基于势博弈的异构多智能体系统任务分配和重分配

鞠 锴 [1] 冒泽慧 [1] 姜 斌 [1] 马亚杰 [1]

摘 要 针对异构多智能体系统, 基于势博弈理论提出一种新的任务分配和重分配算法. 考虑任务执行同步性和任务时效性的多重约束, 导致异构多智能体系统中各个体任务执行时间受到多种限制, 建立一个基于势博弈的算法结构, 使系统以分布式方式工作. 在此基础上, 基于势博弈理论设计任务分配算法, 保证在较低复杂度的同时, 可以得到近似最大化期望全局效用的良好分配方案, 并且随后将所提出的方法推广到任务重分配方案实现故障下的容错. 最后, 针对攻击任务场景对所提算法进行仿真验证, 结果表明, 在期望全局效用、容错能力和算法复杂度方面具有全面的性能.

关键词 任务分配, 多智能体系统, 势博弈, 约束优化, 容错

引用格式 鞠锴, 冒泽慧, 姜斌, 马亚杰. 基于势博弈的异构多智能体系统任务分配和重分配. 自动化学报, 2022, **48**(10): 2416−2428

DOI 10.16383/j.aas.c220003

Task Allocation and Reallocation for Heterogeneous Multiagent Systems Based on Potential Game

JU Kai[1] MAO Ze-Hui[1] JIANG Bin[1] MA Ya-Jie[1]

Abstract This paper presents a novel task allocation and reallocation algorithm for heterogeneous multiagent systems based on potential game theory. Considering the multiple constraints of task execution synchronization and task timeliness, which lead to various restrictions on the execution time of each individual task in the heterogeneous multiagent systems, a potential game-based algorithm structure is established to make agents work in a distributed manner. On this basis, the task allocation algorithm is designed based on potential game theory, which produces a promising solution that nearly maximizes the expected global utility and guarantees lower complexity, and afterwards the fault tolerance is achieved by generalizing the proposed algorithm to task reallocation schemes. Finally, the proposed algorithm is verified by simulation of the attack task scenario, and results reveal the comprehensive performance in terms of the expected global utility, the fault tolerance and the algorithm complexity.

Key words Task allocation, multiagent systems, potential game, constrained optimization, fault tolerance

Citation Ju Kai, Mao Ze-Hui, Jiang Bin, Ma Ya-Jie. Task allocation and reallocation for heterogeneous multiagent systems based on potential game. *Acta Automatica Sinica*, 2022, **48**(10): 2416−2428

近年来, 多智能体系统在一致性安全控制、健康管理、编队跟踪等多个领域得到广泛应用[1−3]. 得益于智能体之间的分工与合作, 多智能体系统在执行大规模任务时可以提供比单个智能体更好的性能. 通过采用任务分配算法, 将具有不同要求的任务分配给最合适的异构智能体, 多智能体系统可以在能耗、任务效用等方面获得最佳的整体性能. 因此, 任务分配问题在多智能体系统的研究中引起了更多关注[4−6].

日益复杂的任务需求对任务分配问题提出新的挑战. 一方面, 需要多种资源的任务通常无法由缺乏足够配置的单个智能体处理, 因此这些任务各自需要多个智能体协同执行. 在任务分配问题的标准分类法中[7], 这类任务称为 MR (Multi-robot) 任务, 且对任务分配算法提出智能体间合作的要求. 另一方面, 任务价值的动态变化和任务执行时间限制等所施加的多重约束, 使得任务分配问题的求解更加困难. MR 需求和多重约束将大大扩展可选分配方案的数量, 因此算法需要在解质量和计算时间之间

收稿日期 2022-01-03 录用日期 2022-03-13
Manuscript received January 3, 2022; accepted March 13, 2022
国家自然科学基金 (62020106003, 61922042), 中央高校基础科研基金 (FRF-BD-20-10A), 南京航空航天大学机械结构力学及控制国家重点实验室科研基金 (MCMS-I-0521G05), 高等学校学科创新引智计划 (111 计划) (B20007), 江苏省自然科学基金 (BK20211566) 资助
Supported by National Natural Science Foundation of China (62020106003, 61922042), Fundamental Research Funds for the Central Universities (FRF-BD-20-10A), Research Fund of State Key Laboratory of Mechanics and Control of Mechanical Structures in Nanjing University of Aeronautics and Astronautics (MCMS-I-0521G05), Programme of Introducing Talents of Discipline to Universities of China (111 Project) (B20007), and Natural Science Foundation of Jiangsu Province of China (BK20211566)
本文责任编委 杨涛
Recommended by Associate Editor YANG Tao
1. 南京航空航天大学自动化学院 南京 210016
1. College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016

找到平衡. 此外, 任务数量变化或智能体故障等突发情况对算法的实时性能提出考验. 面对这一系列难点, 传统的任务分配算法, 如蚁群算法[8]、粒子群算法[9] 等不再适合复杂约束下的重规划环境, 特别受限于在大规模可行解空间内寻优难度几何倍数递增的问题. 因此, 设计多重约束下更为实时、高效的任务分配算法具有重要意义.

针对 MR 任务分配问题, 现有的一类编队联盟方法注重于任务的拆分、智能体联盟的形成和将任务分配给合适的联盟[10−12]. 在这些工作中, 联盟根据任务需求的拆分而形成, 但是若任务之间的耦合足够强, 拆分将无法进行. 通过设计按序分配任务的策略, 基于市场的算法[13−15] 可以用来分配 MR 任务. 不幸的是, 决定分配顺序的过程通常取决于智能体对任务的效用, 而这一值同样来源于对任务的拆分. 近年来, 博弈方法为智能体之间的合作与协商提供了良好的性能, 因此广泛应用于优化问题[16−18]. 作为博弈方法的一种, 势博弈的特征为具有能够将智能体策略的变化同等地反映在系统整体性能上的势函数[19−21], 这一特征为各智能体自主决策是否参与合作提供了便利. 受到这一启发, 本文致力于通过智能体间的通信协商, 从整体的角度来分配强耦合的 MR 任务, 从而能够避免编队联盟方法与市场算法需要拆分任务的缺陷. 因此, 如何设计智能体博弈机制成为算法的重点和难点.

在执行之前将每项已知任务分配给合适智能体的过程称为初始分配, 之后智能体根据分配结果执行任务. 任务需求的增加和工作环境的恶化, 使得智能体在执行任务过程中更容易出现故障. 本文仅考虑永久故障, 即故障智能体无法自恢复, 从而无法再执行其剩余任务, 导致这些任务需要重新分配. 在任务分配问题中, 任务需及时执行以避免其效用严重下降, 因此实时的容错能力对于任务重分配算法至关重要. Xu 等[22] 提出一种具有容错能力的动态资源供应方法, 用以恢复云计算工作流的失败任务. Paul 等[23] 通过采用改进的容错调度算法来减轻永久处理器故障对实时应用程序的影响.

智能体故障可以通过硬件和软件方法解决. 硬件方法使用额外智能体替换故障智能体以供后续任务执行, 但在智能体数量或资源有限的情况下不可行. 软件方法使被故障终止的任务在健康智能体上恢复执行, 这一思想类似于在初始分配的基础上处理任务数量的动态变化, 如 Das 等[24] 通过并行分配和执行来处理新任务或智能体的加入. 由于方案调整的灵活性, 拍卖方法在动态环境中得到全面研究[25]. 然而, 这些工作并没有在任务分配问题模型中嵌入智能体故障. 为此, 一些学者为多处理器系统解决了容错策略分配问题, 使得失败的任务被迁移到不同的处理器上恢复执行[26−27]. 这些工作考虑了智能体故障并以尽可能提高系统可靠性作为主要目标, 但它们缺乏对任务执行质量的实质性关注.

本文提出一种将具有多重约束的任务最优地分配给异构多智能体系统, 并且具有快速有效地处理突发智能体故障能力的新算法. 势博弈思想贯穿于算法的整体设计, 主要贡献总结如下.

1) 建立考虑多重约束和智能体故障率的 MR 任务分配问题模型. 建立基于势博弈的框架, 其中智能体是完全分布式的, 且能独立地作出影响全局任务效用的决策.

2) 基于势博弈为无故障环境设计初始分配算法, 之后将其推广到针对永久智能体故障的任务重分配算法. 智能体策略总是迭代地向纳什均衡移动, 且在纳什均衡处获得近似全局最优分配.

3) 构建与本文建立的任务分配问题模型符合程度高的现实场景, 并通过仿真结果验证所提出算法的有效性.

本文的剩余部分安排如下: 在第 1 节中, 给出问题描述和优化目标; 在第 2 节中, 设计基于势博弈的算法以完成无故障情况下所有已知任务的整体最优分配, 称为初始分配; 并在第 3 节中将其推广为重分配算法来处理任务执行过程中的永久性智能体故障; 在第 4 节中, 给出仿真结果来分析所提出算法的有效性; 最后在第 5 节中给出结论.

1　问题描述

考虑最优地将 m 项任务分配给 n 个异构智能体的任务分配问题. 每个智能体在某一时刻至多参与一项任务的执行, 且每项任务需要由多个智能体同时执行. 智能体集合编号为 $A=\{A_1, A_2, \cdots, A_n\}$, 任务集合编号为 $T=\{T_1, T_2, \cdots, T_m\}$. 任务之间相互独立, 且执行任务 T_j 所需要的智能体数量定义为 N_j, 满足 $1 \le N_j \le n,\ j=1, 2, \cdots, m$. 忽略执行任务所消耗的时间, 即一旦所有 N_j 个参与执行的智能体均到达任务 T_j, 则认为该任务已完成. 智能体之间在类型、资源等方面都可以是异构的, 表现为若执行同一任务 T_j 则将提供不同的性能, 且可以通过为各智能体 $A_i,\ i=1, 2, \cdots, n$ 定义一个经归一化后值在 0 到 1 之间的变量 $\lambda_{ij},\ i=1, 2, \cdots, n,\ j=1, 2, \cdots, m$ 来量化. 任务 T_j 更可能由 λ_{ij} 更大的智能体执行, 若 λ_{ij} 等于 0, 则表示智能体 A_i 无法执行任务 T_j. 智能体在执行任务过程中可能遭受不可恢复的永久故障, 且一旦发生, 故障智能体无法再

执行后续任务, 从而产生任务的重分配需求. 每项任务 T_j 都有一个效用函数 $u_j,\ j=1,2,\cdots,m$, 与任务分配方案、各智能体对该任务的贡献、任务执行时间等多种因素有关. 下面将分别量化这些因素对任务效用的影响, 并建立任务分配问题模型.

基于每项任务需要多个智能体合作执行的需求, 模型进一步引入同步性约束, 用来描述单个智能体的努力不足以启动单项任务的情况. 在此约束下, 仅当所有参与执行任务的智能体均到达时, 任务才视为完成. 因此, 任务的开始执行时间由所有参与执行的智能体中最晚的到达时间主导, 即

$$t_j=\max_{A_i\in A}\left\{\chi_{ij}t_{ij}\right\},\ j=1,2,\cdots,m \tag{1}$$

其中, t_j 为任务 T_j 的开始执行时间; t_{ij} 为智能体 A_i 到达任务 T_j 的时间; χ_{ij} 定义为二进制决策变量, 若智能体 A_i 参与执行任务 T_j, 则 χ_{ij} 设为 1, 否则设为 0. 鉴于此, 智能体之间需要合作与协商, 以关于任务执行时间达成共识.

因而, 任务效用 u_j 来源于所有参与执行的智能体的贡献. 贡献随着各智能体能力而不同, 且计算为任务执行的奖励和资源成本的线性组合[7]. 在本工作中, 智能体 A_i 对任务 T_j 的贡献定义为 r_{ij}, 表示为

$$r_{ij}=\alpha\lambda_{ij}r_j-\beta d_{ij} \tag{2}$$

其中, r_j 为任务 T_j 的基础效用; d_{ij} 为智能体 A_i 当前位置与任务 T_j 的距离; α 和 β 分别为智能体能力和距离的权重系数, 取值取决于各自的相对重要性.

进一步引入时效性约束来强调随时间不断提升抵抗智能体行为能力, 导致智能体的可获得效用不断衰减的任务类型. 在此约束下, 每项任务限制在其出现和截止时间之间的时间窗内执行. 任务 T_j 的出现时间定义为 t_j^0, 对于起始存在的任务该值为 0. 利用以任务执行时间为变量的指数函数来表征时效性约束对任务效用的影响. 当一项任务尚未被执行但其基础效用已经随时间衰减到固定阈值时, 该任务宣告失败, 相应时间称为任务截止时间 t_j^d, 即

$$t_j^d=\max_{r_j\mathrm{e}^{-\mu_j\left(t-t_j^0\right)}\geq\delta} t \tag{3}$$

其中, 折扣系数 μ_j 决定时效性约束对任务效用的影响程度; δ 为决定任务能否被成功执行的阈值.

由于各项任务均需求多个智能体和尽可能早的执行时间, 时效性约束使得一项任务的分配对智能体和可用执行时间的占用会导致竞争失败的任务被迫推迟其执行时间, 从而导致个体乃至全局效用的衰减, 这一耦合特性使得分配每项任务时均需要考虑其他任务所受的影响. 两项约束式 (1) 和式 (3) 的引入, 将从任务的执行顺序和执行时间方面大大增加可选任务分配方案的数量, 使得对最优解的搜索更加困难.

注 1. 以空地协同作战、火灾救援、流水线工作等复杂任务为代表的任务分配问题, 均受到单项任务由多种类型的智能体协同执行的同步性约束和任务价值随时间衰减的时效性约束的限制. 因此, 考虑在模型中同时引入这两项约束具有现实意义.

根据同步性约束式 (1) 和时效性约束式 (3), 任务 T_j 的效用计算为

$$u_j=\mathrm{e}^{-\mu_j\left(t_j-t_j^0\right)}\sum_{i=1}^{n}\chi_{ij}r_{ij} \tag{4}$$

其中, t_j 在分配中要求不超过 t_j^d.

此外, 对智能体故障率建模来提升系统可靠性和期望任务执行质量. 永久故障发生的可能性会被智能体已执行任务的特性加剧. 定义智能体 A_i 执行任务 T_j 时发生永久故障的可能性为 f_{ij}, 与 T_j 和智能体 A_i 已经完成的任务有关. 利用 Guo 等[26] 给出的通用可靠性模型, f_{ij} 表示为

$$f_{ij}=1-\mathrm{e}^{-\omega_i\sigma_f t_j}\prod_{k,\,T_k\in s_i^b(T_j)}\mathrm{e}^{-\omega_i\sigma_f t_k} \tag{5}$$

其中, ω_i 为智能体 A_i 的可靠性系数; σ_f 为给定正常数; $s_i^b(T_j)$ 为智能体 A_i 在执行任务 T_j 前已经完成的任务集合. 如果任意参与智能体在执行一项任务时故障, 则将该任务终止且将其效用式 (4) 置 0. 尽管后续将通过重分配严格保证该任务的成功执行, 但建模中的这一处理将尽可能地提升系统可靠性. 因此, 任务 T_j 的期望效用 u_j^{E} 计算为

$$u_j^{\mathrm{E}}=u_j\prod_{i,\,\chi_{ij}=1}(1-f_{ij}) \tag{6}$$

根据式 (4) ~ 式 (6), 任务分配问题模型写为

$$\max_{\chi}u_g^{\mathrm{E}} \tag{7}$$

其中, $\chi=[\chi_{ij}]_{n\times m}$, 且

$$u_g^{\mathrm{E}}=\sum_{j=1}^{m}u_j^{\mathrm{E}}=\sum_{j=1}^{m}\left(\prod_{i,\,\chi_{ij}=1}(1-f_{ij})\right)\times\left(\mathrm{e}^{-\mu_j\left(t_j-t_j^0\right)}\sum_{i=1}^{n}\chi_{ij}\left(\alpha\lambda_{ij}r_j-\beta d_{ij}\right)\right) \tag{8}$$

对 $\forall A_i\in A,\ \forall T_j,T_r,T_s\in T$, 满足约束

$$\chi_{ij}\in\{0,1\} \tag{9}$$

$$\sum_{i=1}^{n}\chi_{ij}=N_j \tag{10}$$

$$0 \le t_j \le t_j^d \tag{11}$$

$$|\chi_{ir} t_r - \chi_{is} t_s| \ge \chi_{ir}\chi_{is}\tau_i(T_r, T_s) \tag{12}$$

$$\chi_{ij} t_j \ge \chi_{ij}\tau_i(A_i, T_j) \tag{13}$$

式 (10) 保证执行每项任务的智能体数量满足要求, 式 (11) 保证每项任务能在截止时间前成功执行. 式 (12)、式 (13) 用以确保任意智能体到达待执行任务的时间足够覆盖所需的路程, 其中 $\tau_i(l_1, l_2)$ 代表智能体 A_i 从 l_1 到 l_2 的位置所需的时间. 至此, 异构多智能体系统的任务分配问题转化为在同步性约束式 (1) 和时效性约束式 (3) 下, 考虑故障率式 (5), 以最大化式 (8), 即全局最优分配为目标的问题.

由式 (7) ~ 式 (13) 可见, 该任务分配问题是一个多重约束下的优化问题, 很少有找到最优解的方法, 即使有, 也需要承担大量的计算资源成本. 因此, 本文追求式 (7) 的近似全局最优, 使得算法能够在分配方案解质量和计算时间之间取得平衡.

2　无故障初始环境下的任务分配

在无故障情况下, 将所有任务 T 最优地分配给处于起始位置的智能体集合 A 的过程称为初始分配. 初始分配完成后, 每项任务将由负责它的智能体执行. 对于多智能体系统任务分配问题, 算法大多在健康情况下进行设计和实现. 在故障发生后, 若能根据健康情况下的算法进行容错的重分配, 则算法的工作量和计算量将大大减少, 效率更会提高. 因此, 本文将首先研究无故障情况下的任务分配算法, 后续将其推广到故障情况, 以表明所提出的算法具有很好的普适性.

定义 $U \subseteq T$ 为未分配任务集合. 所有 m 项任务最初都标记为未分配, 即 $U = T$, 且所有智能体均未被分配任务. 算法每轮将从 U 中选择一项任务以最大化式 (6) 为目标完成最优分配, 并将该任务添加到所分配智能体的任务包中, 直到 U 为空集, 则通过 m 轮将最终确定使式 (8) 近似最大的全局最优分配.

2.1　单项任务最优分配搜索

首先, 算法将设计智能体间针对单项任务的博弈方法, 以为每项未分配任务 $T_j \in U$ 寻找最优分配, 即从智能体集合 A 中选择如式 (10) 要求的 N_j 个对应能力 λ_{ij} 不为 0 的智能体, 使得它们合作执行 T_j 所能获得的期望任务效用式 (6) 最大.

从博弈论角度而言, 智能体为博弈参与者, 基于其位置、运动状态、能力、时间戳等信息参与任务博弈. 要求每个智能体博弈所依据的到达任务时间 t_{ij} 均满足约束式 (11) ~ 式 (13), 以确保经由同步性约束式 (1) 得到的任务执行时间 t_j 同样满足式 (11) ~ 式 (13). 此外, 要求任意两个智能体之间的通信可达且稳定, 以确保博弈能够随时进行.

定义博弈 $\Gamma = (A, S, \{u_{ij}\})$, A 仍然为第 1 节定义的智能体集合. $S = S_1 \times S_2 \times \cdots \times S_n$ 表示策略空间, 其中 $S_i \subseteq 2^T$, $i = 1, 2, \cdots, n$ 为智能体 A_i 关于是否执行 T 中各项任务的所有可选策略集合[28]. $s_i \in S_i$, $i = 1, 2, \cdots, n$ 为其中一项可选策略, 对应分配给智能体 A_i 的一组任务. 当且仅当式 (8) 中的任务分配决策变量 χ_{ij} 为 1 时, 有 $T_j \in s_i$, 即 $s_i = \{T_j | T_j \in T, \chi_{ij} = 1\}$, $i = 1, 2, \cdots, n$. $s = (s_1, s_2, \cdots, s_n)$ 代表所有智能体策略的集合, 也可写为 $s = (s_i, s_{-i})$, 其中 s_{-i} 表示除 A_i 之外其他所有智能体的策略集合. s 对应唯一的决策变量矩阵 χ, 从而代表唯一的全局任务分配, 决定期望全局效用式 (8) 的值. 据此, 式 (6) 所描述的期望任务效用也可表示为 $u_j^{\mathrm{E}}(s_i, s_{-i})$. u_{ij} 表示智能体 A_i 执行其策略 s_i 中任务 T_j 的效用函数, 其值依赖于合作智能体到达该任务的时间, 因此 u_{ij} 与所有智能体的策略相关, 可写为 $u_{ij}(s_i, s_{-i})$. $u_{ij}(s_i, s_{-i})$ 来源于任务 T_j 的期望效用 u_j^{E}, 但由于如式 (1) 和式 (4) 所示其与合作智能体效用的耦合特性, $u_{ij}(s_i, s_{-i})$ 的值不能直接计算为贡献 r_{ij} 占 u_j^{E} 的比例. 根据单个智能体对任务的边际贡献[15], $u_{ij}(s_i, s_{-i})$ 定义为

$$u_{ij}(s_i, s_{-i}) = u_j^{\mathrm{E}}(s_i, s_{-i}) - u_j^{\mathrm{E}}(s_{i,0}, s_{-i}) \tag{14}$$

其中, $s_{i,0}$ 表示智能体 A_i 不参与执行任何任务. 显然若 $T_j \notin s_i$, 有 $u_{ij}(s_i, s_{-i}) = 0$.

效用函数 $u_{ij}(s_i, s_{-i})$ 的定义为智能体的独立决策提供基础. 根据式 (14), 单个智能体对任务的边际贡献仅和该智能体以及参与该任务的其他智能体相关, 因此单个智能体在对一项任务博弈时仅需要少量邻居智能体的信息, 并能据此作出是否参与任务的决策, 这使得系统能在分布式结构下工作. 算法将寻求利用势博弈搭建 $u_{ij}(s_i, s_{-i})$ 与期望任务效用 $u_j^{\mathrm{E}}(s_i, s_{-i})$ 之间的联系, 使得 $u_j^{\mathrm{E}}(s_i, s_{-i})$ 能通过寻找各智能体 A_i 的最优策略 s_i, 以 $u_{ij}(s_i, s_{-i})$ 为媒介达到最大化. 如下给出势博弈的定义.

定义 1[28]**.** 在博弈 $\Gamma = (A, S, \{u_{ij}\})$ 中, 若存在函数 $P: S \to \mathbf{R}$, 使得对于 $\forall A_i \in A$, $\forall T_j \in T$ 和 $\forall s_i, s_i' \in S_i$, 有

$$u_{ij}(s_i, s_{-i}) - u_{ij}(s_i', s_{-i}) = P(s_i, s_{-i}) - P(s_i', s_{-i}) \tag{15}$$

则 Γ 为势博弈, P 为相应的势函数.

根据定义 1, 在势博弈中, 每个智能体由策略变

化引起的效用函数变化会等量地反映在势函数上. 如果能利用智能体效用函数构造势博弈, 使得势函数 P 为期望任务效用式 (6), 则各智能体能通过仅改变自身策略来不断提高式 (6). 在这一框架下, 个体智能体的目标能够与全局目标保持一致. 具体地, 所有智能体的策略均被随机初始化. 每个智能体 $A_i \in A$ 在其他智能体的策略 s_{-i} 保持不变的条件下, 总是朝着提升自身智能体效用 $u_{ij}(s_i, s_{-i})$ 的方向迭代调整策略 s_i. 若没有智能体能作出更好的策略选择, 则作为势函数的期望任务效用式 (6) 达到最大化. 这意味着当所有智能体的策略迭代完成时, 任何智能体都无法再通过独自调整自身策略来获取更高的效用, 称此时的策略集合 s 为纳什均衡, 如下给出其定义.

定义 2[28]**.** 在博弈 $\Gamma = (A, S, \{u_{ij}\})$ 中, 若对于 $\forall A_i \in A$, $\forall T_j \in T$ 和 $\forall s_i^*, s_i' \in S_i$, $s_i' \neq s_i^*$, 有

$$u_{ij}\left(s_i^*, s_{-i}^*\right) \geq u_{ij}\left(s_i', s_{-i}^*\right) \tag{16}$$

则 $s_i^* = (s_1^*, s_2^*, \cdots, s_n^*)$ 称为纳什均衡.

应用势博弈的关键在于, 当任意智能体 $A_i \in A$ 的策略 s_i 迭代时, 其他智能体的策略 s_{-i} 必须保持不变. 然而, 在第 1 节所搭建的 MR 任务分配问题模型中, 每项任务都需要固定数量的智能体合作执行, 这导致一个智能体的策略变化必然会要求另一个智能体的策略同步变化以维持任务所需智能体数量. 为此, 在迭代时令策略中包含该任务的智能体与所有策略中不包含该任务的智能体一一替换, 将两个策略改变的智能体视为一个整体考虑, 以便其他智能体的策略能够视为不变. 据此如下定理成立.

定理 1. 针对目标任务 $T_j \in U$, 考虑由智能体集合 A 为最大化式 (6) 所描述的期望任务效用 u_j^{E} 所进行的博弈 Γ. 如果将某次迭代中具有替换关系的两个智能体视为一个整体参与博弈, 则可以将博弈 Γ 构造为势博弈, 且势函数推导为期望任务效用 u_j^{E}. 经过有限次迭代, 所有智能体执行任务 T_j 的策略将收敛到使 u_j^{E} 最大的纳什均衡.

证明. 针对目标任务 T_j, 考虑在某次迭代中将满足 $T_j \in s_x$ 且 $T_j \notin s_y$ 的智能体 $A_x \in A$ 和 $A_y \in A$ 进行替换. 令 $s_\Delta = s_x \cup s_y$, 且令 A_x 与 A_y 的智能体效用之和为

$$u_{\Delta j}(s_\Delta, s_{-\Delta}) = u_{xj}(s_\Delta, s_{-\Delta}) + u_{yj}(s_\Delta, s_{-\Delta}) \tag{17}$$

其中, $s_\Delta \cup s_{-\Delta} = s$. 由 $T_j \notin s_y$ 可得 $u_{yj}(s_\Delta, s_{-\Delta}) = 0$. 结合式 (14)、式 (17) 可得

$$u_{\Delta j}(s_\Delta, s_{-\Delta}) = u_{xj}(s_\Delta, s_{-\Delta}) = u_j^{\mathrm{E}}(s_x, s_{-x}) - u_j^{\mathrm{E}}(s_{x,0}, s_{-x}) \tag{18}$$

从 s_x 到 s_Δ 的符号转换并未改变智能体策略 s, 因此有 $u_j^{\mathrm{E}}(s_x, s_{-x}) = u_j^{\mathrm{E}}(s_\Delta, s_{-\Delta})$. 定义 $s_{\Delta,0}$ 表示智能体 A_x 和 A_y 均不参与执行任何任务, 同样有 $u_j^{\mathrm{E}}(s_{x,0}, s_{-x}) = u_j^{\mathrm{E}}(s_{\Delta,0}, s_{-\Delta})$, 则式 (18) 可得

$$u_{\Delta j}(s_\Delta, s_{-\Delta}) = u_j^{\mathrm{E}}(s_\Delta, s_{-\Delta}) - u_j^{\mathrm{E}}(s_{\Delta,0}, s_{-\Delta}) \tag{19}$$

令 $s'_\Delta = s'_x \cup s'_y$ 满足 $T_j \notin s'_x$ 且 $T_j \in s'_y$. 由于不参与替换的智能体其策略保持不变, 有 $s'_{-\Delta} = s_{-\Delta}$. 结合式 (19), 策略由 s_Δ 转换为 s'_Δ 所导致的智能体效用变化计算为

$$\begin{aligned} &u_{\Delta j}(s_\Delta, s_{-\Delta}) - u_{\Delta j}(s'_\Delta, s_{-\Delta}) = \\ &\quad \left(u_j^{\mathrm{E}}(s_\Delta, s_{-\Delta}) - u_j^{\mathrm{E}}(s_{\Delta,0}, s_{-\Delta})\right) - \\ &\quad \left(u_j^{\mathrm{E}}(s'_\Delta, s_{-\Delta}) - u_j^{\mathrm{E}}(s_{\Delta,0}, s_{-\Delta})\right) = \\ &\quad u_j^{\mathrm{E}}(s_\Delta, s_{-\Delta}) - u_j^{\mathrm{E}}(s'_\Delta, s_{-\Delta}) \end{aligned} \tag{20}$$

根据定义 1, 博弈 Γ 为势博弈, 且 u_j^{E} 为势函数.

因此, 在所构造的博弈 Γ 中, 算法只需要朝着提升智能体效用之和式 (19) 的方向成对地迭代调整各智能体的策略, 就能实现期望任务效用式 (6) 的增加. 由于智能体效用随着策略的调整呈现单调变化, 且势博弈必然存在纳什均衡点[28], 算法会使得各智能体的策略必然收敛为使式 (6) 最大的纳什均衡策略. 根据势博弈的有限递增属性[19], 任何单方面改进的序列都会在有限时间内收敛于纳什均衡, 则本算法所设计的迭代过程一定会在有限的迭代步内实现收敛. □

针对需要 N_j 个智能体协同执行的一项未分配任务 $T_j \in U$, 其初始分配方案随机给定, 设为 $P_j^0 = \{A_{j,1}^0, A_{j,2}^0, \cdots, A_{j,N_j}^0\}$, 其中 $A_{j,k}^0 \in A$, $k = 1, 2, \cdots, N_j$. 算法 1 给出了智能体集合 A 为寻找使单项任务 T_j 的期望效用 u_j^{E} 最大的最优分配进行博弈的伪代码.

算法 1. 分配单项任务 T_j 的势博弈过程

输入. 任务 T_j, 智能体策略集合 s.

输出. 任务 T_j 的最优分配方案 $P_j^{N_j}$.

1) **for** $k = 1, 2, \cdots, N_j$
2) **for** $\forall A_i \in A$
3) **if** $A_i \notin P_j^{k-1}$ 且 $t_{ij} \leq t_j^d$
4) $s_\Delta \leftarrow s_i \cup s_{j,k}^{k-1}$, 其中 $A_{j,k}^{k-1} \in P_j^{k-1}$;
5) $s'_\Delta \leftarrow s'_i \cup s_{j,k}^{\prime k-1}$, 其中 $A_{j,k}^{k-1} \notin P_j^{k-1}$;
6) **if** $u_\Delta(s'_\Delta, s_{-\Delta}) > u_\Delta(s_\Delta, s_{-\Delta})$
7) $A_{j,k}^{k-1} \leftarrow A_i$;
8) **end if**
9) **end if**
10) **end for**

11) **for** $l = 1, 2, \cdots, N_j$

12) $A^k_{j,l} \leftarrow A^{k-1}_{j,l}$;

13) **end for**

14) $P^k_j \leftarrow \left\{A^k_{j,1}, A^k_{j,2}, \cdots, A^k_{j,N_j}\right\}$;

15) **end for**

算法 1 的输出 $P^{N_j}_j = \{A^{N_j}_{j,1}, A^{N_j}_{j,2}, \cdots, A^{N_j}_{j,N_j}\}$ 即为任务 T_j 的最优分配, 对于 $\forall A_i \in A$, 若 $A_i \in P^{N_j}_j$, 则 χ_{ij} 设为 1; 否则为 0. 将 χ_{ij}, $i = 1, 2, \cdots, n$ 代入式 (6), 可以得到任务 T_j 的最大期望效用. 显然算法 1 的迭代次数存在上界 $n \times N_j$. 相比于遍历 $C^{N_j}_n$ 种组合的枚举法, 所提出的算法将大大减少计算复杂度.

注 2. 在一致性联盟算法[29] 中, 各智能体仅根据个体收益以贪婪算法思想将任务添加到自身策略中, 从而导致分配到该任务的智能体数大于实际所需数量的冲突. 虽然后续设计该智能体间冲突的消除方法, 但也重复消耗算法资源. 而本节所设计的算法则利用智能体间博弈始终维持分配到任务的智能体数量, 从而在分配过程中避免这类冲突现象.

每项未分配任务 $T_j \in U$ 的最优分配 $P^{N_j}_j$ 能最大化其期望效用式 (6), 但 $P^{N_j}_j$ 与所有其他任务的成功执行之间可能存在冲突. 任务的允许执行时间受时效性约束式 (3) 限制, 因此应用一项任务的最优分配后, 其他任务可能缺乏足够式 (10) 要求数量的能在截止时间前到达任务的智能体, 从而无法满足式 (11). 第 2.2 节将设计消除这类冲突的方法.

2.2　冲突消除

针对经由第 2.1 节获得的每项未分配任务的最优分配 $P^{N_j}_j$, $j = 1, 2, \cdots, m$, 本节将主动检测是否应用这些分配会导致与其他任务的成功执行之间的冲突. 如果存在这样的最优分配, 则将其逐级替换为势博弈的次优解直到消除冲突, 以确保所有任务的成功执行. 本节将利用势博弈和第 2.1 节的博弈过程设计最优分配的调整方法.

通过算法 1 已经获得任务 T_j 的最优分配为 $P^{N_j}_j = \{A^{N_j}_{j,1}, A^{N_j}_{j,2}, \cdots, A^{N_j}_{j,N_j}\}$. 利用势博弈中期望任务效用随策略迭代单调变化的特性, 令 $A^{N_j}_{j,1}$ 依次与 $P^{N_j}_j$ 中不包含的智能体博弈, 且过程中固定其他智能体的策略. 除 $P^{N_j}_j$ 外对应期望任务效用 u^{E}_j 最大的博弈结果选择为势博弈的待定次优解之一, 将其定义为 $\bar{P}^1_{j,1}$. $P^{N_j}_j$ 包含的所有智能体 $A^{N_j}_{j,k}$, $k = 1, 2, \cdots, N_j$ 重复此过程, 则可以得到 N_j 个待定次优解, 定义为 $\{\bar{P}^1_{j,1}, \bar{P}^1_{j,2}, \cdots, \bar{P}^1_{j,N_j}\}$, 其中对应期望任务效用 u^{E}_j 最大的方案即为最终的势博弈次优解, 记为 $\bar{P}^1_j$, 用以替换 T_j 的最优分配 $P^{N_j}_j$.

$\bar{P}^1_j$ 应当被检测是否其仍然会导致与其他未分配任务的冲突. 如果会, 则计算势博弈下一级的次优解 $\bar{P}^2_j$. $\bar{P}^2_j$ 来源于 $\bar{P}^1_j$ 分支上的 N_j 个待定次优解 $\{\bar{P}^2_{j,1}, \bar{P}^2_{j,2}, \cdots, \bar{P}^2_{j,N_j}\}$ 和 $P^{N_j}_j$ 分支上除 $\bar{P}^1_j$ 之外的 $N_j - 1$ 个待定次优解 $\{\bar{P}^1_{j,1}, \bar{P}^1_{j,2}, \cdots, \bar{P}^1_{j,N_j}\} \backslash \bar{P}^1_j$. 如果 $\bar{P}^2_j$ 仍然不能应用, 则将出现越来越多的分支, 使得 $\bar{P}^3_j$ 或后续 $\bar{P}^k_j$, $k = 4, 5, \cdots$ (如果有) 的搜索更加混乱. 因此, 算法构建线性表结构以搜索和计算势博弈次优解, 显著提升选择分配方案的灵活性.

线性表用来存储所有待定次优解, 选择次优解的过程一般化为线性表中结点的添加和删除. 添加的方案要求不与表中现存的方案或曾删除的方案重复. 最初, 线性表为空. 在某一轮得到一项任务的最优分配后, 将其添加入线性表并进行检测. 如果应用该最优分配后会存在冲突, 则删除该方案并将其分支下的势博弈待定次优解添加入线性表. 从表中选取对应期望任务效用最大的方案作为次优解, 并检测冲突是否消除. 若已消除, 则用其替换最优分配, 否则重复此过程, 直到搜索到可行方案. 如果直到线性表再次为空也没有找到合适的方案, 则意味着对该被检测任务的任何分配都会导致与其他未分配任务的冲突. 在这种情况下, 被检测任务在这一轮中不允许被分配, 而是在下一轮重新考虑. 图 1 描述了利用线性表选择势博弈次优解的过程, 其中不会引发冲突的最终可行分配被标记为 $\bar{P}^{k_j}_j$.

至此, 算法得到每项未分配任务的可行最优分配方案 $P^{N_j}_j$. 然而, 由于时效性约束式 (3) 带来的耦合特性, 各项任务使式 (6) 最大的最优分配不能保证实现所有任务全局最优的目标式 (7). 第 2.3 节将研究单项任务的最优分配与全局最优的关系.

2.3　全局最优分配搜索

基于第 2.1 节和第 2.2 节的结果, 本节将定义各项任务的优先级来描述一项任务的最优分配趋向于全局最优的程度, 并分配具有最高优先级的未分配任务. 一项任务的分配对多个智能体的占用, 会导致其他任务在时效性约束下所能获得最大期望效用的衰减. 值得一提的是, 智能体的故障率与其曾经执行的任务有关, 随分配过程动态变化, 其影响同样包含在这一衰减中. 因此, 该任务的优先级取决于其他任务受到该任务分配的影响程度, 那么为尽可能最大化式 (8), 应当首先分配所导致衰减最小的任务. 实际上, 这表示该任务的最优分配相比于其他任务更趋近于全局最优. 针对某一轮中的任意未分配任务 $T_j \in U$, 令其通过第 2.1 节和第 2.2

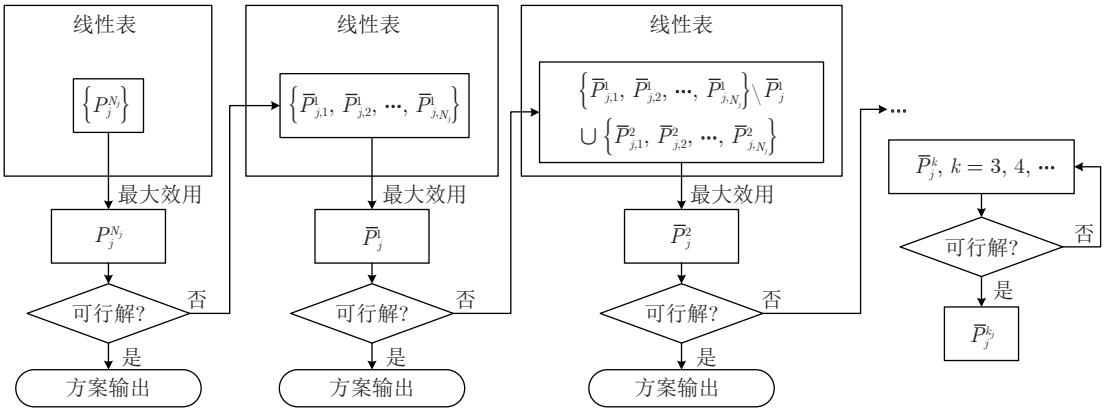

图 1 势博弈次优解的选择过程

Fig. 1 The process of selecting the suboptimal solution of the potential game

节得到的最大期望效用为 u_{jm}^{E}. 假设任务 T_j 按其个体的最优方案完成分配, 且方案中参与智能体的位置、速度、加速度和时间戳得到更新, 在此基础上计算所有其他未分配任务 $T_k \in U,\ k \neq j$ 在下一轮所能获得的最大期望效用 $\bar{u}_{km}^{\mathrm{E}}$. 从 u_{km}^{E} 到 $\bar{u}_{km}^{\mathrm{E}}$ 的变化量化了任务 T_k 受到任务 T_j 分配影响的程度, 据此, 任务 T_j 的优先级权值定义为其对所有其他未分配任务所造成的最严重影响, 即

$$w_j = \max_{T_k \in U,\, k \neq j} \left\{ u_{km}^{\mathrm{E}} - \bar{u}_{km}^{\mathrm{E}} \right\} \tag{21}$$

权值最小的任务 $T_p = \arg\min_{T_j \in U} w_j$ 被赋予最大的优先级, 且确定为这一轮的分配对象. 定义第 l 轮所有智能体状态与时间戳的信息集合为 $x(l)$. 算法 2 给出了计算每项未分配任务优先级权值的伪代码.

算法 2. 优先级权值计算

输入. 未分配任务集合 U, 第 l 轮智能体信息集合 $x(l)$.

输出. 任务优先级权值 $w_j,\ j = 1, 2, \cdots, m$.

1) **for** $\forall T_j \in U$

2) 基于 $x(l)$ 计算 u_{jm}^{E};

3) **end for**

4) **for** $\forall T_j \in U$

5) 将 $x(l)$ 更新为 $x(l+1)$;

6) **for** $\forall T_k \in U,\ k \neq j$

7) 基于 $x(l+1)$ 计算 $\bar{u}_{km}^{\mathrm{E}}$;

8) **end for**

9) $w_j \leftarrow \max_{T_k \in U,\, k \neq j} \left\{ u_{km}^{\mathrm{E}} - \bar{u}_{km}^{\mathrm{E}} \right\}$;

10) **end for**

健康环境下整体初始分配的流程图如图 2 所示. 算法每轮通过第 2.1 节 ~ 第 2.3 节这 3 个步骤完成一项优先级最大的任务 T_p 的分配. 所有参与该任务执行的智能体的状态与时间戳信息应当被更新, 以进行下一轮博弈. 任务 T_p 标记为已分配并且

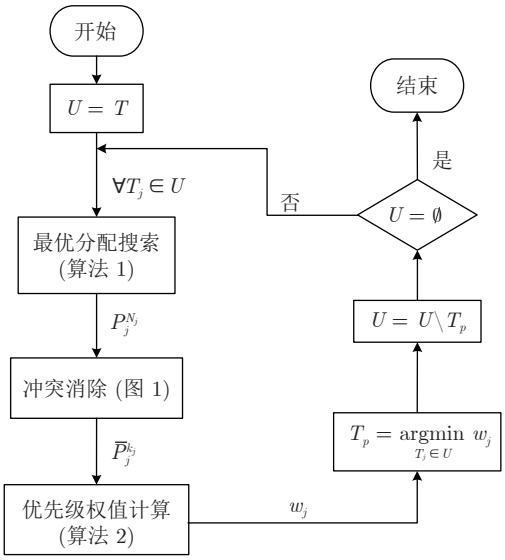

图 2 健康环境下的初始任务分配流程图

Fig. 2 The flow chart of the initial task allocation in the healthy environment

不会参与后续计算. 将未分配任务集合 U 迭代为 $U \backslash T_p$, 智能体集合 A 针对 U 中所有剩余任务重复这 3 个步骤. 经过 m 轮最终得到所有任务的全局近似最优分配, 此时 U 为空.

注 3. 从整个初始分配算法的设计过程中可以总结出所提出算法的优越性. 一方面, 智能体作为整体对单项 MR 任务进行博弈, 从而避免编队联盟方法[10−12] 和市场算法[13−15] 所面对的任务难以拆分的问题. 另一方面, 算法根据博弈过程对分配方案进行优劣排序, 从而确定任务优先级, 解决任务间的耦合. 与传统群体智能算法[8−9] 寻优方向的随机性相比, 所提出算法等效为在一定程度上对寻优方向具有导向性, 因此具有较低的寻优难度.

至此, 算法完成健康环境下所有任务的初始分

配. 算法的设计追求包含全局效用最大化和所有任务成功执行的目标, 同时能够在解质量和计算时间之间取得较好的平衡. 通过循环由这 3 个步骤构成的算法, 全局最优式 (7) 由分布式结构中智能体之间的合作与协商来近似实现.

3 故障情况下的任务重分配

在无故障情况下已经完成所有任务的初始分配, 随后每个智能体根据其策略执行任务. 智能体在任务执行过程中可能会发生永久故障, 以致故障智能体无法再执行剩余任务. 因此, 这些任务需要重分配, 即智能体需要实时调整策略. 特别地, 当协同执行具有多重约束式 (1) 和式 (3) 的 MR 任务时, 单个智能体策略的改变可能会导致整个系统策略的改变. 如果每发生故障就对所有任务重新分配, 则尽管结果会更优, 但所有智能体策略的动态重置将导致分配方案解空间的非线性剧增, 消耗大量的算法资源, 降低重分配的实时性能和效率. 因此, 本节将健康环境下的任务分配算法推广到故障情况下, 同时将健康环境下的分配结果作为重分配的依据, 以减少工作量和计算量. 算法推广的基础是在健康情况下, 一项任务只能在紧跟着前一项任务的位置之后被添加到智能体策略中, 而在故障情况下, 重分配任务将被提供若干个可添加位置供其选择.

如下给出一些服务于算法的准备和说明. 所有任务 $T_j \in T$ 按照初始分配方案下的任务执行时间 t_j 从小到大进行排序, 依次重新编号为 $T_k^s \in T$, 其中任务 T_k^s 的执行时间写作 t_k^s, $k = 1, 2, \cdots, m$.

待重分配的任务需要从初始分配下的智能体策略中移除. 如式 (4) 所示, 任务效用与执行时间有关. 移除由智能体故障终止的任务后, 剩余未执行任务的执行时间需要进行调整以最大化期望全局效用式 (8). 具体地, 调整方法是以排序后执行时间的顺序依次决定剩余任务能否被提前执行.

为方便理解, 下面举例说明移除任务的方法. 给定共有 4 个智能体 A_1, A_2, A_3, A_4 和 6 项任务 $T_1^s, T_2^s, \cdots, T_6^s$, 每个智能体的策略 s_i, $i = 1, 2, 3, 4$ 根据第 2 节所得到的全局最优分配假定如下, 同时标注所有任务的执行时间.

$$
\begin{aligned}
&s_1 : T_1^s\left(t_1^s\right),\ T_3^s\left(t_3^s\right),\ T_4^s\left(t_4^s\right),\ T_6^s\left(t_6^s\right)\\
&s_2 : T_1^s\left(t_1^s\right),\ T_2^s\left(t_2^s\right),\ T_5^s\left(t_5^s\right)\\
&s_3 : T_1^s\left(t_1^s\right),\ T_2^s\left(t_2^s\right),\ T_5^s\left(t_5^s\right),\ T_6^s\left(t_6^s\right)\\
&s_4 : T_2^s\left(t_2^s\right),\ T_3^s\left(t_3^s\right),\ T_4^s\left(t_4^s\right)
\end{aligned}
$$

给定 A_4 在时间 t_f 时发生永久故障, 其中 $t_3^s \le t_f \le t_4^s$, 显然 T_4^s 需要重新分配. 算法在移除 T_4^s 后需要依次决定 T_5^s 和 T_6^s 是否能被提前执行. 结果是, 由于 T_4^s 和 T_5^s 不会相互影响, T_5^s 不能被提前执行. 尽管 T_4^s 和 T_6^s 存在先后执行关系, T_6^s 能否被提前需要进一步讨论. 考虑 A_1 比 A_3 先到达 T_6^s 并等待 A_3, 则在移除 T_4^s 后, A_3 的到达时间并不会改变, 从而 T_6^s 的执行时间也不会改变. 相反, 若 A_3 比 A_1 先到达, 则 T_6^s 的执行时间能够提前.

待重分配的任务被全部移除后, 应当各自重插入到智能体策略中. 与初始分配时智能体始终将任务放置在其策略末尾来参与博弈不同, 待重分配任务的可重插入位置并不唯一. 考虑重插入一项任务, 其按执行时间排序后的编号为 T_r^s. 若将 T_r^s 插入至一个智能体的策略中, 算法需要决定其插入位置. 通过拆分任务的可执行时间窗来讨论可选重插入位置. 若距离故障时刻 t_f 最近的已执行任务其排序后的编号为 T_c^s, 则有 $p = m - m_r - c + 1$ 种位置使得任务执行的顺序不同, 其中 m_r 为待重插入任务数.

仍然讨论上例来阐述可重插入任务的不同位置. 被移除的任务 T_4^s 待重新插入且需要除 A_4 之外的两个智能体执行. 任务 T_5^s 和 T_6^s 调整后的执行时间分别定义为 $\bar{t}_5^s$ 和 $\bar{t}_6^s$. 由于 $t_1^s \le t_2^s \le t_3^s \le t_f \le \bar{t}_5^s \le \bar{t}_6^s$, 显然 T_4^s 有 3 个可选重插入位置, 即 t_f 和 $\bar{t}_5^s$ 之间、$\bar{t}_5^s$ 和 $\bar{t}_6^s$ 之间、$\bar{t}_6^s$ 之后. 对于每个位置, 其在每个智能体策略上的反映将是唯一的. 以 A_3 为例, 若将 T_4^s 分配在 $\bar{t}_5^s$ 和 $\bar{t}_6^s$ 之间执行, 则反映在 A_3 的策略上只能是处于 T_5^s 和 T_6^s 之间. 插入位置的后续任务会受到影响, 但它们不一定需要推迟其任务执行时间. 在极端情况下, 一些智能体可能会花费大量的时间等待合作智能体, 待重分配任务对这段时间窗的占用不会消耗任务执行额外的时间资源.

如果重插入 T_r^s 不会推迟任何任务的执行, 则期望全局效用式 (8) 不会减少, 否则会导致由时效性约束式 (3) 引起的效用衰减. 任务总是包含在多个智能体的策略中, 因此在被推迟的任务数量方面影响可能较为严重. 那么, 重插入 T_r^s 所引起的期望全局效用变化量 Δu_g^{E} 不仅与 T_r^s 自身的效用有关, 还与后续任务被推迟执行的程度有关, 计算为

$$
\Delta u_g^{\mathrm{E}} = u_{r,\,s}^{\mathrm{E}} + \sum_{T_j \in T^d} \left(\mathrm{e}^{-\mu_j\left(t_j' - t_j\right)} - 1\right) u_j^{\mathrm{E}} \tag{22}
$$

其中, $u_{r,\,s}^{\mathrm{E}}$ 为任务 T_r^s 的期望效用; $T^d \subseteq T$ 为被推迟执行的任务集合; t_j' 为任务 T_j 推迟后的执行时间. 如果 T_r^s 的基础收益足够低, 这一变化量甚至可能为负值, 但为严格保证所有任务成功完成, 这样的妥协是必要的.

变化量式 (22) 依赖任务 T_r^s 所重新分配的智能体而不同. 对于每个可重插入位置, 各智能体根据反映在其策略上的唯一位置参与博弈. 利用第 2 节

所提出的算法 1, 从健康智能体中选择出满足数量要求的智能体, 使得期望全局效用的变化量式 (22) 最大化. 在所有可重插入位置上进行单项任务最优分配搜索, 在通过冲突消除确保所有待重分配任务能够成功执行的前提下, 将其中 Δu_g^{E} 最大的结果决定为 T_r^s 的最优重分配. 相同地获得所有待重分配任务的最优重分配后, 所有这些任务被添加到由算法 2 设计的基于优先级的全局分配框架中, 且同样应用式 (21) 每轮完成一项具有最高优先级任务的重分配. 通过对所有任务重复重插入算法, 算法最终能获得重分配的全局最优. 任务重分配的流程图如图 3 所示. 与图 2 所描述的初始分配相比, 重分配额外考虑在搜索单项任务的最优分配过程中智能体参与博弈所基于的重插入位置.

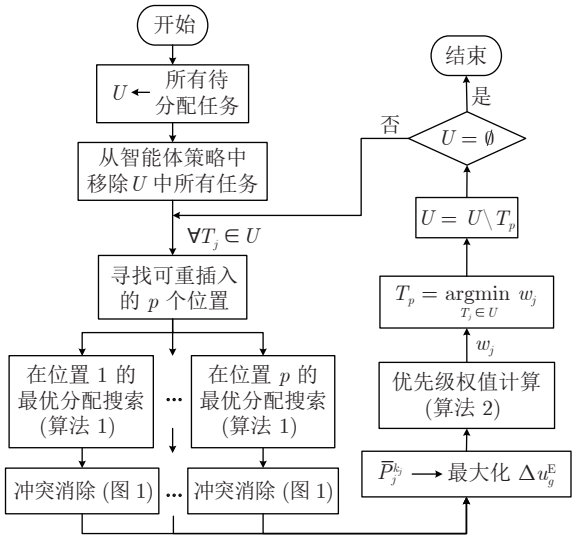

图 3 故障情况下的任务重分配流程图

Fig. 3 The flow chart of the task reallocation in the faulty case

仍然讨论上例. 给定智能体 A_4 在执行 T_2^s 后永久故障, 则 T_3^s 和 T_4^s 需要重新分配. 重分配方案为判断 T_5^s 和 T_6^s 是否可以提前执行, 之后依次重插入 T_3^s 和 T_4^s. 移除和重插入任务的具体过程已在上文分析, 这里不再赘述. 可以发现在此例中, 算法仅需要调整 T_5^s、T_6^s 的执行时间和 T_3^s、T_4^s 的分配. 这表明通过推广健康环境下的分配算法并结合故障信息, 重分配问题被简化为对少量任务的调整, 而不需要重启所有任务的分配. 从这种意义上说, 算法能够避免不必要的时间消耗. 至此, 本节已经完成具有容错能力的任务重分配算法设计, 同时算法解质量和实时性能得到保证.

4 仿真结果

构建一个多个智能体攻击多个任务目标的军事场景, 以验证所提出算法的可行性和有效性. 每个任务目标具有针对智能体行为的初始防御能力, 防御能力越弱对应该任务的基础效用越高. 防御工事随时间加强, 因此智能体攻击目标越晚, 则可获得的效用越低. 如果超过截止时间, 则来自智能体的攻击无法穿透防御, 该任务宣告失败. 每个目标都由所需数量的智能体同时攻击以确保火力的集中程度, 且完成攻击后这些智能体立刻各自前往下一个目标. 所构建的场景显然非常契合本文所建立的任务分配问题模型. 任务目标为最大化期望全局效用, 且确保所有任务成功完成. 算法仅在 XOY 平面上进行仿真, 且表 1、表 2 给出了某次仿真中智能体和任务的参数. 这样的设定和参数选择仅是为了使实验结果中智能体的轨迹清晰, 实际上智能体和任务可在场景中随机分布, 且数量和所有参数均可调.

图 4 给出了无故障情况下的初始最优分配和相应的智能体轨迹. 以 Minimum snap 为目标[30], 基于贝塞尔曲线优化智能体轨迹[31], 使其保持光滑和连续. 在任务分配和重分配阶段, 轨迹规划均在完成一项任务的分配或重分配后进行. 以分配完成上一项任务时智能体的位置、速度、加速度为起点约束, 同时以新分配任务的执行时间为终点约束, 结合最大速度和加速度约束, 轨迹优化问题实质上被转化为能用 MATLAB 直接求解的凸二次规划问题, 因此不再赘述. 每项任务的执行时间在任务点旁边标注. 由图 4 易得各智能体的最优策略分别为 $s_1 = \{T_3, T_4\}$, $s_2 = \{T_2, T_1\}$, $s_3 = \{T_2, T_3\}$, $s_4 = \{T_1, T_4\}$. 以 A_2 的轨迹为例说明任务执行的同步性. A_2 首先在 282.0 s 时与 A_3 同时到达 T_2 并合作攻击 T_2, 相似地随后在 408.0 s 时与 A_4 共同攻击 T_1. 可以看出, 每组合作智能体都满足同步性约束, 且智能体的轨迹连续光滑.

任务分配算法第 2.1 节的目标是获得所有未分配任务的最优分配, 首先考虑第 2.1 节分配单项任务的性能. 对于某个需要两个智能体执行的单项任务, 图 5、图 6 分别给出了在不同智能体数量 n 下采用势博弈算法和枚举法的可得最大期望任务效用和所需迭代次数. 针对每种情况进行 50 次重复实验并取平均值, 其中智能体和任务的位置随机分布. 可以看出, 基于势博弈的算法能提供与枚举法相同的性能, 也就是说, 所提出算法的第 1 步可以获得任意单项任务的最优分配. 相反, 算法复杂度的差异很大. 枚举法和势博弈算法的复杂度分别为 $\mathrm{O}(n^2)$ 和 $\mathrm{O}(n)$, 因此势博弈算法的复杂度显著减少. 此外, 随着智能体数量的增加, 可能会出现能力更强或距离更近的智能体, 从而导致可获得的任务效用增加.

表 1　智能体初始信息

Table 1　Initial information of agents

智能体 A_i	位置 (m, m)	能力 (λ_{i1}, λ_{i2}, λ_{i3}, λ_{i4})	最大速度 (m/s)	最大加速度 (m/s^2)
A_1	(0, 0)	(0.5, 0.5, 1.0, 1.0)	10	0.25
A_2	(1000, 0)	(1.0, 1.0, 0.5, 0.5)	20	0.25
A_3	(3000, 0)	(0.5, 1.0, 1.0, 0.5)	10	0.25
A_4	(6000, 6000)	(1.0, 0.5, 0.5, 1.0)	20	0.25

表 2　任务初始信息

Table 2　Initial information of tasks

任务 T_j	位置 (m, m)	基础效用 r_j	所需智能体数量 N_j	折扣系数 μ_j
T_1	(4000, 4000)	10	2	5×10^{-4}
T_2	(2000, 2000)	20	2	5×10^{-4}
T_3	(0, 2000)	10	2	5×10^{-4}
T_4	(2000, 4000)	10	2	5×10^{-4}

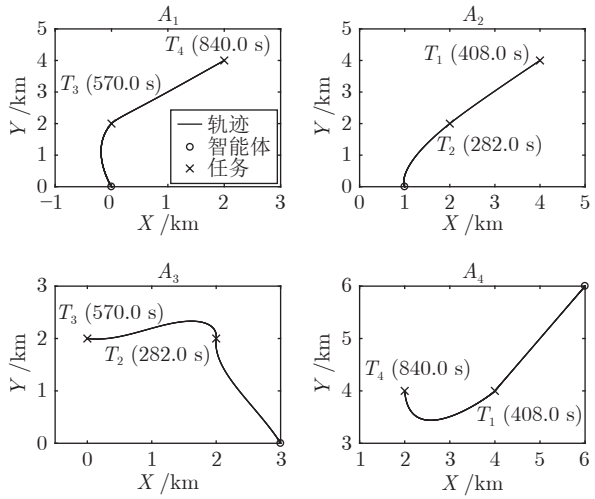

图 4　无故障情况下的最优任务分配方案

Fig. 4　The optimal task allocation scheme in the non-faulty case

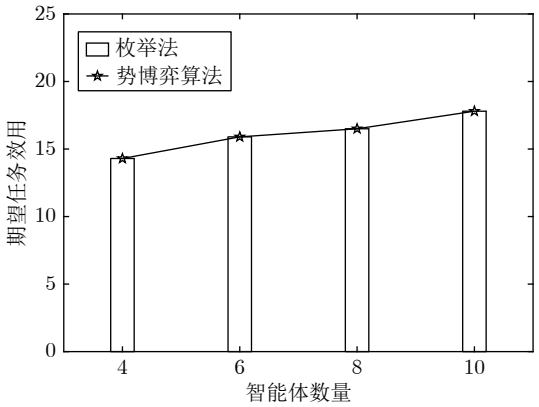

图 5　分配单项任务的最大期望效用对比

Fig. 5　Comparison of the maximum expected utilities for allocating a single task

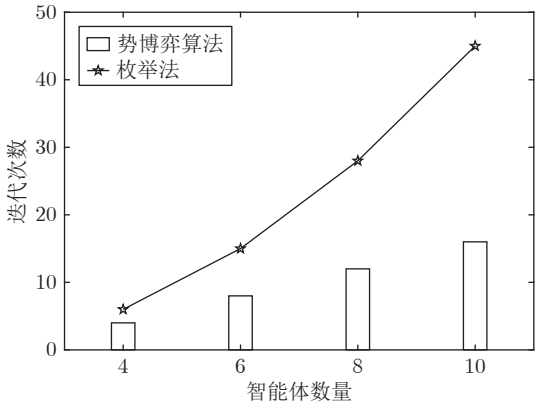

图 6　分配单项任务的所需迭代次数对比

Fig. 6　Comparison of the required number of iterations for allocating a single task

考虑第 2 节中完整的无故障情况下的任务分配算法. 仍然用枚举法作为比较, 图 7、图 8 分别给出了在不同任务数量 m 和不同智能体数量 n 下通过使用势博弈算法所能获得的最大期望全局效用相对于枚举法的衰减量. 两个实验分别在 $m=4$ 和 $n=4$ 的条件下进行, 每个实验重复 50 次取平均值, 其中智能体和任务的位置随机分布. 由图可见, 所提出算法仍然可以提供与枚举法相似的性能. 随着任务数量增加, 所提出算法的性能相对降低, 这是由于每轮在计算每项任务的优先级时, 只考虑分配该任务将造成的下一轮效用衰减量, 而未考虑后续轮次的效用同样会受到影响. 特别地, 如果任务分配的总轮数为 1 或 2, 则算法等同于已经考虑所有轮次的影响. 因此, 由图 7 可见, 当任务数量为 1 或 2 时, 所提出算法能够获得与枚举法相同的最佳效用. 可以推断, 若在计算优先级时考虑某一轮分配

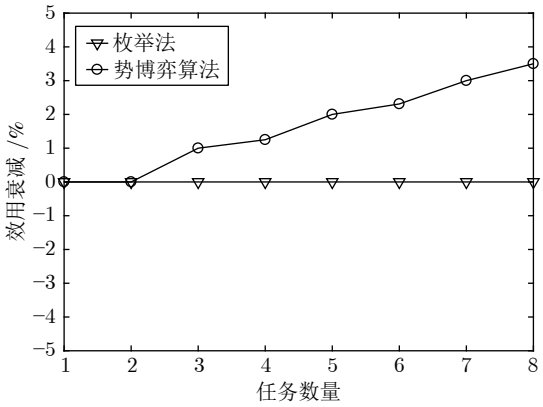

图 7 不同任务数量下的最大期望全局效用衰减

Fig. 7 Reductions of the maximum expected global utility under different number of tasks

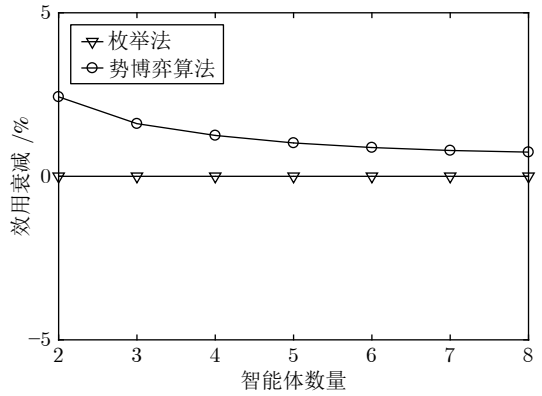

图 8 不同智能体数量下的最大期望全局效用衰减

Fig. 8 Reductions of the maximum expected global utility under different number of agents

对更多后续轮次的影响, 所提出算法会提供更好的性能, 而代价是计算复杂度的提升. 相反, 随着智能体数量的增加, 所提出的算法将提供更好的性能. 这是因为可用于任务执行的智能体越多, 在相邻两轮中选择的两项任务就越有可能由完全不同的智能体执行, 则基于优先级的框架带来的效用衰减越低.

图 9 给出了第 2 节提出的任务分配算法与枚举法的迭代次数的比值随智能体数量和任务数量的变化图. 可以总结出当任务数量满足 $m \geq 4$ 时, 势博弈算法的迭代次数已不足枚举法迭代次数的 1%. 考虑在无故障情况下将均需要两个智能体协同执行的 m 项任务分配给 n 个智能体的初始分配问题, 枚举法的算法复杂度为 $\mathrm{O}\left(m! \times n^{2m}\right)$, 而所提出势博弈算法的复杂度为 $\mathrm{O}\left(m^3 \times n\right)$. 综上所述, 所提出的算法在能有效获得令人满意的任务效用的同时具有较低的复杂度.

为验证重分配算法的容错能力, 在故障情况下进行仿真. 如图 4 所示无故障情况下任务分配算法

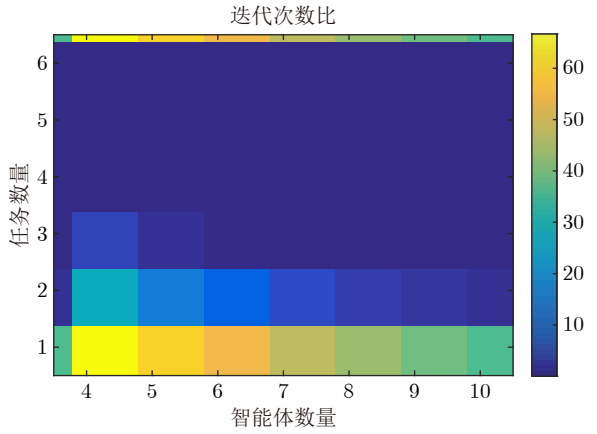

图 9 所提出算法与枚举法迭代次数的比值

Fig. 9 The ratio of the number of iterations between the proposed algorithm and the enumeration method

的仿真结果被用作初始条件. 给定 A_2 在 300 s 时故障并且无法从故障中恢复运行, 此时 A_2 已经执行完 T_2, 则 T_1 需要重新分配. 重分配方案和相应的智能体轨迹如图 10 所示, 其中用虚线表示无故障情况下的仿真结果作为参考.

由图 10 可见, 故障后智能体策略分别调整为 $s_1' = \{T_3, T_4, T_1\}$, $s_2' = \{T_2\}$, $s_3' = \{T_2, T_3\}$, $s_4' = \{T_4, T_1\}$. T_1 被重分配至 A_1 和 A_4, T_3 和 T_4 的执行时间得到调整以追求全局最优. 此外, A_2 在故障时间停止轨迹, 同时所有健康智能体立刻对故障作出反应, 这表明重分配算法在容错方面具有良好的实时性能.

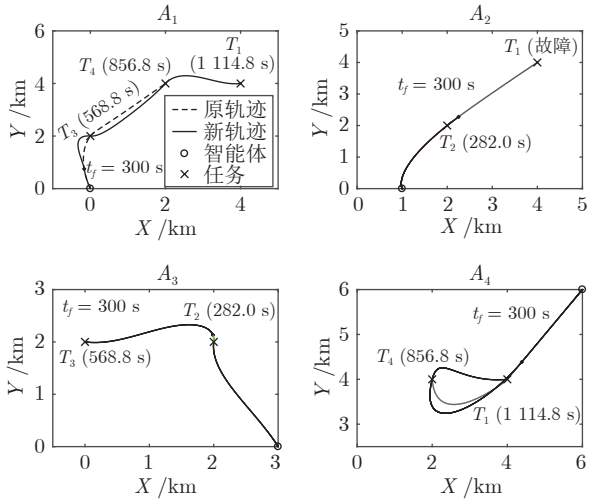

图 10 故障情况下的最优任务重分配方案

Fig. 10 The optimal task reallocation scheme in the faulty case

5 结论

基于势博弈解决异构多智能体系统的任务分配

问题. 多重约束和智能体故障率同时被建模, 这给任务执行时间带来严格的限制. 所构建的势博弈框架允许合作和协商, 从而为无故障和故障情况下的分布式任务分配和重分配算法设计奠定基础. 然后, 在保证所有任务成功执行的前提下, 通过较少次迭代达到近似全局最优, 并且可以在健康的智能体上恢复被智能体故障终止的任务, 这表明所提出算法具有容错性能. 针对能够模拟所研究模型特征的攻击任务场景进行仿真, 与枚举法的比较结果表明所提出算法在期望全局效用和算法复杂度方面的有效性. 算法的一个局限是需要持续通信, 这在大规模系统或恶劣环境中可能不可行, 从而导致博弈无法以最佳方式进行. 智能体在通信范围受限和通信故障时进行博弈的方法, 将在以后的工作中讨论.

References

1 Yang Y, Li Y F, Yue D, Tian Y C, Ding X H. Distributed secure consensus control with event-triggering for multiagent systems under DoS attacks. *IEEE Transactions on Cybernetics*, 2021, **51**(6): 2916−2928

2 Zhang K, Jiang B, Shi P. Adjustable parameter-based distributed fault estimation observer design for multiagent systems with directed graphs. *IEEE Transactions on Cybernetics*, 2017, **47**(2): 306−314

3 Tian Lei, Dong Xi-Wang, Zhao Qi-Lun, Li Qing-Dong, Lv Jin-Hu, Ren Zhang. Distributed adaptive time-varying output formation tracking for heterogeneous swarm systems. *Acta Automatica Sinica*, 2021, **47**(10): 2386−2401
(田磊, 董希旺, 赵启伦, 李清东, 吕金虎, 任章. 异构集群系统分布式自适应输出时变编队跟踪控制. 自动化学报, 2021, **47**(10): 2386−2401)

4 Huang Gang, Li Jun-Hua. Multi-UAV cooperative target allocation based on AC-DSDE evolutionary algorithm. *Acta Automatica Sinica*, 2021, **47**(1): 173−184
(黄刚, 李军华. 基于 AC-DSDE 进化算法多 UAVs 协同目标分配. 自动化学报, 2021, **47**(1): 173−184)

5 Li M C, Wang Z D, Li K L, Liao X K, Hone K, Liu X H. Task allocation on layered multiagent systems: When evolutionary many-objective optimization meets deep Q-learning. *IEEE Transactions on Evolutionary Computation*, 2021, **25**(5): 842−855

6 Li Yong, Li Kun-Cheng, Sun Bai-Qing, Zhang Qiu-Hao, Wang Yi-Na, Yang Jun-You. Multi-robot-multi-task coordination framework based on the integration of intelligent agent and petri net. *Acta Automatica Sinica*, 2021, **47**(8): 2029−2049
(李勇, 李坤成, 孙柏青, 张秋豪, 王义娜, 杨俊友. 智能体 Petri 网融合的多机器人 —— 多任务协调框架. 自动化学报, 2021, **47**(8): 2029−2049)

7 Gerkey B P, Mataric M J. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 2004, **23**(9): 939−954

8 Alencar R C, Santana C J, Bastos-Filho C J A. Optimizing routes for medicine distribution using team ant colony system. *Advances in Intelligent Systems and Computing*, 2020, **923**: 40−49

9 Zhang F. Intelligent task allocation method based on improved QPSO in multi-agent system. *Journal of Ambient Intelligence and Humanized Computing*, 2020, **11**: 655−662

10 Rahman S M M, Wang Y. Mutual trust-based subtask allocation for human-robot collaboration in flexible lightweight assembly in manufacturing. *Mechatronics*, 2018, **54**: 94−109

11 Chen J X, Wu Q H, Xu Y H, Qi N, Guan X, Zhang Y L, et al. Joint task assignment and spectrum allocation in heterogeneous UAV communication networks: A coalition formation game-theoretic approach. *IEEE Transactions on Wireless Communications*, 2021, **20**(1): 440−452

12 Wang D, Li K M, Zhang Q, Lu X F, Luo Y. A cooperative task allocation game for multi-target imaging in radar networks. *IEEE Sensors Journal*, 2021, **21**(6): 7541−7550

13 Sullivan N, Grainger S, Cazzolato B. Sequential single-item auction improvements for heterogeneous multi-robot routing. *Robotics and Autonomous Systems*, 2019, **115**: 130−142

14 Lee D H. Resource-based task allocation for multi-robot systems. *Robotics and Autonomous Systems*, 2018, **103**: 151−161

15 Huang X W, Gong S M, Yang J M, Zhang W J, Yang L W, Yeo C K. Hybrid market-based resources allocation in mobile edge computing systems under stochastic information. *Future Generation Computer Systems*, 2022, **127**: 80−91

16 Xu Y H, Jiang B, Yang H. Two-level game-based distributed optimal fault-tolerant control for nonlinear interconnected systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2020, **31**(11): 4892−4906

17 Liu Kun, Zheng Xiao-Shuai, Lin Ye-Ming, Han Le, Xia Yuan-Qing. Design of optimal strategies for the pursuit-evasion problem based on differential game. *Acta Automatica Sinica*, 2021, **47**(8): 1840−1854
(刘坤, 郑晓帅, 林业茗, 韩乐, 夏元清. 基于微分博弈的追逃问题最优策略设计. 自动化学报, 2021, **47**(8): 1840−1854)

18 Duan H B, Li P, Yu Y X. A predator-prey particle swarm optimization approach to multiple UCAV air combat modeled by dynamic game theory. *IEEE/CAA Journal of Automatica Sinica*, 2015, **2**(1): 11−18

19 Chapman A C, Micillo R A, Kota R, Jennings N R. Decentralised dynamic task allocation: A practical game theoretic approach. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems. Budapest, Hungary: IEEE, 2009. 915−922

20 Wu H, Shang H L. Potential game for dynamic task allocation in multi-agent system. *ISA Transactions*, 2020, **102**: 208−220

21 Monderer D, Shapley L S. Potential games. *Games and Economic Behavior*, 1996, **14**(1): 124−143

22 Xu X L, Mo R C, Dai F, Lin W M, Wan S H, Dou W C. Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. *IEEE Transactions on Industrial Informatics*, 2020, **16**(9): 6172−6181

23 Paul S, Chatterjee N, Ghosal P. A permanent fault tolerant dynamic task allocation approach for network-on-chip based multicore systems. *Journal of Systems Architecture*, 2019, **97**: 287−303

24 Das G P, McGinnity T M, Coleman S A, Behera L. A fast distributed auction and consensus process using parallel task allocation and execution. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. San Francisco, CA, USA: IEEE, 2011. 4716−4721

25 Farinelli A, Iocchi L, Nardi D. Distributed on-line dynamic task assignment for multi-robot patrolling. *Autonomous Robots*, 2017, **41**(6): 1321−1345

26 Guo S C, Huang H Z, Wang Z L, Xie M. Grid service reliability modeling and optimal task scheduling considering fault recovery. *IEEE Transactions on Reliability*, 2011, **60**(1): 263−274

27 Shatz S M, Wang J P. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Transactions on Reliability*, 1989, **38**(1): 16−27

28 Shoham Y, Brown K L. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations*. Cambridge: Cambridge University Press, 2008. 174−177

29 Tang Jia-Yu, Li Xiang-Min, Dai Jin-Jin, Bo Ning. Coalition task allocation of heterogeneous multiple agents with complex constr-

aints. *Control Theory & Applications*, 2020, **37**(11): 2413−2422 (唐嘉钰, 李相民, 代进进, 薄宁. 复杂约束条件下异构多智能体联盟任务分配. 控制理论与应用, 2020, **37**(11): 2413−2422)

30 Elhoseny M, Tharwat A, Hassanien A E. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *Journal of Computational Science*, 2018, **25**: 339−350

31 Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation. Shanghai, China: IEEE, 2011. 2520−2525

鞠 锴 南京航空航天大学自动化学院硕士研究生. 主要研究方向为多智能体系统任务分配.
E-mail: jk_emails@163.com
(**JU Kai** Master student at the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. His main research interest is task allocation for multiagent systems.)

冒泽慧 南京航空航天大学自动化学院教授. 主要研究方向为具有干扰与微小渐变故障的系统故障诊断与容错控制, 高速列车与航天器飞行控制应用. E-mail: zehuimao@nuaa.edu.cn
(**MAO Ze-Hui** Professor at the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. Her research interest covers fault diagnosis and fault-tolerant control of systems with disturbance and incipient faults, and high-speed train and spacecraft flight control applications.)

姜 斌 南京航空航天大学自动化学院教授. 主要研究方向为故障诊断与容错控制及其在飞机、卫星和高速列车中的应用. 本文通信作者.
E-mail: binjiang@nuaa.edu.cn
(**JIANG Bin** Professor at the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. His main research interest is fault diagnosis and fault-tolerant control and their applications in aircraft, satellites, and high-speed trains. Corresponding author of this paper.)

马亚杰 南京航空航天大学自动化学院教授. 主要研究方向为自适应故障诊断, 容错控制.
E-mail: yajiema@nuaa.edu.cn
(**MA Ya-Jie** Professor at the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. His research interest covers adaptive fault diagnosis and fault-tolerant control.)