



中南大學  
CENTRAL SOUTH UNIVERSITY

# 本科毕业设计(论文)

GRADUATION DESIGN (THESIS)

题 目: 考虑边缘计算的无人机航迹  
规划及任务调度研究

学生姓名: \_\_\_\_\_

指导老师: \_\_\_\_\_

学 院: \_\_\_\_\_

专业班级: \_\_\_\_\_

本科生院制

2022 年 5 月

# 考虑边缘计算的无人机航迹规划及任务调度研究

## 摘要

近年来，随着无人机技术的不断发展，无人机在军事和民用领域都得到了广泛的应用。为了推广无人机在城市环境下的应用，本文针对无人机航迹规划及任务调度问题进行研究，在对该问题进行详细描述及分析的基础上，基于边缘计算架构将其分解为了任务计算资源调度问题、多无人机航迹规划及任务分配问题共两个子问题，分别建立了相应的数学模型，并通过设计基于 ASA 算法的静态任务计算资源调度算法、基于 FD 算法的动态任务计算资源调度算法、基于 RRT\*-Connect 算法的静态场景下的无人机航迹规划算法、基于 A\* 算法的动态场景下的无人机航迹规划算法以及基于 TSAVN 算法的多无人机任务分配算法实现对上述两个子问题的求解。最后，对于各个子问题分别设计了相应的仿真实验，验证了本文所提出的算法在求解考虑边缘计算的无人机航迹规划及任务调度各个子问题的有效性及优越性。

仿真实验结果表明：在任务计算资源调度问题中，ASA 算法能够在考虑传输时延等情况下生成较优的计算资源调度方案，同时相较于其他对比算法，其生成的计算资源调度方案与算法的运行时间都有着显著的优越性和鲁棒性；而 FD 算法相较于其他对比算法，虽然其生成的计算资源调度方案较差，但其运行时间能够满足问题场景下所需要的实时性。在多无人机航迹规划及任务分配问题中，RRT\*-Connect 算法的能够在城市密集障碍物静态场景下快速规划出优良的无人机飞行航迹，相较于其他对比算法，其规划的飞行航迹有着更好的性能和鲁棒性；A\* 算法能够根据实时获取的障碍物信息对无人机飞行航迹进行快速修订，相较于其他对比算法，其生成的飞行航迹与运行时间都有着显著的优越性和鲁棒性；同时的 TSAVN 算法能够根据所给飞行航程信息与任务信息，在较短时间内生成无人机的收集型侦察任务分配方案，相较于其他对比算法，其生成的任务分配方案以及算法的运行时间都有着显著的优越性与鲁棒性。

关键词：多无人机 边缘计算 任务分配 航迹规划 资源调度 智能优化算法

# Research on trajectory planning and mission scheduling of UAV considering edge computing

## ABSTRACT

In recent years, with the continuous development of UAV technology, UAVs have been widely used in both military and civilian fields. In order to promote the application of UAVs in urban environments, this thesis investigates the UAV task scheduling and dynamic trajectory planning problem. Based on a detailed description and analysis of the problem, it is decomposed into two sub-problems based on edge computing architecture: the task computing resource scheduling problem, the multi-UAV task allocation and trajectory planning problem, and the single-UAV real-time obstacle avoidance problem. The corresponding mathematical models are established, and the above sub-problems are solved by designing the static mission computing resource scheduling algorithm based on ASA algorithm, the dynamic mission computing resource scheduling algorithm based on FD algorithm, the UAV trajectory planning algorithm in static scenes based on RRT\*-Connect algorithm, the UAV trajectory planning algorithm in dynamic scenes based on A\* algorithm and the multi-UAV mission allocation algorithm based on TSAVN algorithm based on A\* algorithm. Finally, the corresponding simulation experiments are designed for each subproblem to verify the effectiveness and superiority of the proposed algorithms in solving each subproblem of UAV task scheduling and dynamic trajectory planning considering edge computing.

The simulation experimental results show that in the task computational resource scheduling problem, the ASA algorithm is able to generate a better computational resource scheduling scheme resource information, considering the transmission delay. At the same time, its generated computational resource scheduling scheme and the running time of the algorithm are significantly superior and robust compared to other comparative algorithms. The FD algorithm, on the other hand, generates a poorer computational resource scheduling scheme compared to other comparative algorithms, but its running time can meet the real-time performance required in the problem scenario. In the multi-UAV mission assignment and trajectory planning problem, the RRT\*-Connect algorithm is able to quickly plan excellent UAV flight trajectories in

urban dense obstacle scenarios with better performance and robustness compared to other comparative algorithms; the A\* algorithm can quickly revise the UAV flight trajectory based on the obstacle information acquired in real time, which has significant superiority and robustness in terms of generated flight trajectory and running time compared to other comparative algorithms; meanwhile, the TSAVN algorithm is able to generate UAV collection-based reconnaissance mission assignment schemes in a short time based on the given flight distance information and mission information, which has significant superiority and robustness compared to other comparative algorithms in terms of the generated mission assignment schemes and the running time of the algorithm.

**Key words:** Multi-UAV Edge Computing Task Scheduling Trajectory Planning Resource Scheduling Intelligent Optimization Algorithm

## 目录

第 1 章 绪论.....	1
1.1 研究背景及意义 .....	1
1.2 国内外研究现状 .....	4
1.2.1 面向无人机网络的边缘计算研究 .....	4
1.2.2 无人机航迹规划技术研究现状 .....	6
1.2.3 无人机任务分配技术研究现状 .....	7
1.3 研究内容.....	8
1.4 技术路线.....	10
第 2 章 考虑边缘计算的无人机航迹规划及任务调度问题 .....	11
2.1 问题描述及场景设置 .....	11
2.1.1 问题描述 .....	11
2.1.2 场景设置 .....	15
2.2 模型假设.....	16
2.3 符号定义.....	16
2.4 数学模型.....	17
2.4.1 任务计算资源调度模型.....	17
2.4.2 多无人机航迹规划及任务分配模型 .....	19
2.5 本章小结.....	19
第 3 章 考虑边缘计算的无人机航迹规划及任务调度算法 .....	20
3.1 算法设计应用流程 .....	20
3.2 面向边缘计算的任务计算资源调度算法 .....	21
3.2.1 基于 ASA 的静态场景下的任务计算资源调度算法 .....	21
3.2.2 基于 FD 的动态场景下的任务计算资源调度算法 .....	24
3.2.3 算法流程图 .....	25
3.3 多无人机航迹规划及任务分配算法 .....	27

3.3.1	基于 RRT*-Connect 的静态场景下的无人机航迹规划算法设计 .....	28
3.3.2	基于 A* 的动态场景下的无人机航迹规划算法设计 .....	30
3.3.3	基于 TSAVN 的多无人机任务调度分配算法设计 .....	35
3.3.4	多无人机航迹规划及任务分配算法流程图 .....	43
3.4	本章小结 .....	44
第 4 章	算法仿真实验 .....	45
4.1	任务计算资源调度算法仿真实验 .....	45
4.1.1	参数设置 .....	45
4.1.2	算法性能对比实验结果 .....	46
4.2	静态场景下的无人机航迹规划算法仿真实验 .....	47
4.2.1	仿真参数设置 .....	48
4.2.2	仿真实验结果 .....	48
4.2.3	算法性能对比实验结果 .....	49
4.3	动态场景下的无人机航迹规划算法仿真实验 .....	50
4.3.1	仿真实验参数 .....	50
4.3.2	仿真实验结果 .....	51
4.3.3	算法性能对比实验结果 .....	51
4.4	多无人机任务分配算法仿真实验 .....	52
4.4.1	实验参数设置 .....	52
4.4.2	仿真实验结果 .....	53
4.4.3	算法性能对比实验结果 .....	53
4.5	本章小结 .....	55
第 5 章	总结与展望 .....	56
5.1	总结 .....	56
5.2	展望 .....	56
致谢	.....	58
参考文献	.....	59

## 第1章 绪论

### 1.1 研究背景及意义

5G (5th-Generation) 通信技术，即第五代移动通信技术，为4G通信技术的进一步发展的新技术。由于5G通信技术与此前的4G通信技术相比，有着高速率、宽带宽、高可靠、低时延等优势，能够满足在自动驾驶、远程医疗等方面的应用的要求，同时在如今新基建的浪潮下，5G技术与AI、物联网等技术应用得以如雨后春笋般迅猛发展，带来了信息流量及运算算力等方面的巨大变化。

如图1-1所示，根据2021年通信业统计公报的数据，截止2021年底，移动互联网产生的流量比上年增长33.9%，共达2216亿GB，其中手机上网流量就已达到2125亿GB，在总流量中占比高达95.9%；与此同时物联网用户已达13.99亿户，比上年增长23.3%，其中应用于智慧公共视野、智能制造、智慧交通的终端用户占比分别达22.4%、18.1%、15.6%。从以上数据可知，数据流量和物联网还在如喷涌般迅速发展，使得数据的分析和处理也有着极大且还在快速增长的需求。

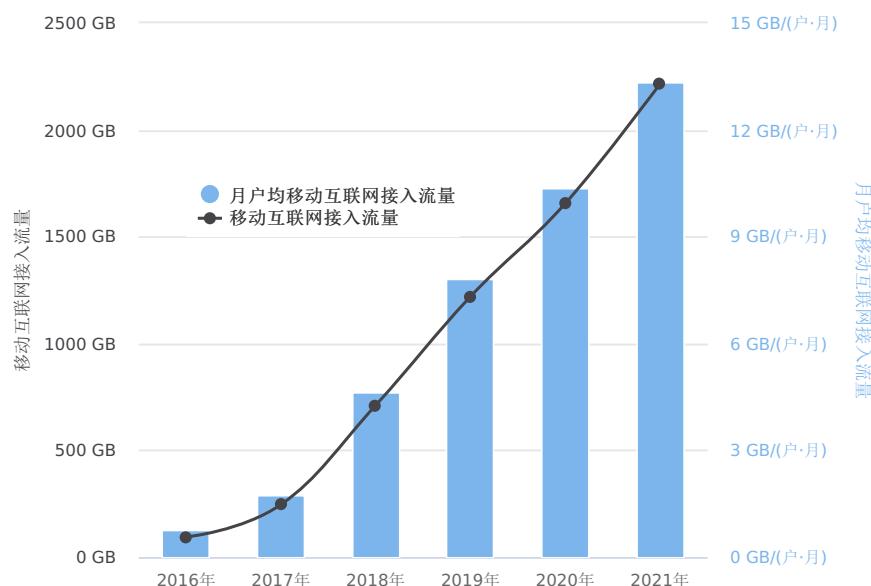


图1-1 2016-2021年移动互联网流量及月户均流量(DOU)增长情况(来源：2021年通信业统计公报)

5G通信技术及其他技术的迅猛发展，促使了用户的边缘数据量快速增长。随着边缘数据量不断增大，在数据传输时存在通信占用带宽过多、处理时占用负载较大、储存占用空间过多、边缘设备性能使用率较低等问题。为了解决这些问题，研究学者提出了

将一部分任务分配至边缘设备中进行处理的边缘计算技术，能够应用于在靠近用户或数据源的边缘设备中，为其提供网络、计算、储存等所需服务。通过这一技术降低了数据中心服务器的荷载，并且为用户提供了高稳定和低时延的良好使用体验；延伸了数据中心-用户的计算框架并形成了数据中心-边缘设备-用户的新型框架；还有助于数据、算力、成本的灵活布置，有助于对时延、稳定性等性能要求较高的需求的满足。

UAV (Unmanned Aerial Vehicle, 无人机) 是一种机上无需载乘任何操作人员、能够自主控制或由机组人员远程遥控的，可搭载一定设备且能够重复利用的无人飞行平台及设备。近些年，无论在科学的研究还是在应用市场中，针对 UAV 的研究和应用受到了越来越广泛的关注，其民用市场规模也逐年递增（见图 1-2）。同时，随着相关技术的快速发展，UAV 的自身性能得到了迅速增强，与传统的载人飞行器相比，UAV 的性价比更高，且具有体积小、重量轻、灵活性好、机动性强，易部署、成本低、广视角、隐蔽性好等诸多优点。

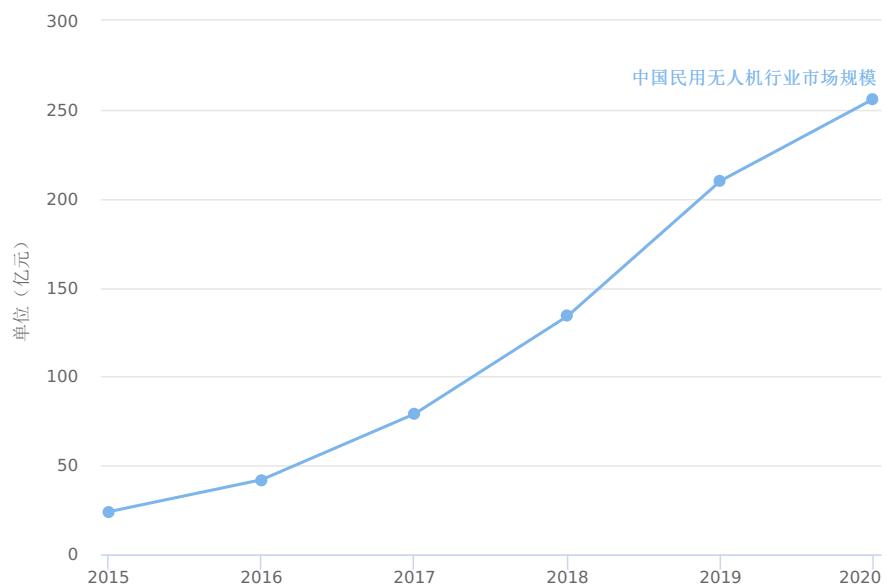


图 1-2 2015-2020 年中国民用无人机市场规模（来源：前瞻产业研究院）

UAV 的上述优点加速了其在相关民用和军事领域的应用，如图 1-3 所示，UAV 目前常应用于在铁路故障巡检、交通调查、物流配送<sup>[1]</sup>、灾难响应<sup>[2]</sup>、森林火情监视等民用领域，以及集群作战<sup>[3]</sup>、对地目标打击等军事领域。目前美军出台了多个无人机项目，例如低成本无人机集群技术项目、小精灵项目、进攻性集群使能战术项目等，大力推动发展各类场景下人机协同或集群作战的能力，走在了世界前列<sup>[3]</sup>。但目前无人机也存在着单机计算能力较弱的缺点<sup>[4]</sup>，因此，将边缘计算技术引入无人机应用领域已经成为无人机的发展方向之一。

我国“十四五”规划已明确提出要加快将中国建设成为交通强国，UAV 作为一种



图 1-3 集群无人机应用场景

新型的交通运输方式，已经成为“交通强国”战略中的重要一环。将多无人机合理有效地应用于交通领域将有利于促进数字交通体系精细化、信息化，加快智慧交通建设立体化、快速化，以及推动交通运输系统高效化、智能化。作为 UAV 关键应用之一，面向城市低空环境的多无人机侦察任务，存在着因设备数量多、基站带宽有限等原因而带来的如通信延迟较大、计算性能有限等问题；同时城市低空环境与其他环境相比，有着空间更为狭窄、障碍数量更多且密集，人员活动更为密集等特点，这也对无人机的航迹规划技术在实时性和避障性提出了更高的要求。以上问题使得 UAV 应用目前还远远达不到“交通强国”战略中的应用需求，而这势必会阻碍我国交通建设迈向更高水平及质量。

基于无人机的边缘计算研究是将边缘计算的架构应用于无人机平台，使得无人机作为边缘端既能够将一定的任务卸载到位于地面基站的服务器中，也可以将一定的任务卸载至其余空闲无人机中的研究，该方法既能够解决数据中心的集中式计算带来的通信占用带宽过多、处理时占用负载较大、储存占用空间过多、边缘设备性能使用率较低等问题，还能够充分利用现有设备的算力，使得设备整体利用率上升，便于控制成本。根据无人机是作为用户节点还是边缘服务器，其可被分为两种不同的场景：一种是无人机作为用户节点，地面基站为其提供计算服务支持，即面向无人机网络的边缘计算场景，如图 1-4a 所示；另一种是在无人机上装备边缘计算服务器，为地面网络提供计算服务，即面向地面网络的无人机边缘计算场景，如图 1-4b 所示。

因此，本文提出面向边缘计算的无人机航迹规划及任务调度方法，以解决在城市低

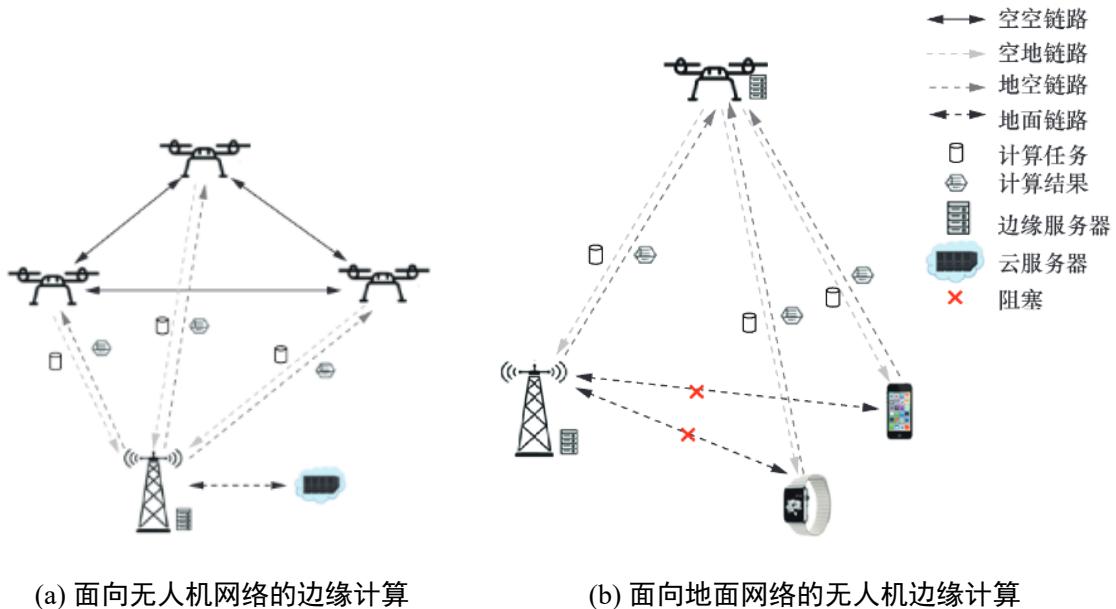


图 1-4 基于无人机的边缘计算研究场景

空环境下的多无人机航迹规划及任务调度任务中的局限性，进一步提高系统的性能使用率、降低运算成本，有助于推广 UAV 在城市低空环境下的可应用性。

## 1.2 国内外研究现状

### 1.2.1 面向无人机网络的边缘计算研究

在面向无人机网络的边缘计算研究的场景中，多架无人机联合执行任务，无人机之间通过空空链路进行连接。无人机边缘端将图像分析等计算密集型任务卸载到位于地面基站的地面端服务器上，地面端服务器完成任务处理后将结果返还给无人机，通过这种方式可以大大降低无人机的能耗和任务的处理时延，从而延长无人机的续航时间，并提升任务的完成效率。

在面向无人机网络的边缘计算研究中，通常会采用三种不同的架构，分别是将无人机作为边缘中转设备的用户设备-无人机-地面基站 3 层架构、Chen et al.<sup>[5]</sup>提出的将无人机作为终端设备的无人机-边缘服务器-云服务器 3 层架构，以及 Cheng et al.<sup>[6]</sup>提出的将无人机既作为终端设备也作为边缘设备的空地一体移动边缘架构。到目前为止，面向无人机网络的边缘计算研究在不同架构下所应用的方法主要分为以下三种：博弈论方法、运筹学方法、智能算法，如表 1-1 所示。

从博弈论角度出发, Messous et al.<sup>[7]</sup>根据纳什均衡提出了一种分布式算法, 构建了一个能耗和时延的加权和的代价函数作为系统收益指标, 通过优化卸载决策降低能量消耗, 减少执行时延, 从而获得最小化代价函数。彭维平等<sup>[8]</sup>通过提出一个基于博弈理论

表 1-1 面向无人机网络的边缘计算研究总结

类型	方法	优缺点
博弈论方法	纳什均衡 <sup>[7]</sup>	算法计算快、可靠性强，适用于求解大规模问题场景；容易限于局部最优解，解质量可能较差。
	博弈理论 + 纳什均衡 <sup>[8]</sup>	
运筹学方法	匈牙利算法 <sup>[9]</sup>	考虑了无人机位置与延迟的关系，贴近真实场景；实时性强，可用于动态调度；但问题规模较大时计算时间较久。
	排队论方法 <sup>[10]</sup>	
智能算法	交替优化 + 逐次凸逼近 <sup>[11]</sup>	
	模拟退火 + 粒子群算法 <sup>[12]</sup>	算法步骤简单易懂；能在可接受时间内，得出一个较优解；容易限于局部最优解，解质量可能较差。
	Q-Learning 强化学习算法 <sup>[13]</sup>	
	强化学习 + 差分演化算法 <sup>[14]</sup>	

和纳什均衡的自适应任务卸载方案来实现最小化能耗和时延的加权和。当电池能量较为充足时，可以通过增加时延的权重来提升用户体验；而当电池容量不足时，可以通过增加能耗的权重来减少无人机的能耗。

从运筹学角度出发，Kim et al.<sup>[9]</sup>使用匈牙利算法提出了最优的任务-无人机-移动边缘服务器匹配方法，最小化了能量消耗和处理时延的加权和，代价函数的权值可以根据用户对 QoS 的不同要求灵活改变。模拟结果显示，提出的算法相较于基于距离的算法具有更好的性能。Zhang et al.<sup>[10]</sup>构建了一种包括一个集中式顶部无人机和一群分布式底部无人机的场景，利用随机几何和排队理论，得到了闭环解的最优响应时延。Cao et al.<sup>[11]</sup>讨论了单个无人机从初始位置飞往最终位置的过程中，将计算任务卸载到沿途 5 个地面基站的场景。在满足无人机最大速度限制和地面基站计算能力有限的情况下，采用交替优化和逐次凸逼近技术联合优化无人机路径和卸载决策方案，

由于智能算法能够在较短时间内获得较好的解，因此不少学者在解决该问题时选择了该种算法。Zhu et al.<sup>[12]</sup>考虑了在不同时延约束下多个边缘服务器为单个无人机提供计算卸载服务的问题，采用拉格朗日乘子法优化数据传输速率，采用基于模拟退火的粒子群优化算法解决任务分配，从而满足不同业务的服务质量 (quality of service, QoS) 要求。Kim et al.<sup>[13]</sup>还提出了一种基于强化学习的边缘辅助无人机计算卸载方法，这种方法同时考虑了能量效率和任务处理时间，利用 Q-Learning 来解决无人机与任务集群的

匹配和寻找最优的边缘服务器这两个问题。Yang et al.<sup>[14]</sup>提出了一种基于差分演化的多无人机部署机制，实现无人机的负载均衡，并在此基础上使用一种基于深度强化学习的无人机任务调度算法，提高了无人机的任务执行效率。

### 1.2.2 无人机航迹规划技术研究现状

无人机航迹规划技术是指在指定的场景下，能够为单架或多架无人机分别规划一条从指定起点出发，到达指定终点的，能够满足无人机的飞行参数约束，并且其性能指标（如最短的飞行航程、最短的飞行时间、最高的飞行安全性等）能达到一个能够接受的较好的飞行路径的技术方法<sup>[15]</sup>。

一般的航迹规划算法主要分为依次是环境建模和路径搜索两个步骤。其中，环境建模是将无人机所处的连续空间转化为便于计算机理解、计算的拓扑空间，并生成路径网络图；而路径搜索则是在路径网络图的基础上，采用一定的路径搜索策略来生成各架无人机在环境下的最佳航迹。到目前为止，无人机航迹规划算法主要分为以下四种：图搜索法、采样算法、人工势场法、智能算法，如表 1-2 所示。

表 1-2 无人机航迹规划技术研究总结

类型	方法	动态航迹规划
图搜索法	切线法 <sup>[16]</sup>	否
	A* 算法 <sup>[17-18]</sup>	是
人工势场法	APF 算法 <sup>[19-20]</sup>	是
采样算法	RRT 算法 <sup>[21]</sup>	是
	RRT* 算法 <sup>[18,22]</sup>	是
	RRT-Connect 算法 <sup>[23]</sup>	是
智能算法	GA 算法 <sup>[24]</sup>	是
	PSO 算法 <sup>[25]</sup>	是
	Q-Learning 强化学习 <sup>[26]</sup>	是

从图论角度出发，Rohnert<sup>[16]</sup>提出利用凸多边形的切线寻找无人机最优航迹的算法，即切线法，但该方法会需要大量的内存以储存各个凸多边形的切线，规划效率低，且生

成的航迹曲折且紧靠障碍物，在实际场景中难以应用。Hart et al.<sup>[17]</sup>提出的 A\* 算法是目前常用的用于在地图中寻找最优路径的算法之一，能够在较短时间内得到最短路径。李春华等<sup>[18]</sup>将路径的约束条件一并融入进 A\* 算法当中，提出了基于稀疏 A\* 算法的三维航迹规划算法，能够快速有效地完成规划任务，提升了航迹搜索的效率。

由 Khatib<sup>[19]</sup>提出的人工势场法（Artificial Potential Field, APF）也是解决无人机航迹规划问题的一种有效方法，其基本原理是借助物理中“场”的概念，将环境虚拟成一个具有势能的场景，其中目标点对无人机产生吸引力，障碍物对无人机产生排斥力，从而规避障碍物到达目标点，但存在着容易陷入局部最优解等缺点。

从空间探索角度出发，快速搜索随机树（Rapidly-exploring Random Tree, RRT）是一种通过在空间中进行采样来生成路径的规划方法，可以不通过对环境空间建模，而是直接从环境空间中进行采样。Lin et al.<sup>[21]</sup>基于快速搜索随机树（Rapidly-exploring Random Tree, RRT）提出闭环 RRT 算法并将其成功应用于无人机航迹规划问题当中。Karaman et al.<sup>[22]</sup>则证明了在航迹规划问题中 RRT\* 算法相较 RRT 算法有着更好的性能。Zhang et al.<sup>[23]</sup>通过结合 APF 算法与 RRT-Connect 算法，在复杂静态的航迹规划问题中取得了更好的性能表现。

智能算法，如启发式智能算法和人工智能算法等，因其具有的广泛的适用性和高效的求解能力，成为了近年来的热门话题，Sonmez et al.<sup>[24]</sup>采用了遗传算法（Genetic Algorithm, GA）进行求解，Wang et al.<sup>[25]</sup>采用了粒子群算法（Partial Swarm Optimization, PSO）进行求解，Yan et al.<sup>[26]</sup>成功地将 Q-Learning 强化学习方法进行改进并使其能够解决航迹规划问题，但此类算法目前存在着求解稳定性较差、计算成本高等缺点。

### 1.2.3 无人机任务分配技术研究现状

无人机任务分配是基于无人机的飞行约束、性能约束以及已知的环境信息，根据给定的任务需求，对无人机分配一组最优的有序任务序列，以达到完成任务的收益最大或完成任务的数量最多，同时体系的整体效益以及可用资源的利用率达到最优的技术方法。无人机任务分配问题是一个有着多参数、多约束的 NP-Hard 问题<sup>[27]</sup>，会随着所给任务、所用无人机的规模的增加，其解空间的大小呈现指数级增加。该问题与常见的组合优化问题，诸如旅行商问题（Traveling Salesman Problem, TSP）、车辆路径规划问题（Vehicle Routing Problem, VRP）和工作流调度问题（Job-shop Problem, JSP）等经典问题存在着相似性。目前常用的方法有精确算法、启发式智能算法等，如表 1-3 所示。

在求解时，精确算法具备稳定搜索到全局最优解的能力，Alidaee et al.<sup>[28]</sup>将无人机任务分配问题转化为混合整数线型规划模型（Mixed Integer Linear Programming, MILP），并使用了分支定界法来对该模型进行求解。Schermer et al.<sup>[29]</sup>同样将其研究的问题建模

表 1-3 无人机任务分配技术研究总结

类型	方法	优缺点
精确算法	分支定界法 <sup>[28]</sup>	可求得最优解，但耗时长，
	线性规划算法 <sup>[29]</sup>	不适用于大规模问题
	列生成算法 <sup>[30]</sup>	
启发式智能算法	GA 算法 <sup>[4,31]</sup>	全局性好，能获得满意解，
	AC 算法 <sup>[32]</sup>	但解的质量不稳定
	PSO 算法 <sup>[33]</sup>	

为 MILP 模型，然后在此基础上应用了线性规划算法，取得了较好的结果，但仅能在问题规模较小下能在可接受的时间范围内得到结果。Faiz et al.<sup>[30]</sup>通过使用列生成算法对 MILP 模型进行降维并对降维后的模型进行求解，使得在问题规模较大时能有更好的计算效率。

精确算法虽然能够稳定搜索到全局最优解，但其在求解大规模问题下的计算成本呈指数级增加，难以应用于实际场景，为了解决这个问题，不少学者使用启发式算法进行求解，启发式算法虽然不能稳定获得全局最优解，但可得到较优解，且所需计算成本远小于精确算法。姚敏等<sup>[4]</sup>构建了一种适用于多目标、多无人机、多任务种类的集群协同任务分配模型，并在其基础上设计了 GA 算法，但由于该模型较为复杂，较难设计 GA 算法的染色体编码方式，而且 GA 算法的运行时间会随着问题规模的增大而相应增加，在大规模问题上的求解能力尚且不足。Zhen et al.<sup>[32]</sup>针对多无人机任务分配问题，设计了改进的分布式蚁群算法（Ant Colony Algorithm, AC），并通过大量的仿真实验验证了该算法在求解该问题上的鲁棒性。Zhu et al.<sup>[33]</sup>将地震后场景下的无人机任务分配问题视为团队定向运动的一种特殊情形，并提出一种基于 SA 算法的高效 PSO 算法（HPSO-SA）来解决该问题。

### 1.3 研究内容

针对无人机航迹规划及任务调度问题，本文在现有相关技术研究的基础上，考虑边缘计算技术，将其分解为两个子问题，分别是任务计算资源调度子问题、多无人机航迹

规划及任务分配子问题。对于各个子问题，以最小化无人机的总航程、最小化任务集合的完成时间、最小化设备的总空闲时间等为优化目标，分别建立面向边缘计算具备负载平衡的任务资源分配模型、多无人机的航迹规划及任务分配模型；然后，提出基于多种启发式智能优化算法的不同算法来分别求解上述问题，并设计对应的性能实验验证算法的有效性及性能优越性。

- **第一章：**介绍了考虑边缘计算的无人机交通侦察动态规划方法的研究背景和意义，查询并了解了问题中所涉及到的相关技术目前在国内外的研究现状，从而以此为基础确定本文具体的研究方向和技术路线；
- **第二章：**描述了无人机航迹规划及任务调度的任务场景和类型，将其分解为了任务计算资源调度问题和多无人机航迹规划及任务分配问题，并根据对问题所处环境进行建模，并提出合理的模型假设，建立对应的任务计算资源调度模型、多无人机航迹规划及任务分配模型，并分别对各个模型的优化目标和约束条件进行介绍；
- **第三章：**介绍了考虑边缘计算的无人机航迹规划及任务调度算法的应用流程，设计并提出了基于 ASA 的静态任务计算资源调度算法、基于 FD 的动态任务计算资源调度算法、基于 RRT\*-Connect 的静态场景下的无人机航迹规划算法、基于 A\* 的动态场景下的无人机航迹规划算法和基于 TSAVN 的多无人机任务调度分配算法来分别解决任务计算资源调度子问题、多无人机航迹规划及任务分配子问题；
- **第四章：**设计并构建仿真实验场景，对任务计算资源分配模型、航迹规划及任务分配模型的求解算法分别进行性能测试，证明算法的有效性。
- **第五章：**总结本文的研究工作，并展望下一步的研究方向及相应工作。

## 1.4 技术路线

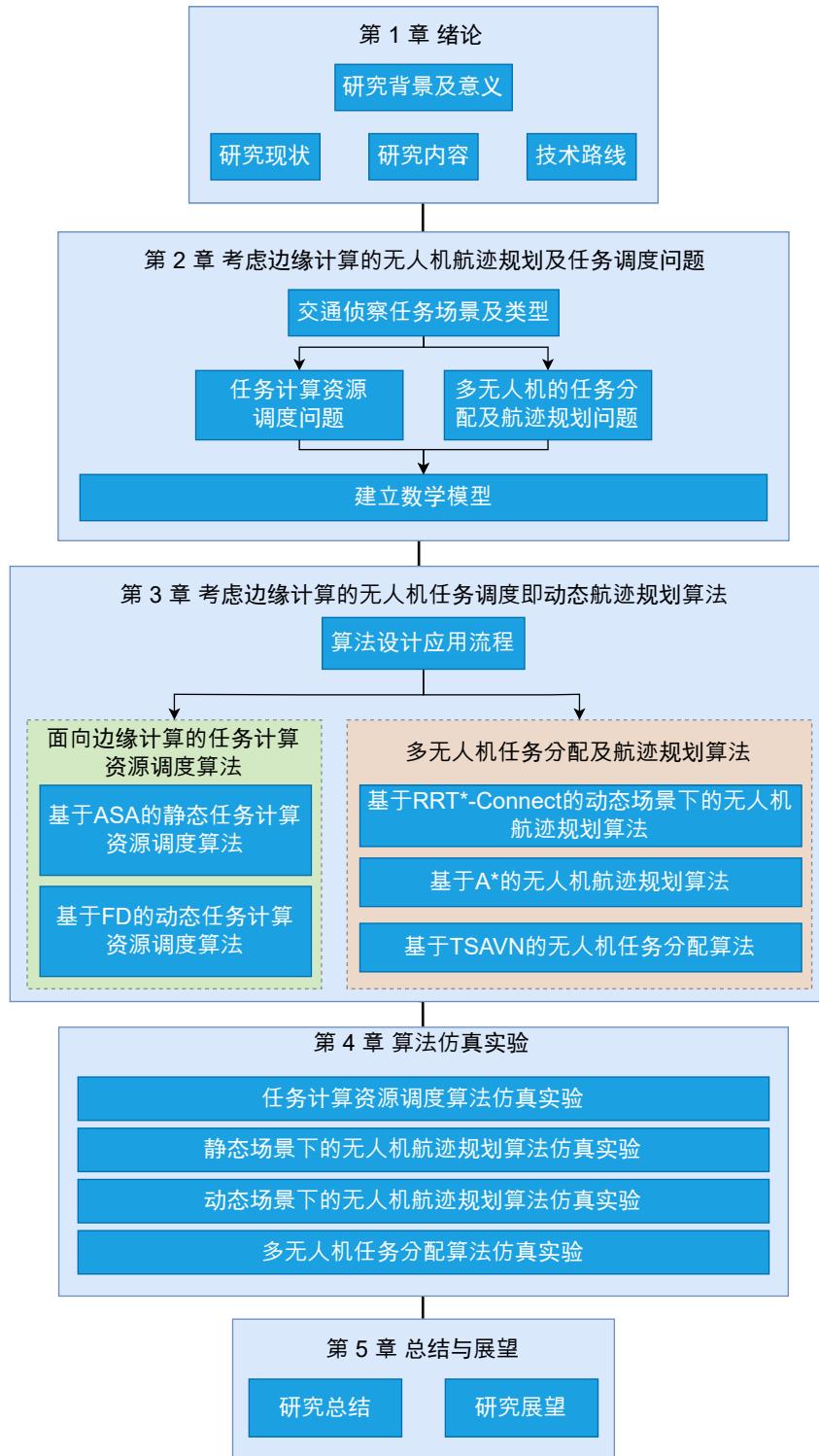


图 1-5 毕业设计技术路线结构图

## 第 2 章 考虑边缘计算的无人机航迹规划及任务调度问题

### 2.1 问题描述及场景设置

#### 2.1.1 问题描述

在交通及交通相关领域中，因为无人机具备隐蔽性好、机动性强、灵活性强，部署容易和应用范围广等特点，可以通过无人机的使用来一定程度上代替人工来完成所需的侦察任务，例如在铁路系统中对铁轨健康状况的检查、事故现场的侦察勘探、高速公路上的交通巡逻任务（见图 2-1）等，因此，基于无人机的交通侦察应用逐渐具备了可行性与实用性。

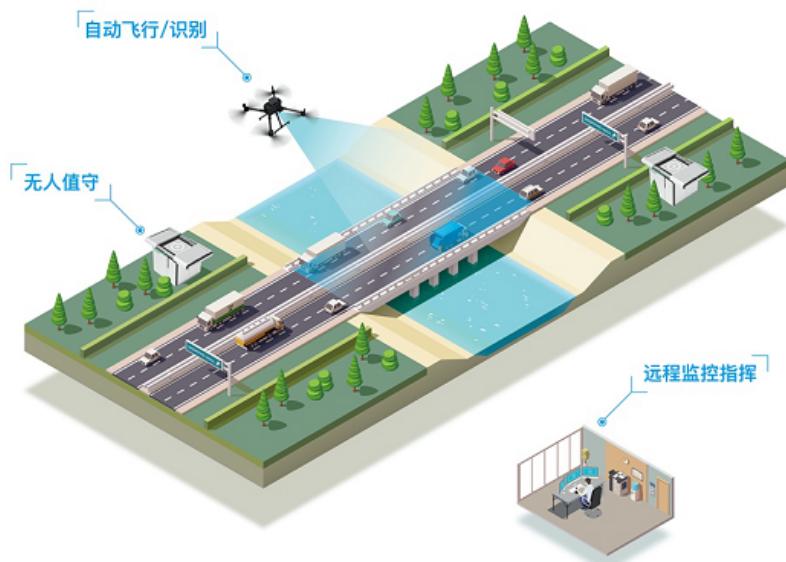


图 2-1 无人机在高速公路上执行交通巡逻任务

本文所研究的问题内容为考虑边缘计算的无人机航迹规划及任务调度问题，为方便讨论，本文将无人机需要执行的侦察任务分为两个部分（见图 2-2a），分别为收集型任务和计算型任务，其关系如图 2-2b 所示。收集型任务主要与无人机的摄像设备以及储存设备有关，因此必须由无人机来完成，且该种任务的完成时间是可知的；而计算型任务则需要用到大量算力，若全靠无人机进行计算，则会使得无人机的续航大幅下降，从而使得无人机交通侦察系统效率大幅降低，而全由服务器进行计算，则容易因数据大量同时传输而造成网络拥堵等现象，同样降低了无人机交通侦察系统的效率。

为便于理解上述概念，此处不妨以侦察城市中道路交叉口当前的道路拥堵情况的侦察任务为例，可以将其分解为对该交叉口拍照或录制视频来收集当前道路情况的信息的收集型任务，以及对上述收集到的信息通过神经网络等方法进行计算得出当前该交叉口

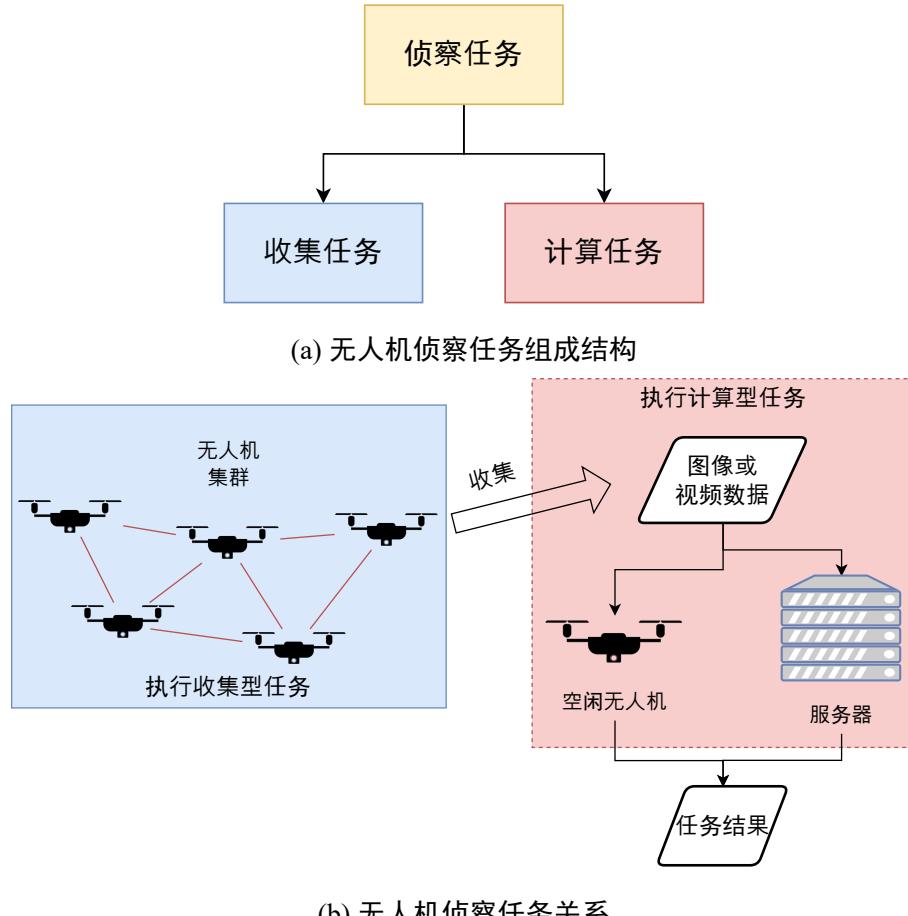


图 2-2 无人机交通侦察任务组成及关系

的道路拥挤情况的计算型任务。对于某个区域而言，存在着 22 个可能出现拥堵的交叉口（见图2-3），需要使用一定数量的无人机来对每个交叉口拥堵情况进行数据收集，在每个任务的数据收集完成后，按照计算资源分配方案将任务数据通过网络通信等方式传输至地面端或边缘端中进行计算，得出各个交叉口拥挤情况后采取相应的措施。

那么基于上述内容，该问题的整体框架如图 2-4所示，具体流程如下：

- (1) 当上级下达指令后，位于地面端的服务器根据任务信息以及地图信息，得到无人机的任务分配方案，包含所需无人机的数量、各无人机所要执行的需进行数据的采集与收集的收集型侦察任务以及相应的飞行航迹；
- (2) 同时服务器根据任务信息、服务器的计算资源以及无人机的计算资源，得到考虑边缘计算下的计算资源分配方案，来安排执行各计算型任务的设备；
- (3) 之后根据得到的无人机的收集型任务分配方案，将对应指令传输至无人机集群中，无人机根据接收到的指令进行飞行控制；
- (4) 在无人机按照飞行航迹进行飞行时，若飞行航迹中存在与航迹冲突的障碍物，那么需要实时对航迹进行修订，使得无人机能够在与障碍物发生碰撞之前对障



图 2-3 示例地图信息

碍物进行回避;

- (5) 在无人机到达目标点并采集完任务数据后，按照计算资源分配方案将任务数据通过网络通信等方式传输至地面端或边缘端中进行计算；
- (6) 最后根据计算所得的侦察任务结果采取对应的措施。

其中数据与指令的传输不在本文的研究范围内，因此本文将考虑边缘计算的无人机航迹规划及任务调度问题分为两个子问题，分别为面向边缘计算的任务计算资源调度优化问题、多无人机的航迹规划及任务分配问题。

任务计算资源的调度优化问题是在有限的计算资源及设备的情况下，通过一定的算法生成任务计算资源调度方案来安排计算设备完成指定的计算型任务，使得整个系统的成本较低，具备实用性的问题。因为无论是无人机集群还是服务器，其计算资源存在着一定的上限，而过多的布置无人机或服务器的数量，虽然能够确保所有计算任务都能够及时准确地完成计算，但会极大增加系统的成本，且有非常严重的资源浪费，而对计算资源进行充分地利用，能极大降低应用成本，提高系统的可行性，因此，在有限计算资源下对任务计算资源进行调度优化十分重要。对于整个系统而言，由于侦察任务有不同的种类，每个种类对于计算资源的要求各有不同，而无人机的计算资源，无论是可用CPU核心数还是可用内存，都与服务器存在着较大的差异，因此会存在着对于不同的侦

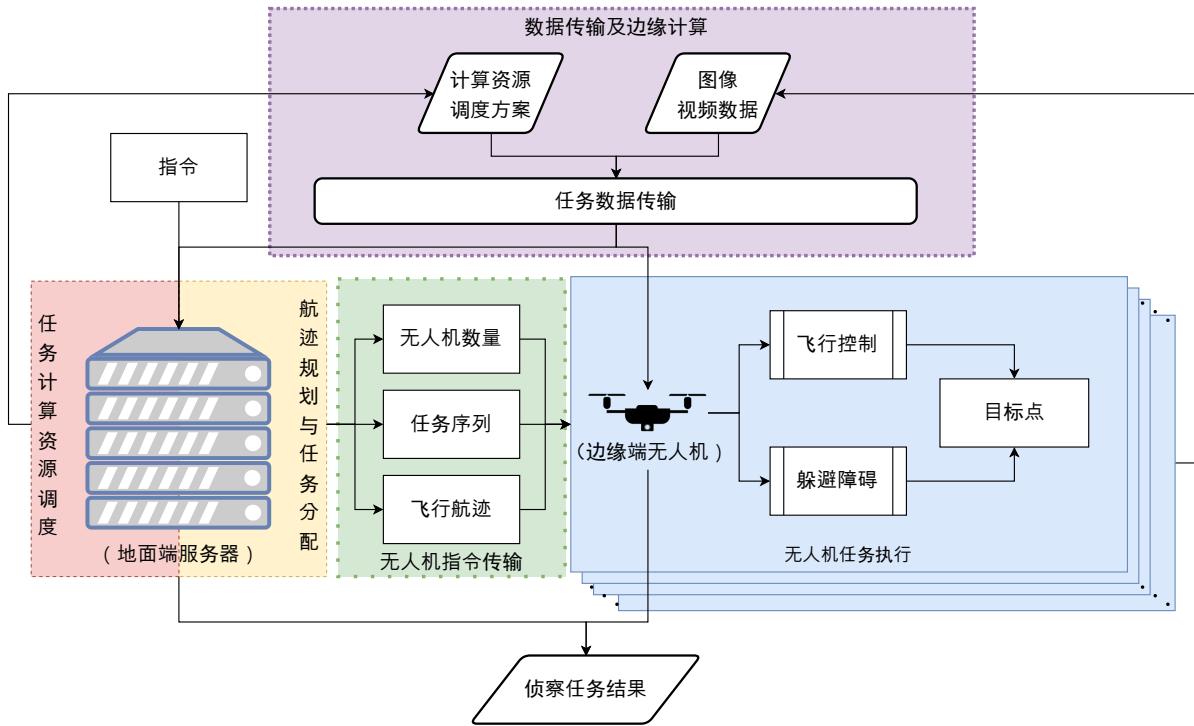


图 2-4 本文研究问题整体框架

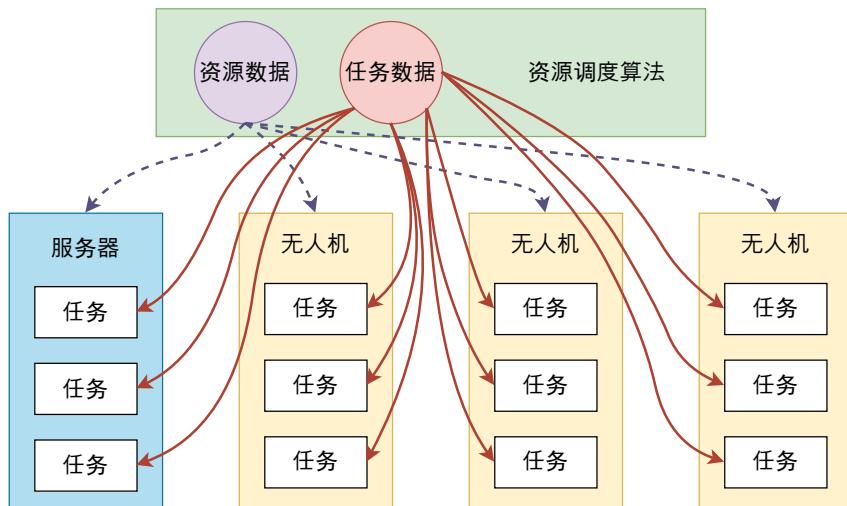


图 2-5 资源调度算法示意图

察任务，其在不同设备的计算完成时间不一致的现象。与此同时，由于无人机与无人机、无人机与服务器之间是通过无线通信技术进行交流，因此还会存在着诸如通信延迟、传输丢包等通信问题，如何处理这些问题也是解决任务计算资源的调度优化问题中极为重要的内容。本文则提出一种具备负载均衡的面向边缘计算的任务计算资源分配算法（如图 2-5 中的资源调度算法），根据设备的计算资源情况以及任务的需求情况，将不同的任务分配至无人机与服务器计算，合理有效地对计算资源进行分配。

由于无人机的计算功率，即是否进行对侦察任务数据的计算、计算所耗费的时间

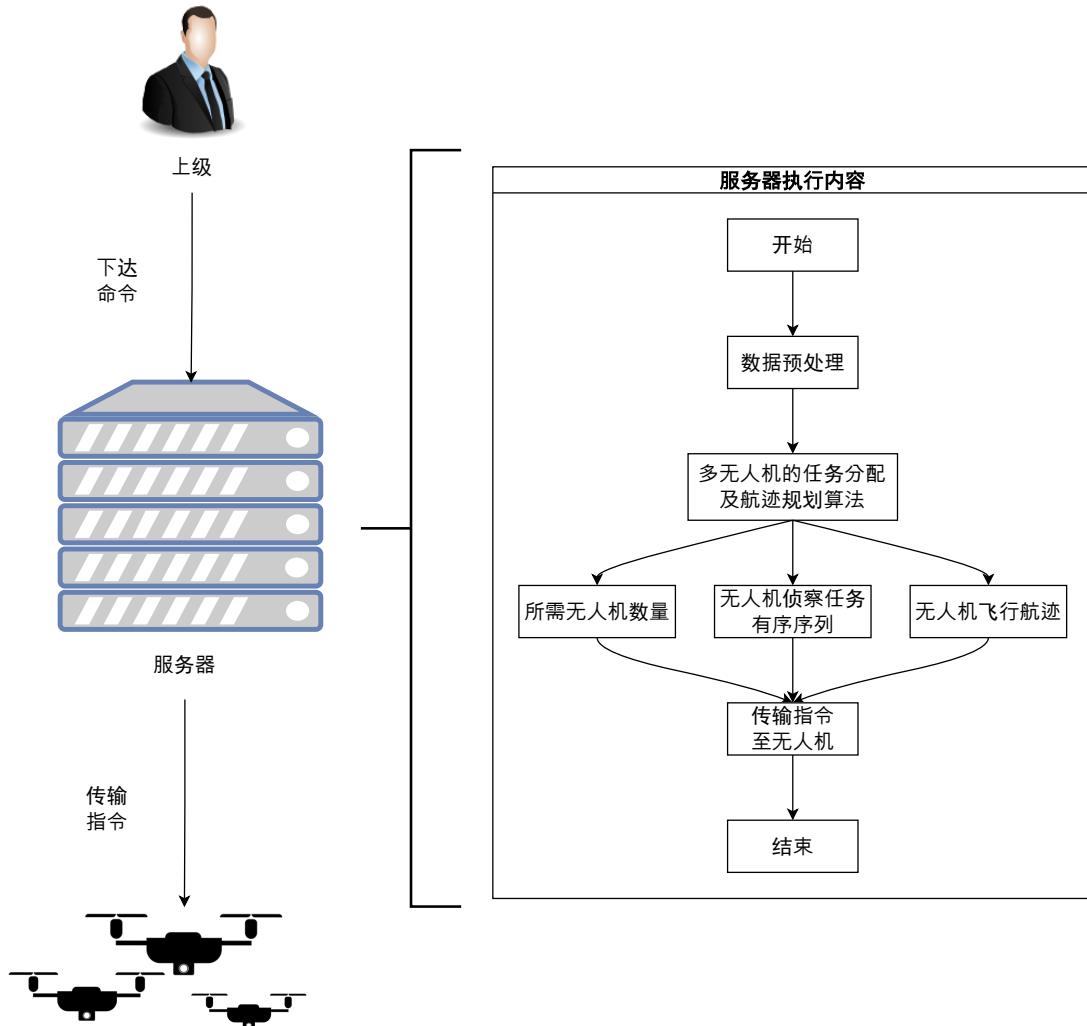


图 2-6 多无人机的航迹规划及任务分配流程

等，都会影响到无人机飞行的续航时间，而这不利于本章算法所涉及的资源分配问题的解决，同时也会对另外两个问题产生影响。因此，本章假定无人机的续航时间为可容许的最小续航时间，而剩余的续航时间所对应的计算资源则供给给可能需要进行计算的计算型侦察任务的数据。

如图 2-6 所示，多无人机的航迹规划及任务分配问题是在给定收集型任务信息、地图信息后，由服务器根据所给信息，在尽可能短的时间内使用一定的算法确定完成任务所需的无人机数量、各无人机所执行的最优侦察任务有序序列、各无人机执行任务序列的最优飞行航迹，以便控制无人机完成所给的侦察任务的问题。

### 2.1.2 场景设置

本文以无人机群在城市环境中低空飞行，完成指定区域的目标识别任务、目标搜索任务等侦察任务为研究的应用场景。为简化整体问题便于研究，本文设定执行侦察任务所用无人机为同一型号的旋翼无人机，环境为存在障碍物高低差且障碍物数量较多、较

为复杂的城市环境作为测试场景。

## 2.2 模型假设

为了能够集中力量研究主要问题，确定研究问题的边界，便于开展问题的研究，在不影响模型建立和求解算法的基础上，本文对考虑边缘计算的无人机航迹规划及任务调度问题提出下列合理的假设以简化模型：

1. 无人机应用场景中，所有的障碍物均以长方体进行拟合；
2. 服务器具有的计算能力和计算资源均强于无人机，使得在计算同一侦察任务的数据使服务器能够以更快的速度完成任务以及同时能够完成更多任务；
3. 通信传输的延迟不会受到除了设备间的欧式距离以外的因素的影响；
4. 无人机与无人机、无人机与服务器之间的网络延迟具备可预测性，即能够根据无人机位置、通信时间等预测设备间的网络延迟；
5. 无人机携带的传感器能够实时感知所在环境的障碍物信息，能够获取无人机与障碍物的距离等所需信息；
6. 无人机能够自行处理部分侦察任务的数据，但计算时间会长于服务器计算时间；
7. 服务器拥有的地图数据为最新数据，与现实地图不存在较大的差距变化，例如现实中存在的大楼在地图数据中显示为空地，仅存在微小的变化，例如大楼外部的空调外机；
8. 无人机进行转向活动时耗费的时间不计；
9. 同一任务在相同设备上运行时间具备可预测性，即能够根据任务类型、任务所需内存及设备类型预测该任务在该设备下完成计算所需时间；
10. 无人机和服务器都可以同时执行任意数量的任务且不会影响它们的运算速度。

## 2.3 符号定义

表 2-1 为小节 2.4 中的数学模型所使用的数学符号：

表 2-1 数学符号及意义

符号	意义
$N$	无人机数量
$\mathbf{O}$	障碍物集合
$\ell_{\max}$	无人机最大航程

$C_k$	第 $k$ 台无人机的航迹坐标
$d(C_k)$	根据第 $k$ 台无人机的航迹坐标计算得出的飞行航程
$\mathbf{J}$	任务集合
$J$	任务数量
$i$	第 $i$ 个任务
$\mathbf{T}$	任务类型集合
$\mathbf{T}_i$	第 $i$ 个任务的类型
$k$	第 $k$ 台设备
$\mathbf{M}$	设备集合
$M$	设备数量
$G_k$	第 $k$ 台设备为地面端, 0 则不是, 1 则是
$L_k$	第 $k$ 台设备的最大内存
$W_i$	执行第 $i$ 个任务所需内存
$D_{ik}$	第 $i$ 个任务上传至第 $k$ 个设备的通信延迟, 不需要上传时为 0
$X_{ik}$	第 $i$ 个任务是否由第 $k$ 个设备处理
$F_{ik}$	第 $i$ 个任务上传至第 $k$ 个设备的上传速度, 不需要上传时为 $\infty$
$P(W_i, \mathbf{T}_i, G_k)$	根据第 $i$ 个任务的所需内存和任务类型预测在第 $k$ 个设备中开始处理到处理完成所需的时间的函数
$t_{ik}$	第 $i$ 个任务从开始执行到在第 $k$ 个设备中处理到执行完成总共的时间
$S_i$	第 $i$ 个任务开始执行的时间
$C_i$	第 $i$ 个任务执行完成的时间
$C_{\max}$	所有任务中最后完成执行的任务的完成时间

## 2.4 数学模型

本小节根据小节 2.1 中对无人机交通侦察动态规划问题进行分解后的问题分别建立任务计算资源调度模型（见小节 2.4.1）、多无人机航迹规划及任务分配模型（见小节 2.4.2）。

### 2.4.1 任务计算资源调度模型

$$\min F_1 = (C_{\max}, C_{\max} \cdot M - \sum_i^J \sum_k^M t_{ik} X_{ik}) \quad (2-1)$$

$$C_{\max} = \max(C_i) \quad (2-2)$$

$$C_i = S_i + \sum_{k \in [1, M]} t_{ik} X_{ik} \quad (2-3)$$

$$t_{ik} = 2D_{ik} + \frac{W_i}{F_{ik}} + P(W_i, \mathbf{T}_i, G_k) \quad (2-4)$$

s.t.

$$\sum_{i \in [1, J]} W_i X_{ik} < L_k, \forall i \in [0, C_{\max}] \forall k \in [1, M] \quad (2-5)$$

$$\sum_{k \in [1, M]} X_{ik} = 1, \forall i \in [1, J] \quad (2-6)$$

$$X_{ik} = \{0, 1\}, \forall i \in [1, J], k \in [1, M] \quad (2-7)$$

$$\sum_{i \in [1, J]} \sum_{k \in [1, M]} X_{ik} = J \quad (2-8)$$

$$P(W_i, \mathbf{T}_i, G_k) > 0, \forall i \in [1, J], k \in [1, M] \quad (2-9)$$

$$C_i > S_i \geq 0, \forall i \in [1, J], k \in [1, M] \quad (2-10)$$

$$D_{ik}, F_{ik}, L_i, W_i, t_{ik}, G_k > 0, \forall i \in [1, J], k \in [1, M], G_k \in \mathbb{Z} \quad (2-11)$$

其中，式 2-1 为模型的目标函数，该目标值的含义如表 2-1 中所示，即任务完成时间  $C_{\max}$  最短、设备空闲时间  $C_{\max} \cdot M - \sum_{i=1}^J \sum_{k=0}^M t_{ik} X_{ik}$  最少，而设备空闲时间最少是为了确保设备的资源利用率，使得本文所提出的算法具备负载平衡的功能，其中任务完成时间的计算方式如式 2-2 所示；而对于第  $i$  任务的完成时间  $C_i$ ，则由式 2-3 中所示，由任务的开始执行时间  $S_i$  与决策确定的完成所需时间  $\sum_{k \in [1, M]} t_{ik} X_{ik}$  之和决定；而对于第  $i$  个任务在第  $k$  台设备上的完成所需时间  $t_{ik}$ ，则由式 2-4 所决定，其中分别为上传延迟  $D_{ik}$ 、上传时间  $\frac{W_i}{F_{ik}}$ 、计算时间  $P(W_i, \mathbf{T}_i, G_k)$  和下载延迟  $D_{ik}$  所决定。

在上述基础上，本文对整个场景进行了约束，其中式 2-5 确保在任意时刻下，任意设备所执行的任务所需内存总和不超过该设备最大可用内存；式 2-6 和 2-7 确保每一个任务只会且只能由一台设备所执行；式 2-8 确保每一个任务都会被完成；式 2-9 是确保对任务完成所需时间的预测在一个符合实际的范围；式 2-11 则对各个变量进行约束。

#### 2.4.2 多无人机航迹规划及任务分配模型

$$\min F_2 = \left( \sum_{k=1}^N d(C_k), N \right) \quad (2-12)$$

s.t.

$$C_k \notin \mathbf{O}, \forall k \in [1, N] \quad (2-13)$$

$$d(C_k) \leq \ell_{\max}, \forall k \in [1, N] \quad (2-14)$$

$$C_1 \cap C_2 \cap \dots \cap C_N = \emptyset \quad (2-15)$$

其中，式 2-12 表示优化目标为无人机数量最少，且无人机的飞行航程之和最小；式 2-13 为无人机与障碍物的飞行避撞约束，即无人机在飞行航迹中任意一点均不与障碍物接触，从而避免飞行过程中发生无人机因与障碍物碰撞而损毁的情况；式 2-14 为无人机的航程约束，确保每一无人机飞行航迹的飞行航程均不超过其最大航程，从而避免无人机因续航耗尽导致的损毁；式 2-15 为无人机与无人机的飞行避撞约束，即无人机在飞行航迹中任意一点均不与其他无人机接触，从而避免飞行过程中发生无人机因与其他无人机碰撞而损毁的情况。

#### 2.5 本章小结

本章主要对无人机航迹规划及任务调度问题进行具体描述和分析，同时将该问题分解成了两个子问题，分别是任务计算资源的调度优化问题、多无人机的航迹规划及任务分配问题，并在提出合理假设和数学符号的基础上，分别建立任务计算资源的调度优化问题、多无人机的航迹规划及任务分配问题的数学模型，并对各个模型的目标函数及约束条件进行了说明。

## 第3章 考虑边缘计算的无人机航迹规划及任务调度算法

### 3.1 算法设计应用流程

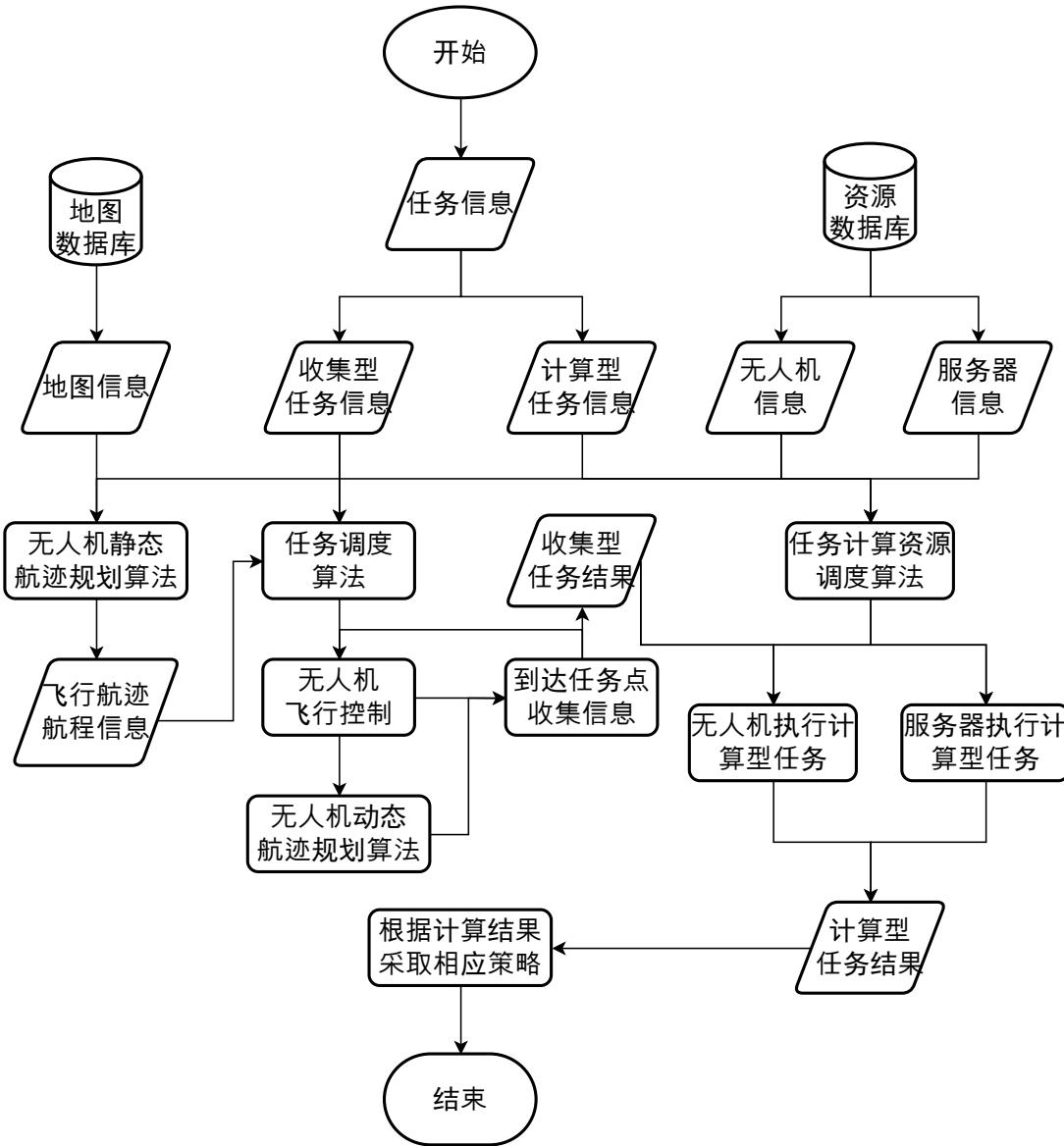


图 3-1 算法设计应用流程

本文针对无人机航迹规划及任务调度问题所设计的算法的应用流程如下，首先将所给的任务信息分解成收集型任务信息和计算型任务信息，然后服务器根据收集型任务信息、从资源数据库中得到的无人机信息、从地图数据库中得到的地图信息，放入静态场景下的无人机航迹规划算法中，得到各点间的无人机飞行航迹和航程的对应信息，再将以上信息以及收集型任务信息放入任务调度算法，得到所需无人机的数量、每个无人机

需要执行的收集型任务有序序列、无人机飞行航迹等信息；同时服务器再根据计算型任务信息、无人机信息和服务器信息通过任务计算资源调度算法，为每一个计算型任务分配对应的计算设备；然后无人机根据所给的任务序列和飞行航迹进行飞行控制，若遇到航迹中的存在的障碍物则使用动态场景下的航迹规划算法修订航迹来回避障碍物，在到达任务点后执行收集型任务，在完成后将收集得到的任务结果传输至执行该收集型任务对应的计算型任务的设备中进行计算，在计算型任务完成后，便能够根据任务结果采取相应的措施，流程如图 3-1所示。

### 3.2 面向边缘计算的任务计算资源调度算法

由于生成任务计算资源调度方案需要无人机的航程信息，而在无人机飞行过程中，随时可能因为出现与原定航迹相冲突的障碍物而对航迹进行修订，进而使其航程信息发生变化，因此在解决该问题时，本文先将该问题分为无人机尚未接收到指令开始执行任务前的静态场景以及无人机开始执行任务，但由于中途进行了避障操作而使得原本的资源调度方案需要重新生成的动态场景。在静态场景中，由于不需要在很短时间内给出调度方案，因而计算资源调度方案的质量比算法计算时间更为重要；而在动态场景中，由于场景信息不断发生变动，为避免计算资源调度方案的生成时间过长导致无人机的数据传输需要进行等待，进而导致系统效率下降，此时算法的运行时间更为重要。

基于以上分析，本文提出的面向边缘计算的任务计算资源调度算法共包含两个算法，分别是用于求解静态场景的参数自适应的模拟退火算法（Adaptive Simulated Annealing, ASA）以及用于求解动态场景的快速决策算法（Fast Decision-making Algorithm, FD）。

#### 3.2.1 基于 ASA 的静态场景下的任务计算资源调度算法

本算法是在传统 SA 算法的基础上，对其进行邻域结构以及参数方面的改进。

##### (1) 邻域结构设计

由于需要执行的任务数是一定的，且一个任务仅由一个设备进行处理计算，而一个设备可能需要处理多个任务，因此可以简单地使用一个数组来作为算法的解，其中数组的元素对应的位置为任务的序号、数组的元素为设备的序号，以此来设计本文所使用的邻域结构。

第一种邻域结构为任务交换邻域结构，如图 3-2所示，具体操作流程如下：

- (1) 选取待交换的两个任务  $M_1$ 、 $M_2$ ；
- (2) 获取用于处理任务  $M_1$  与  $M_2$  的设备  $D_1$  与  $D_2$ ；
- (3) 交换处理这两个任务的设备，即原本由设备  $D_1$  处理任务  $M_1$ ，设备  $D_2$  处理任务  $M_2$ ，变为由设备  $D_2$  处理任务  $M_1$ ，设备  $D_1$  处理任务  $M_2$ 。

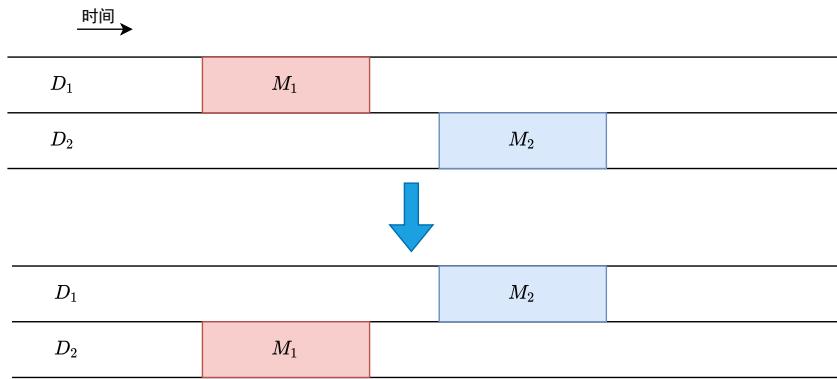


图 3-2 任务交换邻域结构

第二种邻域结构为无人机间任务转移邻域结构，如图 3-3 所示，具体操作流程如下：

- (1) 选取任务  $M_1$ 、 $M_3$ ；
- (2) 获取用于处理任务  $M_1$  与  $M_3$  的无人机设备  $U_1$  与  $U_2$ ；
- (3) 将任务  $M_1$  分配给无人机设备  $U_2$  处理，即原本由无人机设备  $U_1$  处理任务  $M_1$ ，变为由无人机设备  $U_2$  处理任务  $M_1$ 。

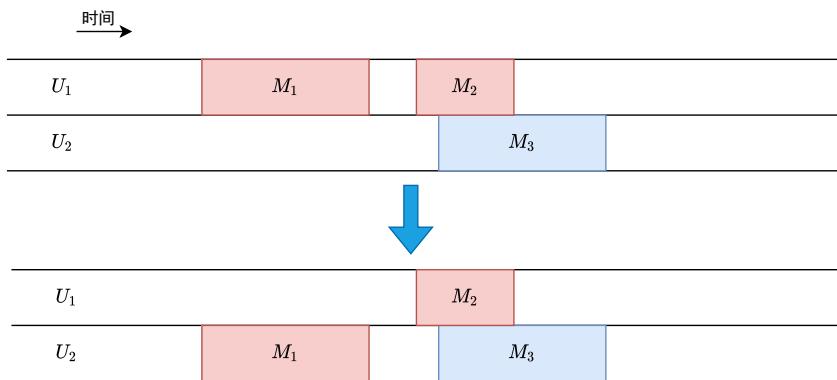


图 3-3 无人机间任务转移邻域结构

第三种邻域结构为无人机向服务器转移任务邻域结构，如图 3-4 所示，具体操作流程如下：

- (1) 选取任务  $M_1$ ；
- (2) 获取用于处理任务  $M_1$  的无人机设备  $U$ ；
- (3) 将任务  $M_1$  分配给服务器设备  $C$  处理，即原本由无人机设备  $U$  处理任务  $M_1$ ，变为由服务器设备  $C$  处理任务  $M_1$ 。

第四种邻域结构为服务器向无人机转移任务邻域结构，如图 3-5 所示，具体操作流程如下：

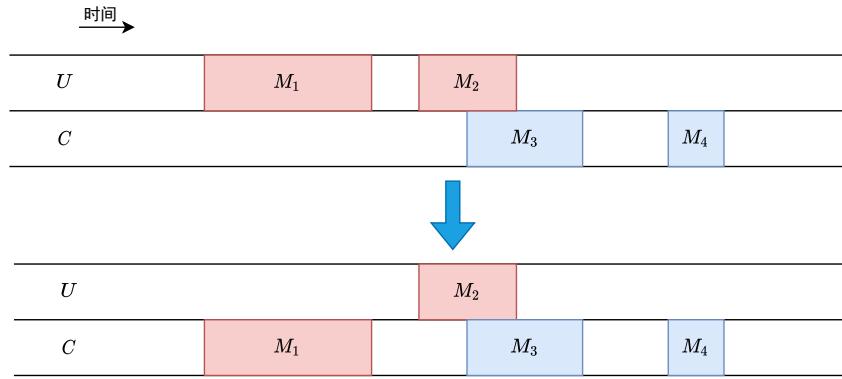


图 3-4 无人机向服务器转移任务邻域结构

- (1) 选取由服务器处理的任务  $M_1$ ;
- (2) 获取无人机设备  $U$ ;
- (3) 将任务  $M_1$  分配给无人机设备  $U$  处理，即原本由服务器设备  $C$  处理任务  $M_1$ ，变为由无人机设备  $C$  处理任务  $M_1$ 。

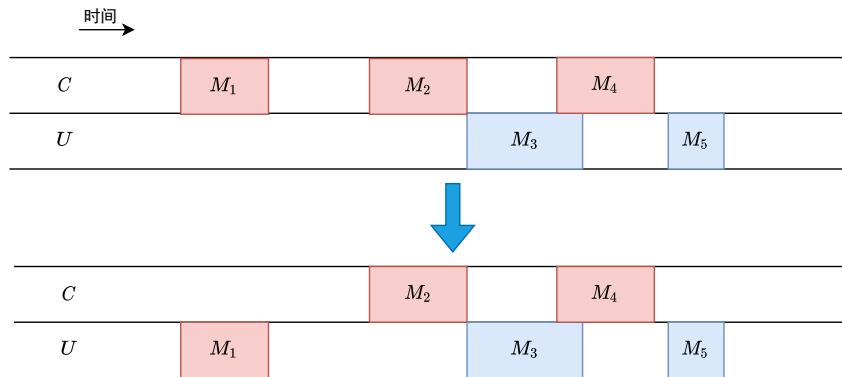


图 3-5 服务器向无人机转移任务邻域结构

## (2) 自适应机制设计

对于小节 1 中所述的邻域结构而言，在不同的情况下每个邻域结构的表现会存在差异，因此为使得算法性能更好，本文提出了基于优胜劣汰机制的邻域结构权重自适应机制，该机制能够使得表现优异的邻域结构在下次选择时被选到的机会更大，而表现较差的邻域结构则被选到的机会更小。

在算法初始化时，每个邻域结构的权重均初始化为 1。在之后权重的迭代更新中，为了避免滚雪球现象，即某个邻域结构在连续非常多次表现优异后，即使此后该邻域领域的表现非常差，选择其他邻域结构也非常小，或是连续非常多次表现较差时，即使此后该邻域领域的表现非常优异，继续选择该邻域结构的概率非常小，因此在设计该机制时，本文根据生成的新解的质量与当前解的质量进行比较，此处引入  $w_i^r$  作为第  $i$  个邻

域结构被第  $r$  次选中并计算完成后的权重， $\Delta E$  为解的质量下降的值，那么有

$$w_i^r = \begin{cases} \bar{w}^r & \text{if } \Delta E > 0 \\ 0.8 \cdot w_i^{r-1} & \text{if } \Delta E < 0 \end{cases}$$

另外，由于模拟退火算法在运行中会存在初期温度高，同时未达到局部最优解的情况，以及末期温度低，同时达到局部最优解的情况，导致其非更优解接受概率的利用程度不高，所以模拟退火算法的性能仍有待提高。

为提高模拟退火算法的性能，更充分地利用其跳出局部最优解的机制，此处采用了自适应温度机制取代了原有的温度机制。

此处引入  $r_i$  作为算法第  $i$  次运行后没有得到更优解的次数，即需要计算非更优解接受概率的次数，当直接得到了更优解时  $r_i = 0$ ，算法未运行时  $r_0 = 0$ ，即

$$r_i = \begin{cases} r_{i-1} + 1 & \text{if } \Delta E > 0 \\ r_{i-1} & \text{if } \Delta E = 0 \\ 0 & \text{if } \Delta E < 0 \end{cases}$$

同时使用迭代次数  $N$  来控制整个算法的逻辑。与模拟退火算法相比，其总的求解次数仅由  $N$  决定，而模拟退火算法的总求解次数为  $N \cdot \log_r \frac{T_0}{T_{\text{end}}}$ 。

设定一个最低温度  $T_{\min}$ ，升温速率  $\rho$  以及温控参数  $\delta$ ，使得第  $i$  次计算温度  $T_i$  时使用如下公式：

$$T_i = T_{\min} + \rho \cdot \ln(1 + \frac{r_i}{\delta})$$

如此便能够在能够寻求到最优解的情况下维持低温，在到达局部最优解、需要到达新的局部最优解时能够通过升高温度来跳出局部最优解，从而使算法具有更好的鲁棒性。

### 3.2.2 基于 FD 的动态场景下的任务计算资源调度算法

在计算资源调度的动态场景中，由于每个无人机有两种决策方式：将任务数据传输至地面端服务器，由地面端服务器进行计算；将任务数据传输至边缘端无人机，由边缘端无人机进行计算。

因此，在满足算法生成的计算资源调度方案具有实用性的条件下，为使得算法的运行时间尽可能短，本算法在无人机决策时采用了贪婪机制（见图3-6），在对所有无人机进行遍历决策后，即可生成计算资源调度方案。

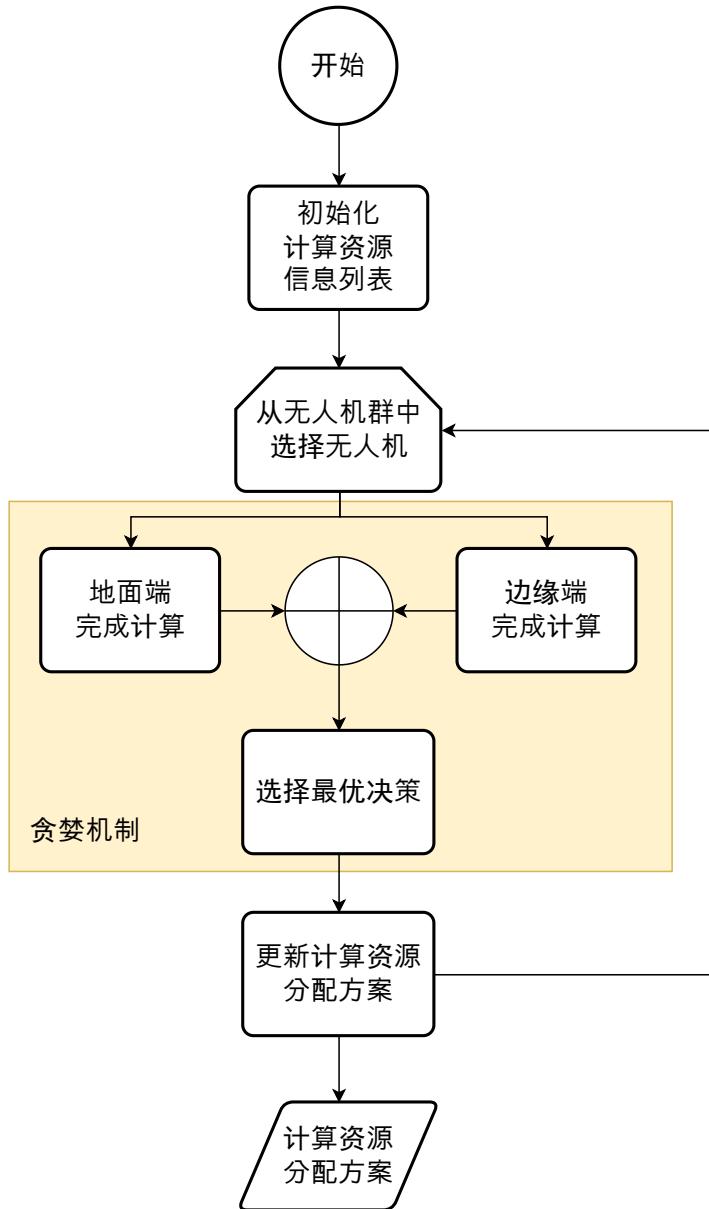


图 3-6 FD 算法流程图

### 3.2.3 算法流程图

根据上述内容，本文提出的面向边缘计算的任务计算资源调度算法的运行流程如下，首先从对应数据库中获取当前的计算型侦察任务的具体信息以及无人机、服务器的计算资源的信息，在初始化算法参数后，根据以上信息生成初始解，然后进行一定次数的迭代；在每步迭代中，算法从任务交换邻域结构、无人机间任务转移邻域结构、无人机向服务器转移任务邻域结构、服务器向无人机转移任务邻域结构中选择一个进行当前解的邻域变动，得到优化结果，判断该优化结果与目前的最优解是否更优，若更优则保存，否则进行概率判断，按随机概率确定是否保存，保存则更新最优解，然后更新算法参数，直至迭代结束，在迭代结束后，即可得到当前任务的计算资源最优分配方案。总

流程如图 3-7 所示。

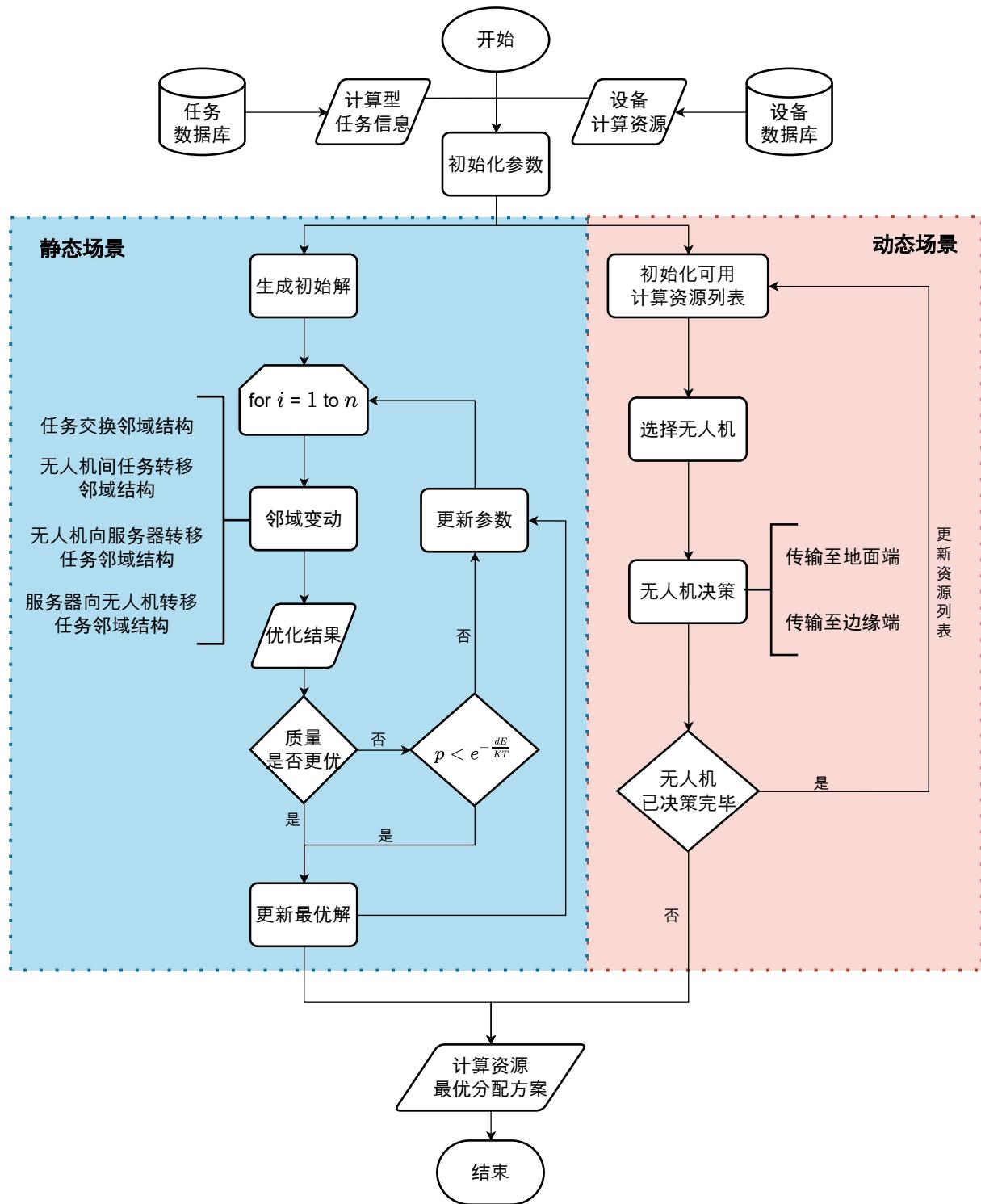


图 3-7 面向边缘计算的任务计算资源调度算法流程图

### 3.3 多无人机航迹规划及任务分配算法

本节针对多无人机航迹规划及任务分配问题，构建了基于快速随机搜索树算法、A\* 算法、模拟退火算法、变邻域搜索算法的多无人机航迹规划及任务分配框架，该框架将多无人机航迹规划及任务分配问题划分为了无人机航迹规划阶段以及任务调度分配阶段，流程如图 3-8 所示，并针对这两个阶段，设计相应的算法。在无人机航迹规划阶段，提出了能够在静态场景下快速获得最优航迹的 RRT\*-Connect 算法以及能够在动态场景下快速获得避障航迹的 A\* 算法；在多无人机任务调度分配阶段，提出了基于模拟退火算法、变邻域算法以及温度自适应模拟退火算法<sup>[34]</sup>的变邻域结构搜索的温度自适应模拟退火算法（Temperature-adaptive Simulated Annealing Algorithm with Variable Neighborhood, TSAVN）。由 RRT\*-Connect 生成无人机起飞点、任务点间的点与点的最优飞行航迹与航程，并利用生成的航程信息为 TSAVN 算法提供依据完成多无人机的任务调度。

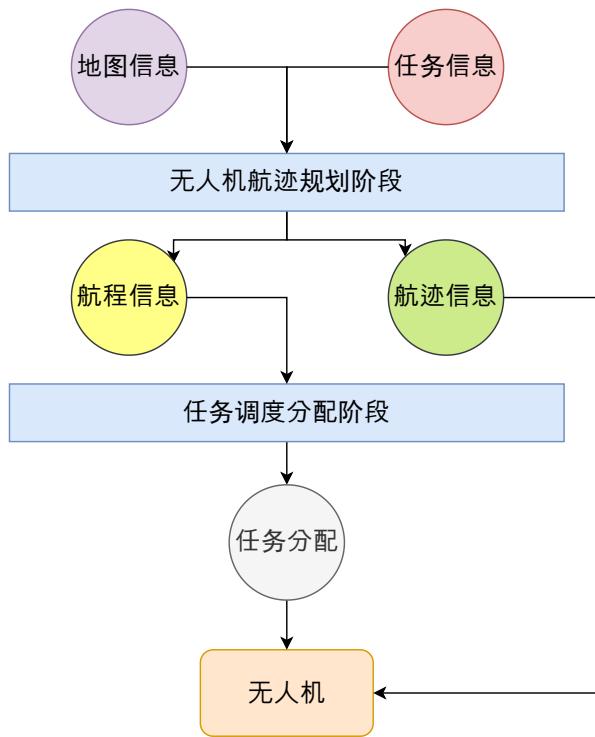


图 3-8 多无人机航迹规划及任务分配流程

多无人机航迹规划及任务分配算法，是在无人机航迹规划及任务调度系统中由服务器为了解决无人机的配给及无人机航迹规划及任务分配问题所采用的算法，为了解决该问题，要求该算法能够在尽可能短的时间内确定最优的任务分配方案，包括

- (1) 所需无人机数目；
- (2) 各无人机所要执行的侦察任务有序序列；

(3) 各无人机完成所给侦察任务有序序列的飞行航迹。

在服务器确定了以上内容后，会分配无人机并将对应的信息传输至无人机中，并由无人机自主根据所给指令完成所给侦察任务，在执行过程中，若出现了与飞行航迹冲突的障碍物，则再通过动态场景下的无人机航迹规划生成新的飞行航迹来规避障碍物。

### 3.3.1 基于 RRT\*-Connect 的静态场景下的无人机航迹规划算法设计

静态场景下的无人机航迹规划算法，是用于在无人机执行任务前为无人机规划能够在尽可能短的时间内安全到达目标点执行任务的飞行航迹的措施。该算法根据已有的地图信息和任务信息，生成点间的最佳飞行航迹，以保障无人机飞行的安全性、稳定性，并确保能够快速到达任务地点执行任务。由于该算法是在任务执行前运行的，因此在可接受的运行时间下其生成的航迹规划结果的质量更为重要。

RRT 算法是一种用于在高维非凸空间中通过构建路径树来生成路径的算法。这种树会通过随机在空间中采样来不断地拓展延伸，来探索未搜索的区域，最终达到目标点并得到起点至目标点的路径。Lavalle<sup>[35]</sup>与 LaValle et al.<sup>[36]</sup>对 RRT 算法进行了延伸，使该算法能够解决存在着障碍物及不同约束的航迹规划问题中，也因此该算法被广泛运用在机器人的运动规划问题当中。

由于原始的 RRT 算法是完全在几何空间中随机采样，在运动规划问题当中会出现难以到达目标点，即难以收敛的现象，因此，为了改善这个问题，往往会在采样时让其有一定的概率选择目标点作为其采样点，并根据与目标点最近的树节点进行再取样，选取该树节点至目标点方向的不与障碍物相触碰的与该树节点间距不超过一定长度的新采样点作为新的树节点。

RRT-Connect 算法是由 Kuffner et al.<sup>[37]</sup>提出，在 RRT 算法的基础上，加入了 Connect 机制的新算法。Connect 机制是在 RRT 的以起点为根节点的随机树的基础上，新增以目标点为根节点的随机树，在树的采样拓展过程中，两棵随机树交错着进行采样拓展，直至它们能够连接在一起，此时可以通过直接连接两棵树来得到从起点至目标点的路径。

RRT\* 算法则是用于解决 RRT 算法中存在的无法对路径进行优化的问题。该算法通过在路径采样拓展的过程中，采用了 Rewire 机制，通过对采样点与树节点的重布线，持续有效地对已有路径进行优化，使得在一定迭代次数和采样点的情况下，能够得到其中的最优且可行的路径。

#### (1) 算法设计

根据 Braun et al.<sup>[38]</sup>的研究结果，由于 RRT 算法得到的路径是基于地图数据采样的结果，而非 A\* 算法中基于启发式距离得到的路径，因此在精度高的地图中，使用 RRT 算法虽然得到的结果并不一定与 A\* 算法得到的结果在质量上相当，但运算的速度会大

幅提升。因此在无人机航迹规划算法的选择上，本文选择了以 RRT 算法为算法基础进行研究。

以上述所提及的算法为基础，本文采用了基于 RRT\* 与 RRT-Connect 算法的 RRT\*-Connect 算法，来完成无人机起飞点与任务点、任务点与任务点间的航迹规划及航程计算任务。

在 RRT\*-Connect 算法中，给予所需的采样次数、选择随机采样的概率、地图信息、起点信息、目标点信息以及随机树单次拓展的最大长度，在每一次采样中，首先根据概率确定是选择从地图随机采样或是选择目标点，然后获取离随机树节点中离该采样点最近的树节点，随后根据给定的随机树单次拓展的最大长度进行重新采样，若点间长度符合长度约束则不需要重新采样，否则进行采样点进行调整，保证拓展的长度符合长度约束，然后再将采样点通过拓展的方式放入随机树中并对随机树使用 Rewire 机制进行路径优化，随后若该采样点能够放入另一棵随机树或已在另一棵随机树中，即两棵随机树能够进行连接，则将两棵树进行合并，此时即得到了最优解，然后在进入下一个循环前将两棵随机树的位置互换，确保两个树交错拓展，以上过程的伪代码如算法 3.1 所示。

在算法 3.1 中，较为关键的部分为拓展函数，即 Extend 函数，该函数首先确定新的采样点在拓展进随机树后，新的航迹是否与障碍物发生碰撞，若发生碰撞则拓展失败；随后先获取一定数量的最近点，并寻找到能使树的根节点到该点的航程最小的树节点作为母节点进行拓展，如图 3-9 所示，新拓展的节点为  $x_{new}$ ，而以  $x_{init}$  为根节点的随机树中，离该节点最近的  $x_{parent}$  则为它默认的母节点，离该点稍远的  $x_{potential\_parent\ 1}$  和  $x_{potential\_parent\ 2}$  则为其潜在母节点，由于  $x_{potential\_parent\ 1}$  至  $x_{new}$  的线路上存在障碍物，故不考虑该点，而  $x_{potential\_parent\ 2}$  与  $x_{new}$  的路线上不会产生碰撞，且连接后从根节点  $x_{init}$  到该节点的路径更短，因此将母节点修正为  $x_{potential\_parent\ 2}$ ；

在拓展完成后再以该点作为中间点，寻找是否存在一个树节点，使得根节点到新拓展的节点再到该树节点的航程，比根节点到该树节点的原本的航程要小，是则对该节点的连接方式进行调整，使起航迹的上一个节点为新拓展的节点，如图 3-10 所示，在将采样点  $x_{new}$  拓展入随机树后，存在着  $x_{potential\_child\ 1}$ 、 $x_{potential\_child\ 2}$  以及  $x_{potential\_child\ 3}$  三点潜在的子节点，但由于  $x_{init} \rightarrow x_{new} \rightarrow x_{potential\_child\ 1}$  的航程大于  $x_{init} \rightarrow x_{potential\_child\ 1}$  的航程  $x_{init} \rightarrow x_{new} \rightarrow x_{potential\_child\ 3}$  的航程大于  $x_{init} \rightarrow x_{potential\_child\ 3}$  的航程， $x_{potential\_child\ 1}$  和  $x_{potential\_child\ 3}$  将不被考虑，而  $x_{init} \rightarrow x_{new} \rightarrow x_{potential\_child\ 2}$  的航程小于  $x_{init} \rightarrow x_{potential\_child\ 2}$  的航程，故取消  $x_{potential\_child\ 2}$  与  $x_{potential\_child\ 3}$  的连接，新增  $x_{potential\_child\ 2}$  与  $x_{new}$  的连接，该机制为 RRT-Connect 算法的 Rewire 重布线机制。

该函数的伪代码如算法 3.2 所示，为便于算法 3.1 中判断采样点是否拓展入随机树成

### 算法 3.1 RRT\*-Connect 算法

**Input:**  $n$ : 采样次数;  $p$ : 选择随机采样的概率;  $\mathcal{M}$ : 地图信息;  $x_{\text{init}}$ : 起点信息;  $x_{\text{goal}}$ : 目标点信息; StepSize: 随机树单次拓展的最大长度。

**Output:**  $\Gamma$ : 从起点  $x_{\text{init}}$  到目标点  $x_{\text{goal}}$  的航迹;  $L$ : 从起点  $x_{\text{init}}$  到目标点  $x_{\text{goal}}$  的航迹的航程。

```

1: 初始化随机树  $\tau.\text{init}();$ 
2: for  $i = 1$  to  $n$  do
3:   if Random()  $\leq p$  then
4:      $x_{\text{rand}} \leftarrow \mathcal{M}.\text{sample}();$ 
5:   else
6:      $x_{\text{rand}} \leftarrow x_{\text{goal}}$ 
7:   end if
8:    $x_{\text{near}} \leftarrow \text{Near}(x_{\text{rand}}, \tau);$ 
9:    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{rand}}, x_{\text{near}}, \text{StepSize});$ 
10:  if Extend( $\tau, x_{\text{new}}, x_{\text{near}}$ )  $\neq \text{false}$  then
11:    if Connect( $\tau, x_{\text{new}}$ ) = true then
12:       $\tau = \text{MergeTree}(\tau);$ 
13:    end if
14:  end if
15:  Swap( $\tau$ );
16: end for
17:  $\Gamma \leftarrow \tau.\text{getRoute}();$ 
18:  $L \leftarrow \tau.\text{getRouteLength}();$ 

```

功，以便于接下来对随机树进行连接、合并等操作，故 Extend 函数需要将是否拓展如随机树成功的结果进行返回至上一层函数。

#### 3.3.2 基于 A\* 的动态场景下的无人机航迹规划算法设计

动态场景下的无人机航迹规划算法，是用于为单体无人机提供应对可能存在突发事件的动态环境下的措施。该算法根据已有的地图信息和配备的传感器探测到的障碍物信息，对由服务器为其规划的路径进行修订，保障其飞行的安全性与稳定性，并确保能够及时到达任务地点。由于该算法是在无人机飞行过程中运行的，需要实时对飞行航迹进行修改，因此该算法对运行时间有着较高的要求。

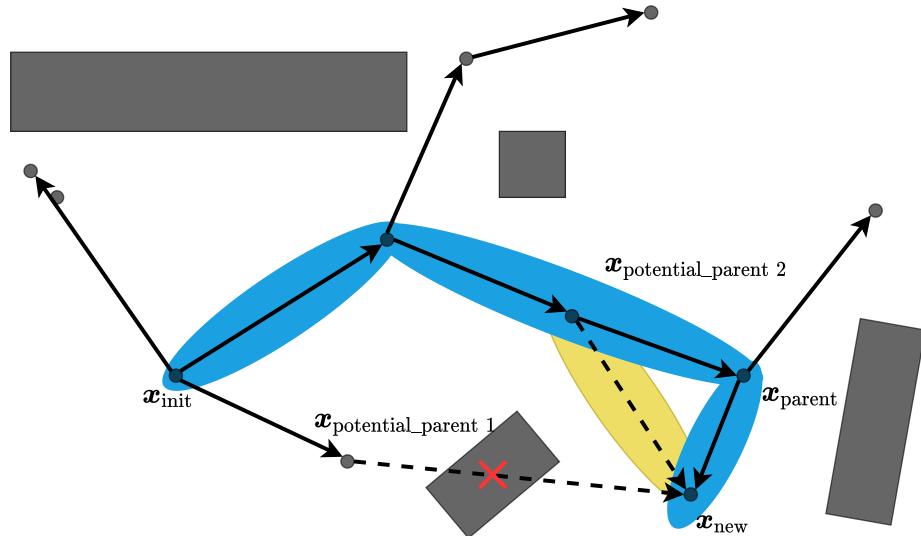


图 3-9 寻找母节点进行拓展流程

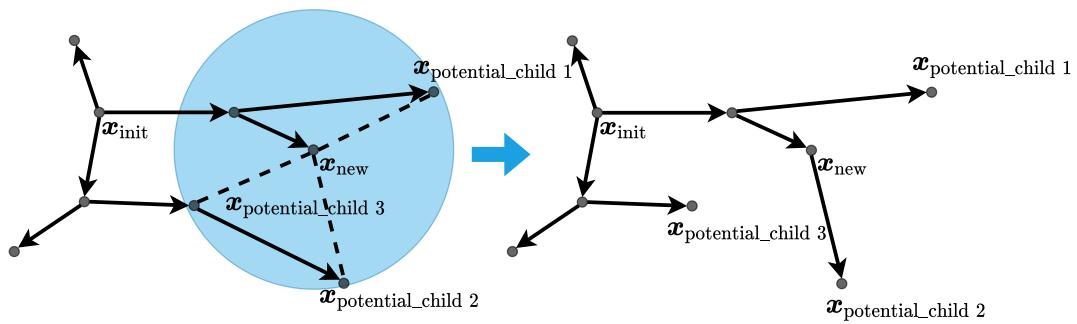


图 3-10 Rewire 重布线流程

---

### 算法 3.2 Extend 函数

**Input:**  $\tau$ : 随机树;  $x_{new}$ : 采样点;  $x_{near}$ : 最近点。

**Output:**  $S$ : 采样点拓展进随机树中是否成功。

```

1: if CollisionFree( $x_{new}$ ) then
2:    $\mathcal{X}_{near} \leftarrow \text{NearC}(\tau, x_{new});$ 
3:    $x_{min} \leftarrow \text{ChooseParent}(\mathcal{X}_{near}, x_{near}, x_{new});$ 
4:    $\tau.addNodeEdge(x_{min}, x_{new});$ 
5:    $\tau.rewire();$ 
6:   return true;
7: end if
8: return false;

```

---

### (1) 算法设计

同样根据 Braun et al.<sup>[38]</sup>的研究结果，在大小较大且精度较低或大小较小且精度较高的地图信息中寻找最短路径，使用 A\* 及其派生算法得到的路径不论是从稳定性还是从运算速度上看，都远远好于 RRT 及其派生算法。而在单无人机的实时避障算法中，由于其对航迹修订的范围仅为无人机传感器能够感知的范围，属于精度较高但地图大小较小的类型，因此适用 A\* 算法来解决该问题。

A\* 算法最初由 Hart et al.<sup>[39]</sup>提出，是将启发式的方法如 BFS 等，应用于常规的方法如 Dijkstra 算法等的一种算法。该算法通过逐步对每一个节点  $n$  进行从起点经过该点再至目标点的估计代价或距离，来不断更新优化，最终得到地图信息中从起点到目标点的最短路径，其距离的估计函数用公式表达为：

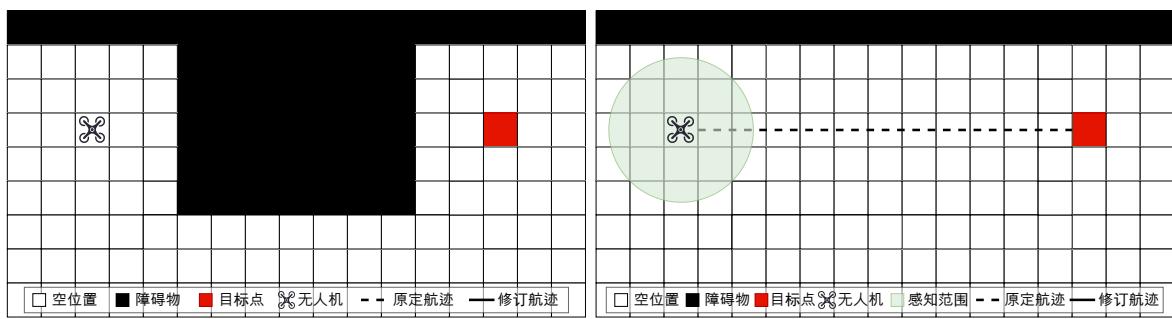
$$f(n) = g(n) + h(n)$$

其中， $f(n)$  为对于节点  $n$ ，从起点经过该点再至目标点的估计代价或距离； $g(n)$  为从起点至节点  $n$  的实际距离； $h(n)$  为从节点  $n$  至目标点的估计距离，通常用欧式距离，即  $d(\mathbf{x}_n, \mathbf{x}_0) = \| \mathbf{x}_n - \mathbf{x}_0 \|^2$ ，进行估计。

A\* 算法的流程的伪代码如算法 3.3 所示。

### (2) 算法应用流程

以图 3-11a 为例，假设实际场景中的障碍物如图 3-11a 中的分布，此时无人机以到达图中的位置，继续沿着原定的飞行航迹前往目标点，而原本为该无人机规划的飞行航迹和已知的地图信息则如图 3-11b 所示。很明显，已知的地图信息与实际信息存在可能在精细度上的差异，缺失了在飞行航迹中存在的障碍物，导致若无人机按照原本规划的飞行航迹不进行修订，则该无人机将会与障碍物发生碰撞从而造成无人机损毁及侦察任务的无法完成。



(a) 单无人机避障实际场景示例

(b) 单无人机避障规划示例

图 3-11 单无人机避障示例

### 算法 3.3 A\* 算法

**Input:**  $M$ : 地图信息;  $S$ : 起点信息;  $D$ : 目标点信息;

**Output:**  $P$ : 飞行航迹;

```

1: 初始化集合  $L \leftarrow \text{init}(S)$ ;
2: while  $\text{!}L.\text{isEmpty}()$  do
3:   估计距离最小的节点  $p_{\min\_fn} \leftarrow \text{FindMin}(L)$ ;
4:   if  $p_{\min\_fn} == D$  then
5:      $P \leftarrow \text{GetTrajectory}(p_{\min\_fn})$ ;
6:     break;
7:   end if
8:   获取四周尚未搜索过的节点  $p_{\text{unexplored}} \leftarrow \text{FindNew}(p_{\min\_fn})$ ;
9:   for all  $p_{\text{cur}}$  in  $p_{\text{unexplored}}$  do
10:    if  $\text{IsBlockedByEnv}(p_{\text{cur}})$  then
11:      continue;
12:    end if
13:    将可行节点添加入集合中  $L.\text{add}(p_{\text{cur}})$ ;
14:  end for
15: end while
16: return  $P$ ;
```

此时，该无人机可以根据其自身携带的能够探测障碍物的传感器设备，如成本高昂但精度与探测范围都很高的激光雷达、成本低廉但精度及探测范围较差的超声波探测器等，通过不断地对已有的地图信息进行修正，并运行 A\* 算法来修订原有航迹，最终平安无事地到达任务点执行侦察任务。以图 3-11 中的情景为例，无人机在按照原有航迹进行飞行前行时，传感器探测到前方存在障碍物，此时运行 A\* 算法进行航迹的修订，之后按照修订后的航迹进行飞行，如图 3-12a 所示；但此时无人机所存储的地图信息明显与图 3-11a 中不同，按新的航迹前行仍然会障碍物相撞，因此在无人机继续前行时，依然检测到了与航迹存在冲突的障碍物，此时需再运行 A\* 算法，进一步修订飞行航迹，如图 3-12b 所示；在完成对障碍物正面的绕行后，无人机继续检测到了与航迹冲突的障碍物，再一次运行了 A\* 算法对航迹进行修订，如图 3-12c 所示；不断重复以上过程，如图 3-12d、图 3-12e 和图 3-12f 所示，直至无人机能够顺利沿着修订航迹到达任务点的位置。

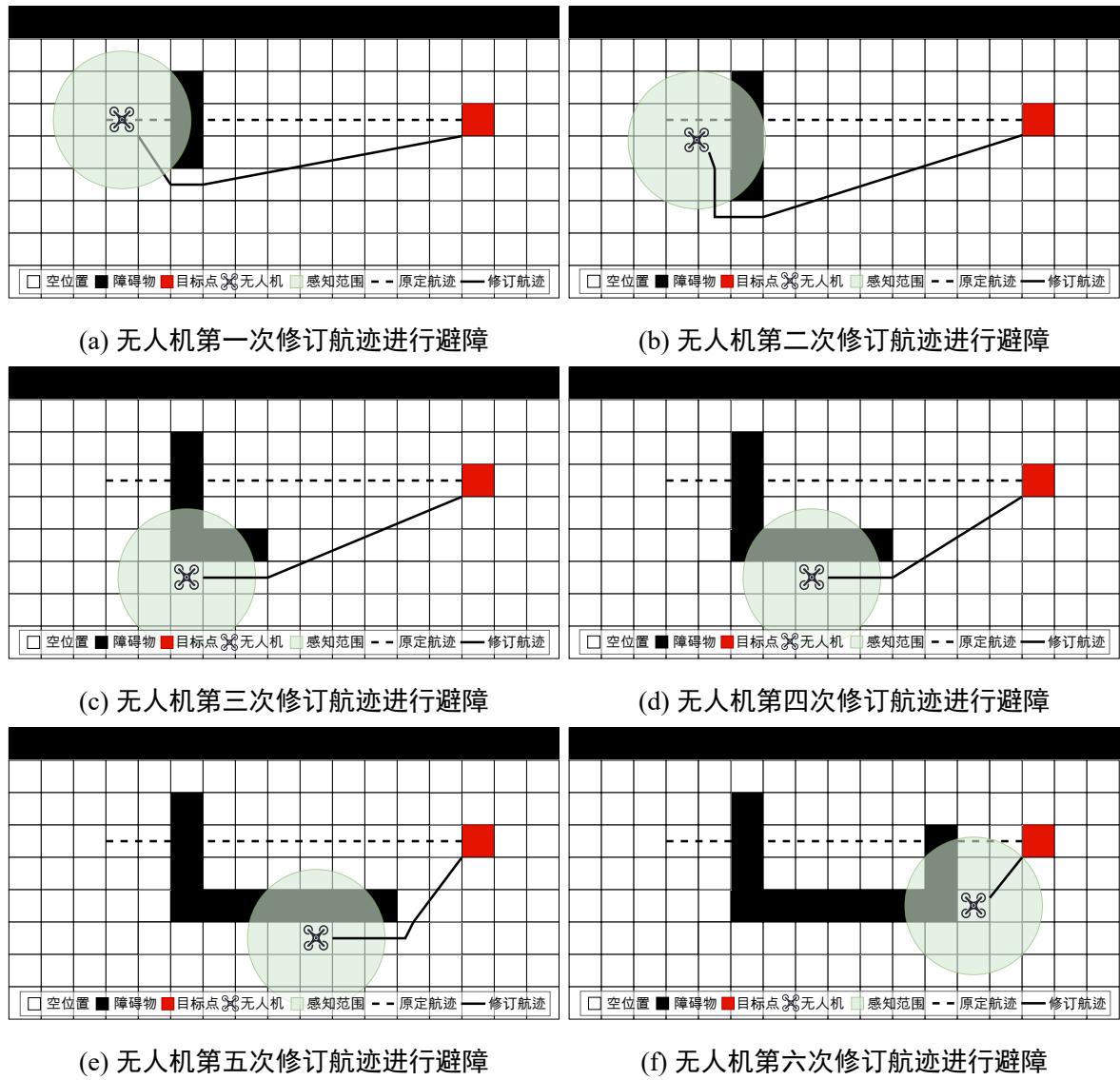


图 3-12 无人机实时修订航迹进行避障

### (3) 算法流程图

根据上述内容，本文提出的单无人机下的实时避障算法的运行流程如下，在无人机收到指令后或完成上一任务点的收集型侦察任务后，由无人机的飞行控制系统控制无人机按照所给飞行航迹进行飞行，并在飞行过程中不断地根据障碍物探测传感器的结果更新其地图数据库储存的地图信息，根据该地图信息来判断当前无人机的飞行航迹是否与已探测到的障碍物存在冲突，若存在冲突，则调用 A\* 算法，先计算当前无人机与航迹点的估计距离，再更新周围点的估计距离，之后不断更新已知点中估计距离最小的点的周围点的估计距离，直到到达目标点，此时根据得到的修订航迹对原有飞行航迹进行修订，直到到达当前任务点开始执行收集型侦察任务。总流程如图 3-13 所示。

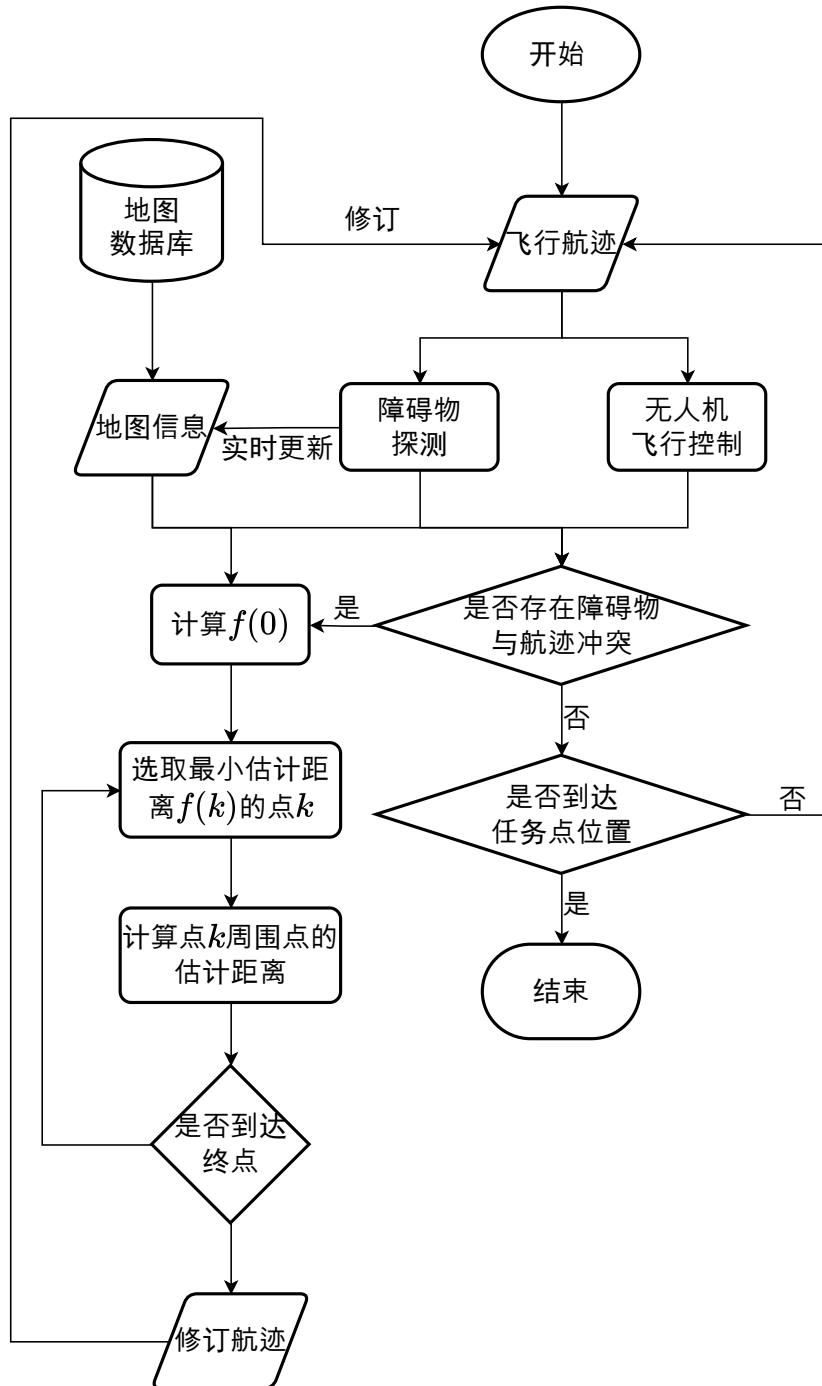


图 3-13 单无人机避障算法流程图

### 3.3.3 基于 TSAVN 的多无人机任务调度分配算法设计

K-Means 算法是一种机器学习中常用的聚类算法，其认为样本间距离越近，则这两个样本的相似性越大。其时间复杂度仅为  $t k n m$ ，其中  $t$  为迭代次数， $k$  为类别簇的个数， $n$  为样本个数， $m$  为样本的纬度，能够在很短时间内将  $n$  个样本根据欧式距离计算样本间的相似度，无需对样本进行任何标注，即可将其聚类分为  $k$  个类别。

模拟退火算法（Simulated Annealing Algorithm, SA）是一种模拟热力学中退火的过程

程，从某一较高的初始温度开始，随着算法迭代次数的增加而逐渐降低温度，结合概率跳跃的特性使得算法在温度高时具备跳出当前所在的局部最优解，在解的邻域结构中随机搜索全局最优解的启发式算法。

变邻域搜索算法（Varied Neighborhood Search Algorithm, VNS）是一种改进型的局部搜索算法，其利用了全局最优解必然在任意邻域结构中仍为局部最优解的特点，利用不同邻域动作所构成的邻域结构进行交替搜索，拓展了解的搜索范围，在集中性与疏散性之间达到了较好的平衡。

### (1) 算法设计

多无人机任务调度分配算法将分为两个阶段，分别是任务进行预分配的预分配阶段，以及基于预分配的任务信息，得到无人机所需数量及各无人机的任务有序序列的再分配阶段，如图 3-14 所示，其中，在任务预分配阶段，由于对于无人机而言，若执行相邻任务距离较远，且跨过了数个其他任务点，那么其最终的航程通常较大，而执行相邻任务距离较远，且中途没有其他任务点，那么其最终的航程通常较小，因此通过对任务点间的飞行航迹的航程进行聚类来得到初步的任务预分配结果，是合理的；之后，由于任务预分配结果仅仅只是根据航程信息对任务进行聚类操作，不具备问题所需的无人机数量、各无人机所要执行的任务有序序列，因此在任务再分配阶段，本文提出了TSAVN 算法，其流程图如图 3-14 所示。

由于直接使用任务点数据会导致解空间过大的问题，故先使用 K-Means 算法根据点间的航程数据，对任务点数据进行聚类，使得各个类内距离尽可能近，类间距离尽可能远，实现任务的预分配，从而能够有效地降低各个部分的解空间，便于并行、高效地运行算法。

本小节以最小化航迹距离作为聚类依据，自下而上对各任务点进行聚类。同时为尽可能保证同一个簇里航迹之和最小，在聚类完成后，将结果放入局部优化器中生成各个簇中最优的车辆安排、配置及路径等数据，即初始解生成。

初始解生成的伪代码如算法3.4所示：

为了增加算法收敛至全局最优解的概率，以及尽可能避免算法生成的解质量较差的情况发生，本小节选取了 3 种不同的解的变化策略，以及 5 种基于这 3 种方式的邻域结构来对全局解中不同任务簇的分类进行一定的调整。

第一种变化策略为任务点的转移，将一个簇中的任务点转移至另一个簇中，过程如图3-15所示。基于该变化策略，本文提出了两种随机性不同的邻域结构。一种在两个簇中的选取及待转移的任务点的选取中全部依赖随机，即随机性较高，其动作过程的伪代码如算法3.5所示；另一种则随机选取一个簇并随机选取簇中的一个任务点，将其转移

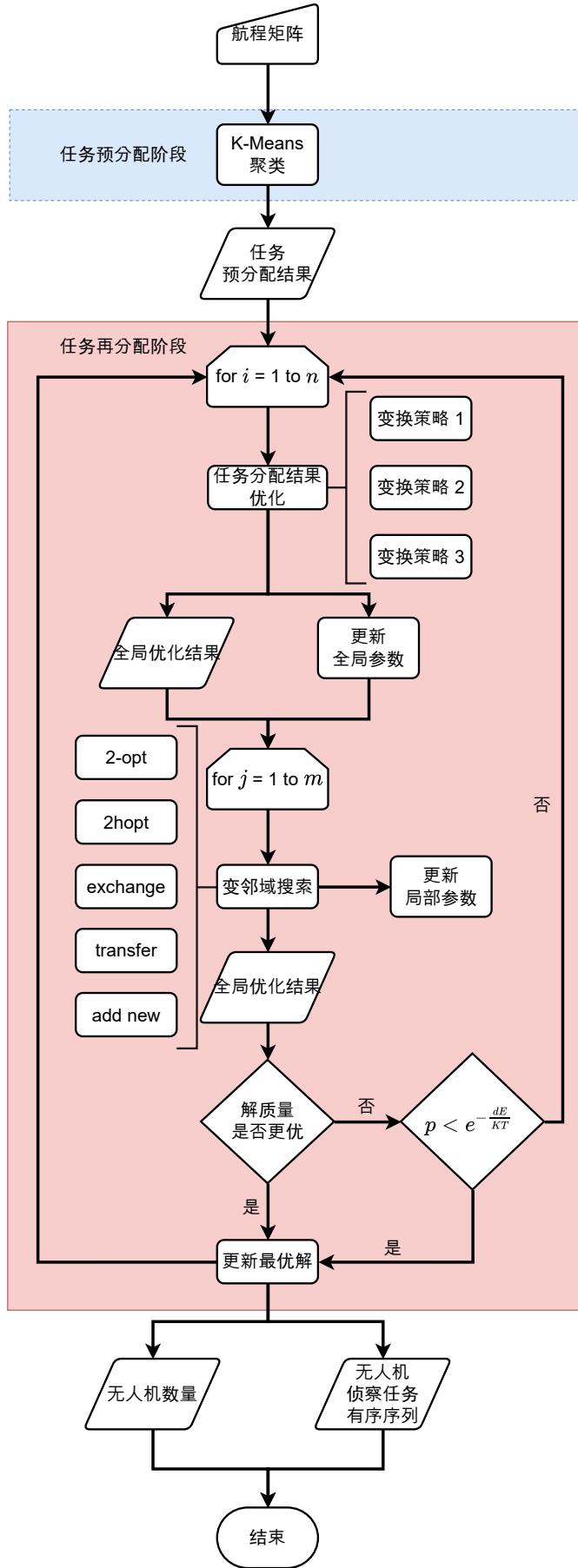


图 3-14 多无人机任务调度分配阶段示意图

### 算法 3.4 初始解生成器

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $M$ : 所有任务点的任务数据集;  $C$ : 所有车型数据集;

**Output:**  $S$ : 初始解;

```

1: 初始化类簇标签  $L$ ;
2: repeat
3:   获取距离矩阵  $D \leftarrow \text{CalcDistanceMatrix}(V, L)$ ;
4:   获取所有拥有最小距离的两个簇的组合  $C \leftarrow \text{FindMinDisCombs}(L, D)$ ;
5:   for  $i, j \in C$  do
6:     if  $\text{notReachCriteria}(i, j, L)$  then
7:       更新标签  $L \leftarrow \text{merge}(i, j, L)$ ;
8:     end if
9:   end for
10:  until  $\text{AllReachCriteria}(L)$ 
11:   $S \leftarrow \text{PartialSolver}(L, V, M, C)$ ;

```

至离该簇最近的另一个簇中，随机性相对于前者更低，但相应的计算时间更多，其动作过程的伪代码如算法3.6所示。

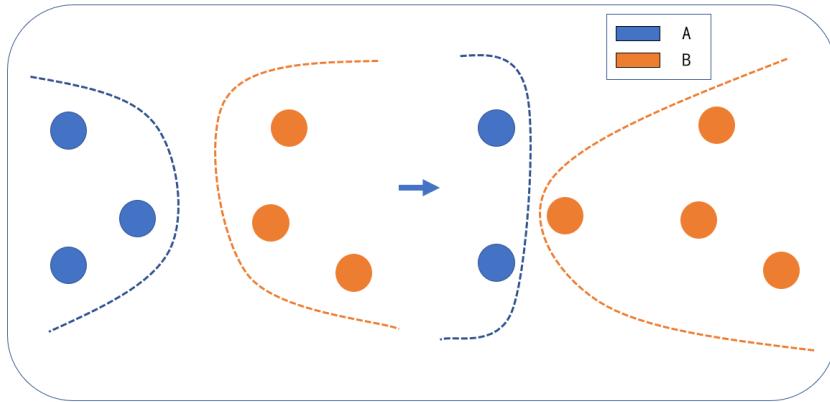


图 3-15 变化策略 1

第二种变化策略为任务点的交换，在两个不同的簇中各自选择一个任务点进行交换，其过程如算法3-16所示。基于该变化策略，本文提出了两种随机性不同的邻域结构，一种在两个簇的选取及各自的待交换的任务点的选取中全部依赖随机，即随机性较高，其动作过程的伪代码如算法3.7所示；另一种则随机选取一个簇，并将离它最近的簇作为另一个簇，分别随机选取簇中的任务点进行交换，随机性相对于前者更低，但相应的

### 算法 3.5 随机转移算子

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $L$ : 先前的类簇标签;

**Output:**  $L^*$ : 新的类簇标签;

- 1: 随机寻找两个类簇  $\ell_1, \ell_2 \leftarrow \text{FindTwoDifferentLabel}(L)$ ;
- 2: 在类簇  $\ell_1$  中寻找一个随机的任务点  $v_1$ ;
- 3: 更新任务点  $v_1$  的标签从  $\ell_1$  变为  $\ell_2$ ;
- 4:  $L^* \leftarrow L$ ;

### 算法 3.6 就近转移算子

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $L$ : 先前的类簇标签;

**Output:**  $L^*$ : 新的类簇标签;

- 1: 随机寻找一个类簇  $\ell_1 \leftarrow \text{FindLabel}(L)$ ;
- 2: 寻找一个距离  $\ell_1$  最近的类簇  $\ell_2 \leftarrow \text{FindNearestLabel}(L, \ell_1)$ ;
- 3: 在类簇  $\ell_1$  中寻找一个随机的任务点  $v_1$ ;
- 4: 更新任务点  $v_1$  的标签从  $\ell_1$  变为  $\ell_2$ ;
- 5:  $L^* \leftarrow L$ ;

计算时间会更多，其动作过程的伪代码如算法3.8所示。

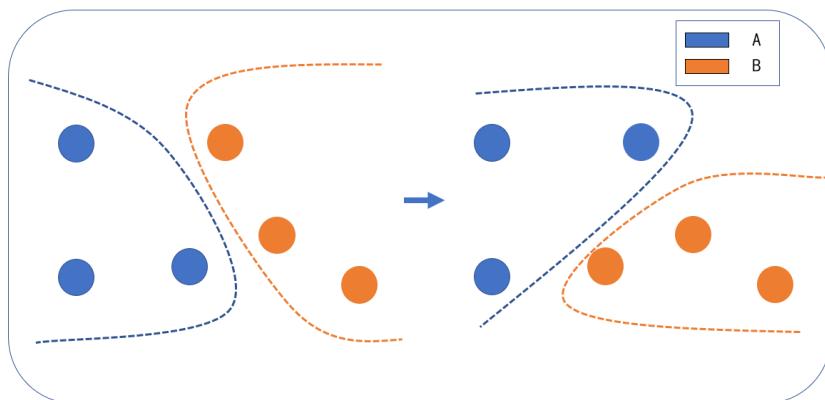


图 3-16 变化策略 2

第三种变化策略则是生成新的簇，选取一个任务点作为一个新的类，其过程如算法3-17所示。基于该变化策略，本文提出了一种邻域结构，该邻域结构为随机选取一个任务点，若该任务点所属的簇包含的任务点不止该任务点，则将该任务点作为一个新的簇，其动作过程的伪代码如算法3.9所示；

为了能够在每一个任务类簇中得到最优解，以获得全局的最优解，同时避免每个任务类簇中发生得到质量较差的解的情况，本小节选取了 5 种不同的邻域结构来对无人机

### 算法 3.7 随机交换算子

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $L$ : 先前的类簇标签;

**Output:**  $L^*$ : 新的类簇标签;

- 1: 随机寻找两个类簇  $\ell_1, \ell_2 \leftarrow \text{FindTwoDifferentLabel}(L)$ ;
- 2: 在类簇  $\ell_1$  中寻找一个随机的任务点  $v_1$ ;
- 3: 在类簇  $\ell_2$  中寻找一个随机的任务点  $v_2$ ;
- 4: 更新任务点  $v_1'$  的标签从  $\ell_1$  变为  $\ell_2$ ;
- 5: 更新任务点  $v_2'$  的标签从  $\ell_2$  变为  $\ell_1$ ;
- 6:  $L^* \leftarrow L$ ;

### 算法 3.8 就近交换算子

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $L$ : 先前的类簇标签;

**Output:**  $L^*$ : 新的类簇标签;

- 1: 随机寻找一个类簇  $\ell_1 \leftarrow \text{FindLabel}(L)$ ;
- 2: 寻找一个距离  $\ell_1$  最近的类簇  $\ell_2 \leftarrow \text{FindNearestLabel}(L, \ell_1)$ ;
- 3: 在类簇  $\ell_1$  中寻找一个随机的任务点  $v_1$ ;
- 4: 在类簇  $\ell_2$  中寻找一个随机的任务点  $v_2$ ;
- 5: 更新任务点  $v_1'$  的标签从  $\ell_1$  变为  $\ell_2$ ;
- 6: 更新任务点  $v_2'$  的标签从  $\ell_2$  变为  $\ell_1$ ;
- 7:  $L^* \leftarrow L$ ;

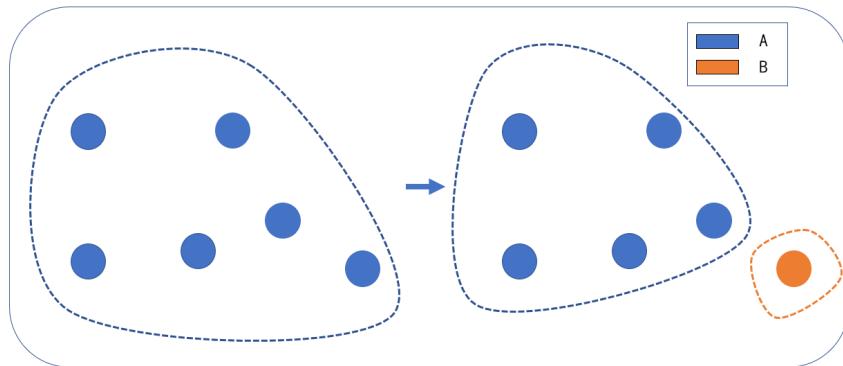


图 3-17 变化策略 3

的侦察任务执行序列进行调整和优化。

第一种邻域结构为 2-opt 方法<sup>[40]</sup>, 如图 3-18 所示。2-opt 方法是选取两任务点  $i$  和  $k$ , 将  $i$  之前的任务序列保持不变, 将  $k$  之后的任务序列保持不变, 将  $i, k$  及其之间的任务的顺序进行翻转, 来得到新的任务序列。

### 算法 3.9 新簇生成算子

**Input:**  $V$ : 无人机起飞点与任务点的点集;  $L$ : 先前的类簇标签;

**Output:**  $L^*$ : 新的类簇标签;

- 1: 随机寻找一个类簇  $\ell_1 \leftarrow \text{FindLabel}(L)$ ;
- 2: 在类簇  $\ell_1$  中寻找一个随机的任务点  $v_1$ ;
- 3: 设置  $v_1$  的类簇标签为一个新的标签;
- 4:  $L^* \leftarrow L$ ;

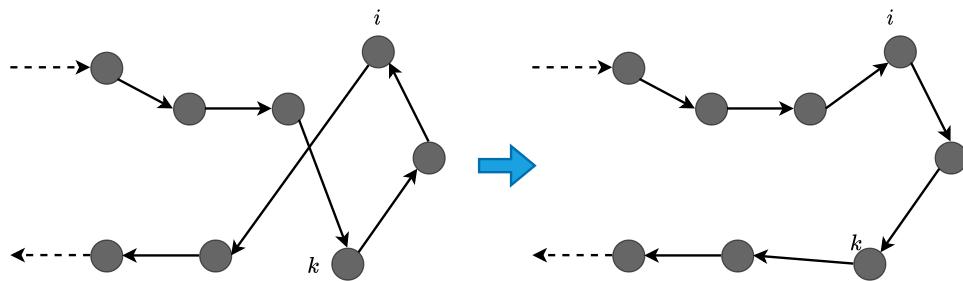


图 3-18 2-opt 邻域结构

第二种邻域结构为 2hopt 方法, 如图 3-19 所示。2hopt 方法是选取两任务点  $i$  和  $k$ , 将  $i$  之前的任务序列保持不变, 将  $i$  与  $k$  之间的任务保持不变, 将  $k$  之后的任务序列保持不变, 仅将任务  $i$ 、 $k$  进行交换, 来得到新的任务序列。

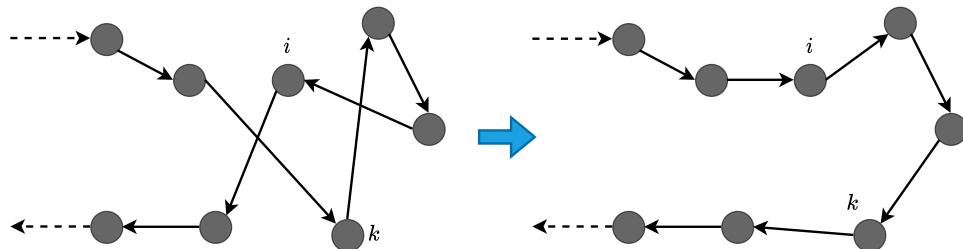


图 3-19 2hopt 邻域结构

第三种邻域结构为 exchange 方法, 如图 3-20 所示。exchange 方法是对于两个无人机的任务序列而言, 各选择其中的一个任务点  $i$  和  $k$ , 将任务点  $i$  和任务点  $k$  进行交换, 其余任务点及序列保持不变, 以此来得到两条新的任务序列。

第四种邻域结构为 transfer 方法, 如图 3-21 所示。transfer 方法是选取两任务点  $i$  和  $k$ , 将任务点  $i$  移出原任务序列, 并将其插入至任务点  $k$  后, 其余任务点及序列保持不变, 以此来得到两条新的任务序列。

第五种邻域结构为 add new 方法, 如图 3-22 所示。add new 方法是选取一个任务点  $i$ , 将其从原任务序列中移除, 其余任务点及序列保持不变, 并将其作为新的无人机的任

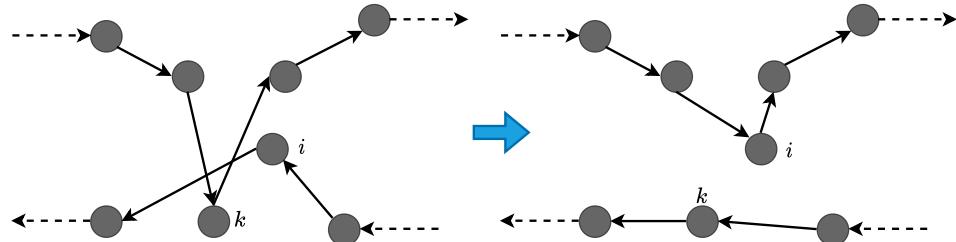


图 3-20 exchange 邻域结构

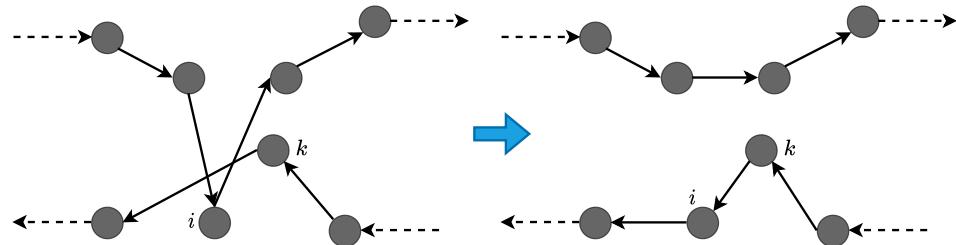


图 3-21 transfer 邻域结构

务序列，以此来得到两条新的任务序列。

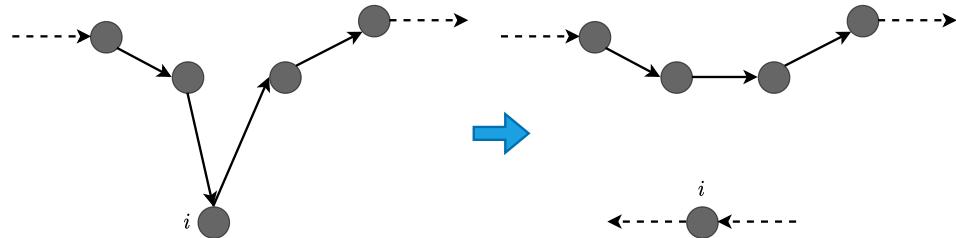


图 3-22 add new 邻域结构

由于模拟退火算法在运行中会存在初期温度高，同时未达到局部最优解的情况，以及末期温度低，同时达到局部最优解的情况，导致其非更优解接受概率的利用程度不高，所以模拟退火算法的性能仍有待提高。

为提高模拟退火算法的性能，更充分地利用其跳出局部最优解的机制，此处采用了自适应温度机制取代了原有的温度机制。

此处引入  $r_i$  作为算法第  $i$  次运行后没有得到更优解的次数，即需要计算非更优解接受概率的次数，当直接得到了更优解时  $r_i = 0$ ，算法未运行时  $r_0 = 0$ ，即

$$r_i = \begin{cases} r_{i-1} + 1 & \text{if } \Delta E > 0 \\ r_{i-1} & \text{if } \Delta E = 0 \\ 0 & \text{if } \Delta E < 0 \end{cases}$$

同时使用迭代次数  $N$  来控制整个算法的逻辑。与模拟退火算法相比，其总的求解

次数仅由  $N$  决定，而模拟退火算法的总求解次数为  $N \cdot \log_r \frac{T_0}{T_{\text{end}}}$ 。

设定一个最低温度  $T_{\min}$ ，升温速率  $\rho$  以及温控参数  $\delta$ ，使得第  $i$  次计算温度  $T_i$  时使用如下公式：

$$T_i = T_{\min} + \rho \cdot \ln(1 + \frac{r_i}{\delta})$$

如此便能够在能够寻求到最优解的情况下维持低温，在到达局部最优解、需要到达新的局部最优解时能够通过升高温度来跳出局部最优解，从而使算法具有更好的鲁棒性。

### 3.3.4 多无人机航迹规划及任务分配算法流程图

由上述内容可知，本文提出的航迹规划及任务分配算法的流程如下，在任务开始前，服务器分别从贮存任务数据的数据库和地图信息的数据库中获取任务数据、地图数据，再将以上数据一齐输入进 RRT\*-Connect 算法中，得到起飞点与任务点、任务点与任务点之间的无人机飞行航迹和飞行航程信息，将航程信息矩阵输入进 K-Means 算法中对任务进行预分配，再将任务的预分配结果、航程矩阵以及无人机的最大航程一齐输入进 TSAVN 算法中，便得到了完成任务所需的无人机数量以及每个无人机需完成的收集型侦察任务有序序列，再根据该序列从此前得到的点间飞行航迹中进行拼接得到各个无人机的飞行航迹，再将以上信息作为指令发送给无人机，由无人机完成所给任务。该算法的总流程如图 3-23 所示。

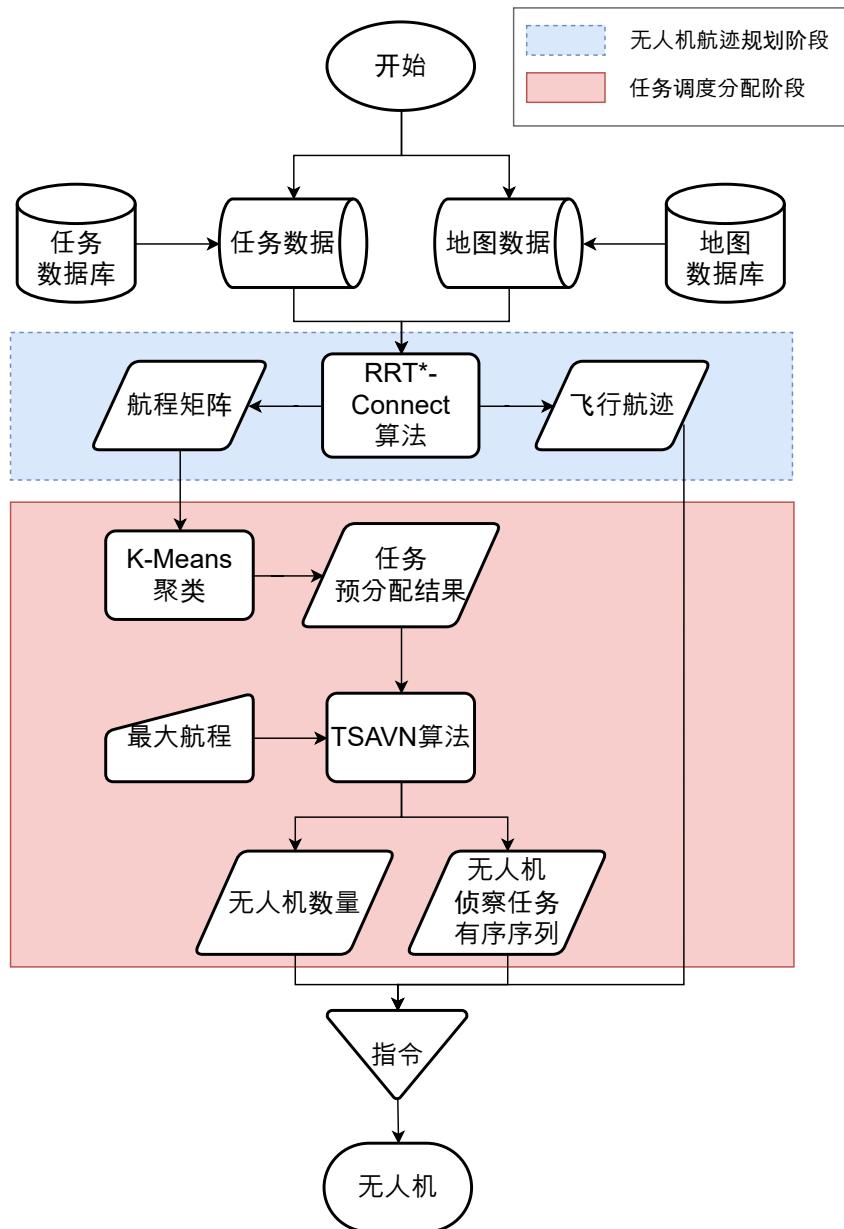


图 3-23 航迹规划及任务分配流程图

### 3.4 本章小结

本章主要对面向边缘计算的任务计算资源调度优化问题、多无人机的航迹规划及任务分配问题分别进行分析，并按照分析内容提出对应的求解算法，并详细说明了各个算法的关键机制、运行流程等内容。

## 第 4 章 算法仿真实验

### 4.1 任务计算资源调度算法仿真实验

为验证任务计算资源调度算法 ASA 和 FD 的有效性和优越性，本节完成并应用了任务生成器来随机生成包含着任务起始时间、所需计算资源、在不同设备上的计算时间、上传时延等的任务数据，并与采用了与 ASA 算法相同的邻域结构的 SA 算法进行测试对比。

#### 4.1.1 参数设置

在设置任务生成器的参数时，本节使用了表 4-1 中的参数来生成近似真实应用场景下的计算型任务数据，其中任务总数  $N$  为 50，每个任务的起始时间  $S_i$  为上下界分别为 1000、0 的离散型均匀分布，每个任务的计算所需资源  $R_i$  为上下界分别为 100、1 的离散型均匀分布，每个任务在无人机上计算需要花费的时间  $U_i$  为上下界 300、1 的离散性均匀分布，每个任务在服务器上计算需要花费的时间  $C_i$  为上下界  $U_i - 1$ 、0 的离散性均匀分布，上传时延  $D_i$  为上下界 100、1 的离散型均匀分布。

表 4-1 资源分配算法任务生成参数

参数	含义	值
$N$	任务总数	50
$S_i$	第 $i$ 个任务的起始时间	[0, 1000]
$R_i$	第 $i$ 个任务所花费的资源	[1, 100]
$U_i$	第 $i$ 个任务在无人机上计算需要花费的时间	[1, 300]
$C_i$	第 $i$ 个任务在服务器上计算需要花费的时间	[0, $U_i - 1$ ]
$D_i$	第 $i$ 个任务上传至其他设备的时延	[1, 100]

测试中采用的实验参数如表 4-2 所示，其中无人机的数量  $N_u$  为 30，每架无人机最大可用的计算资源  $R_u$  为 100，服务器的数量  $N_c$  为 1，每台服务器最大可用的计算资源  $R_c$  为 2000。

本文所提出的 ASA 算法以及对比算法 SA 的参数设置如表 4-3 所示，即对于 SA 来

表 4-2 任务计算资源调度算法实验参数设置

参数	含义	值
$N_u$	无人机数量	30
$R_u$	无人机最大可用计算资源	100
$N_c$	服务器数量	1
$R_c$	服务器最大可用计算资源	2000

表 4-3 算法参数设置

SA			ASA		
参数	含义	值	参数	含义	值
$T_{\text{init}}$	初始温度	1,000	$N$	总迭代次数	1000
$T_{\text{end}}$	终止温度	1	$T_{\min}$	最低温度	1
$\rho_{\text{cool}}$	冷却速率	0.9	$\rho$	升温速率	1
$n_T$	每个温度下的迭代次数	15	$\delta$	温控参数	0.01

说，其初始温度  $T_{\text{init}}$  为  $10^4$ ，终止温度  $T_{\text{end}}$  为  $10^{-1}$ ，冷却速率  $\rho_{\text{cool}}$  为 0.9，每个温度下的迭代次数  $n_T$  为 10，而对于 ASA 来说，其总迭代次数  $N$  为 1000，最低温度  $T_{\min}$  为 1，升温速率  $\rho$  为 1，温控参数  $\delta$  为 1。

#### 4.1.2 算法性能对比实验结果

为了能够直观地体现 ASA 算法的有效性及优越性，本节统计了十次性能测试的性能指标，其具体的性能结果如表 4-4 所示。

根据表 4-4 的试验结果，在目标值方面，本文所提出的 ASA 算法相较其他对比算法，得到的解质量更好，这体现了 ASA 算法在解决无人机任务执行开始前的任务计算资源调度静态问题时的有效性；在运行时间方面，ASA 算法相较于 SA 算法有着更少的算法运行时间，同时 FD 算法的运行时间与其他算法的运行时间更是有着明显的差距，而 FD 算法虽然得到的解的质量较差，但由于其用于无人机任务执行开始后的动态问题，相较

表 4-4 资源分配算法性能比较结果

测试数据	SA		ASA		FD	
	运行时间 (秒)	目标值	运行时间 (秒)	目标值	运行时间 (秒)	目标值
1	203.324	30372	198.497	30313	0.038	30480
2	223.149	29809	177.082	29809	0.039	29809
3	216.077	31359	174.673	31359	0.039	31408
4	234.679	31732	172.580	31686	0.041	32188
5	227.150	31067	197.496	30597	0.040	31116
6	238.868	30780	185.066	30888	0.041	31272
7	209.149	31986	187.116	31309	0.042	31986
8	252.184	32551	179.438	32467	0.041	32683
9	221.427	29638	179.438	29152	0.039	29915
10	195.529	30699	160.609	30606	0.040	30883
<b>平均值</b>	222.238	30956.4	184.156	<b>30818.6</b>	<b>0.040</b>	31174

于目标性能，其运行时间更为重要。因此本文的 ASA 算法与 FD 算法在该问题下的优越性明显。

#### 4.2 静态场景下的无人机航迹规划算法仿真实验

为验证本文所提出的静态场景下的无人机航迹规划算法 RRT\*-Connect 的有效性及优越性，本节完成并应用了地图生成器来随机生成包含着不同障碍物、起终点的地图信息进行测试，并与其他同样能够用于航迹规划的 Basic RRT 算法、基于概率的 RRT 算法、RRT-Connect 以及 RRT\* 算法进行测试比较。

#### 4.2.1 仿真参数设置

在设置地图生成器的参数时，本节使用了表 4-5 中的参数来生成近似真实城市环境的三维地图，其中该地图的大小为  $B_x \times B_y \times B_z$ ，每个位置的生成建筑物的概率  $P_b$  为 10%，建筑物的高度分布为均值  $\mu = B_z/2$  的均值分布，建筑物所占地面面积  $S_b$  为固定为 1，共生成  $N$  个测试地图。

表 4-5 静态场景下的无人机航迹规划算法仿真环境参数

参数	含义	值
$B_x$	地图在 $x$ 轴上的长度	100
$B_y$	地图在 $y$ 轴上的长度	100
$B_z$	地图在 $z$ 轴上的长度	10
$P_b$	建筑物的生成概率	10%
$\mu$	建筑物高度的分布均值	50
$S_b$	建筑物所占地面面积	1
$N$	进行测试的地图数量	100

本文提出的 RRT\*-Connect 算法与对比算法 Basic RRT 算法、基于概率的 RRT 算法、RRT-Connect 算法与 RRT\* 算法均采用相同的参数设置，即表 4-6 中所示，即对于上述每一个算法生成的随机树来说，其可容许的最大的步进拓展的长度  $S$  为 1，在对地图进行采样时选取目标点的概率  $P_t$  为 0.2。

表 4-6 静态场景下的无人机航迹规划算法参数设置

参数	含义	值
$S$	步进长度	1
$P_t$	采样时选取目标点的概率	0.2

#### 4.2.2 仿真实验结果

为了直观地体现 RRT\*-Connect 算法的有效性，本节选取了其中一次测试的结果并使其可视化，其建筑分布、飞行航迹等信息如图 4-1 所示，从图中可以明显看出，基于

RRT\*-Connect 的航迹规划算法在模拟城市的三维环境下成功地躲避了环境中的建筑物，并成功地在较短航程下到达了目标点。

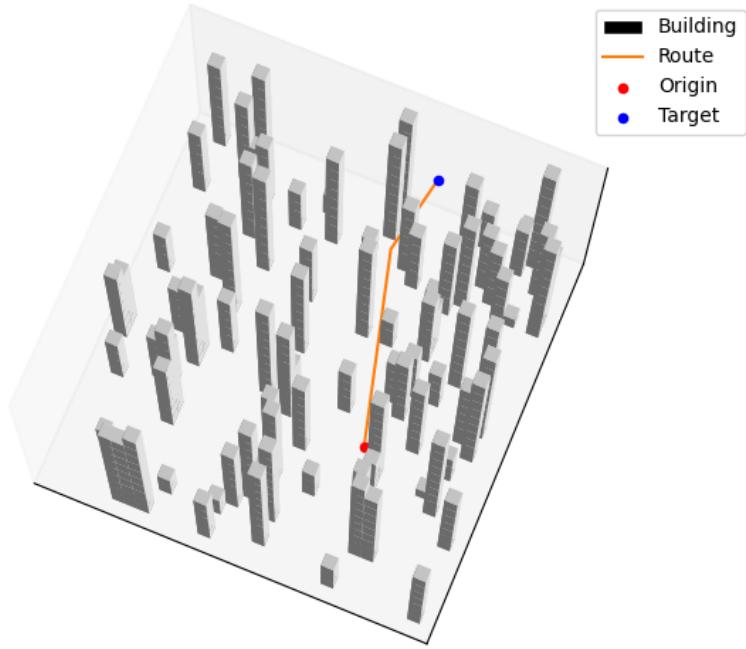


图 4-1 基于 RRT\*-Connect 的航迹规划算法在三维地图下的仿真结果

#### 4.2.3 算法性能对比实验结果

为了直观地体现 RRT\*-Connect 算法的优越性，本节统计了性能测试中得到的性能指标，具体结果如表 4-7 所示。

表 4-7 航迹规划算法性能比较结果

算法	目标值	运行时间
Basic RRT	105.313	4.008
基于概率的 RRT	87.878	3.710
RRT-Connect	83.699	<b>0.076</b>
RRT*	81.686	0.498
RRT*-Connect	<b>79.312</b>	0.132

根据表 4-7 的试验结果，在目标值方面，本文所提出的 RRT\*-Connect 算法相较其他对比算法，得到的解质量更好，这体现出了本文 RRT\*-Connect 算法在解决城市密集障

碍物环境下静态场景下的无人机航迹规划问题时的优越性；在运行时间方面，本文算法相较大部分对比算法而言有着更短的平均计算时间，仅相对于 RRT-Connect 在计算时间方面有着微小的差距，这是由于 RRT\*-Connect 算法采用的 Rewire 机制导致算法计算量增加，进而造成计算时间的略微增加，但其求解结果却有了明显的提升。于此同时，由于本文研究的无人机航迹规划算法是在无人机任务执行开始之前，在服务器上运行生成航迹规划方案，相较于算法运行时间，其目标性能更为重要，因此本文的 RRT\*-Connect 算法在该场景下的优越性更加明显。

### 4.3 动态场景下的无人机航迹规划算法仿真实验

为验证无人机实时避障算法 A\* 的有效性及优越性，本节完成并应用了地图生成器来随机生成包含着不同障碍物、起终点的地图信息进行测试，并与同样能够应用于避障航迹生成的 Basic RRT 算法、基于概率的 RRT 算法、RRT-Connect、RRT\* 以及 RRT\*-Connect 算法进行测试比较。

#### 4.3.1 仿真实验参数

与静态场景下的航迹规划算法所解决的问题不同，动态场景下的无人机航迹规划算法所解决的动态场景下的无人机航迹规划问题的地图数据相对更小，且其实时性使得对算法的运行速度有着很高的要求，因此本文在设置地图生成器的参数时，为便于观察可视化结果，采用了表 4-8 中的参数来随机生成包含不同障碍物、起终点的二维地图信息来生成算法的测试数据，其中，该地图大小为  $B_x \times B_y$ ，每个位置生成障碍物的概率  $P_b$  为 10%，共生成  $N$  个测试地图。

表 4-8 动态场景下的无人机航迹规划算法仿真环境参数

参数	含义	值
$B_x$	地图在 $x$ 轴上的长度	30
$B_y$	地图在 $y$ 轴上的长度	30
$P_b$	障碍物的生成概率	10%
$N$	进行测试的地图数量	100

本文所使用的 A\* 算法与对比算法 Basic RRT 算法、基于概率的 RRT 算法、RRT-Connect 算法、RRT\* 算法与 RRT\*-Connect 算法均采用相同的参数设置，即表 4-9 中所

示,对于Basic RRT及其他基于Basic RRT的算法,采用RRT及其衍生算法下的参数,即其随机树的可容许最大步进拓展长度 $S$ 为1,在对地图进行采样时选取目标点的概率 $P_t$ 为0.2;对于A\*算法而言,仅需设置其每一步的步进长度 $S$ 为1。

表 4-9 动态场景下的无人机航迹规划算法参数设置

RRT 及其衍生算法			A*		
参数	含义	值	参数	含义	值
$S$	步进长度	1	$S$	步进长度	1
$P_t$	采样时选取目标点的概率	0.2			

#### 4.3.2 仿真实验结果

为了能够直观地体现A\*的有效性,本节选取其中一次测试的结果并将其可视化,其障碍物分布、避障航迹如图4-2所示,从图中能够看出A\*算法成功地根据已知障碍物信息规划出了一条能够躲避障碍物的飞行航迹。

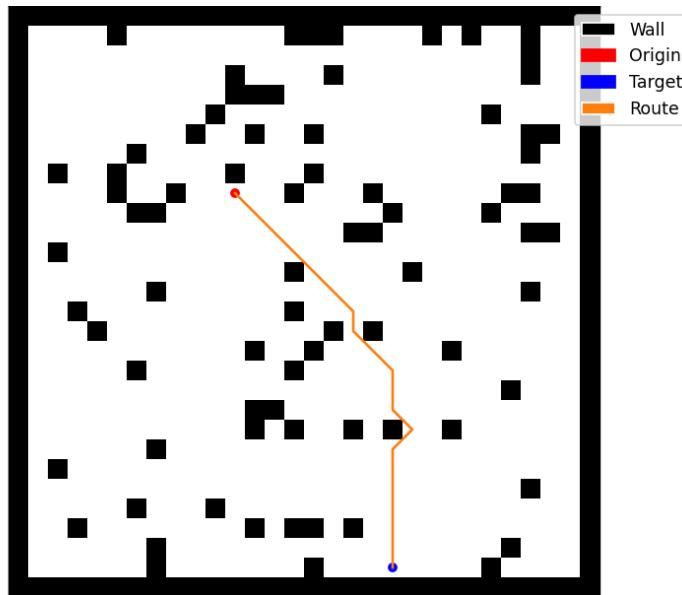


图 4-2 A\* 算法在二维地图下的仿真结果

#### 4.3.3 算法性能对比实验结果

为了直观地体现A\*算法的优越性,本节统计了性能测试中得到的性能指标,具体的性能结果如表4-10所示,

表 4-10 实时避障算法性能比较结果

算法	目标值	运行时间
Basic RRT	29.459	0.070
基于概率的 RRT	24.876	0.077
RRT-Connect	24.553	0.017
RRT*	25.893	0.096
RRT*-Connect	24.183	0.026
A*	<b>18.380</b>	<b>0.006</b>

根据表 4-10 的实验结果，在目标值方面，本文所使用的 A\* 算法相较其他对比算法，得到的解质量更好，这体现了本文 A\* 在解决城市密集障碍物环境下的动态场景下的无人机航迹规划问题的有效性；在运行时间方面，本文相较其他对比算法而言同样有着更少的平均计算时间。综上所述，本文的 A\* 算法与其他算法相比具有显著的优越性。

#### 4.4 多无人机任务分配算法仿真实验

为验证无人机任务分配算法 TSAVN 的有效性及优越性，考虑到无人机任务分配问题可以抽象为 VRP 问题，本节采用了 Gehring et al.<sup>[41]</sup> 的 1000 节点的硬时间窗 VRP 测试集中的 *c1\_10\_1* 数据的前 100 个节点的位置信息作为测试数据，并加入了无人机最大航程的约束，并与同样能够用于无人机任务分配的基于 RFCS (Routing First and Cluster Second, 先规划后分配，在本问题中为先规划任务序列，再将该任务序列分配至无人机中) 的遗传算法与模拟退火算法进行测试比较。

##### 4.4.1 实验参数设置

本节采用的实验参数如表 4-11 所示，其中测试中采用的无人机飞行速度  $V_u$  为 1，无人机最大航程  $\ell_{\max}$  为 3000，在目标函数中，无人机数量的惩罚系数  $P_u$  为 1000。

本文所提出的 TSAVN 算法与对比算法 GA、SA 算法采用的参数设置如表 4-12 中所示，即对于 GA 来说，其种群大小  $S_p$  为 100，最大迭代次数  $N$  为 1000，染色体的变异概率  $P_m$  为 0.01，对于 SA 来说，其初始的温度  $T_{\text{init}}$  为 100，最低温度  $T_{\min}$  为  $10^{-7}$ ，马尔可夫链长度  $n_T$  为 300，对于 TSVAN 来说，最大迭代次数  $N$  为 1000，终止温度  $T_{\text{end}}$  为 1 升温速率  $\rho$  为 1，温控参数  $\delta$  为 1。

表 4-11 多无人机任务分配算法实验参数设置

参数	含义	值
$V_u$	无人机飞行速度	1
$\ell_{\max}$	无人机最大航程	3000
$P_u$	无人机数量惩罚系数	1000

表 4-12 无人机任务分配算法参数设置

GA			SA			TSAVN		
参数	含义	值	参数	含义	值	参数	含义	值
$S_p$	种群大小	100	$T_{\text{init}}$	初始温度	100	$N$	总迭代次数	1000
$N$	迭代次数	1000	$T_{\text{end}}$	终止温度	$10^{-7}$	$T_{\min}$	最低温度	1
$P_m$	变异概率	0.01	$n_T$	每个温度下的 迭代次数	300	$\rho$	升温速率	1
			冷却速率	0.9		$\rho$	$\delta$	温控参数
								1

#### 4.4.2 仿真实验结果

为了直观地体现 TSAVN 算法的有效性，本节选取其中一次测试的结果并使其可视化，其所需无人机的数量、每个无人机的任务序列等信息如图 4-3 所示，其中每一个叉为任务点的位置，每一种颜色的直线为每一个无人机的任务执行序列。各个无人机之间的规划路径不存在明显交叉情况，表明所提出的 TSAVN 算法在解决多无人机任务分配问题方面的有效性。

#### 4.4.3 算法性能对比实验结果

为了直观地体现 TSAVN 算法的优越性，本节统计了性能测试中得到的性能指标，具体结果如表 4-13 所示。

根据表 4-13 的试验结果，在目标值方面，本文所提出的 TSAVN 算法相较其他对比算法，得到的解质量更好，这体现了本文 TSAVN 在解决基于城市交通侦察场景下的无人机任务分配问题的有效性；在运行时间方面，本文算法相较其他对比算法而言同样有

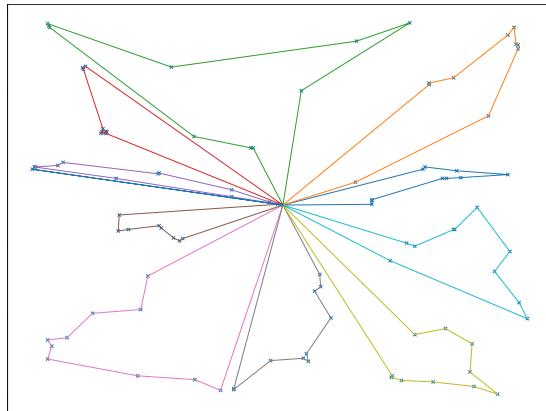


图 4-3 基于 TSAVN 的多无人机任务分配算法仿真结果

表 4-13 任务分配算法性能比较结果

运行次数	GA		SA		TSAVN	
	运行时间 (秒)	目标值	运行时间 (秒)	目标值	运行时间 (秒)	目标值
1	2086.7	66764.45	222.1	20759.26	9.731	18172.71
2	1189.4	55713.22	204.2	19708.96	9.690	17882.67
3	1181.4	52783.19	319.3	19358.90	9.155	17812.66
4	1185.5	52623.57	127.6	20409.14	8.901	18052.71
5	1182.3	55103.41	533.7	19009.00	8.200	17772.66
6	1179.9	53380.95	541.9	18658.84	9.291	17962.69
7	1177.1	55751.29	142.6	20059.25	9.759	17862.59
8	1176.2	54941.32	205.4	19709.03	9.200	17772.56
9	1179.6	53533.34	213.9	19708.96	9.119	18222.80
10	1185.5	53222.29	167.2	20059.34	8.725	17542.58
平均值	1272.370	55381.703	267.790	19744.068	<b>9.1771</b>	<b>17905.663</b>

着更少的平均计算时间。综上所述，本文的 TSAVN 算法与其他算法相比具有显著的优越性。

#### 4.5 本章小结

本章主要对本文所提出的任务计算资源的调度优化算法、多无人机航迹规划及任务分配算法在仿真环境下与不同的算法进行了性能测试，并进行了本文提出的算法与不同的算法在算法求解、运算时间等方面进行了性能的比较，进而得到本文所提出的算法的有效性及优越性。

## 第 5 章 总结与展望

### 5.1 总结

本文的主要研究成果如下：

1. 对无人机航迹规划及任务调度问题进行了具体的描述与内容分析，在基于边缘计算架构将其分解为面向边缘计算的任务计算资源调度优化问题、多无人机的航迹规划及任务分配问题等两个子问题，基于合理假设为这两个子问题分别构建任务计算资源的调度优化模型、多无人机航迹规划及任务分配模型，为系统研究这两个问题提供了研究基础；
2. 针对任务计算资源的调度优化问题，考虑静态和动态两种不同的场景，设计了面向边缘计算的任务资源调度算法 ASA、FD，在性能测试中，通过与 SA 算法的比较，表明 ASA 算法在任务完成时间、设备空闲时间等目标值上更优，并且与 SA 算法相比运行时间有所减少，而 FD 算法虽然得到的解的质量较差，但是其运行时间非常短，能够满足动态场景下的计算资源调度问题的实时性要求；
3. 针对多无人机的任务分配及轨迹规划问题，设计了基于 RRT\*-Connect 的静态场景下的无人机航迹规划算法、基于 A\* 的动态场景下的无人机航迹规划算法和基于 TSAVN 的多无人机任务调度分配算法。在算法性能测试方面，将其与多种算法进行对比，验证了所提算法在求解相应问题时的有效性和优越性。

### 5.2 展望

本文使用的多种算法虽然系统地解决了无人机航迹规划及任务调度问题的两个子问题，取得了较好的成功，但在研究过程中还存在着一些不足，未来需要对本文的研究问题和所用算法进行进一步的优化，主要的展望如下：

1. 无人机实时避障算法中，设置的位置障碍物为固定的，且无人机传感器能够检测到，但实际环境中可能存在着细小的障碍物，例如电线、未知无人机等物体，对于这些物体，由于其可能因为动态性或无法侦测性，可能导致无人机出现损毁等现象，因此有必要进一步研究无人机对于动态或细小的障碍物的动态场景下的航迹规划算法；
2. 在任务计算资源调度优化问题中，为简化问题，本文假设无人机与无人机、无人机与服务器的通信延迟、计算型任务在无人机和服务器上计算所需要的时间是可以通过例如神经网络、支持向量机等模型进行预测的，但并没有进行深入研究及实现，

且该问题对任务计算资源调度优化问题中起着非常大的影响作用，因此有必要进一步研究无人机与无人机、无人机与服务器的通信延迟、计算型任务在无人机和服务器上计算所需要的时间的预测方法；

3. 本文将无人机航迹规划及任务调度问题分解为了两个子问题，但并未将两个子问题合并后进行求解，因此本文的研究成果离实际应用还有一定距离。

## 致谢

## 参考文献

- [1] Peng K, Du J, Lu F, et al. A Hybrid Genetic Algorithm on Routing and Scheduling for Vehicle-Assisted Multi-Drone Parcel Delivery[J]. IEEE Access, 2019, 7: 49191-49200.
- [2] Yang T, Jiang Z, Sun R, et al. Maritime Search and Rescue Based on Group Mobile Computing for Unmanned Aerial Vehicles and Unmanned Surface Vehicles[J]. IEEE Transactions on Industrial Informatics, 2020, 16(12): 7700-7708.
- [3] 刘丽, 王森, 胡然. 美军主要无人机集群项目发展浅析[J]. 飞航导弹, 2018(07): 37-43.
- [4] 姚敏, 王绪芝, 赵敏. 无人机群协同作战任务分配方法研究[J]. 电子科技大学学报, 2013, 42(05): 723-727.
- [5] Chen W, Liu B, Huang H, et al. When UAV Swarm Meets Edge-Cloud Computing: The QoS Perspective[J]. IEEE Network, 2019, 33(2): 36-43.
- [6] Cheng N, Xu W, Shi W, et al. Air-Ground Integrated Mobile Edge Networks: Architecture, Challenges, and Opportunities[J]. IEEE Communications Magazine, 2018, 56(8): 26-32.
- [7] Messous M A, Sedjelmaci H, Houari N, et al. Computation Offloading Game for an UAV Network in Mobile Edge Computing[C]. in: 2017 IEEE International Conference on Communications (ICC). 2017: 1-6.
- [8] 彭维平, 王明坤, 宋成, 等. 多无人机协同直播场景下自适应任务卸载决策[J]. 控制与决策, 2021, 36(04): 974-982.
- [9] Kim K, Hong C S. Optimal Task-UAV-Edge Matching for Computation Offloading in UAV Assisted Mobile Edge Computing[C]. in: 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS). 2019: 1-4.
- [10] Zhang Q, Chen J, Ji L, et al. Response Delay Optimization in Mobile Edge Computing Enabled UAV Swarm[J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 3280-3295.
- [11] Cao X, Xu J, Zhang R. Mobile Edge Computing for Cellular-Connected UAV: Computation Offloading and Trajectory Optimization[C]. in: 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). 2018: 1-5.
- [12] Zhu S, Gui L, Chen J, et al. Cooperative Computation Offloading for UAVs: A Joint Radio and Computing Resource Allocation Approach[C]. in: 2018 IEEE International Conference on Edge Computing (EDGE). 2018: 74-79.
- [13] Kim K, Park Y M, Seon Hong C. Machine Learning Based Edge-Assisted UAV Computation Offloading for Data Analyzing[C]. in: 2020 International Conference on Information Networking (ICOIN). 2020: 117-120.

- [14] Yang L, Yao H, Wang J, et al. Multi-UAV-Enabled Load-Balance Mobile-Edge Computing for IoT Networks[J]. IEEE Internet of Things Journal, 2020, 7(8): 6898-6908.
- [15] Bortoff S. Path Planning for UAVs[C]. in: Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334). 2000, 1(6): 364-368.
- [16] Rohnert H. Shortest Paths in the Plane with Convex Polygonal Obstacles[J]. Information Processing Letters, 1986, 23(2): 71-76.
- [17] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.
- [18] 李春华, 郑昌文, 周成平, 等. 一种三维航迹快速搜索方法[J]. 宇航学报, 2002(03): 13-17.
- [19] Khatib O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots[C]. in: 1985 IEEE International Conference on Robotics and Automation Proceedings. 1985, 2: 500-505.
- [20] 张佳龙, 闫建国, 张普, 等. 基于改进人工势场的无人机编队避障控制研究[J]. 西安交通大学学报, 2018, 52(11): 112-119.
- [21] Lin Y, Saripalli S. Sampling-Based Path Planning for UAV Collision Avoidance[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(11): 3179-3192.
- [22] Karaman S, Frazzoli E. Sampling-Based Algorithms for Optimal Motion Planning[J]. The International Journal of Robotics Research, 2011, 30(7): 846-894.
- [23] Zhang D, Xu Y, Yao X. An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect[C]. in: 2018 37th Chinese Control Conference (CCC). 2018: 4854-4858.
- [24] Sonmez A, Kocyigit E, Kugu E. Optimal Path Planning for UAVs Using Genetic Algorithm[C]. in: 2015 International Conference on Unmanned Aircraft Systems (ICUAS). 2015: 50-55.
- [25] Wang Q, Zhang A, Qi L. Three-Dimensional Path Planning for UAV Based on Improved PSO Algorithm[C]. in: The 26th Chinese Control and Decision Conference (2014 CCDC). 2014: 3981-3985.
- [26] Yan C, Xiang X. A Path Planning Algorithm for UAV Based on Improved Q-Learning[C]. in: 2018 2nd International Conference on Robotics and Automation Sciences (ICRAS). 2018: 1-5.
- [27] Ghazzai H, Menouar H, Kadri A, et al. Future UAV-Based ITS: A Comprehensive Scheduling Framework[J]. IEEE Access, 2019, 7: 75678-75695.
- [28] Alidaee B, Wang H, Landram F. A Note on Integer Programming Formulations of the Real-Time Optimal Scheduling and Flight Path Selection of UAVs[J]. IEEE Transactions on Control Systems Technology, 2009, 17(4): 839-843.
- [29] Schermer D, Moeini M, Wendt O. A Matheuristic for the Vehicle Routing Problem with Drones and Its Variants[J]. Transportation Research Part C: Emerging Technologies, 2019, 106: 166-204.

- [30] Faiz T I, Vogiatzis C, Noor-E-Alam M. A Column Generation Algorithm for Vehicle Scheduling and Routing Problems[J]. *Computers & Industrial Engineering*, 2019, 130: 222-236.
- [31] Jia Z, Yu J, Ai X, et al. Cooperative Multiple Task Assignment Problem with Stochastic Velocities and Time Windows for Heterogeneous Unmanned Aerial Vehicles Using a Genetic Algorithm[J]. *Aerospace Science and Technology*, 2018, 76: 112-125.
- [32] Zhen Z, Xing D, Gao C. Cooperative Search-Attack Mission Planning for Multi-UAV Based on Intelligent Self-Organized Algorithm[J]. *Aerospace Science and Technology*, 2018, 76: 402-411.
- [33] Zhu M, Du X, Zhang X, et al. Multi-UAV Rapid-Assessment Task-Assignment Problem in a Post-Earthquake Scenario[J]. *IEEE Access*, 2019, 7: 74542-74557.
- [34] Wu G, Wang H, Pedrycz W, et al. Satellite Observation Scheduling with a Novel Adaptive Simulated Annealing Algorithm and a Dynamic Task Clustering Strategy[J]. *Computers & Industrial Engineering*, 2017, 113: 576-588.
- [35] LaValle S M. Rapidly-Exploring Random Trees: A New Tool for Path Planning[R]. 1998.
- [36] LaValle S M, Kuffner J J. Randomized Kinodynamic Planning[J]. *The International Journal of Robotics Research*, 2001, 20(5): 378-400.
- [37] Kuffner J, LaValle S. RRT-connect: An Efficient Approach to Single-Query Path Planning[C]. in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). 2000, 2: 995-1001.
- [38] Braun J, Brito T, Lima J, et al. A Comparison of A\* and RRT\* Algorithms with Dynamic and Real Time Constraint Scenarios for Mobile Robots[C]. in: SIMULTECH 2019: Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications. 2019: 398-405.
- [39] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. *IEEE Transactions on Systems Science and Cybernetics*, 1968, 4(2): 100-107.
- [40] Croes G A. A Method for Solving Traveling-Salesman Problems[J]. *Operations Research*, 1958, 6(6): 791-812.
- [41] Gehring H, Homberger J. Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows[J]. *Journal of Heuristics*, 2002, 8(3): 251-276.