Group Role Assignment via a Kuhn–Munkres Algorithm-Based Solution

Haibin Zhu, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, and Rob Alkins

***Abstract*—Role assignment is a critical task in role-based collaboration. It has three steps, i.e., agent evaluation, group role assignment, and role transfer, where group role assignment is a time-consuming process. This paper clarifies the group role assignment problem (GRAP), describes a general assignment problem (GAP), converts a GRAP to a GAP, proposes an efficient algorithm based on the Kuhn–Munkres (K-M) algorithm, conducts numerical experiments, and analyzes the solutions' performances. The results show that the proposed algorithm significantly improves the algorithm based on exhaustive search. The major contributions of this paper include formally defining the GRAPs, giving a general efficient solution for them, and expanding the application scope of the K-M algorithm. This paper offers an efficient enough solution based on the K-M algorithm that outperforms significantly the exhaustive search approach.**

***Index Terms*—Algorithm, assignment problem, group role assignment, role.**

I. Introduction

ROLE-BASED collaboration (RBC) is an emerging methodology to facilitate an organizational structure, to provide orderly system behavior, and to consolidate system security for both human and nonhuman entities that collaborate and coordinate their activities with or within systems [28], [31]. The life cycle of RBC includes three major tasks: *role negotiation, assignment, and execution* [31]. Therefore, role assignment is an important aspect of RBC. It largely affects the efficiency of collaboration and the degree of satisfaction among members involved in the collaboration.

Role assignment can be categorized into three steps: *agent evaluation, group role assignment, and role transfer*. Qualifications are the basic requirements for possible role-related activities [6]. *Agent evaluation* rates the qualification of an agent for a role. It requires a check on the capabilities, experiences, and credits of agents based on role specifications. It is a fundamental yet difficult problem that requires advanced methodologies, such as information classification, data mining, pattern search, and matching. *Group role assignment* initiates a group by assigning roles to its members or agents to achieve its highest performance. *Role transfer* (also called *dynamic role assignment*) reassigns roles to agents or transfers agent roles [32], [33] to meet the requirement of the system changes.

This paper concentrates on the second part of role assignment, i.e., *group role assignment*. The assumption is that a set of agent qualification values for roles is given. This problem can be further divided into three cases: 1) agents are rated as only "yes" or "no" for roles, and all the roles have the same importance; 2) agents are rated between 0 and 1 for roles, and roles are equally important; and 3) agents are rated between 0 and 1 for roles, and roles have different importance.

The major contributions of this paper include the following:

1) formally defining the group role assignment problems (GRAPs);
2) proposing a practical solution based on the Kuhn–Munkres (K-M) algorithm;
3) expanding the application scope of the K-M algorithm.

The remainder of this paper is arranged as follows. Section II introduces the requirement of group role assignment by a real-world problem. Section III revises the Environment—Class, Agent, Role, Groups, and Object (E-CARGO) model in order to formalize the problems of group role assignment. Section IV formally defines three types of GRAPs. Section V describes a general assignment problem (GAP) and the K-M algorithm [7], [13], [17] used to solve the problem. Section VI transforms GRAPs to a GAP and proposes the relevant algorithm. Section VII presents the experimental results of the proposed algorithm. Section VIII compares the algorithms. Section IX reviews the related work, and Section X concludes this paper and indicates topics for future work.

II. Real-World Problem

To understand the problem of group role assignment, a scenario of a soccer team is considered. In a soccer team (Fig. 1), there are 20 players (a_0–a_{19}) in total. In the field, there are four roles and 11 players (in total) for the 4-3-3 formation: one goalkeeper (r_0), four backs (r_1), three midfields (r_2), and three forwards (r_3). Before each game, the most important task of the coach is to choose 11 players to be on the field. Players'

Manuscript received March 24, 2010; revised October 4, 2010 and April 3, 2011; accepted August 6, 2011. Date of publication November 18, 2011; date of current version April 13, 2012. This work was supported in part by the National Sciences and Engineering Research Council of Canada (No. 262075-06), by an IBM Eclipse Innovation Grant, by the National Natural Science Foundation of China under Grant 61034004, and by the National Basic Research Program of China under Grant 2011CB302804. This paper was recommended by Associate Editor A. Bargiela.

H. Zhu is with the Department of Computer Science and Mathematics, Nipissing University, North Bay, ON P1B 8L7, Canada (e-mail: haibinz@nipissingu.ca).

M. C. Zhou is with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China, and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

R. Alkins is with Stroma Service Inc., North Bay, ON P1A 4K3, Canada (e-mail: alkins.rob@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2011.2170414

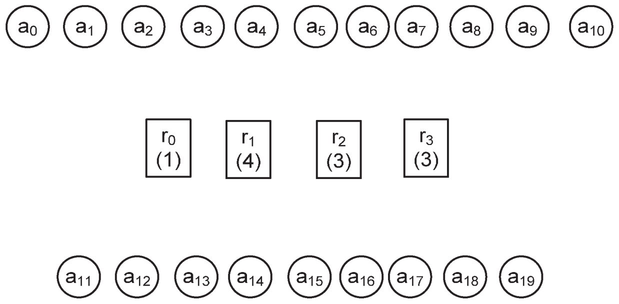
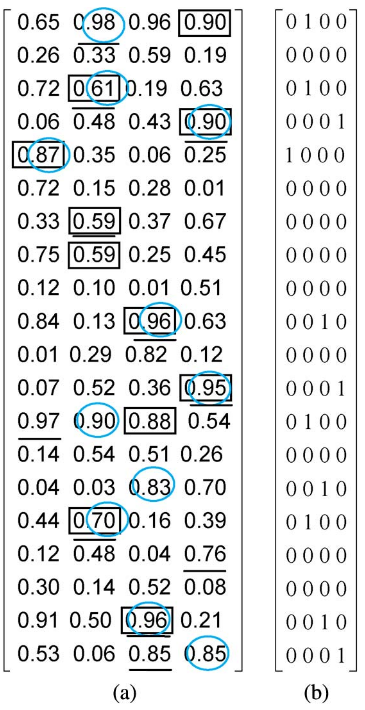

a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
r_0 (1)
r_1 (4)
r_2 (3)
r_3 (3)
a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19}

Fig. 1. Soccer team.



$$\begin{bmatrix} 0.65 & \underline{0.98} & 0.96 & \boxed{0.90} \\ 0.26 & 0.33 & 0.59 & 0.19 \\ 0.72 & \underline{\boxed{0.61}} & 0.19 & 0.63 \\ 0.06 & 0.48 & 0.43 & \underline{\boxed{0.90}} \\ \boxed{0.87} & 0.35 & 0.06 & 0.25 \\ 0.72 & 0.15 & 0.28 & 0.01 \\ 0.33 & \underline{\boxed{0.59}} & 0.37 & 0.67 \\ 0.75 & \boxed{0.59} & 0.25 & 0.45 \\ 0.12 & 0.10 & 0.01 & 0.51 \\ 0.84 & 0.13 & \underline{\boxed{0.96}} & 0.63 \\ 0.01 & 0.29 & 0.82 & 0.12 \\ 0.07 & 0.52 & 0.36 & \underline{\boxed{0.95}} \\ \underline{0.97} & 0.90 & \boxed{0.88} & 0.54 \\ 0.14 & 0.54 & 0.51 & 0.26 \\ 0.04 & 0.03 & 0.83 & 0.70 \\ 0.44 & \underline{\boxed{0.70}} & 0.16 & 0.39 \\ 0.12 & 0.48 & 0.04 & \underline{0.76} \\ 0.30 & 0.14 & 0.52 & 0.08 \\ 0.91 & 0.50 & \underline{\boxed{0.96}} & 0.21 \\ 0.53 & 0.06 & \underline{0.85} & 0.85 \end{bmatrix} \qquad \begin{bmatrix} 0&1&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1 \end{bmatrix}$$

(a) (b)

Fig. 2. Evaluation values of agents for roles and the assignment matrix.

states are affected by modes, emotions, health, fatigue, and past performance.

Suppose that the coach has the data shown in Fig. 2(a) to express the evaluation values of players with respect to each role, where rows represent players and columns represent roles. *These values reflect the individual performance of each player relevant to a specific position*. The coach needs to make role assignment to optimize the whole team's performance. *To simplify this process, the team performance is assumed as a simple sum of the selected players' performance on their designated roles*. At this step, the coach still finds that it is difficult to find an exact solution.

The coach tries the first strategy: *from* r_0 *to* r_3, *select the best players if they have not yet been selected*, i.e., the underlined numbers. The total sum is 9.23 in the first row of Table I.

The second strategy is the following: *from* r_3 *to* r_0, *select the best players if they have not yet been selected*, i.e., the numbers in the rectangles. The total sum is 8.91 in the last row of Table I.

Finally, we obtain all other permutations of the position preferences, i.e., (r_0, r_1, r_3, r_2), (r_0, r_2, r_1, r_3), ..., a total of 22 ways.

TABLE I
COMPARISONS AMONG ASSIGNMENT STRATEGIES

Strategy	**Assignment for {r₀}{r₁}{r₂}{r₃} Note: In the curly braces are the numbers of players.**	**Group Performance**
$\underline{(r_0, r_1, r_2, r_3)}$	$\underline{\{12\}\{0, 2, 6, 15\}\{9, 18, 19\}\{3, 11, 16\}}$	$\underline{9.23}$
(r_0, r_1, r_3, r_2)	{12}{0, 2, 6, 15}{9, 14, 18}{3, 11, 19}	9.30
(r_0, r_2, r_1, r_3)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_0, r_2, r_3, r_1)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_0, r_3, r_1, r_2)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_0, r_3, r_2, r_1)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_1, r_0, r_2, r_3)	{18}{0, 2, 12, 15}{9, 14, 19}{3, 11, 16}	9.35
(r_1, r_0, r_3, r_2)	{18}{0, 2, 12, 15}{9, 10, 14}{3, 11, 19}	9.41
(r_1, r_2, r_0, r_3)	{4}{0, 2, 12, 15}{9, 18, 19}{3, 11, 16}	9.44
(r_1, r_2, r_3, r_0)	{4}{0, 2, 12, 15}{9, 18, 19}{3, 11, 16}	9.44
(r_1, r_3, r_0, r_2)	{18}{0, 2, 12, 15}{9, 10, 14}{3, 11, 19}	9.41
$\boldsymbol{(r_1, r_3, r_2, r_0)}$	**{4}{0, 2, 12, 15}{9, 14, 18}{3, 11, 19}**	**9.51**
(r_2, r_0, r_1, r_3)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_2, r_0, r_3, r_1)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_2, r_1, r_0, r_3)	{4}{2, 6, 12, 15}{0, 9, 18}{3, 11, 19}	9.25
(r_2, r_1, r_3, r_0)	{4}{2, 6, 12, 15}{0, 9, 18}{3, 11, 19}	9.25
(r_2, r_3, r_0, r_2)	{12}{0, 9, 18}{1, 10, 14}{3, 11, 19}	8.79
(r_2, r_3, r_2, r_0)	{4}{0, 9, 18}{10, 12, 14}{3, 11, 19}	8.98
(r_3, r_0, r_1, r_2)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_3, r_0, r_2, r_1)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_3, r_1, r_0, r_2)	{18}{2, 6, 12, 15}{9, 14, 19}{0, 3, 11}	9.10
(r_3, r_1, r_2, r_0)	{4}{2, 6, 12, 15}{9, 18, 19}{0, 3, 11}	9.19
(r_3, r_2, r_0, r_1)	{4}{2, 6, 7, 15}{9, 12, 18}{0, 3, 11}	8.91
$\underline{(r_3, r_2, r_1, r_0)}$	$\underline{\{4\}\{2, 6, 7, 15\}\{9, 12, 18\}\{0, 3, 11\}}$	$\underline{8.91}$

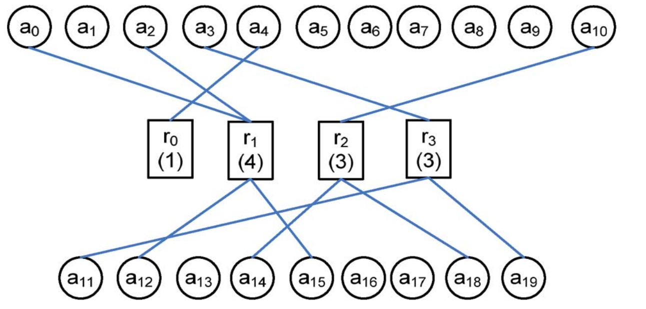

a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
r_0 (1)
r_1 (4)
r_2 (3)
r_3 (3)
a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19}

Fig. 3. Solution for Fig. 1.

Table I shows all the group performances based on 24 different permutations. Among all of the 24 strategies, *the optimum one is* (r_1, r_3, r_2, r_0), i.e., the bold row in Table I. The solution is shown in Fig. 2(a) as circles, in Fig. 2(b) as an assignment matrix, and in Fig. 3 as a graph. Please note that the solution is not intuitive and needs enumerated optimization, e.g., the most qualified player a_{12} is not assigned to the goalkeeper (r_0) because he contributes more for the team when he plays a back (r_1) than the goalkeeper. Based on this assignment, the coach chooses the players to arrange their positions in the field. This assignment obtains the best total group performance of 9.51.

III. CONDENSED E-CARGO MODEL

With the E-CARGO model [28]–[33], a tuple of agent a and role r, i.e., $\langle a, r\rangle$, is called a role assignment (also called agent assignment). A role r is workable if it is assigned enough (expressed by lower range l of range q of base $\mathcal{B}$ of role r in environment e of group g, i.e., $g.e.B[r].q.l$) current agents to play it [30]. For example, the columns in Fig. 2(b) express workable roles for the soccer team shown in Fig. 1 with a 4-3-3 formation.

In formalizing GRAPs, only agents and roles are emphasized. In the following discussions, current agents or roles are our focus, and environments and groups are simplified into vectors and matrices, respectively; $m(= |\mathcal{A}|)$ expresses the size of the agent set $\mathcal{A}$, and $n(= |\mathcal{R}|)$ expresses the size of the role set $\mathcal{R}$.

Some notations require initial clarification. If a and b are objects, $a.b$ denotes b of a, or $a's\ b.\{a, b, \ldots\}$ denotes a set of enumerated elements of a, b, and others. If a and b are integers, $[a, b]$ denotes the set of all the real numbers between a and b including both a and b, and $(a, b]$ denotes that excluding a. If Y is a vector, $Y[i]$ denotes the element at its ith position. If Y is a matrix, $Y[i, j]$ denotes the element at the intersection of row i and column j in Y.

Definition 1—Role Range Vector: A role range vector is a vector of the lower ranges of roles in environment e of group g. The role range vector is denoted as $L[j] \in \mathcal{N}$, where $\mathcal{N}$ is the set of natural numbers and $0 \le j < n$. Supposing that roles in $g.e$ are numbered as $j(0 \le j < n)$ and $\mathcal{B}[j]$ means the tuple for role j, then $L[j] = g.e.\mathcal{B}[j].q.l$.

For example, $L = [1, 4, 3, 3]$ for the soccer team in Fig. 1.

Definition 2—Qualification Matrix: The qualification matrix is an $m \times n$ matrix of values in [0, 1], where $Q[i, j]$ expresses the qualification value of agent i for role j. It is denoted as $Q[i, j] \in [0, 1] (0 \le i < m; 0 \le j < n)$.

For example, Fig. 2(a) shows a qualification matrix for the soccer team in Fig. 1.

Definition 3—Role Assignment Matrix: A role assignment matrix is an $m \times n$ matrix of values in {0, 1}. If $T[i, j] = 1$, agent i is assigned to role j, and agent i is called an *assigned agent*. It is denoted as $T[i, j] \in \{0, 1\} (0 \le i < m; 0 \le j < n)$.

For example, Fig. 2(b) shows an assignment matrix for the soccer team in Fig. 1.

Definition 4—Group Qualification: A group qualification is defined as the sum of the assigned agents' qualifications, i.e., $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j]$.

For example, the group qualification of Fig. 2 is 9.51.

Definition 5—Role Weight Vector: A role weight vector is a vector of the weights of roles in environment e of group g. The role weight vector is denoted as $W[j] \in [0, 1] (0 \le j < n)$. Supposing that roles in $g.e$ are numbered as $j(0 \le j < n)$ and $\mathcal{B}[j]$ means the tuple for role j, then $W[j] = g.e.B[j].w$.

For example, for the soccer team in Fig. 1, if we emphasize attack, we may set $W = [0.6, 0.7, 0.8, 0.9]$, but if we emphasize defense, we may set $W = [0.9, 0.8, 0.7, 0.6]$.

Definition 6—Weighted Group Qualification: A weighted group qualification is defined as the weighted sum of assigned agents' qualifications, i.e.,

$$\sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i, j] \times T[i, j].$$

For example, for the assignment in Fig. 2, if $W = [0.6, 0.7, 0.8, 0.9]$, the weighted group qualification is 7.2, and the assignment shown in Fig. 2(b) is not optimal in this case.

Definition 7—Workable Group: Group g is *workable* if all of its roles are workable, i.e., group g expressed by T is *workable* if $\forall j (\sum_{i=0}^{m-1} T[i, j] \ge [j]) (0 \le j < n)$ [28].

$$\begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&1&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&0&0&1\\ 0&1&0&0\\ 0&0&0&0\\ 0&0&1&0\\ 0&1&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{bmatrix}$$

Fig. 4. Example of a not workable group.

$$\underset{\text{(a)}}{\begin{bmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&1&0\\ 0&1&1&0\\ 0&1&1&0\\ 0&0&1&1\\ 0&0&1&1\\ 0&0&0&1\\ 0&0&0&1 \end{bmatrix}} \quad \underset{\text{(b)}}{\begin{bmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&1&0\\ 0&0&1&0\\ 0&0&0&1\\ 0&0&0&1\\ 0&0&0&1 \end{bmatrix}}$$

Fig. 5. Qualification matrix and solution.

For example, Fig. 2(b) forms a workable group for the team in Fig. 1, but Fig. 4 does not because $L = [1, 4, 3, 3]$. Note that "*not workable*" here just means that this assignment cannot implement the 4-3-3 formation but does not mean that the team cannot compete on the field.

IV. GRAPs

GRAPs can be categorized into different levels based on the different requirements: simple, rated, and weighted role assignment [30].

A. Simple Role Assignment

Definition 8—Simple GRAP (SGRAP): Let $Q[i, j] \in \{0, 1\} (0 \le i < m; 0 \le j < n)$ be the value that represents if a given agent i is able to play a given role j, where 0 and 1 mean no and yes, respectively. The roles are equally important. The problem is to find a role assignment matrix T that makes g workable [30].

For example, suppose that Q is shown as Fig. 5(a) and $L = [1, 4, 3, 3]$. One solution T is as shown in Fig. 5(b).

B. Rated Role Assignment

Assume that agents are qualified to play a role but with a rated value, i.e., a real number from 0 to 1, and the roles are equally important, where 0 means disqualified and 1 completely qualified. In this situation, group role assignment favors a solution with the maximum overall sum of qualifications for all the assigned agents.

$$\begin{bmatrix} 0.71 & 0.6 & 0.0 & 0.22 \\ 0.29 & 0.67 & 0.44 & 0.76 \\ 0.69 & 0.92 & 0.92 & 0.6 \\ 0.0 & 0.0 & 0.53 & 0.0 \\ 0.97 & 0.51 & 0.77 & 0.65 \\ 0.58 & 0.64 & 0.24 & 0.0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

(a) (b) (c)

Fig. 6. Rated qualification matrix and its rated and weighted solutions.

Definition 9—Rated GRAP (RGRAP): Let $Q[i, j] \in [0, 1] (0 \le i < m; 0 \le j < n)$ be the value that represents how well a given agent i plays a given role j, where 0 means the lowest and 1 means the highest. The roles are equally important. The problem is to find a matrix T that makes group g in which the group qualification is the maximum, i.e., $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] \times T[i, j]$ is maximized [30].

For example, suppose that Q is shown in Fig. 2(a) and $L = [1, 4, 3, 3]$. The solution T is shown in Fig. 2(b), and the group qualification is valued at 9.51.

C. Weighted Role Assignment

The rated role assignment can be extended. If the roles in a group have different importance, i.e., the roles have different weights, a weighted role assignment problem occurs.

Definition 10—Weighted GRAP (WGRAP): Let $Q[i, j] \in [0, 1] (0 \le i < m; 0 \le j < n)$ be the value that represents how well a given agent i plays a given role j. The problem is to find a matrix T that creates a group g in which the *weighted group qualification* is the largest, i.e., $\sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i, j] \times T[i, j]$ is maximized [30].

For example, suppose that Q is shown in Fig. 6(a), $L = [2, 1, 1, 2]$, and $W = [0.19, 0.62, 0.42, 0.66]$. A solution is shown in Fig. 6(c). Evidently, this assignment is different from that shown in Fig. 6(b), which is for rated assignment.

V. GAP AND THE K-M ALGORITHM

GRAPs are complicated. The algorithms based on exhaustive search are of exponential complexity [30]. Therefore, it must be improved and made practical. Fortunately, the well-known K-M algorithm for the GAP has been designed with the complexity of $O(m^3)$[7], [13], [17] and has been applied widely in industries [20], and its Java code is also available [4]. If a GRAP is transferred to a GAP, then the GRAP will be solved efficiently. Although there are other algorithms [7], [23] that are better than the K-M algorithm in efficiency, we choose the K-M algorithm in this paper since its solved problem is similar to the GRAP. It enjoys higher reputation, popularity, understandability, and code availability with known efficiency.

A GAP [4], [7], [9], [13], [17] can be described as follows: given $n \times n$ matrix A of real numbers, find a permutation p (p_i, $i = 0, 1, 2, \ldots, n-1$) of integers that minimizes $\sum_{i=0}^{n-1} A[i, p_i]$. For example, for a 3×3 matrix

$$A = \begin{bmatrix} 7 & 5 & 11.2 \\ 5 & 4 & 1 \\ 9.3 & 3 & 2 \end{bmatrix},$$

there are six possible permutations for which the associated sums are shown in Table II.

TABLE II
PERMUTATIONS AND THE SUMS FOR MATRIX A

	p	$\sum_{i=0}^{n-1} A[i, p_i]$
(1)	0, 1, 2	13.0
(2)	**0, 2, 1**	**11.0**
(3)	1, 0, 2	12.0
(4)	1, 2, 0	15.3
(5)	2, 0, 1	19.2
(6)	2, 1, 0	24.5

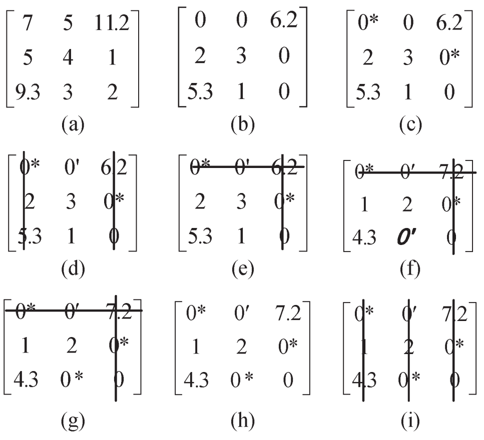


Fig. 7. Example processed by the K-M algorithm.

Permutation (2) gives the smallest sum of 11.0 and is the solution to the assignment problem for this matrix.

Apparently, the RGRAP can be transferred to a GAP by adjusting the number of agents and roles. If $m = \sum_{j=0}^{n-1} L[j]$ in Definition 1, the problem becomes a GAP.

The K-M algorithm is a minimization algorithm for square matrices of GAPs invented by Kuhn [13] and improved by Munkres [17]. To understand it, a definition and a theorem are required.

Definition 11—Independent Set: A set of elements of a matrix is *independent* if none of them occupies the same row or column.

König Theorem: If M is a matrix and z is the number of independent zero elements of M, then there are z lines that contain all the zero elements of M.

The K-M algorithm starts by finding the smallest element in each row. This element is then subtracted from each row. This process is also done for each column. Then, it puts tags to all zeros called as starred or primed zeros. The "starred zeros" form an independent set of zeros, while the "primed zeros" are possible candidates. The algorithm also distinguishes lines (rows or columns): covered and uncovered. A zero is called covered (noncovered) if it is in a covered (noncovered) row or column.

To start, the following conditions should be established: 1) no lines are covered; no zeros are starred or primed; 2) for each row of matrix M, subtract the value of the smallest element from each element in the row; and 3) for each column of the resulting matrix, subtract the value of the smallest element from each element. The K-M algorithm is as follows:

Input:
∘*An $m \times m$ matrix $M$with the above preliminaries.*
Output:
∘*All the subscripts of the selected elements of M, i.e., the starred zeros.*

```
L1: K-M(M):
L2: { for (all zeros $z$ in $M$)
L3: If (there are no starred zeros in the row or column of $z$)
L4: star $z$;
L5: while (true) // the main loop
L6: { cover all the columns containing a $0^*$;
L7: if (all the columns are covered)
L8: return; // the optimal solution is starred;
L9: while (!(all zeros are covered)) //the inner loop
L10: { for (all noncovered zeros $z$)
L11: { prime $z$;
L12: if (there is a starred zero $z^*$ in this row)
L13: { cover row of $z^*$;
L14: uncover column of $z^*$;
L15: }
L16: else
L17: { highlight $z$;
L18: while (there is a starred zero $z^*$ in $z$'s column)
L19: { $z = z^*$;
L20: highlight $z$;
L21: $z :=$ the primed zero in $z$'s row; // This always exists.
L22: highlight $z$;
L23: }// end of while
L24: unstar highlighted starred zeros;
L25: star highlighted primed zeros and remove highlights;
L26: remove primes and uncover all rows and columns;
L27: continue to while (the main loop);
L28: }// end of if
L29: }//end of for
L30: $h :=$ smallest uncovered element in $M$;
L31: add $h$ to each covered row;
L32: subtract $h$ from each uncovered column;
L33: }//end of while—the inner loop
L34: }//end of while—the main loop
L35: }// end of K-M
```

For example, we process the matrix in Fig. 7(a). By initialization, we obtain a matrix in Fig. 7(b). With steps L2–L4, we star the zeros [Fig. 7(c)] From L5–L8, we get Fig. 7(d). From L9–L15, we get Fig. 7(e), and then by step L30, we get $h = 1$. By L31–L32 and back to L9–L11 and L17, we get Fig. 7(f). From L24–L25, we get Fig. 7(g). Continuing to L26, we get Fig. 7(h). Through L27 and back to L5–L6, we get Fig. 7(i). Then, with L7–L8, we complete the process. Finally, we obtain the assignment result as [0, 0], [1, 2], and [2, 1]. The minimum sum of the problem is $7 + 1 + 3 = 11$.

VI. Solutions to GRAPs

To apply the K-M algorithm into solving the GRAPs, many aspects should be considered. Note that the algorithm is always successful because it finds a smallest sum or a largest sum

$$\begin{bmatrix} 0.9 & 0.8 & 0.2 \\ 0.6 & 0.8 & 0.3 \\ 0.3 & 0.3 & 0.0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

(a) (b) (c)

Fig. 8. Matrix that the K-M algorithm may obtain a wrong result.

$$\begin{bmatrix} 0.71 & -36 & -36 & -36 \\ -36 & 0.67 & -36 & 0.76 \\ 0.69 & 0.92 & 0.92 & 0.62 \\ -36 & -36 & -36 & -36 \\ 0.97 & -36 & 0.77 & 0.65 \\ -36 & 0.64 & -36 & -36 \end{bmatrix}$$

Fig. 9. Qualification matrix for a group with no solution when $L = [1, 1, 3, 1]$.

by adaptation. However, in the GRAPs, the result might not be a solution for a group even when the group qualification is optimal. Typically, $Q[i, j] = 0$ means that agent i is not qualified for role j. The improved algorithm for the rated group assignment problem [30] follows the same assumption. To generalize the rated problem, we need a new definition.

Definition 12—Qualification Threshold (τ): τ is a real number in [0, 1] to state that an agent is called qualified in a group only if its qualification value is greater than τ.

With the introduction of τ, an agent may be unqualified for a role if its qualification value is not greater than τ, i.e., the agent should not be assigned to the role. That is to say that, by the K-M algorithm, we may find an optimal result but not a workable assignment matrix. For example, in Fig. 8(a), where $L = [1, 1, 1]$ and $\tau = 0.0$, the K-M obtains the result shown in Fig. 8(b), which is not a successful assignment matrix. Fig. 8(c) is the correct one.

To avoid the result of Fig. 8(b), it is required to check whether the group is workable after obtaining an assignment based on the K-M algorithm:

- For each role, there should be enough assigned agents, i.e., $\sum_{i=0}^{m-1} T[i, j] \times (\lceil Q[i, j] - \tau \rceil) \geq L[j]$, where $0 \leq j < n$.

To find a solution for the GRAP [e.g., Fig. 8(c)], we need to recall the K-M algorithm by adjusting matrix Q to avoid the incorrect optimal result. It is observed that the elements of a qualification matrix are in [0, 1]. The sum of all the elements never exceeds m^2, even considering the square matrix created from Q. Therefore, $\sigma = -m^2$ can be taken as the special value to replace the element (not greater than τ) in matrix Q (Fig. 9). This number protects the K-M algorithm from obtaining an incorrect optimal assignment because the group qualification will be negative if such a value is selected. If the group qualification from the optimal result is negative, the group has definitely no successful assignment.

With τ, the elements with the value of $-m^2$ in Q should not be considered. If there are not enough qualified agents to play a special role in a group, there is no solution for the group. In Fig. 9, if $\tau = 0.6$ and $L = [1, 1, 3, 1]$, no solution can be found. Therefore, it is required to assure if the assignment is feasible before calling the K-M algorithm.

Condition 1: The total agent number should be larger than the required numbers, i.e., $m \geq \sum_{j=0}^{n-1} L[j]$.

$$\begin{bmatrix} 0.71 & 0.71 & -36 & -36 & -36 & -36 \\ -36 & -36 & 0.67 & -36 & 0.76 & 0.76 \\ 0.69 & 0.69 & 0.92 & 0.92 & -36 & -36 \\ -36 & -36 & -36 & 0.53 & -36 & -36 \\ 0.97 & 0.97 & -36 & 0.77 & 0.65 & 0.65 \\ -36 & -36 & 0.64 & -36 & -36 & -36 \end{bmatrix}$$

Fig. 10. Created square matrix.

Condition 2: For each role, there should be enough qualified agents, i.e., $\sum_{i=0}^{m-1} \lceil Q[i,j] - \tau \rceil \geq L[j]$, where $0 \leq j < n$.

To convert a GRAP to a GAP, a square matrix must be created. To form the square matrix used in the K-M algorithm, duplicate columns should be added based on the role range vector L.

Definition 13—Role Index Vector L': A role index vector is an $m(\geq \sum_{j=0}^{n-1} L[j])$ dimensional vector created from a role range vector L. $L'[k]$ is a role number related with column $k(0 \leq k < m)$ in the adjusted qualification matrix Q' defined in **Definition 14**, where

$$L'[k] = \begin{cases} 0 & (k < L[0]) \\ x & \left(\sum_{p=0}^{x-1} L[p] \leq k < \sum_{p=0}^{x} L[p] (0 < x < n)\right) \\ n & \left(k \geq \sum_{p=0}^{n-1} L[p]\right) \quad (0 \leq k < m). \end{cases}$$

Definition 14—Adjusted Qualification Matrix Q': Q' is an $m \times m$ matrix, where $Q'[i,j] \in [0,1]$ expresses the qualification value of agent i for role $L'[j]$, where

$$Q'[i,j] = \begin{cases} Q\left[i, L'[j]\right] & \left(0 \leq i < m; 0 \leq j < \sum_{p=0}^{n-1} L[p]\right) \\ 1 & \left(0 \leq i < m; \sum_{p=0}^{n-1} L[p] \leq j < m\right). \end{cases}$$

From **Definitions 13–14**, Q and L can be combined to create a square matrix Q' used in the GAP. For example, Q shown in Fig. 9 and $L = [2,1,1,2]$ can create Q' in Fig. 10. It is needed to record the column numbers in a new vector $L' = [0,0,1,2,3,3]$.

If $m > \sum_{j=0}^{n-1} L[j]$, it is needed to add $m - \sum_{j=0}^{n-1} L[j]$ columns of 1's to make a square matrix, i.e., every agent is fully qualified for an empty role. This assumption follows the logic implication: $A \rightarrow B$ (A implies B) is true if A is false. Here, we can replace A with "*there is a role*" and B with "*one is qualified to play it*." That is why we define $Q'[i,j] = 1$ $(0 \leq i < m, \sum_{j=0}^{n-1} L[j] < j < m)$.

For example, for $Q(L = [1,2,1])$ in Fig. 11, its square matrix Q' is shown in Fig. 12. If $\tau = 0.6$, the relevant column number vector $L' = [0,1,1,2,3,3]$, where 3 corresponds to empty roles.

Because the K-M algorithm solves a minimization problem, it is necessary to transform it to a maximization one. This is simply a matter of subtracting the entries of Q' from the largest entry of Q' (i.e., 1) to obtain a new matrix M. Minimization of this new matrix results in a maximization of Q'.

$$\begin{bmatrix} 0.36 & 0.76 & 0.72 \\ 0.93 & 0.59 & 0.24 \\ 0.06 & 0.46 & 0.69 \\ 0.40 & 0.10 & 0.74 \\ 0.23 & 0.75 & 0.24 \\ 0.21 & 0.77 & 0.24 \end{bmatrix}$$

Fig. 11. Matrix with $m > \sum_{j=0}^{n-1} L[j]$.

$$\begin{bmatrix} -36 & 0.76 & 0.76 & 0.72 & 1.0 & 1.0 \\ 0.93 & -36 & -36 & -36 & 1.0 & 1.0 \\ -36 & -36 & -36 & 0.69 & 1.0 & 1.0 \\ -36 & -36 & -36 & 0.74 & 1.0 & 1.0 \\ -36 & 0.75 & 0.75 & -36 & 1.0 & 1.0 \\ -36 & 0.77 & 0.77 & -36 & 1.0 & 1.0 \end{bmatrix}$$

Fig. 12. Square matrix transferred from the matrix in Fig. 11.

$$\begin{bmatrix} 37 & 0.24 & 0.24 & 0.28 & 0.0 & 0.0 \\ 0.07 & 37 & 37 & 37 & 0.0 & 0.0 \\ 37 & 37 & 37 & 0.31 & 0.0 & 0.0 \\ 37 & 37 & 37 & 0.26 & 0.0 & 0.0 \\ 37 & 0.25 & 0.25 & 37 & 0.0 & 0.0 \\ 37 & 0.23 & 0.23 & 37 & 0.0 & 0.0 \end{bmatrix}$$

Fig. 13. Square matrix transferred from the qualification matrix in Fig. 12.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Fig. 14. Assignment matrix T for the group in Fig. 11.

For example, for the maximization of Q' shown in Fig. 12, a matrix M (Fig. 13) can be obtained by such assignment:

$$M[i,j] = 1 - Q'[i,j], \ (0 \leq i,j < m).$$

Now, it is ready for calling the K-M algorithm with matrix M. The solution can be obtained as shown in Fig. 14, and the group qualification is 3.2.

To summarize, an algorithm is obtained to transfer an $m \times n$ qualification matrix to an $m \times m$ square matrix and so is a theorem.

```
Input: $m$, $n$, an $m \times n$ matrix $Q$, an $n$ vector $L$, and a qualification threshold $\tau$;
Output: an $m \times m$ square $M$.
Transfer$(\mathbf{Q}, \mathbf{L}, \mathbf{m}, \mathbf{n}, \boldsymbol{\tau}, \mathbf{M})$
{$i := 0$;
while $(i < m)$
{$j := 0$; $k := 0$;
for $(0 \leq i < m, 0 \leq j < n)$
          if $(Q[i,j] > \tau) Q'[i,j] := Q[i,j]$;
          else $Q'[i,j] := -m \times m$;
while $(j < n)$
{ if $(L[j] == 1)$ then $M'[i,k] := Q'[i,k]$; $L'[k] = j$; $k := k + 1$;
    else if $(L[j] > 1)$ then { $x := L[j]$;
```

```
            for $(0 \le g < x)\{M'[i,g] := Q'[i,j];$
    $L'[k+g] := j; k := k+x;$
            }
        }
        $j := j+1;$
    }
    if $(k < m)$ then for $(0 \le h < m-k)\{M'[i,k+h] := 1;\};$
    $i := i+1;$
    }
    for $(0 \le i < m, 0 \le j < m)$ $M[i,j] := 1 - M'[i,j];$
}
```

Theorem 1: An RGRAP can be transferred to a GAP.

Proof: To prove the theorem, we only need to prove that the result subscript vector V from the K-M algorithm can be used to create the result matrix T required by the RGRAP.

From the $m \times m$ matrix M, we obtain an n-dimensional subscript vector V by the K-M algorithm to make $\sum_{i=0}^{m-1} M[i,V[i]]$ the minimum, i.e., $\sum_{i=0}^{m-1} M[i,V[i]] \ge (\sum_{i=0}^{m-1} M[i,j](j \ne V[i]))$

$$\because \sum_{i=0}^{m-1} M[i,V[i]] = \sum_{i=0}^{m-1}(1 - M'[i,V[i]]) = m - \sum_{i=0}^{m-1} M'[i,V[i]]$$

$$\therefore \sum_{i=0}^{m-1} M'[i,V[i]] = m - \sum_{i=0}^{m-1} M[i,V[i]] = \sum_{i=0}^{m-1}(1 - M[i,V[i]]).$$

$$\because \sum_{i=0}^{m-1} M[i,V[i]] \le \left(\sum_{i=0}^{m-1} M[i,j]\,(j \ne V[i])\right)$$

$$\therefore -\sum_{i=0}^{m-1} M[i,V[i]] \ge \left(-\sum_{i=0}^{m-1} M[i,j]\,(j \ne V[i])\right)$$

$$\therefore m - \sum_{i=0}^{m-1} M[i,V[i]] \ge \left(m - \sum_{i=0}^{m-1} M[i,j]\,(j \ne V[i])\right).$$

$$\because m - \sum_{i=0}^{m-1} M[i,V[i]] = \sum_{i=0}^{m-1}(1 - M[i,V[i]]) = \sum_{i=0}^{m-1} M'[i,V[i]]$$

$$m - \sum_{i=0}^{m-1} M[i,j] = \sum_{i=0}^{m-1}(1 - M[i,j]) = \sum_{i=0}^{m-1} M'[i,j]$$

$$\therefore \sum_{i=0}^{m-1} M'[i,V[i]] \ge \sum_{i=0}^{m-1} M'[i,j]\,(j \ne V[i]),\ \text{i.e.,}$$

$$\sum_{i=0}^{m-1} M'[i,V[i]]\ \text{is the maximum sum for } M'.$$

Corresponding to the RGRAP and the transfer algorithm, V can be used to create T, i.e.,

$$\text{for}(0 \le i < m, 0 \le j < n) T[i,j] := 0$$
$$\text{for}(0 \le i < m, 0 \le j < n) T[i, L'[V[i]]] := 1.$$

Therefore, if $T[i,j] = 1$, then $Q'[i,j] \times T[i,j] = M'[i,V[i]]$, i.e.,

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T[i,j] \times Q'[i,j] = \sum_{i=0}^{m-1} M'[i,V[i]].$$

Therefore, $\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T[i,j] \times Q'[i,j]$ is the maximum sum as for Q'. Furthermore

$$\because Q[i,j] \ge Q'[i,j] (0 \le i < m, 0 \le j < n)$$
$$\therefore \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T[i,j] \times Q[i,j].$$

As a result, T is the matrix required by the RGRAP. ■

The algorithm that is used to solve an RGRAP can be described as follows:

Input:
○*An n-dimensional role's lower range vector L; and*
○*An $m \times n$ rated qualification matrix Q.*

Output:
○**Success:** An $m \times n$ *assignment matrix T in which for all columns $j (j = 0, \ldots, n-1)$, $\sum_{i=0}^{m-1} T[i,j] \ge L[j]$, and v as the maximum group qualification.*
○**Failure:** *An $m \times n$ assignment matrix T in which there is at least one column j $(j = 0, \ldots, n-1)$, $\sum_{i=0}^{m-1} T[i,j] < L[j]$, and v as* 0.

```
RatedAssign (L, Q, T, m, n, $\tau$)
{
Step 1: If Conditions 1 and 2 are not satisfied, return Failure;
Step 2: Transfer($Q, L, m, n, \tau, M$);//Call the Transfer algorithm;
Step 3: K-M (M); //Call the K-M algorithm.
Step 4: Form the assignment matrix $T$ based on the result of K-M (M);
Step 5: If $T$ is a successful assignment return Success else Failure;
}
```

As for the complexity, beside the K-M algorithm, steps 1, 2, 4, and 5 have the complexity of $O(m^2)$, which is less than the complexity of the K-M algorithm, i.e., $O(m^3)$[13], [17], [20], [23]. Therefore, the complexity of the aforementioned algorithm is $O(m^3)$.

A WGRAP can be solved by the algorithm for an RGRAP. We have the following result.

Theorem 2: A WGRAP can be solved by the algorithm for an RGRAP.

Proof: To prove this theorem, we just need to construct Q_1 from Q and use Q_1 to obtain T that makes $\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q_1[i,j] \times T[i,j]$ the maximum by the algorithm for an RGRAP and then prove that T also makes $\sum_{j=0}^{n-1}\{W[j] \times \sum_{i=0}^{m-1}(Q[i,j] \times T[i,j])\}$ the maximum.

Supposing that we have a qualification matrix Q and a weight vector W for the WGRAP, we obtain a new matrix Q_1 by $Q_1[i,j] := W[j] \times Q[i,j]$.

Supposing that T is the assignment matrix obtained by the algorithm for the RGRAP based on Q_1, then $\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q_1[i,j] \times T[i,j]$ is the maximum, i.e.,

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q_1[i,j] \times T[i,j] \geq \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q_1[i,j] \times T_1[i,j] (T \neq T_1).$$

Replacing Q_1 with $W[j] \times Q[i,j]$, we have

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W[j] \times Q[i,j] \times T[i,j]$$

$$\geq \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W[j] \times Q[i,j] \times T_1[i,j] (T \neq T_1)$$

$$\because \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} W[j] \times Q[i,j] \times T[i,j]$$

$$= \sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i,j] \times T[i,j]$$

$$\therefore \sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i,j] \times T[i,j]$$

$$\geq \sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i,j] \times T_1[i,j] (T \neq T_1)$$

$$\therefore T \text{ makes } \sum_{j=0}^{n-1} W[j] \times \sum_{i=0}^{m-1} Q[i,j] \times T[i,j] \text{ the maximum.}$$

From Definition 10, we have solved the WGRAP, and the solution is T. ■

Because the RGRAP can be generalized by introducing a threshold, the SGRAP is, in fact, a special case of the RGRAP, i.e., $\tau = 1$, and the group threshold definition τ is modified from "larger than" to "not less than."

VII. Implementation and Performance Experiments

The K-M algorithm, when properly implemented, can operate with the computational complexity of $O(m^3)$[13], [17], [20], [23]. This is better than the exhaustive-search-based algorithm [30].

To verify the performance of the algorithm for RGRAPs, a program is implemented based on that of the K-M algorithm [4]. The hardware platform is a laptop with a CPU of 2.10 GHz and a main memory of 4 GB. The development environment is Microsoft Windows Vista (Home Edition) and Eclipse 3.2.

Ten experiments are designed for RGRAPs in different group sizes (Table III), where the workable group rate tells the percentage of the workable groups among 300 random ones in one experiment. That almost all the cases have 100% of workable groups tells a fact that it is very easy to obtain an assignment solution with m agents for $m/2$ roles when $m \geq 20$, where one role may require one to two agents and $\tau = 0.6$.

TABLE III
Times for the Rated Assignment Algorithm($\tau = 0.6$)

Time / m	Largest (ms)	Smallest (ms)	Average (ms)	Workable Group Rate
10	72.211692	0.003771	0.466843	0.88
20	14.090758	0.100641	0.564071	1
30	32.893077	0.337752	1.457006	1
40	34.022271	0.53212	2.563242	1
50	34.213217	1.573385	3.95924	1
60	48.69348	2.001511	6.833295	1
70	50.828737	3.455048	9.816073	1
80	54.279105	6.857017	16.924448	1
90	74.972518	9.521182	22.492996	1
100	88.991827	11.355284	29.055012	1

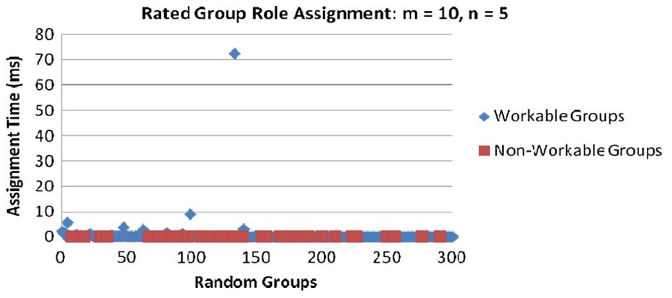


Fig. 15. Distribution of assignment times for rated group role assignment ($\tau = 0.6$): $m = 10$ and $n = 5$.

In an experiment, a group is formed by randomly creating *an agent qualification matrix*: each agent is assigned randomly with values (0, 1] for n roles, i.e., each agent is somewhat qualified to play a role. The number of roles n is set as $m/2$ without loss of generality because the complexity of the K-M algorithm is mainly dependent on m. The lower ranges L of roles are randomly with 1 or 2 to make the corresponding group have enough agents $(\sum_{j=0}^{n-1} L[j] \leq m)$.

Each experiment repeats for 300 randomly created agent qualification matrices to show its generality. The results of the experiments for $\tau = 0.6$ are shown in Table III, and the distribution of assignment times for $m = 10$ and $n = 5$ is shown in Fig. 15.

To check the performance of SGRAPs, new experiments are required. For an SGRAP, a group is formed by randomly creating *an agent qualification matrix* with 0s and 1s. Each agent is assigned randomly with 0 or 1 for n roles. The number of roles n is set as $m/2$. L is set with 1 or 2 to make the corresponding group have enough agents $(\sum_{j=0}^{n-1} L[j] \leq m)$. Each experiment repeats for 300 random groups to show its generality. The results of the experiments are shown in Table IV, and the distribution of assignment times for $m = 10$ and $n = 5$ is shown in Fig. 16. Figs. 15 and 16 just show the distributions of times used by the 300 random groups when $m = 10$ and $n = 5$. Other cases have similar distribution patterns.

TABLE IV
TIMES FOR SIMPLE ASSIGNMENT ALGORITHM($\tau = 1.0$)

Time / *m*	Largest (ms)	Smallest (ms)	Average (ms)	Workable Group Rate
10	5.226223	0.004121	0.094552	0.96
20	3.979277	0.053568	0.111339	1
30	20.111844	0.103226	0.239936	1
40	7.463588	0.172647	0.296842	1
50	36.016519	0.265746	0.563867	1
60	16.39112	0.372534	0.693468	1
70	38.678728	0.515428	1.199225	1
80	102.615556	0.675575	1.315682	1
90	37.040113	0.861073	1.738368	1
100	37.74977	1.030438	1.373509	1

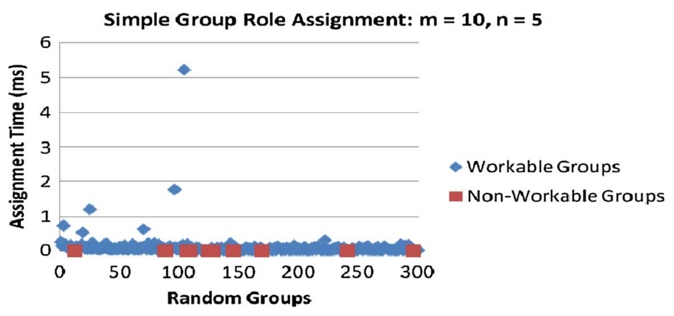


Fig. 16. Distribution of assignment times for simple group role assignment ($\tau = 1.0$): $m = 10$ and $n = 5$.

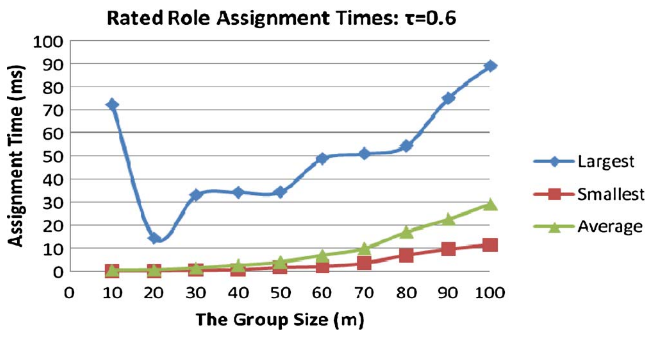


Fig. 17. Trend lines for typical times for different sizes of the groups for rated group role assignment.

VIII. PERFORMANCE ANALYSIS

Tables III and IV show the typical data collected from the experiments stated previously. The largest times required by random groups are not linearly increased because each experiment independently tests 300 random groups. It is possible for some particular smaller size group to take more time than a larger size group because the value distributions significantly affect the time needed in the relevant algorithm. The fact that the largest times are oscillated also demonstrates the importance of members' qualification distributions in a group and that heuristic algorithms may exist.

The smallest and average times used by the rated assignment algorithm increase slowly (slower than $O(m^3)$), as shown in Fig. 17. This situation occurs because the square matrix used by the K-M algorithm is created from random groups. The random groups may create many columns fully with 1s. Therefore, the time used by the algorithm is normally less than $O(m^3)$.

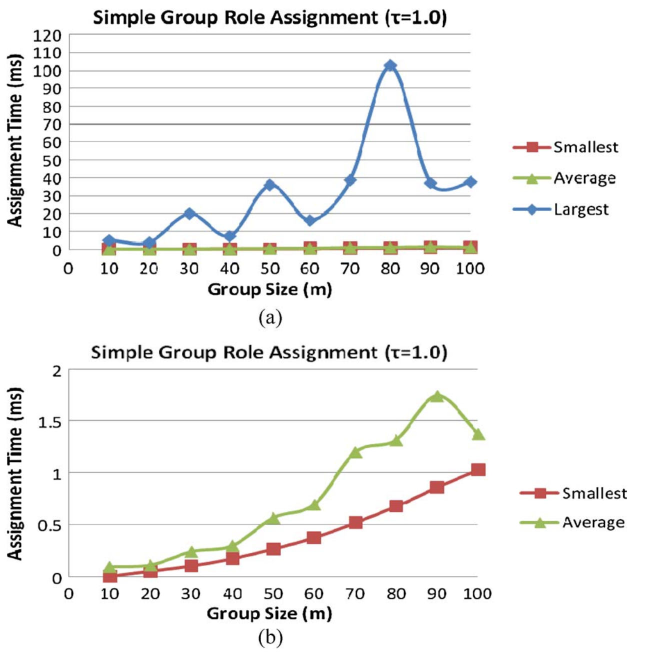


Fig. 18. Trend for typical assignment times according to group sizes.

Fig. 18 is separated into Fig. 18(a) and Fig. 18(b) to show the difference between average and smallest times for the simple group role assignment. The largest times have properties similar to those of RGRAPs. From Figs. 17 and 18 and Tables III and IV, the proposed algorithm for the GRAPs is much better than the algorithm in [30] and can be applied to large groups.

Please note that it is not practical to compare directly the data of the proposed algorithm with that of the exhaustive search algorithm that has an exponential complexity. The latter takes hours and even days after the agent number m exceeds 15. The estimation is as follows. The time used when $m = 10$ is 1.5 s [30], and then, the time required to solve a problem with $m = 30$ is roughly $2^{30}/2^{10} \approx 2^{20} \times 1.5$ s ≈ 18 days. Hence, it is difficult to show the data of the latter algorithm in Figs. 17 and 18.

IX. CASE STUDY BY SIMULATION

A training school hopes to hire one instructor for each section in its summer school of software engineering that offers multicourses and multisections. The administrators hope to optimize the total performance of the summer school. They publish the call for instructors on a Web site and ask applicants to fill in an application form to indicate their requested salary (1000–5500), years of teaching experience (1–20), and desired time segment for each course (1—morning, 2—afternoon, and 3—evening). They want to use the least money to hire the most experienced instructors (one instructor can teach only one course to avoid overload) and, at the same time, choose the instructors who would like to teach at the latest time segment in a day. It is a complex three-objective optimization problem with m applicants and n courses. Note that, if an instructor is allowed to teach multisections, the problem becomes more complex such that the proposed algorithm cannot be applied directly.

Such multiobjective optimization problems are NP-hard [11]. Based on a similar estimation to that in Section VIII, it becomes impractical as the number of applicants increases. Therefore, we need to use a heuristic algorithm to obtain a near-optimal solution. Without loss of generality, a simulation is conducted by randomly creating 20 applicants for 5 courses (a total of ten sections, $L = [2, 1, 2, 3, 2]$). Then, we use a typical weighted sum method [16] by normalizing the three factors.

The time used by the proposed algorithm is 24 ms. It is needed to mention that another simulation for $m = 30 (n = 5)$ takes only 40 ms.

The indication of closeness for salary is 82.61%, for years 75.27%, and for time segment 88%.

Please note that 100% of closeness is impossible because different functions for optimizations may be in conflict. There must be tradeoffs in the final assignment. Therefore, the proposed approach is practical.

X. Related Work

Although role assignment is important in fields such as management [8], organizational behavior and performance [3], [6], [8], scheduling, training, and commanding [23], there is no fundamental research theory and algorithms. Some related researches concern agent systems [10], [19], [21], [24]–[26], Web services [18], access control [1], [2], and sensor networks [5] concentrating on their special aspects.

Al-Kahtani and Sandhu propose implicit user-role assignment by introducing role hierarchies [2] in role-based access control (RBAC), i.e., rule-based (RB)-RBAC model. With rules, implicit user-role assignment is supported, i.e., no human intervention is needed, and RB-RBAC automatically triggers authorization rules to assign users to roles.

Dastani *et al.* [10] determine the conditions under which an agent can enact a role and what it means for the agent to enact a role. They concerned about the norms of how individual agents play individual roles. Their work, in fact, proposes a framework to solve part of *agent evaluation*, as discussed in Section I.

Ng *et al.* [18] discuss the service assignment problems in Web service applications. They assume that different service providers have different profits and commission rates that affect the service assignment. Their work demonstrates that service assignment is a potential application area for the algorithm discussed in this paper.

Odell *et al.* [19] point out that the roles played by an agent may change over time. They apply dynamic classification to deal with adding additional roles or removing roles beyond the minimum of one. They consider the robots' bids for a role and conduct role assignment in a fixed total order. Their contribution focuses on the third step of role assignment, i.e., role transfer.

Stone and Veloso [21] introduce periodic team synchronization domains as time-critical environments in which agents act autonomously. They state that dynamic role adjustment (the third step, i.e., role transfer [32], [33]) allows needed changes for a group of agents to collaborate. They use dynamic role assignments to change formation of a robot soccer team based on the game situation, such as winning or losing.

Vail and Veloso [24] extend the work [22] in role assignment and coordination in multirobot systems, especially in highly dynamic tasks. They develop an approach to sharing sensed information and effective coordination through the introduction of shared potential fields. The potential fields are based on the positions (roles) of the other robots in the team and ball. The robots position themselves in the field by following the gradient to a minimum of the potential field.

Wang *et al.* [25] propose to apply minority game strategies for dynamic role assignment in agent teams. They use soccer robots as an experiment platform and validate their proposed method. They mainly consider the first step, i.e., agent evaluation, and the third step, i.e., role transfer.

Xu *et al.* [26] build a prototype of Role-based Agent Development Environment (RADE). In RADE, dynamic role assignment (the third step, i.e., role transfer) is emphasized. The dynamic process of role assignment is formalized as agent role mapping (A-R mapping).

Zhang [27] proposes a teamwork language Role-Based Multi-Agent Logic Language for Encoding Teamwork (RoB-MALLET) to support multiagent collaboration. She designs rules and related algorithms to regulate the selection of roles and the assignment of roles to agents. Her major job is to evaluate agents' qualifications for roles (the first step).

The aforementioned research indicates a strong need to fundamentally investigate GRAPs and their solutions. Al-Kahtani and Sandhu's work [2] suggests the future work for group role assignments. Also, the algorithm proposed in this paper can be applied to solve service assignment problems [18] and can be integrated with Xu *et al.*'s framework [26] and Zhang's RoB-MALLET [27].

We compare some related work with our proposed algorithms from scope, goal, and efficiency (Table V).

XI. Conclusion

Group role assignment is an important problem in RBC. Efficient algorithms are required for practical applications. This paper has contributed an efficient way to solve this problem. Because of its polynomial complexity, it is highly practical for the role assignment of large groups [12], [15]. Because this solution is established on the evaluation of agents, it reveals the importance of agent evaluation. Possibly, with this proposed solution, the complexity (being NP-hard) of constrained assignment problems [14] can be improved by pertinent domain-oriented agent evaluations.

Further investigations can be conducted along the following directions.

1) GRAPs can be further generalized or specialized. For example, there might be a number of restrictions and constraints [14] to assignment: some conflicting agents cannot be assigned to a certain role at the same time [1]; certain roles must be filled, while other roles are optional; role hierarchies need to be incorporated [2]; and temporal requirements for roles may need to be considered.

TABLE V
COMPARISONS BETWEEN RELATED WORK AND THIS PAPER (NOTE: m IS THE NUMBER OF AGENTS IN A GROUP)

Methods	Scope	Goal	Algorithm	Efficiency
Al-Kahtani and Sandhu [2]	RBAC	To control access to differed resources by assignment roles.	Not Available	Not Applicable
Bertsekas [7]	Assignment	To improve the K-M algorithm.	Yes	O(m^3)
Dastani et al. [10]	Agent collaboration	To verify if an agent is able to play a role.	Not Available	Not Applicable
K-M [13, 17]	Assignment	To minimize the sum of the selected values in a matrix.	Yes	O(m^3)
Ng *et al.* [18]	Service composition	To obtain an optimized service composition within a special scenario based on some constraints.	Not Available	Not Applicable
Odell *et al.* [19]	Agent collaboration	To demonstrate the requirement of dynamic role assignment.	Not Available	Not Applicable
Stone and Veloso [21], Vail and Veloso [24], Wang *et al.* [25]	Robot collaboration	To make a robot soccer team to win a game. To concentrate on policies for robot collaborations.	Not Available	Not Applicable
Toroslu [22]	Incremental Assignment	To solve a special assignment problem, i.e., incremental assignment.	Yes	O(m^2)
Xu *et al.* [26], Zhang[27]	Agent collaboration	To describe agent collaboration.	Not Available	Not Applicable
This work	General Collaboration	To optimize group performance by applying the K-M algorithm.	Yes	O(m^3)

2) Agent evaluation should be investigated based on domain requirements. The constraints mentioned in 1) may be transferred into a combined evaluation value when applying the proposed methodology.
3) Role assignment algorithms must be able to adapt to the changing requirements and to provide continuous solutions that minimize task switching among agents.
4) The optimal scheduling of agents who perform tasks can be considered. The maximization of group utility must be subjected to availability and maximum workload restrictions.
5) It is valuable to investigate the use of Bertsekas algorithm [7] and Toroslu algorithm [22] for group role assignment, which are claimed to achieve better performance than the K-M algorithm.

REFERENCES

[1] G. J. Ahn and H. Hu, "Towards realizing a formal RBAC model in real systems," in *Proc. ACM Symp. Access Control Models Technol.*, Sophia Antipolis, France, 2007, pp. 215–224.

[2] M. A. Al-Kahtani and R. Sandhu, "Induced role hierarchies with attribute-based RBAC," in *Proc. ACM Symp. Access Control Models Technol.*, Como, Italy, 2003, pp. 142–148.

[3] B. E. Ashforth, *Role Transitions in Organizational Life: An Identity-Based Perspective*. Mahwah, NJ: Lawrence Erlbaum Assoc., Inc., 2001.

[4] G. Baker, Java Implementation of the Classic Hungarian Algorithm for the Assignment Problem, 2008. [Online]. Available: http://sites.google.com/site/garybaker/hungarian-algorithm/assignment

[5] M. Bhardwaj and A. P. Chandrakasan, "Bounding the lifetime of sensor networks via optimal role assignments," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, New York, Jun. 2002, vol. 3, pp. 1587–1596.

[6] J. S. Black, "Work role transitions: A study of American expatriate managers in Japan," *J. Int. Business Studies*, vol. 19, no. 2, pp. 277–294, 1988.

[7] D. P. Bertsekas, "A new algorithm for the assignment problem," *Math. Program.*, vol. 21, no. 1, pp. 152–171, Dec. 1981.

[8] R. P. Bostrom, "Role conflict and ambiguity: Critical variables in the MIS user-designer relationship," in *Proc. Annu. Comput. Personnel Res. Conf.*, Miami, FL, 1980, pp. 88–115.

[9] F. Bourgeois and J. C. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Commun. ACM*, vol. 14, no. 12, pp. 802–804, Dec. 1971.

[10] M. Dastani, V. Dignum, and F. Dignum, "Role-assignment in open agent societies," in *Proc. 2nd Int. Joint Conf. Auton. Agents Multiagent Syst.*, Melbourne, Australia, 2003, pp. 489–496.

[11] M. R. Gholamian, S. M. T. Fatemi Ghomi, and M. Ghazanfari, "A hybrid system for multiobjective problems—A case study in NP-hard problems," *Knowl.-Based Syst.*, vol. 20, no. 4, pp. 426–436, May 2007.

[12] A. Janiak, M. Y. Kovalyov, and M. Marek, "Soft due window assignment and scheduling on parallel machines," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 5, pp. 614–620, Sep. 2007.

[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistic Quart.*, vol. 2, pp. 83–97, 1955, (Reprinted in vol. 52, no. 1, 2005, pp. 7–21).

[14] M.-Z. Lee, "Constrained weapon-target assignment: Enhanced very large scale neighborhood search algorithm," *IEEE Trans. Syst., Man Cybern., A, Syst., Humans*, vol. 40, no. 1, pp. 198–204, Jan. 2010.

[15] Y.-W. Leung and R. Y.-T. Hou, "Assignment of movies to heterogeneous video servers," *IEEE Trans. Syst., Man Cybern. A, Syst., Humans*, vol. 35, no. 5, pp. 665–681, Sep. 2005.

[16] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: New insights," *Struct. Multidisciplinary Optim.*, vol. 41, no. 6, pp. 853–862, Jun. 2010.

[17] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, Mar. 1957.

[18] V. T. Y. Ng, B. Chan, L. L. Y. Shun, and R. Tsang, "Quality service assignments for role-based Web services," in *Proc. IEEE Int. Conf. SMC*, Singapore, Oct. 2008, pp. 2219–2224.

[19] J. J. Odell, H. Van Dyke Parunak, S. Brueckner, and J. Sauter, "Changing roles: Dynamic role assignment," *J. Object Technol.*, vol. 2, no. 5, pp. 77–86, Sep.–Oct. 2003.

[20] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vis. Comput.*, vol. 27, no. 7, pp. 950–959, Jun. 2009.

[21] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artif. Intell.*, vol. 110, no. 2, pp. 241–273, Jun. 1999.

[22] I. H. Toroslu and G. Üçoluk, "Incremental assignment problem," *Inf. Sci.*, vol. 177, no. 6, pp. 1523–1529, Mar. 2007.

[23] M. Turoff, M. Chumer, B. Van de Walle, and X. Yao, "The design of a dynamic emergency response management information system (DERMIS)," *J. Inf. Technol. Theory Appl.*, vol. 5, no. 4, pp. 1–35, 2004.

[24] D. Vail and M. Veloso, "Multi-robot dynamic role assignment and coordination through shared potential fields," in *Multi-Robot Systems*, A. Schultz, L. Parkera, and F. Schneider, Eds: Kluwer, 2003, pp. 87–98.

[25] T. Wang, J. Liu, and X. Jin, "Minority game strategies in dynamic multi-agent role assignment," in *Proc. Int. Conf. Intell. Agent Technol.*, Sep. 20–24, 2004, pp. 316–322.

[26] H. Xu, X. Zhang, and R. Patel, "Developing role-based open multi-agent software systems," *Int. J. Comput. Intell. Theory Pract.*, vol. 2, no. 1, pp. 39–56, Jun. 2007.

[27] Y. Zhang, "A role-based approach in dynamic task delegation in agent teamwork," *J. Softw.*, vol. 3, no. 6, pp. 9–20, 2008.
[28] H. Zhu, "Fundamental issues in the design of a role engine," in *Proc. 6th Int. Symp. CTS*, Irvine, CA, May 19–23, 2008, pp. 399–407.
[29] H. Zhu and R. Alkins, "Improvement to rated group role assignment algorithms," in *Proc. IEEE Int. Conf. SMC*, San Antonio, TX, Oct. 2009, pp. 4861–4866.
[30] H. Zhu and R. Alkins, "Group role assignment," in *Proc. Int. Symp. CTS*, Baltimore, MA, May 2009, pp. 431–439.
[31] H. Zhu and M. C. Zhou, "Role-based collaboration and its Kernel mechanisms," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 4, pp. 578–589, Jul. 2006.
[32] H. Zhu and M. C. Zhou, "Role transfer problems and algorithms," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 6, pp. 1442–1450, Nov. 2008.
[33] H. Zhu and M. C. Zhou, "M–M role transfer problems and solutions," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 2, pp. 448–459, Mar. 2009.

Haibin Zhu (M'02–SM'04) received the B.S. degree in computer engineering from the Institute of Engineering and Technology, Zhengzhou, China, in 1983 and the M.S. and Ph.D. degrees in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 1988 and 1997, respectively.

He is a Full Professor with the Department of Computer Science and Mathematics, Nipissing University, North Bay, ON, Canada. He was a Visiting Professor and a Special Lecturer with the College of Computing Sciences, New Jersey Institute of Technology, Newark, from 1999 to 2002 and a Lecturer, an Associate Professor, and a Full Professor at NUDT from 1988 to 2000. He has published more than 100 research papers, 4 books, and 2 book chapters on object-oriented programming, distributed systems, collaborative systems, and computer architecture.

Dr. Zhu is a member of ACM and a Life Member of the Chinese Association for Science and Technology—USA. He has served as Cochair of the Technical Committee of Distributed Intelligent Systems of the IEEE SMC Society, Associate Editor-in-Chief of the *International Journal of Advances in Information and Service Sciences*, Editor of the *International Journal of Intelligent Control and Systems*, Associate Editor of the *International Journal of Agent Technologies and Systems*, Managing Editor of the *International Journal of Electrical, Electronics & Computing Technology, Computer Sciences*, member of the editorial board of the *International Journal of Software Science and Computational Intelligence*, Guest Editor of the IEEE TRANSACTIONS ON SMC(A), organizer of workshops and special sessions on role-based collaboration for more than ten international conferences, and program committee member for more than 40 international conferences. He was the recipient of the 2011 Chancellors' award for excellence in research and 2006–2007 research achievement award from Nipissing University, the 2004 and 2005 IBM Eclipse Innovation Grant Awards, the Best Paper Award from the ISPE International Conference on Concurrent Engineering (ISPE/CE2004), the Educator's Fellowship of OOPSLA'03, a 2nd Class National Award of Excellent Textbook from the Ministry of Education of China in 2002, a 2nd Class National Award of Education Achievement from the Ministry of Education of China in 1997, three 1st Class Ministerial Research Achievement Awards from The Commission of Science Technology and Industry for National Defense of China in 1997, 1994, and 1991, and a 2nd Class Excellent Textbook Award of the Ministry of Electronics Industry of China in 1996.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in electrical engineering from Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990.

He joined New Jersey Institute of Technology (NJIT), Newark, in 1990, where he is currently a Professor of electrical and computer engineering and the Director of the Discrete-Event Systems Laboratory. He is also currently a Professor with Tongji University, Shanghai, China. He has over 400 publications, including 10 books, more than 180 journal papers (majority in IEEE TRANSACTIONS), and 17 book chapters. His research interests are in automated manufacturing systems, lifecycle engineering and sustainability evaluation, Petri nets, wireless ad hoc and sensor networks, system security, semiconductor manufacturing, and embedded control.

Dr. Zhou is a Life Member of the Chinese Association for Science and Technology—USA, where he served as its President in 1999. He was the Founding Chair of the Discrete Event Systems Technical Committee and Founding Cochair of the Enterprise Information Systems Technical Committee of the IEEE SMC Society, and Chair (founding) of the Semiconductor Manufacturing Automation Technical Committee of the IEEE Robotics and Automation Society. He is founding editor for IEEE Press Series on Systems Sciences and Engineering, Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: PART A and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and Editor-in-Chief of the *International Journal of Intelligent Control and Systems*. He served as guest editor for many journals including many IEEE Transactions. He was the General Cochair of the 2003 IEEE International Conference on System, Man and Cybernetics (SMC), Washington DC, on October 5–8, 2003; Founding General Cochair of the 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, on March 21–23, 2004; General Chair of the 2006 IEEE International Conference on Networking, Sensing and Control, Ft. Lauderdale, FL, on April 23–25, 2006; and General Chair of the IEEE Conference on Automation Science and Engineering, Washington D.C., on August 23–26, 2008. He was the Program Chair of the 1998 and 2001 IEEE International Conference on SMC and the 1997 IEEE International Conference on Emerging Technologies and Factory Automation. He organized and chaired over 80 technical sessions and served on program committees for many conferences. He has led or participated in over 40 research and education projects with a total budget of over $10 M funded by the National Science Foundation, Department of Defense, Engineering Foundation, New Jersey Science and Technology Commission, and industry. He was the recipient of the NSF's Research Initiation Award, CIM University-LEAD Award by the Society of Manufacturing Engineers, Perlis Research Award by NJIT, Humboldt Research Award for U.S. Senior Scientists, Leadership Award and Academic Achievement Award by the Chinese Association for Science and Technology—USA, Asian American Achievement Award by the Asian American Heritage Council of New Jersey, and Distinguished Lecturership of the IEEE SMC Society.

Rob Alkins received the B.S. degree in computer science and mathematics (double major) and the M.S. degree from Nipissing University, North Bay, ON, Canada, in 2008 and 2010, respectively.

He is currently a Mathematician/Software Researcher for Stroma Service Inc., North Bay.