data Set 1 is uniform. Data Set 3 is the same one used in CURE [12]. As stated in [12], both BIRCH and single-link cannot partition this data set correctly. The other data sets are obtained from [17]. As stated in [17], both DBScan and CURE cannot successfully partition some of the data sets. The clustering results of algorithm CSM are shown in Fig. 11, while the details are shown in Table 2. Note that the values of parameter m are chosen so that algorithm CSM can obtain the similar clustering results each time. Recall that the algorithm is randomized. Thus, we execute the algorithm 20 times to obtain the average execution time. As shown in these figures, algorithm CSM is able to successfully partition these data sets. Note that the clustering results shown in this paper have been properly sampled in order to be clearly displayed. However, all algorithms are performed on the whole data sets.

First, cohesion is compared with other similarity measures, including those measures ($D0$, $D1$, $D2$, $D3$, $D4$) defined in [31] and the distance of data bubbles defined in [5]. We also define a new similarity measure based on the density decrement and obtain an improvement over the distance of data bubbles. These similarity measures are briefly defined below:

Definition 5. *The similarity measures $D0$, $D1$, $D2$, $D3$, and $D4$ between clusters C_i and C_j are defined in Table 3.*

In the definitions presented in Table 3, the notation c_i is the centroid of Cluster C_i, $c_i[k]$ is the value of c_i in the kth coordinate, $|C_i|$ means the number of points in Cluster C_i, and $var(C_i)$ is the variance of Cluster C_i, which is defined as $\sum_{p \in C_i} |p - c_i|^2$.

Definition 6. *The distance between two data bubbles in Euclidean vector space is defined as*

$$dist(C_i, C_j) = \begin{cases} |c_i - c_j| - (e_i + e_j) + nnDist(1, C_i) + nnDist(1, C_j) \\ \qquad \text{if } |c_i - c_j| - (e_i + e_j) \geq 0, \\ \max(nnDist(1, C_i),\ nnDist(1, C_j)) \text{ otherwise,} \end{cases}$$

where c_i is the centroid of Cluster C_i, e_i is extent of Cluster C_i, and $nnDist(k, C_i)$ is a function of the estimated average k-nearest-neighbor distance within the Cluster C_i. The definitions of e_i and $nnDist(k, C_i)$ are formulated below:

TABLE 3
The Definitions of Similarity Measures

Name	Definition
Centroid Euclidian Distance	$D0 = \lvert\vec{c_i} - \vec{c_j}\rvert$
Centroid Manhattan Distance	$D1 = \sum_{k=1}^{d} \lvert c_i[k] - c_j[k]\rvert$
Average Inter-cluster Distance	$D2 = \sqrt{\frac{\sum_{p_i \in C_i} \sum_{p_j \in C_j} \lvert p_i - p_j\rvert^2}{\lvert C_i\rvert \lvert C_j\rvert}}$
Average Intra-cluster Distance	$D3 = \sqrt{\frac{\sum_{p_i, p_j \in C_i \cup C_j} \lvert p_i - p_j\rvert^2}{(\lvert C_i\rvert + \lvert C_j\rvert)(\lvert C_i\rvert + \lvert C_j\rvert - 1)}}$
Variance Increase Distance	$D4 = var(C_i \cup C_j) - var(C_i) - var(C_j)$

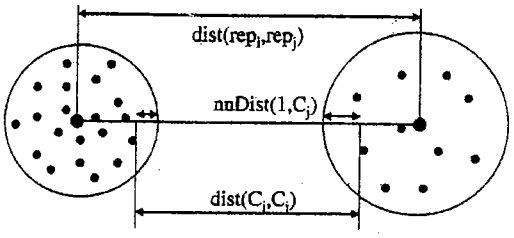


Fig. 12. Distance between two data bubbles.

$$e_i = \sqrt{\frac{\sum_{p_i,p_j \in C_i} |p_i - p_j|^2}{|C_i|(|C_i|-1)}} \text{ and}$$

$$nnDist(k, C_i) = \left(\frac{k}{|C_i|}\right)^{1/d} \times e_i,$$

where d is the dimension of the vector space.

As shown in Fig. 12, the distance between two data bubbles is defined under the assumption that all the data bubbles are spherical. However, this assumption does not always hold. With the covariance matrix of each data bubble (subcluster), we can extend the formula to elliptic data bubbles. This improvement can be made by changing the definition of *extent* as below.

Definition 7. *The extent e_i of an elliptic data bubble C_i along the direction of a unit length vector a is defined as*

$$e_i = 2\sqrt{\frac{1}{a^T \psi_i^{-1} a}},$$

where ψ_i, a $d \times d$ matrix, is the covariance matrix of the data bubble C_i.

As shown in Fig. 13, this formula is derived from the fact that the contours of the *pdf*, in a multivariate normal distribution, satisfies the condition $(v-\mu)^T \psi^{-1} (v-\mu) = c$. Thus, we have

$$(ta)^T \psi_i^{-1} (ta) = c \Rightarrow t^2 \times a^T \psi_i^{-1} a = c$$

$$e_i = t = \left(\frac{c}{a^T \psi_i^{-1} a}\right)^{\frac{1}{2}}.$$

Here, the value of parameter c is chosen as four so that the derived formula is consistent with the original one when the data bubbles are spherical.

Intuitively, if two neighboring clusters merge together, the density of the merged cluster will be expressed as

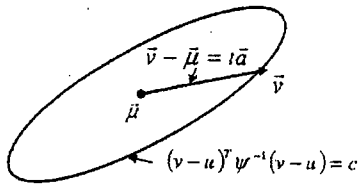


Fig. 13. The definition of extent in an oblique ellipse.

TABLE 4
Summary of Clustering Results of Different Measures

	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Cohesion	O	O	O	O	O	O
Cohesion$^-$	O	O	O	X	X	X
D0	O	O	X	X	X	X
D1	O	O	X	X	X	X
D2	O	O	X	X	X	X
D3	O	O	X	X	X	X
D4	O	O	X	X	X	X
Bubble	O	O	O	X	X	X
E-Bubble	O	O	O	O	X	X
Density	O	O	O	O	X	X

O is successful and X is failed.

$$\frac{|C_i| + |C_j|}{C_i.volume + C_j.volume.}$$

However, the density drops if the two clusters are far away from each other. Thus, we define a new similarity measure as the ratio of expected density over the estimated density.

Definition 8. *The density ratio of merging two clusters, C_i and C_j, is*

$$dd(C_i, C_j) = \frac{\frac{|C_i|+|C_j|}{C_i.volume+C_j.volume}}{(C_i \cup C_j).density} = \frac{(C_i \cup C_j).volume}{C_i.volume + C_j.volume}.$$

We also compare with the similarity measurement defined in our prior work [21], where the *joinability* of two clusters, C_i and C_j, according the existence of a point p_i in Cluster C_i, is defined as

$$join(p_i, C_i, C_j) = \exp\left(-\frac{|p_i - c_i| - |p_i - c_j|}{r_i}\right),$$

where c_i and c_j are the centroids of the two clusters and r_i is the radius of cluster C_i. We improve the definition of *joinability* because it is observed that, once the distance from the point to the two centroids is the same, the value of *joinability* becomes one. Therefore, the value of cohesion becomes large for two clusters that are far away from each other with some noise points located at the middle of the two clusters. Although very rare, the occurrence of this scenario is still deemed a defect and is thus remedied in this paper. In addition, by using multivariate normal distribution, the new definition of *joinability* is more powerful for handling elliptic clusters. We denote the distance measure defined in [21] as Cohesion$^-$.

We replace cohesion with these similarity measures in algorithm CSM and then apply it to the six data sets. The clustering results are summarized in Table 4, where the measure E-Bubble refers to the distance of elliptical data bubbles. As shown, only cohesion can successfully partition all the data sets. To fairly compare these similarity measures, we perform this experiment 20 times. Each time, we perform algorithm CSM with each similarity measure and a unique random number so that the subclusters generated in Phase 1 are identical for each measure. We use the probability of successful clustering to judge the quality of each similarity measure. The probabilities of some better similarity measures are shown in Fig. 15. Note that the success rates of these similarity measures on Data Set 1 and

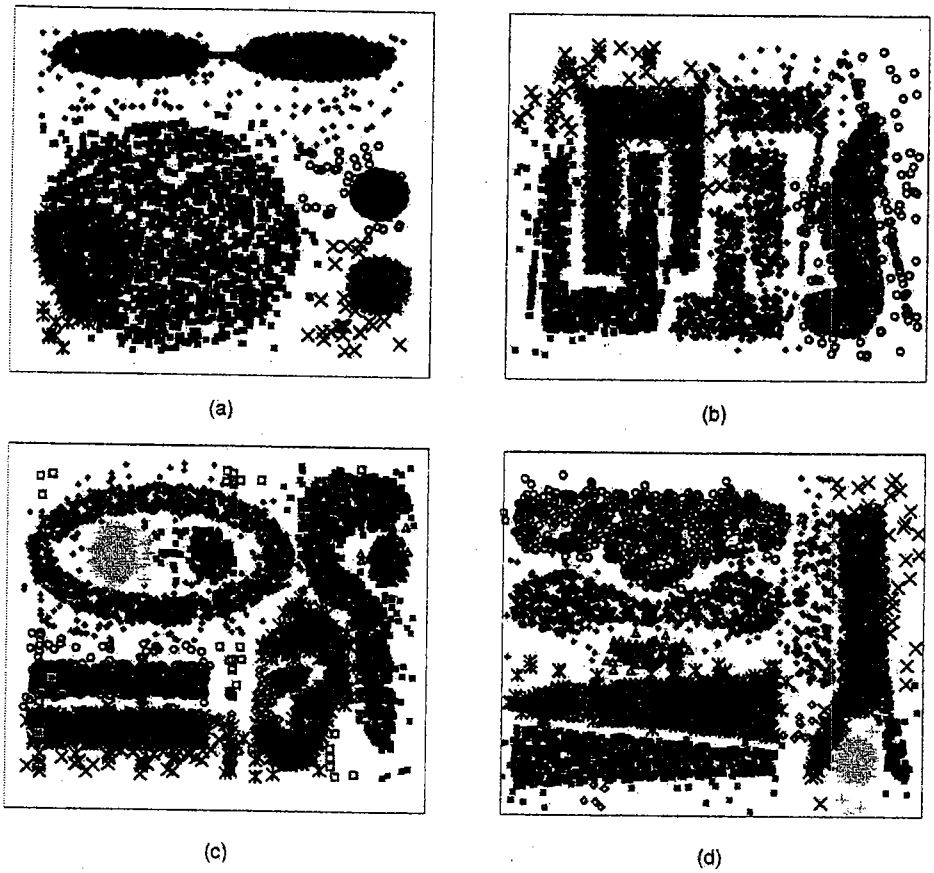


Fig. 14. Some clustering results of different similarity measures. (a) D0 on Data Set 3. (b) Bubble on Data Set 4. (c) Bubble on Data Set 5. (d) E-Bubble on Data Set 6.

Data Set 2 are all 100 percent and, thus, are omitted in Fig. 15. Some of those unsuccessful clustering results are shown in Fig. 14.

4.2 Experiment II: Comparing with Other Algorithms

We also apply other clustering algorithms on those data sets. The programs of algorithm BIRCH and DBScan are obtained from public domain. The k-means, single-link, and complete-link clustering algorithms are implemented as described in [19], [23], [27] using our best efforts. We also implement the algorithm CURE with the outlier elimination enhancement as described in [12]. For comparison reasons, we have carefully chosen the parameters for each algorithm. The clustering results are summarized in Table 5.

Some of unsuccessful clustering results are shown in Fig. 16. Note that the single-link algorithm is equipped with the outlier elimination enhancement as described in CURE. Thus, it can successfully partition Data Set 4. Otherwise, it will fail on all the data sets. In our observation, algorithm CURE fails on Data Set 5 and Data Set 6 because some clusters are in the shape of long stripes, while some noise links exist between neighboring clusters. Recall that algorithm CURE shrinks representatives toward the center of the cluster to avoid the link-effects. However, the shrink

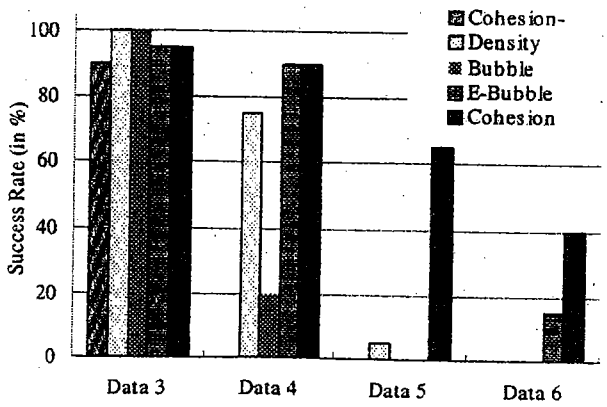


Fig. 15. The probabilities of successfully partitioned input data sets for some similarity measures.

TABLE 5
Summary of Clustering Results of Different Algorithms

	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
CSM	O	O	O	O	O	O
BIRCH	X	X	X	X	X	X
DBScan	O	X	X	O	O	X
CURE	O	O	O	O	X	X
K-means	O	O	X	X	X	X
Single-Link	X	X	X	O	X	X
Complete-Link	O	X	X	X	X	X

O is successful and X is failed.

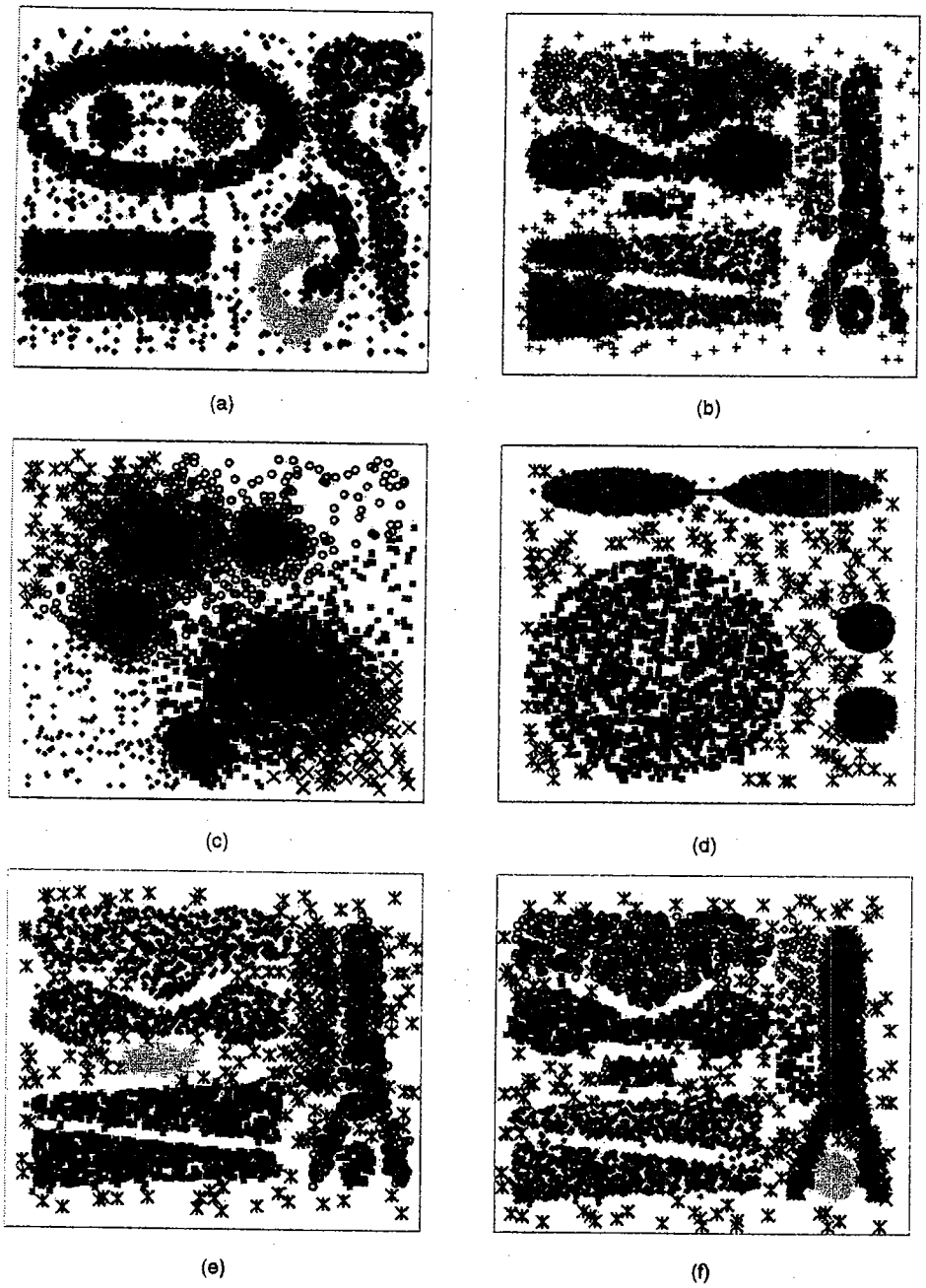


Fig. 16. Some clustering results of different algorithms. (a) Single Link on Data Set 5. (b) CURE on Data Set 6. (c) BIRCH on Data Set 1. (d) DBScan on Data Set 3. (e) DBScan on Data Set 6. (f) DBScan on Data Set 6.

mechanism will cause those slender clusters to be split. On the other hand, if we weaken the shrink mechanism, some clusters will be merged by noise links. In Data Set 4, algorithm CURE successfully finds a balanced shrink factor. However, it fails to find one in Data Set 5 and 6. Algorithm DBScan fails to find the five clusters in Data Set 2 because of the variety of density of those clusters. It also fails in Data Set 3 because there is a strong link between the upper two clusters. Note that the largest cluster in Data Set 1 is much sparser than the others. Thus, if we try to separate the upper two clusters by increasing the values of ε and/or $MinPts$, the other clusters will be merged with one another and/or the largest cluster will be regarded as noise. In Data Set 6, algorithm DBScan also faces a dilemma. As shown in Fig. 16e, it fails to separate two neighboring clusters linked by noise. If we try to separate them by increasing the values ε and/or $MinPts$, it will fail to identify the sparsest cluster, as shown in Fig. 16e. Among these algorithms, only algorithm CSM can obtain the correct clustering results.

Next, we conduct a scale-up experiment by applying these clustering algorithms on Data Set 3 of various sizes. Since the time complexities of these algorithms are much different from one another, we show the experimental results in Fig. 17a and Fig. 17b. For comparison reasons, we

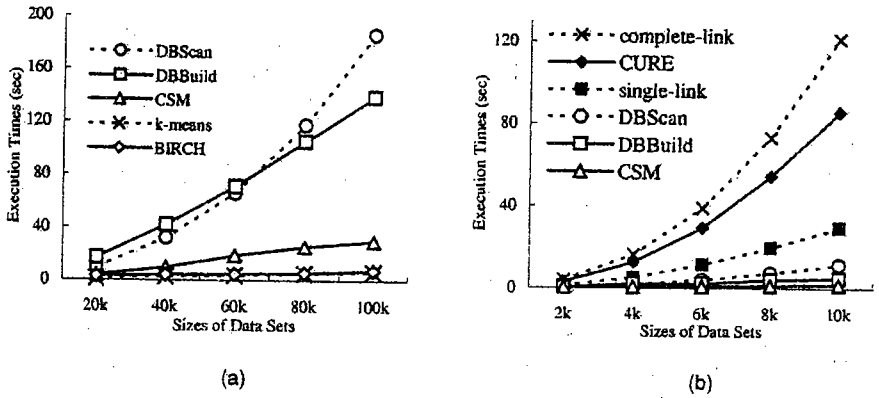


Fig. 17. Scale up experiment on the sizes of input data sets. (a) Comparison of faster algorithms. (b) Comparison of slower algorithms.

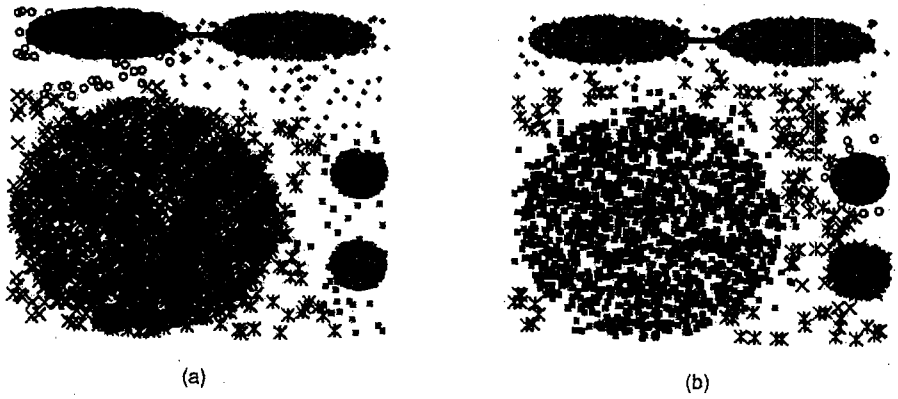


Fig. 18. The clustering results when m is too small or too large. (a) Small value of parameter m (10). (b) Large value of parameter m (256).

have tuned all the parameters so that the algorithm can be executed as fast as possible while accruing acceptable clustering qualities. For example, the value of parameter ϵ is 0.5 (the range of the Data Set 3 is 30×30), which is the smallest value that can successfully identify the four clusters (recall that algorithm DBScan fails to separate the upper two clusters). The value of parameter m in algorithm CSM is chosen as 16 and the number of representatives in algorithm CURE is chosen as 10. Note that the time required by algorithm DBScan to build its search structure, R^*-Tree, is also shown in the figure and denoted as DBBuild. Note that algorithm k-means and algorithm CSM are both randomized algorithms, thus we show the average execution times in these figures. As shown, algorithm CSM is faster than most algorithms (except for BIRCH and k-means) and scales linear to the size of input data set.

4.3 Experiment III: On the Number of Intermediate Clusters

Finally, we conduct a sensitivity analysis on the value of parameter m. It is observed that, when the value of m is too small, the subclusters produced in phase one may not properly partition the input data set, as shown in Fig. 18a. Thus, algorithm CSM results in an incorrect partition. On the other hand, if we partition the input data set into toomany subclusters, algorithm CSM may also fail to partition the input data set due to two problems. The first problem is the existence of many noise clusters. They will merge with other clusters and affect the clustering results. The second problem is those small clusters may form a link and connect two neighboring clusters. As shown in Fig. 18b, the upper two clusters are connected prior to the merging of the small subcluster and the left upper cluster. The first problem may be cured by our outlier resilience mechanism, while the second problem is hard to prevent when m is too large.

Next, we conduct a scale-up experiment on the value of parameter m. Specifically, we apply algorithm CSM on a $20k$-point subset of Data Set 3 with different values of parameter m. As shown in Fig. 19, algorithm CSM scales linear to the value of parameter m.

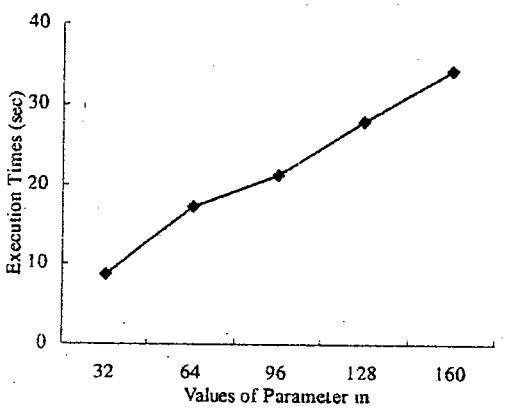


Fig. 19. The scale-up experiment on the value of parameter m.

5 Conclusion

In this paper, we propose a new similarity measure, cohesion, to measure the intercluster distances. By using cohesion, we proposed a two-phase clustering algorithm, CSM, whose time complexity is linear to the size of the input data set. Combining the features of partitional and hierarchical algorithms, algorithm CSM is able to not only resist outliers, but also lead to good clustering results while incurring a much shorter execution time than other algorithms. The time and the space complexities of CSM are also analyzed. As shown by our performance studies, the cohesion-based clustering is very robust and possesses excellent tolerance to outliers in various workloads. More importantly, algorithm CSM is shown to be able to cluster the data sets of arbitrary shapes very efficiently and to provide better clustering results than those by prior methods.

Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments to improve this paper. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC93-2752-E-002-006-PAE.

References

[1] C.C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, and J.-S. Park, "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD '99*, 1999.

[2] K.S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When Is 'Nearest Neighbor' Meaningful?" *Proc. Int'l Conf. Database Theory (ICDT '99)*, pp. 217-235, 1999.

[3] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[4] P.S. Bradley, K.P. Bennett, and A. Demiriz, "Constrained K-Means Clustering," Technical Report MSR-TR-2000-65, Microsoft Research, May 2000.

[5] M.M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander, "Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering," *Proc. ACM SIGMOD '01*, vol. 30, no. 2, pp. 79-90, 2001.

[6] A.G. Buchner and M. Mulvenna, "Discovery Internet Marketing Intelligence through Online Analytical Web Usage Mining," *Proc. ACM SIGMOD '98*, vol. 27, no. 4, pp. 54-61, Dec. 1998.

[7] M.-S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from Database Perspective," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, no. 1, pp. 866-883, Dec. 1996.

[8] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B*, vol. 39, no. 1, pp. 1-38, 1977.

[9] R.C. Dubes, "How Many Clusters Are Best?—An Experiment," *Pattern Recognition*, vol. 20, no. 6, pp. 645-663, 1987.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining*, pp. 226-231, 1996.

[11] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy, *Advances in Knowledge Discovery and Data Mining*. Cambridge, Mass.: MIT Press, 1996.

[12] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. Conf. Management of Data (ACM SIGMOD '98)*, pp. 73-84, 1998.

[13] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Proc. 15th Int'l Conf. Data Eng.*, 1999.

[14] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[15] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, Mass.: Addison-Wesley, 1991.

[16] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computer Surverys*, vol. 31, no. 3, Sept. 1999.

[17] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical Clustering Using Dynamic Modeling," *Computer*, vol. 32, no. 8, pp. 68-75, Aug. 1999.

[18] D.A. Keim and A. Hinneburg, "Clustering Techniques for the Large Data Sets—from the Past to the Future," *Tutorial Notes for ACM SIGKDD '99*, pp. 141-181, Aug. 1999.

[19] B. King, "Step-Wise Clustering Procedures," *J. Am. Statistical Assoc.*, vol. 69, pp. 86-101, 1967.

[20] C.-R. Lin and M.-S. Chen, "On the Optimal Clustering of Sequential Data," *Proc. Second Int'l Conf. Data Mining*, April 2002.

[21] C.-R. Lin and M.-S. Chen, "Robust and Efficient Clustering Algorithm Based on Cohesion Self-Merging," *Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining (ACM SIGKDD '00)*, Aug. 2002.

[22] S.Y. Lu and K.S. Fu, "A Sentence-to-Sentence Clustering Procedure for Pattern Analysis," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 8, pp. 381-389, 1978.

[23] J. McQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math. Statistics and Probability*, 1967.

[24] N.M. Murty and G. Krishna, "A Hybrid Clustering Procedure for Concentric and Chain-Like Clusters," *Int'l J. Computer and Information Sciences*, vol. 10, no. 6, pp. 397-412, 1981.

[25] R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94)*, 1994.

[26] J.O. Pedersen, D.R. Cutting, D.R. Karger, and J.W. Tukey, "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections," *Proc. 15th Int'l Conf. Research and Development in Information Retrieval (ACM)*, pp. 318-329, 1992.

[27] P.H.A. Sneath and R.R. Sokal, *Numerical Taxonomy*. London: Freeman, 1973.

[28] J. Tantrum, A. Murua, and W. Stuetzle, "Hierarchical Model-Based Clustering of Large Datasets through Fractionation and Refractionation." *Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining (ACM SIGKDD '02)*, Aug. 2002.

[29] A.K.H. Tung, J. Han, L.V.S. Lakshmanan, and R.T. Ng, "Constraint-Based Clustering in Large Databases," *Proc. Int'l Conf. Database Theory (ICDT '01)*, Jan. 2001.

[30] C.-H. Yun, K.-T. Chuang, and M.-S Chen, "An Efficient Clustering Algorithm for Market Basket Data Based on Small-Large Ratios," *Proc. 25th Int'l Conf. Computer Software and Applications*, Oct. 2001.

[31] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *Proc. Conf. Management of Data (ACM SIGMOD '96)*, pp. 103-114, 1996.