Role Transfer Problems and Algorithms

Haibin Zhu, Senior Member, IEEE, and MengChu Zhou, Fellow, IEEE

Abstract—Role transfer is a usual activity in an organization, especially in a crisis situation. Role assignment and transfer regulations are important to accomplish it. This paper discusses the general role transfer problem; proposes a role specification mechanism; builds a set of terminologies for role transfer based on a revised E-CARGO (Environment-Class, Agent, Role, Group, and Object) model; and presents algorithms to validate role transfer while maintaining group viability. The contributions include formulating the problem of role transfer in a generalized form; developing a set of algorithms; and presenting a solution when group members are insufficient.

Index Terms—Role, Role Transfer, Information Systems, Organization, Emergency Management

I. INTRODUCTION

Role transfer is a basic requirement for organizations [1] and emergency management systems [15]. It is used to evaluate and check the flexibility of a group when its membership and/or roles change. Role transfer is a complex event that may involve many relevant roles. When an organization encounters a crisis situation, not every one can transfer his/her roles. When one person transfers to a new role, his/her original role needs to be assumed by another if the role is still required in the system. This change might initiate a series of role transfers. For example, in a battle field, if a high rank officer is disabled, a similar or lower rank officer is needed to play his/her role. The role played by the similar or lower rank officer may need to be played by another, etc. A highly-available computer system should meet the similar requirement for computers, sub-systems, components to transfer their functions (or roles) to recover from a faulty state. We generally need duplicate them to tolerate faults.

One specific role should be played by more than one person (computer, sub-system, or system component). One person should be able to play more than one role [15, 17-20]. However, how many roles should a person play or potentially play to deal with an emergency situation? Which roles should a person play? What role-person mapping is effective in dealing with an emergency situation? What group structures can avoid crisis? There are no clear, exact answers for these questions hitherto. This paper intends to demonstrate that specially arranged role-person mapping can help a group survive better in an emergency situation. Our role-based collaboration model E-CARGO (Environment-Class, Agent, Role, Group, and Object) [17, 18] can be successfully adopted in processing such situations.

There are actually two aspects in the research of role transfer. One is that when there are sufficient people and the other is not. We need to investigate different methods to deal with them. Our solution for the latter is to introduce time intervals to solve it. We call the latter temporal role transfer.

The rest of this paper is arranged as follows: In Section II, we discuss some relevant research. In Section III, we demonstrate the revised E-CARGO model that aims at supporting role transfer. Section IV describes fundamental terminologies for role transfer; Section V discusses the role transfer problem and relevant theorems; Sections VI, VII, and VIII describe the algorithms and their implementation for role transfer. Section IX extends the role transfer problem to a temporal role transfer problem and gives its solution algorithm. Section X concludes the paper.

II. RELATED RESEARCH

Although role transfer is an important problem in management, organizational behavior and performance, and system development, there is no comprehensive research on role transfer theory, algorithms and practice. Some research mentioning the similar problem uses different terms, for example, resource assignment [5] and job assignment [10]. Some related research is those in agent systems [14] and wireless communications [2] using the term role assignment. Others mainly investigate people’s or organization’s behaviors [7, 12] when role transfer happens and they use the term role transition. The research by sociologists and psychologists mainly concerns the behavior of people and organizational performance when role transition occurs [1, 3, 4, 13].

Research on the delegation of rights (tasks, authorization, permissions, responsibilities, or even roles) [9, 11, 16] deals with the problem of transferring rights (permissions, responsibilities or roles) to neighbor agents or subordinate users. It mainly provides policies, rules or protocols to guarantee that the transfer (sometimes copy) process is possible, complete, trustable and secure. The results are mainly
used for computer security.

The current research shows that there are indeed strong needs to investigate the role transfer problem. The research results presented in this paper are expected to apply in many different fields, such as, information systems, management, production, and manufacturing industry.

III. REVISED E-CARGO MODEL FOR EMERGENCY SYSTEMS

Without a clear specification of components, there would be no real successful system. By establishing a formal model to define and specify a role, we can obtain a clear view of systems. A well-defined internal structure is a guarantee for a successful system. It supports the robustness, efficiency and correctness of the entire system. By E-CARGO [17, 18], we mean Environments, Classes, Agents, Groups, Roles, and Objects. A role-based system $\Sigma$ is described as a 9-tuple $\Sigma := < C, O, A, M, R, E, G, s_0, H >$, where, $C$ is a set of classes; $O$ is a set of objects; $A$ is a set of agents who are representatives of human users; $M$ is a set of messages; $R$ is a set of roles; $E$ is a set of environments; $G$ is a set of groups; $s_0$ is the initial state; and $H$ is a set of users.

To deal with the role transfer problem in organizations and systems, we emphasize the relationships among roles and agents in a group. Therefore, we restate the details of an agent, role, environment, and group. The definitions of these components are given as follows, where, if $x$ is a set, $|x|$ is its cardinality; $a.b$ means $b$ of $a$ or $a's$ $b$.

A role is defined as $r := < n, I, A_r, A_p, N_r >$ where $n$ is the identification of the role; $I := < M_{in}, M_{out} >$ denotes a set of messages, where $M_{in}$ expresses the incoming messages to the relevant agents, and $M_{out}$ expresses a set of outgoing messages or message templates to roles, i.e., $M_{in}, M_{out} \subset M$; $A_r$ is a set of agents who are currently playing this role; $A_p$ is a set of agents who are potential to play this role; and $N_r$ is a set of objects including classes, environments, roles, and groups that can be accessed by the agents playing this role.

An agent is defined as $a := < n, c_a, s, r, R_p, N_a >$, where $n$ is the identification of the agent; $c_a$ is a special class that describes the common properties of users; $s$ is the profile of the person whom the agent is representing; $r$ means a role that the agent is currently playing. If it is empty, then this agent is free; $R_p$ means a set of roles that the agent has potential to play ($r \notin a.R_p$); and $N_a$ is a set of groups that the agent belongs to.

An environment $e := < n, B >$ where $n$ is the identification of the environment; and $B = \{ < r, q, \Psi > \}$ is a set of 3-tuples of role, number range and an object set. The number range $q$ tells how many users may play role $r$ in this environment and $g$ is a tuple of number $< f, u >$, where $f$ means the lower bound of the number of agents to play this role and $u$ the upper bound. The object set $\Psi$ expresses the objects accessed by the agents who play the relevant role. The objects in $\Psi$ are mutually exclusive, i.e., one complex object in this set can only be accessed by one agent (user). For each tuple, $q \leq |\Psi| \leq q.u$. In an emergency system, we are more interested in $L$.

A group is defined as $g := < n, e, J >$ where $n$ is the identification of the group; $e$ is an environment for the group to work; and $J$ is a set of tuples of an agent, role, and complex object, i.e., $J = \{ < a, r, o > | \exists q, \Psi (o \in \Psi) \land ( < r, g, \Psi > \in e.B ) \}$.

In a crisis, the E-CARGO model can be used to demonstrate the states of a group of people using an emergency system. The agents and related roles in the system clearly tell the people their responsibilities and rights. If one person is called out of the group, another in it must take the position of that person. The computer system can quickly show the new roles to the people in the group if an efficient role transfer mechanism is in place.

IV. BASIC DEFINITIONS

In some situation, especially a crisis, the agent-role relationships need to be changed. Only some agents, roles or groups are critical. We need to discern the differences in order to deal with a crisis effectively. To understand the problem of a crisis situation, we need to define some common terms. In the following discussion, we use agents or people interchangeably and are mainly concerned about $A$, and $A_r$ of a role $r$, and $r$ and $R_p$ of an agent $a$.

Definition 1: Role $r$ is current to agent $a$ if $a$ is currently playing $r$, i.e., $r = a.r$; and $a$ is called as a current agent of $r$, i.e., $a \in r.A_r$. Role $r$ is potential to $a$ if $a$ is qualified to play but not currently playing it, i.e., $r \neq a.r_p$; and $a$ is called as a potential role of $a$, i.e., $a \in r.A_p$. The current role and all potential roles of an agent is called as its role repository, i.e., $\{ r \} \cup R_p$. Role $r$ is critical if it has only enough agents currently playing it, i.e., $\exists g, \Psi ( < r, g, \Psi > \in e.B ) \land ( |r.A_r| = q )$.

Definition 2: Group $g$ is workable if for each role $r$ there are enough agents to play it, i.e., $\forall r \in R_p \exists q, \Psi ( < r, g, \Psi > \in e.B )$. If a role loses its current agent(s), $g$ re-distributes the agents to play roles. Such re-distribution is called as a role transfer. A role transfer is successful if $g$ is workable after it.

Definition 3: An agent is critical if there is no successful role transfer to make group $g$ workable once it leaves $g$. When an agent leaves $g$, $g$ loses this agent and this agent is a lost agent of $g$. Group $g$ is critical if every agent in it is a critical one. A level-n emergency occurs if $g$ has lost $n (\geq 1)$ agents. $g$ is level-n strong if it is workable via successful role transfer after a level-n emergency occurs.

Definition 4: A graph of group $g$ is defined as a tuple, $G = < V, E >$, where, $V$ is the node set and $E$ is the arc set, and $V = R_p \cup A_r$, and $E = \{ < a, a.r > \mid a \in A \} \cup \{ < a, r > \mid a \in A_r \land r \in R_p \}$. There is a partition in $g$ if its graph is not connected [8].

Note that, $a.R_p$ is a set of all the potential roles of an agent $a$. The agent can directly serve the requests relevant to its current role. For an agent, a potential role can become current and vice versa. An agent can have only one current role. That is to say, an agent can hold many potential roles at the same time but only
one current role at a time. By holding only one current role, we can avoid role-role conflicts [4]. If a role transfer for agent \( a \) occurs, \( a \rightarrow r \) is swapped with an \( r \) in \( a \rightarrow R \). The repository of an agent is actually all the roles that it is qualified to play.

In a crisis, people must concentrate on their roles all the time. Every role must have enough players to play it at any time. A critical group in Definition 3 tells that no agent in it can be lost to keep it workable.

V. ROLE TRANSFER PROBLEM

Role transfer problems can be understood by the following examples. For simplicity, we assume that a role must have at least one current agent, i.e., \( I \geq 1 \).

![Fig. 1. Two different role assignments with each role having two agents (current and potential).](image1)

![Fig. 2. Two different role assignments with each role having three agents (current and potential).](image2)

**Case 1:** Fig. 1(a) shows a workable group. Suppose that as the current player of \( R_7 \), i.e., \( A_8 \) leaves the group. Is the group still workable by role transfer? \( R_7 \) can be played by \( A_7 \). Hence, let \( A_7 \) play \( R_7 \). Consequently, \( R_6 \) has no current player. We can let \( A_6 \) play \( R_6 \), then there is no agent to play \( R_5 \). Therefore, the group shown in Fig. 1(b) is not workable.

**Case 2:** Fig. 1(b) shows a group similar to Fig. 1(a). The only difference is that the agents’ qualifications are more than those in Fig. 1(a). After \( A_8 \) leaves the group, \( A_7 \) can play \( R_7 \), \( A_6 \) can play \( R_6 \), and \( A_5 \) can play \( R_5 \). Although \( A_4 \) can play \( R_4 \), we cannot find another qualified agent to play \( R_3 \). Therefore, the group is not workable after \( A_8 \) leaves.

**Case 3:** Fig. 2(a) shows a re-arrangement of Fig. 1(a). If \( A_8 \) leaves the group, we can make it a workable group by letting \( A_3 \) play \( R_3 \) and \( A_2 \) playing \( R_3 \). Now, \( A_8 \) is not a non-critical agent. In fact, we can verify that Fig. 2(a) is a level-1 strong group.

**Case 4:** Fig. 2(b) shows a group similar to Fig. 1(b). The only difference is that the agents’ qualifications are rearranged. After \( A_8 \) leaves the group, we can find \( A_2 \) to play \( R_6 \). The problem is solved. It means that \( A_8 \) is not a critical agent for the group in Fig. 2(b).

VI. ALGORITHM TO FIND A PARTITION

Whether an agent is critical is largely dependent on the group organization. To answer if a group is strong, we need to check if there is a partition in the group. For example, the group in Fig. 1(a) has a partition and it has no successful role transfer after agent \( A_8 \) leaves. Even though an algorithm of graph connectivity can be used to check if a group has a partition, a simpler one can be designed by composing role/agent assignment matrices, because in a group there are evidently two kinds of nodes, i.e., agents and roles. Assume \( M = |A| \) and \( N = |R| \), a group can be expressed by an \( M \times N \) current role matrix \( C \) and an \( M \times N \) potential role matrix \( \Pi \). \( C[i, j] = 1 \) means that agent \( A_i \) is playing currently role \( R_j \); \( \Pi[i, j] = 1 \) means that agent \( A_i \) is qualified to play role \( R_j \).

![Fig. 3. A partition in a group.](image3)

Fig. 3. A partition in a group.

Considering Fig. 3, we can construct the current role matrix \( C \), the potential role matrix \( \Pi \), and their role repository matrix \( S \) as shown in Fig. 4(a), (b) and (c), where \( S \) is defined as \( S = C + \Pi \).

![Fig. 4. The matrices for Fig. 3.](image4)

Fig. 4. The matrices for Fig. 3.

If we have a role repository matrix \( S \), we could arrange the matrix and find a special scheme that tells if there is a partition in a group.

When observing the role repository matrices shown in Fig. 5, we could find that Fig. 5(a) has a partition but Fig. 5(b) not. It is actually to check if we can go through 1s in \( S \) to reach each row and column if the columns and rows are properly arranged.

![Fig. 5. Two role repository matrices.](image5)

Fig. 5. Two role repository matrices.

A special algorithm for checking partitions is designed. To start from an agent (normally, the first row in matrix \( S \)) having a role (current or potential): (1) search and collect all the new roles touched (current or potential) by it and label them “old”; then (2) for all the roles collected, search all the “new” agents touched by all the roles; (3) search and collect all the “new” roles touched by all the agents collected in (2); repeat (2) and (3) until there is nothing new added to the collected roles and agents. If all the agents and roles are collected, we know that the group has no partition; otherwise, there is a partition.
Input: An M×N matrix P that is the role repository matrix.
Output: TRUE if P has partitions; and FALSE if it has no partition.

Process:
1) Create and initialize with 0s an Array A[M] to collect agents;
2) Create and initialize with 0s an Array R[N] to collect roles;
3) Find the first raw i and the first column j in P and P[i, j]=1, if i > 0 return TRUE;
4) A[i]=1; //The first agent is added to the agent array;
5) R[j]=1; // The first role is added to the role array.
6) Set a tag T :=1 to show that at least one new agent is added to the agent array and one new role to the agent array;
7) Collect all the roles touched by agent i;
8) While (T==1) // A new role is added
   { T := 0;
      Collect all the agents touched by the roles in array R and set T=1 if a new agent is added;
      If (T==1) // A new agent is added.
      { T := 0;
          Collect all the roles touched by the agents in array A and set T=1 if a new role is added;
      }
   } 9) If (all agents are added into the array A and all the roles to R) return FALSE else TRUE.

The algorithm is of complexity \(O(MN^2)\). It seems more complex than a typical DFS (Depth First Search) algorithm [6] as DFS has complexity \(O((M+N)^2)\) when an adjacency matrix is used. However, if \(M\) is a constant, it is similar to the typical DFS algorithm, i.e., \(O(N^2)\); if \(N\) is a constant, it is better than DFS, i.e., \(O(M)\).

VII. ROLE TRANSFER ALGORITHM WITH MATRICES

A role transfer can be expressed by converting matrix \(C^0\) into \(C^1\) mathematically. For example, if \(C^0\) is shown as in Fig. 6 (a) and \(C^1\) Fig. 6 (b), a successful transfer exists, i.e., agent \(A_2\) is now playing \(R_2\) and the group is workable. An example of current role matrix \(C^0\) and \(C^1\) (\(M = 8, N = 8\)) is shown in Fig. 1. Fig. 6(a) \((C^0)\) expresses the situation of Fig. 2(a) when \(A_8\) is lost and Fig. 6(b) \((C^1)\) expresses a solution of Fig. 6(a). From \(C^0\) and \(C^1\), there is a role transfer, i.e., \(A_2\) acquires a new current role \(R_2\).

To make a group workable, a successful transfer is needed and can be conducted with the help of matrix \(Q\). \(Q\) expresses the potential roles of each agent. For example, Fig. 7 shows such a matrix for Fig. 1(b) after \(A_8\) leaves.

\[\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}\]

(a)

\[\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}\]

(b)

Fig. 6. A role transfer for Fig. 2(a) expressed by matrices.

With the matrices defined above, we have an algorithm to find a role transfer.

Input: An M×N current role matrix \(C^0\) and an M×N potential roles matrix \(Q\).

\[\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{bmatrix}\]

Fig. 7. A potential roles matrix

Output: Success - an M×N current role matrix \(C^1\) in which there is no column with all zeros. Failure - an M×N current role matrix \(C^1\) in which there is still a column with all zeros.

Process \((C^0, C^1, Q, M, N)\):

1//Note: "=" means assignment from right to left and "==" //means that the left is equal to the right.

If (M=0 or N = 0) then return failure;
If (in matrix Z=Q+C^0 there is one column with all zeros, i.e., \(Z[j, i]=0 (j = 0, 1, \ldots, N-1)\)) then return failure;
\(C^1:=C^0\);
For (all columns of \(C^1\))
{  If (column j has all zeros in \(C^1\))
   {From Q, find row number i \(Q[i, j]\)=1;
    In \(C^1\), check if this agent (i) has no current role, i.e., it is a free agent;
    If (Yes)
    \{ \(C^1[i, j]:= 1, Q[i, j]:=0\);
        \(C^0 := C^1\) and \(Q := Q\);
        Delete row i and column j from \(C^0\) and \(Q\);
        \(M := M -1, N := N-1\);
        Call Process\((C^0', C^1', Q', M, N)\);
        \(Q:= Q\) and \(C^1 := C^1'\) with keeping the original row i and column j;
        Return;
    \}
    Else
    \{Find the column index k such that \(C^1[i, k]=1\);
        If (Yes)
        \{ \(C^1[i, j]:= 1, Q[i, j]:=0, Q[i, k]:=1, C^1[i, k]:= 0\);
            \(C^0 := C^0'\) and \(Q := Q\); delete row i and column j from \(C^0\) and \(Q\);
            \(M := M -1, N := N-1\);
            Call Process\((C^0', C^1', Q', M, N)\);
            \(Q:= Q\) and \(C^1 := C^1'\) but keeping original row i and column j;
            Return;
        \}
        Else
        \{Return failure;
        \}
    \}
    Report success;
}
The algorithm’s complexity $C(M, N) = O(MN) + O(N)$ ($O(MN) + C(M-I, N-I)$). Treating $N$ and $M$ as a constant, we have $C(M) = O(M^2)$ and $C(N) = O(N!)$, respectively. Hence, role transfer is NP-hard with respect to the number of roles.

VIII. ALGORITHM FOR ROLE TRANSFER WITH E-CARGO MODEL

Both specific data structures and algorithms are needed to implement a role transfer algorithm. The E-CARGO model is qualified to provide such structures. A descriptive algorithm is as follows. Suppose that $S$ is a set and $s \in S$, we use $S.remove(s)$ to express “removing $s$ from $S$.” Suppose $e \not\in S$, we use $S.add(e)$ to express “adding $e$ to $S$.”

Input: A group $G$ with a set of roles $R$, a set of agents $A$, a set of $\langle$role, agent$\rangle$ mapping for current agents $U$ and a set of role mapping $\langle$role, agent$\rangle$ for qualified agents $V$, and a role $w$ without a current agent, i.e., $\forall a \in A \langle w, a \rangle \not\in U$.

Output: Success - a group $G$ with a set of roles $R$, a set of agents $A$, a set of $\langle$role, agent$\rangle$ mapping for current agents $U$, a set of role mapping $\langle$role, agent$\rangle$ for qualified agents $V$, $\forall r \in R, \exists a \langle r, a \rangle \in U$. Failure - a group $G$ with a set of roles $R$, a set of agents $A$, a set of $\langle$role, agent$\rangle$ mapping for current agents $U$ and a set of role mapping $\langle$role, agent$\rangle$ for qualified agents $V$ and a role without current agent $w$, i.e., $\forall a \in A \langle w, a \rangle \not\in U$.

Process:

Step 1: Preparing for role transfer:
1) Build $R$ of all the roles in group $G$.
2) Build $A$ of all the agents in group $G$.
3) Set $w$ the role currently without a playing agent.
4) If $|R| > |A|$ report failure, i.e., there is no successful role transfer.
5) Else proceed to Step 2.

Step 2: RoleTransfer($A$, $R$, $w$):
1) If $A = \Phi$ and $R \neq \Phi$ then report failure and exit.
2) If $R = \Phi$ then report success and stop.
3) For all $a \in A$ do
   { If $w \in a.R$ (i.e., $\langle w, a \rangle \in V$) then
     { If $a.r$ is empty then set $a.r$ with $w$.
      Report success and exit.
     }
   }
4) For all $a \in A$ do
   { If $w \in a.R$ (i.e., $\langle w, a \rangle \in V$) then
     { Save $a.r$ to $q$.
      // Note that, $q$ is a temporary space to store the
      //current role.
      Set $a.r$ with $w$;
      Set $A$ with $A.remove(a)$;
      Set $R$ with $R.remove(w)$;
      Set $w$ with $q$.
      Call RoleTransfer($A$, $R$, $w$);
   }

If Role transfer is successful then report success and exit;
Else
   { Set $A$ with $A.add(a)$;
    Set $R$ with $R.add(w)$;
    Set $w$ with $q$.
   }

Step 3: Report failure and exit.
To facilitate the algorithm, we put three special variables into a group RoleSet, AgentSet and eRole, where RoleSet is a set of all the roles in the group; AgentSet is a set of the agents in the group and eRole is the role currently having no current role, i.e., $R$, $A$, and $w$ in the above algorithm.
We have implemented two functions: prepareRoleTransfer() is used to set the above three variables (Step 1) and transferRoles() is used to transfer roles in the group (Step 2).
The key point of the implementation is that we need a graph retrieval algorithm and a back traceable search algorithm.

The complexity of the algorithm is $C(M, N) = O(M+N) + O(M)$ ($O(M^2)$, respectively. Compared with the algorithm using matrices, this one is more affected by the agent number. The difference comes from the recursive searching space: the former searches in the role space but the latter searches in the agent space.

IX. TEMPORAL ROLE TRANSFER

Section V and VII discuss the methods how to check if a group has a successful role transfer when there are enough agents to play roles. That is to say, we assume that there are sufficient agents or people to avoid a crisis situation, i.e., $|A| \geq |R|$. In fact, in an emergency situation, scarcity is often more than sufficiency, i.e., $|A| < |R|$. In this situation, the intuitive solution is letting a person play more than one roles in different time segments.

To express temporal role transfers, we actually have three parameters, i.e., roles, agents, and intervals. The problem becomes: could we check if a group is workable in a period when we allow time sharing agents? That is to say, if every role has enough agents to play at every one $r$ in $p$ ($p = nr$), the group is considered workable. For example, in a crisis, an office still works if each position has at least two hours with a person working on it in a 24-hour day, where, $p = 24$ hours and $r = 2$ hours and $n = 12$.

We can understand this problem with an example shown in Fig. 8. In Fig. 8(a), the group is not workable and has no successful transfer after $A_7$ is lost, because there is no other agent that can play role $R_7$, i.e., $A_7$ is a critical agent. If $A_6$ is lost, the group can be scheduled workable in a period $p$ at scale 2. That is to say, for any period $p$, we can separate it into $2$ intervals and make the group temporally workable shown in Fig. 8(b) and Fig. 8(c). In this period, $A_5$ transfer back and forth
to play roles R4 and R5.

In this crisis situation when agents are not enough, we can still deal with it by having an agent play different roles at different time intervals. We call this as temporal role transfer.

To accommodate this requirement, we need to revise the definition of an agent and role in the E-CARGO model,  

$$a ::= < n, c_w, s, C_r, R_p, N_g >,$$

where $C_r$ replaces the original $r_i$ to express a list of current roles, i.e., a list of tuples in a form of $< r_i, r_k >$ and $r ::= < n, I, \mathcal{A}_i, \mathcal{A}_r, N_g >,$ where $\mathcal{A}_r$ replaces the original $\mathcal{A}_i$ to express a list of tuples of $< a_r, r_i >$.

**Definition 5:** Suppose that a period $r$ is divided into $n$ intervals, i.e., $r = [r_0, r_1, ..., r_{n-1}]$. When an agent $a$ plays role $r$ at an interval $r_i$ ($i = 0, 1, ..., n-1$), $< a_r, r_i >$ is called an agent-interval for $r$ and $< r_i, r_k >$ is called a role-interval for $a$. A group $g$ is temporally workable in $r$ at scale $n$ if for each role $r$ there are enough agent-intervals to play it, i.e., $\forall r \in R (\exists q, \psi \psi < r, q, \psi > \in g.e.B)$ and $\forall r_i \in r (< a_r, r_i > \in r, \mathcal{A}_r) \rightarrow (|\mathcal{A}_r| \geq q.\psi)$.

Before developing an algorithm for temporal role transfer, it is needed to understand the inherent properties of the relationships among roles, agents and intervals. To make the problem more concise, a role assignment to an agent is expressed as a point in a 3-D space (Fig. 9).

Fig. 8. Temporal Role Transfer.

**Properties of role playing matrices for a workable group are:**

$$0 \leq \sum_{j=0}^{m-1} C_k[i,j] = n, j = 0, 1, ..., l-1, k = 0, 1, ..., n-1.$$  

$$0 \leq \sum_{i=0}^{m-1} C_k[i,j] = 1, i = 0, 1, ..., n-1, j = 0, 1, ..., n-1.$$  

$$0 \leq \sum_{i=0}^{m-1} C_k[i,j] = n, i = 0, 1, ..., n-1, j = 0, 1, ..., n-1.$$  

**Definition 6:** $C_0, C_1, ..., C_{n-1}$ are called a role transfer matrix sequence. Each one expresses that the agents $(\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_{m-1})$ that currently play roles $(R_0, R_1, ..., R_{n-1})$ in the period $r$ of $n$ time intervals $(r_0, r_1, ..., r_{n-1})$. In matrix $C_k, C_k[i,j] > 1$ ($k = 0, 1, ..., n-1$) and $j = 0, 1, ..., l-1$ means that agent $\mathcal{A}_i$ plays role $R_j$ at interval $r_k$.

Fig. 9. A role assignment is a point in a 3-D space.

A sequence of matrices $C_0, C_1, ..., C_{n-1}$ is used to express the role assignment, such as $C_k[i,j]$ ($i = 0, 1, ..., m-1; j = 0, 1, ..., l-1$; and $k = 0, 1, ..., n-1$), where $m$ is the number of agents, $l$ is the number of roles and $k$ is the number of intervals. Role transfers occur by following $C_0$ to $C_{n-1}$ ($i = 0, 1, ..., m-1$).

**Definition 6:** $C_0, C_1, ..., C_{n-1}$ are called a role transfer matrix sequence. Each one expresses that the agents $(\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_{m-1})$ that currently play roles $(R_0, R_1, ..., R_{n-1})$ in the period $r$ of $n$ time intervals $(r_0, r_1, ..., r_{n-1})$. In matrix $C_k, C_k[i,j] > 1$ ($k = 0, 1, ..., n-1$) and $j = 0, 1, ..., l-1$ means that agent $\mathcal{A}_i$ plays role $R_j$ at interval $r_k$.

Properties of role playing matrices for a workable group are:

$$0 \leq \sum_{j=0}^{m-1} C_k[i,j] = n, j = 0, 1, ..., l-1, k = 0, 1, ..., n-1.$$  

$$0 \leq \sum_{i=0}^{m-1} C_k[i,j] = 1, i = 0, 1, ..., n-1, k = 0, 1, ..., n-1.$$  

$$0 \leq \sum_{i=0}^{m-1} C_k[i,j] = n, i = 0, 1, ..., n-1, j = 0, 1, ..., n-1.$$  

E.g., (1) means that at any interval, there must be enough agents to play a role; (2) says that an agent can only play one role at one interval; and (3) tells if there is role transfer for a specific agent and role.

$$\exists q, \psi \psi < r, q, \psi > \in g.e.B) \text{ and } \sum_{i=0}^{m-1} C_k[i,j] < q.\psi$$ means that at interval $r_0$ there are not enough agents playing $r_j$.

$$\sum_{i=0}^{m-1} C_k[i,j] = 0$$ means that at interval $r_0$ there is no any agent playing $r_j$.

Compared with the algorithm discussed in Section V, temporal role transfer needs to consider another parameter for role transfer, i.e., the scale $n$. That is to say, we need to check if a group is workable in the period and give a scheme for role transfer between the intervals to guarantee the group workable. Therefore, our problem transfers to a problem of finding a role transfer matrix sequences defined in Definition 6. For simplicity, we suppose that each role needs one current agent only.

**Input:** An $M \times N$ current role matrix C and an $M \times N$ potential roles matrix Q.

**Output:** Success - a sequence of matrices $C_0, C_1, ..., C_{n-1}$ and a scale $s$. Failure - a report for non workable group.

**Process:**

**Step 1:** Preparing the initial matrices (Note: “:=” means assignment from right to left and “−” means that the left is equal to the right):

1. Build and initialize $C_0, C_1, ..., C_{n-1}$, where $N = |R|$, it means that at the extreme situation, one agent can temporally play all the roles if the agent is qualified;
2. $C_0 := C$;
3. If in matrix $Z = Q + C_0$ there is one column with all zeros, i.e., $Z[i,j] = 0$ ($j = 0, 1, ..., k-1$) then report failure, i.e., there is a role that no any agent can play;
4. Else preparation completes, continues to Step 2.

**Step 2:** Build sequences $C_1, ..., C_{n-1}$ for role transfer:

1. Initialize the new matrix $C_{k} := C_{k-1}$;
2. Check every role if it has no current agent;
2.3: if (A role j without current agent is found)
   { Check the potential role matrix Q to find a potential agent;
     if (A potential agent i is found)
     { Find the current role of this agent.
       if (the current role k is found)
       { C[i, j] := s1; //It has one interval and not
         qualified to be transferred to.
       } else
       { C[i, j] := 1;
       }
     }
   }
   { C[i, j] := s+1; // It has one interval and not
     // qualified to be transferred to.
   }

Set all the row i of Q[i, k] with s+1;
// The agent is not potential for the next transfer in
//this round.
}

2.4: T := C + Cs-1 + . . . + C0;

   Check if there is a zero column in T;
   // Check if there is still a role having no agent to play.
   if (True)
   { s++;
     goto 2.1.
   }
else
   {Report success;
     Formulate C1s, C1s-1, . . . C10;
     Return C1s, C1s-1, . . . C10 and s;
   }

The algorithm guarantees that a sequence of matrices and a scale are found and that every role has its current agent in at least one interval. However, it is not balanced, i.e., some roles may have agents play in many (>1) intervals. Its complexity is \(O(n^2MN^3)\), where \(n=1+Max\{|a|R_a|, a\in A\}\), i.e., 1 plus the largest number of potential roles for an agent.

Interestingly, the algorithm of temporal role transfer is simpler than the algorithm of finding a successful role transfer. The reason is that we loosen the restriction of agent and role relationships. In the role transfer algorithm, we need to find an exact role-agent reassignment for all agents and roles. In the temporal algorithm, we only need to find some role-agent couples, and we allow some agent-role assignments in different intervals.

X. CONCLUSION

Role transfer is important in an organization. Specially and carefully assigning current and potential roles is a good and required preparation for crisis situations. At the same time, temporal role transfer is required when agents are not enough. Based on the revised E-CARGO model and matrices, we define the properties of role transfer design and implement the algorithms. With these algorithms, we can exactly recognize whether a group in a crisis is strong, check if a group is workable, and find a scheme of role transfer to keep a group to work. All the presented algorithms are implemented using Java and successfully run for many cases.

Our current algorithms deal with only the situation that one agent is required to play a role. It is valuable for us to find a more generalized algorithm to deal with the situation that many (>1) agents are required to play a role. Moreover, the complexity of the proposed algorithm is unsatisfactory for large systems. Heuristic algorithms are expected.

REFERENCES


