

Group Role Assignment

Haibin Zhu, *Senior Member, IEEE* and Rob Alkins
Nipissing University, Canada
haibinz@nipissingu.ca

ABSTRACT

Role assignment is an important task in Role-Based Collaboration (RBC). The process can be divided into three steps, i.e., agent evaluation, group role assignment, and role transfer. This paper formally identifies various group role assignment problems under acceptable assumptions; proposes solutions based on exhaustive-search; conducts experiments by randomly creating agent qualifications; analyzes the solutions' performance based on experimental results, and indicates future work activities.

KEYWORDS: roles, agents, role assignment, and role-based collaboration.

1. INTRODUCTION

Role-Based Collaboration (RBC) is an emerging methodology to facilitate an organizational structure, provide orderly system behavior, and consolidate system security for both human and non-human entities that collaborate and coordinate their activities with or within systems [12]. The life cycle of RBC includes three major tasks: *role negotiation, assignment and performance* [13]. 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.

To be better understood, role assignment can be categorized into three parts: *agent evaluation, group role assignment, and role transfer*. *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. In the business world, a person is required to acquire new knowledge, skills and habits when qualifying for a new position. Success in finding a new job may depend on similarities between new roles and those previously performed [3], i.e., qualifications are the basic requirements for possible role-related activities. It is a fundamental yet difficult problem that requires advanced methodologies, such as information

classification, data mining, pattern searching, and matching, to produce high-quality evaluations. *Group role assignment* initiates a group by assigning roles to agents, the group members, to obtain the group's highest performance based on the agents' qualifications which are the results of agent evaluation. *Role transfer* (also called *dynamic role assignment*) re-assigns roles to agents or transfers agent roles [14] to meet the requirement of the system changes. Agents' role transfer needs to maintain the following properties:

- Fairness: Starvation or overloading should be avoided, where starvation means that an agent has not received role assignments for a time longer than a limit while overloading means that an agent is assigned too many roles in a limited time.
- Balanced workload: Agents should be transferred to roles according to the workload of agents.
- Least conflict: Agents should be transferred to roles with the least opportunity for agents to have conflicts in sharing information when they are playing roles.

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

The remainder of this paper is arranged as follows: Section 2 discusses related work; Section 3 revises the E-CARGO model in order to formalize the problem of role assignment; Section 4 depicts the simple role assignment problem; Section 5 demonstrates the rated role assignment problem; Section 6 illustrates the weighted role assignment problem; Section 7 presents the experiments of the implemented algorithms and shows

the performances of the algorithms; and Section 8 concludes this paper and points out topics for future work.

2. RELATED RESEARCH

Although role assignment is evidently an important problem in management [4], organizational behavior and performance [1, 3, 4], system construction [2], scheduling, training and commanding [9], there is no fundamental research on role assignment theory and algorithms. Some related research concerns agent systems and wireless communications [5, 7, 10, 11] concentrating on their special cases without generalizations.

Dastani et al. [5] present a research on the determination of the conditions under which an agent can enact a role and what it means for the agent to enact a role. They define possible relations between roles and agents and discuss architectural and functional changes that an agent must undergo 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.

Odell et al. [7] point out that the roles played by an agent may change over time. They describe a case study where such role changes are required, analyzing and classifying the various kinds of role changes that may occur. Their contribution focuses on the third step of role assignment, i.e., role transfer.

Stone and Veloso [10] introduce periodic team synchronization (PTS) domains as time-critical environments in which agents act autonomously. They point out that dynamic role adjustment (the third step, i.e., role transfer [14]) makes possible formation changes allowing for a group of agents to collaborate. They apply their method to a robot soccer team and obtain a convincing result.

Vail and Veloso [11] extend Stone and Veloso's work in role assignment and coordination in multi-robot 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 were based on the positions (roles) of the other robots on the team and the ball. The robots positioned themselves on the field by following the gradient to a minimum of the potential field.

The above research indicates a strong need to fundamentally investigate role assignment problems and their solutions. Their work also demonstrates the importance of the work depicted in this paper, i.e., there is

insufficient consideration of the second step in role assignment.

3. REVISED E-CARGO MODEL

With the E-CARGO model [13], collaboration is based on roles. In E-CARGO, a system Σ can be described as a 9-tuple $\Sigma ::= \langle C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H} \rangle$, where 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 a collaborative 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 play a role together. Every group should work in an environment. An environment regulates a group. With this tight coupling, it is important to consider that a role-based collaborative system is composed of both computers and human beings.

With the participation of people \mathcal{H} , such as joining in a team Σ , accessing objects of the team, sending messages through roles, forming a group in an environment, Σ evolves, develops and functions. The results of the team work are a new state of Σ that is expressed by the values of C , O , \mathcal{A} , \mathcal{M} , \mathcal{E} , \mathcal{G} , and \mathcal{H} . In E-CARGO, agents are considered to have only one central processing unit and each agent can play only one role at a time. To formally state related concepts, these notations require some initial clarification. If S is a set, $|S|$ is its cardinality; suppose v is a variable, $v \rightarrow S$ means v may take an element of S as its value; suppose a and b are objects, $a.b$ means b of a or a 's b and $\{a, b, \dots\}$ means a set of enumerated elements of a , b , and others; suppose a and b are integers, $[a, b]$ and $(a, b]$ mean the set of all the real number between a and b including a and excluding a respectively; suppose \mathcal{Y} is a vector, $\mathcal{Y}[a]$ means the element at a ; suppose \mathcal{Y} is a matrix, $\mathcal{Y}[a, b]$ means the element at row a and column b in \mathcal{Y} . \mathcal{E} , C , \mathcal{A} , \mathcal{R} , \mathcal{G} , O and \mathcal{M} denote the whole set of environments, classes, agents, roles, groups, objects and messages, respectively. The definitions of the components of E-CARGO are restated as follows.

Definition 1: *role*. A role is defined as $r ::= \langle n, I, \mathcal{A}_c, \mathcal{A}_p, \mathcal{A}_o, \mathcal{R}_x, o_r \rangle$ where,

- n is the identification of the role;
- $I ::= \langle \mathcal{M}_{in}, \mathcal{M}_{out} \rangle$ denotes a set of messages, where \mathcal{M}_{in} expresses the incoming messages to the relevant agents, and \mathcal{M}_{out} expresses a set of outgoing messages or message templates to roles, i.e., $\mathcal{M}_{in}, \mathcal{M}_{out} \subset \mathcal{M}$;
- \mathcal{A}_c is a set of agents who are currently playing this role;

- \mathcal{A}_p is a set of agents who are potential to play this role;
- \mathcal{A}_o is a set of agents who used to play this role;
- \mathcal{R}_c is a set of roles interrelated with it; and
- o_r is a set of objects that can be accessed by the agents playing this role.

Definition 2: agent. An agent is defined as $a ::= \langle n, c_a, s, r_c, \mathcal{R}_p, \mathcal{R}_o, \mathcal{N}_g \rangle$, where

- n is the identification of the agent;
- c_a is a special class that describes the common properties of users;
- s is the qualification value of the agent;
- r_c means a role that the agent is currently playing. If it is empty, then this agent is free;
- \mathcal{R}_p means a set of roles that the agent is potentially to play ($r_c \notin a.\mathcal{R}_p$); and
- \mathcal{R}_o means a set of roles that the agent played before; and
- \mathcal{N}_g means a set of groups that the agent belongs to.

All the current role and the potential roles of agent a (i.e., $a.\mathcal{R}_p \cup \{a.r_c\}$) form its repository role set, denoted as \mathcal{R}_e .

Definition 3: environment. $e ::= \langle n, \mathcal{R}_e, \mathcal{B}, o_g \rangle$ where

- n is the identification of the environment;
- \mathcal{R}_e is a set of roles;
- \mathcal{B} is a set of tuples of role, role range and a shared object, i.e., $\langle r, w, q, o_e \rangle$, where $r \in \mathcal{R}_e$. The weight $w \rightarrow [0, 1]$ is to show the importance of the role. The role range (also called cardinalities) q is expressed by $[[l, u]]$ and tells how many agents must (l) and may (u) play this role in this environment. The o_e expresses the object accessed or shared by all the agents playing role r in the tuple. \mathcal{T} denotes the whole set of role ranges; and
- o_g expresses the object shared by all the agents in a group built on this environment.

Definition 4: group. $g = \langle n, e, \mathcal{A}_g, J \rangle$ where

- n is the identification of the group;
- e is an environment for the group to work;
- \mathcal{A}_g is a set of agents called members of this group; and
- J is a set of tuples of agent and role, i.e., $J = \{ \langle a, r \rangle \mid r \in e.\mathcal{R}_e \wedge (\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \exists \langle r, w, q, o_e \rangle \in e.\mathcal{B}) \wedge (a \in r.\mathcal{A}_r \cup r.\mathcal{A}_p) \}$.

With the E-CARGO model, a role engine can be designed to support RBC in the way stated by Shakespeare “All the

world is a stage ($\mathcal{E}, \mathcal{C}, \mathcal{O}$), and all the men and women merely players (\mathcal{A}); they all have their exits and entrances (\mathcal{G}); and one man in his time plays many parts (\mathcal{R}) (As You Like It, Act II, Scene 7)”.

Definition 5: role assignment. For a group g , a tuple $\langle a, r \rangle$ of $g.J$ is called a *role assignment*, also called *agent assignment*.

Note that, in role assignment, we do not differentiate current roles and potential roles [14].

Definition 6: workable role. A role r is *workable* in an environment e if it is assigned enough current agents to play it, i.e., $\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \exists \langle r, w, q, o_e \rangle \in e.\mathcal{B} \wedge (|r.\mathcal{A}_c| \geq q.l)$.

In formalizing role assignment problems, only the relations between agents and roles are important. In the following discussions, current agents or roles are concentrated on and environments and groups are simplified by matrices and vectors. In describing the problems, \mathcal{A} denotes the set of agents and $m (=|\mathcal{A}|)$ the size of the set \mathcal{A} . \mathcal{R} denotes the set of roles and $n (=|\mathcal{R}|)$ the size of \mathcal{R} . L denotes a vector of the lower ranges of roles in an environment e and $L[j] \rightarrow \mathcal{N}$ (\mathcal{N} is the natural number set, $0 \leq j \leq n-1$). Q denotes an *agent qualification (s) matrix* of $m \times n$. T denotes a *role assignment matrix* of $m \times n$ and $T[i, j] \rightarrow \{0, 1\}$ ($0 \leq i \leq m-1; 0 \leq j \leq n-1$). If $T[i, j]=1$, agent i is called an *assigned agent*.

Definition 7: workable group. A group g is *workable* if all its roles are workable, i.e., $\forall r \in g.e.\mathcal{R}_e (\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \exists \langle r, w, q, o_e \rangle \in g.e.\mathcal{B} \wedge (\text{workable}(r) = \text{True}))$, or group g expressed by T is *workable* if $\forall j (\sum_{i=1}^m T[i, j] \geq L[j])$ ($0 \leq j \leq n-1$).

4. THE ROLE ASSIGNMENT PROBLEM

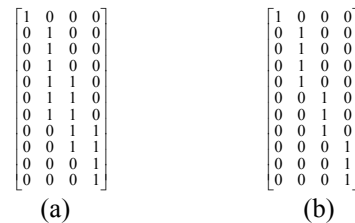


Figure 1. The Capability Matrix and One Solution

Definition 8: simple group role assignment problem. Let $Q[i, j] \rightarrow \{0, 1\}$ ($0 \leq i \leq m-1; 0 \leq j \leq n-1$) be the value that represents if a given agent i is able to play a given role j , 0 means no and 1 yes. The weights of roles are

ignored. The problem is to find a role assignment matrix T that makes g workable.

For example, suppose Q is shown as Figure 1 (a) and L = [1, 4, 3, 3]. One solution T for the role assignment problem is as shown in Figure 1 (b).

A recursive algorithm for the role assignment can be designed as follows. The basic idea is to try all the possibilities of assigning enough agents to each role. The algorithm **Assign**(L, Q, T, m, n) is described by a java-like language as follows. In describing the algorithms, “:=” means assignment from right to left and “=” means the left is equivalent to right.

Input:

- An n dimensional role’s lower range vector L; and
- An m×n qualification matrix Q.

Output:

- **Success:** An m×n assignment matrix T in which for all columns j (j= 0, ..., n-1), $\sum_i T[i, j] \geq L[j]$.
- **Failure:** An m×n assignment matrix T in which there is at least one column j (j= 0, ..., n-1), $\sum_i T[i, j] < L[j]$.

Assign(L, Q, T, m, n)

{

Step 1: Initialization.

result := failure; T:={};
for (all roles (j)) M[j]= $\sum_{i=0}^{m-1} Q[i, j]$;

for (all roles (j)) if (M[j]<L[j]) return result;

Step 2: Build combinations of agent assignment to the first role (role 0).

p := L[0];
k := M[0];

Create k dimensional vector A where each element is a qualified agent for role 0.

s := the number of combinations taking p from k. i.e.,

$$\binom{k}{p} = k!/(p!(k-p)!)$$

Form an s×p matrix V from A (refers to the combinatory generator [2]), where each row is a set of qualified agents. The rows in V are all the combinations of taking p (≤k) from k qualified agents.

Step 3: Try each combination of agent assignment to the first role.

for (each combination (x))

{

Step 3.1: Assign agents.

for (each qualified agent (y) in this combination)

{Assign agent V[x, y] to role 0, i.e., T[V[x, y], 0]:= 1;}

Step 3.2: Prepare for a smaller problem.

Now, role 0 is workable and all its agents should not be considered as candidates to assign in the next recursion;

Q’ := Q; Delete the rows expressed by V[x] and the column 0 from Q;

L’ = L; Delete item 0 in L’;

Now, the number of agents becomes m-p and the number of roles becomes n-1;

Step 3.3: Solve the smaller problem.

result := **Assign**(L’, Q’, T’, m-p, n-1);

Step 3.4: Check the solution of the smaller problem.

if (The inner recursion is successful, i.e., result =success)

{Combine the assignment done in the inner recursion with the assignment at this level, i.e., Q := Q’ and T := T’ by keeping the rows in V[x] and column 0 unchanged;

L := L’ by keeping item 0;

Now, the role assignment at this level is successful; return success;

}

}/end for (each combination)

return result;

}

The correctness of the above algorithm comes from the following facts:

- 1) Every role is considered by recursions.
- 2) Every role assignment combination is considered.
- 3) If there exists a role that has no assignment to make it workable, the algorithm fails.
- 4) After a role is assigned with agents and becomes workable, the problem becomes a smaller one.
- 5) If the smaller problem is solved, the whole problem is solved.
- 6) If the smaller problem is not solved, the assignment in 4) is not accepted and a new assignment is tried.

Because this algorithm is recursive and has process of obtaining combinations, its complexity is high. For the worst cases, if **Step 3** is concentrated on and

$$\sum_{i=0}^{n-1} L[i] = 0, \text{ the complexity is } O\left(\prod_{j=0}^{n-1} \binom{L[j]}{m - \sum_{i=0}^{j-1} L[i]}\right) =$$

$$O\left(\frac{m!}{L[0]!(m-L[0])!} \times \frac{(m-L[0])!}{L[1]!(m-L[0]-L[1])!} \times \dots \times \frac{m-L[0]-L[1]-\dots-L[n-2]}{L[n-1]!(m-L[0]-L[1]-\dots-L[n-2]-L[n-1])!}\right)$$

$$= \frac{m!}{L[0]!L[1]!\dots L[n-1]!L[n-1]!(m-L[0]-L[1]-\dots-L[n-2]-L[n-1])!}$$

$$= O\left(\frac{m!}{\prod_{j=0}^{n-1} L[j]! (m - \sum_{i=0}^{n-1} L[i])!}\right).$$

Suppose there are just enough agents for a group, i.e.,

$$m = \sum_{i=0}^{n-1} L[i], \text{ the complexity expression can be simplified as } O\left(\frac{m!}{\prod_{j=0}^{n-1} L[j]!}\right).$$

5. THE RATED ROLE ASSIGNMENT PROBLEM

If the conditions of role assignment are changed, a rating problem occurs. The assumption is that most agents are not absolutely qualified to play a role but with a rate from 0 to 1 and the roles have the same importance. In this situation, group role assignment does not follow that the most qualified agent with the highest qualification value is assigned to a role. It follows that the overall sum of qualifications for all group agents is a maximum. This is a typical optimization problem.

Definition 9: *rated group role assignment problem.* Let $Q[i, j] \rightarrow [0, 1]$ ($0 \leq i \leq m-1$; $0 \leq j \leq n-1$) be the value that represents how well a given agent i plays a given role j , 0 means lowest and 1 the highest. The weights of roles are ignored. A *group qualification* is defined as the sum of the assigned agents' qualifications, i.e., $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] * T[i, j]$. The problem is to find a matrix

T that makes group g in which the group qualification is the largest, i.e., $\max\left\{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] * T[i, j]\right\}$.

| | |
|--|---|
| $\begin{bmatrix} 0.71 & 0.6 & 0.0 & 0.22 \\ 0.29 & 0.67 & 0.44 & 0.76 \\ 0.69 & 0.92 & 0.92 & 0.6 \\ 0.0 & 0.0 & 0.53 & 0.0 \\ 0.97 & 0.51 & 0.77 & 0.65 \\ 0.58 & 0.64 & 0.24 & 0.0 \end{bmatrix}$ <p>(a)</p> | $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ <p>(b)</p> |
|--|---|

Figure 2. A Rated Qualification Matrix and the Solution.

For example, suppose Q is shown in Figure 2 (a), $L = [2, 1, 1, 2]$. The solution T is shown in Figure 2(b) and the group qualification is 4.21.

An algorithm is designed by following the algorithm for the simple role assignment problem. Two key modifications must be taken:

- 1) For each successful assignment, the group qualification must be recorded.
- 2) All the assignments must be tried.

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 ($j = 0, \dots, n-1$), $\sum_i T[i, j] \geq L[j]$,

and v as the maximum group qualification.

Failure: An $m \times n$ assignment matrix T in which there is at least one column j ($j = 0, \dots, n-1$), $\sum_i T[i, j] < L[j]$,

and v as 0.

RatedAssign (L, W, Q, T, m, n)

{

Step 1: Initialization

$v := 0$;

$T1 := \{0\}$; //All the values of $T1$ are set with 0.

// $T1$ is taken as a temporary matrix for T ;

for (all roles (j)) $M[j] = \sum_{i=0}^{m-1} [Q[i, j]]$ ($0 \leq j \leq n-1$), where

$[x]$ means the upper integer bound of x , e.g.,

$$[0.3] = 1;$$

//Collect the number of qualified agents for each role;

for (all roles (j)) if ($M[j] < L[j]$) return;

//If there is one role require more than the total //number of qualified agents, return.

Step 2: Build combinations of assignments to the role with the most qualified agent.

$p := L[0]$;

$k := M[0]$;

Create k dimensional vector A where each element is a qualified agent for role j .

$s :=$ the number of combinations taking p from k . i.e.,

$$\binom{k}{p} = k! / (p!(k-p)!)$$

Form an $s \times p$ matrix U from A (refers to the combinatory generator [2]), where each row is a set of qualified agents. The rows in U are all the combinations of taking p ($\leq k$) from k qualified agents.

Step 3: Sort the combinations.

The sorting is based on the sum of all the qualification rate values of agents in the combinations.

Step 4: Try each combination of agent assignment of role 0.

$v = 0$;

for (each combination (x))

{

Step 4.1: Initialization.

$v1 = 0$; $v2 = 0$; $T = \{0\}$;

Step 4.2: Assign agents.

for (each qualified agent (y) in this combination)
 {Assign agent U[x, y] to role 0, i.e., T[U[x, y], 0]=
 1;}

$$v1 := \sum_{i=0}^{m-1} Q[i,0] \times T[i,0];$$

Step 4.3: Prepare for a smaller problem.

Now, role 0 is workable and all its agents should not be considered as candidates to assign in the next recursion.

Q := Q; Delete the rows expressed by U[x] and the column j from Q;

L' = L; Delete item 0 in L';

Now, the number of agents becomes m-p and the number of roles becomes n-1;

Step 4.4: Solve the smaller problem.

v2 := RatedAssign(L', Q, T', m-p, n-1, v2);

Step 4.5: Check the solution of the smaller problem.

if (v1+v2>v)

{Combine the assignment done in the inner recursion with the assignment at this level, i.e.,

Q := Q and T := T' by keeping the rows in U[x] and column 0 unchanged;

L := L' by keeping item r;

Now, the role assignment at this level is successful;

if (v<v1+v2) {

v := v1+v2;

T1:=T; //Record a better solution.

}

}//end for (each combination)

T := T1;//The solution is in matrix T.

return v;//The maximum group qualification is in v.

}

The correctness of the above algorithm comes from the following facts:

- 1) Every agent assignment combination for each role is considered.
- 2) If there exists a role that has no assignment to make it workable, the algorithm fails.
- 3) After one agent assignment combination is selected, the problem becomes a smaller one.
- 4) If the smaller problem is solved, one group qualification is obtained.
- 5) If the smaller problem is not solved, the assignment in 3) is not accepted and a new assignment must be tried.
- 6) Every time, the obtained value is compared with the previous value to guarantee that the overall optimal assignment is obtained.

Similar to the complexity of the Assign algorithm, if

Step 4 is concentrated on, $\sum_{i=0}^{-1} L[i] = 0$, and there are

just enough agents for a group, i.e., $m = \sum_{i=0}^{n-1} L[i]$, the

complexity of the RatedAssign algorithm is

$$O\left(\frac{m!}{\prod_{j=0}^{n-1} L[j]!}\right).$$

6. THE WEIGHTED ROLE ASSIGNMENT PROBLEM

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 9: weighted group role assignment problem.

Let $Q[i, j] \rightarrow [0,1]$ ($0 \leq i \leq m-1$; $0 \leq j \leq n-1$) be the value that represents how well a given agent i plays a given role j , 0 means lowest and 1 the highest. Let W be a vector of weights of roles to express the importance of roles and $W[j] \rightarrow [0, 1]$ ($0 \leq j \leq n-1$). A *weighted group qualification* is defined as the *weighted sum of assigned agents' qualifications*, i.e.,

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

The problem is to find a matrix T that creates a group g in which the *weighted group qualification* is the largest, i.e.

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

For example, suppose Q is shown in Figure 2 (a), $L = [2, 1, 2, 1]$, $W = [0.19, 0.62, 0.42, 0.66]$. The solution T is shown in Figure 3. Evidently, this assignment is different from that shown in Figure 2(b).

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

Figure 3. The Solution Figure 2 (a) with Weighed Roles.

The algorithm is actually directly obtained from the **RatedAssign** algorithm discussed in Section 6. The major idea is to adjust the qualification matrix with the weight vector and then use the rated assignment algorithm. The reason is based on the following fact:

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

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

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

To accomplish this adjustment, a new operation “.” called *multiply by columns* is introduced between a vector and a matrix. Suppose that V is an n-dimensional vector and M is an m×n matrix, V·M creates a new m×n matrix M', where M'[i, j]:=V[j]×M[i, j], 0≤i<m and 0≤j<n.

Suppose that the inputs include a role weight vector W and the inputs for the algorithm of **RatedAssign**; and the outputs are the same as those of **RatedAssign**. The algorithm **WeightedAssign** has just two lines as below:

```

WeightedAssign (L, W, Q, T, m, n)
{
    Q':=W·Q;
    RatedAssign(L, Q', T, m, n);
}

```

7. IMPLEMENTATIONS, EXPERIMENTS AND PERFORMANCE ANALYSIS

The above algorithms are implemented in Java and tested by using the mentioned examples and randomly created different *agent qualification matrices*. In the implementations, a fundamental algorithm, i.e., generating combinations is required. The Rosen's algorithm [8] and the Gilleland's implementation [6] are adapted into the proposed algorithms.

From initial analyses, the proposed algorithms demonstrate exponential complexity. To verify the practicability of these algorithms, several experiments are designed. The hardware platform is a laptop with a CPU of 2.10GHz and the development environment is Microsoft Windows Vista (Home Edition) and Eclipse 3.2. Five experiments are conducted for each of the two algorithms (**Assign** and **RatedAssign**) with different agent (m) and role (n) numbers. Each experiment repeats for 300 randomly created agent qualification matrices to show its generality.

7.1. Experiments for Simple Role Assignment

This experiment is made for the simple role assignment problem. A random group is formed by randomly creating an *agent qualification matrix*: each agent is assigned randomly and evenly to be qualified for $\lfloor n/3 \rfloor$ roles to

make the role assignment relatively difficult to solve. The role number is chosen by $\lfloor (m+1)/2 \rfloor$. The lower ranges of a role is randomly set to L with 1 - 3 to make the corresponding group possibly have enough agents ($\sum_{j=0}^{n-1} L[j] \geq m$ (or $\leq m$)). The results are shown in Figures 4 -

8.

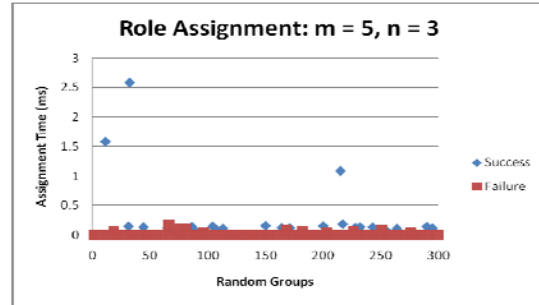


Figure 4. Simple Role Assignment: Experiment 1.



Figure 5. Simple Role Assignment: Experiment 2.

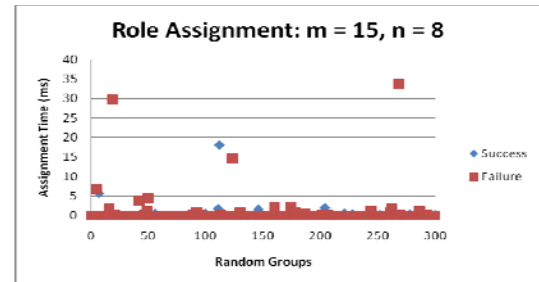


Figure 6. Simple Role Assignment: Experiment 3.

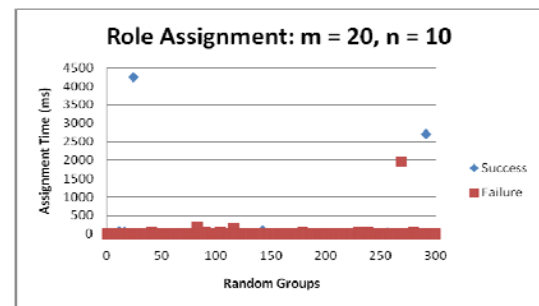


Figure 7. Simple Role Assignment: Experiment 4.

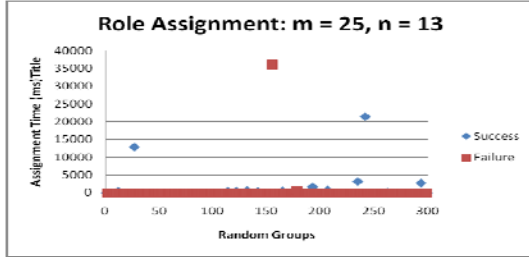


Figure 8. Simple Role Assignment: Experiment 5.

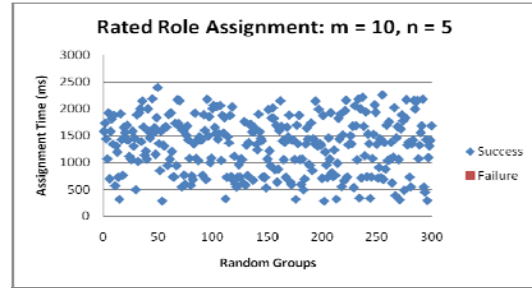


Figure 12. Rated Role Assignment: Experiment 4.

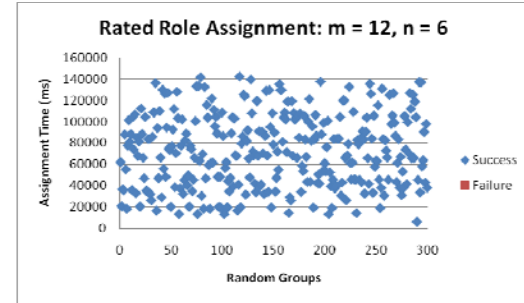


Figure 13. Rated Role Assignment: Experiment 5.

7.2. Experiments for Rated Role Assignment

This experiment is made for the rated role assignment problem. A random group is formed by randomly creating *an agent qualification matrix*: each agent is assigned randomly with values (0, 1] for n roles. The role number n is set as $m/2$. The lower ranges of a role is randomly set to L with 1 or 2 to make the corresponding group have enough agents ($\sum_{j=0}^{n-1} L[j] \leq m$). The results are shown in

Figures 9 - 13.

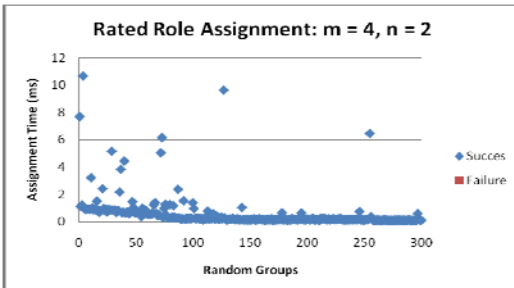


Figure 9. Rated Role Assignment: Experiment 1.

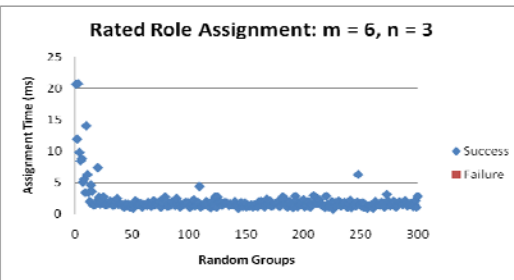


Figure 10. Rated Role Assignment: Experiment 2.

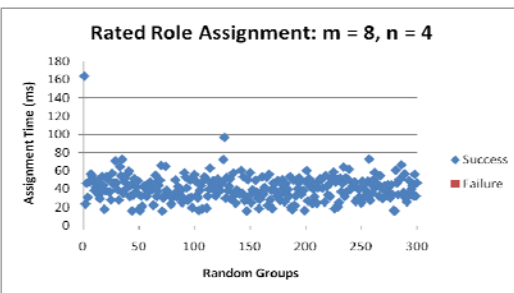


Figure 11. Rated Role Assignment: Experiment 3.

From the data obtained from the experiments, the highest and average times are summarized in Tables I and II.

Table I. The Largest and Average Times (ms) for Simple Assignment

| M \ Time | Largest | Average |
|----------|--------------|------------|
| 5 | 2.581054 | 0.032569 |
| 10 | 1.532597 | 0.045649 |
| 15 | 33.79857 | 0.491157 |
| 20 | 4240.532107 | 34.858953 |
| 25 | 36224.732752 | 278.809374 |

Table II. The Largest and Average Times (ms) for Rated Assignment

| M \ Time | Largest | Average |
|----------|---------------|--------------|
| 4 | 10.661341 | 0.572339 |
| 6 | 20.771215 | 1.978054 |
| 8 | 163.425729 | 40.704468 |
| 10 | 2399.302286 | 1316.465663 |
| 12 | 142494.647714 | 73186.052994 |

8. Analysis

Based on the figures and the tables, it is observed that both simple and rated assignment algorithms demonstrate exponential complexity that supports the theoretical analysis. The time used by rated assignment increases much faster than that of the simple example although

theoretically they have the same scale of complexity. This occurs because:

- 1) The simple one is done when a successful assignment is found;
- 2) The rated one needs to compare all the possible assignments and choose the best one; and
- 3) The random groups are created with different methods.

8. CONCLUSIONS

Role assignments are important in role-based collaboration. This paper contributes clear specifications of *group role assignment problems*, proposes exhaustive-search based algorithms to provide exact solutions to these problems, and conducts a series of experiments to present the performances of the algorithms. The experiments show that the algorithms are practical to small groups.

Further investigation should proceed along the following lines:

- 1) Analyzing the worst cases of agent qualifications and discovering more constrains for the simple assignment problems and improve the algorithm based on the new constrains.
- 2) Developing general heuristic algorithms to improve the performances for large groups, especially for rated role assignment whose complexity is very high.
- 3) Transforming large groups into small groups by special organizations and hierarchies to make the algorithms available.

ACKNOWLEDGMENTS

This research is in part supported by National Sciences and Engineering Research Council, Canada (NSERC: 262075-06), Ontario Partnership on Innovations of Commercialization, and IBM Eclipse Innovation Grant. Thanks also go to Mike Brewes of Nipissing University for proofreading this article.

REFERENCES

- [1] B. E. Ashforth, *Role Transitions in Organizational Life: An Identity-based Perspective*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2001.
- [2] M. Bhardwaj and A.P. Chandrakasan, "Bounding the lifetime of sensor networks via optimal role assignments", in *Proc. of Twenty-First Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, USA., vol. 3, June 2002, pp. 1587-1596.
- [3] J.S. Black, "Work Role Transitions: A Study of American Expatriate Managers in Japan", *Journal of International Business Studies*, vol. 19, no. 2, Summer, 1988, pp. 277-294.
- [4] R.P. Bostrom, "Role Conflict and Ambiguity: Critical Variables in the MIS User-Designer Relationship", *Proc. of the 17th Annual Computer Personnel Research Conference*, Miami, Florida, United States, 1980, pp. 88-115.
- [5] M. Dastani, V. Dignum and F. Dignum, "Role-assignment in open agent societies", *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia, 2003, pp. 489 – 496.
- [6] M. Gilleland , "Combination Generator", avail: <http://www.merriampark.com/comb.htm>, 2007.
- [7] J.J. Odell, H. Van Dyke Parunak, S. Brueckner and J. Sauter, "Changing Roles: Dynamic Role Assignment", *Journal of Object Technology*, vol. 2, no. 5, September-October 2003, pp. 77-86.
- [8] Rosen, K. H., *Discrete Mathematics and Its Applications*, 3rd edition (NY: McGraw-Hill, 1995).
- [9] M. Turoff, M. Chumer, B. Van de Walle and X. Yao, "The Design of a Dynamic Emergency Response Management Information System (DERMIS)", *J. of Information Technology Theory and Application*, 5(4), 2004, pp. 1-35.
- [10] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork", *Artificial Intelligence*, vol. 110, 1999, pp. 241–273.
- [11] D. Vail, M. Veloso, "Multi-Robot Dynamic Role Assignment and Coordination through Shared Potential Fields", In A. Schultz, L. Parkera and F. Schneider (eds.), *Multi-Robot Systems*, Kluwer, 2003, pp. 87-98.
- [12] H. Zhu, "Fundamental Issues in the Design of a Role Engine", *Proc. of the 6th International Symposium on Collaborative Technologies and Systems*, Irvine, CA, USA, May 19-23, 2008, pp. 399-407.
- [13] H. Zhu and M.C. Zhou, "Role-Based Collaboration and its Kernel Mechanisms", *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 36, no. 4, July 2006, pp. 578-589.
- [14] H. Zhu and M.C. Zhou, "Role Transfer Problems and Algorithms", *IEEE Trans. on Systems, Man and Cybernetics, Part A*, vol. 36, no. 6, Nov. 2008, pp. 1442-1450.