Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 315290, 10 pages
http://dx.doi.org/10.1155/2014/315290

Research Article

A Divide-and-Conquer Approach for Solving Fuzzy Max-Archimedean t-Norm Relational Equations

Jun-Lin Lin, Hung-Chjh Chuan, and Laksamee Khomnotai

Department of Information Management and Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taoyuan 32003, Taiwan

Correspondence should be addressed to Jun-Lin Lin; jun@saturn.yzu.edu.tw

Received 2 January 2014; Accepted 22 April 2014; Published 11 May 2014

Academic Editor: Juan Carlos Cortés

A system of fuzzy relational equations with the max-Archimedean t-norm composition was considered. The relevant literature indicated that this problem can be reduced to the problem of finding all the irredundant coverings of a binary matrix. A divide-and-conquer approach is proposed to solve this problem and, subsequently, to solve the original problem. This approach was used to analyze the binary matrix and then decompose the matrix into several submatrices such that the irredundant coverings of the original matrix could be constructed using the irredundant coverings of each of these submatrices. This step was performed recursively for each of these submatrices to obtain the irredundant coverings. Finally, once all the irredundant coverings of the original matrix were found, they were easily converted into the minimal solutions of the fuzzy relational equations. Experiments on binary matrices, with the number of irredundant coverings ranging from 24 to 9680, were also performed. The results indicated that, for test matrices that could initially be partitioned into more than one submatrix, this approach reduced the execution time by more than three orders of magnitude. For the other test matrices, this approach was still useful because certain submatrices could be partitioned into more than one submatrix.

1. Introduction

Solving a system of fuzzy relational equations is a subject of great scientific interest [1, 2]. This work considers a system of fuzzy relational equations of the form

$$\begin{aligned} &\max\{t(x_1, a_{11}), t(x_2, a_{21}), \ldots, t(x_m, a_{m1})\} = b_1 \\ &\max\{t(x_1, a_{12}), t(x_2, a_{22}), \ldots, t(x_m, a_{m2})\} = b_2 \\ &\qquad \ldots, \\ &\max\{t(x_1, a_{1n}), t(x_2, a_{2n}), \ldots, t(x_m, a_{mn})\} = b_n, \end{aligned} \tag{1}$$

where $a_{ij}, b_j, x_i \in [0, 1]$ for each i, $1 \leqslant i \leqslant m$ and for each j, $1 \leqslant j \leqslant n$ and t represents a continuous Archimedean t-norm function. System (1) can be succinctly written in the following equivalent matrix form:

$$X \circ A = B, \tag{2}$$

where $X = (x_i)_{1\times m}$ is the matrix of unknowns, $A = (a_{ij})_{m\times n}$ is the matrix of coefficients, $B = (b_j)_{1\times n}$ is the right-hand side of the system, and the symb "$\circ$" represents a max-Archimedean t-norm composition.

Di Nola et al. [3] indicated that, given a continuous t-norm for t in system (1) and assuming the existence of solutions, the solution set of system (1) can be fully determined by the greatest solution and a finite number of minimal solutions. It is well-known that the greatest solution can be easily computed, but finding all minimal solutions is difficult. Li and Fang [4] demonstrated that the systems of max-t-norm equations can be divided into two categories, depending on the function t in the system. When t is continuous and Archimedean, the minimal solutions correspond one-to-one to the irredundant coverings of a set covering problem. When t is continuous and non-Archimedean, the minimal solutions correspond to a subset of constrained irredundant coverings of a set covering problem. Li and Fang [5] discussed the necessary and sufficient conditions for solving max-t-norm

equations. A survey of similar and other related works is described in [5].

This work focuses on system (1), with t representing a continuous Archimedean t-norm function. Although solving such a system is equivalent to solving a set covering problem, set covering problems are classified as NP-hard problems [5, 6]. Therefore, solving a system of max-Archimedean t-norm equations is NP-hard. Wu and Guu [7] demonstrated that the number of minimal solutions of such a system can grow exponentially as the numbers of variables and equations (i.e., m and n in system (1)) increase. Therefore, solving system (1) that has hundreds or thousands of minimal solutions is not uncommon and can be a challenge.

The concept of partitioning involves grouping related variables and equations and separating unrelated variables and equations. A variable x_i and the jth equation (i.e., $\max\{t(x_1, a_{1j}), \ldots, t(x_m, a_{mj}) = b_j\}$) in system (1) are *related* if the value of x_i can affect whether the jth equation holds. Furthermore, two variables (or two equations or one variable and one equation) in system (1) are related if they are related to a common variable or equation. In system (1) with numerous variables and equations (i.e., m and n are high), it is likely that not all variables and equations are related to one another. Consequently, system (1) may be partitioned into several subsystems, each containing only the related variables and equations. Thus, the original problem is decomposed into several subproblems. Because solving system (1) is an NP-hard problem [5, 6], solving several smaller subproblems is considerably faster than solving the original problem directly. Therefore, partitioning can expedite the process of solving system (1). Notably, even if all variables and equations of system (1) are related, partitioning can still be applied to the subsets of all the variables and equations. For example, many approaches involve reducing system (1) by fixing the value of a certain variable x_i (Rule 5 in Section 5 provides an example). Subsequently, the remaining variables and equations can be partitioned into more than one group such that each group contains only the related variables and equations. The concept of partitioning is discussed further in Section 4.

Based on the concept of partitioning, the first objective of this study was to develop a divide-and-conquer approach for finding all of the minimal solutions of system (1). In this approach, system (1) is first transformed into a binary binding matrix. We propose an algorithm, called PA, in which the concept of partitioning is applied to decompose the binary binding matrix into several submatrices, the irredundant coverings of each submatrix are constructed recursively, and, finally, they are used to form the irredundant coverings of the binary binding matrix. Once all of the irredundant coverings of the binary binding matrix are found, they can be easily converted into the minimal solutions of system (1).

Numerous studies on solving system (1) have been conducted [7–9], but few have provided a performance study of the methods used for solving system (1) in which hundreds or thousands of minimal solutions are involved. Wu and Guu [7] used their method to solve test problems for which the number of minimal solutions ranged from 6 to 100, and the results indicated (Figure 1) that all test problems can be solved in less than 300 ms by using an ordinary PC. However, test problems solved by using such a scale do not fully reflect the difficulty of solving system (1). One problem that hinders the process of performing tests on a large scale is that generating test cases in which system (1) has a high number of minimal solutions is complex.

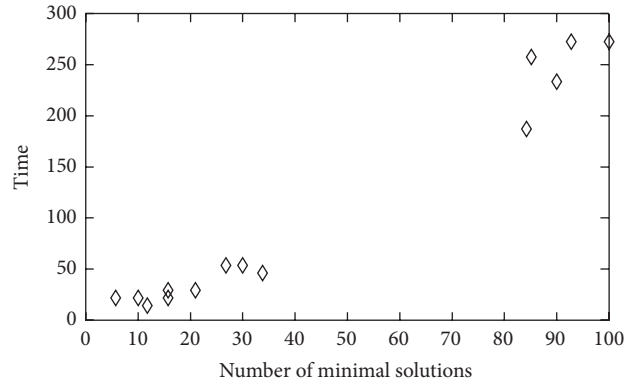

Figure 1: Number of minimal solutions versus execution time (in ms) reported in [7].

To bypass the need to generate complex system (1) test cases, large binary matrices can be generated because any system (1) case can be reduced to a binary matrix, called a *binary binding matrix,* and the problem of solving system (1) can be reduced to the problem of finding all of the irredundant coverings of the binary binding matrix. According to Lin [8], both the reduction process and the conversion of an irredundant covering to a minimal solution can be conducted in polynomial time and, therefore, finding all irredundant coverings is the core process executed in solving system (1). In other words, the time required to find all of the irredundant coverings of the binary binding matrix accounts for most of the time required to solve system (1). This is especially true for complex system (1) cases because the time used for both the reduction of system (1) and the conversion of irredundant coverings to minimal solutions is insubstantial compared with the time used for finding all irredundant coverings. Therefore, the time required to find all irredundant coverings when adopting the proposed approach can be used to represent the performance of this approach. The second objective of this study was to develop an algorithm for generating binary matrices with various characteristics on a considerably large scale to enable the evaluation of the performance of various approaches in finding all irredundant coverings on the test matrices that are difficult to solve.

To evaluate the impact of partitioning on the execution time, the proposed algorithm (PA) was compared with another approach (denoted as the non-PA) proposed by Markovskii [6] for finding all irredundant coverings. The only difference between the PA and non-PA is that the PA contains an additional step for incorporating the concept of partitioning. Several test matrices were generated for this performance study. For test matrices that can be initially partitioned into more than one submatrix, the PA reduces the execution time required by the non-PA by more than three orders of magnitude. Even for test matrices that cannot be initially partitioned into more than one submatrix,

the PA still offers more favorable performance than does the non-PA. This is attributed to the partitioning of the submatrices of the binary binding matrix.

The rest of this paper is organized as follows. In Section 2, the preliminaries of Archimedean t-norm are given. In Section 3, the procedure for constructing the binary binding matrix and minimal solutions is presented. In Section 4, the concept of partitioning is discussed and an algorithm for finding all irredundant coverings is proposed in Section 5. In Section 6, the procedure for generating test matrices is described and the performance results are presented. Finally, conclusion is given in Section 7.

2. Preliminaries

This section describes the basic concepts of t-norm, Archimedean t-norm, and the greatest solution and minimal solutions of fuzzy relational equations. Please refer also to [3–5, 9, 10].

A triangular norm (t-norm for short) is a binary function mapping from $[0,1]^2$ to $[0,1]$ that satisfies the following conditions:

(1) $t(a,b) = t(b,a)$ (commutativity),

(2) $t(a,t(b,c)) = t(t(a,b),c)$ (associativity),

(3) $t(a,b) \leqslant t(a,c)$, if $b \leqslant c$ (monotonicity),

(4) $t(a,0) = 0$ and $t(a,1) = a$, for any $a = [0,1]$ (boundary condition).

A well-known fact is that $t(a,b) \leqslant \min\{a,b\}$ for any t-norm. The commonly seen "min" and "product" are both a t-norm function.

Let $X(A,B) = \{X \in [0,1]^m \mid X \circ A = B\}$ denote the set of all solution vectors of (2), $\mathcal{I} = \{1,2,\ldots,m\}$, let $\mathcal{J} = \{1,2,\ldots,n\}$ be two index sets, and let $X^1 = (x_i^1)_{1\times m}$ and $X^2 = (x_i^2)_{1\times m}$, for all $i \in \mathcal{I}$, be two vectors. For any $X^1, X^2 \in X(A,B)$, the relation $X^1 \leqslant X^2$ holds if and only if $x_i^1 \leqslant x_i^2$ for all $i \in \mathcal{I}$. A solution $\overline{X} \in X(A,B)$ is called the greatest solution if $X \leqslant \overline{X}$ for all $X \in X(A,B)$. Conversely, a solution $\underline{X} \in X(A,B)$ is a minimal solution if $\forall X \in X(A,B)$, where $X \leqslant \underline{X}$ implies that $X = \underline{X}$. As described in Section 1, with a continuous t-norm for t in system (1), the solution set $X(A,B)$ can be completely determined by the greatest solution and a finite number of minimal solutions. This study assumes that t-norms are continuous.

The greatest solution of system (1) with a t-norm for t can be computed explicitly using the "$\rightarrow_t$" operator defined as

$$a \longrightarrow_t b = \sup\{x \in [0,1] \mid t(x,a) \leqslant b\}, \tag{3}$$

where $t(x,a)$ is a t-norm and $b \in [0,1]$. If the solution set of system (1) is not empty, then the greatest solution $\overline{X} = (\overline{x}_i)_{i\in\mathcal{I}}$ can be calculated as follows:

$$\overline{x}_i = \min_{j\in\mathcal{J}} \left(a_{ij} \longrightarrow_t b_j\right) \quad \text{for each } i \in \mathcal{I}. \tag{4}$$

Mostert and Shields [11] subdivided continuous t-norms into three categories, namely, the "min" operation, Archimedean t-norms, and ordinal sums of a family of properly defined Archimedean t-norms. The Archimedean t-norm t is a t-norm with $t(a,a) < a$ for all $0 < a < 1$ [3]. Notably, the well-known "min" operation is not an Archimedean t-norm. Wu and Guu [7] collected six Archimedean t-norm functions as follows:

(1) Algebraic product: $t(a,b) = ab$;

(2) Łukasiewicz t-norm: $t(a,b) = \max\{0, a+b-1\}$;

(3) Einstein product: $t(a,b) = ab/(1+(1-a)(1-b))$;

(4) Hamacher product: $t(a,b) = ab/(\lambda+(1-\lambda)(a+b-ab))$, where $0 \leqslant \lambda \leqslant \infty$;

(5) Yu operation: $t(a,b) = \max\{0, (1+\lambda)(a+b-1)-\lambda ab\}$, where $-1 \leqslant \lambda \leqslant \infty$;

(6) Weber operation: $t(a,b) = \max\{0, (a+b+\lambda ab-1)/(1+\lambda)\}$, where $-1 \leqslant \lambda \leqslant \infty$.

Simple formula for calculating the greatest solution of system (1) that uses any of these six Archimedean t-norm functions for t is also available in Wu and Guu [7].

3. Reduction of Fuzzy Relational Equations to Covering Problem

This section describes the procedure to reduce the problem of finding all minimal solutions of system (1) to the problem of finding all irredundant coverings of the binary binding matrix of system (1). The description follows the results of Lin [8]. Subsequently, in Section 4, we apply the concept of partitioning to the binary binding matrix to expedite finding all irredundant coverings.

Let $M = (m_{ij})_{|I|\times|J|}$ denote a matrix with $i \in I$, $j \in J$ and $m_{ij} \in \{0,1,2\}$, where I and J denote the index sets for the rows and the columns of M, respectively. Notably, both I and J are a set of positive integers. Let $I_j(M) = \{i \in I \mid m_{ij} \neq 0\}$ denote an index set for each $j \in J$ and let $J_i(M) = \{j \in J \mid m_{ij} \neq 0\}$ denote an index set for each $i \in I$. A set $C \subseteq I$ is a *covering* of M if $\bigcup_{i\in C} J_i(M) = J$. A covering C is *irredundant* if each proper subset of C is not a covering of M. The term $\Phi(M)$ denotes the set of all irredundant coverings of M.

Example 1. Consider the matrix M below, where the index sets I and J are indicated on the left and on the top of the matrix, respectively. The set of all irredundant coverings of M is $\Phi(M) = \{\{1,2\},\{1,3\},\{2,3\}\}$:

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix} \end{array}. \tag{5}$$

The *binding matrix* of system (1) is denoted by $\mathcal{M} = (m_{ij})_{m\times n}$ and is given by

$$m_{ij} = \begin{cases} 1 & \text{if } t\left(\overline{x}_i, a_{ij}\right) = b_j \neq 0, \\ 2 & \text{if } b_j = 0, \\ 0 & \text{if } t\left(\overline{x}_i, a_{ij}\right) \neq b_j, \end{cases} \tag{6}$$

for $i \in \mathcal{I}$ and $j \in \mathcal{J}$.

According to Expression (6), if $b_j = 0$, then $m_{ij} = 2$ for each $i \in \mathcal{I}$; that is, all elements in the jth column of $\mathcal{M}$ are 2. Such columns are referred to as *all-2-columns* in a binding matrix. Let $\mathcal{M}^*$ denote a binding matrix $\mathcal{M}$ with all of its all-2-columns removed. It is obvious that if B is not a zero vector, then $\mathcal{M}^*$ is a binary matrix containing one or more columns and the set of irredundant coverings of $\mathcal{M}$ equals that of $\mathcal{M}^*$. The matrix $\mathcal{M}^*$ is referred to as the *binary binding matrix* of system (1).

Example 2. Consider the following fuzzy relational equations with max-Łukasiewicz t-norm composition:

$$X \circ A = B, \quad \text{where } A = \begin{pmatrix} 0.8 & 0.9 & 0.2 & 0.3 \\ 0.1 & 0.7 & 0.8 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0 \\ 0.9 & 0.8 & 0.7 & 0.9 \end{pmatrix}, \quad B = (0.5, 0.6, 0.7, 0). \tag{7}$$

The binding matrix $\mathcal{M}$ is the same as the matrix M in Example 1, while the binary binding matrix $\mathcal{M}^*$ is formed by the first three columns of $\mathcal{M}$.

The *mapping vector* of an irredundant covering $C \in \Phi(\mathcal{M}^*)$ is denoted by $X^C = (x_i^C)_{i \in \mathcal{I}}$ and is given by

$$x_i^C = \begin{cases} 0 & \text{if } i \notin C, \\ \overline{x}_i & \text{otherwise.} \end{cases} \tag{8}$$

Let $\underline{X}(A, B)$ denote the set of all minimal solutions of system (1). Lin [8] proved that if B is not a zero vector, then $\underline{X}(A, B)$ equals the set of mapping vectors of all irredundant coverings of $\mathcal{M}^*$; that is, $\underline{X}(A, B) = \{X^C \mid C \in \Phi(\mathcal{M}^*)\}$. If B is a zero vector, namely, $b_j = 0$ for every $j \in \mathcal{J}$, then it is obvious that $\underline{X}(A, B) = \{\vec{0}\}$. Therefore, $\underline{X}(A, B)$ can be determined with the following procedure.

(1) If B is a zero vector, then $\underline{X}(A, B) = \{\vec{0}\}$ and stop.

(2) Calculate the greatest solution $\overline{X}$ using Expression (4).

(3) If $\overline{X} \notin X(A, B)$, then $\underline{X}(A, B) = \emptyset$ and stop.

(4) Calculate the binding matrix $\mathcal{M}$ using Expression (6).

(5) Let $\mathcal{M}^*$ be obtained from $\mathcal{M}$ with all of its all-2-columns removed.

(6) Construct $\Phi(\mathcal{M}^*)$ by searching all irredundant coverings of $\mathcal{M}^*$.

(7) $\underline{X}(A, B) = \{X^C \mid C \in \Phi(\mathcal{M}^*)\}$, where each X^C is calculated using Expression (8).

Example 3. Consider the fuzzy relational equations in Example 2. Since B is not a zero vector, Step (1) is skipped. Step (2) obtains the greatest solution $\overline{X} = (0.7, 0.9, 0.8, 0.1)$, and Step (3) then finds that $\overline{X} \circ A = B$ holds (i.e., $\overline{X} \in X(A, B)$). Step (4) obtains the binding matrix $\mathcal{M}$, which is the same as the matrix M in Example 1. Step (5) obtains the binary binding matrix $\mathcal{M}^*$, which is the same as $\mathcal{M}$ but without its fourth column. Step (6) yields $\Phi(\mathcal{M}^*) = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. Finally, Step (7) yields $\underline{X}(A, B) = \{(0.7, 0.9, 0, 0), (0.7, 0, 0.8, 0), (0, 0.9, 0.8, 0)\}$.

It is obvious that Steps (1)–(5) can be done in $O(mn)$ time. Li and Fang [4] and Lin [8] proved the bijective (i.e., both one-to-one and onto) mapping between a minimal solution and an irredundant covering, and thus Step (7) can be done in $O(m|\underline{X}(A, B)|)$ time, where $|\underline{X}(A, B)|$ is the number of minimal solutions of (2). Step (6), finding all irredundant coverings, is the most time-consuming step. Therefore, the task of finding all minimal solutions of (2) is reduced to the task of finding all irredundant coverings of a binary matrix, which is the focus of the next two sections.

4. Partitioning

This section describes the concept of partitioning a binary matrix into one or more submatrices such that the irredundant coverings of the binary matrix can be derived from the irredundant coverings of these submatrices. First, the notation used is defined as follows to facilitate the discussion.

Let $M = (m_{ij})_{|I| \times |J|}$ denote a binary matrix with $i \in I$ and $j \in J$, where I and J denote the index sets for the rows and the columns of M, respectively. Here, both I and J are a set of integers (not necessarily contiguous or starting from 1). Let $M[I^k; J^k] = (m_{ij})_{|I^k| \times |J^k|}$ denote a submatrix of M, where $i \in I^k \subseteq I$ and $j \in J^k \subseteq J$; that is, I^k and J^k are the index sets for the rows and the columns of $M[I^k; J^k]$, respectively.

Example 4. Consider the matrices M, M^1, and M^2 below, where the index sets for the rows and the columns are, respectively, indicated on the left and on the top of each matrix. The index sets for the rows of M, M^1, and M^2 are $I = \{1, 2, 3, 4\}$, $I^1 = \{2, 3\}$, and $I^2 = \{2, 3\}$, respectively. The index sets for the columns of M, M^1, and M^2 are $J = \{1, 2, 3\}$, $J^1 = \{3\}$, and $J^2 = \{1, 2, 3\}$, respectively. Here, $M^1 = M[I^1; J^1]$ is a submatrix of M formed by rows 2 and 3 and column 3. Similarly, $M^2 = M[I^2; J^2]$ is a submatrix of M formed by rows 2 and 3 and columns 1, 2, and 3. Notably, M^1 also equals $M^2[I^1; J^1]$ and thus is a submatrix of M^2 :

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \qquad M^1 = \begin{array}{c} \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{c} 3 \end{array} \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{array}$$

$$M^2 = \begin{array}{c} \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \end{array}. \tag{9}$$

Definition 5. Let M^1 and M^2 be two binary matrices. The $\bowtie$ operator is defined by $\Phi(M^1) \bowtie \Phi(M^2) = \{C^1 \bigcup C^2 \mid C^1 \in \Phi(M^1) \wedge C^2 \in \Phi(M^2)\}$.

Given a binary matrix M, then both $\{\emptyset\} \bowtie \Phi(M) = \Phi(M)$ and $\emptyset \bowtie \Phi(M) = \emptyset$ hold.

In Section 1, we described the concept of partitioning, which involves grouping the related variables and equations

of system (1) to reduce the problem size. Because the variables and equations of system (1), respectively, correspond to the rows and columns of its binding matrix, we applied the same concept to the binary binding matrix by grouping the rows and columns that are related. Given a binary matrix $M = (m_{ij})$, row i and column j are *related* if $m_{ij} \neq 0$. Furthermore, two rows (or two columns or one row and one column) are related if they are related to a common row or column.

The matrix M can be considered a bipartite graph $G = (I, J, M)$, where the index sets I and J represent the two sets of vertices of G and the matrix M represents the edges that connect the vertices in I and J. If $m_{ij} \neq 0$, then an edge connects the vertex $i \in I$ and the vertex $j \in J$. Notably, no edge connects two vertices both in I or both in J. Therefore, if two rows (or two columns or one row and one column) are related, then a path connects the corresponding vertices in G. Because a graph is *connected* if every pair of vertices is connected by a path, a connected subgraph of G represents a set of related rows and columns in M. Partitioning is used to find all *components* (i.e., the *maximal connected subgraph*) of G. A connected subgraph of G is maximal if vertices and edges that could be added to the subgraph and still leave it connected do not exist in G. Finding all of the components in a bipartite graph is faster than finding all of the components in a general graph because the former can stop as soon as one of the two sets of vertices is fully explored. A formal definition of partitioning is given as follows.

Definition 6. Given a binary matrix $M = (m_{ij})_{|I|\times|J|}$, the *partitioning* of M, denoted as $\Psi(M)$, is formed by a set of submatrices $M^k = M[I^k, J^k]$ of M, where $k \in P = \{1, 2, \ldots, |\Psi(M)|\}$ and $|\Psi(M)|$ is the number of matrices in $\Psi(M)$, such that the following conditions are satisfied:

(1) $\bigcup_{k\in P} I^k = I$ and $\bigcup_{k\in P} J^k = J$,

(2) $I^{k'} \cap I^{k^*} = J^{k'} \cap J^{k^*} = \emptyset$ for any $k', k^* \in P$ but $k' \neq k^*$,

(3) $m_{ij} = 0$ for any $i \in I^{k'}$ and $j \in J^{k^*}$, where $k', k^* \in P$ but $k' \neq k^*$.

Example 7. Consider the binary matrix M below. For ease of exposition, the index sets for the rows and the columns of M are, respectively, indicated on the left and on the top of the matrix:

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{10}$$

By Definition 6, we have $\Psi(M) = \{M^1, M^2\}$, where $M^1 = M[\{4, 6, 8\}, \{5, 8, 9\}]$ and $M^2 = M[\{1, 2, 3, 5, 7, 9\}, \{1, 2, 3, 4, 6, 7, 10\}]$ are shown as

$$M^1 = \begin{array}{c} \\ 4 \\ 6 \\ 8 \end{array} \begin{array}{c} \begin{array}{ccc} 5 & 8 & 9 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}, \tag{11}$$

$$M^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 7 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 6 & 7 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}. \tag{12}$$

Theorem 8 states that if a binary matrix can be partitioned into several submatrices, as described in Definition 6, then the set of irredundant coverings of the binary matrix can be constructed by performing the $\bowtie$ operation on the sets of irredundant coverings of these submatrices.

Theorem 8. *Given a binary matrix M, if $\Psi(M) = \{M^1, M^2, \ldots, M^p\}$, then $\Phi(M) = \Phi(M^1) \bowtie \Phi(M^2) \bowtie \cdots \bowtie \Phi(M^p)$.*

Proof. Let $P = \{1, 2, \ldots, p\}$ and $M^k = [I^k; J^k]$ for each $k \in P$. By Definition 6, $\Phi(M^1) \bowtie \Phi(M^2) \bowtie \cdots \bowtie \Phi(M^p) \subseteqq \Phi(M)$ holds. Assume to the contrary that there exists $C \in \Phi(M)$ but $C \notin \Phi(M^1) \bowtie \Phi(M^2) \bowtie \cdots \bowtie \Phi(M^p)$. The set C can be decomposed into p disjoint subsets of C such that $C = \bigcup_{k\in P} C^k$, where each $C^k \subseteqq I^k$. Since $C \in \Phi(M)$, by Condition (3) of Definition 6, each C^k must be a covering of M^k. If $C^{k'} \notin \Phi(M^{k'})$ for some $k' \in P$, then there exists a covering $\underline{C}^{k'} \subset C^{k'}$ of $M^{k'}$. Then, $(C \setminus C^{k'} \cup \underline{C}^{k'}) \subset C$ is a covering of M, and consequently $C \notin \Phi(M)$, which contradicts $C \in \Phi(M)$. Therefore, $\Phi(M) \subseteqq \Phi(M^1) \bowtie \Phi(M^2) \bowtie \cdots \bowtie \Phi(M^p)$. □

The advantage of using partitioning is twofold. First, partitioning involves decomposing a matrix into several submatrices with a reduced number of rows and columns, the irredundant coverings of which can be found more efficiently than those of the original matrix. Second, if the covering of submatrices is not minimal, then it is immediately discarded and is not combined with the coverings of other submatrices to form new coverings. This substantially reduces the number of redundant coverings generated. Furthermore, once the set of irredundant coverings of each submatrix is found, Theorem 8 can be applied to obtain the set of irredundant coverings of the original matrix without generating any redundant coverings of the original matrix. Because an irredundant covering of the binary binding matrix of system (1) corresponds to a minimal solution, only minimal solutions are generated. The disadvantage of using partitioning is that if the partitioning of the binary binding matrix contains only one component (i.e., the matrix), then benefits cannot be obtained from partitioning, and the time used to perform

Algorithm PA(M)
Input: a binary matrix $M = \left(m_{ij}\right)_{|I|\times|J|}$ with $i \in I$ and $j \in J$.
Output: $\Phi(M)$
(1) **If** J is a singleton $\{j\}$ **then**
(2) **return** $\left\{\{i\}\, i \in I_j(M)\right\}$;
(3) **If** $J = \emptyset$ **then**
(4) **return** $\{\emptyset\}$;
(5) **If** $\exists j \in J$ such that $I_j(M) = \emptyset$ **then**
(6) **return** $\emptyset$;
(7) **If** there exists a singleton $I_j(M) = \{i\}$ for some $j \in J$ **then**
(8) **let** $M' := M[I \setminus \{i\}; J \setminus J_i(M)]$
(9) **return** $\{\{i\}\} \bowtie$ **PA**(M'); // singleton removal
(10) **If** $\Psi(M) = \{M^1, M^2, \ldots, M^p\}$ and $p > 1$ **then**
(11) **return PA**$(M^1) \bowtie$ **PA**$(M^2) \bowtie \cdots \bowtie$ **PA**(M^p); // partitioning
(12) **Let** row i be the row with the most 1s in M;
(13) **Let** $M^{\times} := M[I \setminus \{i\}; J]$ and $M^{-} := M[I \setminus \{i\}; J \setminus J_i(M)]$;
(14) **Let** $\Phi(M^{-}) :=$ **PA**(M^{-}) and $\Phi(M^{\times}) :=$ **PA**$(M^{\times})$;
(15) **Let** $M^{+} := M[I \setminus \{i\}; J_i(M)]$;
(16) **Let** $\Delta := \{C \in \Phi(M^{-}) C \text{is not a covering of} M^{+}\}$;
(17) **Return** $(\{\{i\}\} \bowtie \Delta) \bigcup \Phi(M^{\times})$; // forced binding

Algorithm 1: Algorithm for finding all irredundant coverings.

partitioning is wasted. The impact of partitioning on performance is discussed further in Section 6.

5. The Divide-and-Conquer Algorithm

We propose a divide-and-conquer algorithm for constructing the set of irredundant coverings of a binary matrix M. The algorithm follows a set of rules specified by Lin [8] either to obtain the irredundant coverings directly or to decompose a binary matrix into one or more submatrices. Rules 1–3 consider some trivial cases of M whose irredundant coverings can be directly derived and are adopted as the termination condition in the proposed algorithm.

Rule 1. If J is a singleton $\{j\}$, then $\Phi(M) = \{\{i\} \mid i \in I_j(M)\}$.

Rule 2. If $J = \emptyset$, then $\Phi(M) = \{\emptyset\}$.

Rule 3. If there exists a column $j \in J$ such that $I_j(M) = \emptyset$, then $\Phi(M) = \emptyset$.

Rule 4 reduces a binary matrix by identifying the indexes of the rows that are required in all irredundant coverings of the matrix.

Rule 4 (singleton removal). If there exists a singleton $I_j(M) = \{i\}$ for some $j \in J$, then $\Phi(M) = \{\{i\}\} \bowtie \Phi(M[I \setminus \{i\}; J \setminus J_i(M)])$.

If none of the above rules are applicable, then the algorithm calculates the partitioning $\Psi(M)$. Subsequently, if $|\Psi(M)| > 1$, then the irredundant coverings of M can be derived from the irredundant coverings of those submatrices in $\Psi(M)$, according to Theorem 8. However, if $|\Psi(M)| = 1$, then Rule 5 [6] is applied to decompose M into three submatrices.

Rule 5 (forced binding). Let $M^{-} = M[I \setminus \{i\}; J \setminus J_i(M)]$, $M^{+} = M[I \setminus \{i\}; J_i(M)]$ and $M^{\times} = M[I \setminus \{i\}; J]$ for some $i \in I$. Then, $\Phi(M) = (\{\{i\}\} \bowtie (\Phi(M^{-}) \setminus \Phi(M^{+}))) \bigcup \Phi(M^{\times})$.

Algorithm 1 shows the proposed algorithm (denoted as PA). Notably, this algorithm does not find any covering that is redundant. If a matrix can be partitioned into more than one submatrix, then lines 10 and 11 of the algorithm are applied to expedite the process of executing the algorithm by using the concept of partitioning. When lines 10 and 11 are excluded, the resulting algorithm, denoted as non-PA, is identical to the algorithm proposed by Markovskii [6] or Lin [8]. The PA is demonstrated in the following examples.

Example 9. Consider the matrix M in Example 4. Select row 1 to apply forced binding (lines 12–17 of PA) and obtain three matrixes M^{-}, M^{+}, and $M^{\times}$, as shown in

$$M^{-} = \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} 3 \\ \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \end{array} \qquad M^{+} = \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} 1\ 2 \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$
$$M^{\times} = \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} 1\ 2\ 3 \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{13}$$

Lines 1-2 of PA yield $\Phi(M^{-}) = \{\{2\}, \{3\}\}$, and subsequently lines 15-16 yield $\Delta = \Phi(M^{-})$. By lines 7–9 and then lines 1-2 of PA, we have $\Phi(M^{\times}) = \{\{2, 3\}\}$. Finally, $\Phi(\mathscr{M}) = (\{\{1\}\} \bowtie \Delta) \cup \Phi(M^{\times}) = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$.

Example 10. Consider the matrix M in Example 7. Since $\Psi(M) = \{M^1, M^2\}$, by Theorem 8, $\Phi(M) = \Phi(M^1) \bowtie \Phi(M^2)$.

To compute $\Phi(M^1)$, we select row 4 of M^1 to apply forced binding (lines 12–17 of PA) and obtain three matrixes M^{1-}, M^{1+}, and $M^{1\times}$ (shown below) such that $\Phi(M^1) = (\{\{4\}\} \bowtie (\Phi(M^{1-}) \setminus \Phi(M^{1+}))) \bigcup \Phi(M^{1\times})$:

$$M^{1-} = \begin{array}{c} \\ 6 \\ 8 \end{array} \begin{array}{c} 9 \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{array} \qquad M^{1+} = \begin{array}{c} \\ 6 \\ 8 \end{array} \begin{array}{c} 5\ 8 \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$
$$M^{1\times} = \begin{array}{c} \\ 6 \\ 8 \end{array} \begin{array}{c} 5\ 8\ 9 \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}. \tag{14}$$

By lines 1-2 of PA, $\Phi(M^{1-}) = \{\{6\}, \{8\}\}$. By lines 7–9 and then lines 1-2 of PA, $\Phi(M^{1\times}) = \{\{6, 8\}\}$. Consequently, $\Phi(M^1) = \{\{4, 6\}, \{4, 8\}, \{6, 8\}\}$.

To compute $\Phi(M^2)$, we select row 1 of M^2 to apply forced binding and obtain three matrixes M^{2-}, M^{2+}, and $M^{2\times}$ (shown below) such that $\Phi(M^2) = (\{\{1\}\} \bowtie (\Phi(M^{2-}) \setminus \Phi(M^{2+}))) \bigcup \Phi(M^{2\times})$. Thus,

$$M^{2-} = \begin{array}{c} \\ 2 \\ 3 \\ 5 \\ 7 \\ 9 \end{array} \begin{array}{c} 2\ 4\ 6\ 7 \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array},$$
$$M^{2+} = \begin{array}{c} \\ 2 \\ 3 \\ 5 \\ 7 \\ 9 \end{array} \begin{array}{c} 1\ 3\ 10 \\ \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \tag{15}$$
$$M^{2\times} = \begin{array}{c} \\ 2 \\ 3 \\ 5 \\ 7 \\ 9 \end{array} \begin{array}{c} 1\ 2\ 3\ 4\ 6\ 7\ 10 \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

Then, M^{2-} can be further partitioned into M^3 and M^4 (shown below), and consequently $\Phi(M^{2-}) = \Phi(M^3) \bowtie \Phi(M^4)$:

$$M^3 = \begin{array}{c} \\ 2 \\ 7 \end{array} \begin{array}{c} 2 \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{array} \qquad M^4 = \begin{array}{c} \\ 3 \\ 5 \\ 9 \end{array} \begin{array}{c} 4\ 6\ 7 \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}. \tag{16}$$

By lines 1-2 of PA, $\Phi(M^3) = \{\{2\}, \{7\}\}$. Calculating $\Phi(M^4)$ is similar to calculating $\Phi(M^1)$, and thus we have $\Phi(M^4) = \{\{3, 5\}, \{3, 9\}, \{5, 9\}\}$. Consequently, $\Phi(M^{2-}) = \{\{2\}, \{7\}\} \bowtie \{\{3, 5\}, \{3, 9\}, \{5, 9\}\} = \{\{2, 3, 5\}, \{2, 3, 9\}, \{2, 5, 9\}, \{3, 5, 7\}, \{3, 7, 9\}, \{5, 7, 9\}\}$.

$M^{2\times}$ can be further partitioned into M^5 and M^6 (shown below), and consequently $\Phi(M^{2\times}) = \Phi(M^5) \bowtie \Phi(M^6)$. By lines 7–9 and then lines 1-2 of PA, $\Phi(M^5) = \{\{2, 7\}\}$. Similar to $\Phi(M^1)$, $\Phi(M^6) = \{\{3, 5\}, \{3, 9\}, \{5, 9\}\}$. Thus, $\Phi(M^{2\times}) = \{\{2, 3, 5, 7\}, \{2, 3, 7, 9\}, \{2, 5, 7, 9\}\}$:

$$M^5 = \begin{array}{c} \\ 2 \\ 7 \end{array} \begin{array}{c} 1\ 2\ 3\ 10 \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \qquad M^6 = \begin{array}{c} \\ 3 \\ 5 \\ 9 \end{array} \begin{array}{c} 4\ 6\ 7 \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}. \tag{17}$$

Therefore, $\Phi(M^2) = \{\{1, 2, 3, 5\}, \{1, 2, 3, 9\}, \{1, 2, 5, 9\}, \{1, 3, 5, 7\}, \{1, 3, 7, 9\}, \{1, 5, 7, 9\}, \{2, 3, 5, 7\}, \{2, 3, 7, 9\}, \{2, 5, 7, 9\}\}$. Since both $\Phi(M^1)$ and $\Phi(M^2)$ are found, $\Phi(M)$ can be derived from $\Phi(M^1) \bowtie \Phi(M^2)$.

6. Performance Study

This performance study focused on evaluating the impact of applying the concept of partitioning. This was achieved by comparing the performance of the PA with that of the non-PA, because the algorithms differed only in whether the concept of partitioning was used. As discussed in Section 3, finding all of the irredundant coverings of the binary binding matrix $\mathscr{M}^*$ is the most time-consuming step in solving system (1). Therefore, we measured only the time required to derive $\Phi(\mathscr{M}^*)$ from $\mathscr{M}^*$ to evaluate the speed at which system (1) can be solved using a given approach.

6.1. Test Matrices. We used binary binding matrices rather than system (1) for this performance study. This offered three advantages. First, a binary binding matrix is a binary matrix only and is independent of any specific Archimedean t-norm function. Second, using binary binding matrices prevents the imprecision caused by floating point truncation from occurring when solving system (1). Third, generating a large binary matrix is easier than generating system (1) with a high number of equations and variables.

Algorithm 2 shows the procedure for generating the test matrices. This procedure involves four parameters, m, n, d, and p, and generates an $m \times n$ binary matrix M with density d such that $\Psi(M)$ contains p submatrices of M with approximately the same number of rows and columns. The *density* of M is defined as the number of nonzero elements in M divided by $m \times n$. In this procedure, an irredundant covering is first injected into the matrix to avoid $\Phi(M) = \emptyset$ (lines 5–9). Then, the elements in the regions of M that correspond to these p submatrices are repeatedly and randomly chosen to assume the value of 1 until the number of elements with a value of 1 reaches $\lceil m * n * d \rceil$ (lines 11–15).

In this study, test matrices were generated with $m = n = 24$, p ranging from 1 to 8 and d ranging from $2/m$ to $1/p$ with an increment of $1/m$. Notably, if $d = 1/m$, then the matrix had at most one irredundant covering. Therefore, only test matrices with a density of no less than $2/m$ were generated in this study. In addition, when the partitioning of a matrix contains p submatrices of equal size, the density of the matrix could not be greater than $1/p$ and, thus, the greatest density was set to $1/p$.

Because this procedure is random, it does not always generate a binary matrix with numerous irredundant coverings. This is especially true when m and n are low or d is near zero

Input: m, n, d and p. Note: $1/m \leqslant d \leqslant 1/p$ and $p \leqslant n$ are required.
Output: $M = (m_{ij})_{|I|\times|J|}$ where $i \in I = \{1, \ldots, m\}$ and $j \in J = \{1, \ldots, n\}$

(1) **Let** $P := \{1, 2, \ldots, p\}$;
(2) Randomly divide I into p disjoint subsets $I^1, I^2, \ldots, I^p$ such that $\bigcup_{k\in P} I^k = I$, and $\left|\left|I^{\hat{k}}\right| - \left|I^{\check{k}}\right|\right| \leqslant 1$ for any $\hat{k}, \check{k} \in P$;
(3) Randomly divide J into p disjoint subsets $J^1, J^2, \ldots, J^p$ such that $\bigcup_{k\in P} J^k = J$, and $\left|\left|J^{\hat{k}}\right| - \left|J^{\check{k}}\right|\right| \leqslant 1$ for any $\hat{k}, \check{k} \in P$;
(4) Initialize m_{ij} to 0 for all $i \in I$ and $j \in J$;
(5) **For** $k = 1$ to p **do** // inject an irredundant covering to avoid $\Phi(M) = \emptyset$.
(6) **For each** $j \in J^k$ **do**
(7) Randomly select i from I^k, and set $m_{ij} := 1$;
(8) **End of for each**
(9) **End of for**
(10) **Let** $l := \lceil m * n * d \rceil - n$;
(11) **While** $l > 0$ **do**
(12) Randomly select i and j from I and J, respectively;
(13) **If** $i \in I^k$, $j \in J^k$ for some $k \in P$, and $m_{ij} = 0$ **then**
(14) **let** $m_{ij} := 1$ and $l := l - 1$;
(15) **End of while**
(16) **Return** M;

Algorithm 2: Procedure for generating test matrices.

or one. Therefore, for each setting of d and p, this procedure was repeated five times to generate five test matrices. For example, when $p = 1$, we varied d from 2/24 to 1/1 with an increment of 1/24 and consequently generated 115 ($= 23 \times 5$) test matrices. In general, given a fixed p value, $(24/p - 1) \times 5$ test matrices were generated in this study. That is, 55, 35, 25, 15, and 10 test matrices were generated for $p = 2$, 3, 4, 6, and 8, respectively. For clarity, Figure 2 shows only the number of irredundant coverings of the test matrices with the lowest or the highest number of irredundant coverings among the respective five test matrices with the same p and d values. Among all the test matrices generated, the matrix with the highest number of irredundant coverings ($|\Phi(M)| = 9680$) was generated using $d = 6/24$ and $p = 3$.

To understand how d affects the number of irredundant coverings, we performed a preliminary check on the number of irredundant coverings for the test matrices with $p = 1$. Specifically, we first grouped the generated matrices with $p = 1$ by their densities. Then, for every two groups with their difference in density being 1/24, a Mann-Whitney U test was performed to compare the difference of the number of irredundant coverings between both groups. The results show that when both groups' densities are less than or equal to 5/24, the number of irredundant coverings is smaller in the group with smaller density; when both groups' densities are greater than or equal to 8/24, the number of irredundant coverings is larger in the group with smaller density; when both groups' densities are between 5/24 and 8/24, there is no significant difference between the two groups in the number of irredundant coverings. Thus, although we did not include test matrices with density less than 2/24 in this study, the generated test matrices have covered a wide range of density to provide meaningful analysis.

6.2. Performance Results. The performance study was conducted on a desktop PC with Pentium D (3.0 GHz) processor and 1 gigabyte main memory, running Windows XP. To evaluate the impact of using partitioning, each test matrix was subjected to two tests, a test in which the PA was used and a test in which the non-PA was used. The results are shown in Figures 3–5.

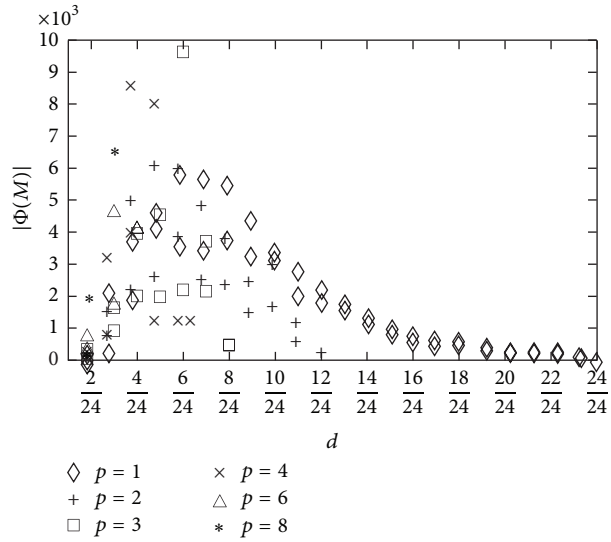

Figure 2: Density d versus the number of irredundant coverings $|\Phi(M)|$.

First, consider the group of test matrices with $p = 1$. Because $p = 1$, the 115 test matrices in this group were generated without intentionally making them capable of being partitioned into more than one submatrix. Prior to comparing the execution times of the PA and non-PA on this group of test matrices, we used the Kolmogorov-Smirnov test to check the normality of the execution time, and the results show that the assumption of normality failed for both the PA and non-PA. Because the assumption of normality

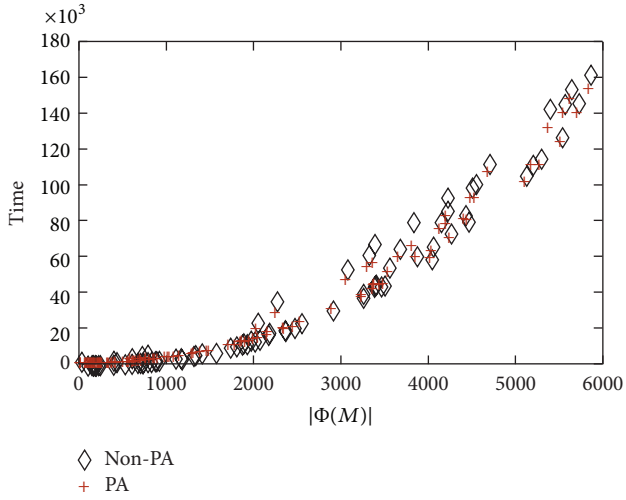

Figure 3: $|\Phi(M)|$ versus execution time (in ms) when $p = 1$.

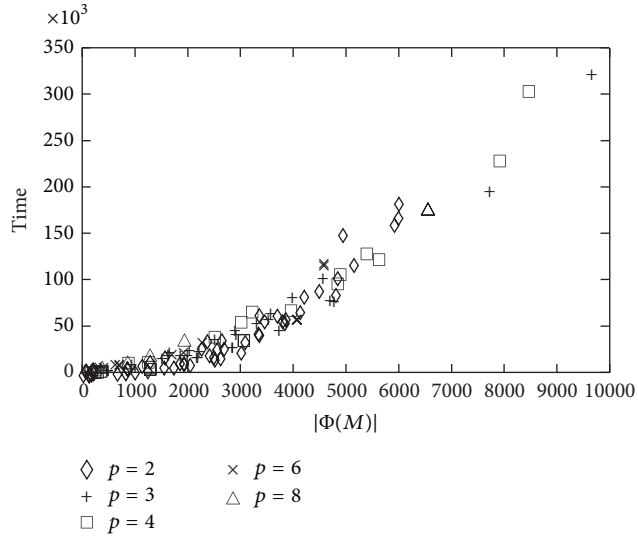

Figure 4: $|\Phi(M)|$ versus execution time (in ms) of non-PA when $p > 1$.

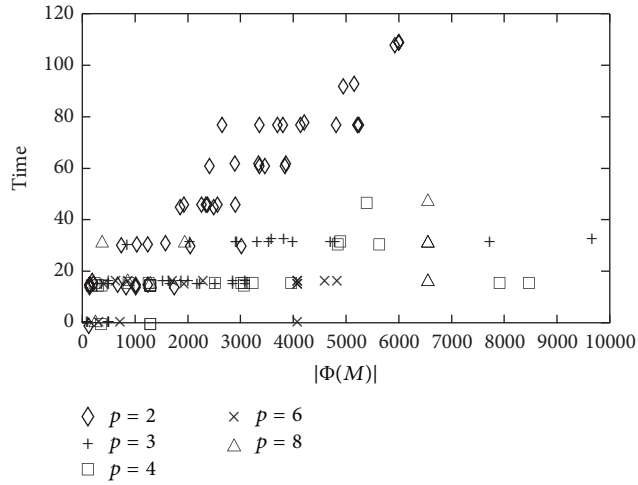

Figure 5: $|\Phi(M)|$ versus execution time (in ms) of PA when $p > 1$.

of distribution was questionable, the Wilcoxon signed-rank test was used as a substitute for a paired t-test to compare the difference between the execution time of the PA and non-PA. The results were in the expected direction and were significant ($z = -7.741$ and $p < 0.05$). Thus, it is statistically significant to say that the execution time of the PA is smaller than that of the non-PA with this group of test matrices. This may be due to the fact that, although the test matrices in this group could not be partitioned into more than one submatrix (i.e., $|\Psi(M)| = 1$), several of the submatrices could and, therefore, the concept of partitioning was still helpful. Figure 3 shows that the execution times of the PA and non-PA exhibited a similar pattern: both were linearly proportional to the square of the number of irredundant coverings of the test matrix. The correlation coefficient between the execution time and the square of the number of irredundant coverings was 0.99502 for the PA and 0.990939 for the non-PA.

For test matrices in which $p > 1$, Figures 4 and 5 reveal that the PA outperformed the non-PA by more than three orders of magnitude. Figure 4 shows that the execution time of the non-PA was still linearly proportional to the square of the number of irredundant coverings and was not affected by the value of p. Figure 5 shows that, for any test matrix in which $p > 1$, the PA required less than 120 ms to determine $\Phi(M)$. If a test matrix can be partitioned into more than one submatrix (i.e., $|\Psi(M)| > 1$), then the PA first identifies the irredundant coverings of these submatrices. Because the number of irredundant coverings of the test matrix is the product of the number of irredundant coverings of these submatrices, finding the irredundant coverings of these submatrices is much faster than finding those of the test matrix. Consequently, the time required by the PA to find all the irredundant coverings can be reduced substantially. We also used the Wilcoxon signed-rank test to compare the execution times of the PA and non-PA on each group of test matrices with the same number of partitions. The results were all in the expected direction and were significant ($z = -6.452$, -5.16, -4.372, -3.408, and -2.803 for the groups of test matrices with 2, 3, 4, 6, and 8 partitions, respectively, and all $p < 0.05$).

7. Conclusion

In a system of fuzzy relational equations with numerous variables and equations, several variables and equations are likely to be unrelated, and when such a situation occurs, partitioning can be used to facilitate substantial reduction of the time required to determine all the minimal solutions. Therefore, considering partitioning when solving fuzzy relational equations is crucial.

Partitioning is not useful when all of the rows and columns of a binary binding matrix M are related, and after applying forced binding (Rule 5 in Section 5) to derive two submatrices $M^{\times}$ and M^{-}, all of the rows and columns of both submatrices are still related. Intuitively, this situation occurs

as the density of M approaches one (i.e., d is near 1 and $p = 1$, as shown in Figure 2). In this situation, the problem contains considerably fewer irredundant coverings, as shown in Figure 2, and, thus, can be solved efficiently with or without considering partitioning.

In addition to max-Archimedean t-norm fuzzy relational equations, numerous types of fuzzy relational equations have been demonstrated to be equivalent to the set covering problem [12]. The partitioning concept and the divide-and-conquer approach discussed in this paper can also be applied to these fuzzy relational equations with little modification.

Instead of generating test matrices (as in Algorithm 2), Hu and Fang [13] proposed procedures for generating test problems with various characteristics for use in max-t-norm fuzzy relational equations, with t being the min, product, or the Łukasiewicz t-norm. Because both the product and Łukasiewicz t-norm are Archimedean t-norms, test problems generated by conducting the related procedures can be used in future studies to evaluate the performance of the proposed method further.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research is supported by the National Science Council under Grants 99-2221-E-155-048-MY3 and 102-2221-E-155-034-MY3.

References

[1] B. De Baets, "Analytical solution methods for fuzzy relational equations," in *Fundamentals of Fuzzy Sets*, vol. 7, pp. 291–340, Kluwer Academic, Boston, Mass, USA, 2000.

[2] K. Peeva and Y. Kyosev, *Fuzzy Relational Calculus: Theory, Applications And Software*, vol. 22, World Scientific, Hackensack, NJ, USA, 2004.

[3] A. Di Nola, S. Sessa, W. Pedrycz, and E. Sanchez, *Fuzzy Relation Equations and Their Applications to Knowledge Engineering*, vol. 3, Kluwer Academic, Dordrecht, The Netherlands, 1989.

[4] P. Li and S.-C. Fang, "On the resolution and optimization of a system of fuzzy relational equations with sup-T composition," *Fuzzy Optimization and Decision Making*, vol. 7, no. 2, pp. 169–214, 2008.

[5] P. Li and S.-C. Fang, "A survey on fuzzy relational equations. I. Classification and solvability," *Fuzzy Optimization and Decision Making*, vol. 8, no. 2, pp. 179–229, 2009.

[6] A. V. Markovskii, "On the relation between equations with max-product composition and the covering problem," *Fuzzy Sets and Systems*, vol. 153, no. 2, pp. 261–273, 2005.

[7] Y.-K. Wu and S.-M. Guu, "An efficient procedure for solving a fuzzy relational equation with max-Archimedean t-norm composition," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 1, pp. 73–84, 2008.

[8] J.-L. Lin, "On the relation between fuzzy max-Archimedean t-norm relational equations and the covering problem," *Fuzzy Sets and Systems*, vol. 160, no. 16, pp. 2328–2344, 2009.

[9] G. B. Stamou and S. G. Tzafestas, "Resolution of composite fuzzy relation equations based on Archimedean triangular norms," *Fuzzy Sets and Systems*, vol. 120, no. 3, pp. 395–407, 2001.

[10] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.

[11] P. S. Mostert and A. L. Shields, "On the structure of semigroups on a compact manifold with boundary," *Annals of Mathematics*, vol. 65, pp. 117–143, 1957.

[12] J.-L. Lin, Y.-K. Wu, and S.-M. Guu, "On fuzzy relational equations and the covering problem," *Information Sciences*, vol. 181, no. 14, pp. 2951–2963, 2011.

[13] C.-F. Hu and S.-C. Fang, "Randomly generating test problems for fuzzy relational equations," *Fuzzy Optimization and Decision Making*, vol. 11, no. 1, pp. 1–28, 2012.