文章编号: 1001-4543(2024)04-0406-11 DOI: 10.19570/j.cnki.jsspu.2024.04.008

基于 E-CARGO 的角色依赖的团队多角色分配问题研究

毛文逸[a], 谭文安[a], 王依婷[b], 张 金[a], 沈倩文[a], 李子璇[a]

(上海第二工业大学 a. 计算机与信息工程学院; b. 信息技术中心, 上海 201209)

摘 要: 随着协同计算的不断进步, 环境 - 类、Agents、角色、组和对象 (environments-classes, Agents, roles, groups and objects, E-CARGO) 模型作为一种基于角色协同 (role-based collaboration, RBC) 技术受到了广泛关注。通过对基于 E-CARGO 的组多角色分配 (group multi-role assignment, GMRA) 模型开展研究, 提出了一个普适度更高的角色依赖的团队多角色分配 (team multi-role assignment with role dependency, TMRARD) 模型, 涉及到引入角色依赖关系和团队共同目标后, 如何将协作单元 (Agents) 有效分配给角色, 以最大化群体表现与协同效应。将 TMRARD 形式化建模, 针对角色依赖约束的复杂性, 对依赖输入规模进行合理性分析, 设计出基于混合整数线性规划 (mixed-integer linear programming, MILP) 的 Gurobi 求解策略, 并用仿真实验验证了该方法的有效性与鲁棒性, 在大规模和复杂约束情形下性能显著, 以期为协同计算的决策支持提供新的思路。

关键词: 协同计算; 角色依赖; 团队多角色分配; 混合整数线性规划; Gurobi 求解器

中图分类号: TP399 **文献标志码**: A

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

MAO Wenyi[a], TAN Wen'an[a], WANG Yiting[b], ZHANG Jin[a], SHEN Qianwen[a], LI Zixuan[a]

(a. School of Computer and Information Engineering; b. Information Technology Center, Shanghai Polytechnic University, Shanghai 201209, China)

Abstract: With the continuous progress of collaborative computing, the environments-classes, Agents, roles, groups and objects (E-CARGO) model has received much attention as a role-based collaboration (RBC) technique. By carrying out research on the E-CARGO-based group multi-role assignment (GMRA) model, a team multi-role assignment with role dependency (TMRARD) model with higher pervasiveness is proposed, which involves how to efficiently assign collaborative units (Agents) to roles to maximize group performance and synergistic effects after the introduction of role-dependency relationships and team common goals. The TMRARD is formally modeled, the scale of dependent inputs is rationally analyzed with respect to the complexity of role-dependent constraints, and the Gurobi solution strategy based on mixed-integer linear programming (MILP) is designed, and simulation experiments are conducted to verify the validity and robustness of the method, which has a significant performance under large-scale and complex constraints, with a view to providing a new way of thinking for the decision-making support of collaborative computing.

Keywords: collaborative computing; role dependency; team multi-role assignment; mixed-integer linear programming; Gurobi solver

收稿日期: 2024-06-20

通信作者: 谭文安 (1965–), 男, 湖北公安人, 教授, 博士, 主要研究方向为协同计算与系统进化、协同学习与知识管理等。
E-mail: watan@sspu.edu.cn

基金项目: 国家自然科学基金项目 (61672022, U1904186), 上海市研究生教育学会研究课题 (ShsgeG202207) 资助

0　引言

随着协同计算的蓬勃发展, 出现了多种角色分配和任务指派方法, Tan 等 [1] 提出了群智感知中面向群组的任务分配, Wang 等 [2] 提出了基于差异隐私的资源分配, 这些方法都旨在提高团队协作效率、任务执行质量和资源分配效果。基于环境-类、Agents、角色、组和对象 (environments-classes, Agents, roles, groups and objects, E-CARGO) 模型作为基于角色协同 (role-based collaboration, RBC) [3,11] 的创新技术, 为协同计算领域注入了新的活力。在该模型中, 组角色分配 (group role assignment, GRA) [4] 成为 RBC 中的一项重要任务, 涉及如何有效地将 Agents 分配给多个角色, 以最大化协同过程中的群体表现。结合现实场景和具体问题, 根据不同约束条件, Zhu [5] 从 GRA 延伸出多个子模型, 如组多角色分配 (group multi-role assignment, GMRA) [6]、冲突 Agents 的组角色分配 (group role assignment with conflicting Agents, GRACA) [7]、合作和冲突因素的组角色分配 (group role assignment with cooperation and conflict factors, GRACCF) [8] 等。根据多优化目标, 文献 [9-10] 中又将 GRA 延伸为预算约束的组角色分配 (group role assignment with budget constraints, GRABC) 等。然而现有研究均主要聚焦于 Agents 之间的冲突与合作关系, 忽略了角色之间可能存在依赖的相互作用。此外, 群组只是一组人的集合, 往往只看重个体目标, 而忽略了团队协作的共同目标。本文扩展了 GMRA, 提出了角色依赖的团队多角色分配 (team multi-role assignment with role dependency, TMRARD) 模型, 以更好地适应现实场景中的约束复杂性, 并优化了团队协同效应。

本文的主要贡献有: ① 完成普适度更高的工程问题 TMRARD 的形式化建模, 并将其归为混合整数线性规划 (mixed-integer linear programming, MILP) 问题。② 增加角色强依赖矩阵的输入, 提出用图论中 Floyd-Warshall 算法求无向图传递闭包, 来寻求除了具有强依赖关系以外的所有弱依赖关系的角色对。为了优化团队共同目标, 引入 Agent- 角色偏好矩阵与协同效应权值。③ 考虑问题可行解和贴近现实场景要求分析了各参数合理且理想的输入规模。④ 针对角色依赖的相关约束, 通过使用 Gurobi 求解器与 Python 中 PuLP 库调用的默认 Coin-or Branch and Cut (CBC) 求解器进行性能对比, 证明了 Gurobi 在求解性能上有明显优势。

本文旨在推动 E-CARGO 的进一步演进, 为各行业管理者提供更可行的协同交互指南。

1　相关工作

1.1　RBC 与 E-CARGO

RBC 是一种将角色作为核心组件的通用计算方法 [11], 是组织和管理参与者 (即 Agents) 协作的具体策略。基本问题及其方法最早由 Zhu [3] 阐明, 旨在通过明确角色、建立规范和提升效率来促进组内与组间合作。Ng 等 [12] 建立了一个质量服务框架, 以支持网络协作系统中的各类用户; 文献 [13-14] 分别基于 RBC 建模了多偏好平衡性指派和社区服务型时空众包任务指派问题。由此表明, 服务分配是其潜在应用领域。RBC 生命周期流程如图 1 所示。

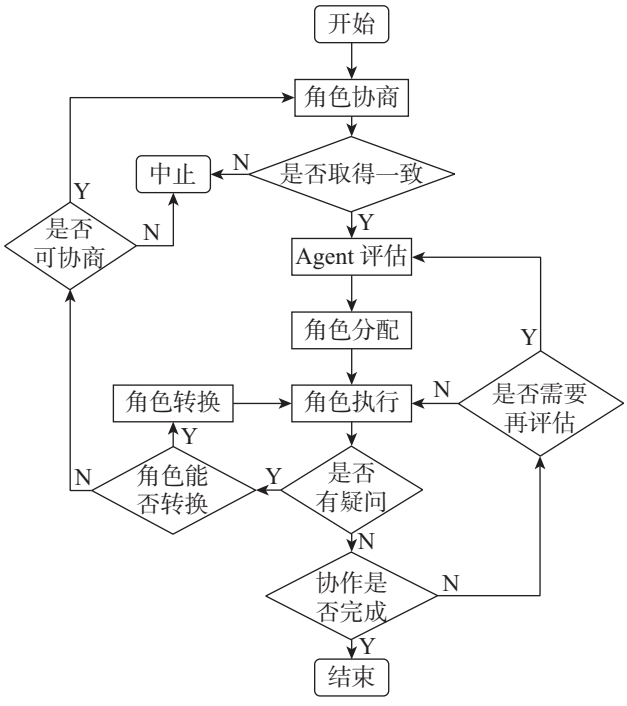

图 1　RBC 生命周期流程图

Fig. 1　Lifecycle flowchart of RBC

作为 RBC 的基本模型, E-CARGO 由 Zhu [15] 提出, 通过将系统划分为 6 大类, 为协同环境的角色分配带来了新的愿景。文献 [16-17] 中以 E-CARGO 为理论背景对在线社会网络和机场登机口调度开展了详细研究; 滕少华等 [18] 将约束集引入 E-CARGO 以求解经典 CSP 约束问题。可见, 适当修正和补充 E-CARGO, 可解决诸多复杂的工程问题。

1.2 GRA 与 GMRA

作为 RBC 生命周期中至关重要的一环, GRA 力求通过 Agents 在各角色上的资格评估来找到最优分配方案, 显著影响 RBC 过程的效率与满意度。Zhang 等 [19] 提出融入培训计划的 GRA, 角色分配基于更新的资格评估矩阵得出培训计划产生的效益; Wang 等 [20] 将少数博弈策略引入特定团队游戏的角色分配问题。然而上述研究都聚焦于 Agents 评估和角色转换, 忽略了角色分配问题及其解决方案。

作为 GRA 的延伸模型, GMRA 从原来的 “一对多” 扩展为 “多对多”, 以便适应复杂的任务需求与多元化的群体职能。这涉及到更细粒度的任务分配, 使各成员在自身能力极限内, 根据具体需求承担不同角色。Liu 等 [21] 将 GMRA 应用于树形结构的任务指派; Liang 等 [22] 将 GMRA 应用于解决服务众包中的团队角色分配。由此可见, GMRA 极大提升了角色分配问题的适应性与灵活性。考虑到角色间存在依赖约束以及团队共同目标下的自治协同问题, 通过扩展 GMRA, 提出了 TMRARD 以提高普适度。

2 TMRARD 形式化建模

2.1 现实场景

为了形象描述 TMRARD, 设想一个现实场景来举例说明: 某软件服务团队共 16 名成员 ($a_0 \sim a_{15}$), 每名成员可执行多个岗位, 公司高层决定指派前端工程师 (r_0) 2 名、后端工程师 (r_1) 3 名、数据库管理员 (r_2) 1 名、测试工程师 (r_3) 2 名、运维工程师 (r_4) 1 名和产品经理 (r_5) 1 名来完成一项软件项目。

指派前, 先评估各成员在各岗位上的执行能力, 得到如表 1 所示的资格评估表, 其中行代表成员, 列代表岗位。岗位指派的目标之一即为最大化被选中员工在其执行岗位上资格评估的总和, 从而最大化协同过程中的群体表现。岗位指派示意图如图 2 所示。

表 1 资格评估表
Tab. 1 Qualification assessment table

	r_0	r_1	r_2	r_3	r_4	r_5
a_0	0.69	0.84	0.61	0.75	0.35	0.22
a_1	0.91	0.63	0.67	0.26	0.63	0.53
a_2	0.05	0.60	0.61	0.49	0.69	0.09
a_3	0.44	0.49	0.85	0.85	0.36	0.41
a_4	0.62	0.33	0.09	0.08	0.89	0.31
a_5	0.98	0.45	0.42	0.83	0.56	0.58
a_6	0.78	0.78	0.91	0.81	0.44	0.80
a_7	0.97	0.37	0.55	0.15	0.36	0.09
a_8	0.94	0.21	0.67	0.20	0.76	0.23
a_9	0.42	0.14	0.41	0.03	0.78	0.94
a_{10}	0.73	0.88	0.60	0.98	0.82	0.56
a_{11}	0.52	0.61	0.78	0.87	0.65	0.07
a_{12}	0.47	0.67	0.75	0.40	0.98	0.58
a_{13}	0.15	0.03	0.55	0.68	0.32	0.86
a_{14}	0.17	0.64	0.62	0.65	0.02	0.35
a_{15}	0.17	0.51	0.50	0.18	0.35	0.91

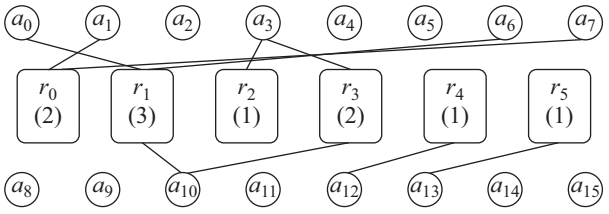

图 2 团队多角色分配图例
Fig. 2 Example of team multi-role assignment

然而, 每个成员因其能力不同都有一个工作极限, 即可执行的最大岗位数, 如表 2 所示。为了保障项目完成效果, 必须保证分配的岗位数不能超出其极限。

表 2 成员可执行的最大岗位数
Tab. 2 Maximum numbers of positions a member can perform

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
1	1	2	3	2	2	2	3	1	2	2	2	1	1	1	1

遵循 RBC 的初始步骤, 将上述场景视为 GMRA 予以解决。然而该场景显然忽略了一种常见情形, 对扩展后的场景描述如下: 假设前后端工程师、后端工程师和数据库管理员、产品经理和前端工程师、产品经理和后端工程师之间存在直接依赖关系, 即强依赖。根据依赖的传递性, 还会产生更多的间接依赖关系, 即弱依赖。如果让某成员执行两个具有依赖关系的岗位, 而该成员在这两个岗位上的资

格评估差异过大, 势必会影响全局成果输出。比如, 若仅按 GMRA 中定义的约束来最优化群体表现, 不得不让一个前后端开发执行能力差异很大的成员从事具有强依赖的前后端相结合的全栈开发 [23], 势必造成团队协作不畅, 用户体验不佳, 项目因不同步而延期等严重后果。为了描述这种直接依赖关系, 用具有直接依赖关系的岗位表来表示 (见表 3)。

表 3　具有直接依赖关系的岗位表
Tab. 3　List of positions with direct dependencies

	r_0	r_1	r_2	r_3	r_4	r_5
r_0		√				√
r_1	√		√			√
r_2		√				
r_3						
r_4						
r_5	√	√				

通过设定资格评估差异阈值来对指派做出约束, 即在不同依赖程度下的岗位指派时, 若某成员在这两个岗位上的资格评估超出相应阈值, 则不能被同时分配。

此外, 高效的团队协同不能仅考虑个体能力。当团队成员被赋予各自偏好的岗位时, 他们更可能感到满意和投入, 从而更努力地达到共同目标。经过偏好调研得到如表 4 所示的成员 - 岗位偏好表, 其中行代表成员, 列代表岗位, 表中数值量化反映了

表 4　成员-岗位偏好表
Tab. 4　Member-position preference table

	r_0	r_1	r_2	r_3	r_4	r_5
a_0	0.2	0.7	0.9	0.5	0.3	0.5
a_1	0.7	0.3	0.9	0.4	0.4	0.9
a_2	0.7	0.3	0.8	0.2	0.8	0.1
a_3	0.2	0.7	0.4	0.8	0.8	0.5
a_4	0.9	0.4	0.3	0.4	0.5	0.1
a_5	0.3	0.1	0.8	0.7	0.7	0.5
a_6	0.4	0.8	0.7	0.6	0.5	0.6
a_7	0.4	0.6	0.9	0.3	0.7	0.1
a_8	0.1	0.5	0.6	0.3	0.6	0.2
a_9	0.9	0.2	0.6	0.7	0.3	0.2
a_{10}	0.5	0.5	0.6	0.1	0.8	0.5
a_{11}	0.2	0.8	0.6	0.5	0.9	0.2
a_{12}	0.1	0.6	0.4	0.8	0.7	0.3
a_{13}	0.5	0.6	0.6	0.9	0.9	0.9
a_{14}	0.3	0.5	0.8	0.7	0.3	0.6
a_{15}	0.4	0.2	0.8	0.7	0.7	0.3

成员对各岗位的偏好程度。所以, 岗位指派的目标还需最大化成员偏好所带来的协同效应, 以此将群组个体目标上升到团队共同目标。

诚然, 在指派时, 资格评估因素必定重要于个体偏好因素。若资格评估的权重默认为 1, 则必须设定一个小于 1 的权重来确定协同效应的重要程度, 以灵活应对具体情形。

2.2　基于 E-CARGO 的二次建模

E-CARGO [24] 将协作系统抽象为九元组, 即 $\sum ::= \langle C, O, A, M, R, E, G, s_0, H\rangle$, 参数符号含义如表 5 所示。

表 5　E-CARGO 参数符号含义
Tab. 5　Meaning of the E-CARGO parameter symbols

参数符号	含义	紧耦合集
C	类集合	A 和 H 为紧耦合集, E 和 G 为紧耦合集
O	对象集合	
A	Agents 集合	
M	消息集合	
R	角色集合	
E	环境集合	
G	群组集合	
s_0	系统初始状态	
H	用户集合	

该系统中的任何群组都应在特定环境中协同。为了便于形式化建模, 本文延续了文献 [6] 中的定义 1~7, 简单说明如下: n 维向量 $\boldsymbol{L}$ 表示每个角色所需的 Agent 数量; 大小为 $m \times n$ 的资格评估矩阵 $\boldsymbol{Q}$ 量化反映每个 Agent 在各角色上的执行能力; 大小为 $m \times n$ 的分配矩阵 $\boldsymbol{T}$ 表示 Agents 是否被分配给角色; m 维向量 $\boldsymbol{L}^a$ 表示每个 Agent 所能被分配的最大角色数量; 团队整体表现值 σ 表示被分配 Agents 的资格评估总和, 即 $\sigma = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \boldsymbol{Q}[i,j] \times \boldsymbol{T}[i,j]$; 要使角色 j 在环境 e 群组 g 中可行, 必须分配给它足够多的 Agents, 即 $\sum_{i=0}^{m-1} \boldsymbol{T}[i,j] \geqslant L[j]$。

同时, 增加新的定义如下:

定义 8　非负整数 m 为 Agent 数量, 即 $m = |A|$; 非负整数 n 为角色数量, 即 $n = |R|$; 非负整数 i 为集合 A 中元素的索引; 非负整数 j 为集合 R 中元素的索引。

定义 9　大小为 $n \times n$ 的角色强依赖矩阵 $\boldsymbol{D}$ 表示角色间是否具有直接依赖关系, 其中

$\boldsymbol{D}[j_1, j_2] \in \{0, 1\}$, 即 0 表示无直接依赖, 1 表示有直接依赖。为了描述的合理性, 令 $\boldsymbol{D}[j_1, j_2] \equiv 0$, 即主对角线元素恒为 0。又因为依赖关系具有相互作用, 若角色 j_1 强依赖于角色 j_2, 则角色 j_2 也必强依赖于角色 j_1。如图 3(a) 所示角色强依赖矩阵 $\boldsymbol{D}$, 是一个关于主对角线对称的矩阵。

定义 10 大小为 $n \times n$ 的最终角色依赖矩阵 $\boldsymbol{C^D}$ 表示角色强依赖矩阵 $\boldsymbol{D}$ 经过算法找到由依赖传递性得出的所有弱依赖, 其中 $\boldsymbol{C^D}[j_1, j_2] \in \{0, 1, 2\}$, 即 0 表示无依赖, 1 表示强依赖, 2 表示弱依赖。同理, 令 $\boldsymbol{C^D}[j_1, j_1] \equiv 0$。如图 3(b) 所示, $\boldsymbol{C^D}$ 也是一个关于主对角线对称的矩阵。

$$\boldsymbol{D}=\begin{pmatrix}0&1&0&0&0&1\\1&0&1&0&0&1\\0&1&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\1&1&0&0&0&0\end{pmatrix} \qquad \boldsymbol{C^D}=\begin{pmatrix}0&1&2&0&0&1\\1&0&1&0&0&1\\2&1&0&0&0&2\\0&0&0&0&0&0\\0&0&0&0&0&0\\1&1&2&0&0&0\end{pmatrix}$$

(a) 角色强依赖矩阵 (a) Strong role dependency matrix

(b) 最终角色依赖矩阵 (b) Final role dependency matrix

图 3 角色依赖矩阵

Fig. 3 Role dependency matrix

定义 11 一位小数 th_s 和 th_w 分别表示强依赖和弱依赖条件下的最大资格差异阈值, 其中 th_s, $\mathrm{th}_w \in (0, 1)$, $\mathrm{th}_s < \mathrm{th}_w$, 即若 Agent 在这两个角色上的资格差异超出相应阈值, 则不能被同时分配。

定义 12 大小为 $m \times n$ 的 Agent- 角色偏好矩阵 $\boldsymbol{P}$ 量化反映各 Agent 对各角色的偏好程度, 其中 $\boldsymbol{P}[i, j] \in (0, 1)$, 如图 4 所示。

$$\boldsymbol{P}=\begin{pmatrix}0.2&0.7&0.9&0.5&0.3&0.5\\0.7&0.3&0.9&0.4&0.4&0.9\\0.7&0.3&0.8&0.2&0.8&0.1\\0.2&0.7&0.4&0.8&0.8&0.5\\0.9&0.4&0.3&0.4&0.5&0.1\\0.3&0.1&0.8&0.7&0.7&0.5\\0.4&0.8&0.7&0.6&0.5&0.6\\0.4&0.6&0.9&0.3&0.7&0.1\\0.1&0.5&0.6&0.3&0.6&0.2\\0.9&0.2&0.6&0.7&0.3&0.2\\0.5&0.5&0.6&0.1&0.8&0.5\\0.2&0.8&0.6&0.5&0.9&0.2\\0.1&0.6&0.4&0.8&0.7&0.3\\0.5&0.6&0.6&0.9&0.9&0.9\\0.3&0.5&0.8&0.7&0.3&0.6\\0.4&0.2&0.8&0.7&0.7&0.3\end{pmatrix}$$

图 4 Agent-角色偏好矩阵

Fig. 4 Agent-role preference matrix

定义 13 一位小数 col_eff 表示协同效应权值, col_eff $\in (0, 1)$, 即协同效应在整个项目中的重要程度。

根据以上定义, 不难得出 TMRARD 的目标即为寻求一个可行的分配矩阵 $\boldsymbol{T}$, 使得

$$\max \sigma = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(\boldsymbol{Q}[i,j] + \text{col\_eff} \times \boldsymbol{P}[i,j]) \times \boldsymbol{T}[i,j] \tag{1}$$

$$\boldsymbol{T}[i,j] \in \{0,1\}, \quad 0 \leqslant i < m; 0 \leqslant j < n \tag{2}$$

$$\sum_{i=0}^{m-1} \boldsymbol{T}[i,j] = \boldsymbol{L}[j], \quad 0 \leqslant i < n \tag{3}$$

$$\sum_{j=0}^{n-1} \boldsymbol{T}[i,j] \leqslant \boldsymbol{L^a}[j], \quad 0 \leqslant j < m \tag{4}$$

$$\boldsymbol{T}[i,j_1] + \boldsymbol{T}[i,j_2] \leqslant 1, \quad 0 \leqslant i < m; 0 \leqslant j_1 < n; \quad 0 \leqslant j_2 < n, \quad \boldsymbol{C^D}[j_1,j_2] = 1 \text{ and } |\boldsymbol{Q}[i,j_1] - \boldsymbol{Q}[i,j_2]| > \mathrm{th}_s; \tag{5}$$

$$\boldsymbol{T}[i,j_1] + \boldsymbol{T}[i,j_2] \leqslant 1, \quad 0 \leqslant i < m; 0 \leqslant j_1 < n, \quad 0 \leqslant j_2 < n, \quad \boldsymbol{C^D}[j_1,j_2] = 2 \text{ and } |\boldsymbol{Q}[i,j_1] - \boldsymbol{Q}[i,j_2]| > \mathrm{th}_w \tag{6}$$

其中, 式 (2) 为 $\boldsymbol{T}$ 中元素的 0~1 约束, 式 (3) 使群组可行, 式 (4) 使 Agents 被分配给有限数量的角色, 式 (5)、(6) 为角色依赖相关约束。

2.3 基于 Floyd-Warshall 算法求无向图传递闭包

Floyd-Warshall 算法 [25] 是图论中一种用于得到所有节点之间最短路径的常用算法, 可以将其应用于计算传递闭包 [25-26]。将强依赖由依赖的传递性找到所有弱依赖的问题, 映射为求解无向图传递闭包的逻辑思路, 即把角色集视为点集 V, 把依赖关系集视为边集 E。虽然求解无向图传递闭包也可使用深度或广度优先遍历, 但由于实验规模的逐步扩大而不得不迭代 $|V|$ 次, 导致效率低下, Floyd-Warshall 算法则融入了动态规划的思想, 能有效提高效率。Floyd-Warshall 与深度或广度优先遍历算法计算效率对比实验分析如表 6 所示。

基于 Floyd-Warshall 算法求解无向图传递闭包的过程如图 5 所示。迭代更新矩阵中的元素, 以求出所有节点之间的最短路径。该过程不仅关注最短路径长度, 还关注节点之间是否存在路径, 算法伪码如图 6 所示。

表 6　Floyd-Warshall 与深度或广度优先遍历算法计算效率对比实验分析

Tab. 6　Experimental analysis of efficiency comparison between Floyd-Warshall and depth or breadth-first search algorithms

图规模 (节点数)	图密度 (边数/节点数)	深度或广度优先遍历运行时间/s	Floyd-Warshall 运行时间/s	相对加速比
100	1.5	0.02	0.005	4.0
500	2.0	0.25	0.08	3.125
1 000	2.5	1.2	0.16	7.5
5 000	3.0	15.6	2.1	7.429
10 000	4.0	60.1	8.2	7.329

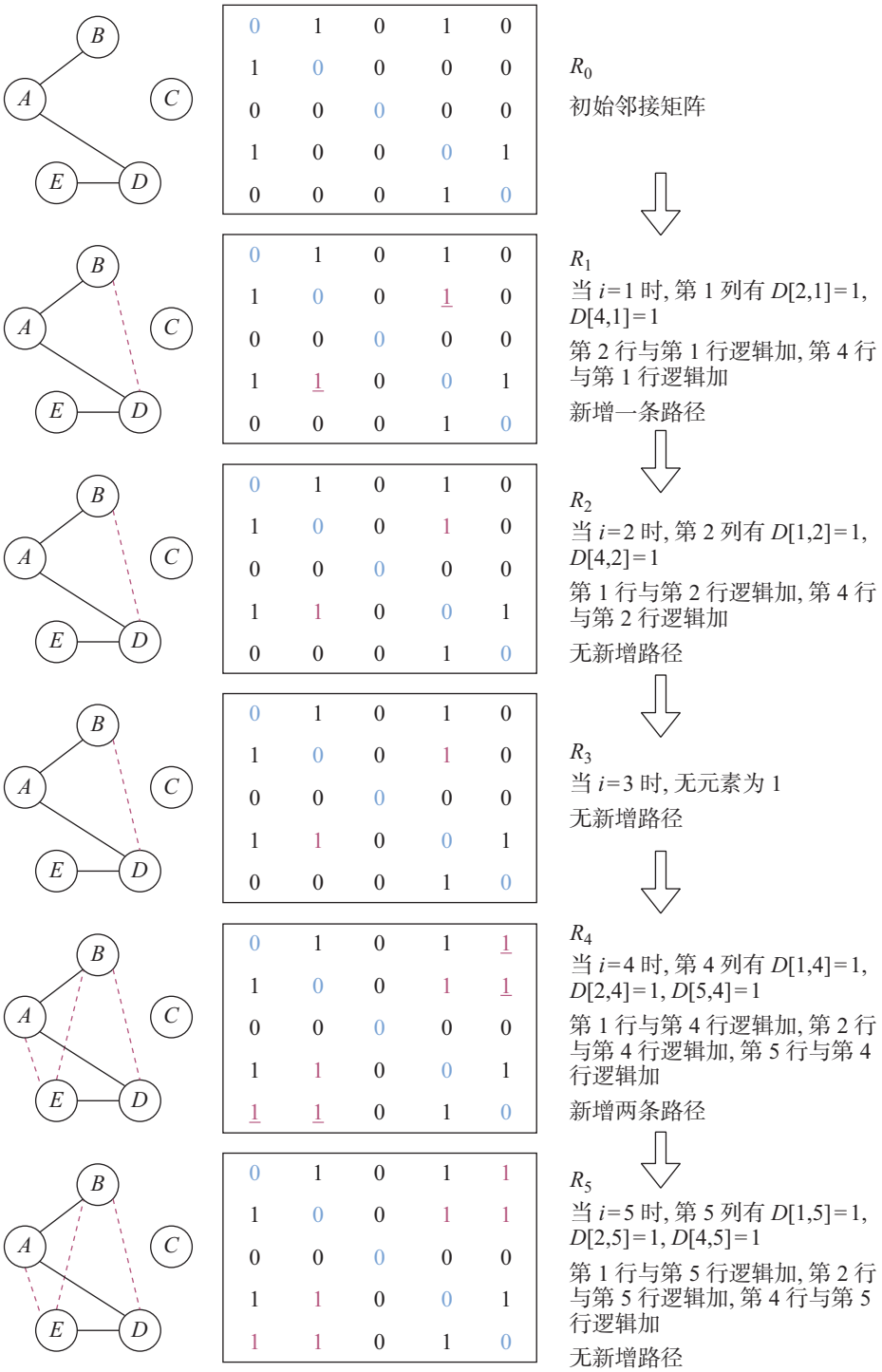

图 5　Floyd-Warshall 求解过程图例

Fig. 5　Illustration of the Floyd-Warshall solution process

Algorithm 1 Transitive Closure Based on Floyd-Warshall
Input: Graph *G*
Output: Matrix *C* which represents the transitive closure of the graph

```
let n be the number of vertices in G
let C be a matrix of size n×n
for each edge (u, v) in G do
    C[u][v] = 1
end for
for k←1 to n do
    for i←1 to n do
        for j←1 to n do
            C[i][j] = max(C[i][j], C[i][k] * C[k][j])
        end for
    end for
end for
return C
```

图 6 Floyd-Warshall 算法伪码
Fig. 6 Pseudo-code of Floyd-Warshall

将图 3(a) 所示的角色强依赖矩阵 $\boldsymbol{D}$ 输入后, 经过 Floyd-Warshall 算法求解就可以得到图 3(b) 所示的最终角色依赖矩阵 $\boldsymbol{C^D}$, 从而找到所有具有强/弱依赖关系的角色对。

3 基于 MILP 的 Gurobi 求解策略

3.1 输入规模合理性分析

根据问题定义, TMRARD 部分继承于 GMRA, Zhu 等 [6] 已证明 GMRA 有可行解的必要条件为 $\sum_{i=0}^{m-1} \boldsymbol{L^a}[i] \geqslant \sum_{j=0}^{n-1} \boldsymbol{L}[j]$。在仿真实验的输入时, 需尽量避免发生该情况。如表 7 所示, 先使用穷举法对问题求解时间进行了初步探索, 发现团体规模将极大程度影响求解时间。显然穷举法是不可行的, 因为即便是小规模团体也需要耗费相当长的求解时间, 当 Agent 数量达到 60, 角色数量达到 30 时, 5 min 内无法得到可行解。

表 7 对问题求解时间的初步探索
Tab. 7 An initial exploration of problem solving time

Agent 数量	角色数量	穷举法求解		
		$t_{\min}$/ms	$t_{\max}$/ms	t_{average}/ms
20	10	1 590.3	12 736.9	5 085.6
40	20	19 281.3	192 036.9	66 204.6
60	30	N/A	N/A	N/A

此外, 角色依赖的约束条件极大增加了问题复杂性, 它是 NP 难的, 无法得到问题有可行解的充要条件。完全随机数下的仿真实验容易造成约束冲突或问题无解; 也可能由于规模过大, 问题未收敛而无法在规定时间内找到最优解。鉴于角色依赖约束带来的复杂性, 需充分研究输入参数取值范围的合理性。前期的灵敏度分析发现角色依赖的输入规模显著影响求解效率和问题有无可行解: 若整个团队依赖程度过高, 甚至经过依赖传递性在团队中形成闭环, 除了不太符合实际场景外, 还会造成求解时间过长, 约束过多而导致问题失去可行解; 若依赖程度过低, 在一定规模的团队中仅有 1∼2 对依赖, 势必会造成求解性能趋于 GMRA, 而淡化了角色依赖约束下的求解性能效果。

而后进行 Agent 数量介于 30∼200 之间的角色依赖的输入规模分析, 实验的软硬件环境及求解器标准库如表 8 所示。根据 Agent 数量将问题规模划为多层并逐层分析, 每层随机生成 50 组进行迭代, 收集角色依赖的平均覆盖率以及能体现角色依赖约束效果的比率。其中, 平均覆盖率是指由矩阵 $\boldsymbol{D}$ 经过传递闭包算法得到的矩阵 $\boldsymbol{C^D}$ 中非零元素的对数占总角色对数的平均比率。由于融入约束条件的求解过程均在求解器中进行, 因此只能通过用同一组随机生成的输入参数, 分别对引入角色依赖约束前后得出的最优团队表现值进行对比, 来判断是否能体现角色依赖约束效果。

表 8 实验环境与求解器标准库
Tab. 8 Experimental environment and standard solver library

环境及标准库	指标	技术参数
硬件环境	处理器 CPU	Intel(R) Core(TM) i5-1035G1 CPU@1.00 GHz 1.20 GHz
	机带 RAM	16.0 GB
软件环境	操作系统 OS	Windows 10 专业版
	PyCharm	PyCharm 2022.3
	Python	Python 3.11
解器标准库	PuLP	PuLP 2.7.0
	gurobipy	gurobipy 11.0.0

为了尽量满足 GMRA 有可行解的必要条件, 将 L^a 与 L 的随机数上限之比设为 5∶3, 延续了 GMRA 性能实验 [6] 中 Agent 数和角色数之间 2∶1 的比例。以每层 Agent 数量的中间值为基准, 不同输入的强依赖角色对数下的指标比率如图 7 所示。其中, 图

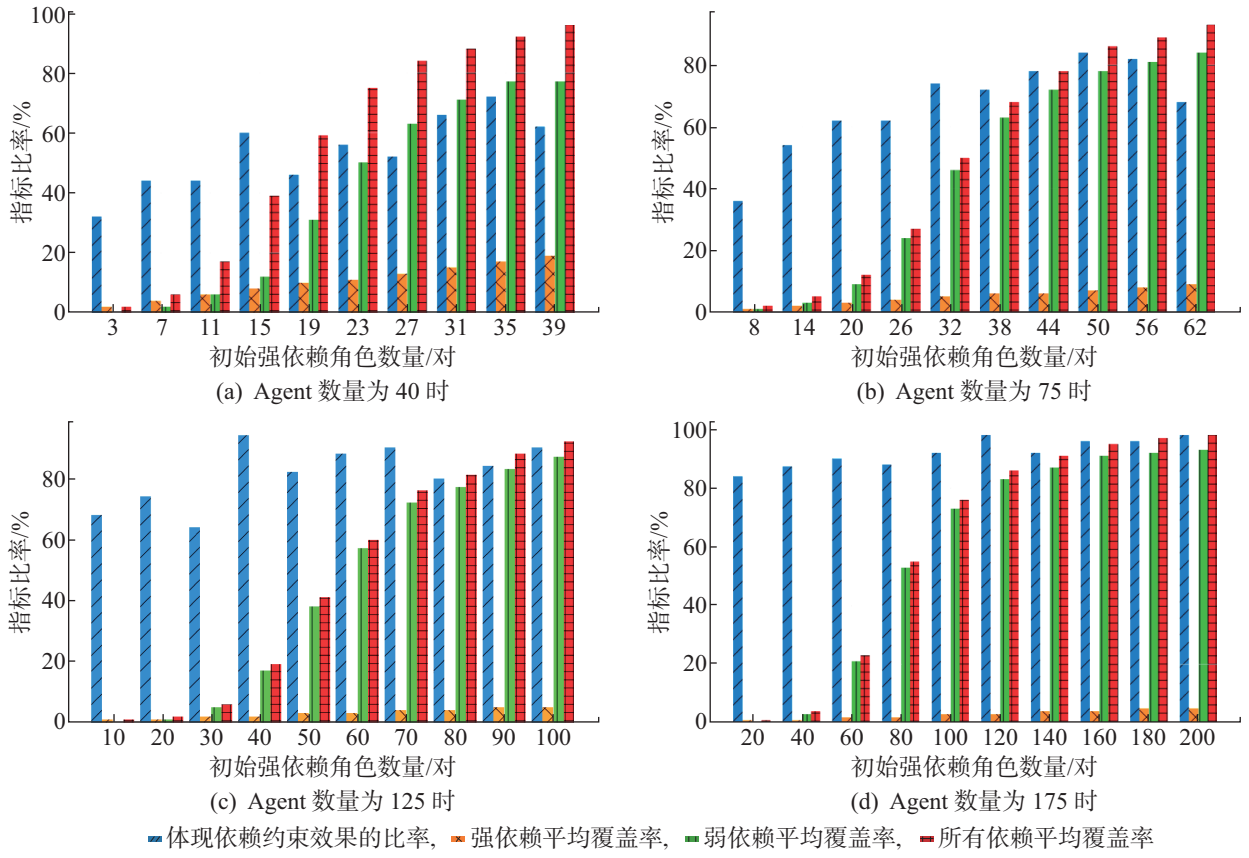

图 7　不同输入的强依赖角色对数下的指标比率

Fig. 7　Indicator ratios under pairs of strongly dependent roles with different inputs

7(a) 为 $m = 40, L = 5, L^a = 8$ 时的指标比率; 图 7(b) 为 $m = 75, L = 9, L^a = 15$ 时的指标比率; 图 7(c) 为 $m = 125, L = 15, L^a = 25$ 时的指标比率; 图 7(d) 为 $m = 175, L = 21, L^a = 35$ 时的指标比率。

结合前期实验和现实场景发现, 当所有依赖的平均覆盖率达到 90% 以上, 将极有可能因角色依赖过多而失去可行解, 并以此条件作为角色依赖输入规模的上界; 为了凸显引入角色依赖约束后的求解性能效果, 本文认为, 当同一组输入参数分别对引入角色依赖约束前后得出的最优团队表现值不同的平均比率达到 50%, 则输入的角色依赖规模能基本达到体现角色依赖约束效果的条件, 并以此作为角色依赖输入规模的下界。在满足以上范围的情况下, 角色依赖约束效果越明显则越有利于性能对比。每层的理想输入规模实验结果如表 9 所示。

表 9　理想输入规模实验结果

Tab. 9　Experimental results on ideal input scale

Agent 数量	角色数量	$\boldsymbol{L}$ 随机数上限	$\boldsymbol{L}^a$ 随机数上限	强依赖对数
30∼50	15∼25	5	8	25
51∼100	26∼50	9	15	50
101∼150	51∼75	15	25	75
151∼200	76∼100	21	35	100

3.2　MILP 与 Gurobi 求解器

根据完成的形式化建模, 当矩阵 $\boldsymbol{Q}$、$\boldsymbol{P}$ 和 $\boldsymbol{T}$ 转换为向量, 再引入整数变量后, 就可将 TMRARD 归结为 MILP [27] 问题, 决策变量被限制为整数值, 而其他变量仍可取任意实数值。借助于专用的优化求解器, 能有效处理整数变量约束, 其强大之处在于能够处理许多实际问题中的离散性与连续性混合的特性。本文选用了传统的用于 MILP 建模和求解的 Python 库 PuLP [28], 并调用其默认开源求解器 CBC, 求解性能如图 8 所示。

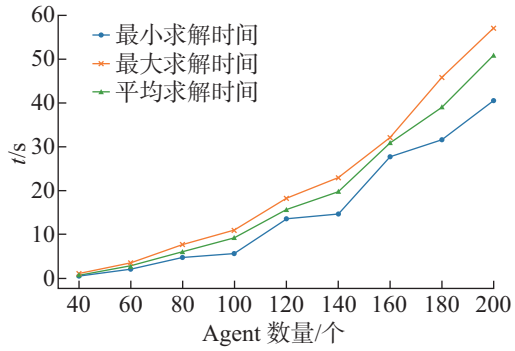

图 8 CBC 求解 TMRARD 性能效果

Fig. 8 CBC solutions for TMRARD performance effects

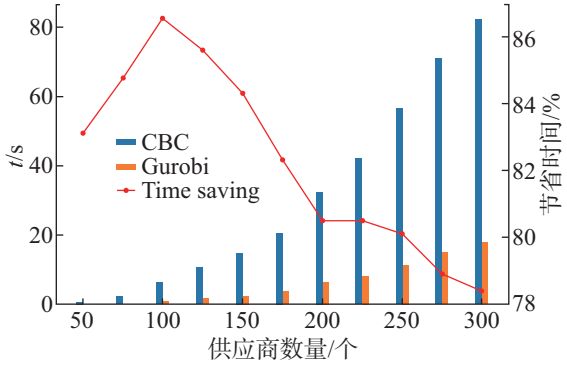

图 9 CBC 与 Gurobi 求解性能对比

Fig. 9 Comparison of the solving performances of CBC and Gurobi

当 Agent 数量达到 100 时, 单次求解时间均在 10 s 以上, 当 Agent 数量达到 200 时, 单次求解时间更是长达 50 s。因此大规模和复杂约束情形严重延长了求解耗时, 性能将成为 CBC 求解的一大瓶颈。而 Gurobi [29] 正是一种高性能优化求解器, 具有强大的性能、高效的算法、友好的用户接口以及广泛的应用领域, 在解决大规模、复杂的优化问题上的性能表现尤为突出。从 gurobipy 标准库中调用 Gurobi, 在使用 Gurobi Python API 时, 可以直接在 Python 中建模与求解, 而无需使用 Gurobi 的建模语言 (Gurobi modeling language, GML) 或编译器, 从而获得更好的性能。为了验证 Gurobi 解决方案的高效性, 将其与 CBC 求解器在同等输入条件下求解 TMRARD 问题的性能进行对比。取 Agent 数量在 40~200 之间, 增量为 20, 每个增量随机生成 50 组输入参数进行迭代。th_s 为 0.1, th_w 为 0.3, col_eff 为 0.2。求解性能对比结果如图 9 所示。

当 Agent 数量达到 100 时, CBC 的平均耗时接近 10 s, 50 组迭代后耗时接近 10 min; 当 Agent 数量达到 200 时, 平均耗时接近 1 min, 50 组迭代后耗时更是接近 1 h。相反, 在同等输入条件下, Gurobi 表现出了更快的求解速度。为了更清晰地呈现其实际效果, 根据公式 $\frac{t_{\mathrm{CBC}}-t_{\mathrm{Gurobi}}}{t_{\mathrm{CBC}}}$ 可知普遍能节省约 80% 的求解时间。不仅如此, Gurobi 表现出更强的稳定性和处理大规模问题的能力。

3.3 仿真对比实验

目前已有诸多 MILP 问题的解法, 其中线性规划松弛法和分支定界法较为经典。线性规划松弛法 [30] 将整数约束松弛为连续约束, 求解得到的线性规划问题, 将结果四舍五入或采用启发式方法修正, 使其满足整数约束; 分支定界法 [31] 通过递归地分裂问题, 构造一个搜索树, 并使用上下界来修剪不可能包含最优解的分支。为了验证基于 MILP 的 Gurobi 求解方案的有效性, 本文将上述两种解法与常见的启发式算法贪心法 (Greedy) 进行了对比, 实验结果如表 10 所示。

表 10 不同问题规模的仿真实验对比结果

Tab. 10 Comparative results of simulation experiments with different problem sizes

Agent 数量	角色数量	Greedy		MILP- 分支定界 -Gurobi		MILP- 线性规划松弛 -Gurobi	
		团体最优表现	平均求解时间/ms	团体最优表现	平均求解时间/ms	团体最优表现	平均求解时间/ms
40	20	22.54	2.6	55.54	127.8	55.68	101.3
60	30	37.52	8.4	138.2	450.5	138.43	442.5
80	40	50.77	18.8	180.96	840	181.4	1 008.6
100	50	63.16	36	228.69	1 361.9	229.38	1 274.9
120	60	80.35	75.5	450.68	3 502.9	452.18	2 552.9
140	70	94.9	117.7	517.51	4 732.2	519.91	3 540.6
160	80	112.43	229.4	828.6	7 310.7	831.61	6 219.3
180	90	124.71	328.7	917.93	9 796.2	922.06	8 416.4
200	100	141.59	430.9	1 051.59	12 519.2	1 057.28	10 030.8

由表中数据可见, 不同方法求得的团队最优表现值差异较大。由于 Greedy 一般无法达到全局最优, 其目标值总体较低, 而 MILP 问题求解方法的目标值均较高, 特别是随着问题规模的增大, 差异更加显著。线性规划松弛法和分支定界法相比, 求解效果差异不大, 但总体而言, 前者求得的目标值更大, 而且求解时间更短。因此基于 MILP 的 Gurobi 求解方案适用于大规模和复杂约束问题, 虽然牺牲了一定的求解时间, 但求得目标值更好, 能提供更优的求解方案。

4 结　论

本文拓展 GMRA 模型, 提出了一种引入角色依赖约束与团队共同目标的 TMRARD 模型。该模型考虑了角色间的强弱依赖关系, 力求在多角色环境下实现有效的团队内角色分配, 以最大化协同过程中的群体表现与协同效应。通过基于修正后的 E-CARGO 建模, 提出基于 MILP 的 Gurobi 求解策略。考虑引入的角色相关依赖复杂性, 逐层讨论了各参数相对理想且合理的输入规模, 并通过性能对比验证了该方案的有效性与鲁棒性。实验结果表明, 在大规模和复杂约束情形下, 使用 Gurobi 求解器具有更显著的性能优势。未来还有许多方面值得探讨。首先, 考虑引入更复杂且贴近实际场景的角色依赖关系模型, 以更真实地反映团队成员的协同共存。其次, 研究动态环境中的执行单元如何动态协同相互支持, 从而完成团队自治。此外, 对于实际应用之需求, 可以探讨更多优化算法和启发式方法, 以提高问题求解效率。

总之, 本文为团队多角色分配问题提供了新的思路, 并期望未来进一步推动该领域的发展, 为协同计算的广泛应用提供更为创新和可行的解决方案。

参考文献:

[1] TAN W A, ZHAO L, LI B, et al. Multiple cooperative task allocation in group-oriented social mobile crowdsensing [J]. IEEE Transactions on Services Computing, 2022, 15(6): 3387-3401.

[2] WANG S P, LI J, WU G J, et al. Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing [J]. IEEE Transactions on Computational Social Systems, 2022, 9(1): 109-119.

[3] ZHU H B. Some issues of role-based collaboration [C]// CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology. Montreal: IEEE, 2003: 687-690.

[4] ZHU H B, ALKINS R. Group role assignment [C]// 2009 International Symposium on Collaborative Technologies and Systems. Baltimore: IEEE, 2009: 431-439.

[5] ZHU H B. Group role assignment with constraints (GRA+): a new category of assignment problems [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2023, 53(3): 1670-1683.

[6] ZHU H B, LIU D N, ZHANG S Q, et al. Solving the group multirole assignment problem by improving the ILOG approach [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 47(12): 3418-3424.

[7] ZHU H B. Group role assignment with conflicting agent constraints [C]// 2011 International Conference on Collaboration Technologies and Systems (CTS). Philadelphia: IEEE, 2011: 516-523.

[8] ZHU H B, SHENG Y, ZHOU X Z, et al. Group role assignment with cooperation and conflict factors [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 48(6): 851-863.

[9] ZHU H. Group role assignment with multiple objectives (GRA++) [C]// E-CARGO and Role-Based Collaboration: Modeling and Solving Problems in the Complex World. New Jersey: Wiley-IEEE Press, 2022: 213-251.

[10] ZHU H B. Maximizing group performance while minimizing budget [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2020, 50(2): 633-645.

[11] ZHU H. Role-based collaboration [C]// WILEY J, Sons, Inc. E-CARGO and Role-Based Collaboration: Modeling and Solving Problems in the Complex World. New Jersey: Wiley-IEEE Press, 2022: 69-102.

[12] NG V T Y, CHAN B, SHUN L L Y, et al. Quality service assignments for role-based web services [C]// 2008 IEEE International Conference on Systems, Man and Cybernetics. Singapore: IEEE, 2008: 2218-2223.

[13] 吴诗珏. 角色协同多偏好平衡性指派研究 [D]. 广州: 广东工业大学, 2022.

[14] 徐静如, 董红斌, 赵炳旭, 等. 具有角色意识的社区服务型时空众包任务分配 [J]. 智能系统学报, 2023, 18(2): 293-304.

[15] ZHU H B. Encourage participants' contributions by roles [C]// 2005 IEEE International Conference on Systems, Man and Cybernetics. Waikoloa : IEEE, 2005: 1574-1579.

[16] 张巍, 金国跃, 滕璐瑶, 等. E-CARGO 模型在线社会网络研究团队机制 [J]. 小型微型计算机系统, 2015, 36(12): 2662-2666.

[17] 刘冬宁, 向佳敏, 曾思敏, 等. 复杂时空网络冲突消解群组角色指派研究 [J]. 工业工程, 2022, 25(4): 143-150.

[18] 滕少华, 张红, 刘冬宁, 等. E-CARGO 模型在 CSP 问题中的描述 [J]. 计算机科学, 2015, 42(2): 241-246.

[19] ZHANG L B, YU Z H, ZHU H B, et al. Group role assignment with a training plan [C]// 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC). Xiamen: IEEE, 2021: 1-6.

[20] WANG T T, LIU J M, JIN X L. Minority game strategies in dynamic multi-agent role assignment [C]// IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. Beijing: IEEE, 2004: 316-322.

[21] LIU D N, HUANG B Y, ZHU H B. Solving the tree-structured task allocation problem via group multirole assignment [J]. IEEE Transactions on Automation Science and Engineering, 2020, 17(1): 41-55.

[22] LIANG L, FU J D, ZHU H B, et al. Solving the team allocation problem in crowdsourcing via group multirole assignment [J]. IEEE Transactions on Computational Social Systems, 2023, 10(3): 843-854.2

[23] 张策, 初佃辉, 吕为工, 等. 物联网全栈人才: 一种计算机类专业系统能力培养目标 [J]. 计算机教育, 2018(2): 1-5.

[24] ZHU H B, ZHOU M C. Role-based collaboration and its kernel mechanisms [J]. IEEE Transactions on Systems, Man, and Cybernetics, 2006, 36(4): 578-589.

[25] 朱参世. 一种 Warshall 和 Floyd 算法的优化方法研究 [J]. 计算机与现代化, 2010(4): 43-45.

[26] 徐涛, 杜昱萱, 吕宗磊. 基于线性规划的传感器节点布局模型 [J]. 计算机科学, 2018, 45(7): 110-115.

[27] 韩蓉. 生产调度混合整数线性规划模型的可行解域分析 [D]. 济南: 山东大学, 2010.

[28] SOTOMAYOR-BELTRAN C, DELGADO A. Introducing to industrial engineering students to linear programming with pulp from Python [J]. International Journal of Engineering and Advanced Technology, 2019, 8(5): 749-751.

[29] SOLANKI S, PRAJAPATI R. Wheat seed classification using Gurobi optimized piecewise linear approximation-based SVM [C]// Artificial Intelligence and Sustainable Computing. ICSISCET 2023. Algorithms for Intelligent Systems. Singapore: Springer, 2024: 407-418.

[30] JIAO H W, SHNAG Y L. Two-level linear relaxation method for generalized linear fractional programming [J]. Journal of the Operations Research Society of China, 2023, 11(3): 569-594.

[31] ZHANG Y Z. A self-adjustable branch-and-bound algorithm for solving linear multiplicative programming [J]. Bulletin of the Malaysian Mathematical Sciences Society, 2024, 47(5): 137.