

上海交通大学硕士学位论文

空空导弹分布式任务分配技术研究

硕士研究生：张贊

学号：118032910119

导师：蔡云泽研究员

申请学位：工程硕士

学科：控制工程

所在单位：自动化系

答辩日期：2020年12月24日

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

**RESEARCH ON DISTRIBUTED TASK
ASSIGNMENT TECHNOLOGY OF
AIR-TO-AIR MISSILES**

Candidate:	Zhang Yun
Student ID:	118032910119
Supervisor:	Prof. Cai Yunze
Academic Degree Applied for:	Master of Engineering
Speciality:	Control Engineering
Affiliation:	Department of Automation
Date of Defence:	December 24, 2020
Degree-Conferring-Institution:	Shanghai Jiao Tong University

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

上海交通大学

学位论文使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于 公开论文

内部论文， 1年/ 2年/ 3年 解密后适用本授权书。

秘密论文，____年（不超过10年）解密后适用本授权书。

机密论文，____年（不超过20年）解密后适用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：

指导教师签名：

日期： 年 月 日

日期： 年 月 日

空空导弹分布式任务分配技术研究

摘要

现代空战中，空空导弹是一种杀伤力强、威慑度高的尖端武器，对夺取战场制空权和主动权具有重要的意义。在空战环境中，常常面对着多目标、多任务的场景，发展空空导弹对集群目标的攻击作战模式是一个重要方向。但随着大规模集群作战单元投入实战的趋势，在空空导弹拦截大规模集群的场景中，由于集中式算法求解大规模任务分配问题的性能面临通信能力和实时性的限制，因此研究使用分布式算法求解任务分配问题具有重要的现实意义。

本文首先结合空空导弹的特点，系统介绍了分布式空空导弹任务分配技术的研究现状及存在的主要问题。接着，本文依据空战的实际特点和需求，考虑空战中的约束条件，建立了整数规划模型。然后，针对空战强博弈的特点，本文利用博弈论的思想，建立了两种博弈论模型，研究并改进了在两种博弈论模型下的分布式任务分配算法，通过仿真与对比试验验证了其有效性，且在大规模问题和非平衡问题优化效果和实时性方面具有显著的优越性。最后，针对空战高动态的特点，本文研究了基于随机博弈模型下的动态任务分配算法，并针对几种非理想场景设计了重分配机制，通过仿真实验研究了分配算法的动态适应性。

本文的主要研究贡献主要有：

1、针对空空导弹任务分配问题，以最短预计剩余攻击时间和最小导弹机动量为优化目标，建立了整数规划模型。利用博弈论中的势博弈模型，结合势博弈在多智能体系统控制中的应用，设计了基于拉格朗日乘子法的含约束条件下智能体效用函数，从理论上证明了其可行性和有效性。改进了博弈学习算法中的 SAP 算法，设计了任务交易机制，以解决 SAP 算法过早收敛，陷入局部最优的问题。通过设计仿真

场景验证了势博奕模型及其学习算法解决任务分配问题的有效性和实时性，并通过与集中式算法的对比验证了本文设计的模型与算法在求解有约束条件下的平衡与非平衡任务分配问题时均能够保留无约束优化时的有效性和实时性，因而具有良好的实用性和可扩展性。

2、针对上述整数规划模型，利用博奕论中的偏好联盟博奕模型，将任务分配问题转化为种群聚类问题，首先结合空战约束条件设计了享乐博奕模型下的关于种群数量的个体效用函数，接着从理论方面分析了该算法的最坏优化值及迭代步数，使用改进的 SAP 算法和一种针对偏好联盟模型的分布式一致性算法分别对博奕模型进行学习优化，并通过仿真对比实验验证了模型与算法的在非平衡指派问题中的有效性和实时性。

3、针对高动态场景下需要及时调整任务分配的问题，研究了基于随机博奕模型的动态任务分配问题。设计了导弹与目标对战随机博奕场景框架，在假定静态模型区间内利用势博奕模型建立并求解任务分配问题，同时在模型状态切换的时间点，设计滚动策略集，兼顾智能体决策的连续性和对新生事件的及时响应性。

关键词：空空导弹，任务分配，分布式，势博奕，享乐联盟博奕，随机博奕

RESEARCH ON DISTRIBUTED TASK ASSIGNMENT TECHNOLOGY OF AIR-TO-AIR MISSILES

ABSTRACT

In modern air warfare, air-to-air missiles are sophisticated weapons with strong lethality and high deterrence. They are of great significance to air dominance and initiative on the battlefield. An air combat environment is often faced with multi-target and multi-task scenarios, thus it is an important direction to develop an air-to-air missile attack combat mode against cluster targets. However, with the trend of large-scale UAV clusters being put into actual combat, in the scenario of air-to-air missiles intercepting large-scale clusters, centralized task assignment algorithms are limited to communication limitations and real-time problems. Therefore, the use of distributed algorithms to solve the task assignment problem have important practical significance.

This article first systematically introduces the related work and main problems of air-to-air missile task assignment technology, combined with the characteristics of air-to-air missiles. Then, according to the actual characteristics and requirements of air combat, this article takes the constraints of air combat in consideration, and establishes an integer programming model. In view of the characteristics of the strong game of air combat, this article introduces the idea of game theory to establish two game theory models, researches and improves the distributed task assignment algorithm under the two game theory models. These algorithms are verified effective and significantly advantageous in real-time large-scale problems through simulation and comparative experiments. Finally, in view of the high dynamic characteristics of air combat, this article studies the task reassignment algorithm

based on the two game theory models proposed in three scenarios, and studies the dynamic adaptability of the reassignment algorithm through simulation experiments.

The main research contributions of this article are as follows:

1. Aiming at the task allocation problem of air-to-air missiles, with the shortest expected remaining attack time and the minimum missile maneuver as the optimization objectives, an integer programming model is established. Using the potential game model in game theory, combined with the application of potential game in the control of multi-agent systems, the individual utility function and negotiation protocol of the missile are designed under unconstrained conditions, and the effectiveness and real-time performance of the algorithm are analyzed through comparative experiments. Then, in view of the characteristics of integer programming problems, task assignment algorithms under constraints are proposed respectively based on Lagrange multiplier method and based on the adjusted Wonderful Life Utility (WLU) . Theoretical analysis and simulation comparison verify that the two algorithms retains the validity and real-time performance of unconstrained optimization, and has good practicability and scalability.

2. Aiming at the above integer programming model, using the hedonic game model in game theory, the task allocation problem is transformed into a population clustering problem. First, the individual utility function with respect to the number of population under the hedonic game model is introduced based on the air combat constraints, and then A distributed negotiation protocol is designed based on the Zeuthen negotiation strategy. The worst performance analysis of the algorithm are theoretically conducted, and the effectiveness and real-time performance of the algorithm are verified through simulation and comparison experiments.

3. Aiming at the problem of timely adjustment of task allocation in high dynamic scenarios, the dynamic task allocation problem based on stochastic

game model is studied. A stochastic game framework of missile and target battle is designed, and the potential game model is used to establish and solve the task allocation problem in the assumed static model interval. At the same time, at the time of state switching, a rolling strategy set is designed to take into account the continuity and matching of the agent's decision. Timely response to new events.

KEY WORDS: Air-to-Air Missile, Task Assignment, Distributed Algorithm, Potential Game, Hedonic Coalition Game, Stochastic Game

目 录

第一章 绪论	1
1.1 课题研究背景及意义	1
1.2 国内外研究现状	3
1.3 主要研究内容与章节结构安排	6
第二章 空空导弹任务分配模型	9
2.1 引言	9
2.2 弹目相对运动模型	9
2.3 剩余攻击时间估计	10
2.4 能量最优制导律	12
2.5 任务分配模型建立	14
2.5.1 基本模型	14
2.5.2 动态任务分配模型	15
2.6 本章小结	16
第三章 基于势博弈的分布式任务分配方法	19
3.1 引言	19
3.2 势博弈模型	19
3.3 面向任务分配问题的势博弈模型设计	20
3.3.1 WLU 效用函数	20
3.3.2 约束条件处理方法	21
3.4 面向任务分配问题的势博弈决策算法	25
3.4.1 博弈学习算法	25
3.4.2 任务交易	30
3.4.3 算法复杂度分析	31
3.5 仿真分析与对比	31
3.5.1 场景规模仿真分析	33
3.5.2 非平衡指派场景仿真分析	37
3.6 本章小结	38

第四章 基于偏好联盟博弈的分布式任务分配方法	41
4.1 引言	41
4.2 偏好联盟博弈模型	41
4.3 面向任务分配的偏好联盟博弈模型设计	43
4.4 基于偏好联盟博弈模型的决策算法	45
4.4.1 SAP 算法	45
4.4.2 分布式一致性算法	46
4.4.3 模型与算法性能分析	48
4.5 仿真分析与对比	52
4.5.1 导弹与目标数量非平衡程度影响仿真分析	53
4.5.2 目标之间分配数量非平衡程度影响仿真分析	55
4.6 本章小结	58
第五章 基于随机博弈模型的分布式动态任务分配框架	61
5.1 引言	61
5.2 随机博弈模型	61
5.3 面向动态任务分配问题的随机博弈框架设计	62
5.3.1 时间窗口划分	62
5.3.2 博弈子模型建立	63
5.3.3 博弈阶段切换	63
5.4 重分配触发机制	64
5.4.1 通信网络非连通	64
5.4.2 目标数量可变	66
5.5 系统设计与仿真验证	67
5.5.1 动态任务分配系统设计	67
5.5.2 仿真实验	68
5.6 本章小结	74
第六章 总结与展望	75
6.1 本文总结	75
6.2 未来展望	76
参考文献	77
致 谢	83

攻读学位期间发表（或录用）的学术论文	85
攻读学位期间获得的科研成果	87

插图索引

图 1-1 论文章节结构安排	7
图 2-1 弹目相对运动模型	9
图 2-2 导弹探测与通信示意图	16
图 3-1 平衡指派场景下算法优化效果对比	34
图 3-2 平衡指派场景下算法运行时间对比	35
图 3-3 固定导弹数量为 55 非平衡指派场景下算法优化效果对比	38
图 3-4 固定导弹数量为 55 的非平衡指派场景下算法运行时间对比	39
图 4-1 联盟划分	42
图 4-2 HCG 模型的回报函数	45
图 4-3 目标数量为 10 的非平衡指派场景下算法优化效果对比	53
图 4-4 目标数量为 10, 不同导弹数量场景下算法运行时间对比	54
图 4-5 20vs10 场景的不同 b_{\max} 条件下算法优化效果对比	56
图 4-6 不同 b_{\max} 条件下 JSFP 算法收敛到不可行解比例变化图	56
图 4-7 20vs10 场景的不同 b_{\max} 条件下算法运行时间对比	57
图 5-1 不可规避冲突	65
图 5-2 非连通场景下分配冲突消解示意图	66
图 5-3 导弹任务分配系统框架示意图	67
图 5-4 基于势博弈模型的动态任务分配效果	69
图 5-5 基于 HCG 模型的动态任务分配效果	70
图 5-6 导弹初始通信网络非连通的动态任务分配效果	72
图 5-7 新增目标场景下的动态任务分配效果	73

表格索引

表 3-1 平衡指派场景下算法性能对比表	34
表 3-2 平衡指派场景下算法运行时间对比表	35
表 3-3 固定导弹数量为 55 非平衡指派场景下算法性能对比表	39
表 3-4 固定导弹数量为 55 的非平衡指派场景下算法运行时间对比表	40
表 4-1 固定目标数量为 10 的非平衡指派场景下算法优化效果对比表	54
表 4-2 固定目标数量为 10 的分配场景下算法运行时间对比表	55
表 4-3 20vs10 场景的不同 b_{\max} 条件下算法性能对比表	59
表 4-4 20vs10 场景的不同 b_{\max} 条件下算法运行时间对比表	60

算法索引

算法 3-1 JSFP 算法流程	27
算法 3-2 GRMFMI 算法流程	29
算法 3-3 SAP 和 sSAP 算法流程	30
算法 3-4 基于势博弈模型的任务分配算法	32
算法 4-1 使用 SAP 的 HCG 决策算法流程	46
算法 4-2 MCA 算法中智能体 \mathcal{M}_i 的决策流程	47
算法 4-3 分布式一致性算法 (DMCA)	48

主要符号对照表

第一章 绪论

1.1 课题研究背景及意义

空战往往在现代战争中扮演着决定性的作用，在近年几次世界局部冲突中充分体现出制空权的重要性，而随着各国军事技术的发展，空战的无人化、集群化、智能化是一个必然的趋势，如在今年的亚美尼亚与阿塞拜疆的纳卡地区冲突中，阿塞拜疆的无人机作战对亚美尼亚造成了巨大杀伤。除了无人机以外，空空导弹作为空战中的一种高威胁高杀伤力武器，如果能将其实现集群化智能化攻击，则会大大增加空战取胜的概率，向军事现代化、智能化迈出重要一步。

空空导弹实现集群智能作战的关键技术包括协同探测技术、协同决策技术、协同制导技术等，贯穿了导弹从发现到打击目标的全流程。其中，空空导弹的协同决策系统是重中之重，其主要功能时负责导弹集群的任务分配，这里的任务可以包括目标的分配，可以是侦查、打击、评估等多任务分配，或者是子集群的划分。任务分配的目标可以概括为：在当前态势环境下对我方资源进行调配以实现我方效益的最大化。随着军事技术的发展，尤其是集群化作战规模逐渐增大的发展趋势，协同决策系统中的任务分配技术的重要性越来越突出，一方面拥有自主决策系统的导弹可以共享态势信息，从而更好地应对快速变化的空战态势，及时根据态势信息选择最优策略，最大化攻击效益。另一方面，虽然空空导弹经历了几代发展，已经具备了全向攻击的能力，但在实战中导弹的命中率仍然受发射条件和目标特性影响较大，尤其是最新一代战机在机动能力、隐身能力和干扰能力方面不断取得进步，为空空导弹的攻击带来了一定的难度。此时协同决策系统可提前布局，为导弹预先选好最优目标，尽量让所有导弹向着有利于命中目标的方向靠拢，从而提高导弹的整体打击效果。

针对任务分配问题，大量研究成果针对不同场景提出了各式各样的任务分配模型和算法，这些方法按照架构主要分为集中式和分布式。典型的集中式任务分配模型是混合整数规划模型（Mixed Integer Linear Programming, MILP），将任务分配问题视为一种组合优化问题。MILP 模型下常用的集中式算法包括匈牙利算法、分支定界法，以及各种启发式优化算法及其改进版本。这些算法下智能体大多没有自主性，只是服从决策中心作出的最优决策，因此本质上说没有体现协同的作用，且集中式算法在大规模集群中的性能也有明显的下降。

相对于集中式算法，分布式算法赋予了各智能体一定的自主性和决策能力，因此具有如下优势：（1）使得集群计算负载能够相对均衡地分配到各个节点上；（2）

对于部分连通或时变网路具有更好的鲁棒性；（3）不需要一个集中的数据收集处理中心，更适用于大规模集群；（4）响应迅速，面对态势变化，分布式架构下各导弹可以立即自行调整策略，及时响应。

但在实际空战场景中，由于受到空空导弹自身、任务目标需求及空战环境等因素的制约，空空导弹的分布式任务分配算法仍然是一个非常复杂的问题，其复杂性主要体现在以下几个方面：

（1）空战环境的复杂性。随着科技的发展，世界先进的导弹和战机都具备了高速高机动能力，且即使在恶劣天气下，也可全天时、全天候作战，这使得空空导弹作战的环境日趋复杂化，使得导弹难以对变化的态势及时作出响应；

（2）目标的异构性。集群作战，如战机集群，往往是由不同种类、不同分工的飞机组成，如一架预警机与多架战斗机制成的侦查护卫编队。因此在任务分配中需要考虑不同类型目标的差异，指派合适的导弹进行攻击；

（3）分布式架构的局限性。分布式架构下，一方面导弹间通信链路往往也是分布式的，受通信带宽和通信频率的限制，导弹获得信息可能是非同步、非一致的，这对任务分配问题带来了很大的困难；另一方面，以目标分配为例的任务分配问题本质上是一个组合优化问题，这是一个典型的 NP-hard 问题，在求解该类问题时，分布式优化算法相比于集中式算法仍存在着优化效果不好的缺点，尤其是在导弹仅能获得局部信息的前提下，如何获得全局最优仍是一个开放的问题。

目前分布式任务分配算法主要有基于市场机制的方法、分布式马尔科夫决策过程方法和博弈论方法。相较于前两种方法，博弈论运用于任务分配问题上的时间并不长，但其在解决分布式任务分配问题中具有独特的优势。首先，博弈论假设参与博弈的智能体均具有“理性决策能力”，即会追求自身利益最大化，因此相对于其他方法更接近于集群智能的概念；其次，博弈论的基本思想是参与博弈的智能体根据获取的外部信息，进行自我决策，实现自身利益最大化，可见使用博弈论思想天然契合分布式架构的需求；最后，参与博弈的智能体能够根据自身获得的信息进行决策的过程，因此导弹可在信息有限的情况下对不确定的或动态的环境及时作出反应。

但使用博弈论思想的任务分配方法仍有许多问题亟待解决。第一，由于智能体只追求自身效用最大化，因此为了解决任务分配问题，需要设计合适的效用函数使得智能体在提高自身效用的同时，最大化全局效用；第二，如何设计合适的协商策略，使得所有智能体的分配快速达到均衡状态；第三，如何保证博弈最终收敛的均衡状态尽量接近全局最优状态。

综上所述，使用博弈论思想建立空空导弹集群的分布式任务分配模型，并设

计快速、准确的分配协商策略具有重要的研究价值和现实意义。

1.2 国内外研究现状

目前国内外对基于博弈论的分布式优化方法解决类似于任务分配问题的研究越来越多。承接前文所述，目前对于该方面的研究主要集中于两个方面：博弈模型的建立和协商学习算法的设计。建立博弈模型的目的在于将任务分配问题转换为一个博弈模型下求解纳什均衡的问题。而参与博弈的智能体将会收敛到什么状态，是不是纳什均衡状态，收敛到哪个纳什均衡状态，是否是全局最优或次优，这些都是学习算法需要解决的问题。而学习算法往往是针对特定的博弈模型进行设计的，在目前在任务分配问题中，常用的博弈模型主要有势博弈模型，联盟形成博弈模型和随机博弈模型。

(1) 势博弈

在博弈模型的建立中，最关键的在于博弈模型的选择与效用函数的设计。关于博弈模型的选择，目前大多数文献中均使用势博弈模型（Potential Game, PG）作为任务分配问题的基础模型。PG 由 Shapely 在 1996 年提出^[1]，近年来 PG 受到学者们关注的原因是 PG 模型与分布式计算天然的契合度。在 PG 模型下智能体自身效用的变化能够同向地反应在全局效用变化上，这样各智能体只需要“自私”地优化自身效用即可提高全局效用。PG 具有的另一个良好性质是，大量文献已经证明，PG 模型必然存在纯纳什均衡，且对于有限策略集博弈，PG 可以在有限次迭代中收敛到纯纳什均衡，这意味着 PG 模型下的优化问题必然可以在多项式时间内获得收敛解。

广义的任务分配问题包含资源调配、性能优化等问题。目前，PG 模型已经被成功被应用于多个领域的相关问题中，并取得了不少成果。博弈理论最初即是应用于经济领域，而 PG 模型已经被广泛应用于经济领域中的定价模型建模，文献 [2] 建立了基于 PG 模型多市场主体的竞争定价模型；文献 [3] 针对拥堵收费定价问题，使用双层规划和 PG 模型结合的方法构建了拥堵收费定价模型。在能源网络方面，文献 [4] 将 PG 模型用于建立多能源微网调度模型，实现能源链的分布式调度。在计算机网络领域，PG 模型被用于解决边缘计算资源最优调配^[5]，任务分拆调度^[6, 7]，针对恶意攻击的分布式优化安全状态估计^[8]，多层次算力网络中的代价感知任务调度^[9] 等问题。此外，在无线传感器网络（Wireless Sensor Network, WSN）领域，传感器网络拓扑控制^[10, 11]、无线资源管理^[12, 13]、节点定位^[14]、目标定位率控制^[15] 等问题，多智能体系统（Multi-agent System, MAS）中的群智能分布式任务调度^[16] 等问题也均有相关研究使用 PG 模型尝试解决。

在狭义的目标分配问题方面, Arslan 等人^[17]最早将 PG 思想引入对目标的分配问题中, 通过设计合适的参与者效用函数, 建立起 PG 模型, 并比较分析了多种效用函数的优缺点, 但没有研究带约束的分配问题。文献 [18] 研究了基于 PG 模型的雷达网目标跟踪分配模型, 加入了约束条件并从理论上证明了可行性, 但其模型并没有真正的实现分布化。以上文献只考虑了静态的目标分配问题, 针对动态目标分配, 文献 [19] 提出了一种短期视野的方法, 通过将动态过程分解为静态问题的序列实现对动态目标的分配。在此基础上, 文献 [20] 提出将智能体完成任务的代价函数引入模型, 该代价函数与智能体状态有关, 从而设计出合适的状态相关效用函数。

在 PG 模型下, 常用的学习算法包括二元 log-linear 学习算法^[21]、虚拟博弈算法^[22]、后悔匹配算法^[23]和空间自适应学习算法^[24]等。这些算法的共性在于根据博弈参与者采取不同决策后的自身效用获得行为策略, 在此基础上不同的算法会引入诸如惰性机制、历史记忆机制、softmax 层等方法进行筛选, 确定最终决策。这些算法大都能够以接近 1 的概率收敛到纳什均衡状态, 但收敛的速度、寻优能力不尽相同。

总体来说, PG 模型应用于任务分配问题已经出现不少研究, 但目前对于含约束条件的分配问题, 如何在分布式架构下建立得到约束可行解的模型, 并找到全局最优解都仍是一个开放的问题。

(2) 联盟形成博弈

除了 PG 模型以外, 联盟形成博弈模型 (Coalition Formation Game, CFG) 也是一种常用的博弈模型, CFG 是合作博弈模型的一种, 在博弈过程中参与者会自行组成若干个联盟, 从全局来看即形成了若干分组。偏好联盟形成博弈 (Hedonic Coalition Formation Game, HCFG) 是一类典型的 CFG 模型^[25], 它最早用于研究经济学中市场主体的联盟模型^[26], 正如其名, HCG 中的参与者对于各个联盟存在着偏好关系, 并最终会选择自己最喜好的联盟加入。正因为 HCG 的这一性质, 之后它被越来越多地应用于多种场景下的分布式优化问题中, 包括无线通信网络^[27]、认知网络^[28]、无人机^[29]等场景中的资源调配、安全传输^[30]、任务分配^[31]等问题。

在 HCFG 的基础上, 对于博弈参与者的偏好关系进行不同的设计, 可以得到偏好联盟博弈模型的衍生模型。比如引入参与者之间的评价值, 并由参与者对联盟成员的评价值求出对联盟的偏好, 如果偏好是由成员评价之和得到, 则可得到可加可分离偏好联盟博弈 (Additive Separable Hedonic Game, ASHG), 文献 [32] 对其稳定性和收敛性进行了详细分析并给予证明。如果偏好是由成员评价平均值得到, 则可得到局部偏好联盟博弈^[33] (Fractional Hedonic Coalition Game, FHCG)。

FHCG 模型被广泛与与社交网络分析相结合，如解决社会福利最大化问题^[34, 35]。

在任务分配领域，HCG 解决任务分配问题的思路是假设任务需要若干博弈参与者组成联盟共同完成，则博弈参与者选择任务的过程便可看成是寻找最优分组的过程。文献 [36] 提出了针对大规模智能体集群系统的联盟形成框架，介绍了效用设计、联盟选择的通用方法；文献 [37] 认为博弈参与者对联盟的偏好仅与联盟成员个数有关，并证明了该偏好关系下 HCG 模型的收敛性和性能下界，通过仿真验证了其在全连通网络和部分连通网络，以及大规模网络分配中的有效性。针对前面文献设计的模型中只涉及同构智能体的情况，文献 [38] 研究了异构智能体组成的 HCG 模型，在偏好函数的设计中加入了依赖于参与者的因素。

在 HCG 模型中，常用的学习算法常常是基于博弈参与者对于联盟的更换决策。当没有参与者有更换联盟的意愿时，模型便收敛到纳什均衡状态，但此时往往是局部最优的。在此基础上，一些文献加入了如联盟交换^[28]、随机更换联盟、联盟划分一致性算法^[37]等机制，以期望避免模型陷入局部最优，提高全局优化能力。

综上所述，在任务分配的研究中，使用 HCG 模型的研究尚不多，但对于多种场景下的分配问题都有研究涉及，且取得了可靠的效果。然而与 PG 模型下的局限一样，对于带约束问题的分配问题建模，目前仍较少有研究涉及。

(3) 随机博弈

随机博弈 (Stochastic Game, SG)，又称为马尔可夫博弈 (Markov Game, MG)，是一种常用于描述多个智能体在多种状态下相互交互状态的博弈模型，在 SG 模型下，模型可分为若干个阶段，在每个阶段中智能体形成一个阶段博弈，进行决策并获得收益，SG 模型中参与者的收益往往设置为未来回报的折现和，因此 SG 模型可看作是博弈论在传统马尔可夫决策过程 (Markov Decision Process, MDP) 中的推广^[39]。在智能体做出决策后，模型会由一个状态转移到另一个状态，转移的概率服从一个与智能体决策有关的概率分布。由于随机博弈模型扩展性好，且适用于描述多种场景，因此 SG 模型被广泛用于多智能体系统的研究中，且可与多种算法相结合，如文献 [40] 中就使用禁忌搜索算法对网络安全攻防随机博弈中的防御策略进行求解。文献 [41] 中使用 SG 模型研究了复杂微电网中的同步、稳定与均衡优化问题。文献 [42] 提出一种基于状态的 PG 模型，本质上是一般 SG 模型的简化，即博弈参与者的回报不再关注未来回报，而只关注当前状态下的回报，并研究了该模型下的纳什均衡收敛性和学习算法。

在任务分配领域，由于在实际中任务常常是处于动态过程中的，因此 SG 模型可用来对动态任务分配问题建模。在这类模型中，通常将 SG 模型的每一个阶

段博弈看作是一个任务分配问题，同时明确状态转移方式。文献[43]中都使用SG模型对无线网络的频谱资源调配进行建模，实现能量损耗最小化。文献[19]中针对机器人城市搜救这一实际场景，使用SG模型建立任务分配框架，并在每一个阶段博弈建立起PG模型求解，提出了解决动态任务分配问题的方案，仿真结果表明该框架可实现整体搜救时间最小化和搜救伤员数量最大化。

由于强化学习思想基本是以马尔可夫过程为前提，因此随机博弈与强化学期的结合涌现出大量的研究成果^[44]，如MinMax-Q算法、Nash-Q学习算法^[45]等。与前面所列文献研究每个阶段博弈的决策不同的是，强化学习希望得到的是针对SG模型状态的价值函数估计，从而可根据价值估计获得每个状态下的最优决策，因此对动态适应性更好，但学习难度也更大^[42]。

总的来说，SG模型适用于动态模型的建模，目前SG模型在多智能体系统中与强化学习的结合是一个热门方向，但仍有诸如多智能体的随机博弈下的纳什均衡特性等关键问题亟待解决，

1.3 主要研究内容与章节结构安排

本文针对空空导弹分布式任务分配问题，使用博弈论的思想对任务分配模型、任务分配算法和动态任务分配框架展开研究，本文的主要研究内容分为三块：

(1) 基于势博弈理论，设计导弹自身的效用函数，建立带约束条件的混合整数规划任务分配模型，证明该模型的可行性、有效性与性能下界，并针对该模型设计势分布式协商算法，通过仿真验证提出的模型与算法的有效性。

(2) 基于偏好联盟博弈理论，设计联盟效用函数，建立带约束条件的联盟划分模型。从理论上推导模型的可行性、收敛性以及性能下界控制条件；设计适用于偏好模型的分布式协商算法，通过仿真验证模型和算法的有效性。

(3) 基于随机博弈理论，设计空空导弹的动态任务分配系统框架。将动态分配过程划分为若干时间窗口，假设一段时间窗口内最优分配方案不变，在该窗口内建立博弈子模型并求解；设计博弈阶段切换机制，包括初始解的生成与新解的选择机制；针对网络非连通和目标数量变化两种特殊场景设计重分配触发机制；通过仿真实验验证该框架对于动态场景的有效性和自适应性。

本文的章节组织结构安排如图1-1所示，具体安排如下：

第一章为绪论部分，首先介绍了选题背景等相关知识，简要阐述了集群空战背景下任务分配技术发展的必要性，对分布式任务分配技术的优缺点进行了归纳，并对基于博弈论思想的任务分配方法做了简要的梳理与分析，为后序博弈模型的建立和算法的设计做了铺垫。

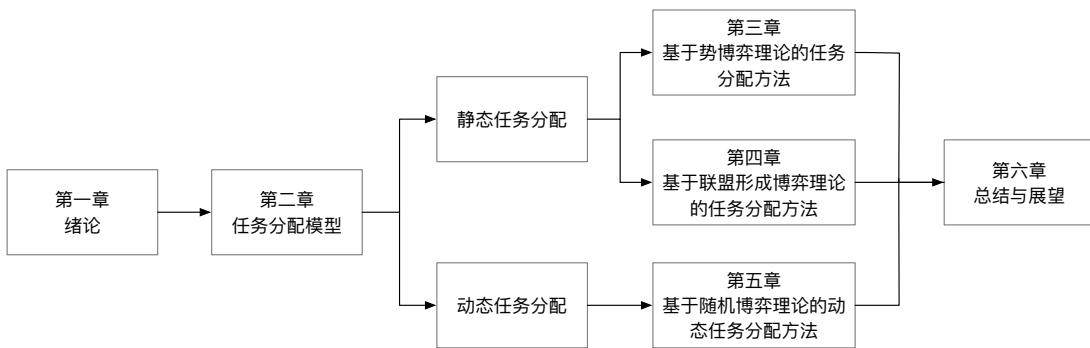


图 1-1 论文章节结构安排

Figure 1-1 Structure arrangement of thesis chapters

第二章为空空导弹任务分配模型，建立了某一时间点的静态任务分配模型和一段时间的动态任务分配模型两种模型。针对静态任务分配模型，从空空导弹的特性出发，以最小化总攻击时间和最小化导弹机动量为目标，基于混合整数规划模型建立了多对多任务分配模型。针对动态任务分配模型，引入了导弹探测半径与通信半径的概念，建立了导弹对目标的探测模型以及导弹之间的通信模型，从而建立起动态场景下的任务分配模型。

第三、四章研究了针对某一时间点的分布式静态任务分配模型及其算法。

第三章研究了基于势博弈理论的分布式任务分配方法，基于势博弈理论，基于 Lagrange 乘子法改进了 WLU 效用函数，建立了带约束条件的混合整数规划任务分配模型，通过数学推导证明了改进效用函数的有效性及性能下界。使用并比较了三种针对势博弈模型的分布式协商策略算法的有效性与性能，并改进了 SAP 算法，通过与集中式启发式算法的仿真对比实验，验证了设计模型与算法的有效性。

第四章基于偏好联盟博弈理论，将任务分配问题看作联盟划分问题，设计了针对联盟划分问题的联盟效用函数，建立了带约束条件的偏好联盟博弈任务分配模型。从理论上推导出该模型的性能下界，得到了提高模型性能的方法。针对该模型设计了针对偏好联盟博弈模型的协商算法，改进了 SAP 使其适应于偏好模型，并使用 DMEA 算法求解。通过仿真实验验证了提出模型和算法的有效性。

第五章研究了动态场景下的分布式任务分配模型及其算法。基于随机博弈理论，将导弹攻击过程划分为若干博弈子阶段，并在每个阶段建立三、四章建立的静态模型并求解。为了博弈子阶段之间的切换连贯，设计了子阶段分配前初始解的交叉生成机制和分配后新解的惰性机制。针对导弹通信网络非连通，以及目标数量发生变化两种场景进行分析并相应设计了重分配触发机制。最后通过场景仿

真验证了所提出的动态分配框架的有效性，且对于动态环境有较好的应变能力。

第六章总结了全文的主要工作和主要贡献，并指出了本文的不足，对下一步工作也提出了展望。

第二章 空空导弹任务分配模型

2.1 引言

在任务分配问题中，首先需要根据导弹自身传感器或其他支持平台收集得到的数据和信息，估计战场态势，建立多对多的任务分配模型。在空空导弹的作战场景下，本节选择了预计剩余攻击时间和当前导弹机动量是两个较为显著且易于获得的影响要素，首先分别介绍了两个要素的估计方法，之后以最小化总攻击时间和最小化导弹机动量为目标函数，建立多对多任务分配模型。

2.2 弹目相对运动模型

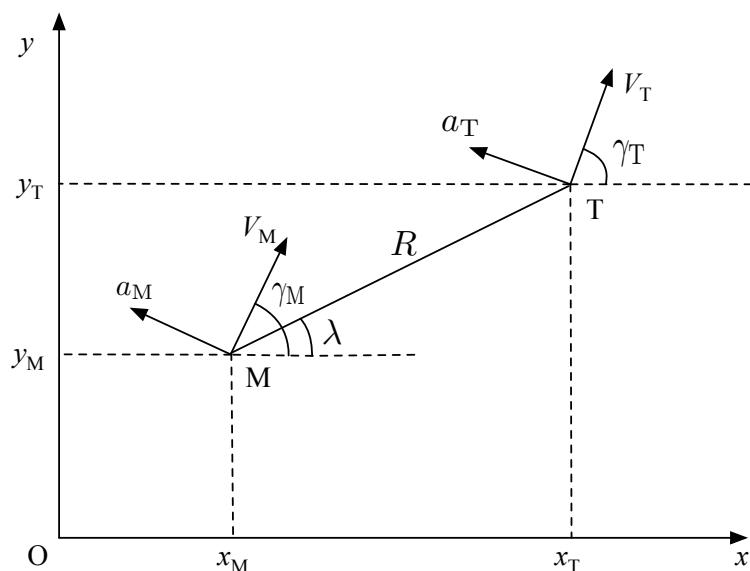


图 2-1 弹目相对运动模型

Figure 2-1 Relative motion model of missile and target

在建立多对多任务分配模型之前，需要先建立一对一的空战模型。本文所考虑的模型是在二维平面内，不考虑导弹和目标具体形状的影响，即将导弹和目标当作质点处理。图2-1展示的是本文所使用的导弹与目标相对运动模型，其中 M 表示导弹， T 表示目标， $[x_{(.)}, y_{(.)}]$, $V_{(.)}$, $a_{(.)}$, $\gamma_{(.)}$ 分别表示导弹和目标的位置、速度、加速度和航向角。 r 为弹目距离， λ 表示弹目视线角。

为了建立弹目相对运动模型，需要作出以下假设：1、导弹与目标速度均为常值；2、目标的速度小于导弹的速度。由此分别建立导弹和目标的运动方程为：

$$\dot{x}_M = V_M \cos \gamma_M, \quad \dot{y}_M = V_M \sin \gamma_M, \quad \dot{\gamma}_M = \frac{a_M}{V_M}, \quad (2-1)$$

$$\dot{x}_T = V_T \cos \gamma_T, \quad \dot{y}_T = V_T \sin \gamma_T, \quad \dot{\gamma}_T = \frac{a_T}{V_T}. \quad (2-2)$$

令 V_R, V_λ 分别表示弹目相对速度平行于和垂直于视线角方向的分量， V_R 表示弹目相对接近速度，则弹目相对运动方程为：

$$V_R \equiv \dot{R} = V_T \cos \sigma_T - V_M \cos \sigma_M, \quad (2-3)$$

$$V_\lambda \equiv R\dot{\lambda} = V_T \sin \sigma_T - V_M \sin \sigma_M, \quad (2-4)$$

其中 $\sigma_{(\cdot)}, (\cdot) \in \{M, T\}$ 分别表示是导弹和目标的前置角，其定义为：

$$\sigma_M = \gamma_M - \lambda, \quad \sigma_T = \gamma_T - \lambda. \quad (2-5)$$

由此可求得视线角变化率：

$$\dot{\lambda} = \frac{V_T \sin \sigma_T - V_M \sin \sigma_M}{R}. \quad (2-6)$$

2.3 剩余攻击时间估计

在导弹攻击过程中，为了使得预计击中时间最短，需要估计剩余攻击时间，本节采用对目标状态投影到弹目视线方向，将目标转化为相对静止状态的思想，实现对机动目标的剩余攻击时间估计。

首先假设目标静止，即 $V_T = 0$ 。此时由式 (2-1)、(2-5) 和 (2-6) 可得：

$$\dot{\sigma}_M = \frac{a_M}{V_M} + \frac{V_M \sin \sigma_M}{R}, \quad (2-7)$$

结合式 (2-3) 可得：

$$\frac{d\sigma_M}{dR} = -\frac{a_M}{V_M^2 \cos \sigma_M} - \frac{1}{R} \tan \sigma_M. \quad (2-8)$$

为方便叙述，令 $\rho = \sin \sigma_M$ ，代入式 (2-8) 可得：

$$\frac{d\rho}{dR} = -\frac{a_M}{V_M^2} - \frac{1}{R}\rho. \quad (2-9)$$

此处为方便推导，考虑导弹采用比例制导律，制导指令为：

$$a_M = NV_M \dot{\lambda}, \quad (2-10)$$

其中 N 为制导比例系数。将式 (2-10) 和 (2-6) 代入式 (2-9) 可得：

$$\frac{d\rho}{dR} = (N-1)\frac{\rho}{R}. \quad (2-11)$$

式 (2-11) 是一个一阶微分方程，其解形式为：

$$\rho = \rho_0 \left(\frac{R}{R_0} \right)^{N-1}, \quad (2-12)$$

其中 ρ_0 和 R_0 分别为初始时刻的对应变量值。将式 (2-3) 两边对 R 求积分，并进行泰勒展开得到：

$$\begin{aligned} t_f &= \frac{1}{V_M} \int_0^{R_0} \frac{1}{\sqrt{1-\rho^2}} dR \\ &= \frac{1}{V_M} \int_0^{R_0} \left(1 + \sum_{i=1}^{\infty} \frac{2i-1}{2^i} \rho^{2i} \right) dR, \end{aligned} \quad (2-13)$$

其中 t_f 为初始时刻导弹已经飞行的时间。为简化问题，仅取式 (2-13) 泰勒展开的前两项代入式 (2-12) 求解，并使用小角度假设 $\rho \approx \sigma_M$ 可得：

$$t_f = \frac{R_0}{V_M} \left(1 + \frac{1}{2(2N-1)} \sigma_{M0}^2 \right). \quad (2-14)$$

因此，若在每个时刻均取该时刻视线角方向作为 x 轴，则可得目标静止时在每个时刻的剩余攻击时间的估计值为：

$$t_{go} = \frac{R}{V_M} \left(1 + \frac{1}{2(2N-1)} \sigma_M^2 \right). \quad (2-15)$$

针对机动目标对式 (2-15) 作出进一步修正，采用虚拟目标的思路，将 V_T 和 a_T 对弹目相对运动的影响均投影到弹目视线角方向上，并修正由目标机动引起的航向角和视线角变化，将目标转化为相对静止状态。

首先得到目标机动引起的目标航向角和视线角的变化分别为：

$$\hat{\sigma}_T = \frac{a_T}{V_T} \cdot \hat{t}_{go}^p, \quad \hat{\lambda} = \frac{V_T \sin \sigma_T}{R} \cdot \hat{t}_{go}^p, \quad (2-16)$$

其中 \hat{t}_{go}^p 表示前一时刻对 t_{go} 的估计值，修正后的剩余攻击时间估计为：

$$\hat{t}_{go} = \frac{R}{V_{MT}} \left(1 + \frac{1}{2(2N-1)} \sigma_M^2 \right) \quad (2-17)$$

其中 $V_{MT} = V_M - V_T \cos(\sigma_T + \hat{\sigma}_T - \lambda - \hat{\lambda})$ 。

2.4 能量最优制导律

除了预计攻击时间最小化以外，由于导弹在攻击过程中大部分时间处于无动力滑翔阶段，在机动追踪目标时会不断消耗能量，因此为了保证击中目标前导弹动能不会耗尽，需要追求导弹机动程度最小化。实际中导弹系统具有高阶动态特性和非线性，难以利用最优控制原理求得闭环解，但对于只考虑导弹低阶系统特性和二次型性能指标取极小的最优控制问题，可以求得闭环解。本节将在导弹在2.3小节估计剩余攻击时间的基础上，基于最优控制原理，设计使得导弹机动最小化的最优制导律。

首先假设目标不机动，即 $a_T = 0$ 将2.2节建立的弹目相对运动模型改写为惯性系下的状态方程形式：

$$\dot{x} = Ax + Bu, \quad (2-18)$$

$$A = \begin{bmatrix} 0 & I_2 & 0 \\ 0 & 0 & I_2 \\ 0 & 0 & -\frac{1}{\tau} I_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} I_2 \end{bmatrix}, \quad (2-19)$$

其中状态向量为 $x = [R^T, \dot{R}^T, a_M^T]^T$ ，各状态分量分别为对应变量的向量形式， I_2 为二阶单位矩阵， τ 为导弹的等效时间常数。

设二次型的性能指标为

$$\min J = \frac{1}{2} \int_{t_0}^{t_f} u^T u dt, \quad (2-20)$$

$$\text{s.t. } R(t_f) = 0. \quad (2-21)$$

其中 t_0 和 t_f 分别表示制导开始和结束的时间，式 (2-20) 可减小机动能量的消耗，有利于保持弹道平直，减小损耗增大射程。式 (2-21) 可约束末端脱靶量为 0。

上述最优控制问题可求解闭环最优控制率形式为

$$u = (\mathbf{B}^T \mathbf{P} k^{-1}) \mathbf{M}, \quad (2-22)$$

其中 k 是当 $u = 0$ 时的预计末端脱靶量，可由下式求得：

$$\mathbf{M} = \mathbf{P}^T \mathbf{X}. \quad (2-23)$$

由下列微分方程

$$\dot{\mathbf{P}} + \mathbf{A}^T \mathbf{P} = 0, \quad \mathbf{P}(t_f) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad (2-24)$$

可求得 \mathbf{P} 的闭环表达式为

$$\mathbf{P} = \begin{bmatrix} 1 \\ t_{\text{go}} \\ \tau^2(e^{-T} + T - 1) \end{bmatrix}, \quad (2-25)$$

其中 t_{go} 是预计剩余攻击时间， $t_{\text{go}} = t_f - t$ ， $T = t_{\text{go}}/\tau$ 。由

$$k = - \int_t^{t_f} (\mathbf{P}^T \mathbf{G} \mathbf{G}^T \mathbf{P}) dt, \quad (2-26)$$

可求得 k 等于

$$k = \tau^3 \left(\frac{1}{2} e^{-2T} + 2T e^{-T} - \frac{T^3}{3} + T^2 - T - \frac{1}{2} \right). \quad (2-27)$$

将式 (2-25) 和式 (2-27) 代入式 (2-22) 和式 (2-23) 求得最优制导律闭合解为

$$u = \frac{N'}{t_{\text{go}}^2} [\mathbf{R} + \dot{\mathbf{R}} t_{\text{go}} - \tau^2(e^{-T} - 1 + T) \mathbf{a}_M], \quad (2-28)$$

其中 N' 为扩展比例系数：

$$N' = T^2(e^{-T} - I_2 + T)(-\frac{1}{2}e^{-2T} - 2Te^{-T} + \frac{1}{3}T^3 - T^2 + T + \frac{1}{2}I_2)^{-1}. \quad (2-29)$$

在目标机动的情况下，可对式(2-28)进行修正，限于篇幅，这里不再展开，直接给出对于机动目标的最优制导律：

$$u = \frac{N'}{t_{\text{go}}^2} [\mathbf{R} + \dot{\mathbf{R}} t_{\text{go}} - \tau^2 (e^{-T} - 1 + T) \mathbf{a}_M - \frac{t_{\text{go}}^2}{2} \mathbf{a}_T]. \quad (2-30)$$

最优性能指标 J^* 为

$$J^* = \frac{1}{2} \mathbf{X}^T(t_0) \mathbf{P}(t_0) \mathbf{P}^T(t_0) \mathbf{X}(t_0). \quad (2-31)$$

2.5 任务分配模型建立

2.5.1 基本模型

假设我方有 n_m 枚导弹，其集合表示为 $\mathcal{M} := \{\mathcal{M}_1, \dots, \mathcal{M}_{n_m}\}$ ；战场上 n_t 个目标需要攻击，其集合表示为 $\mathcal{T} := \{\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_{n_t}\}$ ，其中 \mathcal{T}_0 表示空目标。本文只考虑导弹数量不少于目标数量，即 $n_m \geq n_t$ 的情况。每枚导弹可根据自身能力、所处环境等因素选择部分目标作为可攻击对象，设 $\mathcal{A}_i \subset \mathcal{T}$ 表示第 i 枚导弹的可选目标集。特别地，为后续叙述方便，假设 $\mathcal{T}_0 \in \mathcal{A}_i, i = 1, \dots, n_m$ ，即所有导弹总是可以选择不攻击任何目标。^① 导弹 i 从自己的可选目标集 \mathcal{A}_i 中选出自己的目标 a_i ，则数组 $a = (a_1, \dots, a_{n_m}) \in \mathcal{A}$ 组成了一个分配解。设 $S(a) = \{S_1, \dots, S_{n_t}\}$ ，其中 S_j 表示分配解 a 下选择目标 j 的导弹集合， s_j 表示 S_j 中的导弹个数，即 $s_j = |S_j|, j = 1, \dots, n_t$ 。

选取一个关于分配解 a 的函数 $U_g(a)$ 作为全局效用函数，建立任务分配模型：

$$\max_{a \in \mathcal{A}} U_g(a) = \sum_{\mathcal{T}_j \in \mathcal{T}} U_{\mathcal{T}_j}(a) \quad (2-32)$$

$$\text{s.t. } s_j \leq b_{\max}^{(j)}, j = 1, \dots, n_t \quad (2-33)$$

$$s_j \geq 1, j = 1, \dots, n_t, \quad (2-34)$$

其中 $b_{\max}^{(j)}$ 表示选择目标 j 的导弹的最大数量和最小数量，且有 $b_{\max}^{(j)} \geq 1$ 。因此式(2-33)表示目标 j 不得被超过 $b_{\max}^{(j)}$ 枚导弹攻击，式(2-34)代表每个目标必须有导弹攻击，但由于实际中往往使用多于目标数量的导弹攻击，因此在式(2-34)的约束下，式(2-34)自然成立，因此可将上述问题进一步简化为

^① 显然，根据实际场景以及约束条件可知导弹不攻击任何目标是不符合实际要求的，本文在此作出的假设仅仅是为了后面章节叙述和论证需要，无任何实际意义，对问题的论述和求解也无任何影响。

$$\max_{a \in \mathcal{A}} U_g(a) = \sum_{\mathcal{T}_j \in \mathcal{T}} U_{\mathcal{T}_j}(a) \quad (2-35)$$

$$\text{s.t. } s_j \leq b_{\max}^{(j)}, j = 1, \dots, n_t. \quad (2-36)$$

$U_{\mathcal{T}_j}(a)$ 表示与目标 \mathcal{T}_j 有关的任务效用，为了体现预计剩余攻击时间最小和导弹机动能量消耗最小的目标，利用2.3小节和2.4小节的内容，将任务效用函数设计为：

$$U_{\mathcal{T}_j}(a) = r_j - \sum_{\mathcal{M}_i \in S_j} c_{ij}, \quad (2-37)$$

其中， r_j 是目标 \mathcal{T}_j 的价值，本文假设该值仅与目标属性有关，与攻击的导弹无关； c_{ij} 表示导弹 \mathcal{M}_i 攻击目标 \mathcal{T}_j 的成本函数，其定义为

$$c_{ij} = w_1 t_{ij} + w_2 J_{ij}, \quad (2-38)$$

其中， w_1 和 w_2 为权重值，代表对于攻击时间和攻击消耗能量的权衡程度； t_{ij} 表示导弹 \mathcal{M}_i 攻击目标 \mathcal{T}_j 的预计剩余攻击时间。 J_{ij} 为导弹 \mathcal{M}_i 攻击目标 \mathcal{T}_j 的预计机动能量消耗可由式 (2-31) 得到。

2.5.2 动态任务分配模型

前一小节所介绍的基本任务分配模型只是用于某一静止时间点或短期时间内的任务分配模型，为了研究在高度动态的空战模型中的任务分配问题，本节建立了分布式的动态任务分配模型。本节从导弹对目标的探测和导弹之间的通信入手建立模型。

在实际战场上，导弹发射之前往往是由地面雷达或载机雷达提供目标信息，因此本文假设导弹在发射前已知所有目标信息且全局共享。但在导弹发射后，需要单纯依靠导弹之间通信进行目标的探测、跟踪和分配，地面雷达或载机不再提供目标新的信息。对目标的探测与跟踪并不是本文的重点，因此本文不再展开，且不考虑目标存在发射干扰弹等逃脱行为，则将该问题简化为只要目标在导弹的探测范围内即可认为导弹实现了对目标的探测和跟踪。

对于导弹之间的通信，本文假设导弹的通信方式为广播式，且广播通信半径有限，即在通信半径内的导弹可实现双向通信。为简化问题，本文中的通信不考虑时延、丢包、噪声等问题，即导弹之间的通信是双向且即时可靠的，因此导弹

之间的通信网络可用一个无向图表示，在该无向图中，用节点表示导弹，用边表示直接通信通道。另外，在实际中，导弹之间即使没有建立起直接通道，也可以通过其他导弹进行通信路由，实现信息的传输，因此本文认为两节点间只要存在路径，这两个节点便可实现信息交互。

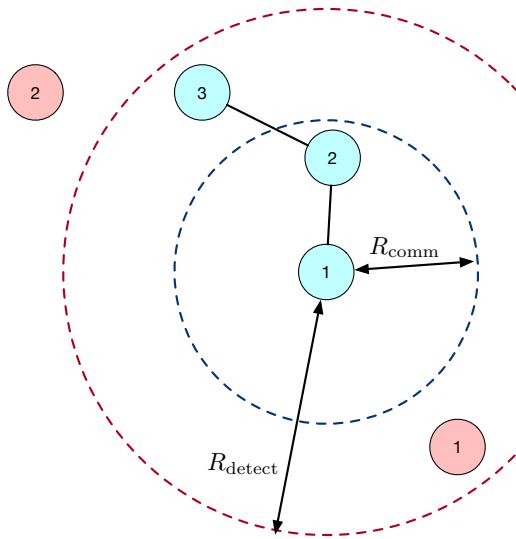


图 2-2 导弹探测与通信示意图

Figure 2-2 Diagram of detection and communication of missiles

用图2-2详细解释本文的假设，图中用蓝色节点代表导弹，红色节点代表目标，红色虚线圈代表导弹的探测范围，蓝色虚线圈代表导弹的通信范围， R_{detect} 和 R_{comm} 分别代表导弹的探测半径和通信半径^①。由于导弹 2 位于导弹 1 的通信范围内，因此导弹 1 和导弹 2 之间建立了通信通道，而导弹 3 超出了导弹 1 的通信范围，因此导弹 1 和 3 之间没有直接的通信通道，但导弹 2 和 3 之间也有一条通道。目标 1 位于导弹 1 的探测范围内，因此导弹 1 可以知晓目标 1 的相关信息，由于信息交互的假设，导弹 2 和 3 均可通过交互得到目标 1 的信息；与此同时，虽然目标 2 不在导弹 1 的探测范围内，但由于目标 2 位于导弹 3 的探测范围内，导弹 1 仍可间接通过导弹 3 获知目标 2 的信息。

2.6 本章小结

本章基于攻击时间最小化和导弹机动消耗最小化的原则建立了空空导弹任务分配模型。首先建立了弹目相对运动模型，对导弹和目标的运动方程进行建模。接

^① 为方便展示在图中将导弹和目标均用圆圈表示，在本文实际建立的模型中将导弹和目标均看作质点，没有大小和半径

着基于该相对运动模型估计剩余攻击时间，并设计导弹遵循最优制导律飞行已获得最小机动消耗。接着，本章定义了全局效用函数和任务效用函数的概念，建立了导弹与目标的静态任务分配模型；最后，本文引入了导弹对目标的探测模型和导弹之间的通信模型与假设，建立了动态环境下的任务分配模型，为后续求解空空导弹的任务分配问题打下基础。

第三章 基于势博弈的分布式任务分配方法

3.1 引言

本章使用第二章中建立的空空导弹任务分配模型，设计了面向任务分配问题的势博弈模型（Potential Game, PG）框架。本章首先介绍了势博弈模型概念，并选择合适的函数作为智能体效用，初步建立了势博弈模型；接着针对任务分配中的约束问题，使用 Lagrange 乘子法对原始势博弈模型进行改进，并证明了模型的可行性与性能下界；然后，针对改进的势博弈模型，使用多种均衡求解算法进行模型求解；最后，通过仿真与对比实验验证了模型与算法的有效性。

3.2 势博弈模型

在任务分配问题框架下，当所有导弹依据最大化自身效用的原则选择的目标不再发生变化时，则称所有导弹达到了均衡状态。博弈论中一个经典的均衡状态是纯纳什均衡，它表示一个任务分配解 $a^* = (a_1^*, \dots, a_{n_m}^*)$ ，满足任意一个导弹不能通过独自改变任务提高自身效用。在具体阐述纳什均衡和势博弈概念前，除第二章使用的概念外，仍需要引入以下概念和符号。设 a_{-i} 表示除了导弹 \mathcal{M}_i 以外的其他导弹的分配集合，即

$$a_{-i} = (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_{n_m}),$$

利用该符号可将一组分配解表示为 (a_i, a_{-i}) 。此外，可类似地定义 \mathcal{A}_{-i} 为

$$\mathcal{A}_i := \mathcal{A}_1 \times \dots \times \mathcal{A}_{i-1} \times \mathcal{A}_{i+1} \times \dots \times \mathcal{A}_{n_m}.$$

设 $U_{\mathcal{M}_i}(a)$ 或 $U_{\mathcal{M}_i}(a_i, a_{-i})$ 表示导弹 \mathcal{M}_i 在分配 a 下的智能体效用，由此可得到纯纳什均衡的定义：

定义 3.1 (纯纳什均衡) 若一个分配解 a^* 满足

$$U_{\mathcal{M}_i}(a_i^*, a_{-i}^*) = \max_{a_i \in \mathcal{A}_i} U_{\mathcal{M}_i}(a_i, a_{-i}^*), \quad \forall \mathcal{M}_i \in \mathcal{M}. \quad (3-1)$$

则称 a^* 是一个纯纳什均衡解。

当所有导弹达到纯纳什均衡时，全局效用未必实现最大化；另一方面，在一些博弈场景下纯纳什均衡未必存在。因此，引入势博弈概念

定义 3.2(有序势博弈与势博弈) 设一个博弈模型中的智能体效用函数为 $U_{\mathcal{M}_i}(a), \mathcal{M}_i \in \mathcal{M}$, 如果存在一个全局势函数 $\phi(a) : \mathcal{A} \mapsto \mathbb{R}$, 满足对任意智能体 $\mathcal{M}_i \in \mathcal{M}$, 任意 $a_{-i} \in \mathcal{A}_{-i}$, 智能体 \mathcal{M}_i 的任意两个分配 $a'_i, a''_i \in \mathcal{A}_i$, 有

$$U_{\mathcal{M}_i}(a'_i, a_{-i}) - U_{\mathcal{M}_i}(a''_i, a_{-i}) > 0 \Leftrightarrow \phi(a'_i, a_{-i}) - \phi(a''_i, a_{-i}) > 0, \quad (3-2)$$

则该博弈模型是一个有序势博弈模型。在此基础上, 若该全局势函数进一步满足:

$$U_{\mathcal{M}_i}(a'_i, a_{-i}) - U_{\mathcal{M}_i}(a''_i, a_{-i}) = \phi(a'_i, a_{-i}) - \phi(a''_i, a_{-i}). \quad (3-3)$$

则称该博弈模型为势博弈模型。若策略集 \mathcal{A} 是有限的, 则称该势博弈为有限势博弈。

由势博弈的定义可知, 势博弈模型中的智能体效用变化会等同地反映在全局效用上, 而在有序势博弈上, 智能体效用与全局效用总是同向变化。易知势博弈是一种特殊的有序势博弈。有限势博弈模型有两点性质:

- (1) 有限势博弈至少存在一个纯纳什均衡;
- (2) 有限势博弈具有有限优化性质 (finite improvement property, FIP)。

性质 (1) 保证了纯纳什均衡的存在性, 因此对于任务分配问题, 至少可以得到一个确定的解; 性质 (2) 保证了任意智能体单方面提高自己效用的行为可以在有限时间内收敛到纯纳什均衡, 从而保证了优化算法可在有限时间内收敛。

3.3 面向任务分配问题的势博弈模型设计

3.3.1 WLU 效用函数

使用势博弈模型解决任务分配问题对导弹的智能体效用函数提出的要求是, 满足

$$U_{\mathcal{M}_i}(a'_i, a_{-i}) - U_{\mathcal{M}_i}(a''_i, a_{-i}) = U_g(a'_i, a_{-i}) - U_g(a''_i, a_{-i}), \quad (3-4)$$

使得所有导弹形成一个势博弈模型, 其中 $U_g(a)$ 即为式 (2-32) 中定义的全局效用函数, 因此在求解分配时所有导弹只需考虑最大化智能体效用便可实现全局效用最大化的目标。

一个显然而直接的思路是使用一致利益效用 (Identical Interest Utility, IIU), 其直接将全局效用作为智能体效用, 但 IIU 的缺点在于需要计算所有任务效用后得到全局效用, 因此本质上仍是集中式的方法。对 IIU 进行改进可以得到更分布

化的有限区域效用 (Range-Restricted Utility, RRU) , RRU 将智能体效用定义为所有该智能体参与的任务的效用之和, RRU 可以使得智能体组成势博弈模型, 但该模型的纳什均衡未必是最优的。此外, IIU 和 RRU 有一个共同的问题在于如果存在大量的智能体同时参与大量的任务, 由于 IIU 和 RRU 都是对一定数量的任务效用求和, 此时一个智能体对任务的贡献微乎其微, 即使它单方面改变决策, 对该任务的效用改变可能很小, 从而反映在智能体效用函数上的变化也很小。这对智能体的寻优和学习带来了很大难度。对此问题作出的改进是等份额效用 (Equally Shared Utility, ESU) , ESU 用智能体参与的任务效用除以参与该任务的智能体总数, 可以有效消除智能体数量带来的效用稀释作用, 但 ESU 的局限在于只有当智能体之间不存在区别时才可以组成势博弈模型。

为了克服 IIU 和 RRU 的缺陷, 且不局限于 ESU 的应用场景, 本节采用的是 WLU 效用函数 (Wonderful Life Utility), WLU 将智能体效用函数定义为每个智能体对全局效用的边际贡献程度, 其具体定义为:

$$U_{\mathcal{M}_i}(a_i, a_{-i}) = U_g(a_i, a_{-i}) - U_g(a_0, a_{-i}). \quad (3-5)$$

由全局效用函数的定义式 (2-32) 可将式 (3-5) 改写为

$$U_{\mathcal{M}_i}(a_i, a_{-i}) = U_{\mathcal{T}_j}(a_i, a_{-i}) - U_{\mathcal{T}_j}(a_0, a_{-i}), \quad \text{if } a_i = \mathcal{T}_j. \quad (3-6)$$

由式 (3-6) 可知, WLU 假设其他智能体不改变目标, 智能体效用只与智能体自身的决策有关, 因此 WLU 排除了冗余信息, 比 IIU 和 RRU 优化能力更强。另外, 下面的命题表明, 使用 WLU 的博弈模型一定是势博弈模型。

命题 3.1 (WLU 可行性) 智能体效用函数为式 (3-6) 的智能体集合组成的博弈模型是一个势博弈模型, 且其势函数就是全局效用函数 $U_g(a)$ 。

证明 只需证明 WLU 满足式 (3-4) 即可。设 $a_i, a'_i \in \mathcal{A}_i$, 则有

$$\begin{aligned} & U_{\mathcal{M}_i}(a_i, a_{-i}) - U_{\mathcal{M}_i}(a'_i, a_{-i}) \\ &= U_g(a_i, a_{-i}) - U_g(a_0, a_{-i}) - [U_g(a'_i, a_{-i}) - U_g(a_0, a_{-i})] \\ &= U_g(a_i, a_{-i}) - U_g(a'_i, a_{-i}) \end{aligned} \quad \square$$

3.3.2 约束条件处理方法

上述势博弈框架和智能体效用函数设计下, 对于智能体选择任务并没有做出任何限制和约束, 因此需要利用约束条件式 (2-36) 将势博弈的纳什均衡限制在

可行解中。本节提出了基于 Lagrange 乘子法将有约束问题转化为无约束问题，并证明了在满足一定的条件下，新的无约束问题下获得的纳什均衡解均为原问题的可行解。

Lagrange 乘子法是一种典型的处理含约束优化问题的方法，其将含约束问题转化为无约束问题，且可证明生成的新的无约束问题的最优解就是原问题的最优解。在势博弈模型下，结合全局效用函数的定义式 (2-35) 以及约束条件式 (2-33)，将式 (2-37) 任务效用函数 $U_{\mathcal{T}_i}(a)$ 修正为：

$$\tilde{U}_{\mathcal{T}_j}(a) = U_{\mathcal{T}_j}(a) + \lambda [b_{\max}^{(j)} - s_j]^{-}, \quad (3-7)$$

其中 $[x]^- = \min\{x, 0\}$, s_j 是第二章中定义的目标同为 \mathcal{T}_j 的导弹数量。由式 (2-32) 得到修正后的全局效用函数为

$$\tilde{U}_g(a) = U_g(a) + \lambda \sum_{j=1}^{n_t} [b_{\max}^{(j)} - s_j]^{-} \quad (3-8)$$

再利用式 (3-6) 提出的 WLU 效用函数计算修正后的智能体效用函数 (modified WLU, mWLU) 为

$$\begin{aligned} \tilde{U}_{\mathcal{M}_i}(a_i, a_{-i}) &= \tilde{U}_g(a_i, a_{-i}) - \tilde{U}_g(a_0, a_{-i}) \\ &= \tilde{U}_{\mathcal{T}_j}(a_i, a_{-i}) - \tilde{U}_{\mathcal{T}_j}(a_0, a_{-i}), \quad \text{if } a_i = \mathcal{T}_j. \end{aligned} \quad (3-9)$$

下面需要验证 mWLU 的有效性，主要分为三部分：(1) 满足 mWLU 效用的智能体能否组成势博弈模型；(2) 该势博弈模型下得到的纳什均衡是否都满足原问题的约束；(3) 该势博弈模型下得到的纳什均衡解的最优性。

3.3.2.1 势博弈成立条件

为了验证 mWLU 函数的有效性，首先需要验证式 (3-9) 定义的智能体效用函数是否可以组成一个势博弈模型。设智能体效用函数为 mWLU 的智能体组成的组成的博弈模型为 $\mathcal{G} = \{\mathcal{M}, \mathcal{A}, U_{\mathcal{M}}(a)\}$ ，实际上，可以得到以下结论：

命题 3.2 (势博弈成立条件) \mathcal{G} 是一个势博弈模型，其势函数为式 (3-8) 定义的全局效用函数。

证明 只需证明全局效用函数 $\tilde{U}_g(a)$ 与式 (3-9) 定义的智能体效用函数 $\tilde{U}_{\mathcal{M}_i}(a_i, a_{-i})$ 满足式 (3-4) 即可。对任意 $a'_i, a''_i \in \mathcal{A}_i$, $a' = (a'_i, a_{-i})$, $a'' = (a''_i, a_{-i})$,

且 S', S'' 分别为 a' 和 a'' 对应的智能体分配集合, s'_j 和 s''_j 分别为 S' 和 S'' 中目标为 \mathcal{T}_j 的智能体数量。假设 $a'_i = \mathcal{T}_j$, $a''_i = \mathcal{T}_k$, $j \neq k$, 则有

$$\begin{aligned} & \tilde{U}_{\mathcal{M}_i}(a'_i, a_{-i}) - \tilde{U}_{\mathcal{M}_i}(a''_i, a_{-i}) \\ &= [\tilde{U}_g(a'_i, a_{-i}) - \tilde{U}_g(a_0, a_{-i})] - [\tilde{U}_g(a''_i, a_{-i}) - \tilde{U}_g(a_0, a_{-i})] \\ &= \tilde{U}_g(a'_i, a_{-i}) - \tilde{U}_g(a''_i, a_{-i}) \end{aligned} \quad (3-10)$$

所以满足 mWLU 效用的智能体可以组成一个势博弈模型, 且势函数为全局效用函数 $\tilde{U}_g(a)$ 。 \square

3.3.2.2 纯纳什均衡可行性分析

在验证势博弈模型成立之后, 下一步需要验证该势博弈模型所得到的纯纳什均衡是否是原问题的可行解, 此处有以下结论:

命题 3.3(纳什均衡可行性) 当 λ 满足 $\lambda > \lambda_0$ 时, 若分配策略 a 是 \mathcal{G} 的一个纯纳什均衡, 则 a 一定满足约束条件 (2-33)。其中

$$\lambda_0 = \max_{\substack{\mathcal{M}_i \in \mathcal{M}, \mathcal{T}_j \in \mathcal{T} \\ s_j > b_{\max}^{(j)}}} \left\{ \frac{U_{\mathcal{T}_j}(a_i, a_{-i}) - U_{\mathcal{T}_j}(a'_i, a_{-i})}{s_j - b_{\max}^{(j)}} \right\} \quad (3-11)$$

证明 设 $a^* = (a_k^*, a_{-k}^*)$ 是势博弈模型 \mathcal{G} 的一个纯纳什均衡, 但不满足约束条件 (2-33), 即存在 $1 \leq j \leq n_t$, $s_j^* > b_{\max}^{(j)}$ 。对于参与任务 \mathcal{T}_j 的智能体之一 \mathcal{M}_k , 设 $a_k^* = \mathcal{T}_j$, $a'_k = \mathcal{T}_l$, $\forall l \neq j$, 则有

$$\begin{aligned} & \tilde{U}_{\mathcal{M}_k}(a_k^*, a_{-k}^*) - \tilde{U}_{\mathcal{M}_k}(a'_k, a_{-k}^*) \\ &= \tilde{U}_{\mathcal{T}_j}(a_k^*, a_{-k}^*) - \tilde{U}_{\mathcal{T}_l}(a'_k, a_{-k}^*) \\ &= U_{\mathcal{T}_j}(a_k^*, a_{-k}^*) - \lambda(s_j^* - b_{\max}^{(j)}) - [U_{\mathcal{T}_l}(a'_k, a_{-k}^*) + \lambda[b_{\max}^{(l)} - s_l']^-] \\ &\leq U_{\mathcal{T}_j}(a_k^*, a_{-k}^*) - U_{\mathcal{T}_l}(a'_k, a_{-k}^*) - \lambda(s_j^* - b_{\max}^{(j)}) \\ &\leq \max_{\mathcal{T}_j \in \mathcal{T}} \{U_{\mathcal{T}_j}(a_k^*, a_{-k}^*) - U_{\mathcal{T}_j}(a'_k, a_{-k}^*)\} - \lambda_0(s_j^* - b_{\max}^{(j)}) \end{aligned} \quad (3-12)$$

由 λ_0 的定义易知 $\tilde{U}_{\mathcal{M}_{k_0}}(a_{k_0}^*, a_{-k_0}^*) - \tilde{U}_{\mathcal{M}_{k_0}}(a'_{k_0}, a_{-k_0}^*) \leq 0$, 这与 a^* 是纯纳什均衡的条件 (3-1) 矛盾, 因此 a^* 必然满足约束条件。 \square

3.3.2.3 纳什均衡最优性

在命题3.3下，势博奔模型 \mathcal{G} 的纳什均衡均为原问题的可行解，因此最后需要考察这些纳什均衡解的最优性。本节选择使用无秩序代价（Price of Anarchy, PoA）作为衡量纳什均衡性能的指标。设纯纳什均衡分配解为 a^* ，全局最优分配解为 a^{opt} ，则 PoA 的定义为全局最效用和纳什均衡效用之比的上界：

$$\text{PoA}(\mathcal{G}) = \sup_{a^*} \left\{ \frac{U_g(a^{\text{opt}})}{U_g(a^*)} \right\} \quad (3-13)$$

由 PoA 的定义可知，PoA 越大，代表最差情况下纯纳什均衡效用越小，即意味着该势博奔模型的性能越差。关于本节建立的势博奔模型 \mathcal{G} 的 PoA，我们可以用以下命题得到 PoA 的上界，从而保证了势博奔模型性能的下界。

命题 3.4 势博奔模型 \mathcal{G}_{PG} 的 PoA 上界为

$$\text{PoA}(\mathcal{G}_{\text{PG}}) \leq 1 + \eta_{\text{PG}} \quad (3-14)$$

其中

$$\eta = \max_{\substack{\forall a, a' \in \mathcal{A} \\ \forall \mathcal{M}_i \in \mathcal{M}}} \left\{ \frac{\tilde{U}_g(a') - \tilde{U}_g(a'_i, a_{-i})}{\tilde{U}_g(a)} \right\} \quad (3-15)$$

证明 设 a^{opt} 是全局最优分配解， a^* 是一纯纳什均衡分配解。由纯纳什均衡的定义式 (3-1) 可得

$$\tilde{U}_{\mathcal{M}_i}(a_i^*, a_{-i}^*) \geq \tilde{U}_{\mathcal{M}_i}(a_i^{\text{opt}}, a_{-i}^*), \quad \forall \mathcal{M}_i \in \mathcal{M}. \quad (3-16)$$

将式 (3-4) 代入可得

$$\tilde{U}_g(a_i^*, a_{-i}^*) \geq \tilde{U}_g(a_i^{\text{opt}}, a_{-i}^*). \quad (3-17)$$

不等式右侧可以写为

$$\tilde{U}_g(a_i^{\text{opt}}, a_{-i}^*) = \tilde{U}_g(a_i^{\text{opt}}, a_{-i}^{\text{opt}}) - [\tilde{U}_g(a_i^{\text{opt}}, a_{-i}^{\text{opt}}) - \tilde{U}_g(a_i^{\text{opt}}, a_{-i}^*)], \quad (3-18)$$

所以有

$$\tilde{U}_g(a^*) \geq \tilde{U}_g(a^{\text{opt}}) - [\tilde{U}_g(a^{\text{opt}}) - \tilde{U}_g(a_i^{\text{opt}}, a_{-i}^*)], \quad (3-19)$$

两边同时除以 $\tilde{U}_g(a^*)$,

$$\begin{aligned} \frac{U_g(a^{\text{opt}})}{U_g(a^*)} &\leq 1 + \frac{\tilde{U}_g(a^{\text{opt}}) - \tilde{U}_g(a_i^{\text{opt}}, a_{-i}^*)}{\tilde{U}_g(a^*)} \\ &\leq 1 + \max_{\substack{\forall a, a' \in \mathcal{A} \\ \forall \mathcal{M}_i \in \mathcal{M}}} \left\{ \frac{\tilde{U}_g(a') - \tilde{U}_g(a'_i, a_{-i})}{\tilde{U}_g(a)} \right\} \\ &\triangleq 1 + \eta_{\text{PG}} \end{aligned} \quad (3-20)$$

因此

$$\text{PoA}(\mathcal{G}_{\text{PG}}) \leq 1 + \eta_{\text{PG}} \quad (3-21)$$

□

由命题可知，一旦模型的效用函数确定，则模型纳什均衡性能的下界也随之确定。且为了使得设计的模型表现更好，应该通过设计全局效用函数尽量减小 η_{PG} 的值。

3.4 面向任务分配问题的势博弈决策算法

3.4.1 博弈学习算法

3.4.1.1 虚拟博弈算法

虚拟博弈算法 (Fictitious Play, FP) 是一种经典的学习算法，最初被用来求解零和博弈中的纳什均衡，但也被提出可以用于多智能体系统的学习算法。在本节，虚拟博弈算法被视为智能体选择目标的机制。在每个时刻 k ，智能体 i 计算自身的经验频率分布 $q_i(k)$ ，并根据该分布选择下一步最优决策。智能体经验频率分布定义为

$$q_i^j(k) = \frac{1}{k} \sum_{t=0}^{k-1} I\{a_i = A_i^j\}, \quad A_i^j \in \mathcal{A}_i, j = 1, \dots, |\mathcal{A}_i|, \quad (3-22)$$

其中 $I\{\cdot\}$ 为指示函数， $|\mathcal{A}_i|$ 表示目标集 \mathcal{A}_i 的势。

智能体 i 在 FP 下选择目标的策略是假设其他智能体会独立地根据各自的经验频率分布随机选择一个目标。在此假设下，可计算出智能体 i 选择目标 A_i^j 的期望效用为

$$\begin{aligned} U_i(A_i^j, q_{-i}(k)) &= \mathbb{E}_{a_{-i}}[U_{\mathcal{M}_i}(A_i^j, a_{-i})] \\ &= \sum_{a_{-i} \in \mathcal{A}_{-i}} U_{\mathcal{M}_i}(A_i^j, a_{-i}) \prod_{a_j \in \mathcal{A}_{-i}} q_j^{a_j}(k), \end{aligned} \quad (3-23)$$

其中 $q_{-i}(k) := \{q_1(k), \dots, q_{i-1}(k), q_{i+1}(k), \dots, q_{n_m}(k)\}$ 。因此，智能体 i 在 k 时刻选择的最优目标 $a_i(k)$ 为

$$a_i(k) \in \arg \max_{a \in \mathcal{A}_i} U_i(a, q_{-i}(k)), \quad (3-24)$$

若同时存在多个最优目标，则在多个最优目标中任选一个。

FP 的优点在于除了极少数特殊情况，根据经验频率分布选出的目标分配解几乎都会收敛到纳什均衡。但 FP 的缺点也是显然的，每个智能体在作出决策时需要知道所有智能体的经验频率分布，且计算出所有组合情况下的效用的期望，因此 FP 的计算时间和计算复杂度会随着智能体数量的增加而急剧增加。

为了提高 FP 的可行性，联合策略虚拟博弈 (Joint Strategy Fictitious Play, JSFP) 在 FP 的基础上做出了改进。JSFP 最主要的改进在于在计算不同目标选择的效用时，取消了其他智能体选择目标的独立性条件，从而在计算经验频率分布时，计算的是一个分配组合 a 的经验频率而不是某个特定目标的频率。在 JSFP 下，联合经验频率 $z(\hat{a}, k)$ 的计算公式为

$$z(\hat{a}, k) = \frac{1}{k} \sum_{t=0}^{k-1} I\{\hat{a}(k) = \hat{a}\}, \quad \hat{a} \in \mathcal{A}, \quad (3-25)$$

类似地可定义出 $z_{-i}(\bar{a}, k)$ 的定义为

$$z_{-i}(\hat{a}_{-i}, k) = \frac{1}{k} \sum_{t=0}^{k-1} I\{\hat{a}_{-i}(k) = \hat{a}_{-i}\}, \quad \hat{a}_{-i} \in \mathcal{A}. \quad (3-26)$$

因此此时智能体 i 选择目标 \hat{a}_i 的期望效用是

$$U_i(\hat{a}_i, k) = \sum_{a_{-i} \in \mathcal{A}_{-i}} U_{\mathcal{M}_i}(\hat{a}_i, a_{-i}) z_{-i}(\hat{a}_{-i}, k) \quad (3-27)$$

JSFP 第二个改进的地方在于，虽然式 (3-27) 似乎仍需要计算所有组合策略情况下的期望，但通过将式 (3-26) 代入式 (3-27) 可得

$$U_i(\hat{a}_i, z_{-i}(k)) = \frac{1}{k} \sum_{t=0}^{k-1} U_{\mathcal{M}_i}(\hat{a}_i, a_{-i}(k)). \quad (3-28)$$

上式表示在前 $k - 1$ 次迭代智能体 i 如果选择目标 \hat{a}_i , 而其他智能体选择不变的情况下, 智能体 i 可获得的平均效用。令 $\bar{U}_i^{\hat{a}_i}(k) = U_i(\hat{a}_i, z_{-i}(k))$, 则式 (3-28) 可改写为迭代形式:

$$\bar{U}_i^{\hat{a}_i}(k) = \frac{k-1}{k} \bar{U}_i^{\hat{a}_i}(k-1) + \frac{1}{k} U_{\mathcal{M}_i}(\hat{a}_i, a_{-i}(k)). \quad (3-29)$$

JSFP 的算法流程如 3-1 所示。

算法 3-1 JSFP 算法流程

```

Input:  $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$ 
Output: 均衡解  $a^*$ 
// 初始化参数
1 随机初始化分配解  $a$ ;
2  $\bar{U}_i(0) \leftarrow \mathbf{0}_{1 \times n_i}, i = 1, \dots, n_t$ ;
3  $k \leftarrow 1$ ;
4 while true do
5   for  $i = 1 : n_m$  do
6     计算选择不同目标  $\bar{a}_i$  的  $U_{\mathcal{M}_i}(\hat{a}_i, a_{-i}(k))$ ;
7     使用式 (3-29) 更新  $\bar{U}_i(k)$ ;
8      $a(k) \leftarrow \arg \max_{a \in \mathcal{A}_i} \bar{U}_i(k)$ ;
9      $k \leftarrow k + 1$ ;
10  end
11 end

```

3.4.1.2 后悔匹配算法

后悔匹配算法 (Regret Matching, RM) 是借鉴了 JSFP 的思想, 与 JSFP 计算前 $k - 1$ 个时刻选择目标的平均效用不同, RM 计算的是前 $k - 1$ 次迭代不选择该目标将会损失的效用。沿用前面的符号, 在第 k 次迭代, 智能体 i 计算器对于目标 \bar{a}_i^j 平均后悔值为

$$R_{\mathcal{M}_i}^j(k) = \frac{1}{k-1} \sum_{t=1}^{k-1} [U_{\mathcal{M}_i}(\bar{a}_i^j, a_{-i}(t)) - U_{\mathcal{M}_i}(a(t))] , j = 1, \dots, |\mathcal{A}_i|. \quad (3-30)$$

式 (3-30) 可改写为迭代形式

$$R_{\mathcal{M}_i}^j(k+1) = \frac{k-1}{k} R_{\mathcal{M}_i}^j(k) + \frac{1}{k} [U_{\mathcal{M}_i}(\bar{a}_i^j, a_{-i}(t)) - U_{\mathcal{M}_i}(a(t))], \quad k > 1. \quad (3-31)$$

在得到对每个目标的后悔值 $R_{\mathcal{M}_i}(k) = [R_{\mathcal{M}_i}^1(k), \dots, R_{\mathcal{M}_i}^{|\mathcal{A}_i|}(k)]$ 后，智能体计算当前选择目标的概率分布 $p_i(k)$

$$p_i(k) = \frac{[R_{\mathcal{M}_i}(k)]^+}{\mathbf{1}^T[R_{\mathcal{M}_i}(k)]^+}, \quad (3-32)$$

其中 $[x]^+ = \max\{x, 0\}$, $\mathbf{1} = [1, 1, \dots, 1]$ 。

在 RM 的基础上进一步改进得到带有记忆和惰性的广义 RM 算法 (Generalized RM with Fading Memory and Inertia, GRMFMI)。将式 (3-31) 改写为

$$\tilde{R}_{\mathcal{M}_i}^j(k+1) = (1 - \rho)\tilde{R}_{\mathcal{M}_i}^j(k) + \rho[U_{\mathcal{M}_i}(\bar{a}_i^j, a_{-i}(t)) - U_{\mathcal{M}_i}(a(t))], \quad j \in \{1, \dots, |\mathcal{A}_i|\}. \quad (3-33)$$

将惰性概念引入智能体 i 选择目标的概率分布，智能体有 $1 - \alpha_i$ 的概率会继续选择前一时刻的目标，此时目标选择概率分布为

$$\tilde{p}_i(k) = \alpha_i P_i(\tilde{R}_{\mathcal{M}_i}(k)) + [1 - \alpha_i] \mathbf{v}^{a_i(k-1)}, \quad (3-34)$$

其中 $\mathbf{v}^{a_i(k-1)}$ 表示第 $a_i(k-1)$ 个元素为 1，其余为 0 的 $|\mathcal{A}_i|$ 维向量。 $P_i(x)$ 是一个概率分布向量，其每个分量的表达式为

$$P_i^l(x) = \begin{cases} \frac{e^{\frac{1}{\tau}x^l}}{\sum_{x^m > 0} e^{\frac{1}{\tau}x^m}} I\{x^l > 0\}, & \text{if } \mathbf{1}^T[x]^+ > 0, \\ \frac{1}{|\mathcal{A}_i|}, & \text{if } \mathbf{1}^T[x]^+ = 0. \end{cases}, \quad (3-35)$$

其中 $\tau > 0$ 是一个参数，当 τ 较小时， \mathcal{M}_i 会倾向于选择最大后悔值的目标，当 τ 较大时， \mathcal{M}_i 会倾向于在有正后悔值的目标中随机选择一个。

综上所述，GRMFMI 的算法流程如算法3-2所示。

3.4.1.3 空间自适应博弈算法

空间自适应博弈算法 (Spatial Adaptive Play, SAP) 原本是空间博弈中的一种自适应学习方法，本节将针对任务分配问题的特点，将其改造为适用于任务分配问题的 SAP 算法。

算法 3-2 GRMFMI 算法流程

```

Input:  $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$ 
Output: 均衡解  $a^*$ 
// 初始化参数
1  $\tilde{R}_{\mathcal{M}_i}^j(0) \leftarrow 0;$ 
2  $k \leftarrow 1;$ 
3 while true do
4   for  $i = 1 : n_m$  do
5     使用式 (3-31) 计算  $\tilde{R}_{\mathcal{M}_i}^j(k)$ ;
6     使用式 (3-34) 计算目标概率分布  $\tilde{p}_i(k)$ ;
7     根据  $\tilde{p}_i(k)$  随机选择目标  $a_i(k)$ ;
8      $k \leftarrow k + 1$ ;
9   end
10 end

```

不同于之前几种算法，SAP 算法的特点是在每次迭代只等可能地随机选择一个智能体进行目标的选择，其他智能体保持目标不变。被选中的智能体 \mathcal{M}_i 根据式计算其目标选择概率分布 $p_i(k)$

$$\max_{p_i(k) \in \Delta(|\mathcal{A}_i|)} p_i^T(k) \begin{bmatrix} U_{\mathcal{M}_i}(\bar{a}_i^{(1)}, a_{-i}(k-1)) \\ \vdots \\ U_{\mathcal{M}_i}(\bar{a}_i^{(|\mathcal{A}_i|)}, a_{-i}(k-1)) \end{bmatrix} + \tau \mathcal{H}(p_i(k)), \quad (3-36)$$

其中 $\mathcal{H}(\mathbf{x}) = -\mathbf{x}^T \log(\mathbf{x})$, $x^l \neq 0, l = 1, \dots, |\mathcal{A}_i|$ 。

根据式 (3-36) 可求得 $p_i(k)$ 的解析解形式为

$$p_i(k) = \sigma \left(\frac{1}{\tau} \begin{bmatrix} U_{\mathcal{M}_i}(\bar{a}_i^{(1)}, a_{-i}(k-1)) \\ \vdots \\ U_{\mathcal{M}_i}(\bar{a}_i^{(|\mathcal{A}_i|)}, a_{-i}(k-1)) \end{bmatrix} \right), \quad (3-37)$$

其中 $\sigma(\cdot)$ 为 softmax 函数。

为了进一步提高 SAP 算法的效率，还可对 SAP 算法进行改进得到部分选择 SAP (Selective SAP, sSAP) 算法。与 SAP 算法不同的是，sSAP 算法在计算 $p_i(k)$ 时，只在 \mathcal{A}_i 中挑选了 n_i 个目标 ($1 \leq n_i < |\mathcal{A}_i|$) 作为下一步可选目标集，其中包括了上一步选择的目标 $a_i(k-1)$ ，因此 sSAP 算法下 $p_i(k)$ 的计算公式为

$$p_i(k) = \sigma\left(\frac{1}{\tau} \begin{bmatrix} U_{\mathcal{M}_i}(a_i(k-1)) \\ U_{\mathcal{M}_i}(\bar{a}_i^{(1)}, a_{-i}(k-1)) \\ \vdots \\ U_{\mathcal{M}_i}(\bar{a}_i^{(n_i)}, a_{-i}(k-1)) \end{bmatrix}\right), \quad (3-38)$$

SAP 算法和 sSAP 算法的流程如算法3-3所示。

算法 3-3 SAP 和 sSAP 算法流程

```

Input:  $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$ 
Output: 均衡解  $a^*$ 
// 初始化参数
1  $k \leftarrow 1;$ 
2 while true do
3   随机选择一个智能体  $\mathcal{M}_i$ ;
4    $n_i = |\mathcal{A}_i|$ ; // 使用 SAP 算法
5   (或者) 确定  $n_i$  的值,  $1 \leq n_i < |\mathcal{A}_i|$ ; // 使用 sSAP 算法
6   从  $\mathcal{A}_i$  中随机选择  $n_i$  个目标 (包括  $a_i(k-1)$ ) ;
7   使用式 (3-37) 计算  $p_i(k)$ ;
8   根据  $p_i(k)$  随机选择目标  $a_i(k)$ ;
9    $k \leftarrow k + 1$ ;
10 end

```

3.4.2 任务交易

3.4.1小节的 SAP 学习算法可以快速求得任务分配问题的一个可行解，且该可行解由式 (3-14) 限定了最优性能下界。但此时得到的仍是一个次优解，为了在次基础上进一步优化，本节设计了任务交易算法，在多个纳什均衡解中寻找更优解。

当智能体及其邻居节点均达到自身的最优状态时，根据势博弈架构的性质，智能体已经无法通过单向的决策提高自身和全局的功用，因此此时需要智能体之间进行双方的协商与交易，在纳什均衡的基础上继续寻找更优的分配方案。

本文提出的智能体双方交易算法的主要思路是：假设参与交易的两个智能体分别为 \mathcal{M}_i 和 \mathcal{M}_j ，且 $a_i = \mathcal{T}_k, a_j = \mathcal{T}_l$ ，其他不参与交易的智能体不改变自己的任务，即当前的任务分配方案为 $a = (a_1, \dots, a_i, \dots, a_j, \dots, a_{n_m})$ ，若交易达成，则两个智能体交换任务，交易后的任务分配方案为 $a' = (a_1, \dots, a'_i, \dots, a'_j, \dots, a_{n_m})$, $a'_i =$

$\mathcal{T}_l, a'_j = \mathcal{T}_k$ 。判断是否交易的条件时，参与交易的两个智能体分别计算如果交换任务之后，自身的效用变化值，

$$\Delta U_{\mathcal{M}_x}(a, a') = U_{\mathcal{M}_x}(a') - U_{\mathcal{M}_x}(a), x \in \{i, j\}. \quad (3-39)$$

如果 \mathcal{M}_i 和 \mathcal{M}_j 的效用变化值之和大于 0，即交易后双方整体效用上升，则交易达成；如果小于 0，则交易不进行；若等于 0，则以一定概率进行交易。

结合3.4.1小节的学习算法，基于势博弈模型的任务分配算法流程如算法3-4所示。算法中的 `UnilateralDecision` 函数实现的是单方决策算法，`GetNeighbors` 函数用于获得当前节点的邻居节点，`JudgeEquilibrium` 函数用于判断当前节点及其所有邻居节点是否达到了均衡状态。

3.4.3 算法复杂度分析

在势博弈模型学习的每次迭代过程中，首先需要进行信息的交互。智能体之间互相需要交流的信息包括每个智能体选择的目标 a_i ，每个智能体执行该任务所获得的回报 r_{ij} 和成本 c_{ij} 。智能体信息的交互方式不是本文的研究重点，结合本文在第二章中建立的导弹通信模型，可以认为智能体间完成所有信息交互的时间复杂度为 $O(d)$ ，其中 d 为集群通信网络的直径。

完成信息的一致后，需要计算智能体效用函数和任务效用函数，同时这也是最消耗资源与时间的一步。本章使用 WLU 效用函数作为智能体效用函数，意味着每次智能体效用的计算需要调用两次任务效用函数的计算。针对 JSFP 和 GRMFMI 算法，每个智能体需要对可选择的所有目标计算一次智能体效用函数，从而 JSFP 和 GRMFMI 算法在一次迭代中每个智能体的时间复杂度均为 $O(2|\mathcal{A}_i|)$ 。SAP 算法在每次迭代中只有一个智能体被选中进行决策，其迭代的时间复杂度仍然是 $O(2|\mathcal{A}_i|)$ ，但全局的通信负载和计算资源消耗只是 JSFP 和 GRMFMI 算法的 $1/n_m$ 。

3.5 仿真分析与对比

根据第二章建立的任务分配模型，随机生成导弹与目标参数，使用本章提出的势博弈模型，并分别使用 JSFP、GRMFMI、SAP 的决策算法对该模型进行求解。为了对比效果，使用几种集中式启发算法进行对比，使用到的启发式算法包括遗传算法（GA）、粒子群算法（PSO）、退火遗传算法（SAGA）、蚁群退火遗传算法（ACOSAGA）。这四种启发式算法可分为三类，其中 GA 和 PSO 是经典的启发式算法，效果良好，SAGA 是结合了退火和遗传两种算法的混合启发式算法，ACOSAGA 算法是使用蚁群算法（ACO）为 SAGA 算法寻找最优启发式算子的超

算法 3-4 基于势博弈模型的任务分配算法

Input: $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$

Output: 最优解 a^*

// 初始化

1 $k \leftarrow 1;$

2 $satisfied \leftarrow \mathbf{0}_{1 \times n_m};$
// 每个智能体的决策过程
// 循环终止条件为所有智能体达到满意状态

3 **while** $\neg satisfied$ **do**

4 $a_i^* = \text{UnilateralDecision}(a_i, a_{-i}))$; // 智能体进行单方决策

5 $satisfied(i) \leftarrow 1$; // 智能体 \mathcal{M}_i 进行最优决策后满意值置为 1

6 neighbors = GetNeighbors(i); // 获取节点邻居
// 判断节点及其邻居是否均达到均衡，若均达到均衡，则返回 flag 为 1

7 flag = JudgeEquilibrium($\{i\} \cup \text{neighbors}$);

8 **if** $flag$ **then**
 // 计算交易前后效用净变化值，选择净变化为正且最大的邻居

9 $\Delta U \leftarrow \max_{\mathcal{M}_k \in \text{neighbors}} \{\Delta U_{\mathcal{M}_i}(a, a') + \Delta U_{\mathcal{M}_k}(a, a')\};$

10 trader $\leftarrow \arg \max_{\mathcal{M}_k \in \text{neighbors}} \{\Delta U_{\mathcal{M}_i}(a, a') + \Delta U_{\mathcal{M}_k}(a, a')\};$

11 **if** $\Delta U < 0$ **then**
 TraderFlag $\leftarrow 1$; // 如果交易使得效用上升，则交换任务

12 **else if** $\Delta U == 0$ **then**
 TraderFlag $\leftarrow \text{rand}\{0, 1\}$; // 如果交易使得效用不变，则随机选择是否交

13 换任务

14 **end**

15 **end**

16 **if** $TraderFlag == 1$ **then**
 tmp $\leftarrow a_{\text{trader}}^*$;

17 $a_{\text{trader}}^* \leftarrow a_i$;

18 $a_i^* \leftarrow \text{tmp}$;

19 // 将交易双方的满意值重置，重新进行单向决策

20 $satisfied(i) \leftarrow 0$;

21 $satisfied(\text{trader}) \leftarrow 0$;

22 **end**

23 **end**

24 $k \leftarrow k + 1$;

25 **end**

启发式算法。使用启发式算法求解任务分配，不需要建立势博弈模型，只需得到全局目标函数即可。将 GA 和 SAGA 算法的参数设置为种群数量为 20，迭代次数为 200 步，交叉算子为 PMX 算子，交叉概率为 0.4，变异算子为 EM 算子，变异概率为 0.4。ACOSAGA 算法使用蚁群算法对 SAGA 算法中使用的交叉和变异算子进行寻优，其中可选交叉算子为 PMX、OX1、OX2、CX、PBX，可选变异算子为 EM、IM、DM、SIM、SM、IVM、LM。

由于不论是本章使用的势博弈模型下的三种算法，还是其他启发式算法都面临着单次优化所得解不唯一，且收敛解与初始解有关的情况。因此本节将七种算法针对同一场景的任务分配问题求解 100 次，每次求解的初始解均为随机产生，比较各算法下优化性能的稳定性。此外，随着问题规模的增加，得到全局最优解的难度迅速增加，且目标函数值，即全局效用函数值也会随之增加。为了比较各算法的有效性，且去除全局效用幅值的影响，本文选择取七种算法针对同一场景下独立运行 100 次计算出的最优值作为参照值，将各算法所得目标值与参照值作比，作为各算法的优化程度表现。下面将使用的“ A 对 B ”表示 A 枚导弹对战 B 个目标的场景。

任务分配中的场景通常可以分为两种，分别为平衡指派场景和非平衡指派场景。平衡指派场景指的是 n 枚导弹攻击 n 个目标，即导弹数量等于目标数量；非平衡指派问题指的是导弹数量不等于目标数量，在本文中只研究导弹数大于目标数的情况。

本节将在平衡指派问题下研究任务分配问题规模对算法性能的影响。由于平衡指派问题排除了目标之间不同分配数量的差异，因此可以较好地反映出问题规模对算法的影响。而对于非平衡指派问题，本节将研究集中式算法与分布式算法对非平衡指派问题的求解性能。在使用集中式算法解决非平衡问题时，一个经典的处理方法是先增加虚拟目标将非平衡问题转换成平衡问题后再解决。因此本质上对于集中式算法来说，不存在所谓的非平衡问题。但对于本章提出的基于势博弈模型的任务分配方法，由于没有加入虚拟目标，也没有对导弹的可选目标加以强制约束，因此在处理非平衡问题和平衡问题时的性能可能有所差别。本节将针对任务分配中的这两种场景进行仿真实验，对比本章介绍的算法与集中式算法的性能。

3.5.1 场景规模仿真分析

首先针对平衡指派场景进行分析，设置导弹或目标数量从 10 增加到 50，在这五种场景下使用本章介绍的三种算法在内的七种算法，独立重复进行 100 次优

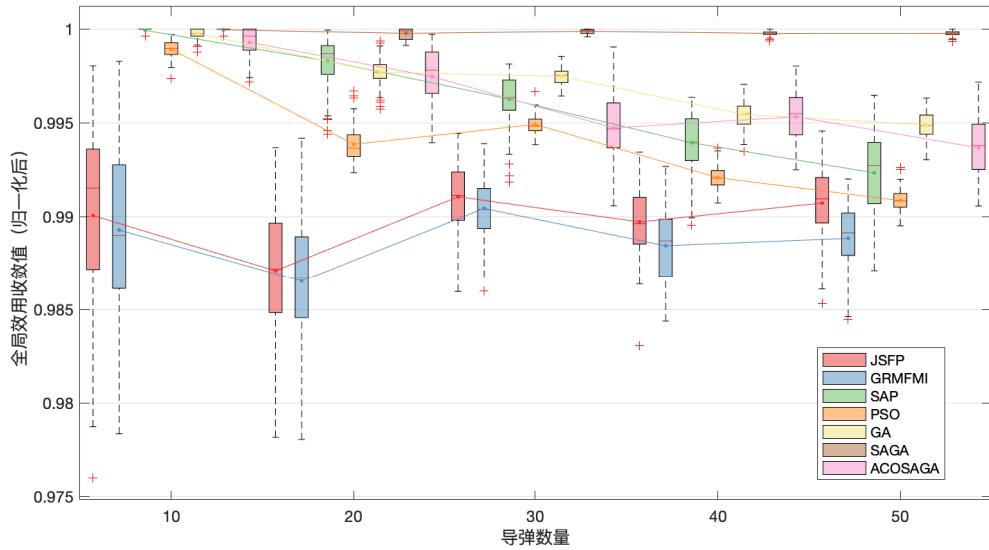


图 3-1 平衡指派场景下算法优化效果对比

Figure 3-1 Comparison of performance in balance scenarios

表 3-1 平衡指派场景下算法性能对比表

Table 3-1 Comparison of algorithms' performance in balance scenarios

场景	JSFP	GRMFMI	SAP	PSO	GA	SAGA	ACOSAGA	
10vs10	平均值	0.9901	0.9893	0.9999	0.9989	0.9998	1	0.9993
	最大值	0.9980	0.9983	1	0.9997	1	1	1
	最小值	0.9760	0.9783	0.9996	0.9974	0.9988	0.9996	0.9972
20vs20	平均值	0.9871	0.9856	0.9983	0.9939	0.9977	0.9998	0.9975
	最大值	0.9937	0.9942	0.9999	0.9967	0.9994	1	0.9997
	最小值	0.9782	0.9780	0.9944	0.9923	0.9957	0.9991	0.9939
30vs30	平均值	0.9911	0.9904	0.9963	0.9949	0.9975	0.9999	0.9947
	最大值	0.9944	0.9939	0.9981	0.9966	0.9986	1	0.9990
	最小值	0.9860	0.9860	0.9919	0.9938	0.9964	0.9996	0.9906
40vs40	平均值	0.9897	0.9884	0.9939	0.9921	0.9954	0.9998	0.9953
	最大值	0.9934	0.9927	0.9964	0.9936	0.9970	1	0.9980
	最小值	0.9831	0.9844	0.9896	0.9907	0.9935	0.9994	0.9925
50vs50	平均值	0.9907	0.9888	0.9923	0.9909	0.9949	0.9998	0.9937
	最大值	0.9946	0.9920	0.9965	0.9926	0.9963	1	0.9972
	最小值	0.9853	0.9845	0.9871	0.9895	0.9930	0.9993	0.9906

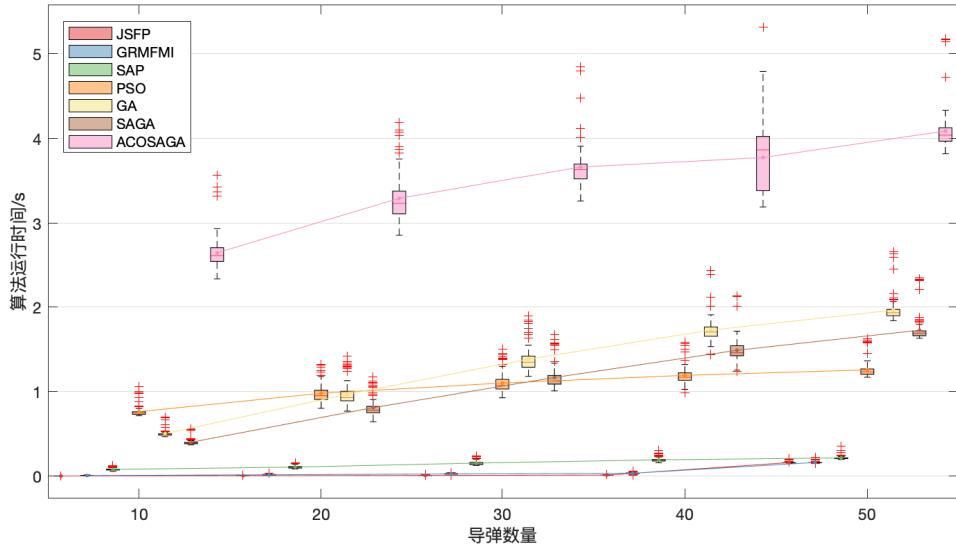


图 3-2 平衡指派场景下算法运行时间对比

Figure 3-2 Comparison of running time in balance scenarios

表 3-2 平衡指派场景下算法运行时间对比表 (单位: s)

Table 3-2 Table of comparison of algorithms' running time in balance scenarios (unit: s)

场景	JSFP	GRMFMI	SAP	PSO	GA	SAGA	ACOSAGA
10vs10	平均值	0.0020	0.0070	0.0801	0.7627	0.5033	0.4010
	最大值	0.0031	0.0153	0.1241	1.0614	0.7026	0.5615
	最小值	0.0015	0.0019	0.0565	0.7140	0.4676	0.3706
20vs20	平均值	0.0055	0.0170	0.1073	0.9803	0.9686	0.8107
	最大值	0.0108	0.0343	0.1618	1.3223	1.4168	1.1736
	最小值	0.0036	0.0056	0.0861	0.8024	0.7690	0.6435
30vs30	平均值	0.0092	0.0268	0.1547	1.1018	1.3845	1.1669
	最大值	0.0160	0.0462	0.2434	1.5053	1.8977	1.6706
	最小值	0.0067	0.0087	0.1263	0.9268	1.1805	1.0086
40vs40	平均值	0.0127	0.0339	0.1933	1.1925	1.7278	1.4894
	最大值	0.0194	0.0557	0.3036	1.5871	2.4452	2.1338
	最小值	0.0085	0.0136	0.1580	0.9871	1.4368	1.2369
50vs50	平均值	0.1604	0.1620	0.2154	1.2572	1.9641	1.7263
	最大值	0.2101	0.2267	0.3529	1.6237	2.6694	2.3463
	最小值	0.1540	0.1493	0.1977	1.1702	1.8390	1.6316

化。

本文使用的仿真平台是 Ubuntu16.04 系统下的 Matlab R2018a。图3-1展示的是得到的收敛值分布箱型图及其平均值曲线，具体平均值、最大值和最小值数据见表3-1，表中加粗数据为本章使用的三种算法中的最大值。图3-2展示的是七种算法完成一次优化求解的运行时间的分布箱型图及其平均值曲线，具体平均值、最大值和最小值数据见表3-2。其中由于分布式架构，JSFP 和 GRMFMI 算法的运行时间是按照所有导弹完成一次迭代后的总时间除以导弹数量作为算法一次迭代的运行时间，而 SAP 算法由于在一次迭代中只有一枚导弹参与迭代，因此不需要进行此操作。

从同个场景下的算法之间的对比角度分析，由图3-1和表3-1中可看出，在优化效果方面，在五种场景下，所有算法基本可以找到较优解，最差优化结果也达到了最好结果的 97.5% 以上。具体而言，SAGA 算法效果最好，基本在所有场景下都得到了七种算法的最大值，其次为 GA 算法，SAP 和 ACOSAGA 算法效果相当，接着为 PSO 算法，最后为 JSFP 和 GRMFMI，两者效果相近，即在优化结果方面顺序为：SAGA>GA>ACOSAGA≈SAP>PSO>JSFP≥GRMFMI。在收敛结果分散程度方面，SAGA 算法仍然是表现最优的算法，其重复实验的结果十分集中，GA 和 PSO 算法其次，SAP 和 ACOSAGA 算法分散程度相当，甚至在导弹数量为 10 和 20 时比 ACOSAGA 更集中，JSFP 和 GRMFMI 仍然是排在最后的算法。因此在收敛结果分散程度方面的排序为：SAGA>GA≈PSO>SAP≈ACOSAGA>GRMFMI≈JSFP，但需要指出的是，考虑到此处结果的幅值范围很小，实际上即使是最分散的情况下，比如 JSFP 算法在 10 个目标的场景下，优化最大值与最小值相差只有 0.022。

从不同场景角度分析，随着任务分配规模的扩大，集中式算法除了 SAGA 外，优化效果都有略微下降。分析本章使用的三种博弈学习算法，可以看出当问题规模较小时，SAP 算法可以达到与集中式算法相当的优化效果，但随着导弹数量的增加，此时由于 SAP 算法机制中每次只有一个智能体进行决策，使得不能完全探索策略空间，因此相对来说性能有所下降；相反地，JSFP 算法和 GRMFMI 算法在导弹数量增加后反而表现有所提高，甚至接近了 SAP 算法，但距离集中式算法的效果还有一定距离。分析认为这是由于 JSFP 算法和 GRMFMI 算法在每次迭代都会让所有智能体选出最优决策，虽然是以一定概率采纳，但在面对较大的策略空间时，相比 SAP 更能搜索到更优解。

由图3-2和表3-2中的时间数据可看出，由于分布式架构的作用，JSFP、GRMFMI 和 SAP 算法的运行时间几乎不受导弹数量的影响，其他六种算法的运行时间随着导弹数量的增加都有所增加，其中 ACOSAGA 算法运行时间最长。由于

分布式架构的优势，JSFP、GRMFMI 和 SAP 算法在五种场景下耗时几乎不变，且均为运行最快的算法。此外，这三种算法运行时间较为集中，而其他算法均或多或少存在耗时较长的特殊运行场景。特别需要指出的是，虽然这里没有将导弹之间的通信一致性过程耗费时间考虑在内。但由于 SAP 算法在每次迭代中只有一个导弹进行决策，因此在通信方面与集中式算法消耗的资源与时间是相当的，在此前前提下 SAP 算法仍具有更好的实时性。

综上所述，本章使用的三种博弈论学习算法均兼具了良好的优化效果，但在优化结果的集中程度方面略差于集中式算法。若将运行时间纳入综合考虑，则本章介绍的基于势博弈模型及其学习算法的任务分配方法在求解规模较大的任务分配问题时具有突出的时间优势，且相比于集中式算法的优化效果减弱程度处于可以接受的范围内。

3.5.2 非平衡指派场景仿真分析

接着，固定导弹数量为 55，目标数量从 10 增加到 50，在不能平分导弹的场景下，会有部分目标所需的导弹数量较大，本节设置的场景是使得目标所需的导弹数量尽量接近^①。仍然使用上述七种算法对这五种场景进行独立重复 100 次优化。正如前文所说，集中式启发算法会事先通过增设虚拟目标的方法，将不平衡指派问题转化为平衡指派问题，而本章使用的基于势博弈模型的分布式算法则不需要进行此操作。下面将分析在不平衡分配问题下本章使用的博弈学习算法的有效性。

图3-3展示的是 100 次优化结果下收敛值的分布箱型图及其平均值曲线，具体平均值、最大值和最小值数据见表3-3。图3-4展示的是七种算法得到最终收敛解的运行时间分布箱型图及其平均值曲线，与3.5.1小节一样，JSFP 和 GRMFMI 的运行时间是由一次求解的总运行时间除以导弹数量得到的平均时间。具体运行时间的平均值、最大值和最小值数据见表3-4。

由图3-3和表3-3数据可看出，在不平衡指派问题中，本章使用的势博弈模型下的三种算法在大多数优化中均能够达到集中式算法最好效果的 98% 以上，其中 JSFP 和 SAP 算法在一些场景下优化效果比 PSO 算法更好，但 GRMFMI 算法相对来说是三种算法中表现较差的。因此七种算法的优化排序为 SAGA>GA>ACOSAGA>PSO≈SAP≈JSFP>GRMFMI。在优化的分散程度方面，势博弈模型下的三种算法相比较集中式算法而言，多次优化的收敛结果较为分散，但从分散的绝对程度来看这样的分散程度是很小的，比如 GRMFMI 算法在五种

^① 第四章的4.5小节中将对非平衡指派场景做进一步深入说明和研究。

场景下优化的最大值和最小值相差最大幅度为 0.013。

此外，从不同场景来看，在不同的非平衡场景中，三种势博弈学习算法的性能随着目标数增加略有下降但下降幅度只在 0.01 范围内，几乎可以认为不变。可见三种学习算法对于非平衡分配问题的求解依然保持着较好的性能。

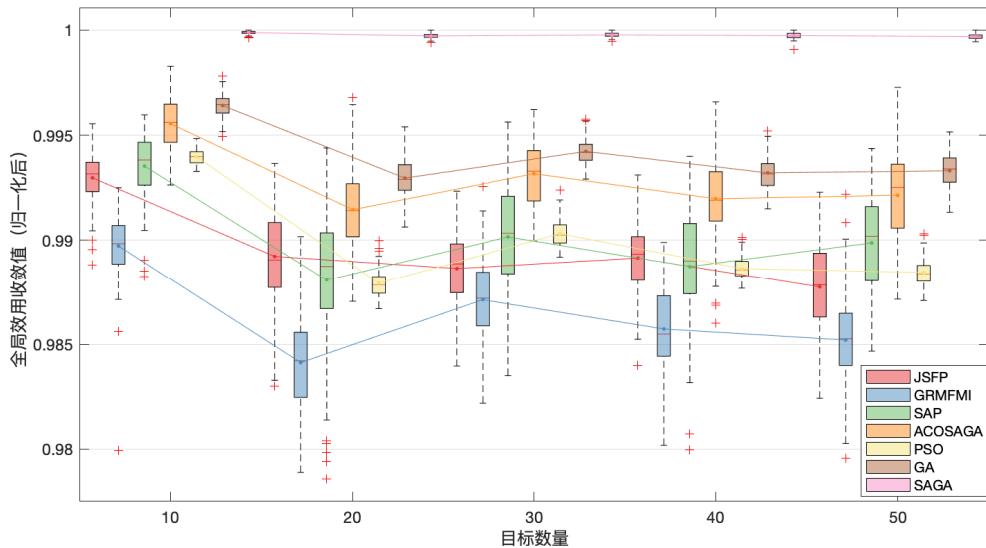


图 3-3 固定导弹数量为 55 非平衡指派场景下算法优化效果对比

Figure 3-3 Comparison of performance in unbalance scenarios with 55 missiles

在运行时间方面，由图3-4和表3-4可知，类似于3.5.1中的分析结果，本章使用的三种分布式算法在实时性方面比集中式算法更具优势，且几乎不会随着问题规模的改变而产生变化。

综上所述，针对非平衡指派问题，本章使用的势博弈模型及三种博弈学习算法可以获得良好而稳定的效果，且相较于集中式算法，不需要事先对非平衡问题进行转换。在实时性方面，三种算法仍然具有分布式架构下的突出的实时性优势。

3.6 本章小结

本章阐述了基于势博弈模型和学习算法的分布式任务分配方法。首先介绍了势博弈模型的概念，并利用 WLU 效用函数建立起势博弈模型。接着针对任务分配问题中的约束条件，引入了分布化的 Lagrange 乘子，对 WLU 效用函数进行修正，从理论上证明了基于修正后的 WLU 函数的智能体效用的可行性以及性能下界。之后本章介绍了三种势博弈模型下的学习算法，并提出了任务交易机制，对

表 3-3 固定导弹数量为 55 非平衡指派场景下算法性能对比表

Table 3-3 Comparison of algorithms' performance in unbalance scenarios with 55 missiles

场景	JSFP	GRMFMI	SAP	PSO	GA	SAGA	ACOSAGA	
55vs10	平均值	0.9930	0.9897	0.9935	0.9940	0.9964	0.9999	0.9956
	最大值	0.9956	0.9925	0.9960	0.9948	0.9978	1	<u>0.9983</u>
	最小值	0.9888	0.9799	0.9883	0.9933	0.9949	0.9996	0.9926
55vs20	平均值	0.9892	0.9841	0.9881	0.9879	0.9930	0.9997	0.9915
	最大值	0.9937	0.9902	0.9944	0.9900	0.9954	1	<u>0.9968</u>
	最小值	0.9830	0.9789	0.9786	0.9867	0.9906	0.9994	0.9871
55vs30	平均值	0.9887	0.9871	0.9902	0.9903	0.9942	0.9998	0.9932
	最大值	0.9923	0.9926	0.9956	0.9924	0.9958	1	0.9962
	最小值	0.9840	0.9822	0.9835	0.9892	0.9929	0.9995	0.9900
55vs40	平均值	0.9892	0.9857	0.9887	0.9886	0.9932	0.9997	0.9920
	最大值	0.9931	0.9899	0.9940	0.9901	0.9952	1	0.9966
	最小值	0.9840	0.9802	0.9800	0.9877	0.9915	0.9991	0.9860
55vs50	平均值	0.9878	0.9852	0.9899	0.9885	0.9933	0.9997	0.9921
	最大值	0.9923	0.9922	0.9944	0.9903	0.9952	1	0.9973
	最小值	0.9824	0.9796	0.9847	0.9871	0.9913	0.9994	0.9872

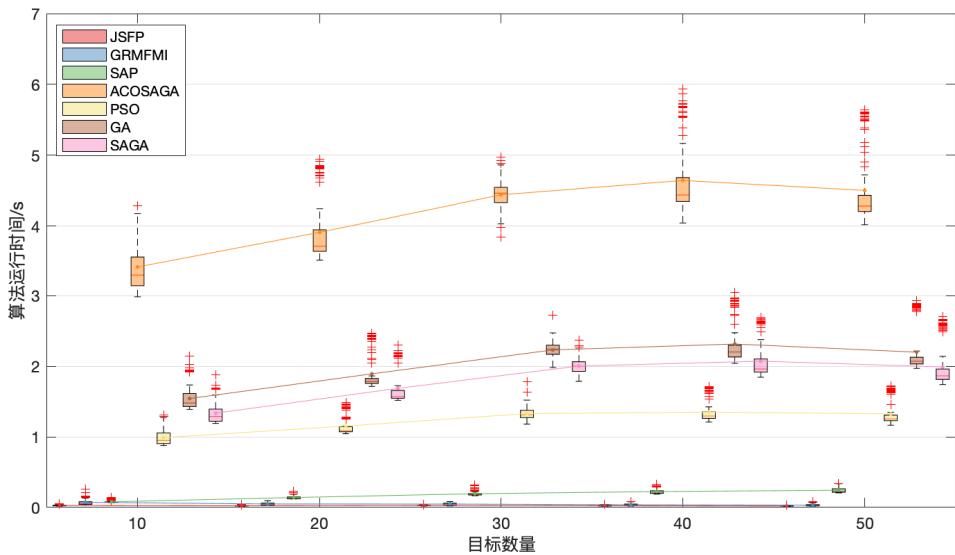


图 3-4 固定导弹数量为 55 的非平衡指派场景下算法运行时间对比

Figure 3-4 Comparison of running time in unbalance scenarios with 55 missiles

表 3-4 固定导弹数量为 55 的非平衡指派场景下算法运行时间对比表 (单位: s)

Table 3-4 Comparison of running time in unbalance scenarios with 55 missiles (unit: s)

场景		JSFP	GRMFMI	SAP	PSO	GA	SAGA	ACOSAGA
55vs10	平均值	0.0286	0.0733	0.0807	0.9909	1.5437	1.3366	3.4217
	最大值	0.0564	0.2566	0.1422	1.3096	2.1459	1.8846	7.4429
	最小值	0.0180	0.0289	0.0664	0.8778	1.3902	1.1920	2.9859
55vs20	平均值	0.0253	0.0499	0.1409	1.1535	1.8938	1.6882	3.9115
	最大值	0.0439	0.0964	0.2298	1.4860	2.4671	2.3054	4.9388
	最小值	0.0166	0.0276	0.1200	1.0448	1.7158	1.5182	3.5176
55vs30	平均值	0.0291	0.0492	0.1950	1.3303	2.2328	2.0036	4.4405
	最大值	0.0396	0.0829	0.3142	1.7790	2.7263	2.3679	4.9753
	最小值	0.0212	0.0223	0.1674	1.1810	1.9835	1.7875	3.8433
55vs40	平均值	0.0250	0.0389	0.2256	1.3524	2.3164	2.0730	4.6430
	最大值	0.0421	0.0838	0.3337	1.7167	3.0509	2.6925	5.9372
	最小值	0.0175	0.0195	0.1890	1.2101	2.0421	1.8454	4.0420
55vs50	平均值	0.0208	0.0323	0.2439	1.3269	2.2009	1.9915	4.5028
	最大值	0.0334	0.0838	0.3408	1.7194	2.9258	2.7018	5.6404
	最小值	0.0152	0.0187	0.2062	1.1657	1.9712	1.7415	4.0176

SAP 算法进行了改进。最后通过仿真验证了势博奕模型在平衡和非平衡任务分配场景下的有效性，及其学习算法优良的实时性。

第四章 基于偏好联盟博弈的分布式任务分配方法

4.1 引言

使用偏好联盟博弈模型 (Hedonic Coalition Game, HCG) 解决任务分配问题的思路是将任务分配问题看作是智能体划分问题，按照任务或目标的不同划分联盟，每个智能体对不同联盟有着不同的偏好，智能体会根据自己的偏好选择自己的联盟，最终所有智能体在确定了自己的所属联盟后，便得到了任务分配的解。本章将建立基于 HCG 的任务分配框架并提出对应求解算法，并通过仿真对比实验验证其有效性。

4.2 偏好联盟博弈模型

在介绍 HCG 模型概念之前，需要先在第二章建立的模型的基础上进行一些修正。沿用第二章中的符号和定义，智能体集合为 $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_{n_m}\}$ ，任务集合为 $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_{n_t}\}$ 。在此章中，不再使用任务效用函数概念，而每个智能体拥有一个智能体效用函数，因此可将智能体效用函数简记为 $U_i(\mathcal{T}_j, p) : \mathcal{T} \times |\mathcal{M}| \mapsto \mathbb{R}$ ，这里的智能体效用函数与第三章中使用 WLU 定义的智能体效用函数不同，除了与当前分配的任务 \mathcal{T}_j 有关外，还与同时选择该任务的智能体个数 p 有关。对于 \mathcal{T}_0 ，定义 $U_i(\mathcal{T}_0, n) = 0, \forall \mathcal{M}_i \in \mathcal{M}, 0 \leq n \leq n_m$ 。设当前分配解为 $a = (a_1, a_2, \dots, a_{n_m})$ ，全局效用函数被定义为

$$U_g(a) = \sum_{\mathcal{M}_i \in \mathcal{M}} U_i(a_i, p_j), \quad (4-1)$$

其中

$$p_j = \sum_{\mathcal{M}_i \in \mathcal{M}} I\{a_i = \mathcal{T}_j\}, \mathcal{T}_j \in \mathcal{T}. \quad (4-2)$$

为了建立 HCG 模型，现引入如下概念。

定义 4.1 (偏好关系) 对于每个智能体 \mathcal{M}_i ，将二元组 $x = (\mathcal{T}_j, p)$ 称为一组联盟对，意味着“智能体 \mathcal{M}_i 将与 p 个队友一起执行任务 \mathcal{T}_j ”，记所有联盟对集合为 $\mathcal{X} = \mathcal{T} \times \{1, \dots, n_m\}$ 。在所有联盟对上定义一个关于效用的偏序关系 \succ_i ，对任意 $x_1, x_2 \in \mathcal{X}$ ， $x_1 \succ_i x_2$ 意味着智能体 \mathcal{M}_i 对联盟对 x_1 有较强的偏好，此外还可定义

$x_1 \sim_i x_2$ 为智能体 \mathcal{M}_i 对 x_1 和 x_2 的偏好无差别, $x_1 \succeq_i x_2$ 意味着智能体 \mathcal{M}_i 对联盟对 x_1 有较弱的偏好。

结合之前定义的智能体效用函数, 偏好关系 \succ_i 可以定义为

$$(\mathcal{T}_1, p_1) \succ_i (\mathcal{T}_2, p_2), \quad \text{if } U_i(\mathcal{T}_1, p_1) > U_i(\mathcal{T}_2, p_2). \quad (4-3)$$

HCG 模型 $\mathcal{G} = (\mathcal{M}, \mathcal{T}, \succeq_i)$ 是由智能体集合, 任务集合和智能体的偏好关系组成的博弈模型。在 HCG 模型框架下解决任务分配问题, 实质上是将任务分配问题看做是对智能体的分组问题, 智能体可根据自己的偏好自行选择加入某个联盟小组, 下面的定义引入了任务分配中智能体联盟的概念。

定义 4.2 (联盟划分) 对于 HCG 模型 \mathcal{G} , 定义一个集合 $\Pi = \{S_0, S_1, \dots, S_{n_l}\}$, 其中元素联盟集合 $S_j \subseteq \mathcal{M}, j = 1, \dots, n_m$ 为同时选择任务 \mathcal{T}_j 的所有智能体集合, 特别地, S_0 代表该智能体没有选择任何任务。由于每个智能体只能选择一个任务, 因此 $\cup_{j=0}^{n_l} S_j = \mathcal{M}, S_i \cap S_j = \emptyset, i \neq j$ 。为了后续论述方便, 用 $\Pi(i)$ 表示智能体 \mathcal{M}_i 选择的任务序号, 用 $S_{\Pi(i)}$ 表示智能体 \mathcal{M}_i 所属的联盟集合, 即 $S_{\Pi(i)} = \{S_j \in \Pi | \mathcal{M}_i \in S_j\}$ 。

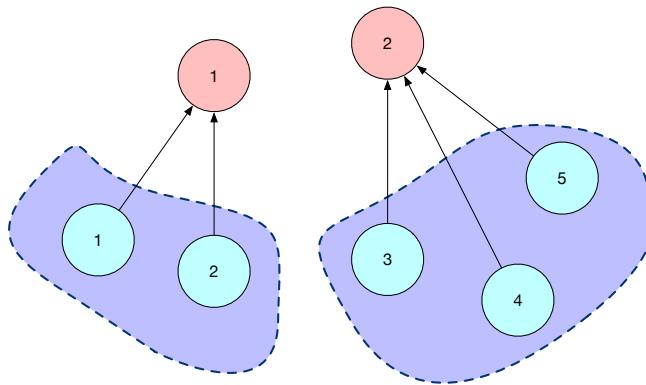


图 4-1 联盟划分

Figure 4-1 Partition of coalition

如图4-1所示, 智能体 1, 2 (蓝色) 同时选择目标 1 (红色), 因此组成了一个联盟, 同理智能体 3, 4, 5 组成了关于目标 2 的联盟。当所有智能体确定自己的联盟后, 即可得到一组任务分配方案。有了联盟划分的概念, 便可引入 HCG 模型下的纳什均衡概念。

定义 4.3 (HCG 模型的纳什均衡) 当在划分 Π 下, 对于任意智能体 $\mathcal{M}_i \in \mathcal{A}$ 都有

$$(t_{\Pi(i)}, |S_{\Pi(i)}|) \succeq_i (t_j, |S_j \cup \{\mathcal{M}_i\}|), \quad \forall S_j \in \Pi, \quad (4-4)$$

则划分 Π 被称为是纳什均衡划分.

定义中 $S_j \cup \{\mathcal{M}_i\}$ 是指智能体离开当前联盟加入新的联盟。因此, 在联盟的概念下, HCG 模型的纳什均衡状态意味着每个智能体均根据自身的效用函数选择自己最“喜爱”的联盟, 在此状态下, 所有智能体不会产生离开当前联盟的动力, 此时也就得到了任务分配的稳定方案。

4.3 面向任务分配的偏好联盟博弈模型设计

针对任务分配问题建立的 HCG 模型需要解决智能体效用函数设计问题。若采用式 (4-3) 定义的偏好关系, 则智能体效用函数直接关系着智能体的偏好关系。因此首先为了保证 HCG 模型纳什均衡状态存在的必然性, 需要对智能体效用函数 $U_i(\mathcal{T}_j, p)$ 做出一定的要求。

定义 4.4 (社交疏远性质) 如果智能体的偏好关系满足对任意任务 $\mathcal{T}_j \in \mathcal{T} \setminus \{\mathcal{T}_0\}$,

$$(\mathcal{T}_j, p_1) \succeq_i (\mathcal{T}_j, p_2), \quad p_1 < p_2, \quad p_1, p_2 \in \{1, \dots, n_m\}, \quad (4-5)$$

即智能体效用随着同联盟的成员数量增加而递减, 则称该智能体具有社交疏远性质 (Social Inhibition Characteristic, SIC)。若使用式 (4-3) 的定义, 则 SIC 表现在智能体效用函数上为

$$U_i(\mathcal{T}_j, p_1) > U_i(\mathcal{T}_j, p_2), \quad p_1 < p_2, \quad p_1, p_2 \in \{1, \dots, n_m\}. \quad (4-6)$$

带有 SIC 的智能体组成的 HCG 模型一定存在纳什均衡。事实上有如下定理。

定理 4.1 设 HCG 模型 $\mathcal{G} = (\mathcal{M}, \mathcal{T}, \succeq_i)$ 中的智能体都具有 SIC, 则该模型一定存在纳什均衡。

证明 可使用数学归纳法证明。当智能体数量 $n_m = 1$ 时, 定理显然成立。

假设 $n_m = k$ 时, 定理成立, 即模型 $\mathcal{G} = (\mathcal{M}, \mathcal{T}, \succeq_i)$ 存在纳什均衡。则当 $n_m = k+1$ 时, 新模型为 $\tilde{\mathcal{G}} = (\tilde{\mathcal{M}}, \mathcal{T}, \succeq_i)$ 。设 $\mathcal{M}_r \in \tilde{\mathcal{M}}, \mathcal{M}_r \notin \mathcal{M}$, 则 $\tilde{\mathcal{M}} = \mathcal{M} \cup \{\mathcal{M}_r\}$ 。

对一个联盟划分 Π , 和任意的智能体 \mathcal{M}_i , 定义联盟可容纳额外成员数 $\Delta_{\Pi(i)}$ 为当前联盟在不会使智能体 \mathcal{M}_i 不选择其他联盟的情况下, 可额外增加的最大智能体个数, 即

$$\Delta_{\Pi(i)} := \min_{S_j \in \Pi \setminus \{S_{\Pi(i)}\}} \max_{\Delta \in \mathbb{Z}} \{\Delta |(\mathcal{T}_{\Pi(i)}, |S_{\Pi(i)}| + \Delta) \succeq_i (\mathcal{T}_j, |S_j \cup \{\mathcal{M}_i\}|)\}. \quad (4-7)$$

由 SIC 定义可知, $\Delta_{\Pi(i)}$ 满足以下性质: (1) 如果划分 Π 是纳什均衡的, 则对于任意智能体 \mathcal{M}_i , 有 $\Delta_{\Pi(i)} \geq 0$; (2) 如果 $\Delta_{\Pi(i)} < 0$, 则智能体 \mathcal{M}_i 会选择离开当前联盟; (3) 智能体 \mathcal{M}_i 在更换联盟后, 设新联盟划分为 Π' , 则有 $\Delta_{\Pi'(i)} \geq 0$ 。

设 Π_0 是博弈 \mathcal{G} 的一个纳什均衡。当智能体 \mathcal{M}_r 选择了一个任务之后, 会形成一个新的联盟划分 Π_1 , 由第 (3) 点性质可知, $\Delta_{\Pi_1(r)} \geq 0$ 。此时若不存在智能体 $\mathcal{M}_q \in \mathcal{A}$, 使得 $\Delta_{\Pi_1(q)} < 0$, 则可知划分 Π_1 是一个纳什均衡。

假设此时至少存在一个智能体 \mathcal{M}_q 满足 $\Delta_{\Pi_1(q)} < 0$, 则该智能体必定在 \mathcal{M}_r 所在的联盟中。由于此时智能体 \mathcal{M}_q 会选择另一个联盟加入, 并再次形成一个新的联盟划分 Π_2 , 因此此时智能体 \mathcal{M}_r 满足 $\Delta_{\Pi_2(r)} \geq 1$ (因为在 \mathcal{M}_q 移动之前已经满足不小于 0)。换句话说, 此时 \mathcal{M}_r 将不会再改变自己的联盟, 即使其他智能体不断的变换自己的联盟。这意味着至多经过 $|\widetilde{\mathcal{M}}|$ 次迭代, 在最终的划分 $\tilde{\Pi}$ 下, 所有的智能体都会满足 $\Delta_{\tilde{\Pi}(i)} \geq 0$, 因此 $\tilde{\Pi}$ 是一个纳什均衡。

由于定理在 $n_m = k + 1$ 时成立, 因此归纳可得原定理成立。 \square

根据以上定理, 结合第二章中的效用函数定义, 定义本章使用的智能体效用函数为

$$U_i(\mathcal{T}_j, p) = \frac{r(\mathcal{T}_j, |S_j|)}{|S_j|} - c_i(\mathcal{T}_j) \quad (4-8)$$

其中 $r(\mathcal{T}_j, |S_j|)$ 为导弹和联盟成员一起攻击目标 \mathcal{T}_j 可获得的回报, 并假设联盟所有成员平分任务回报。在空战场景中, 攻击同一个目标的导弹数量越多, 击中的概率越大, 但当数量超过一定界限时, 再增加导弹数量对于目标的攻击起到的作用很小, 即随着联盟成员的数量增加, 导弹所得到的边际回报在递减。因此可定义任务回报函数 $r(\mathcal{T}_j, |S_j|)$ 为

$$r(\mathcal{T}_j, |S_j|) = r_j^0 \cdot \log_{\varepsilon_j}(|S_j| + \varepsilon_j - 1), \quad (4-9)$$

其中 r_j^0 代表联盟中只有一位成员时的回报值, 本文采用的是式 (2-37) 定义的任务效用函数, $\varepsilon_j > 0$ 是一个正数, 与边际回报的递减速率有关。该函数的图像如图4-2 (a) 所示, 图中 $r_j^{\min} = 10, \varepsilon_j = 3$ 。将任务回报函数代入式 (4-8) 可得智能体效用函数为

$$U_i(\mathcal{T}_j, |S_j|) = \frac{r_j^0 \cdot \log_{\varepsilon_j}(|S_j| + \varepsilon_j - 1)}{|S_j|} - c_i(\mathcal{T}_j), \quad (4-10)$$

图4-2 (b) 所示的是 $r_j^{\min} = 10, \varepsilon_j = 3$ 时的智能体效用函数图像, 由图像可知, 该智能体效用符合 SIC 要求。

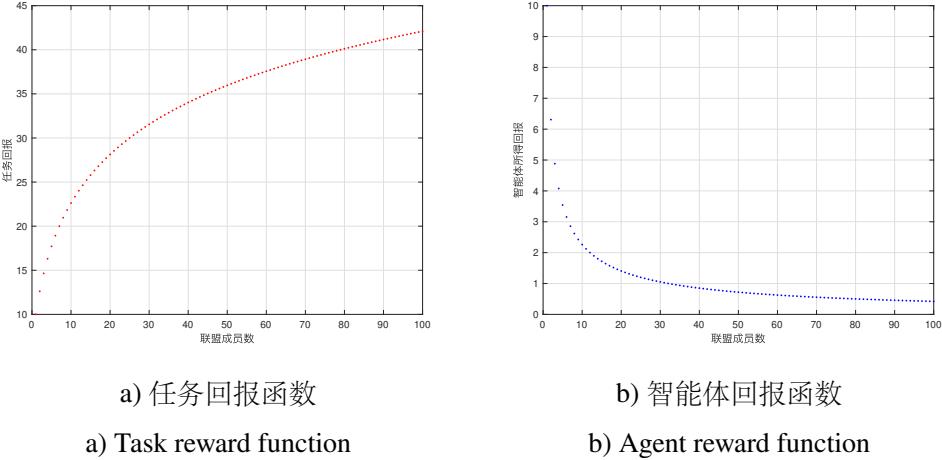


图 4-2 HCG 模型的回报函数

Figure 4-2 Reward function of HCG model

与第三章中遇到的约束条件处理问题类似，在本章使用 HCG 模型的场景中，需要限定得到的联盟划分满足约束条件式（2-33）。由于本节定义的智能体效用函数与联盟成员数有关，而约束条件也是对于执行同一任务的智能体个数的限定，因此可以直接对智能体效用函数进行进一步改进，使得智能体在加入一个联盟时，会考虑到加入该联盟后该联盟成员数是否仍满足约束条件。如果加入该联盟后约束条件不再满足，则智能体不会选择加入该联盟。具体来说，引入改进智能体效用函数 $\tilde{U}_i(\mathcal{T}_j, |S_j|)$ 为

$$\tilde{U}_i(\mathcal{T}_j, |S_j|) = \begin{cases} U_i(\mathcal{T}_j, |S_j|), & \text{if } |S_j| \leq b_{\max}^{(j)}, \\ 0, & \text{otherwise} \end{cases} \quad (4-11)$$

当智能体即将选择加入的联盟已经达到约束边界时，若智能体加入，则得到的效用将会是 0，因此智能体最终不会选择再加入该联盟，从而保证了求解的可行性。

4.4 基于偏好联盟博弈模型的决策算法

4.4.1 SAP 算法

在 HCG 模型下的智能体决策算法仍可使用第三章3.4.1节中使用的几种决策算法，本节选择使用 SAP 算法，下面算法4-1将直接给出使用 SAP 的 HCG 决策算法流程。算法中的 $\text{CoWorkerNums}_j(k)$ 表示 k 时刻智能体 j 的联盟成员数量。为了

适应 HCG 模型的需要，若智能体更换了联盟，则需要将自己更改联盟后新联盟和原有联盟更新后的成员数告诉其他智能体。

算法 4-1 使用 SAP 的 HCG 决策算法流程

```

Input:  $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$ 
Output: 均衡解  $a^*$ 
// 初始化参数
1  $k \leftarrow 1;$ 
2 随机生成任务分配初始解  $a(0);$ 
    // 计算初始每个任务联盟的成员数
3  $\text{CoWorkerNums}_j(0) \leftarrow \sum_{\mathcal{M}_i \in \mathcal{M}} I\{a_i(0) = \mathcal{T}_j\};$ 
4 while true do
5     随机选择一个智能体  $\mathcal{M}_i;$ 
6      $n_i = |\mathcal{A}_i|;$ 
7     for  $j = 1 : n_i$  do
8         | 根据式 (4-10) 计算  $U_i(\mathcal{T}_j, |S_j| + 1);$ 
9     end
10    根据式 (3-37) 计算  $p_i(k);$ 
11    根据  $p_i(k)$  随机选择任务  $\mathcal{T}_l;$ 
        // 更新联盟划分和联盟成员数
12     $a_i(k) \leftarrow \mathcal{T}_l;$ 
13     $\text{CoWorkerNums}_{a_i(k-1)}(k) \leftarrow \text{CoWorkerNums}_{a_i(k-1)}(k) - 1;$ 
14     $\text{CoWorkerNums}_{a_i(k)}(k) \leftarrow \text{CoWorkerNums}_{a_i(k)}(k) + 1;$ 
15     $k \leftarrow k + 1;$ 
16 end

```

4.4.2 分布式一致性算法

本节将提出 HCG 模型下的另一种决策算法，称为分布式一致性算法（Distribute Mutual Consistence Algorithm, DMCA）。MEA 在每次迭代中，所有智能体都会做出自己的决策，但智能体在交互中会只保留一个智能体的决策结果。算法4-2和算法4-3给出了使用 DMCA 决策算法的两个阶段，其中算法4-2是智能体 \mathcal{M}_i 在每次迭代的决策过程，算法4-3是一致性算法的实现。

每个智能体在决策过程中都有自己的划分方式 Π^i ；变量 satisfied 是一个布尔变量，用于表示智能体是否对当前划分 Π^i 满意，即不会离开当前联盟； $r^i \in \mathbb{N}$ 是表示智能体做出了新决策，改变了联盟划分的次数； $s^i \in [0, 1]$ 是一个服从 0 到 1

算法 4-2 MCA 算法中智能体 \mathcal{M}_i 的决策流程

```

Input:  $\mathcal{M}, \mathcal{T}, \mathcal{A}, r$ 
Output: 联盟划分  $\Pi$ 
// 初始化参数
1 satisfied  $\leftarrow false$ ;
2 evolvedi  $\leftarrow 0$ ; // 智能体更新决策次数
3 stampi  $\leftarrow 0$ ; // 时间戳
4  $\Pi^i \leftarrow \{S_0 = \mathcal{M}, S_j = \emptyset, \forall \mathcal{T}_j \in \mathcal{T}\};$ 
5 while true do
    // 每次迭代智能体做出新决策
    6 if satisfied = false then
        7    $(\mathcal{T}_{j^*}, |S_{j^*}|) \leftarrow \arg \max_{S_j \in \Pi^i} U_i(\mathcal{T}_j, |S_j \cup \{\mathcal{M}_i\}|);$ 
        8   if  $(\mathcal{T}_{j^*}, |S_{j^*}|) >_i (\mathcal{T}_{\Pi^i(i)}, |S_{\Pi^i(i)}|)$  then
        9     智能体  $\mathcal{M}_i$  加入  $S_{j^*}$ , 更新划分  $\Pi^i$ ;
        10    evolvedi  $\leftarrow evolved^i + 1$ ;
        11    stampi  $\leftarrow \text{rand}[0, 1];$ 
        12  end
        13  satisfied  $\leftarrow true$ ;
    14 end
    智能体  $\mathcal{M}_i$  向邻居发送信息  $I^i = \{evolved^i, stamp^i, \Pi^i\}$ , 并从其邻居节点获取信
        15 息  $I^k, \forall \mathcal{M}_k \in \mathcal{N}_i$ ;
    16 构造信息集  $\mathcal{I}^i = \{I^i\} \cup \{I^k, \forall \mathcal{M}_k \in \mathcal{N}_i\}$ ;
    // 运行 DMCA 算法
    17 {evolvedi, stampi,  $\Pi^i\}, satisfied \leftarrow DMCA(\mathcal{I}^i);$ 
18 end

```

之间平均分布的随机变量，它会在划分 Π 每次更新时被生成，作用是作为一种时间戳，用于在后续交互时提供一致的依据。智能体在每次迭代中的决策过程是这样的：首先根据当前已知的划分 Π^i ，检查在该划分下，假设其他智能体不改变联盟，找出自己加入哪个联盟会得到最高效用（算法4-2第 7 行）。如果加入该联盟所得效用比自己当前联盟可得到的效用更高，则智能体会选择加入新联盟，同时增价更新次数 r^i ，并随机生成一个时间戳 s^i （算法4-2第 8-11 行）。

由于每个智能体存储的都是自己的划分方式，因此在进行交互协商时，为了达成一致，只能有一种划分方式被广泛接受，称这个划分方式为有效划分。算法4-3) 介绍的 MEA 使得智能体能够在局部通信的情况下获得有效划分。智能体交互时传递的信息集为 $I^i = \{evolved^i, stamp^i, \Pi^i\}$ ，包括自己的划分方式，划分更新次

算法 4-3 分布式一致性算法 (DMCA)

```

Input:  $\mathcal{I}^i$ 
Output: {evolvedi, stampi,  $\Pi^i$ }, satisfied
// 初始化参数
1 satisfied  $\leftarrow$  true;
2 for  $M_k \in \mathcal{M}^i$  do
3   if ( $evolved^k > evolved^i$ ) or ( $evolved^k = evolved^i$  and  $stamp^k > stamp^i$ ) then
4      $evolved^i \leftarrow evolved^k$ ;
5      $stamp^i \leftarrow stamp^k$ ;
6      $\Pi^i \leftarrow \Pi^k$ ;
7     satisfied  $\leftarrow$  false;
8   end
9 end

```

数和更新时间戳。在交互时，更新次数更多的划分方式被认为更加有效，如果更新次数一样，则选择时间戳更大的划分方式（算法4-3第3行）。在确定更有效的划分方式后，智能体将自己的划分方式、更新次数和时间戳统一为有效划分的对应值，同时将自己的满意值置为 false 使得智能体会在下次迭代做出新的决策（算法4-3第4-7行）。

4.4.3 模型与算法性能分析

(1) 收敛性分析

对于 HCG 模型下的任务分配算法收敛性，有如下定理。

定理 4.2 (收敛性) 若 HCG 模型 \mathcal{G} 下的智能体都具有 SIC 性质，则 \mathcal{G} 收敛到一个纳什均衡划分的迭代次数至多为 $|\mathcal{M}| \cdot (|\mathcal{M}| + 1)/2$ 。

证明 证明可以从只包含一个智能体的模型开始，逐一在模型中增加智能体并且找到新模型的纳什均衡划分。由定理 1 的证明过程可知，当一个新的智能体加入一个已得到纳什均衡划分的模型时，至多需要原有智能体个数加一次策略改变，新的模型就可以获得新的纳什均衡划分。因此可得，对于含有 $|\mathcal{M}|$ 个智能体的模型，要获得其纳什均衡划分，至多需要的迭代次数为

$$\sum_{k=1}^{|\mathcal{M}|} k = \frac{|\mathcal{A}| \cdot (|\mathcal{A}| + 1)}{2} \quad (4-12)$$

□

而实际在使用4.4小节中的决策算法的场景下，不必像定理4.2的证明中在等到所有智能体达到纳什均衡后再加入新的智能体，因此实际迭代次数会小于定理中给出的上界。

(2) 复杂度分析

假设算法4-2中在每次迭代过程中智能体进行的主要流程，即第6-17行，为一个迭代步。由于智能体之间的通信架构特点，可能存在某些智能体的迭代步只是执行了发送信息的工作，并没有改变自身的信息（如 Π^i , evolved^i , stamp^i ），将这种迭代称为伪迭代过程，与其他改变了信息的正常迭代区分开来。

注意到在一次正常迭代发生前，伪迭代至多只会发生 d_G 次， d_G 为通信网络的直径。因此根据定理4.2可知，模型收敛到纳什均衡所需的迭代步数为 $O(d_G n_m^2)$ 。特别地，如果通信网络是全连通，则 $d_G = 1$ ，迭代步数变为 $O(n_m^2)$ 。

接着考虑每次迭代过程中的计算复杂度。每个智能体在一次迭代中需要比较包括 \mathcal{T}_0 在内的 $n_t + 1$ 个任务联盟，因此计算复杂度为 $O(n_t)$ ，结合前文所述的迭代步数复杂度，可知算法的总复杂度为 $O(d_G n_t n_m^2)$ 。但注意到定理4.2的结果是保守的，因此实际复杂度会比上述结果更小。

(3) 优化性能分析

关于HCG模型下得到的纳什均衡最优值和全局最优相比较的结果，可类比于命题3.4的证明得到HCG模型 \mathcal{G}_{HCG} 的PoA指标上界为

$$\text{PoA}(\mathcal{G}_{\text{HCG}}) \leq 1 + \eta_{\text{HCG}}, \quad (4-13)$$

其中

$$\eta_{\text{HCG}} = \sum_{S_j \in \Pi} \max_{\mathcal{M}_i \in \mathcal{M}, p \leq |\mathcal{M}|} \left\{ p [U_i(\mathcal{T}_j, p) - U_i(\mathcal{T}_j, |S_j \cup \{\mathcal{M}_i\}|)] \right\} \quad (4-14)$$

下面将针对前文式(4-11)所定义的智能体效用函数这一特殊情况，推导出关于PoA上界的更具体的结论。此处暂时性地引入任务效用函数概念 $U_{\mathcal{T}_j}(\mathcal{T}_j, p)$ ，定义任务效用函数为执行该任务的所有智能体效用之和，即

$$U_{\mathcal{T}}(\mathcal{T}_j, p) = \sum_{\mathcal{M}_i \in S_j} \tilde{U}_i(\mathcal{T}_j, |S_j|). \quad (4-15)$$

结合全局效用函数的定义式(4-1)有

$$U_g(a) = \sum_{\mathcal{M}_j \in \mathcal{M}} \tilde{U}_i(a_i, p_i) = \sum_{\mathcal{T} \in \mathcal{T}} U_{\mathcal{T}_j}(\mathcal{T}_j, p) \quad (4-16)$$

则可得到如下命题。

命题 4.1 设 HCG 模型 \mathcal{G}_{HCG} 中智能体效用函数 $U_i(\mathcal{T}_j, |S_j|)$ 定义为式 (4-11)，且 $\varepsilon_j = \varepsilon > 1$ ，令 $b_{\max} = \max_j \{b_{\max}^{(j)}\}$ ，则 $\text{PoA}(\mathcal{G}_{\text{HCG}})$ 满足

$$\text{PoA}(\mathcal{G}_{\text{HCG}}) \leq 1 + \eta' \quad (4-17)$$

其中

$$\eta' = \log_\varepsilon(b_{\max} + \varepsilon) - 1. \quad (4-18)$$

证明 首先引入一个符号 \oplus 。给定两个划分 $\Pi^A = \{S_0^A, \dots, S_{n_t}^A\}$ 和 $\Pi^B = \{S_0^B, \dots, S_{n_t}^B\}$ ， $\Pi^A \neq \Pi^B$ ，

$$\Pi^A \oplus \Pi^B := \{S_0^A \cup S_0^B, S_1^A \cup S_1^B, \dots, S_{n_t}^A \cup S_{n_t}^B\}, \quad (4-19)$$

由于 $\bigcup_{j=0}^{n_t} S_j^A = \bigcup_{j=0}^{n_t} S_j^B = \mathcal{M}$ ，因此可能存在一个智能体 \mathcal{M}_i 在 $\Pi^A \oplus \Pi^B$ 的两个不同联盟中出现了两次，但在此处我们分析时将这种出现了两次的智能体看做是两个不同的智能体。

由于式 (4-11) 定义的智能体效用函数，使得最终得到的纳什均衡解一定是可行解，因此 $\tilde{U}_i(a_i, p_i) = 0$ 的情况可以暂不考虑。在可行解范围内，任务效用函数 $U_{\mathcal{T}_j}(\mathcal{T}_j, p)$ 满足关于智能体数量 p 单调递增的性质，另外全局效用函数为任务效用函数之和，因此对全局效用函数有

$$U_g(\Pi^A) \leq U_g(\Pi^A \oplus \Pi^B). \quad (4-20)$$

将全局最优划分 Π^{opt} 和纳什均衡划分 Π^* 分别取代上式中的 Π^A 和 Π^B ，则不等式左侧即为全局最高效用 $U_g(\Pi^{\text{opt}})$ ，不等式右侧可写为

$$U_g(\Pi^{\text{opt}} \oplus \Pi^*) = \sum_{\mathcal{T}_j \in \mathcal{T}_{\Pi^*}} U_{\mathcal{T}}\left(\mathcal{T}_j, |S_j^{\text{opt}} \cup S_j^*|\right) + \sum_{\mathcal{T}_k \in \mathcal{T}^-} U_i\left(\mathcal{T}_k, |S_k^{\text{opt}} \cup S_k^*|\right), \quad (4-21)$$

其中 \mathcal{T}^- 为在 Π^* 中满足 $S_j^* = \emptyset, S_j^{\text{opt}} \neq \emptyset$ 的任务 \mathcal{T}_j 集合。但在实际场景中认为除 \mathcal{T}_0 以外各任务均有智能体去执行，因此实际上 $\mathcal{T}^- = \emptyset$ ，所以式 (4-21) 实际上等于

$$\begin{aligned} U_g(\Pi^{\text{opt}} \oplus \Pi^*) &= \sum_{\mathcal{T}_j \in \mathcal{T}_{\Pi^*}} U_{\mathcal{T}}\left(\mathcal{T}_j, |S_j^{\text{opt}} \cup S_j^*|\right) \\ &= \sum_{j=1}^{n_t} U_{\mathcal{T}}\left(\mathcal{T}_j, |S_j^{\text{opt}} \cup S_j^*|\right). \end{aligned} \quad (4-22)$$

代入式 (4-9)，其中总代价函数 $C = 2 \sum_{i=1}^{n_m} \sum_{j=1}^{n_t} c_i(\mathcal{T}_j)$ 是一常数，为简化证明，推导时暂时假设代价值为 0，将上式等号右侧化为

$$\begin{aligned}
 & \sum_{j=1}^{n_t} U_{\mathcal{T}}(\mathcal{T}_j, |S_{\Pi^*(i)}^j \cup S_j^*|) \\
 &= \sum_{j=1}^{n_t} r_j^0 \log_{\varepsilon_j} (|S_j^{\text{opt}} \cup S_j^*| + \varepsilon_j - 1) \\
 &\leq \sum_{j=1}^{n_t} r_j^0 \log_{\varepsilon_j} (|S_j^{\text{opt}}| + |S_j^*| + \varepsilon_j - 1) \\
 &= \sum_{j=1}^{n_t} r_j^0 \left[\frac{\log_{\varepsilon_j} (|S_j^{\text{opt}}| + |S_j^*| + \varepsilon_j - 1)}{\log_{\varepsilon_j} (|S_j^*| + \varepsilon_j - 1)} \cdot \log_{\varepsilon_j} (|S_j^*| + \varepsilon_j - 1) \right]. \tag{4-23}
 \end{aligned}$$

若令 $x = |S_j^*|$, $y = |S_j^{\text{opt}}|$, 并代入 $\varepsilon_j = \varepsilon > 1$, 由于 Π^{opt} 一定满足约束条件, 因此 $|S_j^*| \leq b_{\max}$, 其中 $b_{\max} = \max_j \{b_{\max}^{(j)}\}$

$$\frac{\log_{\varepsilon}(x + y + \varepsilon - 1)}{\log_{\varepsilon}(x + \varepsilon - 1)} \leq \frac{\log_{\varepsilon}(x + b_{\max} + \varepsilon - 1)}{\log_{\varepsilon}(x + \varepsilon - 1)} \tag{4-24}$$

且 $f(x) = \frac{\log_{\varepsilon}(x + a)}{\log_{\varepsilon}(x)}$, $a > 0, \varepsilon > 1$ 在 $[1, \infty)$ 上是递减函数, 因此有

$$\frac{\log_{\varepsilon}(x + b_{\max} + \varepsilon - 1)}{\log_{\varepsilon}(x + \varepsilon - 1)} \leq \frac{\log_{\varepsilon}(b_{\max} + \varepsilon)}{\log_{\varepsilon}(\varepsilon)} = \log_{\varepsilon}(b_{\max} + \varepsilon), \quad 1 \leq x \leq b_{\max} \tag{4-25}$$

因此

$$\begin{aligned}
 U_g(\Pi^{\text{opt}} \oplus \Pi^*) &\leq \sum_{j=1}^{n_t} r_j^0 \left[\log_{\varepsilon}(b_{\max} + \varepsilon) \cdot \log_{\varepsilon} (|S_j^*| + \varepsilon - 1) \right] \\
 &= \log_{\varepsilon}(b_{\max} + \varepsilon) \sum_{j=1}^{n_t} r_j^0 \left[\log_{\varepsilon} (|S_j^*| + \varepsilon - 1) \right] \\
 &= \log_{\varepsilon}(b_{\max} + \varepsilon) U_g(\Pi^*). \tag{4-26}
 \end{aligned}$$

而由式 (4-20) 可得

$$U_g(\Pi^{\text{opt}}) \leq U_g(\Pi^{\text{opt}} \oplus \Pi^*). \tag{4-27}$$

结合式 (4-26) 和式 (4-27) 即可得

$$U_g(\Pi^{\text{opt}}) \leq \log_{\varepsilon}(b_{\max} + \varepsilon) U_g(\Pi^*) \quad (4-28)$$

$$\begin{aligned} \text{PoA}(\mathcal{G}_{\text{HCG}}) &\leq 1 + [\log_{\varepsilon}(b_{\max} + \varepsilon) - 1] \\ &\triangleq 1 + \eta'. \end{aligned} \quad (4-29)$$

□

由上述定理可知, 若要控制 HCG 模型的性能下界 $\text{PoA} < 1 + \eta_0$, 其中 $0 < \eta_0 < 1$, 则需要设置衰减参数 ε 满足

$$b_{\max} \leq \varepsilon^{\eta_0+1} - \varepsilon. \quad (4-30)$$

4.5 仿真分析与对比

本章仿真实验主要研究势博弈模型和 HCG 模型两种博弈模型下的任务分配算法性能。本章沿用第三章的仿真场景, 参与对比的算法是 HCG 模型下的 SAP 算法 (HCGSAP)、HCG 模型下的 DMCA 算法 (HCGDMCA), 与势博弈模型下的 JSFP、GRMFMI 和 SAP 算法 (为了区分 HCGSAP, 本章中是博弈模型下的 SAP 算法用 PGSAP 表示, 如下列图表所示)。为了体现出算法优化效果, 结合第三章的仿真结果, 仍然选取 SAGA 算法结果作为参考值, 以下表格中的数据均经过 SAGA 算法的归一化后结果。

与第三章中的势博弈模型下的任务分配算法类似, 本章使用的 HCG 模型下的任务分配算法也不需要考虑任务分配的平衡问题, 即不需要像集中式算法求解时需要将非平衡指派问题转换为平衡指派问题。但在第三章中没有针对非平衡问题中的非平衡程度对算法性能的影响做进一步研究, 本节将在加入 HCGSAP 和 HCGDMCA 两种算法之后, 对于任务分配问题的非平衡程度对两种模型、五种算法的性能影响进行仿真分析。

本节研究的任务分配问题的非平衡程度主要包含两种。第一种为导弹与目标数量之间的非平衡程度, 为了简化问题, 同时也是为了控制变量, 该种非平衡程度下设置的仿真场景为目标数量一定, 导弹数量为目标数量的不同整数倍; 第二种为目标分配数量之间的非平衡程度, 即导弹与目标数量一定, 不同目标所需分配数量不一致, 本节通过式 (4-18) 中使用的 b_{\max} 值来体现这一非平衡程度。下面将各自介绍仿真结果。

4.5.1 导弹与目标数量非平衡程度影响仿真分析

图4-3展示的是五种算法在固定目标数量为 10，导弹数量由 20 到 60 变化的五种场景下的优化效果对比，其中每个场景下导弹对目标是平均分配。其平均值、最大值与最小值具体数据如表4-1所示，表中加粗数字表示五种算法中最优结果，下划线数字表示五种算法中的第二优结果。与第三章中的分布式算法类似，图4-3中 HCGDMCA 算法运行时间计算的是所有导弹迭代完一次后的总时间除以导弹数量。

从表4-1中可以看出，两种模型下使用 SAP 算法的性能表现最优，在多数场景下都能优化到 0.99 以上。且由图中可知使用 SAP 算法的两种方法的收敛分散程度最小，这表明 SAP 算法在两种模型下都有着较好的优化性能。但值得注意的是，随着导弹数量的增加，两种模型下使用 SAP 算法的都呈现出性能下降的特点，而其他算法则显示出先上升后下降的特点。分析认为这是由于随着导弹数量的增加，SAP 算法在一次迭代过程中只有一枚导弹进行决策，且任务交易机制只有在相邻导弹都达到局部最优时才能进行，因此在一定时间内，SAP 算法可探索到的策略空间占比在减小，从而不能收敛到最优的可能性在增加。

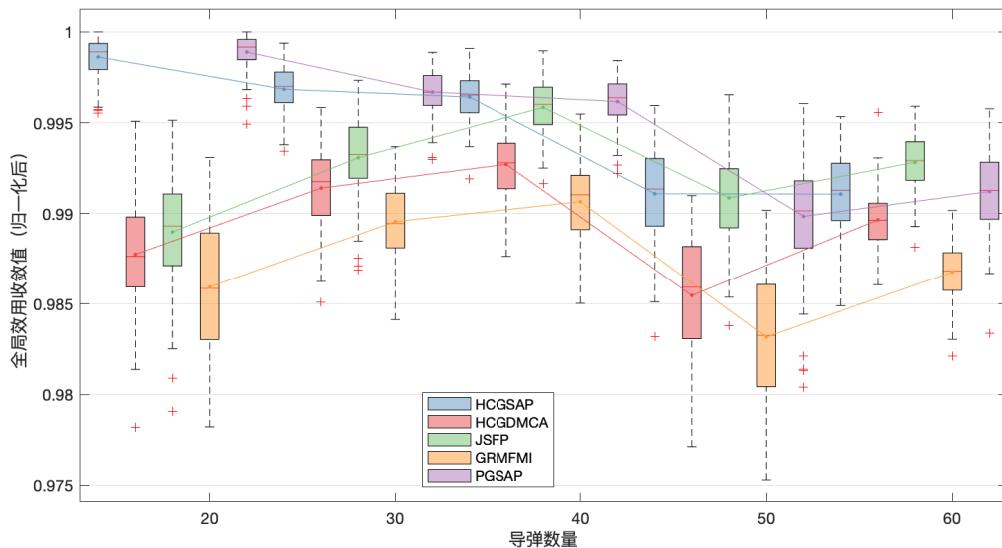


图 4-3 目标数量为 10 的非平衡指派场景下算法优化效果对比

Figure 4-3 Comparison of performance in unbalance scenarios with 10 targets

表 4-1 固定目标数量为 10 的非平衡指派场景下算法性能对比表

Table 4-1 Comparison of algorithms' performance in unbalance scenarios with 10 targets

场景		HCGSAP	HCGDMCA	JSFP	GRMFMI	PGSAP
20vs10	平均值	0.9986	0.9877	0.9890	0.9859	0.9989
	最大值	<u>1</u>	0.9951	0.9951	0.9931	1
	最小值	0.9955	0.9782	0.9791	0.9782	<u>0.9949</u>
30vs10	平均值	0.9968	0.9914	0.9931	0.9896	<u>0.9967</u>
	最大值	0.9994	0.9958	0.9973	0.9937	<u>0.9989</u>
	最小值	0.9935	0.9851	0.9896	0.9841	<u>0.9930</u>
40vs10	平均值	0.9964	0.9927	0.9958	0.9907	<u>0.9962</u>
	最大值	0.9991	0.9971	<u>0.9990</u>	0.9955	0.9984
	最小值	<u>0.9919</u>	0.9876	0.9916	0.9850	0.9922
50vs10	平均值	0.9911	0.9855	<u>0.9909</u>	0.9832	0.9898
	最大值	<u>0.9960</u>	0.9910	0.9937	0.9902	0.9961
	最小值	0.9832	0.9771	<u>0.9830</u>	0.9753	0.9804
60vs10	平均值	0.9911	0.9897	0.9928	0.9868	<u>0.9912</u>
	最大值	0.9953	0.9956	0.9959	0.9902	<u>0.9958</u>
	最小值	0.9849	<u>0.9861</u>	0.9881	0.9821	0.9834

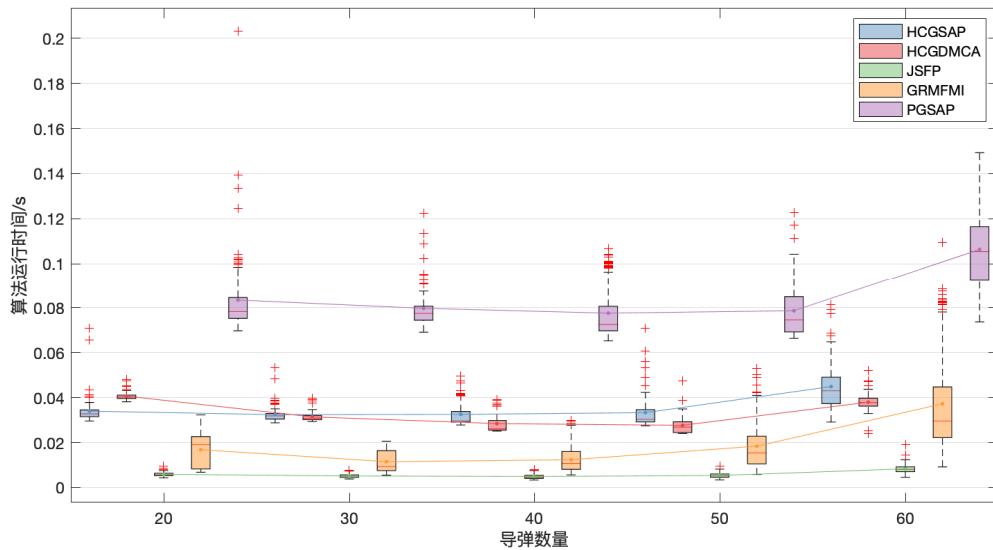


图 4-4 目标数量为 10，不同导弹数量场景下算法运行时间对比

Figure 4-4 Comparison of running time in the scenarios with 20 targets

表 4-2 固定目标数量为 10 的分配场景下算法运行时间对比表 (单位: s)

Table 4-2 Comparison of running time in unbalance scenarios with 10 targets (unit: s)

场景	HCGSAP	HCGDMCA	JSFP	GRMFMI	PGSAP
20vs10	平均值 0.0340	0.0407	0.0058	0.0168	0.0834
	最大值 0.0708	0.0482	0.0095	0.0323	0.2032
	最小值 0.0296	0.0381	0.0043	0.0068	0.0698
30vs10	平均值 0.0324	0.0314	0.0051	0.0115	0.0798
	最大值 0.0533	0.0398	0.0077	0.0206	0.1223
	最小值 0.0288	0.0293	0.0038	0.0055	0.0691
40vs10	平均值 0.0325	0.0284	0.0049	0.0123	0.0776
	最大值 0.0495	0.0391	0.0081	0.0297	0.1065
	最小值 0.0278	0.0251	0.0033	0.0056	0.0653
50vs10	平均值 0.0333	0.0277	0.0053	0.0184	0.0787
	最大值 0.0710	0.0475	0.0096	0.0530	0.1226
	最小值 0.0275	0.0240	0.0034	0.0058	0.0664
60vs10	平均值 0.0449	0.0380	0.0083	0.0372	0.1063
	最大值 0.0812	0.0521	0.0190	0.1094	0.1750
	最小值 0.0291	0.0239	0.0045	0.0092	0.0737

4.5.2 目标之间分配数量非平衡程度影响仿真分析

为了研究目标之间分配数量差异, 即式(4-18)中使用的 b_{\max} 对算法带来的影响, 本节的仿真场景设置为在 20vs10 的场景下对目标取不同的 b_{\max} 值。在 20vs10 的场景下, b_{\max} 的取值范围为 [2, 11], 即一个临界情况是当 $b_{\max} = 2$ 时, 意味着所有目标都被平均分配两枚导弹; 另一个临界情况是当 $b_{\max} = 11$ 时, 意味着只有一个目标可被分配 11 枚导弹, 其余 9 个目标只能被分配一枚导弹。

图4-5展示的是 b_{\max} 从 2 变化到 11 的过程中, 不同算法重复运行 100 次得到的优化结果。需要说明的是, 由于在部分场景下 JSFP 算法最终没有收敛到可行解, 因此图4-5中展示的 JSFP 曲线只统计了其得到可行解的优化效果。除了 JSFP 算法外, 其余四种算法均可收敛到较好结果, 其具体性能数据如表4-3所示。表中加粗数据为表现最好的算法, 下划线为排名第二的算法, 由此可见势博弈模型和 HCG 模型下的 SAP 算法都取得了较好的效果, 而 HCG 模型下的 DMCA 算法和势博弈模型下的 GRMFMI 算法表现在伯仲之间。

从不同场景的表现稳定性来看, 除了 JSFP 算法外, 其余四种算法面对不同

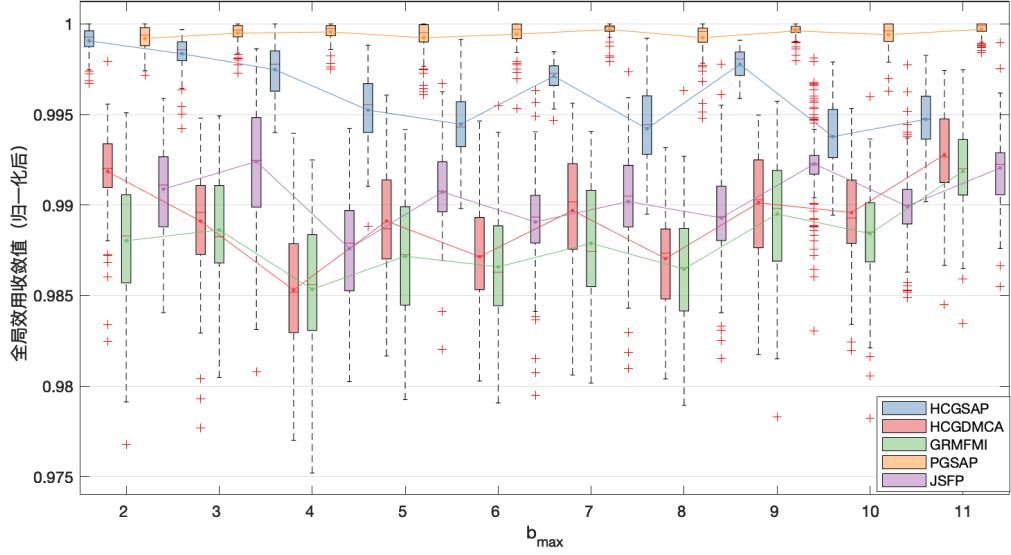
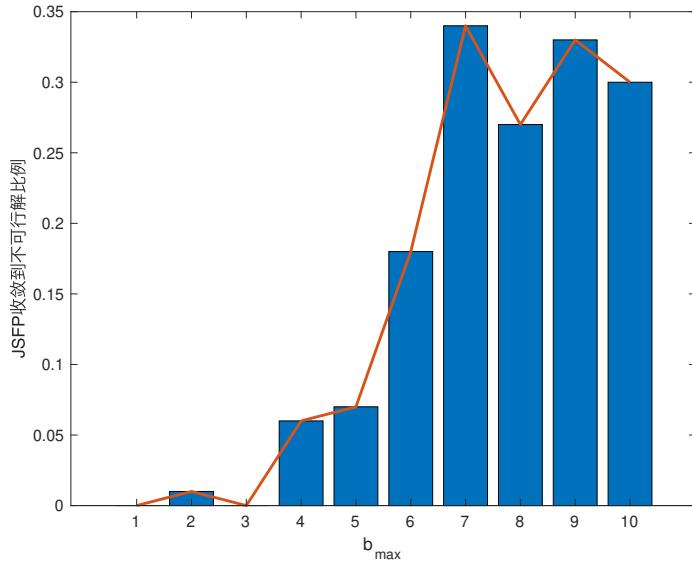
图 4-5 20vs10 场景的不同 b_{\max} 条件下算法优化效果对比Figure 4-5 Comparison of performance in 20vs10 scenarios with different b_{\max} 图 4-6 不同 b_{\max} 条件下 JSFP 算法收敛到不可行解比例变化图

Figure 4-6 Ratio of optimizations in which JSFP converges to an infeasible solution

非平衡程度的场景表现总体稳定。图4-6显示了 JSFP 算法随着 b_{\max} 的增大，收敛到不可行解的次数也在增加，最高可接近 35% 的迭代过程中会收敛到不可行解。总体而言，PGSAP 算法的稳定性最好，其次为 HCGSAP 算法，HCGDMCA 和 GRMFMI 的优化结果的分散程度相对较大，JSFP 算法因为存在收敛到不可行解的情况，因此排名最后。

图4-7展示的是不同 b_{\max} 场景下的算法运行时间对比图。从图中可直接得到的结论是 HCGDMCA 和 GRMFMI 算法运行最快，其次为 HCGSAP 算法，表4-4的数据印证了这一结论。分析 PGSAP 和 HCGSAP 运行时间较长的原因在于本文提出的任务交易机制，在导弹局部达到均衡状态后，由于任务的交易，导致算法没有收敛，因此相较于没有任务交易机制的 HCGDMCA 和 GRMFMI 算法，PGSAP 和 HCGSAP 算法的运行时间更长。

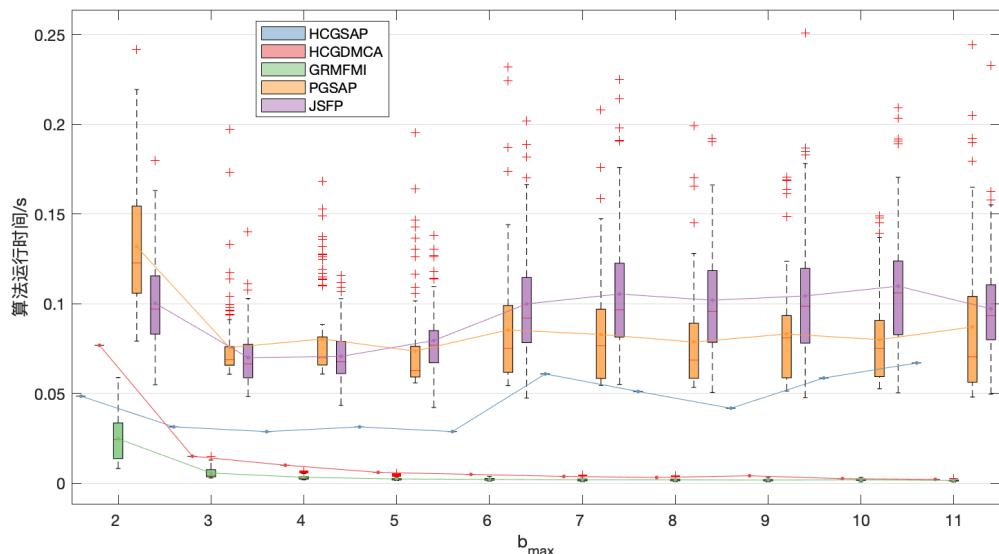


图 4-7 20vs10 场景的不同 b_{\max} 条件下算法运行时间对比

Figure 4-7 Comparison of running time in 20vs10 scenarios with different b_{\max}

从图表可得到的第二个结论是中在 b_{\max} 较小时，随着 b_{\max} 的增大，几种算法的运行时间有下降的趋势，但当 b_{\max} 较大时，算法运行时间基本保持不变。分析原因在于当 b_{\max} 较小时，目标之间分配导弹的数量相差不大，且根据本文模型的设计，导弹在每次决策时有较多目标可以选择，因此需要迭代更多次数才能达到纳什均衡状态；而当 b_{\max} 较大时，目标之间分配导弹的数量差距变大，越来越多的目标只能被一枚导弹选中，此时根据模型机制的设计^①，已有导弹选择的目标

① 主要是由于式3-9和式4-8的设计。

基本不会再被其他导弹选中，因此导弹的可选目标集变小，所有导弹会更快达到纳什均衡状态。但这样的副作用是导致算法过早收敛，陷入局部最优而无法跳出，4-5和4-3也证明了 HCGDMCA 和 GRMFMI 算法的效果相对较差。而 PGSAP 和 HCGSAP 算法由于任务交易机制的存在，使得算法可以避免这种过早收敛。

综上所述，根据两种指派问题非平衡程度的仿真实验结果可知，在面对非平衡指派问题时，本文介绍的两种博弈模型，五种算法总体上表现较好，其中本文提出的改进 SAP 算法可以相对来说可以得到更好的分配解，且在实时性上仍没有失去优势。HCGDMCA 和 GRMFMI 算法虽然优化效果相比两种模型下的 SAP 算法较差，但仍可在大多数场景下得到 97% 以上的优化效果，且实时性优势突出。JSFP 算法虽然优化能力较强，但在分配问题的非平衡程度较大时，会存在收敛到不可行解的情况。

4.6 本章小结

本章将任务分配问题看作联盟形成过程，建立了基于偏好联盟博弈模型 (HCG) 的任务分配算法，并介绍了两种基于该模型的学习算法：改进 SAP 算法和针对 HCG 模型的分布式一致性算法 (DMCA)。在设计智能体效用函数时，假设智能体满足社交疏远性质，并针对约束条件改进智能体效用函数，得到了基于 HCG 的分布式任务分配模型。从理论上证明了该模型的有效性、算法复杂度和最优化。使用第三章中提出的改进 SAP 算法和 DMCA 算法对 HCG 任务分配模型进行学习优化。通过与第三章的势博弈学习算法共同进行仿真实验，从两种非平衡程度设计仿真实验，验证了本文使用的基于两种博弈论模型的分布式任务分配算法在非平衡问题中的有效性和突出的时间优势。

表 4-3 20vs10 场景的不同 b_{\max} 条件下算法性能对比表Table 4-3 Table of comparison of performance in 20vs10 scenarios with different b_{\max}

		HCGSAP	HCGDMCA	JSFP	GRMFMI	PGSAP
$b_{\max} = 2$	平均值	<u>0.9991</u>	0.9918	0.9909	0.9880	0.9992
	最大值	<u>1</u>	0.9979	0.9959	0.9951	<u>1</u>
	最小值	<u>0.9967</u>	0.9825	0.9841	0.9768	0.9972
$b_{\max} = 3$	平均值	<u>0.9984</u>	0.9891	0.9825	0.9886	0.9995
	最大值	<u>0.9997</u>	0.9948	0.9986	0.9949	<u>1</u>
	最小值	<u>0.9942</u>	0.9777	0	0.9805	0.9973
$b_{\max} = 4$	平均值	<u>0.9975</u>	0.9853	0.9876	0.9853	0.9996
	最大值	<u>1</u>	0.9940	0.9942	0.9925	<u>1</u>
	最小值	<u>0.9940</u>	0.9770	0.9803	0.9752	0.9975
$b_{\max} = 5$	平均值	<u>0.9952</u>	0.9891	0.8916	0.9872	0.9992
	最大值	<u>0.9988</u>	0.9961	0.9967	0.9942	<u>1</u>
	最小值	<u>0.9888</u>	0.9817	0	0.9793	0.9961
$b_{\max} = 6$	平均值	<u>0.9945</u>	0.9871	0.8802	0.9866	0.9994
	最大值	<u>0.9992</u>	0.9946	0.9963	0.9955	<u>1</u>
	最小值	<u>0.9898</u>	0.9803	0	0.9791	0.9953
$b_{\max} = 7$	平均值	<u>0.9971</u>	0.9897	0.8614	0.9879	0.9997
	最大值	<u>0.9985</u>	0.9956	0.9973	0.9941	<u>1</u>
	最小值	<u>0.9947</u>	0.9806	0	0.9802	0.9979
$b_{\max} = 8$	平均值	<u>0.9942</u>	0.9870	0.7222	0.9865	0.9992
	最大值	<u>0.9992</u>	0.9932	0.9978	0.9963	<u>1</u>
	最小值	<u>0.9895</u>	0.9804	0	0.9789	0.9948
$b_{\max} = 9$	平均值	<u>0.9978</u>	0.9901	0.6251	0.9895	0.9996
	最大值	<u>0.9991</u>	0.9950	0.9981	0.9957	<u>1</u>
	最小值	<u>0.9959</u>	0.9817	0	0.9783	0.9980
$b_{\max} = 10$	平均值	<u>0.9938</u>	0.9896	0.7028	0.9884	0.9994
	最大值	<u>0.9979</u>	0.9953	0.9978	0.9960	<u>1</u>
	最小值	<u>0.9894</u>	0.9820	0	0.9782	0.9963
$b_{\max} = 11$	平均值	<u>0.9938</u>	0.9928	0.6051	0.9919	0.9997
	最大值	<u>0.9979</u>	0.9974	0.9990	0.9975	<u>1</u>
	最小值	<u>0.9894</u>	0.9845	0	0.9835	0.9984

表 4-4 20vs10 场景的不同 b_{\max} 条件下算法运行时间对比表 (单位: s)Table 4-4 Table of comparison of running time in 20vs10 scenarios with different b_{\max} (unit: s)

		HCGSAP	HCGDMCA	JSFP	GRMFMI	PGSAP
$b_{\max} = 2$	平均值	0.0484	0.0205	0.1002	0.0247	0.1318
	最大值	0.0612	0.0268	0.1796	0.0588	0.2416
	最小值	0.0304	0.0179	0.0548	0.0081	0.0791
$b_{\max} = 3$	平均值	0.0328	0.0182	0.0678	0.0058	0.0852
	最大值	0.0600	0.0257	0.1336	0.0162	0.2017
	最小值	0.0267	0.0161	0.0439	0.0026	0.0646
$b_{\max} = 4$	平均值	0.0331	0.0094	0.0750	0.0033	0.0753
	最大值	0.0675	0.0141	0.1292	0.0082	0.2237
	最小值	0.0231	0.0084	0.0517	0.0016	0.0581
$b_{\max} = 5$	平均值	0.0284	0.0073	0.0842	0.0024	0.0801
	最大值	0.0535	0.0106	0.1473	0.0065	0.1733
	最小值	0.0246	0.0068	0.0536	0.0013	0.0600
$b_{\max} = 6$	平均值	0.0275	0.0053	0.0852	0.0017	0.0797
	最大值	0.0438	0.0081	0.1633	0.0043	0.1894
	最小值	0.0231	0.0051	0.0511	0.0010	0.0560
$b_{\max} = 7$	平均值	0.0301	0.0040	0.0873	0.0017	0.0754
	最大值	0.0465	0.0042	0.1577	0.0037	0.1751
	最小值	0.0226	0.0039	0.0504	$< 10^{-3}$	0.0545
$b_{\max} = 8$	平均值	0.0356	0.0035	0.0912	0.0016	0.0777
	最大值	0.0808	0.0050	0.1815	0.0030	0.1849
	最小值	0.0225	0.0034	0.0507	$< 10^{-3}$	0.0548
$b_{\max} = 9$	平均值	0.0444	0.0028	0.0894	0.0016	0.0811
	最大值	0.0801	0.0040	0.1401	0.0039	0.1607
	最小值	0.0220	0.0027	0.0524	$< 10^{-3}$	0.0527
$b_{\max} = 10$	平均值	0.0465	0.0028	0.1034	0.0016	0.0887
	最大值	0.0877	0.0040	0.2011	0.0030	0.2559
	最小值	0.0210	0.0027	0.0489	$< 10^{-3}$	0.0531
$b_{\max} = 11$	平均值	0.0576	0.0023	0.0977	0.0016	0.0950
	最大值	0.0887	0.0034	0.1938	0.0030	0.2268
	最小值	0.0327	0.0020	0.0261	$< 10^{-3}$	0.0509

第五章 基于随机博弈模型的分布式动态任务分配框架

5.1 引言

第三章和第四章是针对静态场景下的任务分配问题，但在解决第二章中提出的动态场景下的任务分配问题时，会面临两大问题：一是环境动态变化难以预测，如出现导弹通信网络变化，或目标数量可能出现变化等情况，使得求解模型中的重要变量，如任务效用函数发生变化，原纳什均衡被破坏；二是前两章提出的利用博弈理论提出的决策算法，由仿真结果可知最终收敛结果与收敛速度具有一定的波动性，实际上与算法开始迭代时的初始解有关。

为了解决以上问题，本章利用随机博弈模型建立了分布式动态任务分配框架，并根据实际场景中可能发生的事件，设计了任务重分配触发机制，在此基础上结合前两章的模型与算法，实现动态环境下的任务分配，并通过仿真结果验证了该框架具有较好动态自适应性。

5.2 随机博弈模型

随机博弈模型一般是有多个参与者参与的具有状态概率转移的动态博弈过程。下面的定义给出了随机博弈的概念。

定义 5.1 (随机博弈) 随机博弈可用一组元组表示 $\mathcal{G} = \langle N, \Omega, \{S_i, U_i\}_{i \in N}, q \rangle$ ，其中 $N = \{1, 2, \dots, n\}$ 表示博弈参与者集合； Ω 表示模型所有的可能状态集合；对于每个可能状态 $\omega \in \Omega$ ，都有一个与之相关的阶段博弈 $\gamma(\omega)$ ，该博弈具有策略空间 S^ω 和效用空间 $U^\omega(s)$ ； $q(\omega^{t+1} | \omega^t, s)$ 是状态转移概率，表示在 t 时刻模型状态 w^t 下，参与者选择策略 s 后，模型在 $t+1$ 时刻转移到状态 w^{t+1} 的概率。

随机博弈一般具有多个阶段，在每个博弈阶段，博弈模型会处在一个特定状态中，参与者根据当前状态和预期回报选择自己的最优策略，接着模型会依据状态转移概率转入下一状态，从而开始新一轮博弈。博弈参与者被认为掌握当前状态的信息，且它所能得到的回报仅仅取决于当前状态和它在该状态下采取的策略，而状态转移概率分布完全由当前状态和参与者的决策策略决定。

5.3 面向动态任务分配问题的随机博弈框架设计

将随机博弈模型应用于导弹任务分配问题，实际上是将某一时刻的任务分配看作是整个模型的状态之一，参与分配的导弹（智能体）即为博弈的参与者，在每个时刻做出的决策即为选择目标。在选择目标后，导弹和目标的飞行会使得环境发生改变，因此下一时刻的任务分配问题便会发生改变。具体而言，本章基于随机博弈模型建立的任务分配框架包含以下几个部分：

- (1) 状态集合，为每个时刻的任务集合 $\mathcal{T}(t), \mathcal{T}(t) \subseteq \mathcal{T}$, t 为当前时刻；
- (2) 智能体集合 $N = \{1, 2 \dots, i, \dots, n_m\}$, 表示博弈的参与者，每个智能体拥有自己的策略集 $\mathcal{A}_i(t) = \{a_k\}$, 即为智能体的可选任务集合，且智能体效用函数定义为 $U_i(a_i, a_{-i})$ ；
- (3) 状态转移概率函数 $q(w^{t+1} | w^t)$ ；
- (4) 全局效用函数 $U_g(a)$ 。

其中(1)、(2)中的任务集合和智能体集合可直接使用前两章中的概念，使用每个时刻参与分配的导弹和目标集合即可。对于(2)中的智能体效用，本章仍使用第三章中的WLU效用和第四章中的效用函数。(3)中的状态转移概率函数取决于模型框架的设计，(4)中的全局效用函数仍沿用前几章的定义。结合第二章2.5.2小节建立的动态环境模型的特点及可能面临的问题，下面将详细针对模型框架进行阐述。

5.3.1 时间窗口划分

动态模型下每个时刻环境的状态都与前一时刻不同，一方面不可能对每一时刻建立任务分配模型进行求解，另一方面求解任务分配也需要时间。因此本文将攻击过程划分为若干时间窗口 $[t, t + w]$ ，每一时间窗口作为一个博弈阶段，在同一窗口内，假设导弹和目标态势关系变化不大，最优任务分配结果不会发生改变，因此可以在一个时间窗口内建立起任务分配模型并求解。

时间窗口长度 w 的选取是一个需要权衡的问题。一方面，由于在一个窗口期内，认为态势变化不大，因此如果时间窗口过长，导弹和目标的位置已发生较大变化，态势变化不大的前提假设不再成立，且一些突发事件可能会被忽略而不能及时响应；另一方面，如果时间窗口过短，一方面可能任务分配结果可能还未解出，另一方面鉴于导弹的运动特性，如果频繁重分配可能导致弹道过于弯曲而造成能量浪费。

根据以上分析，结合第二、三、四章里的相关模型和假设，可以给出时间窗口长度 w 应该满足的条件为

$$w \geq t_C + t_A, \quad (5-1)$$

其中, t_C 表示导弹之间进行通信并达成信息一致所需的时间, t_A 表示导弹进行任务分配决策达到收敛所需的时间。式(5-1)的含义是窗口时间的长度不得少于导弹需要完成信息交互和任务分配求解两个过程的时间。

5.3.2 博弈子模型建立

根据前一小节论述, 在划分出一段时间窗口后, 需要在每个时间窗口期内建立一个博弈子模型以求解任务分配问题, 并且此时可视为静态分配问题求解。因此本小节建立的博弈子模型是基于前两章提出的势博弈分配模型和享乐联盟博弈分配模型。

具体而言, 在一个新的博弈阶段开始时, 各导弹会根据2.5.2小节介绍的模型, 统计自己可跟踪的目标以及可通信的导弹, 并建立起新的通信网络。根据这些信息, 建立势博弈分配模型或享乐联盟博弈分配模型, 此时在博弈模型中需要明确各导弹的可选目标集, 除了限定为可跟踪到的目标集合 $\mathcal{A}_i^{\text{Detect}}$ 的子集, 导弹还需要找出自身可攻击到的目标集合 $\mathcal{A}_i^{\text{Attack}}$ 。结合导弹具有最大过载的特点, 本文选择的目标是否可行的判据是: 判断导弹攻击该目标所需要的制导过载指令是否会超出导弹的最大过载。设导弹最大过载为 g_{\max} , 导弹在当前状态下攻击该目标预计过载指令为 a_M , 若 $a_M > g_{\max}$, 则该目标不再是可选目标, 所以最终各导弹的可选目标集为 $\mathcal{A}_i = \mathcal{A}_i^{\text{Detect}} \cap \mathcal{A}_i^{\text{Attack}}$ 。最优分配的求解具体过程与原理前两章已详细阐述, 此处不再赘述。

5.3.3 博弈阶段切换

在不同博弈阶段切换的时间点, 除了导弹和目标的状态信息、导弹的通信网络发生变化需要更新外, 最重要的是任务分配解的传递与切换。根据前面章节的论述和仿真实验可知, 在一个博弈阶段内建立的分配模型所得到的最优分配解效果, 以及算法收敛速度均与初始解有关。因此在新的博弈阶段子模型建立后, 需要决定新的模型下开始算法迭代的初始分配解。如果每次都采用随机生成的解, 则每次算法收敛到对应的纳什均衡解会造成时间的浪费, 且前文已提到, 随机生成的解最终可能得到不同的均衡解, 甚至效果较差的均衡解; 另一方面, 如果一直采用前一阶段的分配解作为新阶段的初始解, 则由于原分配解已经是纳什均衡解, 且分配模型中的约束条件的存在, 可能会限制导弹选择其他目标, 使得模型陷入局部最优。

为解决该问题，本文使用的方法是，将前一时刻的分配解与随机生成解以一定概率进行交叉变异，生成新的分配解作为新的博弈阶段的初始解。每个导弹以一定概率 p 选择原分配解 a_k 作为初始解，以 $1 - p$ 的概率从 $k + 1$ 时刻可选目标集中随机选择一个目标 a_{rand}^j 作为初始解，则生成 $k + 1$ 时刻博弈模型的初始解 $a_{k+1}(0) = \{a_{k+1}^1(0), a_{k+1}^2(0), \dots, a_{k+1}^{n_m}(0)\}$

$$a_{k+1}^j(0) = I\{e < p\}a_k^j + I\{e \geq p\}a_{\text{rand}}^j, e = \text{rand}(0, 1). \quad (5-2)$$

此处不需要考虑新解的可行性，因为博弈分配算法会自行消解分配冲突，其中原理前面章节已经论述。

切换过程需要考虑的另一个问题是，在上一个博弈阶段得到的任务分配解，是否接受并传递给下一博弈阶段。由于考虑到频繁切换目标会对导弹带来能量的消耗甚至最终不能成功击中目标，因此本文采用的是带惰性的传递方法，当导弹集群获得 k 时刻的分配结果时，若与当前结果不一致，则会以 q 的概率采用新分配，而以 $1 - q$ 的概率拒绝新分配。为了保持分配解的可行性，新分配解的接受或拒绝对所有导弹保持一致，即在同一个连通分支内的所有导弹只要有一枚导弹接受或拒绝新方案，该分支内所有导弹均会作出同样的选择。

5.4 重分配触发机制

虽然在上述随机博弈框架下，假设在一个博弈阶段内态势不会发生重大变化，但实际情况中，发生足以需要重新分配的场景和事件仍是存在的。针对这类突发状况，需要制定一套事件触发机制，明确事件触发信号和触发后重分配方法。

5.4.1 通信网络非连通

在第二章建立的任务分配模型中，导弹存在着通信半径，因此导弹之间建立的通信网络可能存在着非连通的情况。在此情况下，导弹集群往往会展开两个或多个连通分支，无法相互之间共享信息。在此情况下两个不在同一通信分支的导弹可能会生成相互冲突的分配解而无法消除，为了解决这一问题，首先需要引入以下命题。

命题 5.1 (可消解冲突) 如果两枚导弹之间不存在通信，则它们在一个博弈阶段内可以消解冲突的条件是

$$R_{\text{comm}} \geq 2V_{\max}T_k\Delta t + R_{\text{col}}, \quad (5-3)$$

其中, V_{\max} 表示导弹的最大速度, T_k 表示第 k 个博弈阶段的时间步数, Δt 表示单位时间周期, R_{col} 表示导弹碰撞半径。

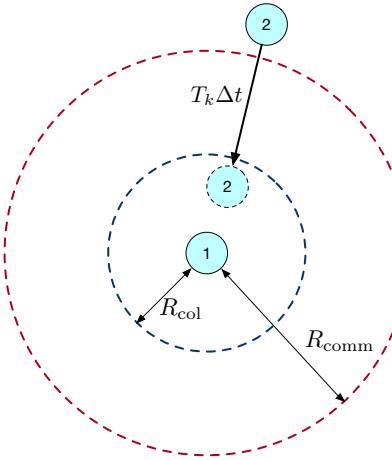


图 5-1 不可规避冲突

Figure 5-1 Illustration of an undetectable collision

在一个时间窗口的博弈阶段内, 导弹不会改变分配目标, 因此如果出现两个没有建立通信的导弹选择了同一目标产生冲突, 则会由于制导律的作用相互靠近, 若在时间窗口期内距离小于碰撞距离, 则两枚导弹就会发生碰撞, 这一过程如图5-1所示, 在第 k 个博弈阶段刚开始时导弹 2 仍处于导弹 1 的通信半径 (红色虚线圆) 外, 但在该阶段时间内, 导弹 2 迅速进入了导弹 1 的最小碰撞半径范围 (蓝色虚线圆), 此时两导弹难以避免碰撞。

因此为了避免这种情况, 对导弹通信半径施加上述条件, 使得在同一窗口期内, 导弹在碰撞前能够及时建立起通信关系。而新的通信关系一旦建立, 会触发重分配条件, 原先的分配解在新模型下不再是可行解, 从而冲突会被消解, 得到的新分配解会适用于新通信网络。这一过程示意图如图5-2所示, 导弹 1 和 2 (蓝色实线实心圆) 同时选择了目标 1 (红色实心圆), 但由于互相处于通信范围外 (红色虚线圆), 并未建立通信, 因此冲突无法消解。但随着对目标的接近, 两导弹之间的距离逐渐减小 (蓝色实心虚线圆), 且达到了建立通信的条件, 此时触发重分配条件, 立即结束当前博弈阶段, 建立新的模型重新分配, 因此导弹 1 会重新选择目标 2, 冲突由此消解。

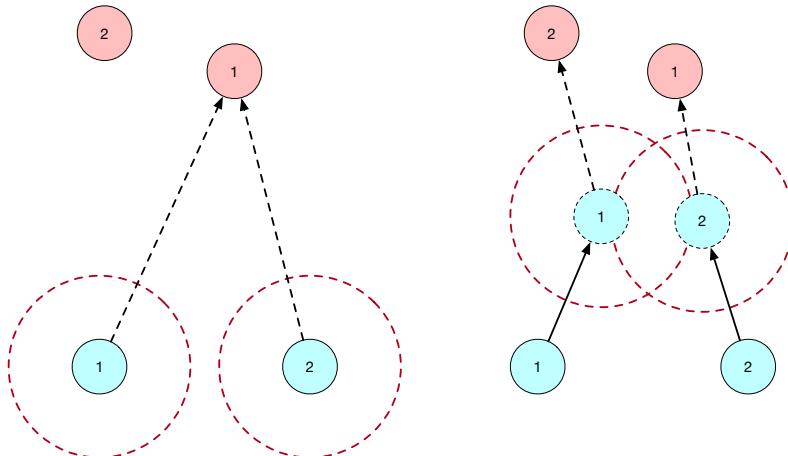


图 5-2 非连通场景下分配冲突消解示意图

Figure 5-2 Resolution of an assignment conflict in the disconnected communication scenes

5.4.2 目标数量可变

目标数量变化可分为目标数量增加和目标数量减少两种场景。目标增加的原因通常有两种：一是导弹发射前已知该目标，但导弹发射后该目标处于导弹探测区外，随着攻击过程的进行才进入导弹探测范围；二是导弹发射前该目标未出现或未被地面雷达或载机捕捉，在导弹飞行过程中被导弹自行捕获。本文不考虑原先被跟踪的目标逃出导弹探测范围的情况，因此目标减少的原因通常就是被导弹击中。

针对目标增加的第一种原因，由于导弹在发射前对于目标数量等信息已知，因此只需结束当前博弈阶段，触发重分配机制，建立新的博弈阶段并求解新的分配解即可，为了使得新探测到的目标进入分配解，在建立新博弈阶段时，首先发现该目标的导弹在确定初始解时不再使用式(5-2)所示的交叉算子，而是直接使用新目标作为初始解，由于在原分配中没有导弹选择新目标，因此在分配模型中新目标将被保留，只能在导弹之间交换而不会被某个导弹单方面放弃。

针对目标增加的第二种原因，由于第二章模型建立中式(2-33)的约束，因此在之前的讨论中导弹数量往往不会多于攻击所有已有目标所需的数量之和，而没有多余的导弹攻击新增的目标。假设新目标 T_w 所需的导弹数量为 $b_{\max}^{(w)}$ ，在不考虑发射新的导弹的情况下，解决这一问题的方案是改变现有模型，从原分配解中满足 $b_{\max}^{(j)} > 1$ 的目标的攻击导弹中抽出 $b_{\max}^{(w)}$ 个导弹，用于攻击新目标^①。

针对目标减少的场景，往往是由于导弹击中目标，因此也包含着导弹数量减

^① 显然这种方案成立的条件是 $n_m - n_t \geq b_{\max}^{(w)}$ ，本文的讨论是基于满足该条件的场景下，对于不满足该条件的场景，本文不再讨论。

少的场景。本章建立的模型中，导弹击中目标的判定条件是弹目距离小于指定阈值 R_{\min} 代表目标进入导弹杀伤范围，或弹目相对接近速度 $V_R > 0$ ，代表弹目距离不再减小。此时击中目标数量若达到 $b_{\max}^{(j)}$ ，则目标被击落，攻击该目标的导弹也退出通信网络与分配模型。当一些导弹攻击完成后，其邻居节点可能失去连接通道，从而不再连通，此时将回到5.4.1小节中所论述的场景。

5.5 系统设计与仿真验证

5.5.1 动态任务分配系统设计

结合5.3节和5.4节的内容，图5–3给出了动态环境下基于随机博弈模型的任务分配系统框图。

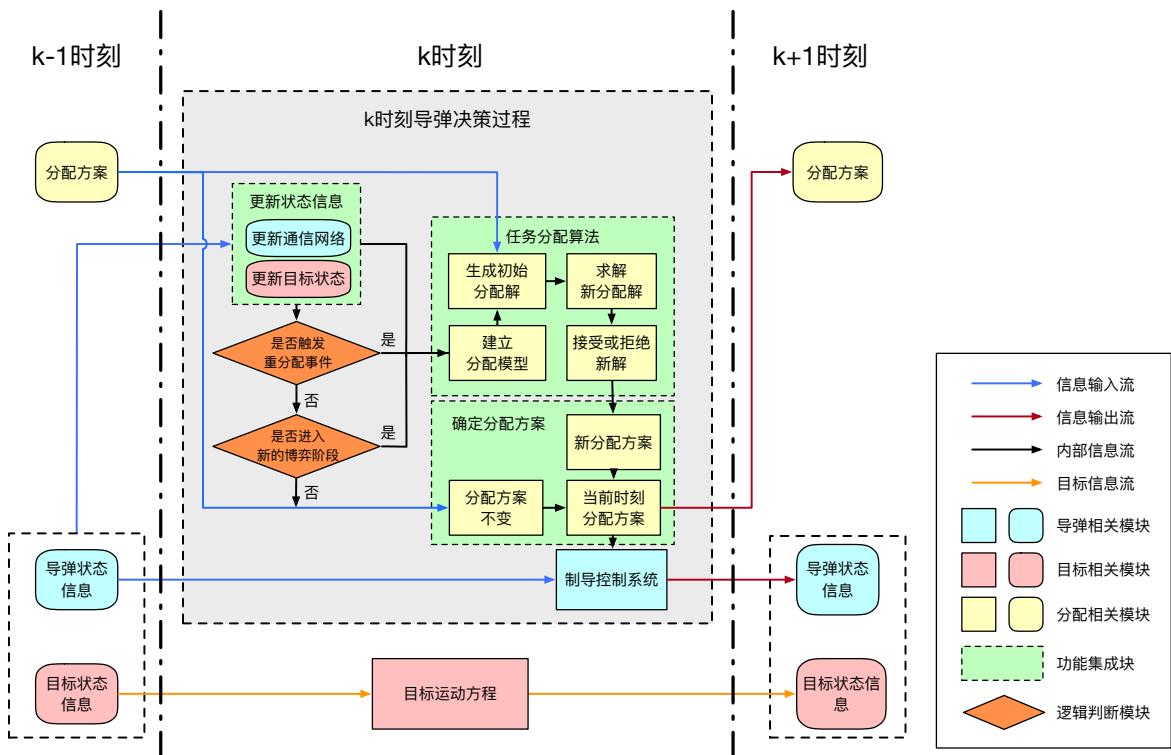


图 5–3 导弹任务分配系统框架示意图

Figure 5–3 Illustration of framework of missiles task assignment system

其中目标运动部分独立在外，不在控制范围内。导弹模块主要分为态势信息收集与更新、事件触发判断、任务分配和制导控制四个模块，各部分的主要功能为：

(1) 态势信息收集与更新。在每个时刻导弹需要获知自身状态信息与可选目标信息，并寻找可建立通信关系的同伴，建立通信网络，共享目标信息。

(2) 事件触发判断部分实现的功能是根据当前时刻收集到的导弹与目标状态信息，判断是否发生了需要立即重新分配的事件。根据上文的讨论，这些事件主要包括导弹通信网络的改变、新目标的出现和目标被导弹击中等场景。发现事件发生的导弹立即进入重分配过程，并将事件情况向可通信导弹发出，促使其他导弹也进入重分配过程。

(3) 任务分配部分在两种场景下会被调用。一种是正常的博弈阶段结束，进入下一阶段时会获取新形势进行新的任务分配过程；另一种则是根据事件触发判断部分的结果，在正常阶段结束前提前进入重分配过程。该部分的主要流程是根据获取的态势建立分配模型，同时利用5.3.3小节中提出的初始解生成方法，生成代入求解模型的初始分配，在得到当前态势下收敛的分配解后，再以一定概率接受或拒绝该分配。

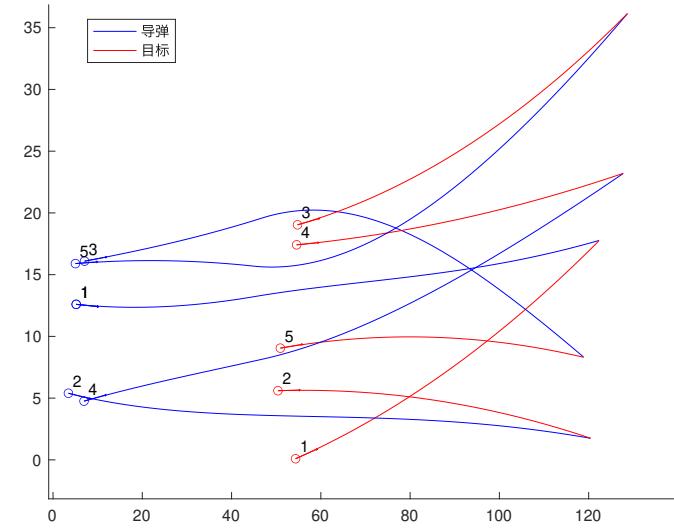
(4) 制导控制部分负责根据导弹计算出的分配结果，计算制导指令，并控制导弹向指定的目标飞去。

在每个时间点，导弹不断重复以上四个步骤，直到该导弹击中目标即停止。当所有导弹均击中目标时，整个导弹集群任务分配系统停止运行。

5.5.2 仿真实验

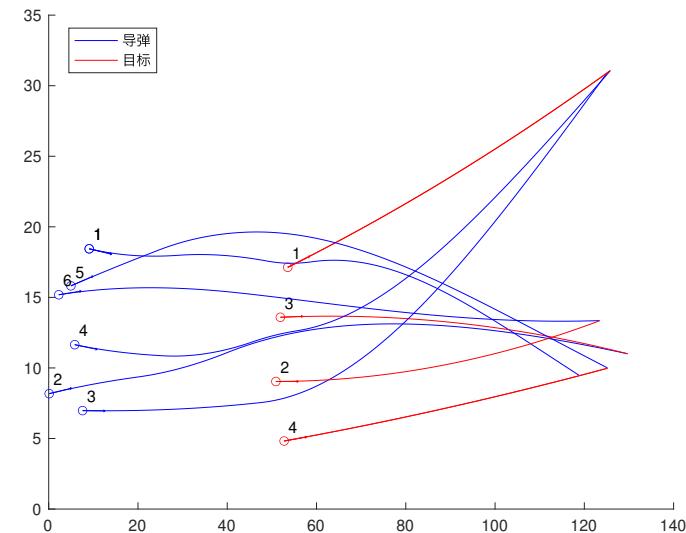
根据5.5.1小节建立的动态任务分配系统，结合第三章和第四章建立的任务分配模型和算法，进行动态环境下的空空导弹任务分配仿真实验。实验中导弹和目标的初始位置、速度、角度等信息随机生成，目标以一定的加速度做机动运动，导弹在发射前可以根据地面雷达或载机收集到已探测到的目标状态信息，并生成初始分配解。在发射后，导弹进入“发射后不管”状态，即导弹与地面或空中平台不再建立通信，只依赖于互相之间的通信进行协同攻击。

图5-4和图5-5分别展示的是使用势博弈模型和HCG模型的动态任务分配结果，图5-4(a)和图5-5(a)展示的是平衡分配场景(5vs5)的分配效果，可以看出所有导弹都被分配到一个目标，且导弹的轨迹相对平滑，说明在此情况下导弹在攻击过程中更改目标的次数不多。而图5-4(b)和图5-5(b)展示的是非平衡分配场景(6vs4)下的分配效果。由图中可以看出，相比平衡分配，在非平衡分配下导弹轨迹更加曲折，说明导弹在中途更换了目标，这表明在飞行过程中导弹态势的变化较大，且对于需要多个导弹攻击的目标，导弹之间相对位置的变化，以及目标的机动可能会导致导弹选择不同的队友进行协同攻击。此外，导弹轨迹在攻击后期



a) 平衡分配场景分配效果 (5vs5)

a) Performance of balance assignment scenario (5vs5)

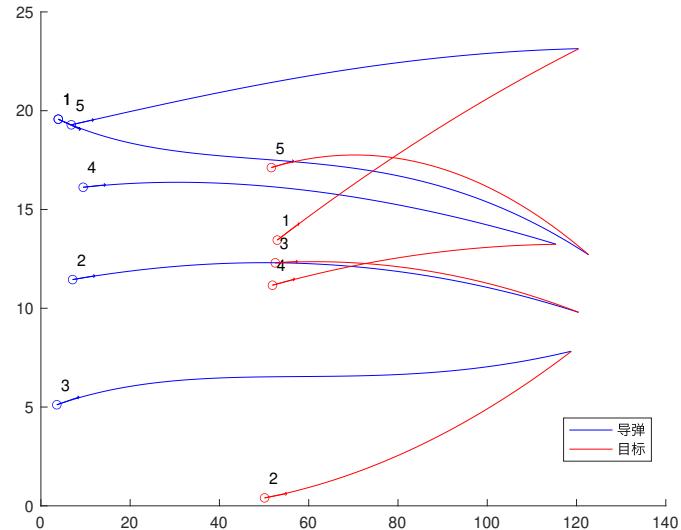


b) 非平衡场景分配效果 (6vs4)

b) Performance of unbalance assignment scenario (6vs4)

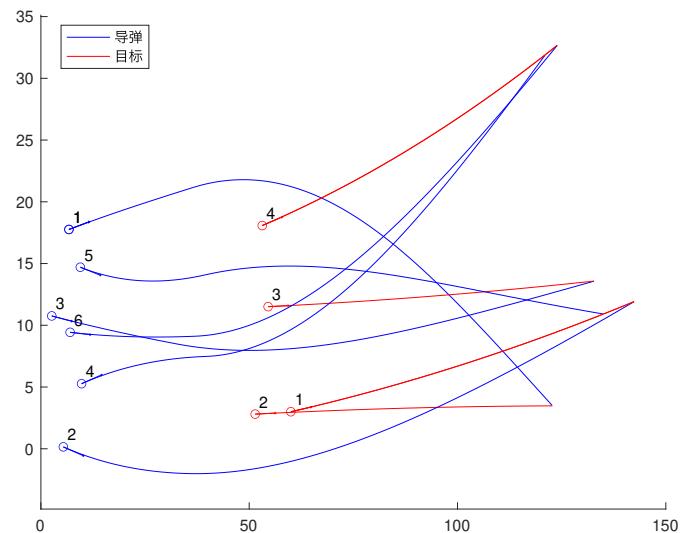
图 5-4 基于势博弈模型的动态任务分配效果

Figure 5-4 Dynamic task assignment based on potential game model



a) 平衡分配场景分配效果 (5vs5)

a) Performance of balance assignment scenario (5vs5)



b) 非平衡场景分配效果 (6vs4)

b) Performance of unbalance assignment scenario (6vs4)

图 5-5 基于 HCG 模型的动态任务分配效果

Figure 5-5 Dynamic task assignment based on HCG model

往往是较为平滑的，这是因为当接近目标的时候，导弹对于各自目标的优势已经足够明显，而此时更换目标所需的时间和机动能量成本过大，因此此时导弹往往不会再更换分配方案。

以上仿真场景中，导弹与目标的状态都假定是事先已知，且导弹通信网络为全连通网络。为了验证本章提出的动态任务分配系统在非理想场景下的有效性，下面将针对5.4小节中讨论的两种重分配场景进行仿真验证。

(1) 通信网络非连通

为了验证在通信网络非连通场景下的系统有效性，将仿真场景设置为导弹初始位置分为两个独立的集群，集群之间距离大于导弹的通信半径，因此两集群之间在初始时没有通信关系。在攻击过程中，两集群可以各自独立地对目标进行跟踪和分配。

图5-6表示的是 10vs10 场景下的导弹攻击过程。在初始时刻，1-5 号导弹和 6-10 号导弹各形成一个集群，且彼此之间的距离超过了导弹的通信半径，因此这两集群之间不存在通信关系，并且部分目标超出了导弹探测范围。由图5-6(a) 也可看出，在初始分配时，6-10 号导弹中只有 6 号选择了 9 号目标，其余都选择了 2 号目标，因为其余目标都不在这个集群内导弹的探测范围内。值得注意的是，这里有部分目标，比如 10 号，处于所有导弹的探测范围外，因此在初始时刻没有导弹选择该目标。

当到了 30s 的时刻，如图5-6(b) 所示，此时随着导弹之间距离的接近，两个集群间开始建立起通信，因此原本冲突的分配得到消解，比如原来集中选择 2 号目标的几枚导弹现在有了各自不同的目标。与此同时，由于导弹与目标的接近，原本不被任何导弹探测的目标现在也进入了导弹集群的探测范围，因而成为可攻击目标，被导弹选中，此时可发现所有导弹都得到了可行分配。

图5-6(c) 展示的是攻击后期的态势图，随着弹目间距离越来越小，导弹的分配方案也趋于稳定，直至如图5-6(d) 所示所有导弹完成攻击。

(2) 目标数量变化

在第一种非连通网络场景下，已经可以看出动态任务分配系统对于新增目标的响应机制，但为了更加直接地验证目标数量可变的场景下的系统动态响应能力，设置仿真场景中的部分目标初始位于导弹集群的探测范围外，因此事先导弹并不知道这些目标的存在，但随着攻击过程的进行，弹目间距离缩小，原本不被发现的目标将可能被发现，从而触发系统的重分配机制。

图5-7展示的是 10vs10 场景下出现新增目标的场景。在初始时刻，1-5 号目标处于导弹集群的探测范围外，因此导弹初始分配方案如图5-7(a) 所示，只在 6-10

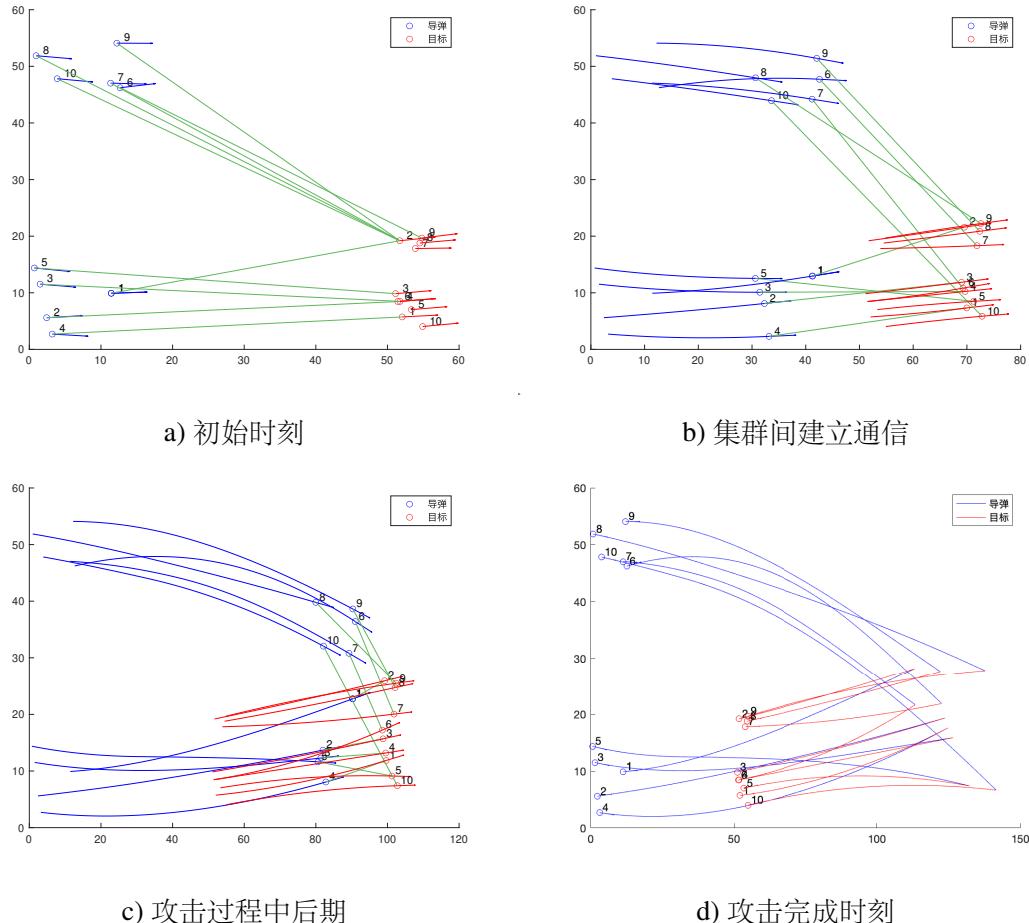


图 5-6 导弹初始通信网络非连通的动态任务分配效果

Figure 5-6 Dynamic task assignment in the scenario with disconnected initial communication network

号目标分配。由于导弹的速度远远大于目标，因此随着攻击过程的进行，原本在探测范围外的目标开始可以被导弹集群捕捉到，此时会触发重分配，增加了可选目标的导弹会重新选择目标，如图5-7(b)所示。

由于弹目距离的远近不同，一部分目标会率先被导弹击中，击中后的导弹与目标同时退出分配模型，剩下未完成攻击的导弹和未被击中的目标继续参与分配过程。如图5-7(c)所示，图中已经完成攻击的导弹与目标位置重叠，为了图示清晰不再画出轨迹，而其他还在攻击过程的导弹则对剩余目标重新分配，并继续飞行，直至所有目标均被击中，攻击完成，如图5-7(d)所示。

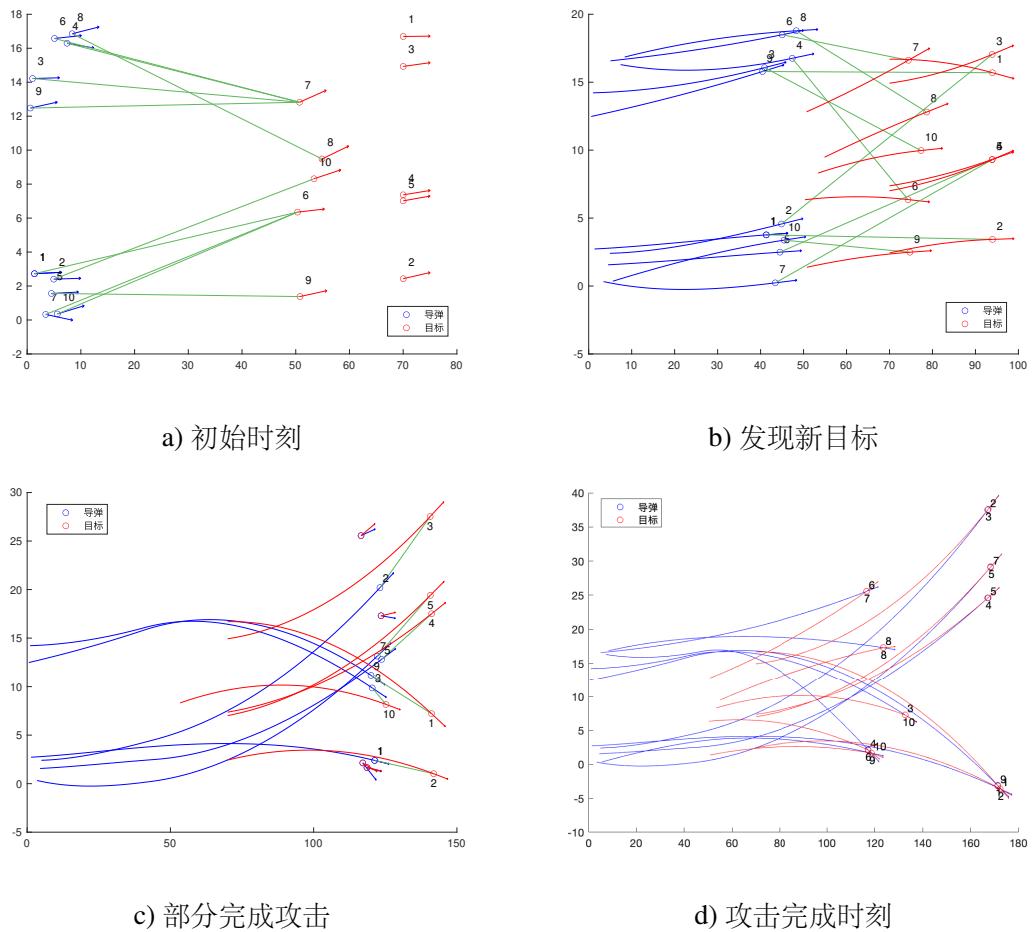


图 5-7 新增目标场景下的动态任务分配效果

Figure 5-7 Dynamic task assignment in the scenario with new target appearance

通过以上仿真实验结果可以看出，本章基于随机博弈模型和任务重分配触发机制设计的动态任务分配系统对于非连通网络和目标数量可变两种非理想场景都具有较好的自适应性和动态响应能力，可以完成在较为复杂场景下的导弹集群动

态任务分配。

5.6 本章小结

本章针对动态环境下的任务分配问题，基于随机博弈模型，将动态分配问题划分为若干时间窗口下的阶段性博弈模型，在每个博弈阶段使用势博弈或偏好联盟博弈建立任务分配模型求解。针对博弈阶段之间的切换，本章设计了阶段开始时的分配解变异机制和阶段结束时的分配解判别机制。由于实际场景中可能遇到新生事件或各种非理想情况，本文针对导弹通信网络非连通和目标数量变化这两种典型场景，各自分析了系统响应方案，并设计了任务重分配机制。最终本章给出了通过设计仿真场景进行实验，验证了本章设计的动态任务分配系统在非理想场景或新生事件下具有良好的适应能力。

第六章 总结与展望

6.1 本文总结

随着集群技术的发展，在现代空战中，大规模集群空战是各个主要军事大国争夺的科技制高点。本文以空空导弹集群为研究目标，考虑了导弹的预计攻击时间和机动消耗能量，将未来的态势因素纳入模型考虑之中，建立了空空导弹任务分配模型。本文以博弈论为理论工具，使用势博弈模型、偏好联盟博弈模型研究了静态分布式任务分配方法，并以随机博弈模型为基础探讨了动态分布式任务分配的解决方案，取得了一些成果，本文的工作和主要贡献如下：

1. 针对分布式静态任务分配问题，本文基于势博弈模型理论设计了任务效用函数和基于 WLU 的智能体效用函数，针对现有使用势博弈模型解决任务分配问题的文献大多没有考虑含约束分配问题，本文引入 Lagrange 乘子对 WLU 效用函数进行修正，并从理论上证明了修正后的 WLU 效用函数的可行性，推导出了使用修正 WLU 效用函数的势博弈模型的性能下界。接着，结合现有文献成果，本文介绍了三种博弈学习算法：JSFP、GRMFMI 和 SAP 算法，并针对博弈学习算法容易过早收敛，陷入局部最优的问题，在 SAP 算法的基础上设计了任务交易机制，提出了含任务交易的 SAP 算法。通过与几种集中式启发算法的对比仿真验证，结果表明本文设计的模型与算法在平衡分配问题和非平衡分配问题上都具有良好的效果，且相比集中式算法在较大规模问题上更具时间优势。
2. 将静态任务分配问题看作联盟形成问题，本文基于偏好联盟博弈理论建立了任务分配模型。在偏好联盟博弈模型中，引入智能体对联盟的偏好关系，并假设智能体对联盟的偏好仅与联盟成员数量有关，而与具体成员无关，并从理论上证明当智能体偏好关系满足社交疏远性质时，偏好联盟博弈模型一定存在纳什均衡。本文根据以上结论设计了边际效用递减的任务效用函数和满足社交疏远性质的智能体效用函数，并针对具体函数推导出偏好联盟博弈模型的性能下界。针对博弈学习算法，本文除了使用势博弈模型中的带任务交易的 SAP 算法外，还介绍了一种分布式一致算法 DMCA，用于实现分布式架构下的联盟形成问题。通过设计仿真实验，本文对比分析了非平衡分配问题中两种非平衡程度对于学习算法的性能影响，结果表明本文提出的两种博弈模型下的五种学习算法在非平衡问题中具有良好的求解能力，并且有着良好的实时性。
3. 针对动态任务分配模型，本文基于随机博弈理论，将动态任务分配模型分解为若干时间窗口下的静态博弈阶段，在每个阶段博弈中建立势博弈或偏好联盟

博弈模型并求解。针对博弈阶段之间的切换问题，设计了初始分配生成机制和分配传递机制，以保证在满足动态要求的前提下，保持分配的稳定性和可行性。针对动态环境中可能出现的非理想情况，本文针对非连通场景和目标数量变化两种场景设计了重分配触发机制，分析了非连通情况下可以触发重分配和消解分配冲突的条件，并针对几种目标数量变化的场景下的重分配进行了分析。本文建立了完整的分布式动态任务分配系统，并通过仿真实验验证了该系统的有效性，并设计了几种非理想情况下的重分配场景，验证了系统在突发事件下具有良好的动态自适应性。

6.2 未来展望

本文基于博弈论的思想对空空导弹的分布式任务分配模型和算法进行了相关研究，但实际场景中大规模多对多空战场景要更为复杂，分配难度也更大，且本文提出的模型和算法也存在着一些局限与问题还需要进一步的研究。下一步工作主要包括以下几个方面：

1. 动态任务分配环境的建模问题。本文建立的任务分配问题模型是基于导弹预计剩余攻击时间和预计机动能量，虽然一定程度上考虑了未来的因素，但实际模型更加复杂，包括导弹集群的路径规划、目标机动、导弹和目标的运动特性等因素对分配的影响尚未考虑在内。如何选取对任务分配有影响的因素，并处理这些因素的动态变化对分配的影响，结合导弹与目标的运动规律，建立更符合实际的任务分配模型是一个仍待解决的问题。
2. 算法的精确度和普适性问题。基于博弈论的任务分配模型和算法可以在较快时间内获得较为精确的解，但受限于模型的设置，这类算法往往比集中式算法更容易陷入局部最优中，因此也导致了求解结果较为分散的问题。针对以上问题，本文虽然提出了一些解决思路，但并没有完全解决。此外，基于博弈论的分配模型受限于模型函数和参数的设计，设计不当可能会使得模型在某些场景下失效，本文针对模型的设计和参数的选择范围进行了初步探究，但这个问题仍需更多的工作。
3. 动态任务重分配机制设计问题。本文只研究了通信网络非连通和目标数量变化两种场景下的重分配问题，但实际动态场景中，需要进行重分配的事件更多更复杂，比如导弹发生故障、目标突然丢失等等；此外，当重分配事件增多时，如何协调这些事件的优先级和处理方式本文没有深入展开研究。

参考文献

- [1] Monderer D, Shapley L S. Potential Games[J]. Games & Economic Behavior, 1996, 14(1): 124-143.
- [2] 刘宏国, 宗振国, 张春秋, 等. 能源互联网背景下基于势博弈的多供电主体竞争策略研究[J]. 电测与仪表, 2020: 1-7.
- [3] 刘鹏煌. 基于势博弈的拥堵收费双层定价模型[J]. 物流科技, 2020, 43(05): 109-115+118.
- [4] 邵崇, 张少华. 基于势博弈和 IAPSO 的多能源微网分布式调度方法[J]. 工业控制计算机, 2020, 33(11): 136-139.
- [5] Mahn T, Wirth M, Klein A. Game Theoretic Algorithm for Energy Efficient Mobile Edge Computing with Multiple Access Points[C]. in: 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). 2020: 31-38.
- [6] 刘继军, 邹山花, 卢先领. MEC 中资源分配与卸载决策联合优化策略[J]. 计算机科学与探索, 2020: 1-11. DOI: 10.3778/j.issn.1673-9418.2006087.
- [7] 张艮山, 刘旭宁. 资源受限移动边缘计算任务拆分卸载调度决策[J]. 计算机应用与软件, 2019, 36(10): 268-273+278.
- [8] 巩俊辉, 胡晓辉. 基于博弈论的 WSN 拓扑控制及入侵检测研究[D]. 兰州交通大学, 2020.
- [9] 刘泽宁, 李凯, 吴连涛, 等. 多层次算力网络中代价感知任务调度算法[J]. 计算机研究与发展, 2020, 57(09): 1810-1822.
- [10] 贺秋歌, 王慧娇, 董荣胜. 基于势博弈水下无线传感器网络拓扑控制算法[J]. 计算机工程与设计, 2017, 38(10): 2616-2622.
- [11] 何亚光, 赵子豪, 李泽滔. 基于势博弈的 WSN 非均匀拓扑控制算法[J]. 计算机工程, 2019, 45(09): 65-69.
- [12] Chen X, Huang J. Spatial Spectrum Access Game[J]. IEEE Transactions on Mobile Computing, 2015, 14(3): 646-659. DOI: 10.1109/TMC.2014.2326673.

- [13] 郝晓辰, 姚宁, 汝小月, 等. 基于生命期模型的无线传感器网络信道分配博弈算法[J]. 物理学报, 2015(14): 1-11.
- [14] 贾杰, 张桂园, 陈剑, 等. 无线传感器网络中基于潜在博弈的分布式节点定位[J]. 电子学报, 2014, 42(09): 1724-1730.
- [15] Moragrega A, Closas P, Ibáñez C. Potential Game for Energy-Efficient RSS-Based Positioning in Wireless Sensor Networks[J]. IEEE Journal on Selected Areas in Communications, 2015, 33(7): 1394-1406. DOI: 10.1109/JSAC.2015.2430172.
- [16] 曹欣, 高林. 基于势博弈的群智感知分布式任务调度与激励机制[D]. 哈尔滨工业大学, 2020.
- [17] Arslan G, Marden J R, Shamma J S. Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation[J]. Journal of Dynamic Systems, Measurement, and Control, 2007, 129(5): 584-596 [2020-12-16].
- [18] 韩鹏, 丁桂强, 刘国彬. 基于博奕论的多雷达多目标分配方法研究[J]. 现代雷达, 2020, 42(02): 16-22+29.
- [19] Chapman A C, Micillo R A, Kota R, et al. Decentralized Dynamic Task Allocation Using Overlapping Potential Games[J]. The Computer Journal, 2010, 53(9): 1462-1477. DOI: 10.1093/comjnl/bxq023.
- [20] Bakolas E, Lee Y. Decentralized Game-Theoretic Control for Dynamic Task Allocation Problems for Multi-Agent Systems[J]. ArXiv:2009.08628 [cs, eess], 2020 [2020-12-16].
- [21] Marden J R, Shamma J S. Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation[C]. in: 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton). Monticello, IL, USA: IEEE, 2010: 1171-1172 [2020-12-16].
- [22] Marden J R, Arslan G, Shamma J S. Joint Strategy Fictitious Play With Inertia for Potential Games[J]. IEEE Transactions on Automatic Control, 2009, 54(2): 208-220 [2020-12-16].
- [23] Lhazmir S, Elmachkour M, Kobbane A. Green opportunistic access for cognitive radio networks: A regret matching based approach[C]. in: 2018 International Conference on Advanced Communication Technologies and Networking (CommNet). 2018: 1-6. DOI: 10.1109/COMMNET.2018.8360261.

- [24] Deng W, Kamiya S, Yamamoto K, et al. Replica Exchange Spatial Adaptive Play for Channel Allocation in Cognitive Radio Networks[C]. in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). 2019: 1-5. DOI: 10.1109/VTCSpring.2019.8746346.
- [25] Dreze J H, Greenberg J. Hedonic Coalitions: Optimality and Stability[J]. Econometrica, 1980, 48(4): 987 [2020-12-18].
- [26] Bogomolnaia A, Jackson M O. The Stability of Hedonic Coalition Structures[J]. Games and Economic Behavior, 2002, 38(2): 201-230 [2020-12-18].
- [27] 魏斯文, 杨婷婷. 基于联盟博弈的海上认知无线电网络资源分配[D]. 大连海事大学, 2020.
- [28] Saad W, Han Z, Basar T, et al. Hedonic Coalition Formation Games for Secondary Base Station Cooperation in Cognitive Radio Networks[C]. in: 2010 IEEE Wireless Communication and Networking Conference. Sydney, Australia: IEEE, 2010: 1-6 [2020-12-16].
- [29] Saad W, Han Z, Basar T, et al. A selfish approach to coalition formation among unmanned air vehicles in wireless networks[C]. in: 2009 International Conference on Game Theory for Networks. Istanbul, Turkey: IEEE, 2009: 259-267 [2020-12-18].
- [30] 曹扬, 张斌, 邱雪松, 等. 协作干扰下的无线安全传输联盟享乐博弈[J]. 北京邮电大学学报, 2017, 40(S1): 20-23+33.
- [31] Saad W, Han Z, Basar T, et al. Hedonic Coalition Formation for Distributed Task Allocation among Wireless Agents[J]. IEEE Transactions on Mobile Computing, 2011, 10(9): 1327-1344 [2020-12-16].
- [32] Aziz H, Gaspers S, Gudmundsson J, et al. Welfare Maximization in Fractional Hedonic Games[C]. in: International Conference on Artificial Intelligence. 2015.
- [33] Aziz H, Brandl F, Brandt F, et al. Fractional Hedonic Games[J/OL]. ACM Trans. Economics and Comput., 2019, 7(2): 6:1-6:29. <https://doi.org/10.1145/3327970>. DOI: 10.1145/3327970.

- [34] Aziz H, Gaspers S, Gudmundsson J, et al. Welfare Maximization in Fractional Hedonic Games[C/OL]. in: Yang Q, Wooldridge M J. Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. AAAI Press, 2015: 461-467. <http://ijc.ai.org/Abstract/15/071>.
- [35] Chen S, Liu W, Liu J, et al. Maximizing Social Welfare in Fractional Hedonic Games using Shapley Value[C]. in: 2019 IEEE International Conference on Agents (ICA). 2019: 21-26. DOI: 10.1109/AGENTS.2019.8929212.
- [36] Janovsky P, Deloach S A. Multi-agent Simulation Framework for Large-Scale Coalition Formation[C]. in: 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI). Omaha, NE, USA: IEEE, 2016: 343-350 [2020-12-16].
- [37] Jang I, Shin H S, Tsourdos A. Anonymous Hedonic Game for Task Allocation in a Large-Scale Multiple Agent System[J]. IEEE Transactions on Robotics, 2018, 34(6): 1534-1548 [2020-12-18].
- [38] Czatnecki E, Dutta A. Hedonic Coalition Formation for Task Allocation with Heterogeneous Robots[C]. in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). Bari, Italy: IEEE, 2019: 1024-1029 [2020-12-18].
- [39] 李世豪, 丁勇. 复杂空战环境下基于博弈模型的无人机机动决策方法研究[D]. 南京航空航天大学, 2019.
- [40] 孙骞, 薛雷琦, 高岭, 等. 基于随机博弈与禁忌搜索的网络防御策略选取[J]. 计算机研究与发展, 2020, 57(04): 767-777.
- [41] 周洪, 要若天, 余昶, 等. 复杂微电网控制中的随机博弈与优化研究[J]. 智能科学与技术学报, 2020, 2(03): 251-260.
- [42] Marden J R. State based potential games[J]. Automatica, 2012, 48(12): 3075-3088 [2020-12-16].
- [43] Liu X, Ding G, Yang Y, et al. A stochastic game framework for joint frequency and power allocation in dynamic decentralized cognitive radio networks[J]. AEU - International Journal of Electronics and Communications, 2013, 67(10): 817-826 [2020-12-18].
- [44] 马文, 李辉, 王壮, 等. 基于深度随机博弈的近距空战机动决策[J]. 系统工程与电子技术, 2020: 1-12.

- [45] 刘豪, 赵高长; 苏军. 多智能体博弈强化学习算法及其均衡研究[D]. 西安科技大学, 2020.

致 谢

感谢那位最先制作出博士学位论文 **LATEX** 模板的交大物理系同学！

感谢 William Wang 同学对模板移植做出的巨大贡献！

感谢 @weijianwen 学长一直以来的开发和维护工作！

感谢 @sjtug 以及 @dyweb 对 0.9.5 之后版本的开发和维护工作！

感谢所有为模板贡献过代码的同学们，以及所有测试和使用模板的各位同学！

感谢 **LATEX** 和 SJTUThesis，帮我节省了不少时间。

攻读学位期间发表（或录用）的学术论文

- [1] Chen H, Chan C T. Acoustic cloaking in three dimensions using acoustic metamaterials[J]. Applied Physics Letters, 2007, 91:183518.
- [2] Chen H, Wu B I, Zhang B, et al. Electromagnetic Wave Interactions with a Metamaterial Cloak[J]. Physical Review Letters, 2007, 99(6):63903.

攻读学位期间获得的科研成果

[1] 第一发明人,“永动机”,专利申请号 202510149890.0