Original paper

Soft Computing 6 (2002) 428 – 435 © Springer-Verlag 2002
DOI 10.1007/s00500-001-0157-3

Fuzzy relation equations (I): the general and specialized solving algorithms

L. Chen, P. P. Wang

Abstract In this article, we develop a new method and an algorithm to solve a system of fuzzy relation equations. We first introduce a solution-base-matrix and then give a tractable mathematical logic representation of all minimal solutions. Next, we design a new universal algorithm to get them. Two simplification rules are found to simplify the solution-base-matrix. We show that a polynomial time algorithm to find all minimal solutions for a general system of fuzzy relation equations simply does not exist with expectation of $P = NP$. Hence, the problem of solving fuzzy relation equations is an *NP*-hard problem in terms of computational complexity. Our universal algorithm is still useful when one does not solve a large number of equations. In many real applications the problem of solving fuzzy relation equations can be simplified in polynomial time problems. In this article, we will discuss several cases of practical applications which have such polynomial algorithms.

Keywords Fuzzy relation equation, Algorithm, Polynomial time problem, *NP*-hard problem

1 Introduction

Solving fuzzy relation equations is a classic problem in fuzzy logic and system modeling using fuzzy methodology. The fuzzy relation equations have many applications in dynamic system control, taxonomy, classification, knowledge engineering and inferences [5–7, 10, 11].

For a (typical) set X, a subset of X, S, can be described as a function $\mu_S : X \to \{0, 1\}$, where $\mu_S(x) = 1$ if and only if $x \in S$. In 1965, L.A. Zadeh extended such binary representation of values to a continuous interval $\mu_S : X \to [0, 1]$. In such a representation, an element $x \in X$ is partially in S. Such kind of subset, S, is called a fuzzy subset of X. The concept of fuzzy subset has made a great impact to the modern science and engineering. Similarly, a fuzzy relation R can be viewed as an extension of a binary relation on the sets X and Y. In addition, R can also be represented by a matrix, namely a fuzzy relation matrix.

Similar to a system of linear equations in algebra, fuzzy relation equations are the mathematical formulation in the heart of fuzzy logic. We shall first review the concept of fuzzy relation.

A relation on the sets X and Y is a collection of pairs (a, b), where $a \in X$ and $b \in Y$. If X and Y are finite sets and $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, then a relation R can be represented by a binary ($\{0, 1\}$) matrix with n rows and m columns, $M_R = [r_{ij}]$ such that $(x_i, y_j) \in R$ if and only if $r_{ij} = 1$. In other words, a relation on X and Y is a mapping $R : X \times Y \to \{0, 1\}$. A fuzzy relation is defined as a mapping $R : X \times Y \to [0, 1]$. Or we can say a fuzzy relation R is a $n \times m$ matrix, where each entry has a real number between 0 and 1.

Let $A = (a_1, \ldots, a_n)$ be a fuzzy subset of the set X, denoted by $A \in F(X)$; $B = (b_1, \ldots, b_m) \in F(Y)$, and $R = [r_{ij}]_{n \times m} \in F(X \times Y)$. We call

$$A \circ R = B \tag{1}$$

a system of fuzzy relation equations. The problem is to calculate A if given R, B. The process of solving fuzzy relation equations is based on the operator "$\circ$" chosen. If "$\circ$" is a standard arithmetic "plus" and "product," the problem is to solve a standard system of linear equations. We know that we have the Gaussian elimination method. If we choose a standard fuzzy max–min composition operator, we have the well-known form of fuzzy relation equations. One also can use "max–product" and "plus–min" compositions.

More generally, given a t-norm T [6, 7, 10, 11], one can define Sup-T compositions that include both "max–min" and "max–product" compositions. More information on the general Sup-T compositions can be found in [2], and [3].

It is still a challenge problem to find a simple solving algorithm today. In order to focus on the algorithm issues, this paper will only consider the simplest composition, max–min operator. Fuzzy relation equations was first studied by Sanchez [12]. He showed that "If Equation (1) has a solution, then it must have the unique maximum solution." Some basic properties and characteristics on this issue were described in DiNola et al.'s book [5] and [2]. Some related research results can also be found in [3, 4, 13, 15] and [16].

Li Chen (✉)
Department of Computing and Information Systems, University of Luton, Luton, LU1 3JU, UK, and
Scientific & Practical Computing, P.O. Box 17, Centerville, UT 84014, USA
e-mail: moorechen@yahoo.com

Paul P. Wang
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

The authors wish to express their many thanks to the reviewers, especially to Professor S. Sessa for their important comments and suggestions.

According to the Sanchez's theorem, if a system of fuzzy relation equations has a solution, then the solution set is a partially ordered set. Therefore, the solution set of the given equations can be characterized by the maximum solution and all the minimal solutions.

Taking advantage of the Sanchez's theorem, to determine all the minimal (lower bounds) solutions is our goal. Therefore, solving the fuzzy relation equations turns out equivalently as an optimization problem.

Although there are many theoretical and practical results, the problem nevertheless is particularly attractive due to its potential importance in both theory and applications. Our aim here is to obtain some simple procedures in order to yield the solutions. The ultimate strategy, like the existing most solving procedures, is to simplify the original equations by "pushing" as far as we can.

In this article, based on Tsukamoto and Terano's solving method [14], we create a matrix, called the solution-base-matrix Q. We then use a logical expression which basically is analogous to Q. It is our intent to prove that each item of the logical expression would correspond to a minimal solution of the system of Eqs. (1).

According to Theoretical Computer Science, a problem is called solvable (feasible) if there is an algorithm to get the solution(s) with a finite number of steps or with limited time. Some problems are not feasible in Computer Science such as "Halting Problem," that is to determine when the machine stops [1, 8, 9]. For some problems, such as "Traveling Salesman Problem," to find the shortest cycle in traveling through all cities in a map, there is an algorithm which finds a solution in limited time. However, the time cost of the algorithm is exponentially growing when increases the number of cities (the size of the problem). For example, if the number of cities is 100, an $O(2^n)$ time algorithm may cost 100 years to obtain a solution. That is to say, a theoretically feasible problem may not be (practically) tractable [8].

There are two major classes of feasible problems: polynomial time class and *NP*-hard class. Basically, for a polynomial time problem, we have a polynomial (time) algorithm to solve it using our "modern" digital computers. The mathematical model of "modern" digital computers is the so called "Deterministic Turing Machine" [9]. A more powerful model of computers is "Nondeterministic Turing Machine." If a problem can be solved in a non-deterministic Turing machine in polynomial time, we call it a *NP*-problem. In the *NP*-problem class, there is a subclass called "*NP*-Complete class," where if any of them is in polynomial time class, then all *NP*-problems are in polynomial time class (Cook's theorem) [8], that is we have the famous unsolved "$P = NP$ problem." The Traveling Salesman Problem is such an *NP*-Complete problem. An *NP*-hard problem is defined as at least hard as an *NP*-complete problem.

Unfortunately, the problem of finding all the minimal solutions can be proved to be a *NP*-hard problem, that is to say: "There is no efficient algorithm to solve this problem unless $P = NP$." Thus, there is no simple procedure (such as the Gaussian elimination method in linear algebra) to solve the fuzzy relation equations. This result shows there exists a theoretical limitation to find all the minimal solutions for a very large system of fuzzy relation equations.

However, in many real world problems, the numbers of variables may not be extremely large. Furthermore, the real fuzzy relation equation may have a special structure such that a fast algorithm may exist.

After the theoretical development in Sect. 2, we give an $O(n!)$ time algorithm for the general fuzzy relation equations in Sect. 3. This algorithm is based on two rules which can be applied to the solution-base-matrix Q and its logical expression. When the number of variables n is not too big, say $n \leq 32$, the algorithm is still practical. To find all the minimal solutions by hand, unfortunately, is not realistic when $n > 5$ since the human's mind can easily make mistakes. This universal algorithm will be discussed extensively in Sect. 3. Furthermore, we have implemented this algorithm by C++ in Windows 95/NT, and the program gives us the correct answers for all examples we have tested. We believe this result is novel.

In Sect. 4, we will discuss several polynomial time algorithms. There we will prove that if the fuzzy relation matrix is k-diagonal, where k is a constant, then there is a polynomial time algorithm to obtain all the minimal solutions (if the number of minimal solutions is polynomially limited). We also consider other three practical problems, such as Fixed Priority Problem, Minimum Resources Problem and Restricted Subset Problem.

2
The minimal solutions of fuzzy relation equations

In this section, we show how to mathematically solve the fuzzy relation equations. We firstly introduce some basic concepts. Secondly, we introduce the solution-base-matrix Q of the Eqs. (1). We will prove that a "compatible" vector of Q is a solution of (1). We find that all solutions can be represented by a logical expression. We also show that the problem of solving fuzzy relation equations is *NP*-hard.

2.1
Basic concepts

Let S be the set of all vectors, each of which satisfies (1), S is called the solution set of (1). Let $A = (a_1, \ldots, a_n)$, $A' = (a'_1, \ldots, a'_n)$ be two vectors. If $a_i \leq a'_i$ for all $i = 1, \ldots, n$, then A is called to be "less than" A', denoted by $A \leq A'$. We also denote that $A + A' = (\max\{a_1, a'_1\}, \ldots, \max\{a_n, a'_n\})$. In addition, we have [5]:

Lemma 1 $(S, \leq)$ is an upper semi-lattice, i.e., if A, A' are in S, then $A + A'$ is in S.

Lemma 2 If A, A' are in S, then B is also in S if $A \leq B \leq A'$.

According to Sanchez's theorem [10, 12], $(S, \leq)$ has a maximum element $A_{\max}$ when S is not empty. Therefore, if we can obtain $A_{\max}$ and all the minimal elements of $(S, \leq)$, then we have completely the set S.

The set of the minimal elements of $(S, \leq)$ is denoted by Low(S).

For convenience, $b\varepsilon r$ represents the solutions of $x \cdot r = b$, and $b\hat{\varepsilon} r$ represents the solutions of $x \cdot r \le b$; where $x \cdot r$ is the min of x and r. Then we have,

$$b\varepsilon r = \begin{cases} b & \text{if } r > b \\ [\mathrm{b}, 1] & \text{if } r = b \\ \emptyset & \text{if } r < b \end{cases} \tag{2}$$

$$b\hat{\varepsilon} r = \begin{cases} [0, \mathrm{b}] & \text{if } r > b \\ [0, 1] & \text{if } r \le b \end{cases} \tag{3}$$

In (2), $\emptyset$ means the empty set. The following two theorems are variations of Sanchez's theorem:

Theorem 1 Let $\hat{S}$ be the set of the solutions X of the inequality $X \circ R \le B$. Then $x = (x_1, x_2, \ldots, x_n) \in \hat{S}$ if and only if

$$x_i \in \bigcap_{j=1}^{m} (b_j \hat{\varepsilon} r_{ij}), \quad \text{for } i = 1, \ldots, n \tag{4}$$

Theorem 2 If S is non-empty, then the maximum solution $A_{\max}$ of (1) equals the maximum solution $\hat{A}_{\max}$ of $A \circ R \le B$. Furthermore, the maximum solution is

$$A_{\max} = \hat{A}_{\max} = \left(\sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{1j})\right), \ldots, \sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{nj})\right)\right) \tag{5}$$

Example 1 Decide if the following fuzzy relation equations have solutions? Obtain the maximum solution if it exists.

$$(x_1 \quad x_2 \quad x_3 \quad x_4) \circ \begin{pmatrix} 0.3 & 0.5 & 0.7 & 0.9 & 0.8 \\ 0.2 & 0.4 & 0.3 & 0.6 & 0.5 \\ 0.7 & 0.4 & 0.2 & 0.1 & 0.6 \\ 0.8 & 0.9 & 0.7 & 0.2 & 0.4 \end{pmatrix} = (0.7 \quad 0.4 \quad 0.4 \quad 0.3 \quad 0.6) \tag{6}$$

According to (5), we have

$$\begin{aligned} \sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{1j})\right) &= \sup\{(0.7\hat{\varepsilon}0.3) \cap (0.4\hat{\varepsilon}0.5) \cap (0.4\hat{\varepsilon}0.7) \\ &\quad \cap (0.3\hat{\varepsilon}0.9) \cap (0.6\hat{\varepsilon}0.8)\} \\ &= \sup\{[0,1] \cap [0,0.4] \cap [0,0.4] \\ &\quad \cap [0,0.3] \cap [0,0.6]\} \\ &= 0.3 \end{aligned}$$

$$\begin{aligned} \sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{2j})\right) &= \sup\{[0,1] \cap [0,1] \cap [0,1] \\ &\quad \cap [0,0.3] \cap [0,1]\} \\ &= 0.3 \end{aligned}$$

$$\begin{aligned} \sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{3j})\right) &= \sup\{[0,1] \cap [0,1] \cap [0,1] \\ &\quad \cap [0,1] \cap [0,1]\} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \sup\left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{4j})\right) &= \sup\{[0,0.7] \cap [0,0.4] \cap [0,0.4] \\ &\quad \cap [0,1] \cap [0,1]\} \\ &= 0.4 \end{aligned}$$

We have got the vector $(0.3, 0.3, 1, 0.4)$. Substituting it in (6), we have:

$$(0.3 \quad 0.3 \quad 1 \quad 0.4) \circ \begin{pmatrix} 0.3 & 0.5 & 0.7 & 0.9 & 0.8 \\ 0.2 & 0.4 & 0.3 & 0.6 & 0.5 \\ 0.7 & 0.4 & 0.2 & 0.1 & 0.6 \\ 0.8 & 0.9 & 0.7 & 0.2 & 0.4 \end{pmatrix}$$

It is easy to verify the above form is equal to $(0.7, 0.4, 0.4, 0.3, 0.6)$. It is just the maximum solution of (6).

2.2 The base matrix of solutions

We now introduce the so called solution-base-matrix of (1) which forms the core concept developed in this paper.

Definition 1 The matrix Q is called a solution-base-matrix if

$$Q = [q_{ij}]_{n \times m} \tag{7}$$

where

$$q_{ij} = (b_j \varepsilon r_{ij}) \cap \left(\bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{ij})\right) \tag{8}$$

Definition 2 Let $t = (t_1, \ldots, t_n)$, where t_i is a real number. t is compatible with the solution-base-matrix Q if

$$\forall_j \exists_i (t_i \ne 0) \wedge (t_i \in q_{ij}), \quad \text{for } j = 1, \ldots, m \tag{9}$$

is true. In addition, let $T = \{t | t \text{ is compatible with } Q\}$.

The following theorem generates the connection between the solution-base-matrix Q and the solutions:

Theorem 3 Let $t \in T$. Define

$$t \,\hat{\cap}\, \hat{S} = \{x = (x_1, \ldots, x_n) | x_i \in (t_i \,\hat{\cap}\, \hat{S}_i) \quad \text{for } i = 1, \ldots, n\} \tag{10}$$

where $\hat{S}_i = \bigcap_{j=1}^{m}(b_j\hat{\varepsilon} r_{ij})$ and

$$t_i \,\hat{\cap}\, \hat{S}_i = \begin{cases} \hat{S}_i & \text{if } t_i = 0 \\ \min\{t_i, \sup(\hat{S}_i)\} & \text{otherwise} \end{cases} \tag{11}$$

Then $x \in (t \,\hat{\cap}\, \hat{S})$ is a solution of (1) and $S = \{x | x \in (t \,\hat{\cap}\, \hat{S}) \text{ where } t \in T\}$.

Proof. First, if $t \in T$ and $t' \in (t \,\hat{\cap}\, \hat{S})$, then, for all $j = 1, 2, \ldots, m$, we have

$$t'_1 r_{1j} + t'_2 r_{2j} + \cdots + t'_n r_{nj} \le b_j$$

According to the Definition 2, there must be an i such that $t_i \ne 0$ and $t_i \in q_{ij}$. So $t'_i = t_i$ or $t'_i = \sup(\hat{S}_i)$.

If $t'_i = t_i \neq 0$, $t'_i \in b_j \varepsilon r_{ij}$ by the definition of Q. Thus,

$$t'_1 r_{1j} + t'_2 r_{2j} + \cdots + t'_n r_{nj} = b_j$$

If $t'_i = \sup(\hat{S}_i)$, according to the Theorem 2, we have also the above equality. Therefore, t' is a solution of (1).

If t' is a solution of (1), we want to define $t = (t_1, \ldots, t_n)$ that is compatible with Q and such that $t' \in (t \hat{\cap} \hat{S})$. First, we have by Theorem 1:

$$t'_i \in \bigcap_{j=1}^{m} (b_j \hat{\varepsilon} r_{ij})$$

In addition, for every j, $j = 1, \ldots, m$, there must exist an i such that $t'_i \cdot r_{ij} = b_j$ in order to satisfy each equation:

$$t'_1 r_{1j} + t'_2 r_{2j} + \cdots + t'_n r_{nj} = b_j$$

In other words, there must be some i's such that $t'_i \in (b_j \varepsilon r_{ij})$. Then, for these i's, let $t_i = t'_i$. We do these assignments for every $j = 1, \ldots, m$.

It is still possible that we have not assigned any value to t_k for some k in $\{1, \ldots, n\}$. For those k's, we put $t_k = 0$. We can see that t is compatible with Q and $t' \in (t \hat{\cap} \hat{S})$.

Based on the proof of Theorem 3, we can also obtain:

Theorem 4 The minimal elements in T can be one-to-one mapping to the minimal elements in S.

Example 2 Calculate Q of the Eqs. (6).
According to Example 1, we have obtained: $\hat{s}_1 = [0, 0.3]$, $\hat{s}_2 = [0, 0.3]$, $\hat{s}_3 = [0, 1.0]$, and $\hat{s}_4 = [0, 0.4]$. So we have

$$Q = \begin{pmatrix} (0.7\varepsilon 0.3) \cap \hat{s}_1 & (0.4\varepsilon 0.5) \cap \hat{s}_1 & (0.4\varepsilon 0.7) \cap \hat{s}_1 & (0.3\varepsilon 0.9) \cap \hat{s}_1 & (0.6\varepsilon 0.8) \cap \hat{s}_1 \\ (0.7\varepsilon 0.2) \cap \hat{s}_2 & (0.4\varepsilon 0.4) \cap \hat{s}_2 & (0.4\varepsilon 0.3) \cap \hat{s}_2 & (0.3\varepsilon 0.6) \cap \hat{s}_2 & (0.6\varepsilon 0.5) \cap \hat{s}_2 \\ (0.7\varepsilon 0.7) \cap \hat{s}_3 & (0.4\varepsilon 0.4) \cap \hat{s}_3 & (0.4\varepsilon 0.2) \cap \hat{s}_3 & (0.3\varepsilon 0.1) \cap \hat{s}_3 & (0.6\varepsilon 0.6) \cap \hat{s}_3 \\ (0.7\varepsilon 0.8) \cap \hat{s}_4 & (0.4\varepsilon 0.9) \cap \hat{s}_4 & (0.4\varepsilon 0.7) \cap \hat{s}_4 & (0.3\varepsilon 0.2) \cap \hat{s}_4 & (0.6\varepsilon 0.4) \cap \hat{s}_4 \end{pmatrix} \tag{12}$$

Thus,

$$Q = \begin{pmatrix} \emptyset & \emptyset & \emptyset & 0.3 & \emptyset \\ \emptyset & \emptyset & \emptyset & 0.3 & \emptyset \\ [0.7, 1.0] & [0.4, 1.0] & \emptyset & \emptyset & [0.6, 1.0] \\ \emptyset & 0.4 & 0.4 & \emptyset & \emptyset \end{pmatrix} \tag{13}$$

2.3 The logical expression of solution-base-matrix

We now give a mathematical representation of all the minimal solutions based on the base matrix Q. We build a matrix $P = [P_{ij}]_{n \times m}$, where P_{ij} is a propositional variable. If $q_{ij} = \emptyset$ in Q, then $P_{ij} \equiv 0$ (which means always false).

Let

$$\Pi = \prod_{j=1}^{m} (P_{1j} + P_{2j} + \cdots + P_{nj}) \tag{14}$$

We can expand Π into a Disjunction Normal Form (DNF) [9]:

$$\Pi = \sum_{\xi_i \in \{1,2,\ldots,n\}} P_{\xi_1 1} \cdots P_{\xi_m m} \tag{15}$$

For each term in (15), collect the P_{ij}'s which have the same ξ_k. For instance, $P_{11}P_{22}P_{13}P_{14}P_{25}$ is substituted by $(P_{11}P_{13}P_{14})(P_{22}P_{25})$.

Any item of Π can be represented by

$$c = (P_{1k_1}, \ldots, P_{1k_{t_1}})(P_{2k_{t_1+1}}, \ldots, P_{2k_{t_2}}) \cdots (P_{nk_{t_{n-1}+1}}, \ldots, P_{nk_{t_n}}) \tag{16}$$

where $k_1 \cdots k_{t_1} \cdots k_{t_n}$ is a permutation of $1, 2, \ldots, n$.

Let

$$\Delta = \{c | c \text{ is an item of } \Pi\} \ . \tag{17}$$

Theorem 5 Each element in S can map naturally to an element of Δ.

Proof. Assume $s = (s_1, \ldots, s_n)$ is a solution of (1), i.e. $s \in S$. So s is compatible with Q by Theorem 4. Assign a value to each P_{ij} as follows:

$$P_{ij} = \begin{cases} 1 & \text{if } s_i \in q_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Such group of assignments makes that the value of Π is true. Thus, there must be an item in Π corresponding to the assignments. The theorem is proved.

Now, we define a partial ordering "$\leq$" on Δ. Let c be in Δ, $M_c = (M_c(1), \ldots, M_c(n))$ is a vector with n components, where

$$M_c(i) = \max\{\inf(q_{ik_{t_{i-1}+1}}), \ldots, \inf(q_{ik_{t_i}})\} \ . \tag{19}$$

Then, we define $c \leq c'$ in Δ if and only if for all i, $M_c(i) \leq M_{c'}(i)$. For convenience, we put $\max\{\emptyset, \ldots, \emptyset\} = 0$. It is clear that for all i, $M_c(i) = 0$ or $M_c(i) = b_j$ for some j. Thus $M_c \in S$.

Theorem 6 Let c be an element of Δ. Then c is minimal in Δ if and only if M_c is a minimal element of S.

Proof. Let c be a minimal element of Δ. Suppose that there is a minimal solution $s = (s_1, \ldots, s_n) \in S$ and

$$(s_1, \ldots, s_n) < (M_c(1), \ldots, M_c(n)) \tag{20}$$

then s is compatible with Q. If $s_i \neq 0$, then $s_i \in q_{ij}$ for some j. Assume there are k q_{ij}'s, such that

$$s_i \in q_{ij_1}, \ldots, s_i \in q_{ij_k}$$

i.e.

$$s_i \in \bigcap_{t=1}^{k} q_{ij_t}$$

So, for all i with $s_i \neq 0$, we have

$$s_i \geq \inf\left(\bigcap_{t=1}^{k} q_{ij_t}\right)$$
$$\geq \max\{\inf(q_{ik_{t_{i-1}+1}}), \ldots, \inf(q_{ik_{t_i}})\}$$

By (20), there must be an i such that

$$\max\{\inf(q_{ik_{t_{i-1}+1}})\}, \ldots, \inf(q_{ik_{t_i}})\} < M_c(i) \;,$$

that is there must be a c' in Δ such that $c' < c$, so c is not minimal in Δ. This is a contradiction.

Conversely, let $s = M_c$ be a minimal solution in S. For $s_i \neq 0$, there exist t_i q_{ij}'s, such that

$$s_i \in q_{ik_{t_{i-1}+1}} \cap \cdots \cap q_{ik_{t_i}}$$

i.e.

$$s_i \geq \max\{\inf(q_{ik_{t_{i-1}+1}}), \ldots, \inf(q_{ik_{t_i}})\}$$

Define a c' in Δ as follows:

$$c' = (q_{1k_1} \cdots q_{1k_{t_1}}) \cdots (q_{nk_{t_{n-1}+1}} \cdots q_{nk_{t_n}})$$

Thus

$$M_c(i) = s_i \geq M_{c'}(i) \quad \text{for all } i.$$

Since s is a minimal solution in S, then $M_c = s = M_{c'}$, that is $c = c'$ and therefore c is a minimal element of Δ.

Example 3 Calculate P and Π for Q of the Eqs. (6). According to the definitions of P, Π and Δ and Example 2, we can easily get

$$P = \begin{pmatrix} 0 & 0 & 0 & P_{14} & 0 \\ 0 & 0 & 0 & P_{24} & 0 \\ P_{31} & P_{32} & 0 & 0 & P_{35} \\ 0 & P_{42} & P_{43} & 0 & 0 \end{pmatrix} \tag{21}$$

$$\begin{aligned}\Pi(P) &= (P_{31})(P_{32} + P_{42})(P_{43})(P_{14} + P_{24})(P_{35}) \\ &= P_{31}P_{32}P_{43}P_{14}P_{35} + P_{31}P_{32}P_{43}P_{24}P_{35} \\ &\quad + P_{31}P_{42}P_{43}P_{14}P_{35} + P_{31}P_{42}P_{43}P_{24}P_{35}\end{aligned} \tag{22}$$

and

$$\begin{aligned}\Delta(P) = \{&c_1 = (P_{14})(P_{31}P_{32}P_{35})(P_{43}), \\ &c_2 = (P_{24})(P_{31}P_{32}P_{35})(P_{43}), \\ &c_3 = (P_{14})(P_{31}P_{35})(P_{42}P_{43}), \\ &c_4 = (P_{24})(P_{31}P_{35})(P_{42}P_{43})\}\end{aligned} \tag{23}$$

We have

$$M_{c_1} = (0.3,\ 0,\ 0.7,\ 0.4) \quad M_{c_2} = (0,\ 0.3,\ 0.7,\ 0.4)$$
$$M_{c_3} = (0.3,\ 0,\ 0.7,\ 0.4) \quad M_{c_4} = (0,\ 0.3,\ 0.7,\ 0.4)$$

Thus,

$$\mathrm{Low}(S) = \{(0.3,\ 0,\ 0.7,\ 0.4),\ (0,\ 0.3,\ 0.7,\ 0.4)\} \tag{24}$$

and the whole solution set of (6) is

$$S = (0.3, [0, 0.3], 0.7, 0.4) \cup ([0, 0.3], 0.3, 0.7, 0.4) \tag{25}$$

2.4
NP-hardness of solving fuzzy relation equations

In this section, we will show that the problem to obtain all the minimal solutions is an *NP*-hard problem. The strategy leading to the proof of the *NP*-hardness is to transform polynomially an *NP*-complete problem into the problem of solving fuzzy relation equations.

We choose a well known *NP*-complete problem which is called "The Minimum Cover Problem" [8]:

The minimum cover problem: Given a set S and collection C of its subsets, for an integer K, does C contain a subset C' with $|C'| \leq K$, such that $\cup_{c \in C'} c = S$?

Let us make an algebraic representation of the minimum cover problem. Such representation must be completed in a polynomial time of the size of S. Assume $|S| = m$, $S = \{s_1, \ldots, s_m\}$, and $C = \{c^{(1)}, \ldots, c^{(n)}\}$. A $\{0, 1\}$-vector, with m components, $v = (v_1, \ldots, v_m)$ is defined for each $c \in C$,

$$v_j = \begin{cases} 1 & \text{if } s_j \in c \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

and $B = (1, \ldots, 1)$ to represent the set S, having B n 1's as components.

Consequently, the minimum cover problem has been translated into the following problem: Is there a $\{0, 1\}$-vector A, with n-components, having K or less 1's such that

$$A \cdot (v^{(1)} \ldots v^{(n)})^{\mathrm{T}} = B? \tag{27}$$

If there is a polynomial time algorithm, we can find all the minimal solutions of the Eq. (27), we can count each of them for the 1's in the solution. That is to say, solving Eq. (27) (in terms of its lower bound solutions) is harder than the minimum cover problem. In other words, the minimum cover problem becomes a special case of solving fuzzy relation equations. Thus we have:

Theorem 7 To get all the minimal solutions of (1) is *NP*-hard.

Based on the above discussion, we can introduce the "the fuzzy minimum cover problem" as follows:

Given a fuzzy subset $B = (b_1, \ldots, b_m)$ of a universal set $S = \{1, 2, \ldots, m\}$, where b_j is the membership value of the element $j \in S$ and a collection $C = \{r_1, \ldots, r_n\}$ of its fuzzy subsets, for an integer K, there is a vector $A = (a_1, \ldots, a_n)$ such that

$$A \circ (r_1, \ldots, r_n)^{\mathrm{T}} \geq B \tag{28}$$

and $\sum_{i=1}^{n} a_i \leq K$ (standard summation).

It is not difficult to know the fuzzy minimum cover problem is a *NP*-hard problem. In applications, we can interpret the solution as the percent of resources. The problem has practical applications in intelligent and complex systems and we will give some examples in the next paper.

3
The universal algorithm and its implementations

Although there is no polynomial time algorithm to completely solve the fuzzy relation equations unless $P = NP$, a universal algorithm, nevertheless, is still desirable. On the other hand, this result of *NP*-hardness shows a theoretical time limitation for a large system of fuzzy relation equations. However, the numbers of variables may not be very large in many real world applications. In this situation, a universal algorithm is very useful for analyzing the properties of a system of fuzzy relation equations.

In this section we will give an $O(n!)$ time algorithm to solve general fuzzy relation equations. This algorithm is based on two rules which can apply to the solution-base matrix Q and its logical expression. We have successfully implemented the algorithm in C++ under Windows 95/NT environment. As it turns out, the program gives us the correct answers as it should.

3.1 Simplifying and separating rules

According to Theorems 4, 5, 6 and (14)–(23), two rules can be obtained naturally. In order to program easily, we only deal with solution-base-matrix Q here. However, the validation of the correctness about the two rules is primarily based upon the logical expression of Q. In other words, the mathematical principle is due to the logical expression of Q.

Rule 1: Simplifying rule

For a single column in Q, namely the j-th column, if only one element is not empty, namely q_{ij}, in that column, then there must be an value, $\alpha = \inf(q_{ij})$ appearing in all the minimal solutions. Thus every other column having a non-empty element with the same row index, e.g. $q_{ij'}$, can be deleted (if $q_{ij'}$ contains α).

The correctness of the rule is referred to (16), (19), and (23). This is quite similar to the elimination procedure in Gaussian method of a system of linear equations.

Rule 2: Separating rule

When a column has more than one non-empty elements, e.g. t non-empty elements, then Q can be separated into $Q_1, Q_2, \ldots, Q_t$, each of which has only one non-empty element in that column. Other columns remain intact (or no change). The minimal solutions of Q are the minimal solutions of all $Q_1, Q_2, \ldots, Q_t$.

We can see that if we use the two rules alternatively, we can find all the minimal solutions. However, the Rule 2 may cost exponential time in computation. The correctness of the rule is referred to (14), (15), and (22).

3.2 The solving algorithm

We give the following algorithm:

Algorithm A

```
Step 1: Finding the maximum solution if it exists.
        According to formula (5), we can get the maximum
        vector, and we exam if it is the solution of the Eq.
        (1). If so, these equations have solutions. We then
        will continue the process. Otherwise, the algorithm
        will be halted.
Step 2: Generate the solution-base-matrix. According to
        formulas (7), (8), we can get the solution-base-ma-
        trix $Q$.
Step 3: If the number of rows in $Q$ is 0, halt. If there is a
        column has only one non-empty element $q_{ij}$, get $\alpha$,
        then go to Step 4; otherwise go to Step 5.
Step 4: Delete all columns if they contain a non-empty
        element at the same row ($i$) which contains $\alpha$. Set
        the simplified matrix as the new $Q$, go to Step 3.
Step 5: Separate $Q$ into $t$ $Q$'s, $Q_1, \ldots, Q_t$, each of which has
        a column containing only one non-empty element.
        Go to Step 3.
```

Theorem 8 The time complexity of Algorithm A is $O(n!)$, where n is the number of rows in the matrix R.

Proof. We can ignore Steps 1 and 2 since these are linearly solvable. Of course R has sizes $m \times n$. Steps 3–5 are the iteration. Every iteration reduces size of the matrix Q by 1 in both rows and columns. In addition, if the i-th element in first column is non-empty, after expansion, all non-empty elements at the i-th row will be eliminated. The remaining matrix has sizes $(n-1) \cdot (m-1)$. So the complexity is

$$T(n) = n \cdot T(n-1) + c \cdot (n \cdot m)$$

where c is a constant. So $T(n) = O(n!)$.

4 Tractability: polynomial time problems

Many problems related to fuzzy relation equations are not *NP*-hard in real world applications. In other words, some of real problems based on fuzzy relation equations can be solved in polynomial time. In this section, we will discuss several useful cases.

The polynomial problems of solving fuzzy relation equations is a set containing all equations, each of which can be solved in polynomial time. Such set is denoted by *P*-FRE.

A question we can ask:

"Is there a polynomial time algorithm to decide if a system of fuzzy relation equations in *P*-FRE?"

It is not difficult to show that

Lemma 3 In the solution-base-matrix, if the number of columns (in which the non-zero elements are more than two) is less than $\log(n)$, then the problem is *P*-FRE.

4.1 Polynomial time class and k-diagonal problem

The k-diagonal problem means that only k diagonal lines have non-empty elements in R (or in Q). For example,

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1k} & & & \\ & r_{22} & \cdots & r_{1(k+1)} & & \\ & & \cdots & \cdots & \cdots & \\ & & & r_{n(n-k+1)} & \cdots & r_{nm} \end{pmatrix} \tag{29}$$

We might as well assume $n = m$.

Lemma 4 Let A_1 be the set of all solutions of $XR_1 = B_1$, and let A_2 be the set of all solutions of $XR_2 = B_2$. Then $A = \{(\alpha, \beta)\}$, where $\alpha \in A_1$ and $\beta \in A_2$ is the set of all solutions of $XR = B$, where

$$R = \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix} \quad \text{and} \quad B = (B_1\ B_2) \tag{30}$$

In addition, all the minimal solutions of $XR = B$ are $A^{(\min)} = \{(\alpha^{(\min)}, \beta^{(\min)})\}$, where $\alpha^{(\min)}$ and $\beta^{(\min)}$ are minimal solutions of $XR_1 = B_1$ and $XR_2 = B_2$, respectively.

Example 4 Let

$$R = \begin{pmatrix} 0.1 & 0.4 & 0.5 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0.9 & 0.7 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.8 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.3 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0.5 & 0.7 & 0.9 & 0.8 \\ 0 & 0 & 0 & 0 & 0.2 & 0.4 & 0.3 & 0.6 & 0.5 \\ 0 & 0 & 0 & 0 & 0.7 & 0.4 & 0.2 & 0.1 & 0.6 \\ 0 & 0 & 0 & 0 & 0.8 & 0.9 & 0.7 & 0.2 & 0.4 \end{pmatrix}$$

$$B = (\,0.8 \quad 0.7 \quad 0.5 \quad 0 \quad 0.7 \quad 0.4 \quad 0.4 \quad 0.3 \quad 0.6\,)$$

The maximum solution is,

$$A_{\max} = (\,0.00 \quad 0.80 \quad 0.70 \quad 0.50 \quad 0.30 \quad 0.30 \quad 1.00 \quad 0.40\,)$$

The minimal solutions are:

$$(\,0.00 \quad 0.80 \quad 0.50 \quad 0.00 \quad 0.30 \quad 0.00 \quad 0.70 \quad 0.40\,)$$

$$(\,0.00 \quad 0.80 \quad 0.50 \quad 0.00 \quad 0.00 \quad 0.30 \quad 0.70 \quad 0.40\,)$$

$$(\,0.00 \quad 0.80 \quad 0.00 \quad 0.50 \quad 0.30 \quad 0.00 \quad 0.70 \quad 0.40\,)$$

$$(\,0.00 \quad 0.80 \quad 0.00 \quad 0.50 \quad 0.00 \quad 0.30 \quad 0.70 \quad 0.40\,)$$

Theorem 9 The minimal solutions problem of k-diagonals can be solved in $O(n \cdot n^{k \cdot \log(k)})$ if there are at most $O(n^k)$ minimal solutions.

Proof. According to the above Lemma 4 and the universal algorithm, we know that Q must be a k-diagonal matrix. We want to simplify the k-diagonal matrix Q into the Q_1, Q_2-diagonal forms. First, we separate the matrix R into four parts Q_1, K_1, K_2 and Q_2, where

$$Q = \begin{pmatrix} Q_1 & K_1 & 0 \\ 0 & K_2 & Q_2 \end{pmatrix} . \tag{31}$$

and Q_1, Q_2 have sizes $(n/2 - k/2) \times (n/2 - k/2)$. In fact, K_1 is lower triangular, and K_2 is upper triangular. Expanding $(K_1, K_2)^{\mathrm{T}}$ in Q, merge the solution in Q_1 and Q_2. So the complexity is

$$T(n) = 2(T(n/2 - k/2) \cdot k^k) \tag{32}$$

So $T(n) < O(n \cdot n^{k \cdot \log(k)})$. Since k is a constant, then this is a polynomial time problem.

4.2 Practical problems and their optimal solutions

In most cases, we do not need all lower bound solutions of (1). We usually wish to have only one optimal solution for a real problem. That is to say, it is possible to have the problem solved in polynomial time. The question is what do we mean an optimal solution? The different requirements have different answers. For example, if one just needs maximum solution, then we can provide it in the linear time $n \times m$.

Let J be the set of all row indexes, i.e. $J = \{1, \ldots, n\}$. Let E be a subset of J. If one requires all components corresponding to E to be appeared (not 0) in the solution vector, then to find a solution with a minimal number of non-zero components is such a problem.

The non-zero components indicate the resources for some applications. A complex system can be described as to dispatch the node (e.g. index i) into several tasks, each part should satisfy constraints, e.g., a system of fuzzy relation equations.

We will discuss three interesting/practical problems and sketch their polynomial time algorithms:

(1) Fixed priority problem. Given a permutation of $\{1, \ldots, n\}$, $i_1, \ldots, i_n$, the problem is to get a minimum number of non-zero components in the (minimal) solution vector such that if the i_j-component is not zero, then all $i_1, \ldots, i_{j-1}$ components are not zero.

For this problem, exchange rows of Q to meet the order of $i_1, \ldots, i_n$. This process will only change the component position of the solution vector. This process will only cost $O(n^2)$ time. For all i, from $i = 1$ to n, let S_i be a subset of indexes of columns that contains all rows, each of which has a non-empty element in row $1, \ldots,$ or row i. In this case, we say row $1, \ldots,$ row i cover S_i. To meet the requirement of the problem, we need to test that if $S_i = \{1, \ldots, m\}$. In addition, if $S_i \neq \{1, \ldots, m\}$ and $S_i = S_{i+1}$ then the problem has no solution. Otherwise, we can easily get the solution by obtaining $S_{i+1} - S_i$ for all i.

(2) Minimum resources problem. Find a minimal solution vector such that the arithmetic summation of all components is minimal.

This is a very practical problem. Given n sites, that is to say to find the minimal "resources" solution for all sites in order to get a task done. This problem may not be a polynomial problem, but we can design a fast algorithm to approximate the solution of the problem. The strategy is to find the row with the high contribution and a minimum cost. The high contribution means that it has largest number of non-empty elements in the row. The minimum cost means that it has the smallest value to satisfy these columns. Let $C(i)$ be the number of non-empty elements of the row i in Q, and $v(i)$ is the minimum value of the intersection of these non-empty elements. Choose the i so that $C(i)/v(i)$ is minimum for all i. Repeat the process until all columns are covered. Such a strategy is called a "greedy" method [8]. One can consider the ratio of this method as an optimum solution.

(3) Restricted subset problem. Given a subset of the index set of the components, D , to find a minimal solution in D.

For this problem, find all the corresponding rows. If the rows cover all columns of Q, then there is a minimal solution containing the whole or part of D.

Another problem is to find a minimal solution vector with the smallest difference between its components, that is to balance the resources for each site.

In the future papers, we shall try to give examples of applications of the above problems.

References

1. **Cormen TH, Leiserson CE, Rivest RL** (1993) Introduction to Algorithms, MIT Press, Cambridge, UK
2. **De Baets B** (2000) Analytical solution methods for fuzzy relational equations, in Dubois D, Prade H (eds), Fundamentals of Fuzzy Sets, The Handbook of Fuzzy Sets Series, Kluwer Academic Publishers, Dordrecht, NL
3. **De Beats B, Kerre E** (1994) A primer on solving fuzzy retional equations on the unit interval. International J. Uncertain. Fuzziness Knowledge-Based Systems **2**: 205–225
4. **Imai H, Miyakoshi M, Da-te T** (1997) Some properties of minimal solutions for a fuzzy relation equation. Fuzzy Sets and Systems **90**: 335–340
5. **Di Nola A, Sessa S, Pedrycz W, Sanchez E** (1989) Fuzzy Relation Equations and their Applications to Knowledge Engineering, Kluwer Academic Publishers, Dordrecht, NL
6. **Dubois D, Prade H** (1980) Fuzzy Sets and Systems: Theory and Applications, Academic Press, New York, NY
7. **Dubois D, Prade H** (2000) (eds) The Handbook of Fuzzy Sets Series, Kluwer Academic Publishers, Dordrecht, NL
8. **Gary MR, Johnson DS** (1979) Computers and Intractability, H. Freeman Press
9. **Hopcroft JE, Ullman JD** (1979) Introduction to Automata Theory, Languages, and Computation, Reading, Addison-Wesley, MA
10. **Klir GJ, Yuan B** (1995) Fuzzy Sets and Fuzzy Logic: The Theory and Applications, Prentice Hall, Englewood Cliff, NJ
11. **Pedrycz W, Gomide F** (1998) An Introduction to Fuzzy Sets: Analysis and Design, MIT Press, Cambridge, UK
12. **Sanchez E** (1976) Resolution of composite fuzzy relation equations, Information and Control **30**: 38–48
13. **Togai M, Wang PP** (1983) A study of fuzzy relations and their inverse problem, Proceedings of 13th International Symposium on Multiple Valued Logic, Kyoto, pp 279–285
14. **Tsukamoto Y, Terano T** (1979) Failer diagnosis by using fuzzy logic. Proc. IEEE Conference on Decision Control, Vol **2**, pp 1390–1395
15. **Wagenknecht M, Hartmann K** (1987) On direct and inverse problems for fuzzy equation systems with tolerances. Fuzzy Sets and Systems **24**: 93–102
16. **Wagenknecht M, Hartmann K** (1988) Application of fuzzy sets of type 2 to the solution of fuzzy equation systems. Fuzzy Sets and Systems **25**: 183–190