Research Article

Assignment Problem for Team Performance Promotion under Fuzzy Environment

Chi-Jen Lin

Department of Industrial Engineering and Management, TaHwa University of Science and Technology, No. 1 TaHwa Road, Cyong-Lin, HsinChu County 307, Taiwan

Correspondence should be addressed to Chi-Jen Lin; lcj@tust.edu.tw

Received 27 November 2012; Revised 23 May 2013; Accepted 27 May 2013

Academic Editor: Asier Ibeas

Copyright © 2013 Chi-Jen Lin. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper constructs a general fuzzy assignment problem (GFAP) based on a real-world scenario and proposes a solution procedure. Suppose a project team consists of \( n \) workers and a manager. The \( n \) workers are responsible for performing \( n \) jobs and the manager for restraining the total cost. The corresponding cost for a worker to perform his assigned job is not defined deterministically but as a subnormal fuzzy interval with increasing linear membership function. Job quality is then linearly and positively related to the cost of the job and is taken as the performance of the worker. On the other hand, the performance of the manager is negatively related to the total cost and is defined as a fuzzy interval with a decreasing linear membership function. It is common practice for a company to regard the lowest performance among members as the team performance in order to increase overall team performance. Hence, using the max–min criterion, a mixed nonlinear programming model of the GFAP is constructed. The model can be transformed into a general 0-1 fractional programming problem with max–min objective function. An algorithm that combines simplex and trade-off approaches is proposed to solve the problem. A numerical example and the computational results show that the constructed model and the proposed algorithm are useful and efficient.

1. Introduction

The assignment problem (AP) is a well-known and practical mathematical model. Given two disjoint \( n \)-sets \( I = \{1,2,\ldots,n\} \), \( J = \{1,2,\ldots,n\} \) and the set \( K = \{(i,j) \mid i \in I \text{ and } j \in J\} \), the AP can be stated as the following 0-1 programming problem:

\[
\text{min} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \quad (c_{ij} \geq 0)
\]

s.t. \( \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i \in I \)

\( \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j \in J \)

\( x_{ij} \in \{0,1\} \quad \text{for } (i,j) \in K. \)

Let \( X \) denote a feasible solution of (1). In a simplex-type algorithm, \( X \) is highly degenerate and consists of exactly \( n \) variables with the value 1; the remaining \( n^2-n \) variables have the value 0. Defining \( \omega = \{(i,j) \mid x_{ij} = 1, x_{ij} \in X\}, \omega \) constitutes the assignment corresponding to \( X \). Often (1) is practically interpreted as \( n \) jobs that must be performed by \( n \) workers at minimal total cost, and \( c_{ij} \) is the job cost associated with worker \( i \) performing job \( j \). Here, \( \omega \) implies that each job must be assigned to one and only one worker, and vice versa. In the model, all \( c_{ij} \) values are deterministic.

However, in many real-world applications, job costs are not deterministic. Fuzzy theory has been supported to slash the risk of improper models and solutions which do not reflect the practical problems [1]. In recent years, many researchers have investigated the AP and its variants under fuzzy environments. Sakawa et al. [2] dealt with production and workforce assignment problems of a housing material manufacturer. Belacela and Boulasselb [3] developed

In this paper, a general FAP (GFAP) of Lin and Wen [5] is presented and an efficient solution procedure is proposed. The GFAP excludes the upper bound constraint on total cost, which makes the model more complicated and expands the application area. In addition to the team performance in human resource management, the GFAP can be used to situations that equally emphasize the time and cost, the quality and quantity, the humanity and technology, and so forth. The remainder of this paper is structured as follows. In the next section, the GFAP is constructed as a mixed nonlinear programming model. In Section 3, the constructed model is transformed into a general 0-1 fractional programming problem. The proposed algorithm for solving this problem is described in Section 4. A numerical example is illustrated in Section 5. In Section 6, computational results and two special cases are discussed. Finally, conclusions and suggestions for further research are given in Section 7.

2. Model Construction

Suppose that a project team consists of n workers and a manager. The n workers are responsible for performing n jobs and the manager for restraining the total cost. In many applications, the cost of a job depends not only on the skill of the worker but also on the job quality achieved. In general, the higher the job quality achieved, the greater the job cost required, and since a worker’s performance is defined as job quality, the worker desires a job cost in order to raise his personal performance. On the other hand, the manager is required to restrain the total cost as much as possible. Thus, the total cost is negatively related to the performance of the manager. \( \bar{c}_j \) and \( \bar{c}_T \) denote the cost of worker \( i \) performing job \( j \) and the total cost, respectively.

Let \( Q_{ij} (0 < Q_{ij} \leq 1) \) denote the highest achievable job quality \( q \) performed by worker \( i \). The membership function of job cost \( \bar{c}_j \) is defined as the monotonically increasing linear function shown in (2). In this function, \( \alpha_{ij} \) is the minimum cost for worker \( i \) performing job \( j \), and \( \beta_{ij} \) is the minimum cost associated with worker \( i \) performing job \( j \) to reach \( Q_{ij} \). When the input job cost lies between \( \alpha_{ij} \) and \( \beta_{ij} \), the job cost is linearly and positively related to the job quality, that is, the performance of the worker. However, any expense exceeding \( \beta_{ij} \) is inefficient since the quality can no longer be enhanced.

Without loss of generality, it is assumed that \( 0 < \alpha_{ij} < \beta_{ij} \). If \( x_{ij} = 0 \) in any feasible solution, there is no real expense; therefore \( x_{ij} = 1 \) is added to the following:

\[
\mu_{ij}(c_{ij}) = \begin{cases} 
q_{ij} & \text{if } \gamma_{ij} \geq \beta_{ij}, \ x_{ij} = 1 \\
\frac{Q_{ij}(c_{ij} - \alpha_{ij})}{\beta_{ij} - \alpha_{ij}} & \text{if } \alpha_{ij} \leq c_{ij} \leq \beta_{ij}, \ x_{ij} = 1 \\
0 & \text{otherwise}
\end{cases}
\]  
(2)

\( \gamma_{ij} \) denotes \( \bar{c}_j \). Matrix \( \bar{c}_{ij} \) is shown as follows:

\[
\bar{c}_{ij} = \begin{bmatrix} \alpha_{ij} \\ \beta_{ij} \end{bmatrix}
\]  
(3)

In addition, the \( \alpha_{ij} \) values form the matrix \( [\alpha_{ij}] \), the \( \beta_{ij} \) values form the matrix \( [\beta_{ij}] \), and the \( Q_{ij} \) values form the matrix \( [Q_{ij}] \).

Further, we define

\[
\gamma_{ij} = \left[ \begin{array}{c} \beta_{ij} - \alpha_{ij} \\ Q_{ij} \end{array} \right] \quad \text{for } (i, j) \in K,
\]  
(4)

where \( \gamma_{ij} \) is indeed the slope of the membership function.

The membership function of total cost \( \bar{c}_T \) is defined as the monotonically decreasing linear function shown in (5). \( a \) and \( b \) are the lower and upper bounds of \( \bar{c}_T \), respectively.

Total cost is related to the performance of the manager. When the total cost is lower than \( a \), his performance is \( 1 \); when the total cost is greater than \( b \), his performance is \( 0 \); and when the total cost lies between \( a \) and \( b \), the total cost is linearly and negatively related to the performance of the manager. \( \langle a, b \rangle \) denotes the fuzzy interval \( \bar{c}_T \):

\[
\mu_T(c_T) = \mu_T \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \right) = \begin{cases} 
1 & \text{if } c_T \leq a \\
\frac{b - \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}}{b - a} & \text{if } a \leq c_T \leq b \\
0 & \text{if } c_T \geq b
\end{cases}
\]  
(5)

Since the performance of worker \( i \), denoted as \( \pi_i \), is the job quality achieved, then

\[
\pi_i = \mu_{ij}(c_{ij}) \quad \text{for } x_{ij} = 1, \ i = 1, 2, \ldots, n.
\]  
(6)

In addition, the performance of the manager, denoted as \( \pi_m \), is as follows:

\[
\pi_m = \mu_T \left( \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \right).
\]  
(7)

Usually, the company takes the lowest performance among members of the team, including all workers and the manager, as the team performance in order to increase overall performance. To maximize, the performance of each team
Mathematical Problems in Engineering

member has to be equally emphasized. Hence, the Bellman-Zadeh criterion [12], which maximizes the minimum of all membership functions corresponding to that solution, is selected for this problem. Therefore,

$$\max_{\pi_i, \pi_m} \min_{\forall i} \left( \pi_i, \pi_m \right)$$

(8)

or

$$\max_{x_i \in \mathbb{R}} \min_{x_i \in \mathbb{R}} \left( \mu_i(x_i) \right)$$

(9)

where $x_{ij}$ is an element of a feasible solution $X$ of (1). The FAP model can then be constructed as follows:

$$\max_{x_i \in \mathbb{R}} \min_{x_i \in \mathbb{R}} \left( \mu_i(x_i) \right)$$

s.t. $\sum_{j=1}^{n} x_{ij} = 1$ for $i \in I$

$$\sum_{i=1}^{n} x_{ij} = 1$$

for $j \in J$

$x_{ij} \in \{0, 1\}$ for $(i, j) \in K$.

Utilizing the membership functions illustrated in (2) and (5), (10) can be represented as the following equivalent problem:

$$\max \lambda$$

s.t. $\lambda \cdot x_{ij} \leq \frac{Q_{ij} (c_{ij}^\lambda - \alpha_{ij})}{\beta_{ij} - \alpha_{ij}} x_{ij}$ for $(i, j) \in K$

$$\lambda \leq \frac{b - \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^\lambda x_{ij}}{b - a}$$

$$\sum_{j=1}^{n} x_{ij} = 1$$ for $i \in I$

$$\sum_{i=1}^{n} x_{ij} = 1$$ for $j \in J$

$$0 \leq \lambda x_{ij} \leq Q_{ij}^\lambda x_{ij}$$ for $(i, j) \in K$

$c_{ij}^\lambda \geq 0$, $x_{ij} \in \{0, 1\}$ for $(i, j) \in K$,

where $c_{ij}^\lambda$ denotes the $\lambda$-cut of $c_{ij}$ and $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^\lambda x_{ij}$ is the corresponding total cost $c_{ij}^\lambda$. Since $x_{ij}$, $c_{ij}^\lambda$, and $\lambda$ all are decision variables, (11) can be treated as a mixed nonlinear programming problem.

3. Model Transformation

Let $\omega$ be a specific feasible assignment of (1), that is, of (10). If discussion is confined on $\omega$, (11) can be simplified as follows:

$$\max \lambda$$

s.t. $\lambda \leq \frac{Q_{ij} (c_{ij}^\lambda - \alpha_{ij})}{\beta_{ij} - \alpha_{ij}}$ for $(i, j) \in \omega$

$$c_{ij}^\lambda \geq 0, \lambda \geq 0$$ for $(i, j) \in \omega$.

Theorem 1. Let $\lambda^*$ be the optimal objective value of (12) corresponding to $\omega$, then

$$\lambda^* = \max \min \left( \frac{b - \sum_{(i, j) \in \omega} c_{ij}}{b - a}, \min_{(i, j) \in \omega} \left( Q_{ij} \right) \right).$$

(13)

Proof. Transforming (12) into a linear programming problem, we find the following:

$$\max \lambda$$

s.t. $-c_{ij}^\lambda + \frac{Q_{ij}}{Q_{ij}} \cdot \lambda \leq -\alpha_{ij}$ for $(i, j) \in \omega$

$$\sum_{(i, j) \in \omega} c_{ij}^\lambda + (b - a) \cdot \lambda \leq b$$

$$\lambda \leq \min_{(i, j) \in \omega} \left( Q_{ij} \right)$$

$$c_{ij}^\lambda \geq 0, \lambda \geq 0$$ for $(i, j) \in \omega$.

The dual problem of (14) can then be obtained as follows:

$$\min \sum_{(i, j) \in \omega} \left( \alpha_{ij} t_{ij} \right) + bt_{n+1} + \min_{(i, j) \in \omega} \left( Q_{ij} \right) \cdot y$$

s.t. $-t_{i} + t_{n+1} \geq 0$ for $i \in I$

$$\sum_{(i, j) \in \omega} \left( \beta_{ij} - \alpha_{ij} \right) t_{ij} + (b - a) t_{n+1} + y \geq 1$$

(15)

$$t_{i} \geq 0, \quad y \geq 0.$$}

By (14), $c_{ij}^\lambda \geq \alpha_{ij} + ((\beta_{ij} - \alpha_{ij})/Q_{ij}) \cdot \lambda > 0$ holds for $(i, j) \in \omega$. According to the complementary slackness theorem,

$$-t_{i} + t_{n+1} = 0 \quad \text{for } i \in I$$

(16)

or

$$t_{n+1} = t_{1} = t_{2} = \cdots = t_{n}.$$
Using (17) and (4), (15) can be simplified into the following:

\[
\begin{align*}
\min & \quad (b - \sum_{(i,j) \in \omega} \alpha_{ij}) t_{m+1} + \min_{(i,j) \in \omega} (Q_{ij}) \cdot y \\
\text{s.t.} & \quad \left( b - a + \sum_{(i,j) \in \omega} y_{ij} \right) t_{m+1} + y \geq 1 \\
& \quad t_{m+1} \geq 0, \quad y \geq 0
\end{align*}
\]

(18)

The dual problem of (18) is as follows:

\[
\begin{align*}
\max & \quad \lambda \\
\text{s.t.} & \quad \left( b - a + \sum_{(i,j) \in \omega} y_{ij} \right) \cdot \lambda \leq b - \sum_{(i,j) \in \omega} \alpha_{ij} \\
& \quad \lambda \leq \min_{(i,j) \in \omega} (Q_{ij}) \\
& \quad \lambda \geq 0
\end{align*}
\]

(19)

(19) can be rewritten as follows to complete the proof:

\[
\begin{align*}
\max & \quad \lambda \\
\text{s.t.} & \quad \lambda \leq \frac{b - \sum_{(i,j) \in \omega} \alpha_{ij}}{b - a + \sum_{(i,j) \in \omega} y_{ij}} \\
& \quad \lambda \leq \min_{(i,j) \in \omega} (Q_{ij}) \\
& \quad \lambda \geq 0
\end{align*}
\]

(20)

Obviously, \( \lambda^* \) depends on the given \( \omega \); that is, \( \lambda^* \) is a function of \( \omega \). The following corollary is trivial.

**Corollary 2.** Let \( \omega_1, \omega_2, \ldots, \omega_p \) be all of the feasible assignments of (1). \( \lambda_1^*, \lambda_2^*, \ldots, \lambda_p^* \) are the corresponding optimal objective function values in (12). Let \( \lambda^*_1 = \max(\lambda_1^*, \lambda_2^*, \ldots, \lambda_p^*) \); then \( \omega_1^* \) (subsequently denoted by \( \omega^* \)) is the optimal assignment of (11) and \( \lambda^*_1 \) (subsequently denoted by \( z^* \)) is the optimal objective function value in (11).

By Theorem 1 and Corollary 2, (11) can be transformed to (21), which is a general 0-1 fractional programming problem:

\[
\begin{align*}
\max & \quad \min \left( \frac{b - \sum_{i=1}^{n} \alpha_{ij} x_{ij}}{(b - a + \sum_{j=1}^{n} y_{ij} x_{ij})} \right) \\
& \quad \min \left( 1 - (1 - Q_{ij}) x_{ij} \right) \quad \forall (i, j) \in K \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in I \\
& \quad x_{ij} \in \{0, 1\} \quad \forall (i, j) \in K
\end{align*}
\]

(21)

**4. Solution Algorithm**

Of course, (21) can be solved by branch and bound (B&B) algorithms. However, in this section the S&T algorithm—a hybrid algorithm employing simplex and trade-off approaches [13]—is proposed to obtain the optimal assignment and optimal objective function value in (21). There is a tradeoff between the total cost and the jobs quality achieved, that is, between the performance of manager and workers. The S&T algorithm balances the maximum of manager’s performance and the minimum of the workers’ performances to maximize team performance. Usually, some S&T algorithm procedures must be repeated several times to solve problems with different updated parameters. In each solution, the maximum of manager’s performance decreases (or remains unchanged) and the minimum of the workers’ performances simultaneously increases until the maximal team performance is reached.

To obtain the maximal manager’s performance, (22) is extracted from (21) as follows:

\[
\begin{align*}
\max & \quad f = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\alpha_{ij} x_{ij}}{n} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in I \\
& \quad \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in J \\
& \quad x_{ij} \in \{0, 1\} \quad \forall (i, j) \in K
\end{align*}
\]

(22)

Since the constraints of (22) form a totally unimodular matrix and its objective function is a pseudomonotonic model [14], the condition \( x_{ij} \in \{0, 1\} \) in (22) can be replaced with \( x_{ij} \geq 0 \). Furthermore, we can define \( \overline{\alpha}_{ij} = b/n - \alpha_{ij} \) and \( \overline{y}_{ij} = (b - a)/n + y_{ij} \). Equation (22) then becomes equivalent to the following:

\[
\begin{align*}
\max & \quad f = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\overline{\alpha}_{ij} x_{ij}}{\overline{y}_{ij} x_{ij}} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in I \\
& \quad \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in J \\
& \quad x_{ij} \geq 0 \quad \forall (i, j) \in K
\end{align*}
\]

(23)

Equation (23) is a special linear fractional problem, that is, a fractional assignment problem. Some algorithms for
solving fractional assignment problems have been developed by Shigeno et al. [15] and Kabadi and Punnen [14] who showed that the optimum of (23) can be found at an extreme point of its solution polyhedron. A simplex-type algorithm can then be employed to solve (23).

The S&T algorithm consists of initialization and optimization procedures. The initialization procedure is first performed in a transportation table, as shown in Figure I. The FCMP method [16] is employed to obtain a basic feasible solution \( X^0 \) of the maximal-objective-function transportation problem with \([\alpha_{ij}/y_{ij}]\) as the profit matrix and 1 as the amount of each supply and demand. \( X^0 \) is then used as the initial solution to (23), as shown in Figure 2. This is viewed as the fractional transportation problem [17]. Next, the objective function value \( f \) of \( X^0 \) in (23) is determined and associates \( u_i \) and \( v_j \) with row \( i \) and column \( j \), respectively. Set \( u_i = 0 \) and identify the other \( u_i \) and \( v_j \) using the equation \( u_i + v_j = \alpha_{ij} - f \cdot y_{ij} \) according to the basic variables of \( X^0 \). Define the assignment as \( \omega = \{(i, j) \mid x_{ij} = 1, x_{ij} \in X^0\} \). Remove the degenerated basic variables of \( X^0 \) from the table.

Taking \( \omega \) as the initial assignment, subprocedures mixed approaches of [5, 11, 18–20] are used repeatedly to improve \( \omega \). When \( \Delta_{ij} = u_i + v_j + f \cdot y_{ij} - \alpha_{ij} \geq 0 \) for each \((i, j) \notin \omega \), \( \omega \) is the optimal assignment of (23). Based on \( \omega \), the other condition in (21) is extracted. Let

\[
G = \min_{(i,j) \notin \omega} (Q_{ij}) ;
\]

then \( z = \min (f, g) \) is the objective function value in (21) corresponding to \( \omega \). To find the assignment \( \omega^* \) that maximizes the objective function value in (21), elements of matrix \([\alpha_{ij}]/y_{ij}\) can be repeatedly changed so that \( f \) decreases or remains constant and \( g \) increases to improve \( z \) until the optimal objective function value \( z^* \) in (21) is reached. The proposed S&T algorithm consisting of initialization and optimization procedures is summarized as follows.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{The results of the FCMP method.}
\end{figure}

**Phase I. Initialization**

**Step 1:** Let \( z^* \to -\infty \) and \( \omega^* = \emptyset \). Apply the FCMP method to get \( X^0 \) and \( f \).

**Step 2:** Determine all values of \( u_i \) and \( v_j \) according to \( X^0 \).

**Step 3:** Remove the degenerate basic variables of \( X^0 \) and obtain \( \omega \).

**Phase II. Optimization**

**Step A:** Compute \( \Delta_{ij} \), for each \((i, j) \notin \omega \). If all \( \Delta_{ij} \geq 0 \); then proceed to Step E. Otherwise, choose \((k, \ell)\) as the pivot cell where \( \Delta_{k\ell} = \min (\Delta_{ij}) < 0 \).

**Step B:** Proceed with the following labeling procedure until further labeling is impossible or row \( k \) has received a label. If row \( k \) has been labeled, go to Step C; otherwise, go to Step D.

**Labeling Procedure**

1. **Give label \( k \) to column \( \ell \).**
2. **If column \( q \) has been labeled, label the row \( i \) with \( q \), where \( x_{iq} = 1 \).**
3. **If row \( p \) has been labeled, label the nonlabeled column \( j \) with \( p \), where \( \Delta_{pj} = 0 \).**

**Step C:** Form a unique cycle by following the labels in reverse order. In the cycle, every odd cell is \( x_{ij} = 0 \), and every even cell is \( x_{ij} = 1 \). The odd cells are the recipient cells and thus become \( x_{ij} = 1 \); the even cells are the donor cells, which become \( x_{ij} = 0 \). Let \( x_{ij} \) remain unchanged if cell \((i, j)\) is not in the cycle. The new assignment is obtained and is taken as the new \( \omega \). The objective function value corresponding to the new \( \omega \) is taken as the new \( f \).

By keeping \( u_k \) unchanged, the new values of \( v_\ell \) and \( v_{\ell(k)} \) can be obtained by the following:

\[
\begin{align*}
\nu_\ell &= \bar{\alpha}_{k\ell} - u_k - f \cdot y_{k\ell}, \\
\nu_{\ell(k)} &= \bar{\alpha}_{k,\ell(k)} - u_k - f \cdot y_{k,\ell(k)},
\end{align*}
\]

where \( L(k) \) is the label of row \( k \).

Other new variables \( u_i \) and \( v_j \) related to the cycle can be obtained by keeping \( \Delta_{ij} = 0 \) in the cycle. If the new values of \( u_i \) and \( v_j \) are not related to the cycle, hold \( u_i \) unchanged and identify \( v_j = \bar{\alpha}_{ij} - u_i - f \cdot y_{ij} \), where \((i, j) \in \omega \). Return to Step A.

**Step D:** Let \( T = \{(i, j) \mid \Delta_{ij} > 0 \text{ and row } i \text{ labeled, but column } j \text{ is not labeled}\}, e = \min_{(i,j)\notin T} (|\Delta_{ij}|); \text{ or } e = |\Delta_{k\ell}| \text{ if } T \text{ is a null set.}

If row \( i \) is labeled, \( e \) is subtracted from \( u_i \); and column \( j \) is labeled, \( e \) is added to \( v_j \). Recompute \( \Delta_{ij} \) for \((i, j) \notin \omega \). If all \( \Delta_{ij} \geq 0 \), proceed to Step E. If \( \Delta_{k\ell} < 0 \) still holds, take \((k, \ell)\) as the pivot cell; if \( \Delta_{k\ell} \geq 0 \), choose \((\tilde{k}, \tilde{\ell})\) as the new pivot cell where \( \Delta_{k\ell} = \min (\Delta_{ij}) < 0 \). Replace \( k \) with \( \tilde{k} \), \( \ell \) with \( \tilde{\ell} \), and return to Step B.

**Step E:** Determine \( g = \min_{(i,j)\notin w} (Q_{ij}) \) and \( z = \min (f, g) \). If \( z \leq z^* \), proceed to Step H; otherwise, proceed to Step F.
Step F. Let $z^* = z$ and $\omega^* = \omega$. If $z \geq f$, proceed to Step H; otherwise, proceed to Step G.

Step G. Update $[\bar{c}_{ij}]$, where

$$[\bar{c}_{ij}] = \begin{bmatrix} \bar{c}_{ij} & \text{if } Q_{ij} > z^* \\ -M (M \rightarrow \infty) & \text{if } Q_{ij} \leq z^* \end{bmatrix} \text{ (26)}$$

Revise the objective function value $f$ corresponding to $\omega$. To renew $u_i$ and $v_j$, hold $u_i$ unchanged and identify $v_j = \bar{c}_{ij} - u_i - f \cdot \bar{V}_{ij}$, where $(i, j) \in \omega$. Go back to Step A.

Step H. Terminate the algorithm. The current $\omega^*$ and $z^*$ are the optimal assignment and optimal value of (21), respectively.

5. Numerical Example

In this section, for demonstration purpose, the S&T algorithm is applied to a numerical example.

Suppose the matrices $[\bar{c}_{ij}]$ and $[Q_{ij}]$ are

$$[\bar{c}_{ij}] = \begin{bmatrix} (5, 22) & (7, 16) & (23, 39) \\ (5, 21) & (21, 31) & (16, 31) \\ (14, 28) & (23, 32) & (5, 15) \end{bmatrix}, \text{ (27)}$$

$$[Q_{ij}] = \begin{bmatrix} 0.85 & 0.6 & 0.8 \\ 0.64 & 0.8 & 0.75 \\ 0.7 & 0.72 & 0.8 \end{bmatrix}; \text{ (28)}$$

then define matrices

$$[\alpha_{ij}] = \begin{bmatrix} 5 & 7 & 23 \\ 5 & 21 & 16 \\ 14 & 23 & 5 \end{bmatrix}, \quad [\beta_{ij}] = \begin{bmatrix} 22 & 16 & 39 \\ 21 & 31 & 31 \\ 28 & 32 & 15 \end{bmatrix},$$

and

$$[\gamma_{ij}] = \begin{bmatrix} 20 & 15 & 20 \\ 25 & 12.5 & 20 \\ 20 & 12.5 & 12.5 \end{bmatrix}.$$

Specify the sum of maximal elements in each row of matrix $[\bar{c}_{ij}]$ as $b = 102$ and the sum of minimal elements in each

![Figure 2: Computing values of $u_i$, $v_j$, and $\Delta_{ij}$.](image)

6. Computational Experience

The computational test in this section verifies the efficiency of the proposed S&T algorithm. An index-based B&B algorithm is also used to solve (21). Both the S&T algorithm and the index-based B&B algorithm are coded in Visual Basic 6.0. They are compared with the global optimal function in...
### Figure 3: Set of table for Optimization Procedure.

<table>
<thead>
<tr>
<th></th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>$u_1 = 0$</td>
<td>$\frac{-5104 - 1911M}{279}$</td>
<td>$\frac{5029 - 15M}{279}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{2901 + 10M}{279}$</td>
<td>49</td>
<td>1</td>
</tr>
<tr>
<td>[1]</td>
<td>$u_2 = \frac{3481}{279}$</td>
<td>$\frac{-7398 - 274M}{279}$</td>
<td>$\frac{-405 - 15M}{279}$</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>[3]</td>
<td>$u_3 = \frac{863}{279}$</td>
<td>$\frac{2496 - 274M}{279}$</td>
<td>$\frac{29}{279}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{6275 + 10M}{279}$</td>
<td>49</td>
<td>1</td>
</tr>
</tbody>
</table>

(a)
Table 1: Computational results for the general case.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>B&amp;B algorithm Average time (sec.)</th>
<th>S&amp;T algorithm Average time (sec.)</th>
<th>LINGO Average time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 × 3</td>
<td>—</td>
<td>3.333 × 10^-4</td>
<td>1.90667</td>
</tr>
<tr>
<td>4 × 4</td>
<td>6.6667 × 10^-4</td>
<td>1.3333 × 10^-3</td>
<td>10.545</td>
</tr>
<tr>
<td>5 × 5</td>
<td>1.3333 × 10^-3</td>
<td>4.333 × 10^-3</td>
<td>28.7183</td>
</tr>
<tr>
<td>6 × 6</td>
<td>4.6667 × 10^-3</td>
<td>7.666 × 10^-3</td>
<td>34.0763</td>
</tr>
<tr>
<td>7 × 7</td>
<td>6.7 × 10^-3</td>
<td>1.0 × 10^-2</td>
<td>34.9223</td>
</tr>
<tr>
<td>8 × 8</td>
<td>2.203 × 10^-2</td>
<td>1.4333 × 10^-2</td>
<td>61.9323</td>
</tr>
<tr>
<td>9 × 9</td>
<td>7.41 × 10^-2</td>
<td>2.303 × 10^-2</td>
<td>85.8511</td>
</tr>
<tr>
<td>10 × 10</td>
<td>0.2236</td>
<td>3.67 × 10^-2</td>
<td>131.4427</td>
</tr>
<tr>
<td>11 × 11</td>
<td>0.6369</td>
<td>5.5067 × 10^-2</td>
<td>145.2227</td>
</tr>
<tr>
<td>12 × 12</td>
<td>3.4179</td>
<td>8.2433 × 10^-2</td>
<td>216.4897</td>
</tr>
<tr>
<td>13 × 13</td>
<td>7.9434</td>
<td>0.1225</td>
<td>374.1963</td>
</tr>
<tr>
<td>14 × 14</td>
<td>42.0261</td>
<td>0.17357</td>
<td>466.2703</td>
</tr>
<tr>
<td>15 × 15</td>
<td>128.2167</td>
<td>0.24067</td>
<td>656.5563</td>
</tr>
<tr>
<td>16 × 16</td>
<td>516.42</td>
<td>0.3388</td>
<td>870.1540</td>
</tr>
</tbody>
</table>

Table 2: Computational results when \(b \to \infty\).

<table>
<thead>
<tr>
<th>Problem size</th>
<th>B&amp;B algorithm Average time (sec.)</th>
<th>S&amp;T algorithm Average time (sec.)</th>
<th>LINGO Average time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 × 3</td>
<td>—</td>
<td>1.3333 × 10^-3</td>
<td>13.141</td>
</tr>
<tr>
<td>4 × 4</td>
<td>3.3333 × 10^-4</td>
<td>5 × 10^-3</td>
<td>26.9487</td>
</tr>
<tr>
<td>5 × 5</td>
<td>6.6667 × 10^-4</td>
<td>8.6667 × 10^-3</td>
<td>62.3437</td>
</tr>
<tr>
<td>6 × 6</td>
<td>1.7 × 10^-3</td>
<td>1.47 × 10^-2</td>
<td>33.1177</td>
</tr>
<tr>
<td>7 × 7</td>
<td>3 × 10^-3</td>
<td>2.4367 × 10^-2</td>
<td>41.2426</td>
</tr>
<tr>
<td>8 × 8</td>
<td>6.33 × 10^-3</td>
<td>4.1733 × 10^-2</td>
<td>40.42</td>
</tr>
<tr>
<td>9 × 9</td>
<td>5.33 × 10^-3</td>
<td>7.1766 × 10^-2</td>
<td>49.6247</td>
</tr>
<tr>
<td>10 × 10</td>
<td>4.27 × 10^-2</td>
<td>0.11916</td>
<td>59.917</td>
</tr>
<tr>
<td>11 × 11</td>
<td>2.003 × 10^-2</td>
<td>0.19493</td>
<td>64.9183</td>
</tr>
<tr>
<td>12 × 12</td>
<td>3.873 × 10^-2</td>
<td>0.29843</td>
<td>84.1493</td>
</tr>
<tr>
<td>13 × 13</td>
<td>0.2944</td>
<td>0.4453</td>
<td>94.3677</td>
</tr>
<tr>
<td>14 × 14</td>
<td>1.603</td>
<td>0.6356</td>
<td>119.5947</td>
</tr>
<tr>
<td>15 × 15</td>
<td>0.3962</td>
<td>0.89496</td>
<td>145.8433</td>
</tr>
<tr>
<td>16 × 16</td>
<td>39.8416</td>
<td>1.24846</td>
<td>173.3940</td>
</tr>
</tbody>
</table>

The sizes of the problems tested range from 3 × 3 to 16 × 16. For each problem of \(N \times N\) size, numbers of uniform distribution between 10 and 10 + 20\(N\) are randomly generated for \(\alpha_{ij}\) and numbers between \(6N\) and \(10N\) are generated for the difference between \(\beta_{ij}\) and \(\alpha_{ij}\). Numbers with a uniform distribution between 0.6 and 1.0 are generated for \(Q_{ij}\). For each problem size, 30 problems are tested and the average computational times (in seconds) of the three approaches are shown in Table 1. These results show that from the perspective of computational time, the S&T algorithm is greatly superior to the LINGO package. In particular, the S&T algorithm is superior to the index-based B&B algorithm when dealing with an increasing problem size.

Depending on the random number generated, the manager or one of the workers may be the one who decides the team performance. Now, we move to a discussion of two special cases. In the first case, it is assumed that the company supports the team at any expense. Thus, \(b \to \infty\) and the computational results of the three approaches are shown in Table 2. They illustrate that, although a small problem size causes the B&B algorithm to perform faster, the S&T algorithm is still efficient. The elapsed time of the LINGO package is also reduced. However, the variances of the elapsed times of the B&B algorithm and the LINGO package are large. Practically, \(b \to \infty\) causes the membership function in (5) to approach \(\mu_T(c_T) = 1\). Thus, the manager is never the one who determines the team performance. The workers also obtain sufficient expenses and can all reach their highest quality (performance). The team performance is decided by one of the workers. Therefore, \(b \to \infty\) simplifies (21) to the following:

\[
\begin{align*}
\max \min \quad & \left( (1 - (1 - Q_{ij}) x_{ij}) ; \forall (i, j) \in K \right) \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1; \quad \forall i \in I \\
& \sum_{i=1}^{n} x_{ij} = 1; \quad \forall j \in J \\
& x_{ij} \in \{0, 1\}; \quad \forall (i, j) \in K.
\end{align*}
\]
Table 3: Computational results when $b$ is sufficiently small.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Index-based B&amp;B algorithm</th>
<th>S&amp;T algorithm</th>
<th>LINGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3 \times 3$</td>
<td>$3.3333 \times 10^{-4}$</td>
<td>$6.667 \times 10^{-4}$</td>
<td>$0.405$</td>
</tr>
<tr>
<td>$4 \times 4$</td>
<td>$6.6667 \times 10^{-4}$</td>
<td>$1.6667 \times 10^{-3}$</td>
<td>$0.8493$</td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>$1.6667 \times 10^{-3}$</td>
<td>$3 \times 10^{-3}$</td>
<td>$1.6643$</td>
</tr>
<tr>
<td>$6 \times 6$</td>
<td>$4.6667 \times 10^{-3}$</td>
<td>$6 \times 10^{-3}$</td>
<td>$2.9837$</td>
</tr>
<tr>
<td>$7 \times 7$</td>
<td>$7 \times 10^{-3}$</td>
<td>$7.6667 \times 10^{-3}$</td>
<td>$5.0093$</td>
</tr>
<tr>
<td>$8 \times 8$</td>
<td>$2.27 \times 10^{-2}$</td>
<td>$1.0033 \times 10^{-2}$</td>
<td>$7.962$</td>
</tr>
<tr>
<td>$9 \times 9$</td>
<td>$7.6133 \times 10^{-2}$</td>
<td>$1.2367 \times 10^{-2}$</td>
<td>$13.798$</td>
</tr>
<tr>
<td>$10 \times 10$</td>
<td>$0.2226$</td>
<td>$1.67 \times 10^{-2}$</td>
<td>$18.0663$</td>
</tr>
<tr>
<td>$11 \times 11$</td>
<td>$0.9057$</td>
<td>$2.27 \times 10^{-2}$</td>
<td>$26.4913$</td>
</tr>
<tr>
<td>$12 \times 12$</td>
<td>$3.725$</td>
<td>$2.8367 \times 10^{-2}$</td>
<td>$39.366$</td>
</tr>
<tr>
<td>$13 \times 13$</td>
<td>$11.3644$</td>
<td>$3.94 \times 10^{-2}$</td>
<td>$84.0933$</td>
</tr>
<tr>
<td>$14 \times 14$</td>
<td>$54.5097$</td>
<td>$5.2767 \times 10^{-2}$</td>
<td>$90.954$</td>
</tr>
<tr>
<td>$15 \times 15$</td>
<td>$179.3896$</td>
<td>$7.1433 \times 10^{-2}$</td>
<td>$116.3357$</td>
</tr>
<tr>
<td>$16 \times 16$</td>
<td>$1143.4145$</td>
<td>$9.3467 \times 10^{-2}$</td>
<td>$143.4327$</td>
</tr>
</tbody>
</table>

In fact, problem (30) is a variant of the bottleneck AP. Hence, a proper algorithm for the bottleneck AP will improve the efficiency of (21) if $b \to \infty$ is known in advance.

In the second special case, $b$ is small; that is, the budget is insufficient for the team and none of the workers can reach the highest quality (performance) with the provided money. The computational results of the three approaches are shown in Table 3. It is clear that the proposed S&T algorithm is the most efficient. The elapsed time of the LINGO package is shorter than the value in Table 1; however, that of the index-based B&B algorithm is longer. In this case, the team performance is therefore chiefly determined by the manager and \( \min (1 - (1 - Q_{ij})x_{ij}) \) for all \((i, j) \in K\) in the objective function of (21) is redundant. Thus,

\[
\begin{align*}
\min & \left( \frac{b - \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_{ij}x_{ij})}{b - a + \sum_{i=1}^{n} \sum_{j=1}^{n} (\gamma_{ij}x_{ij})} \right), \\
\min & \left( 1 - (1 - Q_{ij})x_{ij} \right) \forall (i, j) \in K
\end{align*}
\] \quad (31)

Problem (21) can then be simplified to (22) and can be solved with Phase I and Steps A, B, C, and D of Phase II of the proposed S&T algorithm.

An overview of Tables 1–3 reveals that, among the three approaches, the S&T algorithm is the most efficient.

7. Conclusion

This study formulated a GFAP to overcome the uncertain environment in practice and developed the S&T algorithm to solve the problem. A numerical example was presented to demonstrate the solution procedure and the computational results of the S&T algorithm were compared with those of the B&B algorithm and the LINGO package. The results show that, when all aspects are considered, the S&T algorithm is the most useful and efficient. In terms of future research, the development of a sensitivity analysis procedure for modifying the membership function could be an interesting field of study with many potential applications.

References

Submit your manuscripts at http://www.hindawi.com