Avoiding Conflicts by Group Role Assignment

Haibin Zhu, *Senior Member, IEEE*

***Abstract*—Role assignment is a critical element in the role-based collaboration process. There are many constraints to be considered when undertaking this task. This paper formalizes the group role assignment problem when faced with the constraint of conflicting agents, verifies the benefits of solving the problem, proves that such a problem is a subproblem of the extended integer linear programming (x-ILP) problem, proposes a practical approach to the solution, and assures performance based on the results of experiments. The contributions of this paper include: 1) formalization of the proposed problem; 2) verification of the benefit achieved by avoiding conflicts in role assignment through simulation; 3) theoretical proof that conflict avoidance is a subproblem of the x-ILP problem that is nonpolynomial-complete; and 4) a practical solution based on the IBM ILOG CPLEX optimization package (ILOG) and verification of the scale of problems that can be solved with ILOG. The proposed approach is validated by simulation experiments. Its efficiency is verified by comparison with the previous exhaustive search-based approach.**

***Index Terms*—Agents, conflicting agents, role assignment, role-based collaboration (RBC), roles.**

I. Introduction

ROLE-based collaboration (RBC) is an emerging computational methodology that uses roles as a primary underlying mechanism to facilitate collaborative activities that include abstraction, classification, separation of concerns, dynamics, sharing, resource allocation, interactions, coordination, and decision making [29]–[36]. Many significant challenges have arisen through continuous research effort in RBC [29]–[36] and interestingly, these challenges have numerous corresponding engineering problems in the real world. Solving such problems meets the requirements of engineering practice. Our previous research has solved several problems typical in role assignment and transfer [29]–[34], [36]. The challenge of role assignment has been revealed as a complex process throughout the life cycle of RBC (Fig. 1), i.e., agent evaluation, role assignment, and role transfer. Group role assignment (GRA) seeks an optimal role-to-agent assignment based on the results of agent evaluations (Fig. 1) [31], [36] and greatly affects collaboration efficiency and the degree of satisfaction of members involved in RBC.

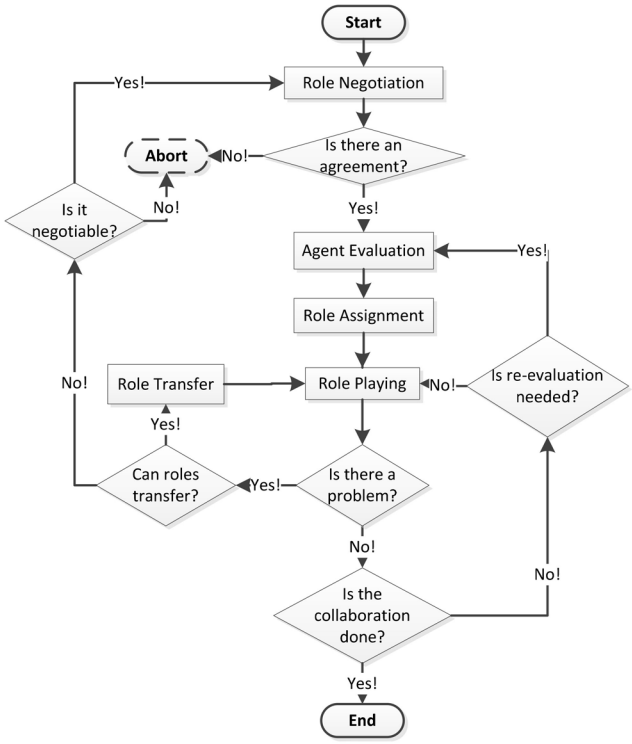

Fig. 1. Life cycle of RBC.

GRA is itself a complex problem where the exhaustive-search algorithm has an exponential increase in complexity. In our previous work, an efficient algorithm was developed by using the Hungarian algorithm (also called Kuhn–Munkres algorithm [13], [15]) [36].

The GRA solution solves a role assignment problem [36] considering the following constraints: 1) the number of agents required for each role and 2) only one role assigned to an agent.

After the GRA problem is solved, role assignment becomes a straightforward process if there are no additional constraints. However, there can be many constraints in role assignment. Agent conflicts are an example.

Conflicts are a common phenomenon in the real world. For example, in transportation, different chemicals (agents) cannot be stored in the same container (a role); in the animal world, a hunter and a prey (agents) cannot be placed in the same cage (a role); and in wireless communication, various devices (agents) cannot be put in the same area (a role) due to potential interference [8]. There are also many types of

Manuscript received July 19, 2014; revised January 15, 2015; accepted April 20, 2015. Date of publication June 12, 2015; date of current version March 15, 2016. This work was supported by the Natural Sciences and Engineering Research Council, Canada, under Grant RGPIN262075-2012, Grant RGPIN262075-2013, Grant EGP 437912-12, and Grant EGP449357-13. This paper is extended significantly from our previous work [30], [32]. This paper was recommended by Associate Editor L. Fang.

The author is with the Department of Computer Science and Mathematics, Nipissing University, North Bay, ON P1B 8L7, Canada, and also with the School of Management and Engineering, Nanjing University, Nanjing 210093, China (e-mail: haibinz@nipissingu.ca).

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/TSMC.2015.2438690

conflicts among people in society. For example, unresolved emotional conflicts may prevent John and Matt from working together. Furthermore, multiagent systems rarely exist without conflicts [14], [24].

In the real world, frustrations arise in collaboration because pre-existing agent conflicts may not be considered in the role assignment process. Much effort can be expended in resolving or handling such conflicts [2], [3], [10]–[12] during collaboration, i.e., role playing (Fig. 1). A primary objective of this paper is to establish a mechanism whereby conflicts are dealt with before the concrete cooperation process (i.e., role playing) begins.

To successfully accomplish the assignment task as an important step of collaboration, it is necessary to prevent conflicting agents from involvement with the same role (task, position, or area). This idea is called conflict avoidance [25]. Although such problems arise in the research of RBC, they are also important and challenging in the domains of administration, production, and engineering.

In this paper, we assume that it is possible for managers or administrators to find agents that will not be in conflict through role assignment. We admit that there are many cases, due to situation dynamics, where conflicts are unavoidable during collaboration and appropriate resolution is required after the collaborative process begins. However, such a reality does not reduce the significance of solving the proposed problem.

The major contributions of this paper are the formalization of the problem, verification of the benefits in solving such a problem, theoretical proof that the problem is a subproblem of an nonpolynomial (NP)-complete problem, and the provision of a practical solution.

This paper is organized as follows. It describes a real-world scenario related to the proposed problem in Section II, condenses the environments—classes, agents, roles, groups, and objects (E-CARGO) model in Section III, formally defines and specifies the problem in Section IV, and verifies the benefits of solving the problem through simulation in Section V. Section VI offers a theoretical proof that the problem is a subproblem of the extended integer linear programming (x-ILP) problem. Practical solutions based on the IBM ILOG CPLEX optimization package (ILOG) are set out in Section VII; comparisons with initial solutions appear in Section VIII; performance data obtained from experiments is analyzed in Section IX; and related work is discussed in Section X. This paper concludes by pointing out topics for future work in Section XI.

II. Real-World Scenario

In company X, Ann, the Chief Executive Officer, has just signed a contract, the value of which is a million dollars. She asks Bob, the Human Resources Officer, to organize a team from employees of the company. Bob drafts a position list shown in Table I for the team and a candidate staff shortlist shown in Table II. Then, Bob initiates an evaluation process and asks the branch officers to evaluate employees for each possible position (Table II). After that, when Ann and Bob have a meeting with the branch officers to assign positions to candidate employees, the officers report that from historical experiences, employee conflicts arise for various reasons, such as personality characteristics, working styles, emotional issues, and political beliefs (Table III). That is to say, when constructing an agile group in a single office, performance cannot be compromised due to employee conflicts. Ann instructs Bob to assign the most qualified candidates to positions while avoiding employee conflicts. After some consideration, Bob suggests that a satisfactory solution, in light of such a challenge, may require a significant amount of time. Fortunately, Ann, as an experienced administrator, understands the complexity of the problem and does not demand an unreasonable response timeframe.

From the above scenario, Ann and Bob actually follow the initial steps of RBC and Bob encounters a problem of GRA that forces consideration of additional constraints. The final optimized assignment, avoiding conflicts,

TABLE I
Required Positions

Position	Project Manager	Senior Programmer	Programmer	Tester
Required Number	1	2	4	2

TABLE II
Candidates and Position Evaluations

Positions / **Candidates**	Project Manager	Senior Programmer	Programmer	Tester
Adam	0.18	**0.82**	0.29	0.01
Brian	0.35	0.80	**0.58**	0.35
Chris	0.84	**0.85**	0.86	0.36
Doug	0.96	0.51	0.45	**0.64**
Edward	0.22	0.33	**0.68**	0.33
Fred	0.96	0.50	0.10	**0.73**
George	0.25	0.18	0.23	0.39
Harry	0.56	0.35	**0.80**	0.62
Ice	0.49	0.09	0.33	0.58
Joe	0.38	0.54	**0.72**	0.20
Kris	**0.91**	0.31	0.34	0.15
Larry	0.85	0.34	0.43	0.18
Matt	0.44	0.06	0.66	0.37

TABLE III
Conflicts*

Conflicts / **Candidates**	1	2	3
Adam	Brian		
Brian	Adam		
Chris			
Doug			
Edward	Fred	Larry	Matt
Fred	Edward		
George			
Harry			
Ice			
Joe			
Kris			
Larry	Edward		
Matt	Edward		

*Note: Adam is in conflict with Brian, and Edward is in conflict with Fred, Larry, and Matt.

is shown in Table II, i.e., a tuple set as: {<Adam, Senior Programmer>, <Brian, Programmer>, <Chris, Senior Programmer>, <Doug, Tester>, <Edward, Programmer>, <Fred, Tester>, <Harry, Programmer>, <Joe, Programmer>, and <Kris, Project Manager>}. The total sum of assigned evaluation values is 6.73. This scenario clearly demonstrates the significance of the proposed problem. It appears that Doug's assignment may not be the best use of his talents. However, it can be supported if we consider overall team performance.

III. Condensed E-CARGO Model

With the E-CARGO model [35], a system $\sum$ can be described as a nine-tuple $\sum ::= <\mathcal{C}, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H}>$, where $\mathcal{C}$ is a set of classes, O is a set of objects, $\mathcal{A}$ is a set of agents, $\mathcal{M}$ is a set of messages, $\mathcal{R}$ is a set of roles, $\mathcal{E}$ is a set of environments, $\mathcal{G}$ is a set of groups, s_0 is the initial state of the system, and $\mathcal{H}$ is a set of users. In such a system, $\mathcal{A}$ and $\mathcal{H}$, $\mathcal{E}$ and $\mathcal{G}$ are tightly-coupled sets. A human user and his/her agent perform a role together. Every group should work in an environment. An environment regulates a group.

In the following discussions, we focus on current agents or roles [33]–[36]. Environments and groups are simplified into vectors and matrices, respectively. Furthermore, we use non-negative integers m (=$|\mathcal{A}|$) to express the size of the agent set $\mathcal{A}$, n (=$|\mathcal{R}|$) the size of the role set $\mathcal{R}$, i, i_1, i_2, ... the indices of agents, and j, j_1, j_2, ... the indices of roles. Note: we use $\mathcal{N}$ to denote the set of non-negative integers.

Definition 1: A role range vector L is an n vector of the lower bound of the ranges of roles in environment e of group g.

Note: L is a valuable component in the E-CARGO model. It reveals many challenges in role assignment and the process of RBC. Without this component, we would not discover the problems discussed in this paper. L is in fact a simplified and derived concept from the component environment in the E-CARGO model. It provides an indication of the minimum number of agents required for each role within a properly functioning group, i.e., $L[j] \geq 1 (0 \leq j < n)$.

Definition 2: A qualification matrix Q is an $m \times n$ matrix, where $Q[i, j] \in [0, 1]$ expresses the qualification value of agent $i \in \mathcal{N}$ $(0 \leq i < m)$ for role $j \in \mathcal{N}$ $(0 \leq j < n)$. $Q[i, j] = 0$ indicates the lowest value and 1 the highest.

Note that, a Q matrix can be obtained by comparing all the Ⓠs of agents with all the ®s of roles [32].

Definition 3: A role assignment matrix T is defined as an $m \times n$ matrix, where $T[i, j] \in \{0, 1\}$ $(0 \leq i < m, 0 \leq j < n)$ indicates whether agent i is assigned to role j or not. $T[i, j] = 1$ means yes and 0 no.

Definition 4: The group performance σ (named after the sigma Σ that has been used as a system in the E-CARGO model) of group g is defined as the sum of the assigned agents' qualifications, that is

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

$$\begin{bmatrix} 0.18 & 0.82 & 0.29 & 0.01 \\ 0.35 & 0.80 & 0.58 & 0.35 \\ 0.84 & 0.85 & 0.86 & 0.36 \\ 0.96 & 0.51 & 0.45 & 0.64 \\ 0.22 & 0.33 & 0.68 & 0.33 \\ 0.96 & 0.50 & 0.10 & 0.73 \\ 0.25 & 0.18 & 0.23 & 0.39 \\ 0.56 & 0.35 & 0.80 & 0.62 \\ 0.49 & 0.09 & 0.33 & 0.58 \\ 0.38 & 0.54 & 0.72 & 0.20 \\ 0.91 & 0.31 & 0.34 & 0.15 \\ 0.85 & 0.34 & 0.43 & 0.18 \\ 0.44 & 0.06 & 0.66 & 0.37 \end{bmatrix} \quad \begin{bmatrix} 0\,1\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}$$

(a) (b)

Fig. 2. Qualification matrix Q (a) and assignment matrix T (b).

Definition 5: Role j is workable in group g if it has been assigned a sufficient number of agents, i.e., $\sum_{i=0}^{m-1} T[i,j] \geq L[j]$.

Definition 6: T is workable if each role j is workable, i.e., $\forall (0 \leq j < n) \sum_{i=0}^{m-1} T[i,j] \geq L[j])$. Group g is workable if T is workable.

From the above definitions, group g can be expressed by a Q, an L, and a T. In the following discussions, we assume that $m \geq \sum_{i=0}^{n-1} L[j]$ unless special cases are clearly specified.

For example, Fig. 2(a) is a qualification matrix for Table II. Fig. 2(b) is an assignment matrix that makes the group (Table II) work with vector $L = [1, 2, 4, 2]$ in Table I. The sum of the assigned values is 6.96.

IV. Problem Formalizations

In fact, Definitions 3–6 concern a group state after role assignment that is highly dependent on various conditions. Therefore, we need to specify constraints before assignment is undertaken. Conflicting agents can be formally defined as follows [30], [32].

Definition 7: Two different agents $i_1 \in \mathcal{N}$ and $i_2 \in \mathcal{N}$ $(0 \leq i_1, i_2 < m, i_1 \neq i_2)$ are in conflict on roles if i_1 and i_2 cannot be assigned to the same role. We say that i_1 is a conflicting agent of i_2 on roles and vice versa.

Definition 8: Two different agents i_1 and i_2 $(0 \leq i_1, i_2 < m, i_1 \neq i_2)$ are in conflict in groups if i_1 and i_2 cannot be assigned to the same group. We say that i_1 is a conflicting agent of i_2 in groups and vice versa.

Definition 9: A conflicting agent matrix is defined as an $m \times m$ matrix A^c (A^c $[i_1, i_2] \in \{0, 1\}$, $0 \leq i_1, i_2 < m$), where $A^c[i_1, i_2] = 1$ expresses that agent i_1 is in conflict with agent i_2 and 0 means not. We define $A^c[i, i] = 0$ $(0 \leq i < m)$ for simplicity.

Note: A^c is a symmetric matrix along the diagonal from $A^c[0, 0]$ to $A^c[m-1, m-1]$, i.e., $(A^c[i_1, i_2] = A^c[i_2, i_1])$ $(0 \leq i_1, i_2 < m, i_1 \neq i_2)$.

For Table III, we have the corresponding A^c shown in Fig. 3(a).

$$\begin{bmatrix}0100000000000\\1000000000000\\0000000000000\\0000000000000\\0000010000011\\0000100000000\\0000000000000\\0000000000000\\0000000000000\\0000000000000\\0000000000000\\0000100000000\\0000100000000\end{bmatrix} \quad \begin{bmatrix}0100\\0010\\0100\\0001\\0010\\0001\\0000\\0010\\0000\\0010\\1000\\0000\\0000\end{bmatrix}$$

(a) (b)

Fig. 3. Conflict matrix A^c for Table III (a) and a corresponding assignment matrix T (b).

Definition 10: GRA with conflicting agents on roles (GRACAR) is to find a workable T to

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

subject to

$$T[i,j] \in \{0,1\} \quad (0 \le i < m, 0 \le j < n) \tag{1}$$

$$\forall j \sum_{i=0}^{m-1} T[i,j] = L[j] (0 \le j < n) \tag{2}$$

$$\forall i \sum_{i=0}^{n-1} T[i,j] \le 1 (0 \le i < m) \tag{3}$$

$$\forall (0 \le i_1, i_2 < m)(0 \le j < n)(i_1 \ne i_2) A^c[i_1, i_2] \times (T[i_1, j] + T[i_2, j]) \le 1 \tag{4}$$

where expression (1) is a 0-1 constraint; (2) makes the group workable; (3) means that each agent can only be assigned to one role; and (4) shows that no conflicting agents are assigned to the same role.

For example, Fig. 3(b) is a T following Definition 10. The sum of the assigned evaluation values is 6.73 compared with the sum of 6.96 that indicates the possibility of conflicts. This also indicates that the avoidance of conflicts does not significantly degrade team performance. However, if conflicts arise during the role playing phase, much additional effort is required for their resolution.

Definition 11: GRA with conflicting agents in a group (GRACAG) is to find a workable T to

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

subject to (1)–(3), and

$$\forall (0 \le i_1, i_2 < m)(0 \le j_1, j_2 < n) (i_1 \ne i_2) A^c[i_1, i_2] \times (T[i_1, j_1] + T[i_2, j_2]) \le 1 \tag{5}$$

where (5) means that no conflicting agents can be assigned to the same group.

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

(a)

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

(b) (c)

Fig. 4. Example. (a) Q matrix. (b) T matrix. (c) Conflict matrix A^c.

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

(a)

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

(b)

Fig. 5. A^c and T matrices for different problems. (a) Workable T considering A^c on roles. (b) Workable T considering A^c in a group and $L = [1\ 1\ 1\ 1]$.

From the above definitions, GRACAG has more restrictions in obtaining T than GRACAR does. For example, suppose that Q is shown as Fig. 4(a), $L = [2\ 1\ 1\ 2]$, then T in Fig. 4(b) is a solution when no conflict is considered. However, if the conflicting agent matrix in Fig. 4(c) is considered in a GRACAR problem, a solution is shown in Fig. 5(a). If Fig. 4(c) is considered in a GRACAG problem, then there is no solution for $L = [2\ 1\ 1\ 2]$. Fig. 5(b) shows a solution for $L = [1\ 1\ 1\ 1]$. From these examples, it is understandable that there might be no a solution for a group expressed by *Q, L,* and A^c. Also, we know that if $L[j] = 1$ for all $(0 \le j < n)$, no conflict occurs for GRACAR. This fact indicates the importance of L in the E-CARGO model.

To help solving GRACAR and GRACAG problems, we need to know if a problem is solvable.

Definition 12: The number of required agents n_a is defined as $\sum_{j=0}^{n-1} L[j]$.

Definition 13: The number of conflicts n_c is defined as $\sum_{i_1=0}^{m-1}\sum_{i_2=i_1+1}^{m-1} A^c[i_1, i_2]$.

Theorem 1: The GRACAR and GRACAG problems are solvable if $m - n_c \ge n_a$.

Proof: To solve the problems of GRACAR and GRACAG, we need to find a subset $\mathcal{A}'$ from the original agent set $\mathcal{A}$, and that there are no conflicts in $\mathcal{A}'$ and that the number of agents must meet the requirement of L, i.e., $|\mathcal{A}'| \ge n_a$.

$\because m - n_c \ge n_a$.

$\therefore$ We may delete one agent from each conflicting pair of agents, i.e., let set $\mathcal{X} = \{i | A^c[i, i'] = 1 (0 \le i < m, i+1 \le i' < m)\}$ and $\mathcal{A}' = \mathcal{A} - \mathcal{X}$. Now, $\mathcal{A}'$ is free of conflicts.

$\because$ One agent may be conflict with more than one other agent

$\therefore |\mathcal{X}| \le n_c$

$\therefore |\mathcal{A}'| = |\mathcal{A}| - |\mathcal{X}| \ge n_a$.

Theorem 1 is proved. ■

Note that Theorem 1 is a sufficient condition but not necessary. For the example discussed in Section II, we do not have to remove any agents for GRACAR. For GRACAG, we only need to remove 2 (Adam and Edward) but not 4 to acquire the solution. Although Theorem 1 just states the sufficient condition, it is useful for HR officers when shortlisting candidates.

V. Benefits of Avoiding Conflicts

To quantitatively present the benefits of avoiding conflicts, we conduct simulation with random groups. In the simulation, we choose a typical size, i.e., $m = 100$, $n = 16$, without loss of generality, because we concentrate only on benefit but not efficiency. This is very common in a middle-sized company or factory. We need two new definitions in this situation.

Definition 14: The number of assigned conflicts n_{ac} is defined as $\sum_{j=0}^{n-1} \sum_{i_1=0}^{m-1} \sum_{i_2=i_1+1}^{m-1} A^c[i_1, i_2] \times T[i_1, j] \times T[i_2, j]$, where T is obtained without considering A^c, i.e., the result of GRA.

Definition 15: The conflict rate p_c (named after percentage of conflicts) is defined as $n_c/[m \times (m-1)/2]$, i.e., the number of conflicts divided by the total number of pairs of agents.

For example, the p_c in Table III is $4/(13 \times 6) \approx 5.13\%$.

To estimate the benefits of conflict avoidance, we need an assumption. Conflicts are different and affect collaborations in varying degrees [24], [25]. Some conflicts may be ignored, others must be resolved, and others may even lead to loss of life, i.e., on the battlefield, soldiers needing to accomplish the same task must not be hampered by conflicts. By drawing an analogy between bugs in software [19] and conflicts in collaboration, we can categorize conflicts into different types: 1) ignorable (0 loss); 2) mild (20% loss); 3) annoying (40% loss); 4) disturbing (60% loss); 5) serious (80% loss); and 6) extreme (100% loss). Others may produce larger losses (>100%). Because we are discussing collaboration, conflicts that are categorized as worse than extreme are ignored. In fact, if we assume that a loss is more than 40%, conflict avoidance may provide a higher benefit λ to be discussed below. Therefore, we consider that the conflicting agents lose 40% (the annoying scale) of their original qualifications and that a conflicting pair affects both parties.

For each group, we randomly choose the following data within the range of the definitions.

1) $Q[i, j] (0 \le i < m, 0 \le j < n)$, i.e., the qualification value of each agent for each role.
2) $A^c[i_1, i_2] (0 \le i_1, i_2 < m)$, i.e., the conflicts between agents.
3) $L[i] (0 \le j < n)$, $1 \le L[i] \le 6$ to make $m \ge n_a$.

For each group, we collect several data items.

1) σ_1, the maximum group performance without considering conflicts.
2) σ_2, the maximum group performance after avoiding conflicts.
3) σ_2/σ_1, the comparison between σ_2 and σ_1 in percentage.
4) n_a, the number of the required agents.
5) n_c, the number of conflicts.
6) n_{ac}, the number of assigned conflicts in the T corresponding to σ_1.

The benefit λ is calculated as follows.

1) We obtain each agent's average qualification, that is
$$\sigma_1/n_a.$$
2) Then, the lost group performance is
$$\sigma_1 \times 2 \times 0.4 \times n_c/n_a.$$
3) The adjusted group performance is
$$(1 - 0.8 \times n_c/n_a) \times \sigma_1.$$
4) Then, we have the benefit
$$\lambda = \left[\sigma_2 - (1 - 0.8 \times n_c/n_a) \times \sigma_1\right] \Big/ \left[(1 - 0.8 \times n_c/n_a) \times \sigma_1\right].$$

TABLE IV
Simulation Averages ($m = 100$ and $n = 16$)

p_c	σ_1	σ_2	σ_2/σ_1	n_a	n_c	n_{ac}	λ
1/3	54.05	53.30	98.61%	56.14	859.47	14.58	24.37%
1/4	53.80	53.35	99.16%	55.85	618.94	10.74	17.12%
1/5	53.54	53.21	99.38%	55.57	492.52	08.18	12.60%
1/6	54.01	53.73	99.48%	56.10	397.67	06.86	10.25%
1/7	54.83	54.59	99.56%	56.95	343.22	06.41	09.36%
1/8	54.29	54.06	99.58%	56.37	321.61	06.01	08.84%
1/9	53.86	53.69	99.68%	55.90	267.02	04.72	06.88%
1/10	53.43	53.26	99.68%	55.47	247.63	04.50	06.59%
1/11	53.70	53.55	99.72%	55.78	227.13	04.14	05.99%
1/12	53.94	53.80	99.74%	56.01	203.97	03.71	05.32%
1/13	54.04	53.92	99.78%	56.14	186.10	03.31	04.70%
1/14	53.96	53.84	99.78%	56.01	182.60	03.28	04.65%
1/15	53.38	53.27	99.79%	55.45	169.86	03.04	04.36%
1/16	53.73	53.63	99.81%	55.79	151.13	02.95	04.21%
1/17	54.19	54.09	99.82%	56.25	146.60	02.78	03.90%
1/18	54.05	53.97	99.85%	56.13	132.33	02.40	03.36%

To observe the differences among different configurations, we also change the conflict rate p_c from 1/3 to 1/18. To make the data convincing, we create 300 groups for each conflict rate. Finally, we compute averages of the 300 numbers for each of the seven data items. Table IV shows the simulation results.

By observing the simulation results, it is evident that conflict avoidance does not significantly degrade group performance. Group performance ranges from 98.61% to 99.85% in the 16 types of groups with different conflict rates. Every assignment in these groups introduces conflicts corresponding to the conflicting rate p_c.

Based on the above simulation data and estimations, the right-most column λ of Table IV assures the benefit of avoiding conflicts. However, it is not a trivial matter to solve GRACAR and GRACAG problems. We tried several initial algorithms [29], [30]. The complexity of these

initial algorithms is exponential. Experiments also demonstrate the complexity of the initial solutions to the proposed problems [29], [30]. Therefore, we need to analyze the complexity of the proposed problem and develop a practical solution.

VI. GRACAR and GRACAG Problems Are Subproblems of NP-Complete Problem

In this section, we provide proof that GRACAR and GRACAG problems are subproblems of the x-ILP that can be restricted to the integer linear programming (ILP) problem that is NP-complete [18], [20], [21], [27]. This proof assists in finding solutions using tools that solve ILP problems.

Theorem 2: The GRACAR and GRACAG problems are in the NP class.

Proof: According to Garey and Johnson [7], a problem is in NP if there is a nondeterministic algorithm used to guess a solution and check in polynomial time whether it adheres to the constraints.

For GRACAR, we may discuss a problem $\boldsymbol{\Pi}$ derived from GRACAR to answer whether $\sigma \geq \boldsymbol{b}(\sigma = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j])$ but not to determine the maximum value of σ. If a guessed T' for $\boldsymbol{\Pi}$ is provided, we can use the following algorithm to check if $\sigma \geq \boldsymbol{b}$ and T' follows constraints (1)–(4), where $\sigma \geq \boldsymbol{b}$, (1)–(3) are straightforward.

For (4), the algorithm is (in Java-like format)

for j **from** 0 **to** $n-1$ **do**
  for i_1 **from** 0 **to** m $-$ 1 **do**
    for i_2 **from** i_1+1 **to** m $-$ 1 **do**
      if $(A^c[i_1, i_2] \times (T'[i_1, j] + T'[i_2, j])>1)$ **return** false;
return true;

The time used by this algorithm is polynomial, i.e., $O(n \times m^2) \leq O(m^3)$.

By considering other constraints, the total time is still $O(m^3)$, i.e., polynomial.

Now, $\boldsymbol{\Pi}$ is in NP.

Therefore, GRACAR is in NP.

In the same way, we may determine the time of checking as $O(m^3)$ for GRACAG.

Theorem 2 is proved. ■

Definition 21: The x-ILP problem is a class of problems to find an $X = [x_0 x_1 \ldots x_{n'-1}]$ to obtain

$$\max \sum_{j=0}^{n'-1} c_j x_j$$

subject to

$$x_j \in \{0, 1\} \quad (0 \leq j < n') \tag{6}$$

$$\sum_{j=0}^{n'-1} a_{ij} x_j \prec b_i (0 \leq i < m') \tag{7}$$

where "$\prec$" expresses "$<$," "$>$," "$=$," "$\neq$," "$\leq$," or "$\geq$."

Theorem 3: The x-ILP problem is NP-complete.

Proof: Based on the method provided in [7, pp. 63–65], we use proof by restriction. The x-ILP problem can be restricted to the ILP problem [18], [20], [21] by replacing "$\prec$" with "$\leq$," that is

$$\sum_{j=0}^{n'-1} a_{ij} x_j \leq b_i (0 \leq i < m'). \tag{8}$$

Because the ILP problem is NP-complete [7], the x-ILP problem is NP-complete.

Theorem 3 is proved. ■

Note that, even though the expression of the ILP problem here is a little different from that in [7], it is common to express the ILP problem in the form of

$$\max \sum_{j=0}^{n'-1} c_j x_j$$

and constraints (6) and (8) in other contributions [18], [20], [21].

To simplify descriptions, we let $C = [c_0 c_1 \ldots c_{n'-1}]$, $B = [b_0 b_1 \ldots b_{m'-1}]$, and $A = [a_{ij}](0 \leq i < m', 0 \leq j < n')$.

Theorem 4: The GRACAR problem is a subproblem of the x-ILP problem.

Proof: According to Garey and Johnson [7], a problem $\boldsymbol{\Pi}$ can be expressed as $<\boldsymbol{D}, \boldsymbol{Y}>$, where $\boldsymbol{D}$ is the set of all the problem instances of $\boldsymbol{\Pi}$ and $\boldsymbol{Y}$ is the set of all the problem instances that can be successfully solved. Another problem $\boldsymbol{\Pi}'$ expressed as $<\boldsymbol{D}', \boldsymbol{Y}'>$ is a subproblem of $\boldsymbol{\Pi}'$ if $\boldsymbol{D}' \subseteq \boldsymbol{D}$, and $\boldsymbol{Y}' = \boldsymbol{Y} \cap \boldsymbol{D}'$.

Now, let x-ILP $= \boldsymbol{\Pi}$ and GRACAR $= \boldsymbol{\Pi}'$.

We prove $\forall d' \in D' \rightarrow d' \in D$.

From a GRACAR problem d', we have L, Q, A^c, and T.

Let

$$\begin{aligned} n' &= m \times n \\ c_j &= Q[j/m, j\%m](0 \leq j < n') \\ x_j &= T[j/m, j\%m](0 \leq j < n'). \end{aligned}$$

We have $\sum_{j=0}^{n'-1} c_j x_j = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j]$.

That is, the objective of the ILP and that of the GRACAR problem is the same.

At the same time, constraint (1), $T[i,j] \in \{0, 1\}(0 \leq i < m, 0 \leq j < n)$ is the same as constraint (6), i.e., $x_j \in \{0, 1\}$, $(0 \leq i < n')$.

Let

$$\begin{aligned} m' &= n + m + n \times n_c \\ b_j &= L[j](0 \leq j < n) \\ b_j &= 1(n \leq j < m') \\ a_{ij} &= 0(0 \leq i < m', 0 \leq j < n') \\ a_{i(i+k\times n)} &= 1(0 \leq i < n, 0 = k < m) \\ a_{i(k\times n+j\%n)} &= 1(n \leq i < m+n, 0 \leq j < n', \\ &\qquad 0 = k < n, k+n = i) \\ a_{(m+n+i\times n+k)(i\times n+k)} &= 1(0 \leq i < m, \exists j(0 \leq j < m) \ni A^c[i,j] \\ &\qquad = 1, 0 \leq k < n) \\ a_{(m+n+i\times n+k)(j\times n+k)} &= 1(0 \leq i, j < m, A^c[i,j] = 1, \\ &\qquad 0 \leq k < n). \end{aligned}$$

Then, constraints (2)–(4) are the same as constraint (7), i.e., $\sum_{j=0}^{n'-1} a_{ij}x_j = b_i (0 \le i < n)$, and $\sum_{j=0}^{n'-1} a_{ij}x_j \le b_i (n \le i < m')$.

Now, a GRACAR problem is expressed as an x-ILP problem, i.e., $\forall d' \in D' \to d' \in D$

$$\because \forall d' \in D' \to d' \in D$$
$$\therefore \boldsymbol{D'} \subseteq \boldsymbol{D}.$$

At the same time, in the above transformation, no new constraints and conditions are introduced. That is to say, if problem d' is solvable in D', then it is still solvable in D.

Now, let us prove $\boldsymbol{Y'} = Y \cap D'$.

We define $S(d)$ as a predicate that d is solvable and use "$\leftrightarrow$" to express "is equivalent to"

$$\because d \in Y \leftrightarrow d \in D \wedge S(d)$$
$$d' \in Y' \leftrightarrow d' \in D' \wedge S(d'),$$
$$d' \in D' \to d' \in D$$
$$\therefore d' \in D' \wedge S(d') \to d' \in D \wedge S(d'), \text{ i.e.,}$$
$$\forall d' \in Y' \to d' \in D \wedge S(d') \wedge d' \in D'$$
$$\because d' \in D \wedge S(d') \leftrightarrow d' \in Y$$
$$\therefore \forall d' \in Y' \to d' \in Y \wedge d' \in D'$$
$$\therefore \boldsymbol{Y'} \subseteq \boldsymbol{Y} \cap \boldsymbol{D'}$$
$$\because d' \in D \wedge S(d') \wedge d' \in D' \to d' \in D' \wedge S(d') \to d' \in Y'$$
$$\therefore \forall d' \in D \wedge S(d') \wedge d' \in D' \to d' \in Y'$$
$$\because d' \in D \wedge S(d') \leftrightarrow d' \in Y$$
$$\therefore \forall d' \in Y \wedge d' \in D' \to d' \in Y'$$
$$\text{i.e., } \boldsymbol{Y} \cap \boldsymbol{D'} \subseteq \boldsymbol{Y'}$$
$$\therefore \boldsymbol{Y'} = \boldsymbol{Y} \cap \boldsymbol{D'}.$$

Therefore, Theorem 4 is proved. ■

To better understand the proof, consider the example shown in Fig. 4.

From Figs. 4 and 5, $m = 6$, $n = 4$, and $L = [2\ 1\ 1\ 2]$. Q and A^c are as shown in the figures.

Let

$$n' = m \times n = 24$$
$$m' = n + m + n \times n_c = 4 + 6 + 4 \times 3 = 22$$
$$C = \big[0.71\ 0.6\ 0.0\ 0.22\ 0.29\ 0.67\ 0.44\ 0.76\ 0.69$$
$$0.92\ 0.92\ 0.6\ 0.0\ 0.0\ 0.53\ 0.0\ 0.97\ 0.51\ 0.77$$
$$0.65\ 0.58\ 0.64\ 0.24\ 0.0\big]$$
$$B = [2\ 1\ 1\ 2\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$

and A is shown in Fig. 6.

Theorem 5: The GRACAG problem is a subproblem of the x-ILP problem.

Proof: We can use the technique as shown in the proof of Theorem 4 to prove that $\boldsymbol{D'} \subseteq \boldsymbol{D}$ and $\boldsymbol{Y'} = \boldsymbol{Y} \cap \boldsymbol{D'}$.

The only additional part we need to prove is to determine A as follows.

Let

$$a_{ij} = 0 \quad (0 \le i < m', 0 \le j < n')$$
$$a_{i(i+k\times n)} = 1 \quad (0 \le i < n, 0 \le k < m)$$

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

Fig. 6. Matrix A for the GRACAR problem.

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

Fig. 7. Matrix A for the GRACAG problem.

$$a_{i(k\times n+j\%n)} = 1$$
$$(n' \le i < n + n', 0 \le j < n', 0 \le k < n, k + n' = i)$$
$$a_{(2n'+i\times n+k)(i\times n+k)} = 1$$
$$(0 \le i < m, \exists j (0 \le j < m) \ni A^c[i, j] = 1, 0 \le k < n); \text{ and}$$
$$a_{(2n'+i\times n+k)(i\times n+k+k')} = 1$$
$$(0 \le i, j < m, A^c[i, j] = 1, 0 \le k, k' < n).$$

Then, constraints (2), (3), and (5) are the same as constraint (7), i.e., $\sum_{j=0}^{n'-1} a_{ij}x_j = b_i (0 \le i < n)$, and

$$\sum_{j=0}^{n'-1} a_{ij}x_j \le b_i (n \le i < m').$$

Now, the GRACAG problem is expressed as an x-ILP problem.

Therefore, Theorem 5 is proved. ■

For example, for the GRACAG problem shown in Fig. 4, m', n', B, and C are the same as those for the GRACAR problem, and A is shown in Fig. 7.

Theorem 6: GRACAR and GRACAG problems are transformed into an x-ILP problem in polynomial time.

Proof: We may estimate the time of the transformation from GRACAR to x-ILP in the following steps.

1) To form C, we use $O(m \times n) < O(m^2)$ due to $0 \leq n < m$.
2) To form B

$$\begin{aligned}
&\because \text{ because } 0 \leq n < m \text{ and } 0 \leq n_c < m \times (m-1)/2 \\
&\therefore O(m') \\
&= O(n + m + n \times n_c) \\
&< O(m + m + m \times m \times (m-1)/2) \\
&< O(2m + m^3/2 - m^2/2) \\
&< O(m^3/2 + m^3/2) \\
&= O(m^3) \text{ if } m \geq 2.
\end{aligned}$$

3) To form A

$$\begin{aligned}
&\because m' = n + m + n \times n_c, n' = m \times n, 0 \leq n < m \text{ and} \\
&\qquad 0 \leq n_c < m \times (m-1)/2 \\
&\therefore O(m' \times n' + m \times n' + m^2 + m^2) \\
&< O[(m + n + n \times m \times (m-1)/2) \times m \times n + m \\
&\qquad \times m \times n + 2m^2] < O(m^5) \text{ if } m \geq 3.
\end{aligned}$$

In summary, the time used to transform GRACAR to x-ILP is $O(m^5)$.

We can use the same technique to estimate that the time used to transform GRACAR to x-ILP is also $O(m^5)$.

Therefore, Theorem 6 is proved. ■

Note: Based on only Theorems 2–6, we may not state that GRACAR and GRACAG are NP-complete. However, based on our initial experiments [29], [30], GRACAR and GRACAG are much more difficult to solve than the GRA problem, of which an efficient solution is set out in [29] and [30]. We can state only that GRACAR and GRACAG are still OPEN problems based on the theory of Garey and Johnson [7], i.e., they may or may not have efficient (in polynomial-time) solutions.

VII. Practical Solutions

From the above theoretical analysis, we know that the GRACAR and GRACAG problems are subproblems of the x-ILP problem. If we find an approach that solves the x-ILP problem in an acceptable amount of time, the GARCAR and GARCAG problems will have practical solutions. Fortunately, ILOG is available to solve x-ILP problems.

We decided to solve the GRACAR and GRACAG problems by transferring them to x-ILP problems. Note that, usage of the ILOG package is different from usage of the optimization programming language (OPL) of the IBM ILOG CPLEX optimization studio [9]. Using the package by designing a Java program could obtain better performance by bypassing the OPL compiler.

A. Solve the GRACAR Problem

To accomplish the transferring task, we need the following two steps.

Step 1: Determine the four elements (i.e., objective function coefficients, constraint coefficients, right-hand side constraint values, and upper and lower bounds) required by the ILOG package, i.e., use vector L and matrices Q, A^c, and T to define an x-ILP problem in ILOG.
In this case, matrix Q expresses the objective function coefficients; and T the variables. The upper and lower bounds of T are 1 and 0.

Step 2: Add the objective and constraint expressions.
The objective of the GRACAR problem can be expressed by 1-D array forms of matrices Q and T. In the ILOG package, we can maximize this formula based on the objective.

$$\begin{bmatrix}
X_{00} & X_{01} & X_{02} & X_{03} \\
X_{10} & X_{11} & X_{12} & X_{13} \\
X_{20} & X_{21} & X_{22} & X_{23} \\
X_{30} & X_{31} & X_{32} & X_{33} \\
X_{40} & X_{41} & X_{42} & X_{43} \\
X_{50} & X_{51} & X_{52} & X_{53}
\end{bmatrix}$$

Fig. 8. Assignment matrix T.

For the constraint expressions, there are three general types of constraints in the GRACAR problem, i.e., constraints (2)–(4). Constraints (2) and (3) are a relatively straightforward transformation from a matrix to a 1-D array. Constraint (4) needs additional considerations. For each role, we must check all potential conflicts to make sure there are none in the assignment matrix T. Because ILOG can deal only with x-ILPs but not quadratic expressions, we must revise constraint (4) to satisfy requirements of ILOG. If $A^c[i_1, i_2] = 1$, then we add the constraint $T[i_1, j] + T[i_2, j] \leq 1$. It ensures that there is at least one zero in $T[i_1, j]$ and $T[i_2, j]$, i.e., only one of i_1 and i_2 is assigned to role j when agents i_1 and i_2 are in conflict.

Take Fig. 4 in Section IV as an example. First of all, we should initialize an ILOG object in Java by "IloCplex cplex = new IloCplex();" and add our optimization objective to it. The class IloCplex implements the interface of concert technology [9] which is used for creating variables and constraints. It also provides functionality for solving mathematical programing problems and accessing solution information. Next, we have to initialize assignment matrix T.

In Fig. 8, each $X_{ij} (0 \leq i < 6 \text{ and } 0 \leq j < 4)$ is a variable that can be assigned with a value of 1 or 0. In this case, T is declared in Java as

$$\text{IloIntVar[]}X = \text{cplex.intVarArray}(m * n, 0, 1)$$

where the first parameter indicates the size of matrix T in the 1-D array form (called a scalar in ILOG), the second and third parameters define the range of each variable within this array. Note that, ILOG does not support 2-D arrays. Actually X is a 1-D array corresponding to matrix T. Hence from the programming perspective, we have to match the positions of variables in X to those in matrix T. Obviously, our objective is to maximize the dot production of Q_a (the array form of Q) and X.

In this case

$$\begin{aligned} Q_a \cdot X = {} & 0.71X_{00} + 0.60X_{01} + 0.00X_{02} + 0.22X_{03} \\ & + 0.29X_{10} + 0.67X_{11} + 0.44X_{12} + 0.76X_{13} \\ & + 0.69X_{20} + 0.92X_{21} + 0.92X_{22} + 0.60X_{23} \\ & + 0.00X_{30} + 0.00X_{31} + 0.53X_{32} + 0.00X_{33} \\ & + 0.97X_{40} + 0.50X_{41} + 0.77X_{42} + 0.65X_{43} \\ & + 0.58X_{50} + 0.64X_{51} + 0.24X_{52} + 0.00X_{53}. \end{aligned}$$

To add the optimization objective, we invoke the following method in Java:

$$\text{cplex.addMaximize(cplex.scalProd}(X, Q_a)).$$

The next step is to add the constraints to ILOG. Constraint (2) is expressed as

$$\begin{aligned} X_{00} + X_{10} + X_{20} + X_{30} + X_{40} + X_{50} &= 2 \\ X_{01} + X_{11} + X_{21} + X_{31} + X_{41} + X_{51} &= 1 \\ X_{02} + X_{12} + X_{22} + X_{32} + X_{42} + X_{52} &= 1 \\ X_{03} + X_{13} + X_{23} + X_{33} + X_{43} + X_{53} &= 2. \end{aligned}$$

Constraint (3) is represented as

$$\begin{aligned} X_{00} + X_{01} + X_{02} + X_{03} &\le 1 \\ X_{10} + X_{11} + X_{12} + X_{13} &\le 1 \\ X_{20} + X_{21} + X_{22} + X_{23} &\le 1 \\ X_{30} + X_{31} + X_{32} + X_{33} &\le 1 \\ X_{40} + X_{41} + X_{42} + X_{43} &\le 1 \\ X_{50} + X_{51} + X_{52} + X_{53} &\le 1. \end{aligned}$$

Constraint (4) is

$$\begin{aligned} X_{00} + X_{40} &\le 1 \\ X_{10} + X_{20} &\le 1 \\ X_{20} + X_{40} &\le 1 \\ X_{03} + X_{43} &\le 1 \\ X_{13} + X_{23} &\le 1 \\ X_{23} + X_{43} &\le 1. \end{aligned}$$

To add these constraints to ILOG, we follow three steps.

1) Declare an expression object by calling

   IloLinearNumExpr expr = cplex.linearNumExpr().

2) Add all the terms to the object expr by invoking the method: expr.addTerm(...).
3) Add the constraint expression to ILOG by invoking: cplex.addEq(expr), cplex.addLe(expr) or cplex.addGe(expr) depending on the situation encountered.

After transferring the objective of GRACAR to the ILP form and adding all the constraints into ILOG, we can invoke method cplex.solve() to solve the problem. If the invocation succeeds, we can query the optimized objective value by method cplex.getObjValue() and retrieve the X values (that can be converted into matrix T) by double[] val = cplex.getValues(X). In this case, the objective is 4.15 and T is shown in Fig. 9 that is the same as T in Fig. 5(a).

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

Fig. 9. Assignment result of our solution with the ILOG package.

TABLE V
TEST PLATFORM CONFIGURATION

Hardware	
CPU	Intel core i5-2520M @2.5GHz × 4 cores
MEM	6GB DDR3@1333MHz
HDD	WD WD3200BEKT @7200 rpm
Software	
OS	Windows 7 Ultimate with SP1
Eclipse	Version: Indigo Service Release 2 Build id: 20120216-1857
JDK	1.7.0_05-b05 64Bit Server VM build 23.1-b03, mixed mode

B. Solve the GRACAG Problem

Most of this solution is similar to that of solving the GRACAR problem in Section VII-A. The major difference is to change constraint (4) to (5). For the example in Fig. 4, constraint (5) should be

$$\begin{aligned} X_{00} + X_{01} + X_{02} + X_{03} + X_{40} + X_{41} + X_{42} + X_{43} &\le 1 \\ X_{10} + X_{11} + X_{12} + X_{13} + X_{20} + X_{21} + X_{22} + X_{23} &\le 1 \\ X_{20} + X_{21} + X_{22} + X_{23} + X_{40} + X_{41} + X_{42} + X_{43} &\le 1. \end{aligned}$$

In fact, the solutions presented in this section are ILOG versions of the algorithms used to prove that a GRACAR or GRACAG problem is a subproblem of the x-ILP problem, i.e., Theorems 4 and 5.

VIII. COMPARISONS

To verify the proposed approach, we conduct comparisons between the proposed approach and the initial solutions in [29] and [30]. In Sections V and VI, we state that the initial solutions are not practical because they require considerable time to obtain results even for small groups. In this section, we confirm these theoretical analyses. The comparison experiment is conducted using the platform shown in Table V.

This experiment is conducted to compare the new solution with the exhaustive search solution. In each step, we repeat the test for 300 rounds. In each round, Q, A^c, and L are randomly generated. To compare the impact of the ratio of n/m on performance, we form two groups of tests whose n/m ratios are 1/2 and 1/3, respectively. To determine the influence of conflict rates, we use the p_c values of 10% and 25% for each group.

To generate a conflict matrix A^c based on p_c, we use the following simple algorithm (in Java-like format):

for i_1 **from** 0 **to** m-1 **do**
  for i_2 **from** i_1 **to** m-1 **do** {
    if (i_1 is equal to i_2) **then** $A^c[i_1, i_2] = 0$;
    else if (a random real number in [0, 1] $\le p_c$)

TABLE VI
COMPARISON BETWEEN THE ILOG SOLUTION AND THE EXHAUSTIVE SEARCH

m	*n*	Conflict rate	Solution with ILOG (ms)			Exhaustive search Solution (ms)		
			Ave	Max	Min	Ave	Max	Min
10	5	0.25	12.42	260.41	4.31	201.38	20137.85	77.01
20	10	0.25	17.45	128.89	9.83	N/A[a]	N/A	N/A
40	20	0.25	69.55	439.75	41.65	N/A	N/A	N/A
10	5	0.1	6.85	98.81	3.85	312.52	797.27	88.22
20	10	0.1	15.49	106.40	6.67	N/A	N/A	N/A
40	20	0.1	40.98	328.87	24.06	N/A	N/A	N/A
10	3	0.25	7.25	92.07	4.18	22.35	136.29	9.16
20	6	0.25	15.55	125.07	6.48	N/A	N/A	N/A
40	13	0.25	49.06	227.18	28.52	N/A	N/A	N/A
10	3	0.1	7.64	93.19	3.76	17.82	112.13	2.58
20	6	0.1	11.78	102.93	4.89	N/A	N/A	N/A
40	13	0.1	29.62	165.48	12.21	N/A	N/A	N/A

[a]N/A means no optimized result in 30 minutes.

then { $A^c[i_1, i_2] = 1; A^c[i_2, i_1] = 1;$ }
else { $A^c[i_1, i_2] = 0; A^c[i_2, i_1] = 0;$ }
}

In each round of testing, we record times used by the random cases, then determine the maximum, minimum, and average times for the round. In this manner, we hope to discover performance trends for each solution. Unfortunately, the exhaustive search solution can only provide results in a reasonable time when m equals 10. When m grows to 20, regardless of the conflict rate or the n/m ratio is, time values are unacceptable. Tests are terminated after 30 min without obtaining a result. Comparable results occur only when m equals 10. Table VI indicates the test results with different configurations of $<m, n, p_c>$.

From Table VI, we find that the performance of the new solution with ILOG has an obvious relationship with problem size m. In general, smaller problems require less time. In contrast, from these limited test results, the exhaustive search solution does not have a clear relationship between performance and conflict rate, e.g., when m equals 10 and n equals 5, the average solution times are 201.38 and 312.52 ms while the conflict rates are 0.25 and 0.10, respectively. However, m values affect the average times in both cases. Smaller n values always take less time. For instance, in the case of $m = 10$ and conflict rate $= 0.25$, when $n = 5$, the average time is 201.38 ms; and when $n = 3$ the average time drops to 22.35 ms. A similar situation exists when the conflict rate is 10%.

IX. PERFORMANCE EXPERIMENTS

To illustrate the performance and the limit of the proposed solution using ILOG, we make performance experiments using different testing configurations. These experiments use the same platform as shown in Table V. As for the limit of the solution with ILOG, we conduct tests (called limitation tests) by initializing m to 50 which is increased by 50 after each step, and the upper bound of m is set to 1000. For each step, we test 300 rounds. For each round, we randomly generate a qualification matrix Q, a role range vector L and a conflict matrix A^c with a fixed conflict rate of 25%. The results are illustrated in Fig. 10.

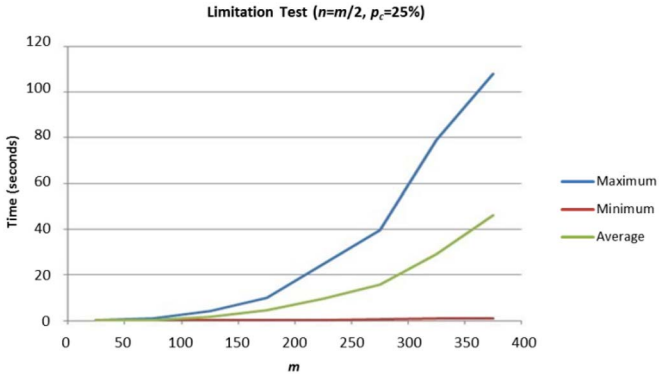

Fig. 10. Limitation test.

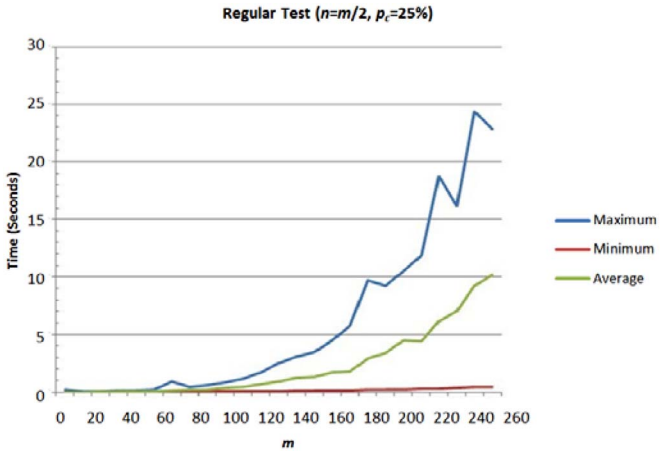

Fig. 11. Regular test.

During the process of this experiment, a "limit exceeded" error occurred while initializing the linear expression at step $m = 450$ ($n = m/2$). This is a limitation of ILOG. Hence, to get a reasonable running time without an error, ILOG restricts the number of variables in assignment matrix T to fewer than 10 000.

In order to understand the growth trend of elapsed time, we set up another test with random Qs, A^cs, and Ls. The scales of Qs, A^cs, and Ls are based on the number of agents (m) and roles (n) which grows after each step by a predefined number. In each step, we test 300 rounds to get a more accurate result. To compare the influence of all parameters, we conduct three tests: 1) regular test: we test the solution with a normal setting that is called a regular configuration; 2) regular test with a different ratio of n/m: we change the ratio n/m and keep all other parameters unchanged from 1); and 3) regular test with a different conflict rate: we change the conflict rate but keep all other parameters unchanged from 1).

For the regular test, the conflict rate is 25% and the test range of m is from 10 to 250 with a growing interval of 10 after each step, and n equals to $m/2$. In each step, we repeat the test for 300 rounds, and then determine the maximum, minimum and average time. Results are shown in Fig. 11.

From Fig. 11, we can observe that the maximum and average times apparently increase while the size of a problem grows. In contrast, the minimum times increase from 3 to 453 ms where m equals to 10 and 250, respectively.

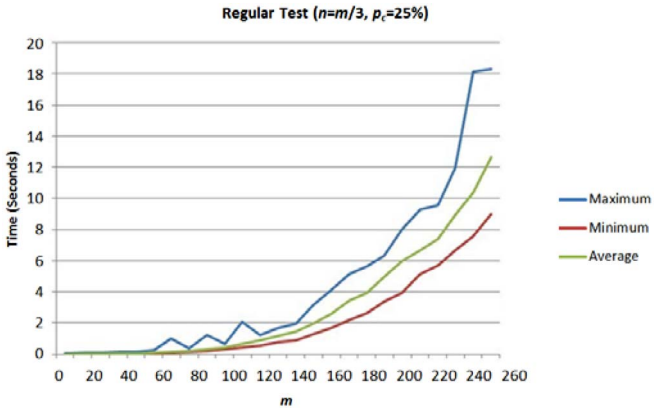

Fig. 12. Regular test with a different ratio of n/m.

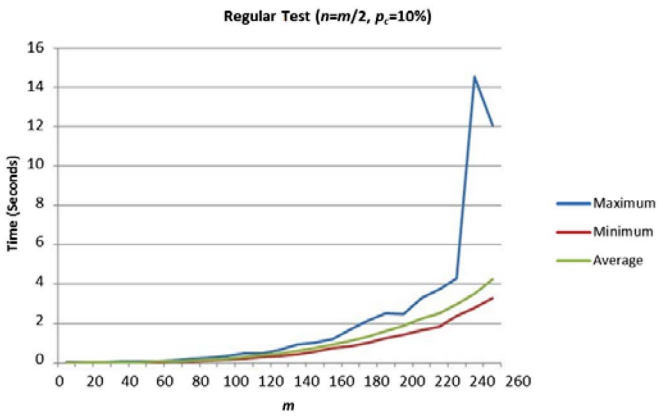

Fig. 13. Regular test with a different conflict rate.

Also in most cases, average times approximately equal the mid-value over the minimum–maximum range. This is considered reasonable due to the random numbers in all the cases used.

Results for the regular test, where the n/m ratio changes from 1/2 to 1/3, are shown in Fig. 12. Compared with Fig. 11, it is very clear that differences between maximum, average, and minimum times are reduced in Fig. 12. The trend of the maximum time is almost the same as that in the regular test but values in this test are obviously smaller. In general, the time value decreases while the n/m ratio decreases.

Fig. 13 indicates results obtained when the conflict rate changes from 25% to 10% while all other regular test parameters remain the same.

In Fig. 13, it is evident that the values of average and maximum times are much smaller than those in Figs. 10 and 11, even when a rapid increase in the maximum time occurs at $m = 230$. The performance of this test is better than that of the regular test due to a much smaller average time.

From Figs. 10–13, one can observe a fluctuation and sharp drop in the maximum time curve. In fact, the maximum time curve indicates values used by one specific case out of 300 random groups of different sizes. The fluctuations and drops in the maximum curve mean that solving a problem of a group with a larger m (e.g., 250) may use less time than that of a group with a smaller m (e.g., 240). This is possible because they are different groups with different configurations including m, n, L, Q, and A^c. Note that, this fluctuation does not affect the overall trend. The average time curve is smoother and may demonstrate the efficiency of the algorithm more appropriately.

In summary, our experiments suggest that the size of a problem is less than 200 ($m = 200$). Otherwise, the solution time is possibly greater than 10 s, which is not practical in applications. Also, the limit of 10 000 variables in the current ILOG package indicates a similar scale limitation. A new issue, beyond the scope of this paper, is to investigate the impact of parallel processing in solution performance.

X. Related Work

Role assignment affected by agent conflicts is an important and challenging problem in organizational performance [4], system construction [26], [28] and system management [4]. To the author's knowledge, there is a lack of fundamental and comprehensive research on this issue. Some related research work focuses on role assignment for agents in multiagent systems [4], [6], [28] or nodes in networked systems [1].

Bhardwaj and Chandrakasan [1] presented a real-world application that requires role assignment. They introduce the role assignment framework and use it to derive bounds of the lifetime of a sensor network for a variety of data gathering scenarios that can be used to promote general role assignment algorithms.

Dastani *et al.* [4] presented research on the determination of conditions under which an agent could adopt a role and what it means for the agent to perform a role. They define possible relationships between roles and agents then discuss architectural and functional changes that an agent must face when it enters an open agent system. Their work in fact proposes a framework to solve part of the first step of role assignment discussed in the introduction.

Durfee *et al.* [5] proposed a new formulation of the team formation by modeling the assignment and scheduling of expert teams as a hybrid scheduling problem. Their work demonstrates the significance and complexity of the problem of assignment and scheduling in expert teams. It also demonstrates that the assignment and scheduling problem in expert teams can be transferred into an ILP or a mixed ILP problem and that it is possible for a team formation problem to be solved by a standard mathematical programming package such as ILOG used in this paper.

Nyanchama and Osborn [16] described a role-graph model for role-based access control. In discussing the model, they clarify the roles in conflict. They use the graph model to provide taxonomy for conflict types. They simplify the complex problem of role assignment in consideration of role conflict by partitioning the role graph into nonconflicting collections that can together be assigned to an agent.

Odell *et al.* [17] pointed out that the roles played by an agent might change over time. They describe a case study where such role changes are required, analyze and clarify the various kinds of role changes that may occur. Their contribution focuses on the third step of role assignment, i.e., role transfer.

Shen *et al.* [22] proposed a multicriteria assessment model capable of evaluating the suitability of individual workers for a specified task according to their capabilities, social relationships, and existing tasks. Candidates are ranked based on their suitability scores to support workflow administrators in selecting appropriate workers to perform the tasks assigned to a given role. Their work can be applied into making the Q matrix for GRA.

Stone and Veloso [23] introduced periodic team synchronization domains as time-critical environments in which agents acted autonomously. They point out that dynamic role adjustment (the third step, i.e., role transfer [16], [32], [33]) makes possible formation changes allowing for a group of agents to collaborate. They verify their method using a soccer team of robots.

Tessier *et al.* [24] presented an overview of conflicting agents in the context of multiagent systems. They define conflict based on propositional attributes. They also describe the categorization of human conflict handling methods, i.e., flight, destruction, subservientness, delegation, compromise, and consensus. Finally, they propose a conflict handling action model including recognition, diagnosis, avoidance, resolution, management, adaptation, belief update, and process analysis. In fact, our proposed solution is one way of conflict avoidance in their model or it can be considered as one instance of the "flight" method in human conflict handling.

Vail and Veloso [26] extended Stone and Veloso's work [23] 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. Such fields are based on positions (roles) of other robots on the team and the ball. Robots position themselves on a field by following the gradient to a minimum of the potential field.

Zhang [28] proposed a teamwork language role-based multiagent logic language for encoding teamwork 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 objective is to provide a technique for task assignment based on decision trees.

The aforementioned research indicates a strong need to fundamentally investigate GRAs including the problems discussed in this paper.

XI. Conclusion

Conflict avoidance during GRA is a complex problem. We first formalize the GRACAR and GRACAG problems. Next, we verify the benefit of avoiding conflicts through simulations. We also prove that they are actually subproblems of the x-ILP problem that is NP-complete. After that, we propose a practical approach to solving these problems with the IBM ILOG optimization package. Experiments indicate that this proposed solution performs well enough for a relatively large set of instances of the problem ($m = 200$). In fact, our solutions to such abstract problems can be applied to many different fields with similar constraints, such as industrial engineering, business management, service computing, cloud computing, and multiagent systems.

From this paper, it is clear that further investigations may be required along the following directions.

1) Although the proposed algorithm improves initial solutions, there are still opportunities to improve it. Because the proposed problem has not been assured as NP-complete, it is still possible to find an efficient solution, i.e., finding the necessary condition to make GRACAR and GRACAG solvable may help upgrade the efficiency of a solution.
2) In theory, we may try to prove GRACAR and GRACAG are NP-complete to assure that there are no efficient algorithms.
3) There may be additional constraints that require further investigation, e.g., we may need to consider a vector V with constraint $\forall(0 \leq i < m) \sum_{j=0}^{n-1} T[i,j] \leq V[i])$.
4) It may be valuable to investigate the opposite views, i.e., if there are too many constraints, a simple solution may be obtained by reorganizing teams or groups.
5) For better practical applications, nonuniform contributions of a group may be worthy of investigation, i.e., different roles or role groups may have different combined performance requirements.

Acknowledgment

The author would like to thank Dr. T. Vassilev of Nipissing University for his help in assuring the expression of ILP problems, Y. Sheng of Nanjing University for his discussion with the author concerning the proposed problems, and M. Brewes of Nipissing University and Y. Zhu of the University of Waterloo for their assistance in proofreading this paper. He would also like to thank the anonymous reviewers for their valuable comments in revising this paper.

References

[1] 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. (INFOCOM)*, vol. 3. New York, NY, USA, Jun. 2002, pp. 1587–1596.

[2] M. Bristow, L. Fang, and K. W. Hipel, "Agent-based modeling of competitive and cooperative behavior under conflict," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 7, pp. 834–850, Jul. 2014.

[3] B. Canbaz, B. Yannou, and P.-A. Yvars, "Resolving design conflicts and evaluating solidarity in distributed design," *IEEE Trans. Syst, Man, Cybern., Syst.*, vol. 44, no. 8, pp. 1044–1055, Aug. 2014.

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

[5] E. H. Durfee, J. C. Boerkoel, Jr., and J. Sleight, "Using hybrid scheduling for the semi-autonomous formation of expert teams," *Future Gener. Comput. Syst.,* vol. 31, pp. 200–212, Feb. 2014.

[6] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: An organizational view of multi-agent systems," in *Agent-Oriented Software Engineering IV* (LNCS 2935), P. Giorgini, J. P. Müller, and J. Odell, Eds. Berlin, Germany: Springer, 2004, pp. 214–230.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[8] W. Guo, W. M. Healy, and M. C. Zhou, "Impacts of 2.4-GHz ISM band interference on IEEE 802.15.4 wireless sensor network reliability in buildings," *IEEE Trans. Instrum. Meas.,* vol. 61, no. 9, pp. 2533–2544, Sep. 2012.

[9] IBM. (2013). *ILOG CPLEX Optimization Studio*. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/
[10] D. M. Kilgour and K. W. Hipel, "The graph model for conflict resolution: Past, present, and future," *Group Decis. Negotiat.,* vol. 14, no. 6, pp. 441–460, Nov. 2005.
[11] R. A. Kinsara, D. M. Kilgour, and K. W. Hipel, "Inverse approach to the graph model for conflict resolution," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 5, pp. 734–742, May 2015.
[12] M. Klein, "Supporting conflict resolution in cooperative design systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 6, pp. 1379–1390, Dec. 1991.
[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart.*, vol. 2, nos. 1–2, pp. 83–97, 1955.
[14] T. Malsch and G. Weiss, "Conflicts in social theory and multi-agent systems," in *Conflicting Agents*, C. Tessier, L. Chaudron, and H.-J. Müller, Eds. New York, NY, USA: Springer, 2002, pp. 111–149.
[15] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.
[16] M. Nyanchama and S. Osborn, "The role graph model and conflict of interest," *ACM Trans. Inf. Syst. Security*, vol. 2, no. 1, pp. 3–33, Feb. 1999.
[17] 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.
[18] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, Oct. 1981.
[19] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. New York, NY, USA: McGraw-Hill, 2004.
[20] R. L. Rardin, *Optimization in Operations Research.* Upper Saddle River, NJ, USA: Prentice Hall, 1997.
[21] A. Schrijver, *Theory of Linear and Integer Programming*. Chichester, U.K.: Wiley, 2000.
[22] M. Shen, G.-H. Tzeng, and D.-R. Liu, "Multi-criteria task assignment in workflow management systems," in *Proc. 36th Annu. Hawaii Int. Conf. Syst. Sci.*, Big Island, HI, USA, Jan. 2003, pp. 1–9.
[23] 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, 1999.
[24] C. Tessier, H. J. Müller, H. Fiorino, and L. Chaudron, "Agents' conflicts: New issues," in *Conflicting Agents*, C. Tessier, L. Chaudron, and H.-J. Müller, Eds. New York, NY, USA: Springer, 2002, pp. 1–30.
[25] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.
[26] 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. Boston, MA, USA: Kluwer Academic, 2003, pp. 87–98.
[27] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1999.
[28] Y. Zhang, "A role-based approach in dynamic task delegation in agent team work," *J. Softw.*, vol. 3, no. 6, pp. 9–20, 2008.
[29] H. Zhu, "Role assignment for an agent group in consideration of conflicts among agents," in *Proc. Can. Conf. Artif. Intell.,* Toronto, ON, Canada, May 2012, pp. 267–279.
[30] H. Zhu, "Group role assignment with conflicting agent constraints," in *Proc. ACM/IEEE Int. Symp. Coll. Technol. Syst.,* Philadelphia, PA, USA, 2011, pp. 431–439.
[31] H. Zhu and R. Alkins, "Group role assignment," in *Proc. Int. Symp. Coll. Technol. Syst.*, Baltimore, MD, USA, May 2009, pp. 431–439.
[32] H. Zhu and L. Feng, "An efficient approach to group role assignment with conflicting agents," in *Proc. IEEE Int. Conf. Comput.-Support. Cooper. Work Design (CSCWD)*, Whistler, BC, Canada, Jun. 2013, pp. 145–152.
[33] H. Zhu and M. C. Zhou, "Efficient role transfer based on Kuhn–Munkres algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans,* vol. 42, no. 2, pp. 491–496, Mar. 2012.
[34] H. Zhu and M. Zhou, "M–M role-transfer problems and their solutions," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 2, pp. 448–459, Mar. 2009.
[35] 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.
[36] H. Zhu, M. C. Zhou, and R. Alkins, "Group role assignment via a Kuhn–Munkres algorithm-based solution," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans,* vol. 42, no. 3, pp. 739–750, May 2012.

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 and the Founder and the Director of Collaborative Systems Laboratory, Nipissing University, North Bay, ON, Canada, and a Visiting Professor with the Department of Control Science and Engineering, Nanjing University, Nanjing, China. He was a Visiting Professor and a Special Lecturer with the College of Computing Sciences, New Jersey Institute of Technology, Newark, NJ, USA, from 1999 to 2002 and a Lecturer, an Associate Professor, and a Full Professor with NUDT from 1988 to 2000. He has published 140+ research papers, four books, and two book chapters on object-oriented programming, distributed systems, collaborative systems, and computer architecture.

Dr. Zhu was a recipient of the 2011 Chancellors' Award for Excellence in Research and the 2006–2007/2011–2012 Research Achievement Award from Nipissing University, the 2004 and 2005 IBM Eclipse Innovation Grant Award, the Best Paper Award from the ISPE International Conference on Concurrent Engineering in 2004, the Educator's Fellowship of OOPSLA'03, the Second Class National Award of Education Achievement from the Ministry of Education of China in 1997, the Second Class National Award of Excellent Textbook from the Ministry of Education of China in 2002, three First Class Ministerial Research Achievement Awards from the Commission of Science Technology and Industry for National Defense of China in 1991, 1994, and 1997, and the Second Class Excellent Textbook Award of the Ministry of Electronics Industry of China in 1996. He is serving and has served as Co-Chair of the Technical Committee of Distributed Intelligent Systems of IEEE SMC Society, an Associate Editor for the IEEE SYSTEMS, MAN, AND CYBERNETICS MAGAZINE and the *International Journal of Agent Technologies and Systems*, an Associate Editor-in-Chief for the *International Journal of Advances in Information and Service Sciences*, a Managing Editor for the *International Journal of Electrical, Electronics and Computing Technology, Computer Sciences*, an Editorial Board Member of the *International Journal of Software Science and Computational Intelligence*, a Guest Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, an organizer of the workshops and special sessions on role-based collaboration for over ten international conferences, and a Program Committee Member for over 40 international conferences. He is a member of ACM and a Life Member of the Chinese Association for Science and Technology, USA.