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　赟

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 13 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 13, 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
学位论文版权使用授权书

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。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保　密□，在 _____ 年解密后适用本授权书。

本学位论文属于

不保密☑。

（请在以上方框内打✓）

学位论文作者签名：某某　　指导教师签名：某某

日　期：某年某月某日　　日　期：某年某月某日

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，如无人机集群投入实战的趋势，在空空导弹拦截大规模集群的场景中，由于集中式算法在求解大规模任务分配问题时面临通信限制和实时性问题，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，通过仿真试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，设计了无约束条件下导弹个体效用函数和协商协议，通过对比实验分析了该算法的有效性和实时性。然后针对整数规划问题的特点，提出了基于拉格朗日乘子法和基于改进美好生活效用（Wonderful Life Utility, WLU）的有约束条件下任务分配算法，通过理论分析和仿真对比验证了两种算法在求解有约束条件下的任务分配问题时保留了无约束优化时的有效性

和实时性，具有良好的实用性和可扩展性。

2、针对上述整数规划模型，利用博弈论中的享乐博弈模型，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弈模型下的关于种群数量的个体效用函数，然后基于 Zeuthen 策略设计了一种分布式协商协议，接着从理论方面分析了该算法的最坏优化值及迭代步数，并通过仿真对比实验验证了算法的有效性和实时性。

3、针对高动态场景下需要及时调整任务分配的问题，研究了基于随机博弈模型的动态任务分配问题。设计了导弹与目标对战随机博弈场景框架，在假定静态模型区间内利用势博弈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，势博弈，享乐联盟博弈，随机博弈

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

插图索引

表格索引

算法索引

主要符号对照表

第一章　绪论

1.1　课题研究背景及意义

1.2　国内外研究现状

1.2.1　空空导弹任务分配技术研究现状

1.2.2　博弈论研究现状及其在任务分配技术中的应用

1.3　主要研究内容

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

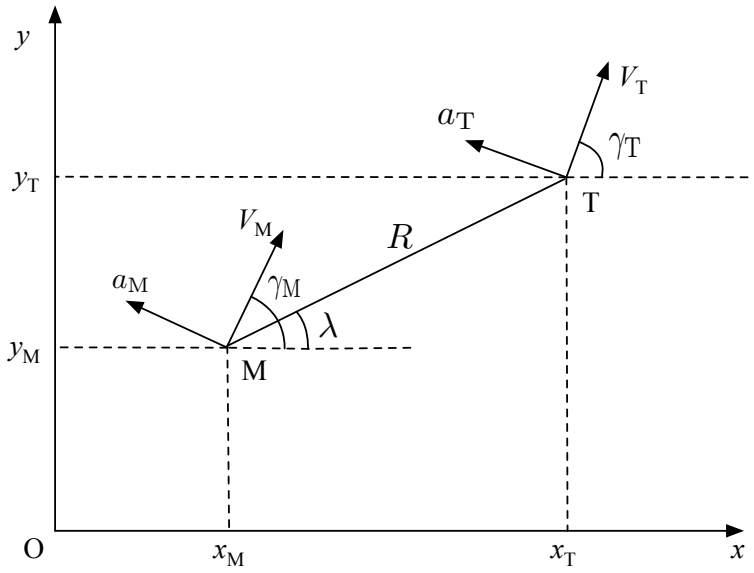


图 2–1 弹目相对运动模型

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，$[x_{(\cdot)}, y_{(\cdot)}], V_{(\cdot)}, a_{(\cdot)}, \gamma_{(\cdot)}$ 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：

$$\dot{x_{\mathrm{M}}} = V_{\mathrm{M}}\cos\gamma_{\mathrm{M}},\ \dot{y_{\mathrm{M}}} = V_{\mathrm{M}}\sin\gamma_{\mathrm{M}},\ \dot{\gamma_{\mathrm{M}}} = \frac{a_{\mathrm{M}}}{V_{\mathrm{M}}}, \tag{2–1}$$

$$\dot{x_{\mathrm{T}}} = V_{\mathrm{T}}\cos\gamma_{\mathrm{T}},\ \dot{y_{\mathrm{T}}} = V_{\mathrm{T}}\sin\gamma_{\mathrm{T}},\ \dot{\gamma_{\mathrm{T}}} = \frac{a_{\mathrm{T}}}{V_{\mathrm{T}}}. \tag{2–2}$$

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：

$$V_R \equiv \dot{R} = V_{\mathrm{T}}\cos\sigma_{\mathrm{T}} - V_{\mathrm{M}}\cos\sigma_{\mathrm{M}}, \tag{2–3}$$

$$V_\lambda \equiv R\dot{\lambda} = V_{\mathrm{T}}\sin\sigma_{\mathrm{T}} - V_{\mathrm{M}}\sin\sigma_{\mathrm{M}}, \tag{2–4}$$

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

$$\sigma_{\mathrm{M}} = \gamma_{\mathrm{M}} - \lambda, \quad \sigma_{\mathrm{T}} = \gamma_{\mathrm{T}} - \lambda. \tag{2–5}$$

由此可求得视线角变化率：

$$\dot{\lambda} = \frac{V_{\mathrm{T}}\sin\sigma_{\mathrm{T}} - V_{\mathrm{M}}\sin\sigma_{\mathrm{M}}}{R}. \tag{2–6}$$

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 $V_{\mathrm{T}} = 0$。此时由式 (2–1)、(2–5) 和 (2–6) 可得：

$$\dot{\sigma_{\mathrm{M}}} = \frac{a_{\mathrm{M}}}{V_{\mathrm{M}}} + \frac{V_{\mathrm{M}}\sin\sigma_{\mathrm{M}}}{R}, \tag{2–7}$$

结合式 (2–3) 可得：

$$\frac{d\sigma_{\mathrm{M}}}{dR} = -\frac{a_{\mathrm{M}}}{V_{\mathrm{M}}^2\cos\sigma_{\mathrm{M}}} - \frac{1}{R}\tan\sigma_{\mathrm{M}}. \tag{2–8}$$

为方便叙述，令 $\rho = \sin\sigma_{\mathrm{M}}$，代入式 (2–8) 可得：

$$\frac{d\rho}{dR} = -\frac{a_{\mathrm{M}}}{V_{\mathrm{M}}^2} - \frac{1}{R}\rho. \tag{2–9}$$

此处为方便推导，考虑导弹采用比例制导律，制导指令为：

$$a_{\mathrm{M}} = NV_{\mathrm{M}}\dot{\lambda}, \tag{2–10}$$

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

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

式 (2–11) 是一个一阶微分方程，其解形式为：

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

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：

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

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 $\rho \approx \sigma_{\mathrm{M}}$ 可得：

$$t_f = \frac{R_0}{V_{\mathrm{M}}}\left(1+\frac{1}{2(2N-1)}\sigma_{\mathrm{M0}}^2\right). \tag{2–14}$$

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：

$$t_{\mathrm{go}} = \frac{R}{V_{\mathrm{M}}}\left(1+\frac{1}{2(2N-1)}\sigma_{\mathrm{M}}^2\right). \tag{2–15}$$

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：

$$\hat{\sigma}_{\mathrm{T}} = \frac{a_{\mathrm{T}}}{V_{\mathrm{T}}} \cdot \hat{t}_{\mathrm{go}}^{p}, \quad \hat{\lambda} = \frac{V_{\mathrm{T}} \sin \sigma_{\mathrm{T}}}{R} \cdot \hat{t}_{\mathrm{go}}^{p}, \tag{2–16}$$

其中 $\hat{t}_{\mathrm{go}}^{p}$ 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：

$$\hat{t}_{\mathrm{go}} = \frac{R}{V_{\mathrm{MT}}}\left(1 + \frac{1}{2(2N-1)}\sigma_{\mathrm{M}}^{2}\right) \tag{2–17}$$

其中 $V_{\mathrm{MT}} = V_{\mathrm{M}} - V_{\mathrm{T}} \cos(\sigma_{\mathrm{T}} + \hat{\sigma}_{\mathrm{T}} - \lambda - \hat{\lambda})$。

2.4 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 $a_{\mathrm{T}} = 0$ 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：

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \tag{2–18}$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & \boldsymbol{I}_2 & 0 \\ 0 & 0 & \boldsymbol{I}_2 \\ 0 & 0 & -\frac{1}{\tau}\boldsymbol{I}_2 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau}\boldsymbol{I}_2 \end{bmatrix}, \tag{2–19}$$

其中状态向量为 $\boldsymbol{x} = [\boldsymbol{R}^{\mathrm{T}}, \dot{\boldsymbol{R}}^{\mathrm{T}}, \boldsymbol{a}_{\mathrm{M}}^{\mathrm{T}}]^{\mathrm{T}}$，各状态分量分别为对应变量的向量形式，$\boldsymbol{I}_2$ 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

$$\min \quad J = \frac{1}{2}\int_{t_0}^{t_f} \boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}dt, \tag{2–20}$$

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

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

$$u = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}k^{-1})\boldsymbol{M}, \tag{2–22}$$

其中 k 是当 $u = 0$ 时的预计末端脱靶量，可由下式求得：

$$\boldsymbol{M} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{X}. \tag{2–23}$$

由下列微分方程

$$\dot{\boldsymbol{P}} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} = 0,\ \boldsymbol{P}(t_f) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{2–24}$$

可求得 $\boldsymbol{P}$ 的闭环表达式为

$$\boldsymbol{P} = \begin{bmatrix} 1 \\ t_{\mathrm{go}} \\ \tau^2(e^{-T} + T - 1) \end{bmatrix}, \tag{2–25}$$

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

$$k = -\int_t^{t_f} (\boldsymbol{P}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{P})dt, \tag{2–26}$$

可求得 k 等于

$$k = \tau^3(\frac{1}{2}e^{-2T} + 2Te^{-T} - \frac{T^3}{3} + T^2 - T - \frac{1}{2}). \tag{2–27}$$

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

$$u = \frac{N'}{t_{\mathrm{go}}^2}[\boldsymbol{R} + \dot{\boldsymbol{R}}t_{\mathrm{go}} - \tau^2(e^{-T} - 1 + T)\boldsymbol{a}_{\mathrm{M}}], \tag{2–28}$$

其中 N' 为扩展比例系数：

$$N' = T^2(e^{-T} - \boldsymbol{I}_2 + T)(-\frac{1}{2}e^{-2T} - 2Te^{-T} + \frac{1}{3}T^3 - T^2 + T + \frac{1}{2}\boldsymbol{I}_2)^{-1}. \tag{2–29}$$

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：

$$u = \frac{N'}{t_{\mathrm{go}}^2}[\boldsymbol{R} + \dot{\boldsymbol{R}} t_{\mathrm{go}} - \tau^2(e^{-T} - 1 + T)\boldsymbol{a}_{\mathrm{M}} - \frac{t_{\mathrm{go}}^2}{2} a_{\mathrm{T}}]. \tag{2–30}$$

最优性能指标 $\boldsymbol{J}^*$ 为

$$\boldsymbol{J}^* = \frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t_0)\boldsymbol{P}(t_0)\boldsymbol{P}^{\mathrm{T}}(t_0)\boldsymbol{X}(t_0). \tag{2–31}$$

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 $\mathcal{M} := \{\mathcal{M}_1, \cdots, \mathcal{M}_{n_v}\}$；战场上有 n_t 个目标需要攻击，其集合表示为 $\mathcal{T} := \{\mathcal{T}_0, \mathcal{T}_1, \cdots, \mathcal{T}_{n_t}\}$，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 $\mathcal{T}_0 \in \mathcal{A}_i$, $i = 1, \cdots, n_m$，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 $a = (a_1, \cdots, a_{n_m}) \in \mathcal{A}$ 组成了一个分配解。设 $S(a) = \{S_1, \cdots, S_{n_t}\}$，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 $s_j = |S_j|, j = 1, \cdots, n_t$。

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：

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

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

$$s_j \geq 1,\ j = 1, \cdots, n_t, \tag{2–34}$$

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。

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

$$\text{s.t.} \quad s_j \le b_{\max}^{(j)},\ j = 1,\cdots,n_t. \tag{2–36}$$

$U_{\mathcal{T}_j}(a)$ 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：

$$U_{\mathcal{T}_j}(a) = \max\{0, r_j - w_1 \max_{\mathcal{M}_i\in S_j} t_{ij} - w_2 C_{\mathcal{T}_j}\}, \tag{2–37}$$

其中，w_1 和 w_2 为权重值；r_j 是攻击目标 $\mathcal{T}_j$ 可获得的奖励值，本文假设该值仅与目标属性有关，与攻击的导弹无关；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。$C_{\mathcal{T}_j}$ 表示导弹攻击目标 $\mathcal{T}_j$ 的成本函数，定义为：

$$C_{\mathcal{T}_j} = \sum_{\mathcal{M}_i\in S_j} J_{ij}, \tag{2–38}$$

其中 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。

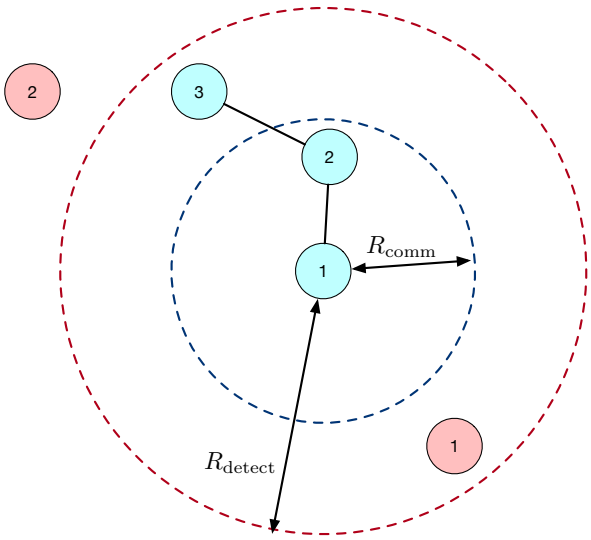


图 2–2 导弹探测与通信示意图

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，$\boldsymbol{R}_{\text{detect}}$ 和 $\boldsymbol{R}_{\text{comm}}$ 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

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

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　基于势博弈模型的分布式任务分配

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 $a^* = (a_1^*, \cdots, a_{n_m}^*)$，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

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

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

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

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

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

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

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

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:

$$U_{\mathcal{M}_i}(a_i', a_{-i}) - U_{\mathcal{M}_i}(a_i'', a_{-i}) = \phi(a_i', a_{-i}) - \phi(a_i'', a_{-i}). \tag{3–3}$$

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 美好生活效用函数

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

$$U_{\mathcal{M}_i}(a_i', a_{-i}) - U_{\mathcal{M}_i}(a_i'', a_{-i}) = U_g(a_i', a_{-i}) - U_g(a_i'', a_{-i}), \tag{3–4}$$

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，本节采用的是美好生活效用函数（Wonderful Life Utility, WLU），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：

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

由全局效用函数的定义式（2–32）可将式（3–5）改写为

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

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

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

□

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：

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

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

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

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

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

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 $\widetilde{U}_g(a)$ 与式（3–9）定义的智能体效用函数 $\widetilde{U}_{\mathcal{M}_i}(a_i, a_{-i})$ 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

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

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 $\widetilde{U}_g(a)$。 □

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

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

证明 设 $a^* = (a^*_i, a^*_{-i})$ 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 $1 \leq j \leq n_t, s^*_j > b^{(j)}_{\max}$。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

$$\begin{aligned}
&\widetilde{U}_{\mathcal{M}_k}(a^*_k, a^*_{-k}) - \widetilde{U}_{\mathcal{M}_k}(a'_k, a^*_{-k}) \\
&= \widetilde{U}_{\mathcal{T}_j}(a^*_k, a^*_{-k}) - \widetilde{U}_{\mathcal{T}_l}(a'_k, a^*_{-k}) \\
&= U_{\mathcal{T}_j}(a^*_k, a^*_{-k}) - \lambda(s^*_j - b^{(j)}_{\max}) - [U_{\mathcal{T}_l}(a'_k, a^*_{-k}) + \lambda[b^{(l)}_{\max} - s'_l]^-] \\
&\leq U_{\mathcal{T}_j}(a^*_k, a^*_{-k}) - U_{\mathcal{T}_l}(a'_k, a^*_{-k}) - \lambda(s^*_j - b^{(j)}_{\max}) \\
&\leq \max_{\mathcal{T}_j \in \mathcal{T}} \{U_{\mathcal{T}_j}(a^*_i, a^*_{-i}) - U_{\mathcal{T}_j}(a'_i, a^*_{-i})\} - \lambda_0(s^*_j - b^{(j)}_{\max})
\end{aligned} \tag{3–12}$$

由 λ_0 的定义易知 $\widetilde{U}_{\mathcal{M}_{k_0}}(a^*_{k_0}, a^*_{-k_0}) - \widetilde{U}_{\mathcal{M}_{k_0}}(a'_{k_0}, a^*_{-k_0}) \leq 0$，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。 □

3.3.2.3 纳什均衡最优性

在命题3.3下，势博弈模型 $\mathcal{G}$ 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 的定义为全局最优效用和纳什均衡效用的最小值之比：

$$\text{PoA}(\mathcal{G}) = \frac{U_g(a^{\text{opt}})}{\min_{a^*} U_g(a^*)} \tag{3–13}$$

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，即意味着该势博弈模型的性能越差。关于本节建立的势博弈模型 $\mathcal{G}$ 的 PoA，我们可以用以下命题得到 PoA 的上界，从而保证了势博弈模型性能的下界。

命题 3.4 势博弈模型 $\mathcal{G}_{\text{PG}}$ 的 PoA 上界为

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

其中

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

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

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

将式（3–4）代入可得

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

不等式右侧可以写为

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

所以有

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

两边同时除以 $\widetilde{U}_g(a^*)$，

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

因此

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

□

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 基于势博弈模型的任务分配算法

3.4.1 单方决策

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

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

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

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

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

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

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

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

类似地可定义出 $z_{-i}(\overline{a}, k)$ 的定义为

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

因此此时智能体 i 选择目标 $\hat{a}_i$ 的期望效用是

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

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

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

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 $\overline{U}_i^{\hat{a}_i}(k) = U_i(\hat{a}_i, z_{-i}(k))$，则式（3–28）可改写为迭代形式：

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

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
随机初始化分配解 a;
$\overline{U}_i(0) \leftarrow \mathbf{0}_{1\times n_t}, i = 1, \cdots, n_t$;
$k \leftarrow 1$;
while *true* **do**
  for $i = 1 : n_m$ **do**
    计算选择不同目标 $\overline{a}_i$ 的 $U_{\mathcal{M}_i}(\hat{a}_i, a_{-i}(k))$;
    使用式（3–29）更新 $\overline{U}_i(k)$;
    $a(k) \leftarrow \arg\max_{\alpha\in\mathcal{A}_i} \overline{U}_i(k)$;
    $k \leftarrow k+1$;
  end
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 $\overline{a}_i^j$ 平均后悔值为

$$R_{\mathcal{M}_i}^j(k) = \frac{1}{k-1}\sum_{t=1}^{k-1}\left[U_{\mathcal{M}_i}(\overline{a}_i^j, a_{-i}(t)) - U_{\mathcal{M}_i}(a(t))\right], j = 1, \cdots, |\mathcal{A}_i|. \tag{3–30}$$

式（3–30）可改写为迭代形式

$$R^j_{\mathcal{M}_i}(k+1)=\frac{k-1}{k}R^j_{\mathcal{M}_i}(k)+\frac{1}{k}[U_{\mathcal{M}_i}(\overline{a}^j_i,a_{-i}(t))-U_{\mathcal{M}_i}(a(t))],\quad k>1. \tag{3–31}$$

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

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

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

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

$$\widetilde{R}^j_{\mathcal{M}_i}(k+1)=(1-\rho)\widetilde{R}^j_{\mathcal{M}_i}(k)+\rho[U_{\mathcal{M}_i}(\overline{a}^j_i,a_{-i}(t))-U_{\mathcal{M}_i}(a(t))],\quad j\in\{1,\cdots,|\mathcal{A}_i|\}. \tag{3–33}$$

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

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

其中 $\boldsymbol{v}^{a_i(k-1)}$ 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

$$P^l_i(x)=\begin{cases}\dfrac{e^{\frac{1}{\tau}x^l}}{\sum_{x^m>0}e^{\frac{1}{\tau}x^m}}I\{x^l>0\}, & \text{if } \mathbf{1}^{\mathrm{T}}[x]^+>0,\\ \dfrac{1}{\mathcal{A}_i}, & \text{if } \mathbf{1}^{\mathrm{T}}[x]^+=0.\end{cases}, \tag{3–35}$$

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

$\widetilde{R}^j_{\mathcal{M}_i}(0) \leftarrow 0$;

$k \leftarrow 1$;

while *true* **do**

  for $i = 1 : n_m$ **do**

    使用式（3–31）计算 $\widetilde{R}^j_{\mathcal{M}_i}(k)$;

    使用式（3–34）计算目标概率分布 $\tilde{p}_i(k)$;

    根据 $\tilde{p}_i(k)$ 随机选择目标 $a_i(k)$;

    $k \leftarrow k + 1$;

  end

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 $p_i(k)$

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

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

根据式（3–36）可求得 $p_i(k)$ 的解析解形式为

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

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

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

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

$k \leftarrow 1$;

while *true* **do**

  随机选择一个智能体 $\mathcal{M}_i$;

  $n_i = |\mathcal{A}_i|$; // 使用 SAP 算法

  (或者）确定 n_i 的值，$1 \le n_i < |\mathcal{A}_i|$; // 使用 sSAP 算法

  从 $\mathcal{A}_i$ 中随机选择 n_i 个目标（包括 $a_i(k-1)$）;

  使用式（3–37）计算 $p_i(k)$;

  根据 $p_i(k)$ 随机选择目标 $a_i(k)$;

  $k \leftarrow k+1$;

end

3.4.2 任务交易

3.4.1小节的单方决策算法可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，本节设计了纳什均衡选择算法，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效用，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 $a = (a_1, \cdots, a_i, \cdots, a_j, \cdots, a_{n_m})$，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 $a' = (a_1, \cdots, a'_i, \cdots, a'_j, \cdots, a_{n_m}), a'_i = \mathcal{T}_l, a'_j = \mathcal{T}_k$。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，

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

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小节的单方决策算法，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。

3.5 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）。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。其中 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计算出的最优值中的最大值作为参照值，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。

3.5.1 优化结果与优化速度分析

图3–1至图3–4展示的分别是 10 对 10、50 对 50、50 对 40、100 对 100 四种场景下包括本章使用的三种算法在内的七种算法的单次任务分配的优化过程分析。

由图中可看出，在问题规模较小时，如 10 对 10 场景下，集中式启发式算法往往能够收敛到最优解，而使用势博弈模型的三种分布式算法表现较差

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

$k \leftarrow 1$;

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

while *!satisfied* **do**

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

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

  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

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

  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

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

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

    if $deltaU < 0$ **then**

      TraderFlag$\leftarrow$ 1;// 如果交易使得效用上升，则交换任务

    else if $deltaU == 0$ **then**

      TraderFlag $\leftarrow$ rand$\{0, 1\}$;// 如果交易使得效用不变，则随机选择是否交换任务

    end

    if *TraderFlag* == 1 **then**

      tmp$\leftarrow a^*_{\text{trader}}$;

      $a^*_{\text{trader}} \leftarrow a_i$;

      $a_i^* \leftarrow$tmp;

      // 将交易双方的满意值重置，重新进行单向决策

      satisfied(i) $\leftarrow$ 0;

      satisfied(trader)$\leftarrow$ 0;

    end

  end

  $k \leftarrow k + 1$;

end

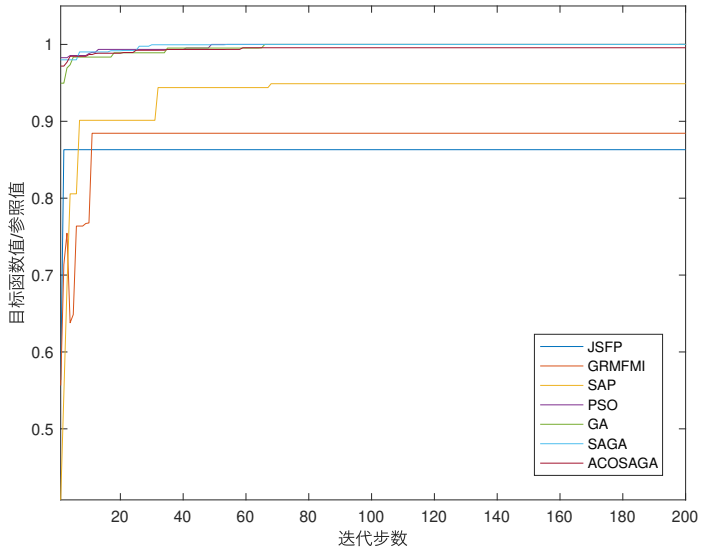


图 3–1 10 对 10 场景算法收敛过程比较

Figure 3–1 Comparison of convergence in the 10 vs 10 scenario

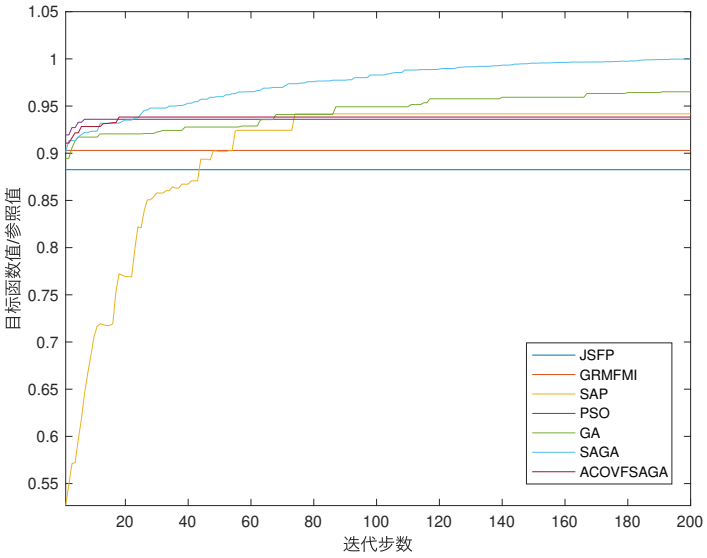


图 3–2 50 对 50 场景算法收敛过程比较

Figure 3–2 Comparison of convergence in the 50 vs 50 scenario

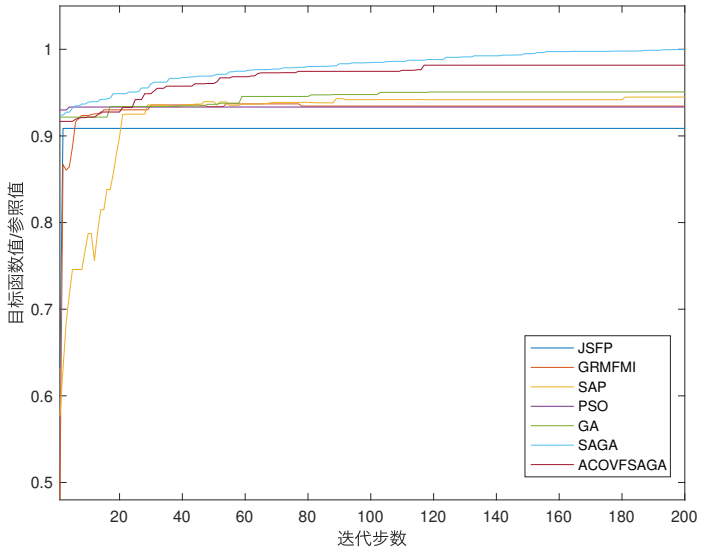


图 3–3 50 对 40 场景算法收敛过程比较

Figure 3–3 Comparison of convergence in the 50 vs 40 scenario

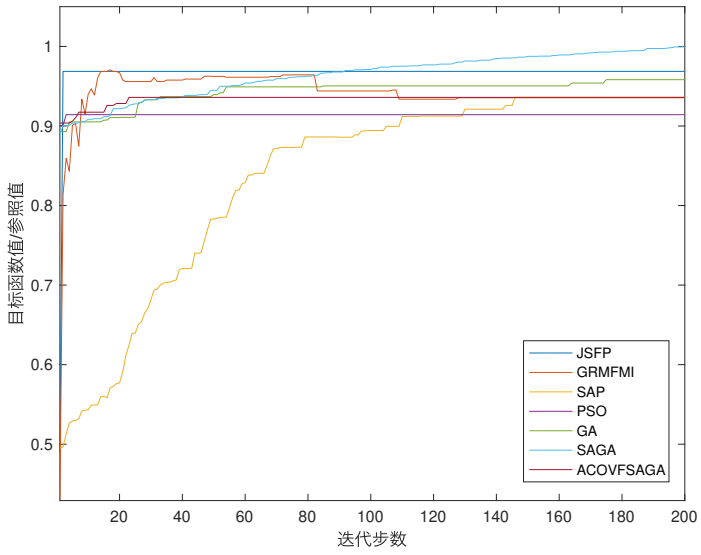


图 3–4 100 对 100 场景算法收敛过程比较

Figure 3–4 Comparison of convergence in the 100 vs 100 scenario

表 3–1 不同场景下算法性能对比表

Table 3–1 Comparison

算法名称	10 对 10 最优值	50 对 50 最优值	50 对 40 最优值	100 对 100 最优值
JSFP	0.8631	0.8826	0.9087	0.9687
GRMFMI	0.8844	0.9030	0.9345	0.9355
SAP	0.9489	0.9360	0.9640	0.9361
PSO	1	0.9030	0.9333	0.9142
GA	1	0.9651	0.9508	0.9582
SAGA	1	1	1	1
ACOSAGA	0.9957	0.9383	0.9816	0.9360

表 3–2 不同场景下算法性能对比表

Table 3–2 Comparison

算法名称	10 对 10 收敛时间 (s)	50 对 50 收敛时间 (s)	50 对 40 收敛时间 (s)	100 对 100 收敛时间 (s)
JSFP	0.1084	2.8038	2.2428	10.2660
GRMFMI	0.1180	2.6990	2.2611	10.611
SAP	0.0236	0.1783	0.1440	0.1269
PSO	0.6274	0.7506	0.7718	1.1059
GA	0.1974	0.5684	0.6785	1.0526
SAGA	0.1820	0.4831	0.5509	0.9783
ACOSAGA	0.8480	1.0003	1.1950	1.4667

3.5.2 优化性能稳定性分析

3.5.1展示的是一次分配下的算法优化过程图。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。图3–5至图3–8展示的是每种场景下算法运行 100 次得到的最优解函数值，同样经过幅值处理后，所绘制的箱图。

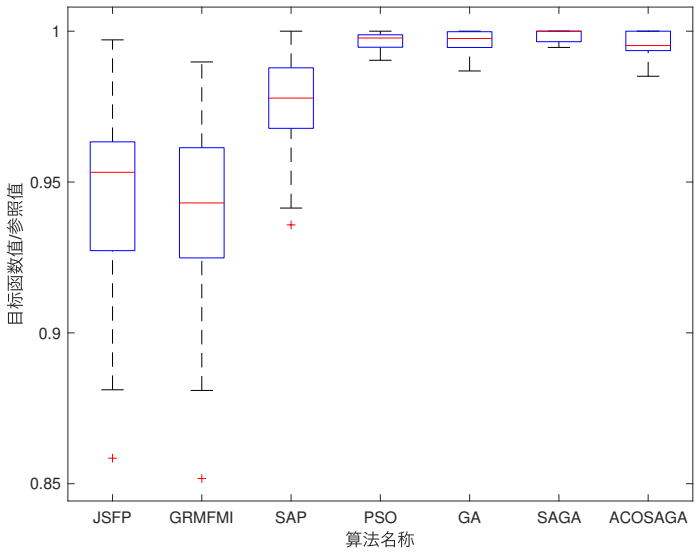


图 3–5 10 对 10 场景算法优化性能稳定性比较

Figure 3–5 Comparison of performance stability in the 10 vs 10 scenario

3.5.3 算法时间与规模关系分析

3.6 本章小结

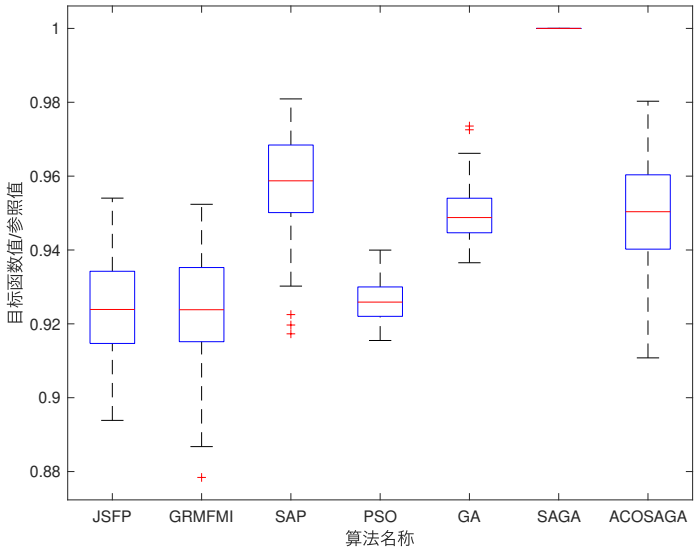


图 3–6 50 对 50 场景算法优化性能稳定性比较

Figure 3–6 Comparison of performance stability in the 50 vs 50 scenario

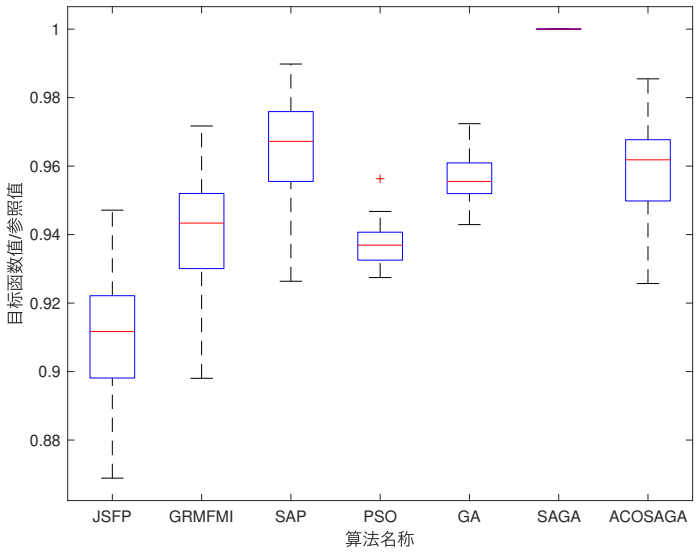


图 3–7 50 对 40 场景算法优化性能稳定性比较

Figure 3–7 Comparison of optimization stability in the 50 vs 40 scenario

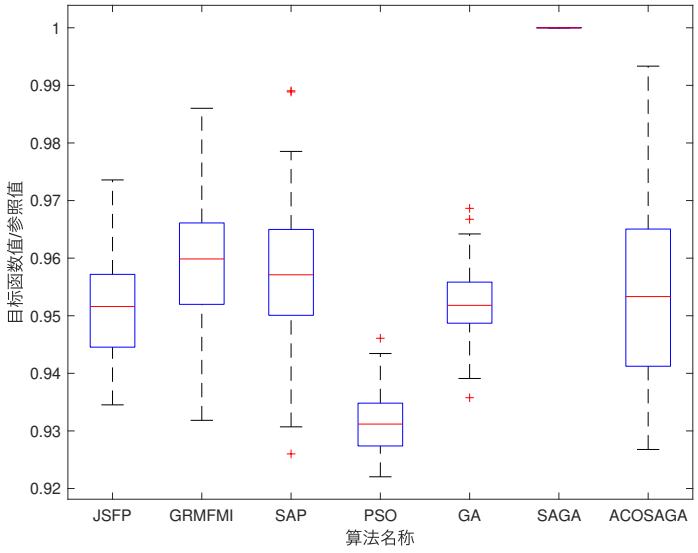


图 3–8 100 对 100 场景算法优化性能稳定性比较

Figure 3–8 Comparison of performance stability in the 100 vs 100 scenario

第四章　基于享乐联盟博弈的分布式任务分配

4.1　引言

使用享乐联盟博弈模型（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　享乐联盟博弈模型

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 $\mathcal{M}=\{\mathcal{M}_1,\cdots,\mathcal{M}_{n_m}\}$，任务集合为 $\mathcal{T}=\{\mathcal{T}_0,\mathcal{T}_1,\cdots,\mathcal{T}_{n_t}\}$。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 $U_i(\mathcal{T}_0,n)=0,\forall\mathcal{M}_i\in\mathcal{M},0\le n\le n_m$。设当前分配解为 $a=(a_1,a_2,\cdots,a_{n_m})$，全局效用函数被定义为

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

其中

$$p_j=\sum_{\mathcal{M}_i\in\mathcal{M}}I\{a_i=\mathcal{T}_j\},\ \mathcal{T}_j\in\mathcal{T}. \tag{4–2}$$

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 $\mathcal{X}=\mathcal{T}\times\{1,\cdots,n_m\}$。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

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

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 $\Pi = \{S_0, S_1, \cdots, S_{n_t}\}$，其中元素联盟集合 $S_j \subseteq \mathcal{M}, j = 1, \cdots, n_m$ 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。由于每个智能体智能选择一个任务，因此 $\cup_{j=0}^{n_t} S_j = \mathcal{M},\ S_i \cap S_j = \varnothing,\ i \neq j$。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 $S_{\Pi(i)} = \{S_j \in \Pi | \mathcal{M}_i \in S_j\}$。

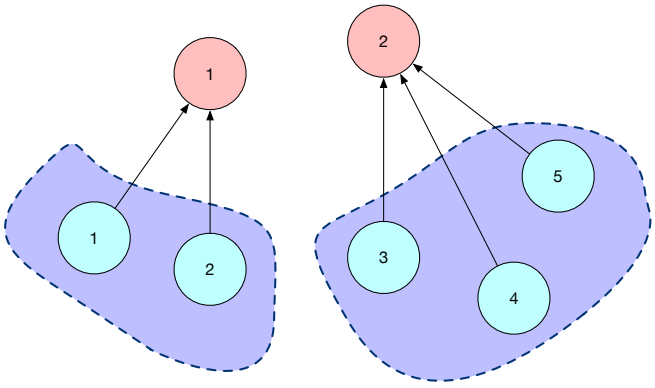


图 4–1 联盟划分

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

$$(t_{\Pi(i)}, |S_{\Pi(i)}|) \succeq_i (t_j, |S_j \cup \{a_i\}|), \quad \forall S_j \in \Pi, \tag{4–4}$$

则划分 Π 被称为是纳什均衡划分.

定义中 $S_j \cup \{a_i\}$ 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 面向任务分配的享乐联盟博弈模型设计

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 $\mathcal{T}_j \in \mathcal{T} \setminus \{\mathcal{T}_0\}$，

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

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

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

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 $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{M}}, \mathcal{T}, \succeq_i)$。设 $\mathcal{M}_r \in \widetilde{\mathcal{M}}, \mathcal{M}_r \notin \mathcal{M}$，则 $\widetilde{\mathcal{M}} = \mathcal{M} \cup \{\mathcal{M}_r\}$。

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

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

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 $\widetilde{\Pi}$ 下，所有的智能体都会满足 $\Delta_{\widetilde{\Pi}(i)} \geq 0$，因此 $\widetilde{\Pi}$ 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。 □

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

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

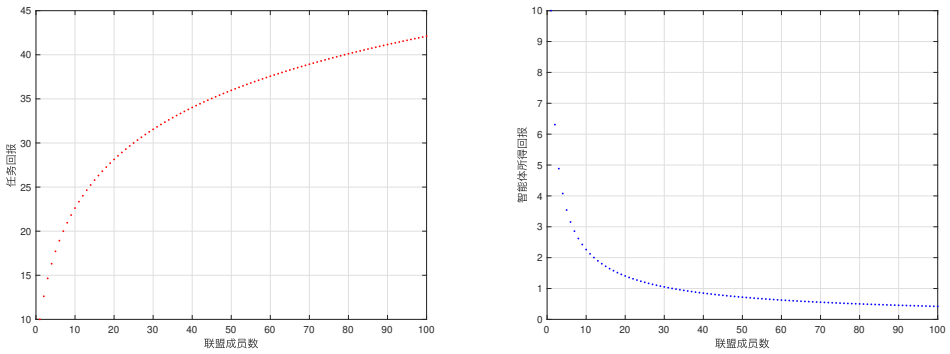
其中 $r(\mathcal{T}_j, |S_j|)$ 为导弹和联盟成员一起攻击目标 $\mathcal{T}_j$ 可获得的回报，并假设联盟所有成员平分任务回报。在空战场景中，攻击同一个目标的导弹数量越多，击中的目标的概率越大，但当数量超过一定界限时，再增加导弹数量对于目标的攻击起到的作用很小，即随着联盟成员的数量增加，导弹所得到的边际回报在递减。因此可定义任务回报函数 $r(\mathcal{T}_j, |S_j|)$ 为

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

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

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

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。



a) 任务回报函数　　b) 智能体回报函数

a) Task Reward Function　　b) Agent Reward Function

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 $\widetilde{U}_i(\mathcal{T}_j, |S_j|)$ 为

$$\widetilde{U}_i(\mathcal{T}_j, |S_j|) = \begin{cases} U_i(\mathcal{T}_j, |S_j|), & \text{if } |S_j| \le b_{\max}^{(j)}, \\ 0, & \text{otherwise} \end{cases} \tag{4–11}$$

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 基于享乐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。

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

$k \leftarrow 1$;

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

$\text{CoWorkerNums}_j(0) \leftarrow \sum_{\mathcal{M}_i \in \mathcal{M}} I\{a_i(0) = \mathcal{T}_j\}$;

while *true* **do**

  随机选择一个智能体 $\mathcal{M}_i$;

  $n_i = |\mathcal{A}_i|$;

  for $j = 1 : n_t$ **do**

    根据式（4–10）计算 $U_i(\mathcal{T}_j, |S_j| + 1)$;

  end

  根据式（3–37）计算 $p_i(k)$;

  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

  $a_i(k) \leftarrow \mathcal{T}_l$;

  $\text{CoWorkerNums}_{a_i(k-1)}(k) \leftarrow \text{CoWorkerNums}_{a_i(k-1)}(k) - 1$;

  $\text{CoWorkerNums}_{a_i(k)}(k) \leftarrow \text{CoWorkerNums}_{a_i(k)}(k) + 1$;

  $k \leftarrow k + 1$;

end

4.4.2 分布式互斥算法

本节将提出 HCG 模型下的另一种决策算法，称为互斥算法（Mutual Exclusion Algorithm, MEA）。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 MEA 的分布式决策算法，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 MEA 的实现。

每个智能体在决策过程中都有自己的划分方式 $\boldsymbol{\Pi}^i$；变量 satisfied 是一个布尔变量，用于表示智能体是否对当前划分 $\boldsymbol{\Pi}^i$ 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 之间平均分布的随机变量，它会在划分 $\boldsymbol{\Pi}$ 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 $\boldsymbol{\Pi}^i$，检查在该划分下，假设其他智能体不改变联

算法 **4–2** MEA 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 Π

// 初始化参数

1 satisfied $\leftarrow false$;

2 evolved$^i \leftarrow 0$;// 智能体更新决策次数

3 stamp$^i \leftarrow 0$;// 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** $(\mathcal{T}_{j^*}, |S_{j^*}|) \succ_i (\mathcal{T}_{\Pi^i(i)}, |S_{\Pi^i(i)}|)$ **then**

9 智能体 $\mathcal{M}_i$ 加入 S_{j^*}，更新划分 Π^i;

10 evolved$^i \leftarrow$ evolved$^i + 1$;

11 stamp$^i \leftarrow$ rand[0, 1];

12 **end**

13 satisfied $\leftarrow$ *true*;

14 **end**

15 智能体 $\mathcal{M}_i$ 向邻居发送信息 $I^i = \{\text{evolved}^i, \text{stamp}^i, \Pi^i\}$，并从其邻居节点获取信息 $I^k, \forall \mathcal{M}_k \in \mathcal{N}_i$;

16 构造信息集 $\mathcal{I}^i = \{I^i\} \cup \{I^k, \forall \mathcal{M}_k \in \mathcal{N}_i\}$;

// 运行互斥算法

17 $\{\text{evolved}^i, \text{stamp}^i, \Pi^i\}$, satisfied $\leftarrow$ DMEA($\mathcal{I}^i$);

18 **end**

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 $I^i = \{\text{evolved}^i, \text{stamp}^i, \Pi^i\}$，包括自己的划分方式，划分更新次数和更新时间戳。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

算法 **4–3** 分布式互斥算法（DMEA)

Input: $\mathcal{I}^i$

Output: $\{\text{evolved}^i, \text{stamp}^i, \Pi^i\}$, 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 $\text{evolved}^i \leftarrow \text{evolved}^k$;

5 $\text{stamp}^i \leftarrow \text{stamp}^k$;

6 $\Pi^i \leftarrow \Pi^k$;

7 satisfied $\leftarrow false$;

8 **end**

9 **end**

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

4.4.3.1 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

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

□

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。

4.4.3.2 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，可能存在着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 $\boldsymbol{\Pi}^i, \text{evolved}^i, \text{stamp}^i$），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。

4.4.3.3 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 $\mathcal{G}_{\text{HCG}}$ 的 PoA 指标上界为

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

其中

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

下面将针对前文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

$$U_{\mathcal{T}}(\mathcal{T}_j, p) = \sum_{\mathcal{M}_i \in S_j} U_i(\mathcal{T}_j, |S_j|). \tag{4–15}$$

结合全局效用函数的定义式（4–1）有

$$U_g(a) = \sum_{\mathcal{M}_j \in \mathcal{M}} U_i(a_i, p_i) = \sum_{\mathcal{T} \in \mathcal{T}} U_{\mathcal{T}_j}(\mathcal{T}_j, p) \tag{4–16}$$

若使用式（4–10）定义的智能体效用函数，使得任务效用函数 $r(\mathcal{T}_j, p)$ 满足关于智能体数量 p 单调递增的性质，则可以进一步限定 PoA 的上界，实际上可得到如下命题。

命题 4.1 设 HCG 模型 $\mathcal{G}_{\mathrm{HCG}}$ 中智能体效用函数 $U_i(\mathcal{T}_j, |S_j|)$ 定义为式（4–10），且 $\varepsilon_j = \varepsilon > 1$，则 $\mathrm{PoA}(\mathcal{G}_{\mathrm{HCG}})$ 满足

$$\mathrm{PoA}(\mathcal{G}_{\mathrm{HCG}}) \leq 1 + \eta'_{\mathrm{HCG}} \tag{4–17}$$

其中

$$\eta' = \log_{\varepsilon}(n_m + \varepsilon) - 1 \tag{4–18}$$

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

$$\boldsymbol{\Pi}^A \oplus \boldsymbol{\Pi}^B := \{S_0^A \cup S_0^B, S_1^A \cup S_1^B, \cdots, S_{n_t}^A \cup S_{n_t}^B\}, \tag{4–19}$$

由于 $\cup_{j=0}^{n_t} S_j^A = \cup_{j=0}^{n_t} S_j^B = \mathcal{M}$，因此可能存在一个智能体 $\mathcal{M}_i$ 在 $\boldsymbol{\Pi}^A \oplus \boldsymbol{\Pi}^B$ 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。

由式（4–10）定义的智能体效用函数，使得任务效用函数 $r(\mathcal{T}_j, p)$ 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

$$U_g(\boldsymbol{\Pi}^A) \leq U_g(\boldsymbol{\Pi}^A \oplus \boldsymbol{\Pi}^B). \tag{4–20}$$

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

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

其中 $\mathcal{T}^-$ 为在 $\boldsymbol{\Pi}^*$ 中满足 $S_j^* = \varnothing, S_j^{\mathrm{opt}} \neq \varnothing$ 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 $\mathcal{T}^- = \varnothing$，所以式（4–21）实际上等于

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

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

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

若令 $x = |S_j^*|, y = |S_j^{\text{opt}}|$，并代入 $\varepsilon_j = \varepsilon > 1$，由于

$$
\frac{\log_\varepsilon(x + y + \varepsilon - 1)}{\log_\varepsilon(x + \varepsilon - 1)} \le \frac{\log_\varepsilon(x + n_m + \varepsilon - 1)}{\log_\varepsilon(x + \varepsilon - 1)} \qquad (4\text{–}24)
$$

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

$$
\frac{\log_\varepsilon(x + y + \varepsilon - 1)}{\log_\varepsilon(x + \varepsilon - 1)} \le \frac{\log_\varepsilon(n_m + \varepsilon)}{\log_\varepsilon(\varepsilon)} = \log_\varepsilon(n_m + \varepsilon),\ 1 \le x \le x_m,\ 1 \le y \le n_m, \qquad (4\text{–}25)
$$

因此

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

而由式（4–20）可得

$$
U_g(\Pi^{\text{opt}}) \le U_g(\Pi^{\text{opt}} \oplus \Pi^*). \qquad (4\text{–}27)
$$

结合式（4–26）和式（4–27）即可得

$$
\begin{aligned}
U_g(\Pi^{\text{opt}}) &\le \log_\varepsilon(n_m + \varepsilon) U_g(\Pi^*) \qquad (4\text{–}28) \\
\text{PoA}(\mathcal{G}_{\text{HCG}}) &\le 1 + \left[\log_\varepsilon(n_m + \varepsilon) - 1\right] \\
&\triangleq 1 + \eta'. \qquad (4\text{–}29)
\end{aligned}
$$

□

4.5 仿真分析与对比

4.5.1 算法性能对比

4.6 本章小结

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 $\mathcal{G} =< N, \Omega, \{S_i, U_i\}_{i \in N}, q >$，其中 $N = \{1, 2 \cdots, n\}$ 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 $N = \{1, 2 \cdots, i, \cdots, n_m\}$，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。本文后续将通过仿真实验确定时间窗口长度 w 的最优值。

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 $a_{\text{M}} > g_{\max}$，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 $a_{k+1}(0) = \{a_{k+1}^1(0), a_{k+1}^2(0), \cdots, a_{k+1}^{n_m}(0)\}$

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

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 事件触发重分配机制

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，导弹集群往往会形成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

$$R_{\text{comm}} \geq 2V_{\max}T_k\Delta t + R_{\text{col}}, \tag{5–2}$$

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。

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

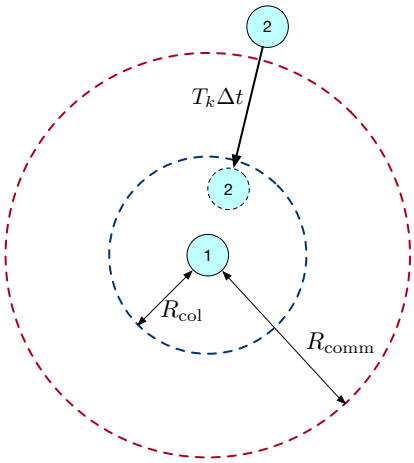


图 5–1 不可规避冲突

Figure 5–1 Illustration of an undetectable collision

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。

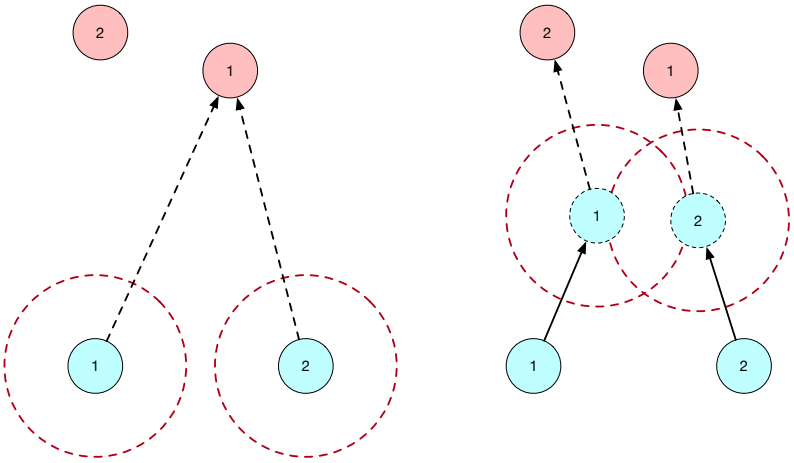


图 5–2 非连通场景下分配冲突消解示意图

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

5.4.2 目标数量变化

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（5–1）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会被保留，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 $\mathcal{T}_w$ 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 动态任务分配系统流程

结合5.3节和5.4节的内容，图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。

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 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。

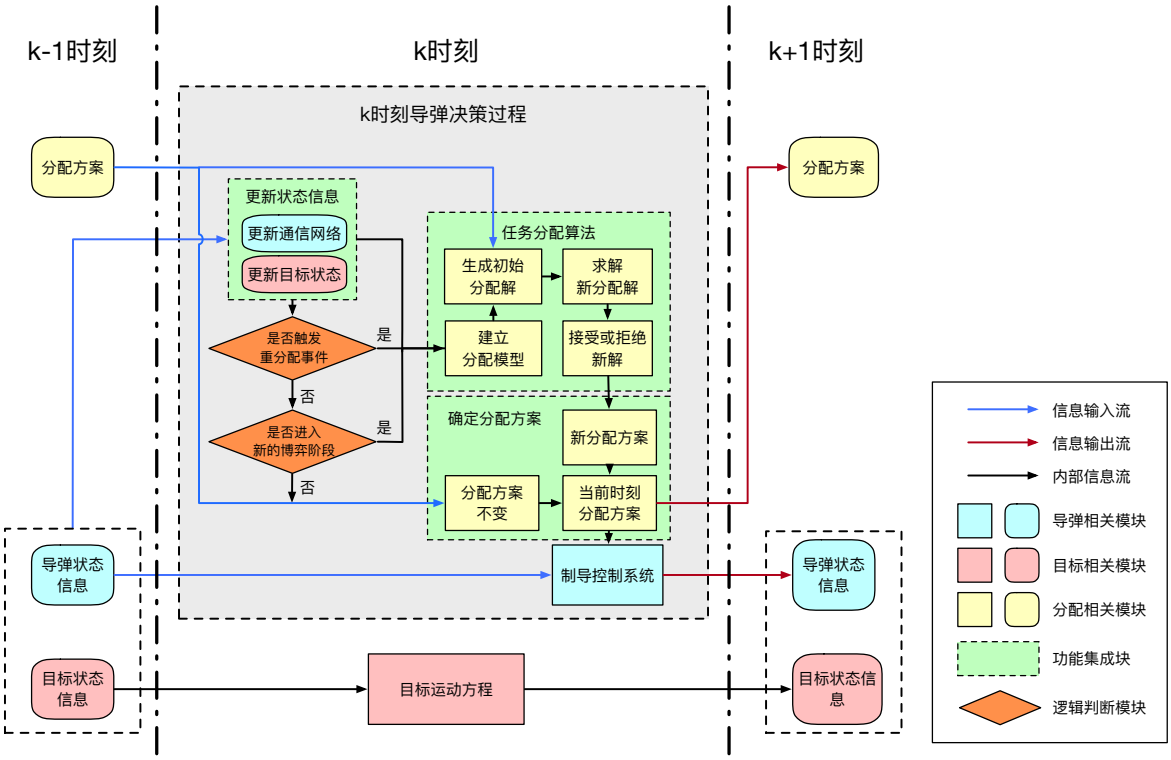


图 5–3 导弹任务分配系统框架示意图

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

（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态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，计算制导制令，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。

5.6 仿真对比与分析

5.7 本章小结

第六章　总结与展望

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

个人简历

基本情况

某某，yyyy 年 mm 月生于 xxxx。

教育背景

- yyyy 年 mm 月至今，上海交通大学，博士研究生，xx 专业
- yyyy 年 mm 月至 yyyy 年 mm 月，上海交通大学，硕士研究生，xx 专业
- yyyy 年 mm 月至 yyyy 年 mm 月，上海交通大学，本科，xx 专业

研究兴趣

LaTeX 排版

联系方式

- 地址：上海市闵行区东川路 800 号，200240
- E-mail: xxx@sjtu.edu.cn