

Li Chen · Paul P. Wang

Fuzzy relation equations (II): the branch-point-solutions and the categorized minimal solutions

© Springer-Verlag 2006

Abstract This paper presents some novel theoretical results as well as practical algorithms and computational procedures on fuzzy relation equations (FRE). These results refine and improve what has already been reported in a significant manner. In the previous paper, the authors have already proved that the problem of solving the system of fuzzy relation equations is an NP -hard problem. Therefore, it is practically impossible to determine all minimal solutions for a large system if $P \neq NP$. In this paper, an existence theorem is proven: there exists a special branch-point-solution that is greater than all minimal solutions and less than the maximum solution. Such branch-point-solution can be calculated based on the solution-base-matrix. Furthermore, a procedure for determining all branch-point-solutions is designed. We also provide efficient algorithms which is capable of determining as well as searching for certain types of minimal solutions. We have thus obtained: (1) a fast algorithm to determine whether a solution is a minimal solution, (2) the algorithm to search for the minimal solutions that has at least a minimum value at a component in the solution vector, and (3) the procedure of determining if a system of fuzzy relation equations has the unique minimal solution. Other properties are also investigated.

Keywords Fuzzy relation equations · Computational complexity · Minimal solutions · Branch-point-solutions · Algorithms

L. Chen
Department of Electrical Engineering and Computer Science
University of the District of Columbia
Washington, DC 20008, USA

P. P. Wang (✉)
Department of Electrical and Computer Engineering
Duke University,
Durham, NC 27708, USA
E-mail: ppw@ee.duke.edu

1 Introduction

In the previous work [1], we have introduced a solution-base-matrix as well as gave a tractable mathematical logic representation of all minimal solutions. We also designed an universal algorithm for the purpose of obtaining the logical representation of all minimal solutions. In addition, we showed that a polynomial time algorithm to find all minimal solutions may not exist if $P \neq NP$. Thus, the problem of solving fuzzy relation equations is an NP -hard problem so far as the computational complexity is concerned [2] [4].

In this paper, based on the solution-base-matrix, we will prove that a branch-point-solution that is in between all minimal solutions and the maximum solution exists. That is the branch-point-solution will be greater than all minimal solutions and less than the maximum solution. Such branch-point-solution can be calculated mathematically. Furthermore, a procedure for determining all branch-point-solutions is designed. We will also focus on efficient algorithms that determine and search for certain types of minimal solutions. We have obtained (1) an efficient algorithm to determine if a solution is a minimal solution, (2) an algorithm to search for the minimal solutions that has at least a minimal value at a component in the solution vector, and (3) a procedure to determine where a system of the fuzzy relation equations indeed has the unique minimal solution.

2 Background

Let $A = (a_1, \dots, a_n)$ be a fuzzy subset of set X , denoted by $A \in F(X)$; $B = (b_1, \dots, b_m) \in F(Y)$, and $R = [r_{ij}]_{n \times m} \in F(X \times Y)$. The system of the fuzzy relation equations can then be written in the following matrix equation [1] [3] [5].

$$A \circ R = B \quad (1)$$

Solving the system of fuzzy relation equations is to calculate A if given R, B under the assumption, say, we choose operator “ \circ ” to be the standard fuzzy max-min operator (composition). According to Sanchez’s theorem [6], solving

the fuzzy relation equations is equivalent to find all minimal solutions.

Let S be the solution set of (1). The set of the minimal solutions of S is denoted by $Low(S)$. For convenience, $b\epsilon r$ represents the solutions of $xr = b$, and $b\hat{\epsilon}r$ represents the solutions of $xr \leq b$; where xr is the min of x and r . Then we have,

$$b\epsilon r = \begin{cases} b & \text{if } r > b \\ [b,1] & \text{if } r = b \\ \emptyset & \text{if } r < b \end{cases} \quad (2)$$

$$b\hat{\epsilon}r = \begin{cases} [0,b] & \text{if } r > b \\ [0,1] & \text{if } r \leq b \end{cases} \quad (3)$$

Theorem 1 (Sanchez's Theorem) *If the solution set S is non-empty, then the maximum solution of (1), A_{\max} , equals to the maximum solution of $A \circ R \leq B$, \hat{A}_{\max} , i.e.,*

$$A_{\max} = \hat{A}_{\max} \quad (4)$$

where

$$\hat{A}_{\max} = (\sup(\cap_{j=1}^m (b_j \hat{\epsilon} r_{1j})), \dots, \sup(\cap_{j=1}^m (b_j \hat{\epsilon} r_{nj}))).$$

The main result of our previous paper is that we are able to translate the problem of finding all minimal solutions to that of finding the all independent terms in a logical expression. The vehicle for the expression, as it turns out, is the introduction of the solution-base-matrix Q represented as follows:

$$Q = [q_{ij}]_{n \times m}, \quad (5)$$

where

$$q_{ij} = (b_j \epsilon r_{ij}) \cap (\cap_{j=1}^m (b_j \hat{\epsilon} r_{ij})). \quad (6)$$

In [1], we also designated the compatible vector of the solution-base-matrix: Let $t = (t_1, \dots, t_n)$, where t_i is a real number. t is compatible with the solution-base-matrix Q if

$$\forall j \exists i (t_i \neq 0) \wedge (t_i \in q_{ij}), \quad \text{for } j = 1, \dots, m. \quad (7)$$

is true. In addition, let $T = \{t | t \text{ is compatible with } Q\}$. We then obtained the following theorem:

Theorem 2 *Let $t \in T$. Define*

$$t\hat{\cap}\hat{S} = \{x = (x_1, \dots, x_n) | x_i \in (t_i \hat{\cap}\hat{S}_i) \text{ for } i = 1, \dots, n\} \quad (8)$$

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

$$t_i \hat{\cap}\hat{S}_i = \begin{cases} \hat{S}_i & \text{if } t_i = 0, \\ \min\{t_i, \sup(\hat{S}_i)\} & \text{otherwise.} \end{cases} \quad (9)$$

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

In order to represent and design the algorithm for finding all the minimal solutions, we defined 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

$$\Pi = \Pi_{j=1}^m (P_{1j} + P_{2j} + \dots + P_{nj}) \quad (10)$$

We can expand π into a Disjunction Normal Form (DNF) [4]:

$$\pi = \sum_{\xi_i \in \{1,2,\dots,n\}} P_{\xi_1 1} \cdots P_{\xi_m m} \quad (11)$$

Any item of π can be represented by

$$c = (P_{1k_1}, \dots, P_{1k_{t_1}})(P_{2k_{t_1+1}}, \dots, P_{2k_{t_2}}) \cdots (P_{nk_{t_{n-1}+1}}, \dots, P_{nk_{t_n}}) \quad (12)$$

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

Let

$$\Delta = \{c | c \text{ is an item of } \Pi, c \text{ is not false.}\} \quad (13)$$

Theorem 3 *Each element in S can map naturally to an element of Δ .*

3 Modification of the general algorithm

The intension of this section is to modify the simplifying rule of the general algorithm as presented in the first paper [1]. Firstly, we correct two typo errors found in the first paper. At line 7 of page 428, "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 $P = NP$." should read "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 the expectation of $P \neq NP$."

At the last line of page 432, "and the whole solution set of (6) is

$$S = (0.3, [0, 0.3], 0.7, 0.4) \cup ([0, 0.3], 0.3, 0.7, 0.4). \quad (14)$$

should instead read "and the whole solution set of (6) is

$$S = (0.3, [0, 0.3], [0.7, 1], 0.4) \cup ([0, 0.3], 0.3, [0.7, 1], 0.4). \quad (15)$$

The Simplifying rule should be accordingly modified here to meet all situations. (line 24 of page 433 in [1])

"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).

Let's look closer via an example. It shows that if $q_{ij'}$ does not contain α , the element $q_{ij'}$ of the array can not be eliminated.

$$(x_1 \ x_2) \circ \begin{pmatrix} 0.5 & 0.1 \\ 0.5 & 0.3 \end{pmatrix} = (0.5, 0.3)$$

The maximum solution hence is (1.00, 1.00), and the minimal solutions are (0.50, 0.30) and (0.00, 0.50), respectively.

Its solution base matrix Q will be:

$$\begin{pmatrix} [0.50, 1.00] & \emptyset \\ [0.50, 1.00] & [0.30, 1.00] \end{pmatrix}$$

This rule is not entirely correct, because if it is absolutely true, then the minimal solutions of above example will be reduced to only one, i.e. (0.50, 0.30).

This rule should be modified to the following:

Rule 1: Simplifying Rule

For a single column in Q , namely the j -th column, assume this column has only one nonempty element, namely q_{ij} , if there is a non-empty element in the column k and the row i , i.e. $q_{ik} \neq \emptyset$, then Q can be simplified by Q_{-k} or $Q_i + Q_{-k}$ based on the following conditions:

- (i) If $q_{ij} \subset q_{ik}$ or $q_{ij} = q_{ik}$, Q can be reduced to Q_{-k} , where Q_{-k} is obtained by deleting column k from Q .
- (ii) If $q_{ik} \subset q_{ij}$, Q can be represented by $Q_i + Q_{-k}$, where Q_i is obtained by replacing q_{ik} with empty, and Q_{-k} is obtained by deleting column k from Q and replacing q_{ij} with q_{ik} (or $q_{ik} \cap q_{ij}$).

Proof The actual meaning of deleting a column is to reduce the number of variables in a term of the logical expression (10) that corresponding to a minimal expression. $P_{ij} = True$ means that $a_i \in q_{ij}$ refer to Eq. (1). To find a minimal vector $A = (a_1, \dots, a_n)$ also means to find a minimal cover for all m -components in R . Because P_{ij} and q_{ij} has one-to-one correspondence, we might as well to assign q_{ij} has the meaning of the logical variable P_{ij} .

This situation can be represented as follows:

$$\dots(\dots + q_{ik} + \dots)(q_{ij})\dots$$

In order to make both $(\dots + q_{ik} + \dots)$ and (q_{ij}) true, let $P_k = (\dots + q_{ik} + \dots) = (P'_k + q_{ik})$. Thus, $P_k(q_{ij}) = P'_k q_{ij} + q_{ik} q_{ij} q_{ij}$ must be true for all cases. If $q_{ij} \subset q_{ik}$ or $q_{ij} = q_{ik}$, i.e. $\inf(q_{ij}) \geq \inf(q_{ik})$, $q_{ik} \cdot q_{ij} = q_{ij}$, "column k is satisfied when we choose $a_i = \inf(q_{ij})$," so the column k can be deleted. Therefore, we obtained the case (i) of the simplifying rule.

If $q_{ik} \subset q_{ij}$ and $q_{ik} \neq q_{ij}$, i.e. $\inf(q_{ij}) < \inf(q_{ik})$, there is an valuation such that P_k is true with the value of $a_i = \inf(q_{ij})$. However, such a valuation requires P'_k is true because $q_{ik} \cdot q_{ij}$ is false. Or we make that $q_{ik} \cdot q_{ij}$ is true by assigning " $\inf(q_{ik})$," to a_i , i.e. we ignore the value of $P'_k \cdot q_{ij}$ (by deleting the column k). Therefore, we got (ii). This situation is exactly as it is described in above example. So we complete the proof. \square

4 The Branch-point-solution

In this section, we will first prove that there exists a unique solution between any minimal solution and the maximum solution. The new solution is said to be the branch-point-solution because the structure of the solutions begins to have many

branches after this point. This solution can be determined mathematically based on the solution-base-matrix. Second, we will present a procedure for all branch-point-solutions.

4.1 The Branch-point-solution Q_{inf}

A branch-point-solution should be referred to as a minimum ancestor solution for two minimal solutions. In other words, a branch-point-solution is the closest ancestor of two minimal solutions. Let $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ be the two minimal solutions. It should not be difficult to prove that $z = (\max(x_1, y_1), \dots, \max(x_n, y_n))$ is a branch-point-solution.

However, the above strategy will not work if we ask to find a branch point solution that is an ancestor for all minimal solutions due to the established fact that finding all minimal solutions is NP -hard [4]. In other words, there are so many minimal solutions, and they are exponentially increasing along with the size of the solution vector.

Let's revisit the logical expression of the solution-base-matrix, Π . Assuming that we eliminate all $P_{ij} \equiv 0$ variables, so we have:

$$\Pi = \prod_{j=1}^m (\sum_{\xi \in \text{Col}_j} P_{\xi j}) \quad (16)$$

where $\text{Col}_j = \{i \mid Q_{ij} \neq \emptyset\}$.

After we obtain the solution-base-matrix Q , we need to apply the simplifying rule condition (i) to Q . This procedure will simplify Q , and Q only contains the single matrix. The simplified solution-base-matrix should be used as the new solution-base-matrix. Otherwise, the following definition may not always generate a true branch point. Please refer to Example 3 in Sect. 5.

We now define a special vector

$$Q_{\text{inf}} = (\inf(\bigcap_{\xi \in \text{Row}_1} q_{1\xi}), \dots, \inf(\bigcap_{\xi \in \text{Row}_n} q_{n\xi})) \quad (17)$$

where $\text{Row}_i = \{j \mid q_{ij} \neq \emptyset\}$. Q_{inf} may or may not correspond to a term in Π ; however, we will prove that Q_{inf} is a solution of the fuzzy relation equations.

Lemma 1 Q_{inf} is compatible with Q if Eq. (1) has a solution.

Proof If there is a column has all empty element in Q then this FRE has no solution. If not, Q_{inf} satisfies the definition of compatibility of solution-base-matrix.

Theorem 4 Q_{inf} is a solution.

Proof If there is a column has all empty element in Q , then this FRE has no solution.

Let $Q_{\text{inf}} = (a_1, \dots, a_n)$ where $a_i = \inf(\bigcap_{\xi \in \text{Row}_i} Q_{i\xi})$ for all i . We might as well assume that a_i is not 0. Based upon Theorem 2,

$$a_i \hat{\cap} \hat{S}_i = \begin{cases} \hat{S}_i & \text{if } a_i = 0, \\ \min\{a_i, \sup(\hat{S}_i)\} & \text{otherwise.} \end{cases} \quad (18)$$

According to the definition of Q_{inf} ,

$$a_i = \inf(\bigcap_{\xi \in \text{Row}_i} Q_{i\xi}) \leq \sup(\hat{S}_i)$$

so for all i , if $a_i \neq 0$, we have

$$a_i \hat{\wedge} \hat{S}_i = a_i.$$

If $a_i = 0$, since $\hat{S}_i = \cap_{j=1}^m (b_j \hat{\wedge} r_{ij})$ always contains 0, then \hat{S}_i contains a_i . Therefore, $Q_{\text{inf}} \in (Q_{\text{inf}} \hat{\wedge} \hat{S})$ that is a solution of the FRE. \square

Theorem 5 Let A_{min} be a minimal solution of the equation, then $A_{\text{min}} \leq Q_{\text{inf}}$.

Proof Let $A_{\text{min}} = (m_1, \dots, m_n)$ be a minimal solution. If it is not true that $A_{\text{min}} \leq Q_{\text{inf}} = (a_1, \dots, a_n)$, then we have at least an i such that $m_i > a_i$ then let

$$A_Q = (m_1, \dots, m_{i-1}, a_i, m_{i+1}, \dots, m_n),$$

we desire to prove that A_Q is also a solution. This is because $m_i \neq 0$ and A_{min} is a minimal solution then $m_i \in q_{ij}$ for some j based on Theorem 3. So A_{min} is compatible with Q , and $m_i = \inf(m_i \hat{\wedge} \hat{S}_i)$

So $A_Q = (m_Q^{(1)}, \dots, m_Q^{(n)})$ is compatible with Q , and $m_Q^{(i)} = \inf(m_Q^{(i)} \hat{\wedge} \hat{S}_i)$. Therefore, A_Q is a solution. Thus, A_{min} is not a minimal solution. We finished the proof. \square

The following example will explain how do we get Q_{inf} . In next section, we will discuss how do we use Q_{inf} to search for minimal solutions.

Example 1 Let's look Example 1 we presented in [1],

$$\begin{aligned} (x_1 \ x_2 \ x_3 \ x_4) \circ \begin{pmatrix} 0.3 & 0.5 & 0.7 & 0.9 & 0.8 \\ 0.2 & 0.4 & 0.3 & 0.6 & 0.5 \\ 0.7 & 0.4 & 0.2 & 0.1 & 0.6 \\ 0.8 & 0.9 & 0.7 & 0.2 & 0.4 \end{pmatrix} \\ = (0.7 \ 0.4 \ 0.4 \ 0.3 \ 0.6) \end{aligned} \quad (19)$$

we want to obtain $Q_{\text{inf}} = (a_1, \dots, a_n)$ and to verify if it is a solution of the above equation. Based on Example 2 in [1], we have already established

$$Q = \begin{pmatrix} \emptyset & \emptyset & \emptyset & 0.3 & \emptyset \\ \emptyset & \emptyset & \emptyset & 0.3 & \emptyset \\ [0.7, 1] & [0.4, 1] & \emptyset & \emptyset & [0.6, 1] \\ \emptyset & 0.4 & 0.4 & \emptyset & \emptyset \end{pmatrix} \quad (20)$$

According to the definition of Q_{inf} , (17), we have

$$Q_{\text{inf}} = (0.3, 0.3, 0.7, 0.4).$$

It is then straight forward to see that

$$\begin{aligned} (0.3 \ 0.3 \ 0.7 \ 0.4) \circ \begin{pmatrix} 0.3 & 0.5 & 0.7 & 0.9 & 0.8 \\ 0.2 & 0.4 & 0.3 & 0.6 & 0.5 \\ 0.7 & 0.4 & 0.2 & 0.1 & 0.6 \\ 0.8 & 0.9 & 0.7 & 0.2 & 0.4 \end{pmatrix} \\ = (0.7 \ 0.4 \ 0.4 \ 0.3 \ 0.6) \end{aligned} \quad (21)$$

4.2 A procedure for all branch-point-solutions

After we have obtained the first branch-point-solution, in order to obtain all of the other branch-point-solutions, we only need to perform the Separating Rule and the Simplifying Rule alternatively. First, we revisit the Separating Rule as presented in [1]:

“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, \dots, 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, \dots, Q_t .”

Second, we present the procedure to find all branch-point-solutions. According to the logical expression (10), a solution for $Q_i, i = 1, \dots, t$ is also a solution for Q . The procedure is shown below: (1) Use the Simplifying Rule to simplify each Q_i since Q_i has one column with only one non-empty element. 2) Let $Q^{(i)} = Q_i$. Calculate $Q_{\text{inf}}^{(i)}$ based on formula (17). Thus, $Q_{\text{inf}}^{(i)}$ is a solution of Eq. (1) which is smaller than Q_{inf} , i.e. $Q_{\text{inf}} \geq Q_{\text{inf}}^{(i)}$. $Q_{\text{inf}}^{(i)}$ is a new branch-point-solution. 3) Remove all pathological situations and use the Separating Rule on each $Q^{(i)}$. Repeat the process and we will have all branch-point-solutions.

The above procedure (algorithm) will reach all minimal solutions as well as find all branch-point-solutions. However, the Separating Rule may cost exponential time in computation [2] [4].

5 The decision and search algorithms for minimal solutions

Given a vector X , it is quite important to know if the vector is a solution or it may even be a minimal solution. It is easy to test if it is a solution since we only need to calculate the equation $X \circ A = B$. This section will present two algorithms: (i) an $O(n^2m)$ algorithm to decide if X is a minimal solution, and (ii) an $O(n^2m^2)$ algorithm to find a minimal solution.

Furthermore, we shall solve the following problem: Given a X , if X is a solution, we want to find a minimal solution A_{min} such that $A_{\text{min}} < X$. If X is not a solution, it is then desirable to find a nearest minimal solution for X . The solution-base-matrix will be proven to be very helpful in achieving what we desired.

5.1 The decision algorithm

To decide if a vector is a solution or not is straight forward since we only need to spend $O(nm)$ time to multiply this vector with the matrix R in Eq. (1), then compare the result with vector B . An alternative way is to use Theorem 2 when

we need to check for a series of vectors. This method will save time because we can calculate \hat{S} before the testing process. In order to check if a solution A is a minimal solution, we have to test if there is a solution that is smaller than A . This problem differs from the question of finding a minimal solution.

If a solution A is not a minimal solution, there must exist an i , and $a'_i < a_i$ such that $A' = (a_1, \dots, a'_i, \dots, a_n)$ is a solution. Thus, if we scan through and lower down each component of the vector (one at a time), then we will get the answer. However, the problem is how do we choose a smaller a'_i based on a_i . We do not want to test every possible $x < a_i$ since there are infinitive numbers of x .

According to Q , each row provides us the hint to find the next a'_i . It must hold true that

$$\min\{\inf(q_{ij}) | j = 1, \dots, m \text{ and } \inf(q_{ij}) \neq 0\} \leq a'_i$$

or $a'_i = 0$. Note that $\inf(q_{ij}) = 0$ means $q_{ij} = \emptyset$.

We also know that if A is a minimal solution, then $a_i \neq 0$ must be an element of $\{\inf(q_{ij}) | j = 1, \dots, m \text{ and } \inf(q_{ij}) \neq 0\}$. If we sort all these $\inf(q_{ij})$, as denoted by $SQ_{\inf}(i)$, then we could let a'_i be just smaller than a_i in the sorted list. So we can design the following algorithm,

5.1.1 The decision algorithm for minimal solutions

Assume that vector A is already a solution.

Step 1 Sort $\inf(q_{ij})$, $j = 1, \dots, m$ in ascending order for each row i , i.e. calculate $SQ_{\inf}(i)$.

Step 2 From $i = 1$ to n , finding a_i in the sorted list $SQ_{\inf}(i)$. If can not find, A is not a minimal solution.

Step 3 Select left adjacent element in $SQ_{\inf}(i)$ to replace a_i in A . This forms A' . If a_i is already the smallest one in $SQ_{\inf}(i)$, use "0" to replace it.

Step 4 Test if A' is a solution of (1) (or a FRE). If true, A is not a minimal solution.

Step 5 Repeat Step 2 until $i = n$. If every A' is not a solution, then A is a minimal solution.

Theorem 6 *Algorithm A is an $O(n(mn))$ time algorithm to decide if a vector is a minimal solution.*

Proof We could sort all rows first that we need to spend $O(mn(\log m))$ time. In Step 2, find a_i in $SQ_{\inf}(i)$ need $O(\log m)$ if we use binary search. Step 3 uses constant time. Step 4 uses $O(nm)$ time. Step 5 is a repeating process. Therefore, the algorithm needs $O(n((mn) + \log m)) + O(mn(\log m))$ time. That is $O(n((mn) + \log m))$. \square

Example 2 We still use the example shown in Example 1. Let's decide if $A = Q_{\inf} = (0.3, 0.3, 0.7, 0.4)$ is a minimal solution? The row 3 in Q is $([0.7, 1], [0.4, 1], \emptyset, \emptyset, [0.6, 1])$, so $SQ_{\inf}(3) = \{0.4, 0.6, 0.7\}$. When we get the Step 2, $i=1$, we found that 0.3 is in $SQ_{\inf}(1) = \{0.3\}$. In Step 3, we use 0

to replace 0.3, so $A' = (0, 0.3, 0.7, 0.4)$. In Step 4, we know A' is a solution of (17). So Q_{\inf} is not a minimal solution. If we test for $A = (0, 0.3, 0.7, 0.4)$, we can see none of $A' = (0, 0, 0.7, 0.4)$, $(0, 0.3, 0.6, 0.4)$ or $(0, 0.3, 0.7, 0)$ is a solution of (17), so $A = (0, 0.3, 0.7, 0.4)$ is a minimal solution. This result matches the finding of Example 3 in [1].

5.2 The minimal solution search algorithms

We will show that there is an $O(n^2m^2)$ algorithm to find a minimal solution. This algorithm is called the vertical-first push algorithm (The name is analogous to the depth-first-search [2]).

The idea of vertical-first push algorithm is to reduce (pushing toward to zero) the value at a component of vector Q_{\inf} (or A_{\max}) until that the vector is no longer the solution or it equals a "0." Assume that this vector is A . We keep the A a solution of the FRE all the time. This algorithm will switch to "push" next component until all components of A are pushed. We can prove that the remaining A is a minimal solution.

An alternative pushing method is called the horizontal-first-push algorithm (borrowed from the breadth-first-search [2]). It pushes one level down at a component of A at a time, then it go to next component. The algorithm will go back to push the first component, repeat this process until none of the component can be pushed while A is maintained as a solution.

5.2.1 Minimal solution algorithm: vertical-first-push

Step 1 Sort $\inf(q_{ij})$ in ascending order for each row, $SQ_{\inf}(i)$; let $A = Q_{\inf}$.

Step 2 Set $i = 1$, finding a_i in $SQ_{\inf}(i)$.

Step 3 Select left adjacent element in $SQ_{\inf}(i)$ to form A' . If a_i is already the smallest one use "0" to replace it.

Step 4 Test if A' is a solution of the FRE. If true, $A = A'$, go to Step 3; otherwise set $i = i + 1$, go to Step 2.

Step 5 If $i > n$, Stop. A is a minimal solution.

The proof of the correctness of Algorithm B is shown below. If there is a solution $M < A$, there must be a first i from index 1 such that $m_i < a_i$. In other words, $m_1 = a_1, \dots, m_{i-1} = a_{i-1}$ but $m_i < a_i$. According to the property: every B such that $M \leq B \leq Q_{\inf}$ is a solution, we have

$$((m_1 = a_1), \dots, (m_{i-1} = a_{i-1}), a_i,$$

$$Q_{\inf}(i+1), \dots, Q_{\inf}(n))$$

and

$$((m_1 = a_1), \dots, (m_{i-1} = a_{i-1}), m_i,$$

$$Q_{\inf}(i+1), \dots, Q_{\inf}(n))$$

are solutions. The algorithm must push a_i until it reaches m_i . So, such a M does not exist. There are only m possible

different values in $SQ_{\text{inf}}(i)$, $i = 1, \dots, n$. Therefore, the time complexity of the algorithm is $O(n^2m^2)$. Thus, we have proved

Theorem 7 *Vertical-first-push algorithm can find a minimal solution in $O(n^2m^2)$ time.*

Similarly, we can design the horizontal-first-push algorithm.

5.2.2 Minimal solution algorithm: horizontal-first-push

Step 1 Sort $\text{inf}(q_{ij})$ in ascending order for each row, $SQ_{\text{inf}}(i)$; let $A = Q_{\text{inf}}$. Set $i = 1$.

Step 2 Finding a_i in $SQ_{\text{inf}}(i)$. (Record the index and next time will not do the search again.)

Step 3 Select left adjacent element in $SQ_{\text{inf}}(i)$ to form A' . If a_i is already the smallest one use "0" to replace it.

Step 4 Test if A' is a solution of the FRE. If true, $A = A'$. Set $i = i + 1 \bmod(n)$. If false, mark "Done" on this component. Set $i = i + 1 \bmod(n)$. Go to Step 2.

Step 5 If all components in A are marked "Done," we have the minimal solution.

Example 3 This example uses the fuzzy relation equations of Klir and Yuan [6, p 159-160].

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

$$B = (0.8 \ 0.7 \ 0.5 \ 0)$$

The algorithm in [6] yields the maximum solution:

$$(0.00 \ 0.80 \ 0.70 \ 0.50)$$

The minimal solutions calculated in [6] are,

$$A_1 = (0.00 \ 0.80 \ 0.50 \ 0.00)$$

$$A_2 = (0.00 \ 0.80 \ 0.00 \ 0.50)$$

Using our method, the solution-base matrix Q is,

$$Q = \begin{pmatrix} \emptyset & \emptyset & \emptyset & [0.0, 0.0] \\ [0.8, 0.8] & [0.7, 0.8] & \emptyset & [0.0, 0.8] \\ \emptyset & [0.7, 0.7] & [0.5, 0.7] & [0.0, 0.7] \\ \emptyset & \emptyset & [0.5, 0.5] & [0.0, 0.5] \end{pmatrix}$$

If the above Q is used to be the solution-base matrix, then $Q_{\text{inf}} = (0, 0.8, 0.7, 0.5)$. This equals to the maximum solution, and it is not strictly a brach-point-solution.

Since the first column of Q has only one nonempty member $q_{21} = [0.8, 0.8]$ and it is the subset of q_{22} and q_{24} , hence, the columns 2 and 4 can be deleted. So the original Q can

be simplified to be a new solution-base-matrix based on the simplifying rule condition (i),

$$Q = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset \\ [0.8, 0.8] & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & [0.5, 0.7] & \emptyset \\ \emptyset & \emptyset & [0.5, 0.5] & \emptyset \end{pmatrix}$$

i.e.,

$$Q = \begin{pmatrix} \emptyset & \emptyset \\ [0.8, 0.8] & \emptyset \\ \emptyset & [0.5, 0.7] \\ \emptyset & [0.5, 0.5] \end{pmatrix}$$

So, Q_{inf} should be $(0, 0.8, 0.5, 0.5)$. This is the new solution vector and a branch-point-solution. We will use vertical-first-push and horizontal-first-push algorithms next to get minimal solutions.

(a) The use of vertical-first-push algorithm (VFP):

Because vertical-first-push algorithm or horizontal-first-push can start at any solution, let $A = A_{\text{max}} = (0, 0.8, 0.7, 0.5)$. We will get a minimal solution from A . Using vertical-first-push algorithm, set $i = 2$, we first get $A' = (0, 0, 0.7, 0.5)$ that is not a solution. A is un-changed. Set $i = 3$, get $A' = (0, 0.8, 0.5, 0.5)$ that is a solution. Set $A = (0, 0.8, 0.5, 0.5)$, get $A' = (0, 0.8, 0, 0.5)$ that is a solution. Set $A = (0, 0.8, 0, 0.5)$, and set $i = 4$. Get $A' = (0, 0.8, 0, 0)$ that is not a solution. Thus, $A = (0, 0.8, 0, 0.5)$ is a minimal solution.

If we let $A = Q_{\text{inf}} = (0, 0.8, 0.5, 0.5)$, the above process will be faster.

(b) The use of horizontal-first-push algorithm (HFP):

Let $A = A_{\text{max}} = (0, 0.8, 0.7, 0.5)$. Let's use HFP algorithm. Mark "Done" on Component 1 and set $i = 2$, we first get $A' = (0, 0, 0.7, 0.5)$ that is not a solution. Mark "Done" on Component 2. A is un-changed. Set $i = 3$, get $A' = (0, 0.8, 0.5, 0.5)$ that is a solution. Set $A = (0, 0.8, 0.5, 0.5)$. Set $i = 4$, and get $A' = (0, 0.8, 0.5, 0)$ that is a solution. Mark "Done" on Component 4. Set $A = (0, 0.8, 0.5, 0)$, and set $i = 1$. Get $A' = (0, 0.8, 0, 0)$ that is not a solution. Mark "Done" on Component 3. Every component is marked. Thus, $A = (0, 0.8, 0.5, 0)$ is a minimal solution. We can see that VFP algorithm and HFP algorithm may generate different results.

Either vertical-first-push or horizontal-first-push algorithm guarantees us to get a minimal solution. To search for all solutions from $A = Q_{\text{inf}}$ will be studied in the following research paper.

5.3 Q_{inf} and $Low(S)$

Let $Low(S)$ be the set of whole minimal solutions of a FRE. It also can be proved that

$$Q_{\text{inf}} \geq \cup\{s | s \in Low(S)\}$$

\cup means to get a maximum number in each corresponding component for all elements in S . We know $\cup\{s | s \in Low(S)\}$ is a solution and that is greater than all minimal solutions.

According to the construction method of $s \in \text{Low}(S)$, each component of s , say s_i is not greater than $Q_{\text{inf}}(i)$, so we have proved that $\cup\{s|s \in \text{Low}(S)\} \leq Q_{\text{inf}}$.

Based on the discussion of Example 3, using the simplified Q , Q_{inf} will be a branch point if Q_{inf} is not a unique minimal solution. If every column of Q contains two or more non-empty elements, there must be more than one minimal solution. Otherwise, (there is a single non-empty element column), according to the proof for the simplifying rule in Sect. 3, after the simplification process, we can say that there is only one minimal solution if each column in Q has only one non-empty element (i.g. condition (1) applied to the rest of columns), or there must be more minimal solutions in either cases of the solution-base matrix has split into two matrices, i.e. condition (2), or there is one column that contains two non-empty elements. As long as there is two non-empty elements in the simplified Q , each time select one of these two element will generate two minimal solutions.

Proposition 1 *Using simplified Q , Q_{inf} will be a branch point if Q_{inf} is not a unique minimal solution.*

However, it may not guarantee $Q_{\text{inf}} = \cup\{s|s \in \text{Low}(S)\}$. To expand Q using Separating Rule may cost exponential time growing in computations. More investigation on this issue is desirable.

6 Uniqueness

We will provide here an algorithm to decide whether the fuzzy relation Eq. (1) has only one minimal solution. It is possible to obtain this solution, if it does exist.

According to Theorem 3, the fuzzy relation equations (FRE) has a unique minimal solution if and only if the logical expression can be reduced to be one simplified item. Therefore, we could design an algorithm that will determine if there are at least two items that are not reducible as well as not identical.

A natural way is to expand the logical expression (8) which is in conjunction normal form introduced in [1].

$$\Pi = \dots (P_{1j} + P_{2j} + \dots) \dots \quad (22)$$

We start at the first pair of parenthesis (first column) which contains more than one none-zero variable. We might as well assume that P_{1j} , P_{2j} come from first column, i.e. $j = 1$, so we at least have two items, such as:

$$(P_{11} + P_{21})P = P_{11}P + P_{21}P \quad (23)$$

where $P' = P_{\xi_2} \dots P_{\xi_m}$, formed by an item in each factor in Π .

In many cases, $P_{11}P$ and $P_{21}P$ are independent. However, in the following three cases, these two items are actually related:

(1) P actually has P_{1s} and P_{2t} , $s, t \geq 2$. They will refer to the first or second component in the solution vector.

(2) P contains p_{1s} but p_{2t} with $\inf(q_{11}) \leq \inf(q_{1s})$ In such a case, P is true then $(p_{11} + p_{21})P$ is true. This condition is the same as the simplification rule.

(3) P contains p_{2t} but p_{1s} , and $\inf(q_{21}) \leq \inf(q_{2t})$ In such a case, P is true then $(p_{11} + p_{21})P$ is also true. This condition is also the same as the simplification rule.

The procedure to test the above cases can find if there are more than two (including two) minimal solutions. However, it is not a complete algorithm for uniqueness test. It could be used for hands-on exercises.

Now, we will provide a simple algorithm for testing the uniqueness. A result to be shown as follows is not only straight forward but it is also very useful.

Proposition 2 *Let $A_{\text{min}} = (m_1, \dots, m_n)$ be a minimal solution. m_i is not zero for an i , $i = 1, \dots, n$. If $X = (x_1, \dots, x_n)$ is a solution with $x_i = 0$ then the FRE has at least two minimal solutions. Moreover, if $x_i < m_i$ then X the FRE has at least two minimal solutions.*

Theorem 8 *There is an $O(n^2m^2)$ time algorithm to solve the problem of uniqueness.*

Proof The idea of this algorithm is to prove there is an minimal solution other than $A_{\text{min}} = (m_1, \dots, m_n)$. Then, start at the smallest index of components to the largest one, (or let i just from 1 to n), make a vector

$$A(i) = (Q_{\text{inf}}^1, \dots, Q_{\text{inf}}^{i-1}, a_i = 0, Q_{\text{inf}}^{i+1}, \dots, Q_{\text{inf}}^n). \quad (24)$$

Pull the value of a_i up until the above is a solution, and denote this solution as $A(i)$ too. So we have n solutions namely $A(1), A(2), \dots, A(n)$. If for all i , $a_i \geq m_i$ then The FRE has the unique minimal solution, otherwise, it has at least two minimal solutions based on Proposition 2.

Now we want to prove that if there is another minimal solution, above algorithm will find an a_i that is smaller than m_i . Assuming that B is another minimal solution, there exists an i such that $x_i < m_i$. So

$$\begin{aligned} B \leq B(x_i) &= (Q_{\text{inf}}^1, \dots, Q_{\text{inf}}^{i-1}, x_i, Q_{\text{inf}}^{i+1}, \dots, Q_{\text{inf}}^n) \\ &\leq Q_{\text{inf}}. \end{aligned} \quad (25)$$

Due to $x_i > 0$. In other words, we can always find $B(x_i)$. Therefore, this algorithm is an $O(n^2m^2)$ time algorithm. The proof is completed. \square

If we use binary search algorithm in the sorted lists for the possible selection of each a_i , then we will have the following significant results.

Theorem 9 *(i) There is an $O(n^2m \log(m))$ algorithm to find a minimal solution. (ii) There is an $O(n^2m \log(m))$ algorithm to decide if a FRE has a unique solution.*

There is an interesting open question based on the following observation: Vertical-first-push and horizontal-first-push search the minimal solution from different ‘‘angles,’’ and they yield the different minimal solutions in Example 3. Does this imply the following statement?

Open Question: Perform the vertical-first-push and horizontal-first-push algorithms on Eq. (1). If they generate the same minimal solution, then the FRE only has one minimal solution.

7 Approximations

Finding all minimal solutions in polynomial time is impossible if $P \neq NP$. For any given X , the question to be addressed then is how to find a solution or even a minimal solution which is near X .

We have described the vertical-first-push and horizontal-first-push algorithms in the previous section. We know that the result may be different if we start at a different component (whom we push first) of vector Q_{inf} .

To find a solution near X , we should have started at X by “push” a component at a time in both up or down direction. If we start at i -component, we will reach the minimal value on this component. We assume that after the first push, we will push $(i + 1), \dots, n, 0, \dots, i - 1$ components in order to find a minimal solution $V(i)$. We also can randomly select a component to “push.” We got V_{rand} . The solution-base-matrix Q provides us with a basis of that push algorithm which will eventually guarantee that we do not have to reduce the value continuously but just use certain numbers, say, $\inf(q_{ij})$.

On the other hand, the Horizontal-First-Push Algorithm will start at i -component, but only reduce one level according to the sorted $\inf(q_{ij})$'s. Then move to $(i + 1) \bmod(n)$, and so on, until we can no longer reduce any component value to maintain a solution. So we have $H(i)$ that indicates

the starting pushing component. We also can use the random selection method to decide next push component. That we can get $H(\text{random})$.

Therefore, we at least have obtained the minimal solution set that contains $V(i)$ and $H(i)$, $i = 1, \dots, n$.

8 Conclusions

New theorems have been presented in this paper for the sake of completeness concerning FRE solutions. Three algorithms as well as some pragmatic issues such as approximations also add another dimensions toward the realization of much needed details in effective applications. More serious research work should be done in the future to build a sound and realistic theory for the fuzzy relation equations.

References

1. Chen L, Wang PP (2002) Fuzzy relation equations (I): the general and specialized solving algorithms. *Soft Computing* 6:428–435
2. Cormen TH, Leiserson CE, Rivest RL (1993) *Introduction to Algorithms*. The MIT Press, Cambridge
3. Di Nola A, Sessa S, Pedrycz W, Sanchez E (1989) *Fuzzy relation equations and their applications to knowledge engineering*. Kluwer, Dordrecht
4. Gary MR, Johnson DS (1979) *Computers and Intractability*. H. Freeman Press, San Francisco
5. Klir GJ, Yuan B (1995) *Fuzzy sets and fuzzy logic : the theory and applications*. Prentice-Hall, Englewoodcliffs
6. Sanchez E (1976) Resolution of composite fuzzy relation equations. *Information and Control* 30(1):38–48