Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Improvement to Rated Group Role Assignment Algorithms

Haibin Zhu, *Senior Member, IEEE* and Rob Alkins
Department of Computer Science and Mathematics, Nipissing University
100 College Drive, North Bay, Ontario, P1B 8L7, Canada
E-mail: haibinz@nipissingu.ca

Abstract

Role assignment is a critical task in Role-Based Collaboration (RBC). It can be divided into three steps, i.e., agent evaluation, group role assignment, and role transfer, where group role assignment is a time-consuming process. This paper clarifies the problem of group role assignment; describes the general assignment problem; adapts the group role assignment problem to the general assignment problem; proposes an algorithm for group role assignment based on the Munkres algorithm; conducts an experiment by randomly creating agent qualifications, analyzes the solutions' performances, and points out the future work. The results show that the improved algorithm significantly improves the algorithm based on exhaustive search.

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: *negotiate roles; assign roles and play roles* [13]. Therefore, role assignment is an important aspect of RBC. It largely affects the efficiency of collaboration and the degree of satisfaction among the members involved in the collaboration. Role assignment can be categorized into three parts: *agent evaluation, group role assignment, and role transfer*.

Group role assignment initiates a group by assigning roles to agents, the group members, to seek the group's highest performance based on the agents' qualifications which are the results of agent evaluation.

Rated group role assignment seeks the optimal assignment of agents to roles. Some assumptions are as follows: agents perform roles within a group; different roles contribute different values to the maximal utility of the group; agents perform some roles better than other agents; an agent can only perform one role at a time; and some roles require more than one agent to play.

The remainder of this paper is arranged as follows: Section 2 formally defines the problem of rated role assignment; Section 3 presents the general assignment problem and how to transfer a rated role assignment problem to a general assignment problem. Section 4 demonstrates the Munkres algorithm and the newly adapted algorithm based on it; Section 5 depicts the experiments and analyzes the performances; Section 6 concludes this paper and points out the future work.

2. THE RATED ROLE ASSIGNMENT PROBLEM

With the E-CARGO model [13], collaboration is based on roles. In E-CARGO, a system $\sum$ can be described as a 9-tuple $\sum ::= <C, O, \mathcal{A}, \mathcal{M}, \mathcal{R}, \mathcal{E}, \mathcal{G}, s_0, \mathcal{H}>$, 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 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, the notations need to be clarified at first. If $\mathcal{S}$ is a set, $|\mathcal{S}|$ is its cardinality; suppose v is a variable, $v \rightarrow \mathcal{S}$ means v may take an element of $\mathcal{S}$ as its value; suppose a and b are objects, $a.b$ means b of a or a's b and $\{a, b, \ldots\}$ 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 $r::= <n, I, \mathcal{A}_c, \mathcal{A}_p, \mathcal{A}_o, \mathcal{R}_x, o_r>$ where,

- n is the identification of the role;
- $I ::= < \mathcal{M}_{in}, \mathcal{M}_{out} >$ 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;
- $\mathcal{R}_x$ 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 $a ::= < n, c_a, s, r_c, \mathcal{R}_p, \mathcal{R}_o, \mathcal{N}_g >$, 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 $\mathcal{R}_s$.

Definition 3: *environment.* $e ::= < n, \mathcal{R}_e, \mathcal{B}, o_g >$ 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., $< r, w, q, o_e >$, 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.* $g = < n, e, \mathcal{A}_g, \mathcal{J} >$ 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
- $\mathcal{J}$ is a set of tuples of agent and role, i.e., $\mathcal{J} = \{ < a, r > \mid r \in e.\mathcal{R}_e \wedge (\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \ni (< r, w, q, o_e > \in e.\mathcal{B})) \wedge (a \in r.\mathcal{A}_r \cup r.\mathcal{A}_p) \}$.

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 ($\mathcal{E}, C, O$), 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 $< a, r >$ of $g.\mathcal{J}$ is called a *role assignment*, also called *agent assignment*.

Note that, in role assignment, we do not differentiate current roles and potential roles that are applied in role transfer [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., $\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \ni (< r, w, q, o_e > \in e.\mathcal{B}) \wedge (|r.\mathcal{A}_c| \geq q.l)$. This is denoted as *workable*(r) = *True*.

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., $\forall r \in g.e.\mathcal{R}_e (\exists w \in [0, 1], q \in \mathcal{T}, o_e \in O \ni < r, w, q, o_e > \in g.e.\mathcal{B} \wedge (workable(r) = True))$ or group g expressed by T is *workable* if $\forall j (\sum_{i}^{m-1} T[i, j] \geq L[j])(0 \leq j \leq n-1)$.

Definition 8: *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., $\max \{ \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i, j] * T[i, j] \}$.

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

An algorithm can be designed based on exhaustive search but its complexity is very high. The complexity of

the designed algorithm is O(m!) under some conditions [5]. Even for groups that are not very large, this is impractical.

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

(a) (b)

Fig. 1. A rated qualification matrix and the solution.

3. THE GENERAL ASSIGNMENT PROBLEM AND THE ADAPTATION OF GROUP ROLE ASSIGNMENT

The general assignment problem [1, 2, 3] was described as: given the n×n matrix A of real numbers, find a permutation p (p_i; i = 1, …, n) of the integers 1, 2, . . . , n that minimizes $\sum_{i=0}^{n-1} A[i, p_i]$, for example, for the 3 × 3 matrix A = $\begin{bmatrix} 7 & 5 & 11.2 \\ 5 & 4 & 1 \\ 9.3 & 3 & 2 \end{bmatrix}$, there are six possible permutations for which the associated sums are shown in Table 1.

Table 1. The permutations and the sums for matrix A.

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

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

The rated group role assignment problem can be made a square problem by adjusting the number of agents and roles. Suppose in Definition 8, m = $\sum_{j=0}^{n-1} L[j]$, the problem is a square assignment problem.

To form the square matrix used in the Munkres algorithm, duplicate columns are added based on the vector L. The matrix Q and the vector L can be combined to create the matrix A in the square matrix assignment problem.

For example, the Q shown in Fig. 1 (a) and L = [2, 1, 1, 2] can create the matrix M shown as Fig. 2. It is needed to record the column numbers in a new vector L' = [0, 0, 1, 2, 3, 3].

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

Fig. 2. The Created Square Matrix

If m> $\sum_{j=0}^{n-1} L[j]$, it is needed to add m-n columns of 1s (i.e., every agent is fully qualified for an empty role) to make a square matrix. For example, for a Q (L = [1, 2, 1]) is shown in Fig. 3 its square matrix can be made as shown in Fig. 4 (a). The relevant column number vector L' =[0, 1, 1, 2, 3, 3] where 3 is the column number corresponding to empty roles. The solution can be obtained as shown in Fig. 4 (b).

$$Q = \begin{bmatrix} 0.36 & 0.76 & 0.72 \\ 0.93 & 0.59 & 0.24 \\ 0.06 & 0.46 & 0.69 \\ 0.4 & 0.1 & 0.74 \\ 0.23 & 0.75 & 0.24 \\ 0.21 & 0.77 & 0.24 \end{bmatrix}$$

L = [1, 2, 1]

Fig. 3. A matrix with m > $\sum_{j=0}^{n-1} L[j]$.

$$\begin{bmatrix} 0.36 & 0.76 & 0.76 & 0.72 & 1.0 & 1.0 \\ 0.93 & 0.59 & 0.59 & 0.24 & 1.0 & 1.0 \\ 0.06 & 0.46 & 0.46 & 0.69 & 1.0 & 1.0 \\ 0.4 & 0.1 & 0.1 & 0.74 & 1.0 & 1.0 \\ 0.23 & 0.75 & 0.75 & 0.24 & 1.0 & 1.0 \\ 0.21 & 0.77 & 0.77 & 0.24 & 1.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

(a) (b)

Fig. 4. A square matrix transferred from the matrix in Fig. 3.

Because Munkres algorithm is for a minimization problem, it is necessary to transform it to a maximization problem. This is simply a matter of subtracting the entries of *Q* from the largest entry of *Q* to obtain a new matrix *M*. Minimization of this new matrix results in a maximization of *Q*.

$$\begin{bmatrix} 0.29 & 0.29 & 0.4 & 1 & 0.78 & 0.78 \\ 0.71 & 0.71 & 0.33 & 0.56 & 0.24 & 0.24 \\ 0.31 & 0.31 & 0.08 & 0.08 & 0.4 & 0.4 \\ 1 & 1 & 1 & 0.47 & 1 & 1 \\ 0.03 & 0.03 & 0.49 & 0.23 & 0.35 & 0.35 \\ 0.42 & 0.42 & 0.36 & 0.76 & 1 & 1 \end{bmatrix}$$

Fig. 5. The square matrix transferred from the qualification matrix in Fig. 2.

For example, for the maximization of the matrix Q shown in Fig. 2, a matrix M can be obtained by such assignment:

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

Please note that the maximum qualification rate is 1. The matrix M is shown in Fig. 5.

4. THE ADAPTED GROUP ROLE ASSIGNMENT ALGORITHM

Munkres algorithm (or the Hungarian algorithm) is a minimization algorithm for square matrices of general assignment problems invented by James R. Munkres [1] in 1957. To understand Munkres algorithm, a definition and a theorem are required.

Definition 9: *independent set*. A set of elements of a matrix are *independent* if non of them occupies the same row or column.

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

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

At the beginning, the following conditions should be established: (a) No lines are covered; no zeros are starred or primed; (b) For each row of the matrix M subtract the value of the smallest element from each element in the row; and (c) For each column of the resulting matrix subtract the value of the smallest element from each element.

The major Munkres algorithm is as follows:

Input:

- *An m×m matrix M with the above preliminaries.*

Output:

- *All the subscripts of the selected elements of M, i.e., the starred zeros.*

```
MUNKRES(M):
{   for (all zeros z in M)
     If (there are no starred zeros in the row or column of z)
       star z;
   while (true) // the main loop
   {
      cover all the columns containing a 0*;
      if (all the columns are covered)
        return; // the optimal solution is starred;
      while (!(all zeros are covered))  //the inner loop
      {
       for (all non-covered zeros z)
         { prime z;
           if (there is a starred zero z* in this row)
             {   cover row of z*;
                 uncover column of z*;
             }
           else
             {
               highlight z;
               while (there is a starred zero z* in z's column)
               {  z=z*;
                  highlight z;
                  z:=the primed zero in z's row;
                        // this always exists
                  highlight z;
               }// End of while
               unstar highlighted starred zeros;
               star highlighted primed zeros and remove highlights;
               remove primes and uncover all rows and columns;
               continue to while (the main loop);
             } // end of if
         } //end of for ()
         h:=smallest uncovered element in M;
         add h to each covered row;
         subtract h from each uncovered column;
       }//end of while (true) – the inner loop
    }//end of while (true) – the main loop
  }// end of MUNKRES
```

The new algorithm for rated role assignment algorithm can be described as follows:

Input:

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

Output:

- **Success:** *An m×n assignment matrix T in which for all columns j (j = 0, …, n-1),* $\sum_{i}^{m-1} T[i,j] \geq L[j]$, *and v as the maximum group qualification.*

Failure: *An m×n assignment matrix T in which there is at least one column j (j= 0, …, n-1),* $\sum_{i}^{m-1} T[i,j] < L[j]$, *and v as 0.*

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

{

Step 1: Expand matrix Q to square matrix M for minimization by duplicating the columns whose corresponding role range is greater than 1 and adjust the qualification rates from maximization to minimization;

Step 2: MUNKRES(M); //Call the Munkres algorithm.

Step 3 : Form the assignment matrix T based on the result of MUNKRES (M);

}

5. IMPLEMENTATION AND PERFORMANCE EXPERIMENTS

The Munkres algorithm, when properly implemented can operate with a computational complexity of $O(n^3)$ [1, 3]. This is much better than the exhaustive search–based algorithm [5].

To verify the performance of this group role assignment algorithm, a Java implementation is made

based on the implementation of the Hungarian algorithm done by Gary Baker [2]. 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. Seven experiments are designed based on this environment and different group sizes (Table II).

In an experiment, a random group is formed by randomly creating *an agent qualification matrix*: each agent is assigned randomly with values (0, 1] for n roles, i.e., no agent is absolutely not qualified to play a role. 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 ($\sum_{j=0}^{n-1} L[j] \leq m$) to guarantee a successful assignment. Each experiment repeats for 300 randomly created agent qualification matrices to show its generality.The results of the experiments are shown in Figs. 6 - 12.

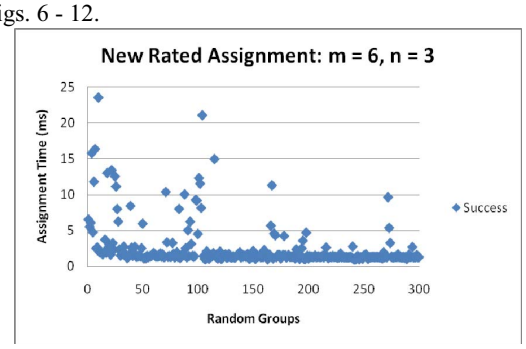

Fig. 6. Experiment 1: m = 6, n = 3.

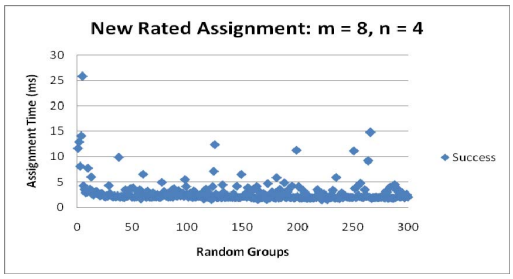

Fig. 7. Experiment 2: m = 8, n = 4.

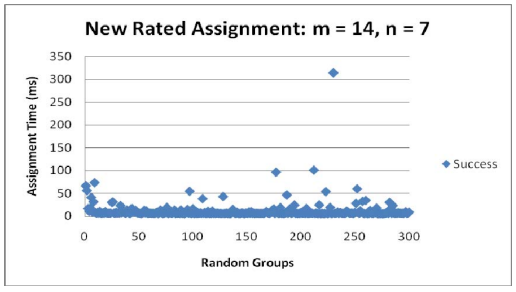

Fig. 8. Experiment 3: m = 14, n = 7.

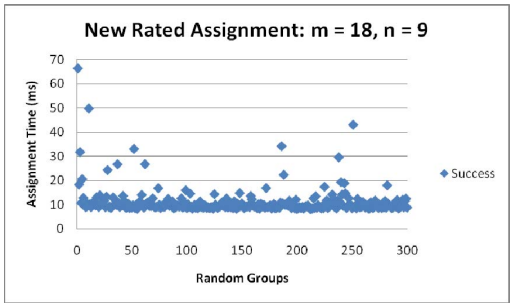

Fig. 9. Experiment 4: m = 18, n = 9.

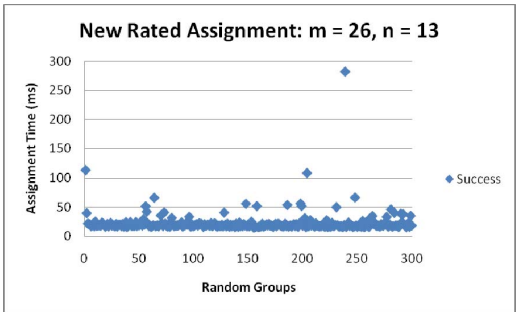

Fig. 10. Experiment 5: m = 26, n = 13.

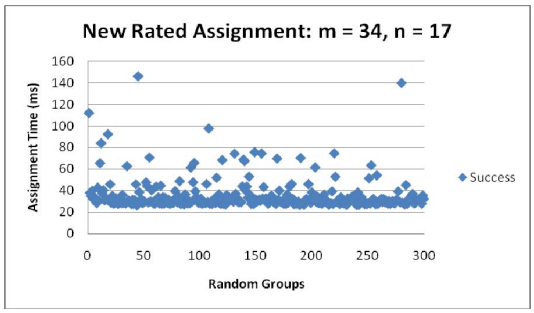

Fig. 11. Experiment 6: m = 34, n = 17.

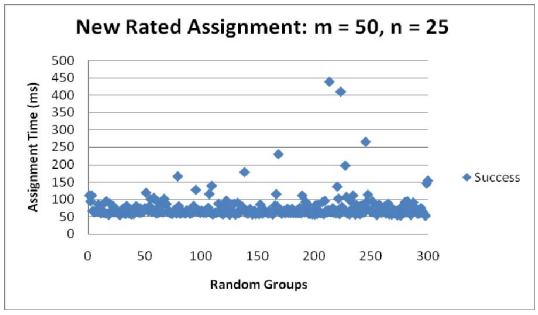

Fig. 12. Experiment 7: m = 50, n = 25.

Table I. The largest and average times (ms) for the rated assignment algorithm

M \ Time	Largest	Smallest	Average
6	23.524288	0.956686	2.488235
8	25.885552	1.537276	2.944034
10	21.938403	2.590972	3.965053
14	313.49564	4.785944	11.028961
18	66.406764	8.176808	10.998644
26	282.209261	15.679157	23.017014
34	146.123181	26.517406	35.876745
50	438.825186	52.898483	75.288766

6. PERFORMANCE ANALYSIS

Table I shows the typical data collected from the seven experiments stated above. It is evident that the typical times (largest, smallest and average times) used by the rated assignment algorithm are better than the complexity of Munkres algorithm $O(n^3)$ (Fig. 13.). This situation occurs , because the square matrix used by the Munkres algorithm is created from the random groups. The random groups may create many columns fully with 1s, therefore, the time used by the algorithm is normally less than $O(n^3)$.

From Figs. 6-13 and Table I, the newly adapted algorithm for the group role assignment is much better than the algorithm proposed in [5]. It is an exciting result.

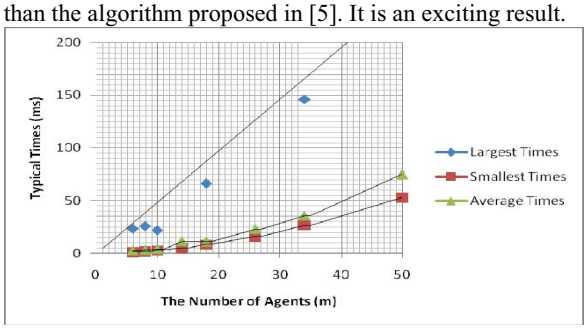

Fig. 13. The trend lines for typical times for different sizes of the groups.

7. CONCLUSIONS

Rated role assignments are an important problem in role-based collaboration. Efficient algorithms are required for practical applications. This paper contributes an efficient way to solve this problem.

Further investigations can be done along the following directions:

1. A number of restrictions could be supported by the algorithm. Examples include the requirement that certain roles must be filled while other roles are optional. Similarly there may be a certain ratio of management roles to subordinate roles.
2. Temporal requirements and continuously changing demand for roles may be considered.
3. In real world, the demand for the services of each role is not constant. Nor is the number of agents available to play these roles. Role assignment algorithms must be able to adapt to these changing requirements and resources, and provide continuous solutions that minimize task switching of agents.
4. The scheduling of agents performing tasks could be considered. The maximization of group utility must be subject to availability and maximum workload restrictions.

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.

REFERENCES

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

[2] G. Baker, "Java implementation of the classic Hungarian algorithm for the assignment problem", http://sites.google.com/site/garybaker/hungarian-algorithm/assignment, 2008.

[3] Wikipedia, "Hungarian algorithm", http://en.wikipedia.org/wiki/Hungarian_algorithm, 2009.

[4] 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.

[5] H. Zhu and R. Alkins, "Group Role Assignment", Submitted to *IEEE Int'l Symposium on Collaborative Technologies and Systems,* Baltimore, MA, May 2009.

[6] 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.

[7] 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.

[8] H. Zhu and M.C. Zhou, "M-M Role Transfer Problems and Solutions", *IEEE Trans. on Systems, Man and Cybernetics, Part A,* vol. 37, no. 6, Nov. 2008, pp. 1442-1450.