E-CARGO-Based Team Multi-Role Assignment Problem with Role Dependency

Wenan Tan¹, Wenyi Mao¹, Ruiling Gao¹, Jin Zhang¹, Weijia Zhuang¹, Qianwen Shen¹, Zixuan Li¹, Lida Xu¹, and Shengchun Sun¹

¹Affiliation not available

March 07, 2024
E-CARGO-Based Team Multi-Role Assignment Problem with Role Dependency

Wenan Tan, Wenyi Mao, Ruiling Gao, Jin Zhang, Weijia Zhuang, Qianwen Shen, Zixuan Li, Lida Xu, Fellow, IEEE, Shengchun Sun

Abstract—With the rapid development of collaborative computing, the Environments - Classes, Agents, Roles, Groups, and Objects (E-CARGO) model has been initially applied as a technique of Role-Based Collaboration (RBC). By extending the Group Multi-Role Assignment (GMRA) model after in-depth study on E-CARGO-based Group Role Assignment (GRA), a Team Multi-Role Assignment with Role Dependency (TMRARD) model is proposed with a higher degree of generalizability. The model aims to address how to effectively assign collaborative units (agents) to multiple roles for maximizing group performance and synergistic effect in the collaborative process with consideration of the role dependencies and the common team goals. By analyzing the characteristics and elements of the E-CARGO model, the TMRARD model is formally modeled based on E-CARGO. After analyzing the rationalization of the scale of role-dependent inputs, a Gurobi solution based on Mixed-Integer Linear Programming (MILP) has proposed and developed with consideration of the complexity of role-dependent constraints. Simulation experiments have verified the effectiveness and robustness of this method, demonstrating high performance in large-scale and complex constraint situations, in order to provide more scientific and efficient decision-making support for the field of collaborative computing.

Index Terms—Collaborative Computing, E-CARGO, Role Dependency, Team Multi-Role Assignment, Mixed-Integer Linear Programming (MILP), Gurobi Solver.

I. INTRODUCTION

With the vigorous development of collaborative computing, a variety of role assignment and task allocation methods have emerged, Tan et al. [1] proposed group-oriented task allocation in mobile crowdsensing, and Wang et al. [2] proposed differential privacy-based resource allocation, all of which are aimed at improving the efficiency of team collaboration, the quality of task execution, and the effectiveness of resource allocation. The Environments - Classes, Agents, Roles, Groups, and Objects (E-CARGO) model as an innovative technique of Role-Based Collaboration (RBC) [3] has injected new vigor into the field of collaborative computing. RBC, an innovative technology, has reinvigorated the field of collaborative computing. Role is a powerful concept that facilitates the management of distributed systems, and collaborative units (agents) accomplish collaborative tasks by playing roles. In this model, Group Role Assignment (GRA) [4] becomes an important task in RBC, which involves how to efficiently assign agents to multiple roles in order to maximize the group performance in the collaborative process. Combining real-world scenarios and specific problems, according to different constraints, Zhu et al. [5] extended the GRA model and extended several sub-models, such as Group Multi-Role Assignment (GMRA) [6], Group Role Assignment with Conflicting Agents (GRACA) [7], and Group Role Assignment with Cooperation and Conflict Factors (GRACCF) [8] and so on. According to the multi-optimization objective, Zhu et al. [9] also extended GRA to Group Role Assignment with Budget Constraints (GRABC) [10] and so on. However, from the above sub-models, all the existing researches mainly focus on the conflict and cooperation relationship between agents, but ignore the possible dependent interactions between the roles. In addition, the group is just a collection of a group of people, which tends to focus only on individual goals, ignoring the characteristic of teamwork that emphasizes common goals. In this paper, we further extend GMRA by proposing a Team Multi-Role Assignment with Role Dependency (TMRARD) model to better adapt to the constraint complexity in real-world scenarios and optimize the team synergistic effect.

The main contributions of this paper are:

- Completing the formal modeling of a more generalized engineering problem, TMRARD, and classifying it as a Mixed-Integer Linear Programming (MILP) problem.

- To add the input of strong role dependency matrix to GMRA, and propose to solve undirected graph passing closures based on Floyd-Warshall algorithm in graph theory to seek all pairs of weak role dependency except those with strong dependencies. To optimize
the common team goal, agent-role preference matrix with synergy weights is introduced.

- Reasonable and desirable input scales for each parameter are analyzed considering the feasible solution of the problem and close to realistic scenario requirements.
- For the constraints related to role dependency, the solution performance comparison between the Gurobi solver and the default CBC solver invoked by the PuLP library in Python proves that the use of Gurobi has a significant advantage in the solution performance.

The aim of this paper is to provide more scientific and efficient decision support in the field of collaborative computing, to promote the further evolution of the E-CARGO model, and to provide more feasible guidelines for collaborative interactions for managers in various industries.

II. RELATED WORK

A. RBC and E-CARGO

Collaboration is required when a task needs to be accomplished by more than one person, and the desired task execution results pose new challenges to the specific collaboration process. Role-based Collaboration (RBC), a general computing approach that uses roles as core components [11], is a specific strategy for organizing and managing collaboration among participants (i.e., agents), where each agent is assigned to one or more roles, which, in turn, define its responsibilities and behaviors in a collaborative environment. The fundamental issues of RBC research and its methodology were first articulated by Zhu [3], which aims to facilitate intra- and inter-group collaboration by clarifying roles, establishing norms, and enhancing efficiency. Ng et al. [12] developed a quality service framework to support various types of users in web-based collaborative systems. This suggests service allocation as a potential application area. The lifecycle flowchart of RBC is shown in Figure 1.

As the basic model of RBC, the Environments - Classes, Agents, Roles, Groups and Objects (E-CARGO) model was first proposed by Zhu [14], which is highly abstracted from collaborative systems. It brings a new vision for role assignment in collaborative environments by dividing the system into six classes. Among them, the relationship between classes can be formalized into mathematical formulas, which enables many complex interaction problems to be solved. Zhang et al. [15] carried out a study on online social networks using E-CARGO as a theoretical background. Teng et al. [16] introduced constraint sets into E-CARGO to solve the classical CSP constraint problem. It can be seen that the E-CARGO model can be modified and supplemented to solve many complex engineering problems.

B. GRA and GMRA

Group Role Assignment (GRA) involves assigning specific roles to different members of a group in order to collaborate well and achieve goals. It involves assigning individuals to appropriate roles based on their abilities and experience to ensure that the group performs efficiently and successfully. As a crucial part of the RBC lifecycle, GRA seeks to find the optimal allocation scheme by assessing the value of the agent in each role, which is seen to be a task that significantly affects the efficiency and satisfaction of the entire RBC process. Zhang et al. [17] proposed GRA with a training program, where the role assignments are based on an updated qualification assessment matrix and derive the benefits generated by the training program. Wang et al. [18] introduced the minority game strategy into the role assignment problem of a specific team game. It can be seen that all of the above studies mainly focus on agent assessment and role switching aspects, while the role assignment problem and its solution also need to be focused and studied.

As an extension of GRA, Group Multi-Role Assignment (GMRA) extends from the original "one-to-many" to "many-to-many", i.e., agents need to be assigned to one or more roles in order to adapt to complex task requirements and diverse group functions. This involves a finer-grained task assignment, so that each member can take on different roles according to the specific needs within the limits of his/her own ability, in order to improve the flexibility and collaborative efficiency of the whole group. Zhu et al. [6] formalized the GMRA problem, and gave an improved solution for the IBM ILOG CPLEX optimization packages. Liu et al. [19] applied the GMRA to the task assignment of a tree structure. Liang et al. [20] applied GMRA to solve team role assignment in service crowdsourcing. It can be seen that GMRA largely
dependencies, i.e., weak dependencies, are created. If a transmissibility of the dependencies, more indirect managers and front-end engineers, and between product managers and back-end engineers, between back-end engineers and database administrators, between product managers and front-end engineers, and between product managers and back-end engineers. Depending on the transmissibility of the dependencies, more indirect dependencies, i.e., weak dependencies, are created. If a member is asked to perform two positions with dependencies, and the member's qualifications in these two positions are assessed to be too different, it will inevitably affect the global result output. A simple example can be given that if the group performance is optimized only according to the constraints defined in GMRA, and a member with a large difference in front-end and back-end development execution capabilities has to be allowed to perform a full stack development [21] combining front-end and back-end with strong dependencies, it will inevitably lead to serious consequences such as poor team collaboration, poor user experience, and project delays due to asynchrony. In order to describe the existence of direct dependencies between jobs, the list of positions with direct dependencies shown in Table IV is used.

Assignments are constrained by setting thresholds for differences in qualification assessments in the case of strong and weak dependencies, i.e., if a member's qualification assessment for both positions exceeds the corresponding thresholds when assigning positions with different levels of dependency, he/she cannot be assigned to both positions.

In addition, efficient team synergy cannot be achieved by considering only individual competencies. Learning as much

### III. TMRARD FORMAL MODELING

#### A. Real-world Scenario

In order to describe the TMRARD problem graphically, let's take an example by envisioning a real-life scenario in the IT industry: a software service team has 16 members ($a_0$-$a_{15}$), and each member can perform multiple positions. The company's executives have decided to assign two front-end engineers ($r_0$), three back-end engineers ($r_1$), one database administrator ($r_2$), two test engineers ($r_3$), one operations and maintenance engineer ($r_4$), and one product manager ($r_5$) to complete a software project, as shown in Table I.

Before assignment, each member's ability to perform in each position was assessed to obtain the qualification assessment table shown in Table II, where the rows represent the members and the columns represent the positions, and each value in the table quantitatively reflects the member's qualification ability in each position. One of the objectives of job assignment is to maximize the sum of the selected employees' qualification assessments in their executive positions. The job assignment schematic is shown in Figure 2.

However, each member has a work limit, i.e., the maximum number of positions that can be performed, depending on their respective capabilities, as shown in Table III. In order to guarantee the effectiveness of project completion, it is necessary to ensure that the number of jobs assigned to each member cannot exceed its limit.

Based on the above scenario, we can follow the initial steps of RBC and solve it as a GMRA problem. However, the scenario clearly ignores a common situation, and we describe the extended scenario as follows: assume that there are direct dependencies, i.e., strong dependencies, between front-end engineers and back-end engineers, between back-end engineers and database administrators, between product managers and front-end engineers, and between product managers and back-end engineers. Depending on the transmissibility of the dependencies, more indirect dependencies, i.e., weak dependencies, are created.
As possible about the preferences and interests of team members and assigning positions based on these characteristics is an important step in building an effective, cohesive team. Team members are more likely to feel satisfied as possible about the preferences and interests of team members and assigning positions based on these characteristics is an important step in building an effective, cohesive team. Team members are more likely to feel satisfied when they are given positions of their own. Cohesive team. Team members are more likely to feel satisfied when they are given positions of their own. Maximizing the synergistic effect of satisfying members' preferences and interests is an important step in building an effective, cohesive team. Team members are more likely to feel satisfied when they are given positions of their own. Therefore, job assignment also needs to consider the common goals of the team.

Admittedly, at the time of assignment, the factor of personal preference. Therefore, job assignment also needs to consider the common goals of the team. Therefore, job assignment also needs to consider maximizing the synergistic effect of satisfying members' preferences, as a way to elevate individual goals in the cluster to the common goals of the team.

Admittedly, at the time of assignment, the factor of personal preference. Therefore, job assignment also needs to consider maximizing the synergistic effect of satisfying members' preferences, as a way to elevate individual goals in the cluster to the common goals of the team.

As a result, we further extend the GMRA problem to form the TMRARD sub-model that introduces role-dependent constraints and synergies to describe this common situation.

### B. E-CARGO-Based Secondary Modeling

In order to complete the relevant formulation of formal modeling, we first briefly introduce the relevant definitions and elements of the E-CARGO model. E-CARGO [22] abstracts the collaborative system as a 9-tuple, i.e., \( \Sigma := \langle C, O, A, M, R, E, G, s_0, H \rangle \), with parameter notation meanings as shown in Table VI.

Any group in the system should accomplish collaborative tasks in a specific environment. E-CARGO is an evolving model that can be cut from different aspects and used as a theoretical background to solve real engineering problems. In order to model the problem formally more easily, the following notational definitions are performed:

**Definition 1:** Let the non-negative integer \( m \) be the number of elements (number of agents) in the set \( A \), i.e., \( m = |A| \). Let the non-negative integer \( n \) be the number of elements (number of roles) in the set \( R \), i.e., \( n = |R| \). Let the non-negative integer \( i \) be the index of the element in the set \( A \), and the non-negative integer \( j \) be the index of the element in the set \( R \).

**Definition 2:** The \( n \)-vector \( L \) is used to represent the number of agents required for each role in the environment \( e \) group \( g \).

**Definition 3:** A qualification assessment matrix \( Q \) of size \( m \times n \) is used to quantitatively reflect the execution capability of each agent on each role, where \( Q[i,j] \in [0,1] \) means that the qualification assessment values of agent \( i \) on role \( j \) are all in the range of 0 to 1, as shown in Figure 3.

**Definition 4:** An assignment matrix \( T \) of size \( m \times n \) is used to indicate whether an agent is assigned to a role or not, where \( T[i,j] \in \{0,1\} \), as shown in Figure 4, where 0 means unassigned and 1 means assigned.

**Definition 5:** The \( m \)-vector \( L^a \) is used to represent the maximum number of roles that can be assigned to each agent in the environment \( e \) group \( g \).

**Definition 6:** The overall team performance value \( \sigma \) is used to represent the sum of qualification assessment of the assigned agents in the environment \( e \) group \( g \), i.e., \( \sigma = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i,j] \times T[i,j] \).

**Definition 7:** For role \( j \) to be feasible in the environment \( e \)
Fig. 3. Qualification assessment matrix.

\[
Q = \begin{pmatrix}
0.69 & 0.84 & 0.61 & 0.75 & 0.35 & 0.22 \\
0.91 & 0.63 & 0.67 & 0.26 & 0.63 & 0.53 \\
0.05 & 0.60 & 0.61 & 0.49 & 0.69 & 0.09 \\
0.44 & 0.49 & 0.85 & 0.85 & 0.36 & 0.41 \\
0.62 & 0.33 & 0.09 & 0.08 & 0.89 & 0.31 \\
0.98 & 0.45 & 0.42 & 0.03 & 0.56 & 0.58 \\
0.78 & 0.78 & 0.91 & 0.81 & 0.44 & 0.80 \\
0.97 & 0.37 & 0.55 & 0.15 & 0.36 & 0.09 \\
0.94 & 0.21 & 0.67 & 0.20 & 0.76 & 0.23 \\
0.42 & 0.14 & 0.41 & 0.03 & 0.78 & 0.94 \\
0.73 & 0.88 & 0.60 & 0.98 & 0.82 & 0.56 \\
0.52 & 0.61 & 0.78 & 0.87 & 0.65 & 0.07 \\
0.47 & 0.67 & 0.75 & 0.40 & 0.98 & 0.58 \\
0.15 & 0.03 & 0.55 & 0.68 & 0.32 & 0.06 \\
0.17 & 0.64 & 0.62 & 0.65 & 0.02 & 0.35 \\
0.17 & 0.51 & 0.50 & 0.18 & 0.35 & 0.91 \\
\end{pmatrix}
\]

Fig. 4. Assignment matrix.

\[
T = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

(a) Strong role dependency matrix
(b) Final role dependency matrix

Fig. 5. Role Dependency Matrix.

group \( g \), it must be assigned enough agents, i.e., \( \sum_{i=0}^{m-1} T[i,j] \geq L[j] \).

Definition 8: A strong role dependency matrix \( D \) of size \( n \times n \) is used to indicate whether there is a direct dependency between roles, where \( D[j_1,j_2] \in \{0,1\} \), i.e., 0 means no direct dependency, and 1 means a direct dependency. For the reasonableness of the description, let \( D[j_1,j_1] \equiv 0 \), i.e., the main diagonal element is always 0. Also, since dependencies have interactions, i.e., if role \( j_1 \) is strongly dependent on role \( j_2 \), then role \( j_2 \) must also be strongly dependent on role \( j_1 \).

Therefore, the list of jobs with direct dependencies shown in Table IV can be mapped to the strong role dependency matrix \( D \) shown in Figure 5(a), which is a matrix symmetric about the main diagonal.

Definition 9: The final role dependency matrix \( C^o \) of size \( n \times n \) used to represent the strong role dependency matrix \( D \) after the algorithm to find all the weak dependencies derived from dependency transmissibility, where \( C^o[j_1,j_2] \in \{0,1,2\} \), i.e., 0 means no dependency, 1 means strong dependency, and 2 means weak dependency. Similarly, let \( C^o[j,j] \equiv 0 \), i.e., the main diagonal element is constant 0. As shown in Figure 5(b), \( C^o \) is also a matrix symmetric about the main diagonal.

Definition 10: Let one decimal \( t_h \) and \( t_w \) denote the maximum qualification difference thresholds under strong and weak dependency conditions in the environment \( e \) group \( g \) respectively, where \( t_h, t_w \in (0,1) \), \( t_h < t_w \), i.e., in the assignment of roles with different degrees of dependency, if the qualification difference of an agent in the two roles exceeds the corresponding thresholds, the agent can’t be assigned to the two roles at the same time.

Definition 11: An agent-role preference matrix \( P \) of size \( m \times n \) is used to quantitatively reflect the degree of each agent’s preference for each role, where \( P[i,j] \in (0,1) \) means that agent \( i \)'s preference values on role \( j \) are all in the range of 0 to 1, as shown in Figure 6.

Definition 12: Let one decimal \( col_eff \) denote the weight of synergies in the environment \( e \) group \( g \), \( col_eff \in (0,1) \), i.e., the importance of synergies in the whole project.

Based on the above definition, it is not difficult to conclude that the goal of the TMRARD problem is to find a feasible assignment matrix \( T \) to

\[
\max \sigma = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (Q[i,j] + col_eff \times P[i,j]) \times T[i,j] \\
\text{s.t.} \ T[i,j] \in \{0,1\} \quad (0 \leq i < m, 0 \leq j < n) \\
\sum_{i=0}^{m-1} T[i,j] = L[j] \quad (0 \leq j < n) \\
\sum_{j=0}^{n-1} T[i,j] \leq L^q[i] \quad (0 \leq i < m) \\
T[i,j_1] + T[i,j_2] \leq 1 \\
(0 \leq i < m, 0 \leq j_1 < n, 0 \leq j_2 < n, \quad (5) \\
C^o[j_1,j_2] = 1 \text{ and } |Q[i,j_1] - Q[i,j_2]| > t_h) \\
T[i,j_1] + T[i,j_2] \leq 1 \\
(0 \leq i < m, 0 \leq j_1 < n, 0 \leq j_2 < n, \quad (6) \\
C^o[j_1,j_2] = 2 \text{ and } |Q[i,j_1] - Q[i,j_2]| > t_w)
\]
expression (4) allows the agent to be assigned to a limited number of roles, and expressions (5) and (6) ensure that the agent will not be assigned to both roles when the difference in the qualification assessments on the two roles with strong/weak dependencies exceeds the corresponding threshold.

When $Q$ is the qualification assessment matrix shown in Figure 3, $C^0$ is the final role dependency matrix shown in Figure 5(b), and $P$ is the agent-role preference matrix shown in Figure 6, Figure 4 shows the assignment matrix $T$ subject to $L = [2,3,1,2,1,1]$, $L_w = [1,1,2,3,2,2,3,1,2,2,1,1,1,1]$, $th_s = 0.1$ , $th_w = 0.3$ , col ef $f = 0.2$ , and the assignment result is shown in Figure 2, when the final overall team performance value $\sigma$ is 8.90.

We continue the initial solution of GMRA [6] by representing the matrices $Q$, $P$ and $T$ as vectors, thus transforming TMRARD into a Mixed-Integer Linear Programming (MILP) problem. The solution is thus accomplished for it using popular Python-based solution tools.

### C. Solving Undirected Graph Passing Closures Based on the Floyd-Warshall Algorithm

The Floyd-Warshall algorithm [23] is a commonly used algorithm in graph theory for obtaining the shortest paths between all nodes, and the same can be applied to compute the passing closure [23], [24]. Passing closure of an undirected graph means that if there exists a path between every pair of nodes in the graph, then there is an edge between these two nodes. This can be described by a matrix, where the elements $(i, j)$ indicate whether a path exists between node $i$ and node $j$ or not. Thus, the problem of finding all weak dependencies between roles with strong dependencies by the transitive nature of the dependencies can be mapped to the logical idea of solving an undirected graph passing closure, i.e., treating the set of roles as the set of points $V$, and the set of dependencies as the set of edges $E$. Although solving an undirected graph passing closure can also be done using either a Depth First Search or Breadth First Search, the gradual increase in the scale of the experiment necessitates iterating $|V|$ times, resulting in inefficiency. The Floyd-Warshall algorithm, on the other hand, incorporates the idea of dynamic programming, which can effectively improve efficiency.

The process of solving the undirected graph passing closure based on the Floyd-Warshall algorithm is shown in Figure 7 with the following steps:

**Step 1:** Initialize the matrix. Create an adjacency matrix describing the undirected graph, where the elements $(i, j)$ equals to 1 means that there is an edge between node $i$ and node $j$, and 0 means that no edge is connected.

**Step 2:** Applying the Floyd-Warshall algorithm. Using the standard steps of the algorithm, we iteratively update the elements of the matrix to find the shortest path between all nodes. In this process, we not only focus on the shortest path length, but also focus on whether there is a path between the nodes, see Algorithm 1 for details.

**Step 3:** Get the transfer closure matrix. After the iteration is

---

**Algorithm 1 Transitive Closure Based on Floyd-Warshall**

**Input:** Graph $G$

**Output:** Matrix $C$

1: let $n$ be the number of vertices in $G$
2: let $C$ be a matrix of size $n \times n$
3: for each edge $(u, v)$ in $G$ do
4:     $C[u][v] = 1$
5: end for
6: for $k \leftarrow 1$ to $n$ do
7:     for $j \leftarrow 1$ to $n$ do
8:         $C[i][j] = \max(C[i][j], C[i][k] * C[k][j])$
9:     end for
10: end for
11: end for
12: return $C$

---

Fig. 7. Illustration of the Floyd-Warshall solution process.
completed, the obtained adjacency matrix is the transfer closure matrix of the undirected graph, where the elements \((i, j)\) is equal to 1 means that there is a path between node \(i\) and node \(j\), and 0 means that there is no path connected.

After inputting the strong role dependency matrix \(D\) shown in Figure 5(a), the final role dependency matrix \(C^0\) shown in Figure 5(b) can be obtained by solving the Floyd-Warshall algorithm to find all pairs of roles with strong/weak dependencies.

### IV. MILP-BASED GUROBI SOLUTION

#### A. Input Scale Rationalization Analysis

According to the problem definition, TMRARD is partially inherited from the GMRA model, and Zhu et al. [6] have proved that the necessary condition for GMRA to have a feasible solution is \(\sum_{a=0}^{m-1} L^a [i] \geq \sum_{j=0}^{n-1} L[j]\). In short, if the resources are insufficient, i.e., the total number of members required for the roles is greater than the sum of the maximum number of roles that can be executed by the agent, the problem will lose its feasible solution. In the input of the simulation experiment, we fully consider and try to avoid this situation.

After extending the GMRA problem, the role-dependent constraints largely increase the complexity of the problem, which is NP-hard to obtain sufficient necessary conditions for the problem to have a feasible solution. The simulation experiments under completely random numbers are prone to not finding a solution during the solution process, i.e., there is a constraint conflict or the problem has no solution. It is also possible that the problem does not converge and the optimal solution cannot be found in the specified time due to the scale is too large. In view of the complexity brought about by role-dependent constraints, we need to fully study the reasonableness of the input parameters first, i.e., the input range of each parameter under different problem scales. The convergence of the initiator can be facilitated by a reasonably clever initialization of the parameters, the most important of which involves role dependencies. Through the sensitivity analysis in the early stage, we find that the input scale of role dependency significantly affects the solution efficiency and the availability of feasible solutions: if the dependency degree of the whole team is too high, or even through the dependency transmissibility to form a closed-loop in the team, in addition to not quite conforming to the practical scenarios, it will also result in the solution time is too long, and too many constraints will lead to the loss of feasible solutions for the problem. If the degree of dependency is too low, with only one or two pairs of dependencies in a team of a certain size, this will inevitably result in the solution performance tending to be a GMRA problem, and dilute the effect of the solution performance under the role-dependent constraints.

Based on these considerations, we conducted an input scale analysis of role dependency with the number of agents between 30 and 200, and the hardware and software environment and solver standard library for the experiment are shown in Table VII. Since each input parameter has great sensitivity with the expansion of the problem scale, the problem scale was divided into several layers and analyzed layer by layer according to the number of agents, and 50 groups were randomly generated in each layer for iteration to collect the average coverage of the role dependency after iteration as well as the ratios that can reflect the effect of the role-dependent constraints. Among them, the average coverage rate refers to the average ratio of the number of pairs of non-zero elements in the matrix \(C^0\) obtained from the matrix \(D\) through the transfer closure algorithm to the total number of pairs of roles. Since the process of incorporating the constraints into the solver are all carried out in the solver, it can only be judged by using the same set of randomly-generated input parameters, respectively, for the comparison of the optimal overall performance values before and after the introduction of the role-dependent constraints, to determine whether they can reflect the effect of the role-dependent constraints or not.

In order to satisfy the necessary condition that GMRA has feasible solutions as much as possible, we set the ratio of \(L^a\) to the upper limit of the random number of \(L\) to 5:3, continuing the ratio of 2:1 between the number of agents and the number of roles in the GMRA performance experiments [6]. Taking the median number of agents per layer as a benchmark, the metrics ratios for different inputs with strong role dependency on the number of pairs are shown in Figure 8. Where (a) is the metrics ratio at \(m = 40, L = 5, L^a = 8\), (b) is the metrics ratio at \(m = 75, L = 9, L^a = 15\), (c) is the metrics ratio at \(m = 125, L = 15, L^a = 25\), and (d) is the metrics ratio at \(m = 175, L = 21, L^a = 35\).

Combining the previous experiments and real-world scenarios, we found that when the average coverage of all dependencies reaches more than 90\%, it will be very likely that the problem will lose feasible solutions due to too many pairs of roles with strong dependencies.
role dependencies, and this condition is used as an upper bound for the scale of the role-dependent inputs. In order to differentiate GMRA and highlight the solution performance effect after the introduction of role-dependent constraints, we consider that when the average ratio of the same set of input parameters that are different for the optimal overall performance values derived before and after the introduction of role-dependent constraints respectively reaches 50%, i.e., the results are different in 25 out of 50 iterations, the input role-dependent scale can basically reach the condition of embodying the effect of the role-dependent constraints, and we use this as a lower bound on the scale for the role-dependent inputs. If the above range is satisfied, the more obvious the effect of role-dependent constraints is, the more favorable the performance comparison is. The experimental results on ideal input scale for each layer are shown in Table VIII.

**B. Mixed-Integer Linear Programming and Gurobi Solver**

Based on the completed formal modeling, when the discrete nature of the decision variables for assigning agents to roles can be modeled by introducing binary integers, the value of maximizing the overall team performance can be represented by a linear objective function, and the linear constraints can be modeled by Linear Programming (LP), we transform the qualification assessment matrix $Q$, the agent-role preference matrix $P$ and the assignment matrix $T$ into vectors, and TMRARD forms a typical LP problem. We introduce integer variables, which are further refined and reduced to the Mixed Integer Linear Programming (MILP) [25] problem, containing the properties of integer optimization problems. In MILP, some or all of the decision variables are restricted to integer values, while others can still take arbitrary real values. Thus from this property, TMRARD is well suited to use the idea of MILP with the help of its dedicated optimization solvers such as CPLEX, Gurobi, etc., all of which are able to efficiently deal with integer variable constraints and search for integer solutions using algorithms such as branch-and-bound, cut plane, and so on, and

![Fig. 8. Indicator ratios under pairs of strongly dependent roles with different inputs.](image-url)
are powerful enough to deal with the mixed nature of discretization and continuity that characterizes many real-world problems.

We first chose the traditional Python library PuLP [26] for modeling and solving linear programming problems and invoked its default open-source solver CBC. Although it embodies convenience and ease-of-use, performance becomes a major limitation due to the significant increase in problem complexity with the introduction of role-dependent constraints, coupled with the growing data scale. The performance effects of TMRARD solved by CBC according to the ideal input scale of each layer shown in Table VIII are shown in Figure 9.

It can be seen that if TMRARD still uses the default CBC of the traditional PuLP library for solving, the time used increases significantly with the increase of data scale. Therefore, on large-scale complex problems, performance will become a major bottleneck in CBC solving. Gurobi [27], on the other hand, is a high-performance optimization solver that can be used to solve a wide range of application areas, and its powerful performance, efficient algorithms, friendly user interface, and a wide range of application areas, and its performance in solving large-scale complex optimization problems is particularly outstanding.

### C. Gurobi Solving and Performance Experiment

In recent years, the Python language has become increasingly popular for big data simulation experiments. Call Gurobi from the gurobipy standard library, which is a commercial solver but free for academic use. When using the Gurobi Python API, you can model and solve optimization problems directly in Python without using the Gurobi Modeling Language (GML) or a compiler, resulting in better performance. Before doing the Python simulation experiments, you need to transform the objective function and constraint function into the form required by Gurobi. The process is described as follows:

1. **Step 1:** Create an optimization model object: 
   ```python
   model = gp.Model("TMRARD_Model")
   ```

2. **Step 2:** To add an objective function, call the following method in Python:
   ```python
   model.setObjective(gp.quicksum(self.Q[i][j] + self.col_eff * self.P[i][j] for i in range(self.m) for j in range(self.n)), GRB.MAXIMIZE)
   ```

3. **Step 3:** To add a constraint function, call the following method in Python:
   1. For the 0-1 constraint on the elements of the assignment matrix embodied in expression (2):
      ```python
      T = []
      for i in range(self.m):
          row = []
          for j in range(self.n):
              row.append(int(x[i, j].x))
          T.extend(row)
      ```
   2. For the role constraint embodied in expression (3):
      ```python
      for j in range(self.n):
          m.addConstr(gp.quicksum(x[i, j] for i in range(self.m)) == self.LA[j], f"role_constraint_{j}")
      ```
   3. For the agent constraint embodied in expression (4):
      ```python
      for i in range(self.m):
          m.addConstr(gp.quicksum(x[i, j] for j in range(self.n)) <= self.LA[i], f"agent_constraint_{i}")
      ```
   4. For the role dependency and qualification assessment discrepancy constraints embodied in expressions (5) and (6):
      ```python
      for i in range(self.m):
          for j1 in range(self.n):
              for j2 in range(self.n):
                  if self.C[j1][j2] == 1 and abs(self.Q[i][j1] - self.Q[i][j2]) > self.th_s: m.addConstr(x[i, j1] + x[i, j2] <= 1, f"dependency_constraint_{i}_{j1}_{j2}"")
                  elif self.C[j1][j2] == 2 and abs(self.Q[i][j1] - self.Q[i][j2]) > self.th_w: m.addConstr(x[i, j1] + x[i, j2] <= 1, f"dependency_constraint_{i}_{j1}_{j2}"")
      ```

Thus, The Python language modeling based on Gurobi solution is completed, in order to verify the efficiency of the Gurobi solution, we compared its performance with that of the default CBC solver invoked by the PuLP library for solving the TMRARD problem under the same input conditions, the hardware and software environments for the experiments and the standard libraries of the solver are shown in Table VII. The number of Agents is taken between 40 and 200 with an increment of 20, and each increment randomly generates 50 sets of input parameters for iteration, and the generation rules are shown in Table VIII. The difference threshold $th_b$ for strong dependency qualification assessment is 0.1, the difference threshold $th_w$ for weak dependency qualification assessment is 0.3, and the synergy weight $col_eff$ is 0.2. The comparison results of the solver performance are shown in Figure 10.
It is not difficult to find that Gurobi exhibits faster solving speed under the same input conditions. In order to present the actual effect of Gurobi’s optimized solution more clearly, we use the result obtained by the formula $\frac{\text{time}_{\text{CBC}} - \text{time}_{\text{Gurobi}}}{\text{time}_{\text{CBC}}}$ to express the percentage of solution time saved, and it can be concluded that it generally saves about 80% of the solution time. Not only that, Gurobi exhibits greater stability and ability to handle large-scale problems, i.e., the difference in solving performance is relatively small for different numbers of agents. In contrast, the CBC solver shows a relatively large increase in solving time as the number of agents increases.

In summary, the optimization problem of team multi-role assignment with complex constraints like TMRARD is more suitable using MILP-based ideas and using Gurobi as a solution.

V. CONCLUSION

This paper provides an in-depth discussion of the E-CARGO-based Team Multi-Role Assignment with Role Dependency (TMRARD) problem. The Group Role Assignment (GRA) problem is investigated to extend its problem domain, and a TMRARD model that introduces role-dependent constraints and common team goals is proposed. The model takes into account the strong and weak dependencies between roles and seeks to achieve effective intra-team role assignment in a multi-role environment to maximize group performance and synergistic effect in the collaborative process.

A Gurobi solution scheme is proposed for extending E-CARGO model based on Mixed-Integer Linear Programming (MILP). Where considers the introduced role-related dependency complexity, the relatively ideal and reasonable input scales for each parameter are discussed layer by layer, and the effectiveness and robustness of the scheme are verified by performance comparison. The experimental results show that in large-scale and complex constraint scenarios, using the Gurobi solver will have more significant performance advantages over the default CBC solver invoked by the traditional PuLP library in solving the optimization problem of team multi-role assignment with complex constraints, such as TMRARD, and will be able to provide more scientific and efficient decision-making support in real engineering problems.

At the same time, we realize that there are still many aspects worth exploring in depth in future research. First, we can consider introducing a more complex role dependency model that is close to the actual scenarios to more realistically reflect the collaborative coexistence of team members. Second, we can study how execution units in dynamic environments can dynamically collaborate and support each other to accomplish team autonomy. In addition, more optimization algorithms and heuristics can be explored to improve the efficiency of problem solving for practical applications.

In conclusion, this paper provides new perspectives and solution ideas for the team multi-role assignment problem and group collaboration, and expects that future research can further promote the development of this field and provide more innovative and feasible solutions for the wide application of collaborative computing.

REFERENCES


