分类号 TP311 密级 公开
UDC 编号

雲南大學

硕士研究生学位论文

题 目 **车联网环境下能耗优先的任务调度算法研究**

Title **Research on Energy Priority Task Scheduling Algorithm in the Vehicle Internet Environment**

学院（所、中心） 信息学院

专业名称 计算机软件与理论

研究方向 分布式计算和智能计算

研究生姓名 学号 12021115016

导师姓名 职称 教授

2024 年 3 月

扉页 1：

<table>
<tr><td rowspan="12">云南大学硕士研究生学位申请简况表</td><td rowspan="6">论文预审</td><td colspan="4">论文预审结果：</td></tr>
<tr><td>专家姓名</td><td>职称</td><td colspan="2">所在单位（校内：学院/校外：所在单位）</td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td rowspan="5">论文送审</td><td>专家姓名</td><td>职称</td><td>所在单位（校内：学院/校外：
所在单位）</td><td>结果</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="6">论文答辩</td><td colspan="4">答辩结果：</td></tr>
<tr><td>答辩专家</td><td>职称</td><td colspan="2">所在单位（校内：学院/校外：所在单位）</td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td colspan="2"></td></tr>
</table>

扉页 2：

论文独创性声明及使用授权

本论文是作者在导师指导下取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，不存在剽窃或抄袭行为。与作者一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

现就论文的使用对云南大学授权如下：学校有权保留本论文（含电子版），也可以采用影印、缩印或其他复制手段保存论文；学校有权公布论文的全部或部分内容，可以将论文用于查阅或借阅服务；学校有权向有关机构送交学位论文用于学术规范审查、社会监督或评奖；学校有权将学位论文的全部或部分内容录入有关数据库用于检索服务。

（内部或保密的论文在解密后应遵循此规定）

研究生签名：_____ 导师签名：_____ 日 期：_____

摘要

关键词：潜在变量建模；情感文本生成；变分自编码器；角色对话

Abstract

Key words: Latent variable modeling; Emotional text generation; Variational autoencoder; Persona-based Dialogue

目录

插图

表格

第1章 绪论

本章主要讲述了车联网中能耗优先调度优先算法的意义，以及其研究现状和成果，提出了本文所要做的任务，阐述了本文的主要研究内容和介绍了本文的组织结构。

1.1 研究背景及研究意义

未来将是“万物互联”的时代，随着射频识别技术、网络协议、数据存储以及传感器技术日新月异的发展，越来越多的物理机器与互联网连接起来，实现了物体之间的信息交互，以及赋予了机器以“智能”。这将深刻地改变人们的生产生活方式，为人们提供更加便利、智能和高效的生活和工作体验[1]【余文科程媛, WENKE YU Y C. 物联网技术发展分析与建议[J/OL]. 物联网学报, 2020, 4(4): 105-109. DOI:10.11959/j.issn.2096-3750.2020.00195.】。

车联网是“万物互联”时代的典型应用[2]【】，汽车的功能越来越超出其原本的范围，不再只是交通出行的工具，而变成了一个智能互联的计算系统，车辆将承担起对车辆自身的调度与管理的任务，以及满足影音娱乐等用户额外的需求。在对车辆自身进行调度与管理的任务中，有些任务对计算资源的要求较高，且对时延敏感，因此需要大量的稳定的计算资源来保证其服务质量。显而易见，车载处理器的计算资源受限于汽车电池[3]，计算资源有限，无法满足对互联的计算任务的要求。

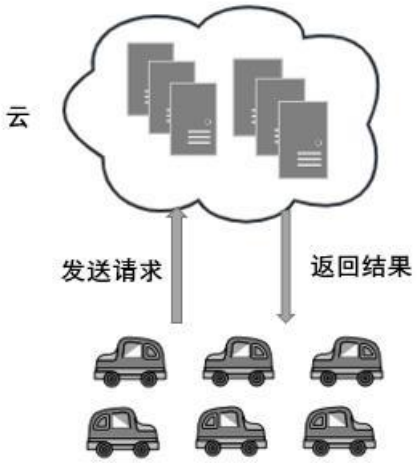


图 3-1：云计算

为了解决计算资源受限的问题，基于云的解决方案通过将任务上传到云端（远端、远程云）来进行处理（如图 3-1），云端本身就是一个计算中心，通常具有超大的规模，能赋予用户超强的计算能力，对大规模数据进行聚合、挖掘、分析、优化，可以在短的时间内计算得到结果。

车辆应用产生的计算任务可以通过网络传输到远程云，满足运算和储存资源的需求，在云端计算并将结果返回给车辆。由于云端超强的计算能力，计算资源能满足车辆任务的需求，因此能够计算和分析复杂的计算任务提供高性能计算资源。

但是因为云端位于网络的远端，往往部署在偏远地区，将车辆计算任务传输到远程云并得到相应的返回结果需要很大的传输时间开销，在数据传输过程中会产生较高的时延。因此对于数据密集型和延迟敏感型的计算任务，即使计算延迟较小（数据量较小），较大的传输延迟仍会使任务总卸载延迟较大。同时，为了将车辆产生的大量数据全部上传到云端，这将会导致网络中的数据量大幅增加，占用大量的网络带宽，使得网络设施负担加重，从而造成网络性能和稳定性的下降，这就是云计算中链路过载的问题。

【[8] Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: The communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

9] 赵梓铭,刘芳,蔡志平,等.边缘计算:平台、应用与挑战[J].计算机研究与发展,2018, 55(002): 327-337.

10] 田辉,范绍帅,吕昕晨,赵鹏涛,贺硕.面向 5G 需求的移动边缘计算[J].北京邮电大学学报, 2017, 40(02): 1-10.】

为了解决云计算的时延以及流量问题，促进数据处理能力向数据源更接近的方向发展，移动边缘计算（MEC）应运而生。MEC 选择了在网络边缘中（边缘端）处理数据，与传统的云计算相比，边缘计算将服务部署在距离用户更近的地方，放置到网络的边缘，可以实现更低的时延和更高的带宽，减少了核心网络的负载，解决了云计算中链路过载的问题。同时，这还有利于数据安全以及隐私保护。

边缘计算也可以通过与物联网技术结合，实现一些与智能车联网的应用，如自动驾驶、智慧城市等[4]【Joint task management in connected vehicle networks by software-defined networking, computing and caching】。美国自然基金委和英特尔在 2016 年开始合作讨论建设无线边缘网络。同一年，中国也兴起了移动边缘计算的研究，在科研院所、高等院校、企业、行业协会等各方单位的共同努力下，边缘计算产业联盟成立。

边缘端位于云端和车辆之间，它由拥有计算能力的边缘设备组成，它是移动边缘计算中的重要组成部分，是解决远程云传输延迟高的一种有效替代方案。边缘云可由边缘服务器、路边单元（Roadside Unit，RSU）、基站（Base Station ，BS）、无线接入点（Access Point，AP）以及移动设备等其他具备计算、存储和通信能力的边缘设备组成。RSU 是一种安装在道路旁边的设备，用于与车辆进行通信。它可以提供实时的交通信息、道路状态、限速提示等服务，并且可以与车辆进行数据交换和通信。RSU 通常会连接到云服务器，通过无线通信网络与车辆进行通信。BS 是无线通信系统中的基站，用于提供无线信号覆盖。在车联网中，BS 可以提供车辆与网络的连接，使车辆能够接收到互联网的信息和服务。BS 通

常会与 RSU 结合使用，共同构成车辆与基础设施之间的通信网络。RSU 和 BS 在车联网中起到了连接车辆与道路基础设施、提供实时信息和服务的作用。

因为边缘云处在网络的边缘，更靠近车辆，车辆到 EC 的传输延迟可以大大降低。因此边缘云可以为车辆提供低延迟计算、高速缓存、位置感知、紧急管理等服务，并能用于实时交互，对于数据密集型和延迟敏感型的车辆计算任务，例如：增强现实，实时视频分析，人类行为识别等任务，它们需要极低的延迟以便快速响应和决策，此时边缘云比远程云具有更大的优势。

边缘计算并不是云计算的反面，边缘计算是云计算的延伸，未来将与云计算协同配合，为用户提供更高质量的服务。

为了使车联网支持云计算和边缘计算，科研组织和联盟实现了车到万物通信(V2X)技术，V2X 主要包括车辆对车辆通信(V2V)、车辆对基础设施通信(V2I)【】。

1.2 国内外研究历史和现状

1.2.1 任务卸载与车辆任务分类

任务卸载是指将移动设备（如智能手机、车载终端等）上的计算任务分解成若干个子任务，其中一部分在本地完成，另一部分则通过网络卸载到云端或者其他设备上进行计算，最终将结果返回到本地设备，从而提高移动设备的计算性能和能耗效率的一种技术。

根据卸载的目标，任务卸载可以分为以下两类：

（1）计算卸载：主要目的是将计算密集型任务卸载到云端或其他远程设备上，减少本地设备的计算负担，提高计算性能和能耗效率；

（2）通信卸载：主要目的是将数据密集型任务卸载到云端或其他远程设备上进行处理，减少本地设备的数据传输负担，提高通信性能和能耗效率。

在本文中，主要研究的是车联网中的通信卸载。

在车联网中，任务卸载主要应用于以下几个方面：

（1）智能驾驶：智能驾驶需要大量的计算因此需要强大的数据处理能力，通过任务卸载技术，可以将一部分计算和数据处理任务卸载到云端或其他设备上，提高计算性能和能耗效率。

（2）车辆监控：车辆监控需要实时收集和处理大量的数据，并进行数据分析和决策，通过任务卸载技术，可以将一部分数据处理任务卸载到云端或其他设备上，提高数据处理效率。

（3）车辆诊断：车辆诊断需要进行大量的数据分析和处理，通过任务卸载技术，可以将一部分数据处理任务卸载到云端或其他设备上，提高诊断效率和精度。

智能车辆的任务并不是都可以卸载的，车辆任务按照其关键程度分为三类：关键任务（Crucial Tasks，CTs）、高优先级任务（High-Priority Tasks，HPTs）和低优先级任务（Low-Priority Tasks，LPTs）。

CTs 是和车辆安全相关的应用，是保证车辆和乘客安全的关键应用程序，如车辆控制、系统监控和事故预防等。由于 CTs 和安全紧密相关，因此享有最高的优先级，必须保留充足的计算资源给它，不能因为 HPTs 和 LPTs 的存在而影响 CTs 的正常运行，因此这类任务也不允许卸载，只允许在本地执行，不属于计算任务卸载的范畴。该类任务的实例是：车辆控制、碰撞预警、红绿灯警告、网上车辆诊断、道路湿滑程度检测等。

HPTs 包括与驾驶相关的应用和可选的安全增强应用，这类应用程序对车辆而言是重要但不是必须的，拥有较高的执行优先级，例如实时路径规划和路况提醒等。这类应用允许出现延迟或卸载失败的情况，但不能影响 CTs。该类任务的实例是：地图导航、平视显示器、视野增强、车辆传感等。

LPTs 是一类为用户提供影音娱乐服务的应用程序，它的优先级较低，例如语音识别，它允许驾驶员发出各种声音命令，通过语音识别命令计算机做一些响应，

而不会使驾驶员分心。该类任务的实例是：虚拟现实、语音识别、视频处理、在线游戏等。

HPTs 和 IPTs 已经被部署到越来越多的车辆上，由于 HPTs 和 LPTs 不会涉及到安全，因此可以将其进行卸载，来提高资源的利用率。

1.2.2 任务卸载的优化目标

（1）最小化卸载时延。车辆所产生的任务可以选择在本地执行或者卸载到边缘服务器上进行执行，前者会产生本地时延和卸载时延，本地时延是指数据在本地运行所消耗的全部时间：卸载时延是指在边缘计算环境中，数据从生成到处理的时间间隔（一般忽略回传时延）。【ZHENG K, MENG H, CHATZIMISIOS P, et al. An SMDP-based resource allocation in vehicular cloud computing systems[J]. IEEE Transactions on Industrial Electronics, 2015, 62(12): 7920-7928.】

（2）最小能量消耗。通常而言，实际的能耗也是必不可缺的指标。对于车辆而言，可以提高车辆的行驶里程，环节“里程焦虑”。对于边缘服务器而言，可以最大限度地提高网络运营商和服务提供商的收入。

1.2.3 车辆边缘计算面临的诸多挑战

目前，基于车联网的车辆边缘计算面临着诸多挑战。

（1）实时性问题，车辆位置、速度、路况等方面的影响会导致车辆任务的实时变化，进而导致车辆任务卸载环境的不稳定性，上一时刻的最优分配方案可能下一时刻就是最劣方案，导致任务卸载方案的可靠性降低。

（2）资源动态变化问题，车辆位置、速度等方面的影响，车辆的计算等资源会发生相应改变。目前，大多数车联网任务卸载的研究都假设车联网中计算资

源在较长时间内保持相对稳定。而在现实情况下，各种因素会导致资源状态发生明显的变化，导致车联网任务卸载算法的适应性降低。

（3）在车联网中，由于任务关键程度的不同，车联网任务卸载环境变得更加复杂。虽然车辆边缘计算可以分担车联网中的计算任务，但是并不是所有任务都适合在边缘设备上进行处理。特别是一些需要大量计算资源、对实时性要求高或者涉及到安全性的任务，无论从效率还是安全性上考虑，都不宜采用边缘计算方式。因此，在车辆边缘计算中，不可能将所有任务都卸载出去。对于那些不适合在边缘设备上进行处理的任务，应当优先本地来完成。这样可以保证车辆的行车安全。

1.3 论文的研究内容和组织结构

本文主要研究了车联网任务卸载算法，在车辆任务不能完全卸载情况下，即有部分任务必须车辆在本地执行的情况下，如何利用能耗与 CPU 频率之间的非线性关系进行任务卸载用来提高整体的能量利用率，使得整体能耗下降。

论文共分为五章，论文的具体内容安排如下：

第一章对全文内容进行概述和章节安排，首先介绍了车联网和移动边缘计算研究背景和意义，并简述了边缘计算中任务卸载的国内外研究现状，对全文内容进行概述，最后给出论文组织章节安排。

第二章介绍车联网中任务卸载的算法研究，常用的算法有深度强化学习，博弈论方法以及启发式的方法。

【第二章介绍深度强化学习基本理论及本文的系统模型。理论部分本文介绍了马尔可夫决策过程、强化学习算法、深度强化学习算法和多智能体强化学。最后介绍本文 VEC 密集型任务调度场景系统模型，并提出了优化问题。】

第 3 章介绍本文的任务卸载模型，并提出了优化问题，并将问题转化为非线性数学规划的形式，因为该问题难以求出最优解，因此使用近似算法即基因算法解决了该问题。

第 4 章针对基因算法运行时间长的问题，在侏儒猫鼬算法的基础上，提出了离散侏儒猫鼬算法，来提高算法的运行速度。

第 5 章全文总结与展望。总结全文的研究内容和主要贡献，以及需要后续深入研究的问题。

第2章 车联网中任务调度算法研究

目前，车联网中的车辆边缘计算已经受到汽车产业和学术界的广泛重视，并产生了大量的研究成果。下面，本节将综述车联网中车辆边缘计算的研究现状。

2.1 基于强化学习的车辆边缘计算任务调度算法研究

【Haydar A, Yılmaz Y. Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey[J].EEE Transactions on Intelligent Transportation Systems, 2022, 23(1): 11–32.】

随着深度强化学习（DRL）算法理论的不断完善，DRL 已广泛应用于车联网中智能控制与资源管理与优化等方面[5]。 DRL 算法可以感知环境的变化并做出决策，能满足车联网场景的需求，被认为是解决车辆边缘计算的有效方案。

深度强化学习是一种通过深层神经网络来训练智能体（Agent）在复杂环境中学习决策的方法。它将强化学习（Reinforcement Learning，RL）与深度学习（Deep Learning，DL）相结合，通过神经网络对状态空间进行近似，实现高效的决策和学习。

强化学习是一种机器学习的方法，通过智能体（Agent）与环境进行交互学习，使得智能体能够在不断的试错中获得最优决策策略。强化学习的核心思想是基于奖励信号，通过选择不同的行动来最大化期望收益。强化学习的基本框架是马尔可夫决策过程（Markov Decision Process，MDP），其中包括状态空间、行动空间、奖励函数和转移概率。智能体通过采取不同的行动，使得状态从一个状态转移到另一个状态，并根据奖励函数获得相应的奖励。强化学习的目标是找到一个最优的策略，使得智能体在长期累计奖励最大的情况下完成任务。

深度学习是一种通过多层神经网络对数据进行特征提取和模式识别的方法。深度学习的核心思想是通过层次化的结构对底层特征进行组合和抽象，实现对高层次的抽象表示。深度学习的经典方法是深度神经网络（Deep Neural Network，DNN），其中包括多层全连接层、卷积层和池化层等。

深度强化学习将强化学习和深度学习相结合，其主要思想是使用深层神经网络来逼近最优策略函数，从而实现对复杂环境中的决策问题的解决。与传统的强

化学习相比，深度强化学习具有更高的泛化能力、更快的训练速度和更好的性能表现。深度强化学习的主要应用领域包括游戏、机器人控制、自然语言处理等。

深度强化学习的策略函数方法包括价值函数法、策略梯度法和 Actor-Critic 方法等。价值函数法是通过估计状态或状态 - 行动对的价值函数来确定最优策略，其中包括 Q-learning 和 Deep Q-Network（DQN）等。策略梯度法是通过直接优化策略函数来求解最优策略，其中包括 Policy Gradient 和 Actor-Critic 方法等。Actor-Critic 方法则结合了价值函数法和策略梯度法的优点，通过同时学习策略函数和价值函数来实现最优决策。

DRL 算法主要包括智能体、环境、动作和奖励等部分。一般而言，模型通常把执行决策的主体定义为智能体，并把能够影响智能体进行决策的因素称为环境。智能体与环境进行交互，不断地积累经验。

【Liu N, Li Z, Xu Z, et al. A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning[C]. 2017 IEEE 37th international conference on distributed computing systems (ICDCS). Atlanta, GA, USA: IEEE, 2017: 372-382.】【提出了一种分层式的结构，以应对高维环境状态空间及高维智能体动作空间问题，解决了云计算中车联网资源分配及发射功率优化问题。】

目前已有众多学者运用 DRL 方法解决车联网中任务卸载问题。如文献[6] 提出了一种新的层次框架来解决云计算系统中的整体资源分配和电源管理问题，分层包括用于将虚拟机资源分配给服务器的全局层和用于本地服务器的分布式电源管理的本地层，以应对算法的高维问题，实现了服务器集群中实现延迟和功耗/能耗之间的最佳权衡。

【Zhan Y, Yao J, Guan H. Intelligent Cloud Resource Management with Deep Reinforcement Learning[J]. IEEE Cloud Computing, 2017, 4(6): 60–69.】【为实现资源管理的智能化，提出了一种基于 DRL 的车联网资源管理方案。】

文献[7] 提出了一种基于深度强化学习的智能云资源管理架构，使云能够直接从复杂的云环境中自动高效地协商最合适的配置。

【Cheng M, Li J, Nazarian S. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers[C]. 2018 23rd Asia

and South Pacific Design Automation Conference (ASP-DAC). Jeju: IEEE, 2018: 129–134.】

文献[8]提出了一种基于 DRL 的、为云服务商提供资源配置和任务调度的方案，用于最大限度地降低具有大量服务器的云服务提供商的能源成本。

【Karthiban K, Raj J S. An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm[J]. Soft Computing, 2020, 24(19): 14933–14942】

文献 31 则提出了一种基于 DRL 的车联网公平性资源分配方案，以解决车联网中资源分配的问题。文献 2】为解决 VEC 任务调度场景任务处理时延和系统能耗最小化问题，利用马尔可夫决策过程（Markovdecisionprocess,MDP）建立任务调度模型，通过 DRL 方法优化任务调度方案，实验验证了 DRL 方法用于解决复杂决策问题的有效性。文献【33】利用车联网中 RSU 和车载计算资源，实现车辆计算任务卸载，其中 RSU 为车辆提供通信功能和计算资源，车辆协作进行数据传输和计算。文献 34 针对车路云协同场景，运用 DRL 方法优化计算资源联合调度方案，降低任务处理时延。文献【35】提出了一种基于 DRL 算法的方案，考虑了车辆和路边单元 RSU 作为移动边缘计算节点和基站，实现最小化能量消耗和数据传输时延的目标。文献【36】研究了车联网中车辆和边缘计算的协作式任务卸载策略，车辆可以将还未计算完成的计算卸载任务迁移到附件的其他空闲车辆，以完成任务计算。文献【37】利用 DRL 算法为车联网求解联合车载计算资源及边缘服务器计算资源的计算卸载策略，以最小化系统能耗。

但是，车联网 VEC 规模不断扩大，任务计算量呈指数增长，传统 DRL 算法无法满足高维度计算需求 81.基于 DRL 的多智能体处理采用集中训练分布式运行方式，大幅降低计算资源开销，适用于执行车联网复杂环境下的决策任务。

文献【39 采用 DRL 方法优化车辆到车辆（VehicletoVehicle.V2V）无线链路的频谱分配方案，并执行多智能体分布式任务处理方式，实现网络信道总容量的提升。

文献【40】提出多智能体强化学习及拉格朗日乘子两级嵌套的组合优化方法，解决车载终端任务卸载及服务器缓存优化问题从而提高系统效能。

文献【41】设计多智能体 DRL 算法解决车联网车载终端任务卸载最优决策问题，最小化任务执行时延。

[26] Haydar A, Yılmaz Y. Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey[J].
EEE Transactions on Intelligent Transportation Systems, 2022, 23(1): 11–32.
27] Tang F, Mao B, Kato N, et al. Comprehensive Survey on Machine Learning in Vehicular Network:Technology, Applications and Challenges[J]. IEEE Communications Surveys & Tutorials, 2021, 23(3):2027–2057.
31] Karthiban K, Raj J S. An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm[J]. Soft Computing, 2020, 24(19): 14933–14942.
32] Zhan W, Luo C, Wang J, et al. Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing[J]. IEEE Internet of Things Journal, 2020, 7(6): 5449–5465.
33] Luo Q, Li C, Luan T H, et al. Collaborative Data Scheduling for Vehicular Edge Computing via Deep Reinforcement Learning[J]. IEEE Internet of Things Journal, 2020, 7(10): 9637–9650.
34] 丁飞, 沙宇晨, 洪莹, 等. 智能网联汽车计算卸载与边缘缓存联合优化策略[J]. 系统仿真学报, :1–11.
35] Ke H, Wang J, Deng L, et al. Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(7): 7916–7929.
36] Sun F, Cheng N, Zhang S, et al. Reinforcement Learning Based Computation Migration for Vehicular Cloud Computing[C]. 2018 IEEE Global Communications Conference (GLOBECOM). Abu Dhabi, United Arab Emirates: IEEE, 2018: 1–6.
37] He X, Lu H, Du M, et al. QoE-Based Task Offloading With Deep Reinforcement Learning in EdgeEnabled Internet of Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(4):2252–2261.
38] 杜威, 丁世飞. 多智能体强化学习综述[J]. 计算机科学, 2019, 46(08): 1–8.
考文献 42[39] Liang L, Ye H, Li G Y. Spectrum Sharing in Vehicular Networks Based on Multi-Agent Reinforcement Learning[J]. IEEE Journal on Selected Areas in

Communications, 2019, 37(10): 2282－2292.
40] 宁兆龙, 张凯源, 王小洁, 等. 基于多智能体元强化学习的车联网协同服务缓存和计算卸载[J].
信学报, 2021, 42(06): 118－130.
41] Zhu X, Luo Y, Liu A, et al. Multiagent Deep Reinforcement Learning for Vehicular Computation Offloading in IoT[J]. IEEE Internet of Things Journal, 2021, 8(12): 9763－9773.

2.2 基于博弈论的车辆边缘计算任务调度算法研究

博弈论是研究决策制定者在冲突或合作情境中进行策略选择的数学理论。当使用博弈论方法对车联网中的任务卸载进行研究时，首先要把目标最优问题转换为博弈论的基础问题，然后对其进行数学建模。建模的目的是利用博弈论来确定均衡状态。均衡状态是指所有的参与者都得到最佳收益的状态，当一个系统的博弈达到均衡时，每个参与者都无法仅通过改变自己的策略而在系统中得到更好的结果。博弈的本质就是多个相互影响的参与者在选择各自的决策时最大化自己的收益。

博弈论的方法主要包括以下几种：

（1） 纳什均衡方法。纳什均衡是博弈论最重要的概念之一。它指的是在一个博弈中，每个玩家都采取了最优的策略，而且没有人想改变自己的策略。这种情况下，所有玩家都处于理性选择的状态，即达到了稳定状态。纳什均衡在很多实际问题中都有广泛的应用。

（2）最小化最大损失方法。最小化最大损失是一种双方博弈的决策方法，也被称为 “最小保证赢法”。这种方法的基本思想是：在不确定对手的行动时，尽量减少自己可能遭受的最大损失。这种策略对于不确定性很大的问题非常有用。

（3）博弈树方法。博弈树是博弈论中常用的一种分析方法。它是一种图形，用于显示博弈的各种可能情况和策略选择，以及每种情况下玩家的收益或损失。通过分析博弈树，可以找到纳什均衡点和最优策略。

【https://arxiv.org/abs/2209.12749】【XU X, LIU K, DAI P, et al. Joint task offloading and resource optimization in NOMA-based vehicular edge computing: a game-theoretic DRL approach[J]. Journal of Systems Architecture, 2023, 134(1): 102780-102825.】

文献将任务卸载子问题建模为精确势博弈，并提出了一种多智能体分布式深度确定性策略梯度来实现纳什均衡。将资源分配子问题划分为两个独立的凸优化问题，并利用基于梯度的迭代方法和 KKT 条件提出了最优解。

【https://www.researchgate.net/publication/335594454_A_Game-Theoretical_Approach_for_User_Allocation_in_Edge_Computing_Environment】【HE Q, CUI G, ZHANG X, et al. A game-theoretical approach for user allocation in edge computing environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3):515-529.】

文献[9]将问题表述为潜在的博弈并证明它存在纳什均衡，设计了一种去中心化算法，用于在博弈中找到纳什均衡。

【CAO H, CAI J. Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: a game-theoretic machine learning approach[J]. IEEE Transactions on Vehicular Technology, 2018, 67(1):752-764.】

文献[10]将多用户卸载决策问题建模为非合作博弈，证明该问题存在纳什均衡，并提出了一种基于机器学习技术的全分布式计算卸载算法来寻找该纳什均衡点。

2.3 基于启发式算法的车辆边缘计算任务调度算法研究

启发式算法是一种解决复杂问题的计算方法，它通过利用经验或启示来指导搜索过程，以找到可能的解决方案。与精确算法不同，启发式算法可能无法保证找到最优解，但通常能够在合理的时间内找到接近最优解的解决方案【刘逸. 粒子群优化算法的改进及应用研究[D]. 西安: 西安电子科技大学, 2013.

IU Y. Research on improvement and application of particle swarm optimization algorithm[D]. Xi’an: Xidian University, 2013.】[11]。启发式算法是基于直观自然现象或者结构自然背后的经验构造的算法，在可接受的时空开销内给出问题的一个可行解（次优解）。贪婪算法是一种直观的启发式算法，它每一步选择当前最好的选择（局部最优），而不考虑长远影响（整体最优）。贪婪算法通常易于实现和理解，但可能会得到次优解。模拟退火算法受到固体退火过程的启发，通过接受一定概率的劣解，以避免陷入局部最优解，并逐渐趋向全局最优解。模拟退火算法适用于求解复杂问题，但需要合适的参数设置和调整。

【BELOGAEV A, ELOKHIN A, KRASILOV A, et al. Cost-effective V2X task offloading in MEC-assisted intelligent transportation systems[J]. IEEE Access, 2020(8): 169010-169023.】【文献研究了多服务器场景下的 3 种 V2X 应用程序卸载成本的问题，提出了低复杂度启发式贪婪算法，该算法以计算时延和成本函数为优化目标来选择每项任务所需成本最低的服务器。】

文献[12]通过考虑通用型任务计算时间分布来分析排队延迟，通过考虑给定概率满足延迟约束，以最小化受延迟和计算资源约束的 ITS 算子费用提出了一个非线性优化问题。针对该问题，该文提出一种线性化方法，并设计了一种基于整数线性规划的算法同时引入了一种称为任务卸载的成本效益启发式算法的启发式算法来解决该问题。

【CHEN M H, DONG M, LIANG B. Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints[J]. IEEE Transactions on Mobile Computing, 2018, 17(12): 2868-2881.】【，用于处理任务分发和计算资源分配的联合任务，以最小化能源消耗和时延的加权成本】

文献[13]研究了一个由多用户、一个边缘节点以及一个云服务器组成的移动云计算系统，提出了一个优化目标为用户之间的能耗、计算和最大延迟的总体成本的非凸二次约束二次规划问题，并提出了一种高效的启发式算法来解决这个问题。

【HUYNH L N T, PHAM Q V, PHAM X Q, et al. Efficient computation offloading in multi-tier multi-access edge computing systems: a particle swarm optimization approach[J]. Applied Sciences, 2019, 10(1): 203-210.】

Huynh 等人[14]开发了一种基于粒子群的启发式算法，用来解决异构网络中多用户多服务器的两层计算卸载策略问题，该问题的优化目标为总计算开销，包括完成时间和能耗。

【BUI K H N, JUNG J J. ACO-based dynamic decision making for connected vehicles in IoT system[J]. IEEE Transactions on Industrial Informatics, 2019, 15(10): 5648-5655.】【设计了一种基于群体智能的蚁群优化算法，用于联网车辆的动态决策，】

文献[15]提出了一种用于共享交通流信息的联网车辆之间的通信框架，将联网车辆视为人工蚂蚁，能够根据交通流量的动态进行自我计算以做出自适应决策，使车辆能计算出到达目的地的最佳路径。

2.4 数学规划

数学规划，又叫优化问题，是一类数学模型，用于在给定约束条件下寻找最优解决方案。它在现代科学、工程和经济领域中具有广泛的应用。数学规划可以被分为线性规划、非线性规划、整数规划等多种类型。

线性规划是数学规划中最常见的类型。线性规划是在给定线性约束条件下最大化或最小化一个线性目标函数。线性规划的目标函数和约束条件都是线性的，这使得它具有许多良好的数学特性，例如可解析性、单调性和凸性。线性规划可以用单纯形算法、内点算法等方法来求解。

非线性规划是指在非线性约束条件下，最大化或最小化一个非线性目标函数。与线性规划相比，非线性规划的求解更加困难，因为非线性函数没有单调性或凸性等良好的数学特性。非线性规划的求解方法包括牛顿法、拟牛顿法等。

整数规划是一类特殊的数学规划，它要求变量取整数值。整数规划在多个领域中得到了广泛的应用，如生产调度、航空运输等。整数规划的求解方法包括分支定界法、割平面法等。

在实际应用中，数学规划可以帮助我们优化资源分配、生产计划、物流运输等问题，使得资源利用更加高效，并且减少浪费。例如，在企业生产计划中，数学规划可以帮助制定最佳的生产计划，以满足市场需求，并且减少库存和成本。在物流运输中，数学规划可以帮助确定最优的路线，以最小化时间和成本。

数学规划是一种强大的工具，可以帮助我们在复杂的环境下寻找最优解决方案。通过使用数学规划，我们可以优化资源分配、提高效率、降低成本，从而为我们的生活和经济发展做出更大的贡献。

常见的数学规划包括线性规划、整数规划、非线性规划和混合整数规划等类型。

线性规划（Linear Programming，LP）：线性规划是一种用于优化的数学方法，其目标是最大化或最小化一个线性表达式，而且受到一组线性等式和不等式的约束。线性规划在工程、经济学、管理学等领域有广泛的应用，例如生产计划、运输和分配问题等。

整数规划（Integer Programming，IP）：整数规划是线性规划的扩展，其决策变量被限制为整数。这种类型的规划适用于需要做出离散决策的问题，比如资源分配、项目排程等。

非线性规划（Nonlinear Programming，NLP）：非线性规划是指目标函数或约束条件中存在非线性项的优化问题。非线性规划可以包括凸优化、非凸优化等问题，通常需要使用不同的算法进行求解。

混合整数规划（Mixed Integer Programming，MIP）：混合整数规划是整数规划的进一步扩展，允许部分决策变量为连续变量，部分为整数变量。混合整数规划在实际问题中很常见，例如生产调度、设施选址等。

2.5 本章小结

第3章 基于基因算法的车辆任务卸载算法

3.1 引言

作为下一代智慧交通的电动智能汽车有许多的问题需要解决，其中电池问题尤为重要，这关系着汽车的行驶里程。而任务卸载是其中较有希望的一个减少能源消耗，提高能源利用率的一种方式。

在最近流行的边缘计算技术中，不仅可以将任务分配给附近的边缘服务器，还可以分配给附近资源丰富的用户。但是其中很多问题需要解决，例如，如何衡量计算资源，如何实时的分配任务，如何分配任务更加公平。

在该问题中，将任务分配给附近的车辆，来提高资源的利用率以及最小化系统的能耗。本论文将移动边缘计算场景车联网中任务卸载的问题抽象为一个数学规划的形式，其中，目标是最小化所有能源消耗的平方，这可以兼顾能耗最小化与公平。

约束包括：所有的任务必须有车辆来做，所有的车辆能够在规定时间内完成任务，以及为了保证通讯的质量，参与资源共享的智能汽车的距离要在一定的距离内。

该问题是一个非线性整数规划问题，没有多项式时间范围内的解决办法。

在本文中，本论文提出了基于用于电动智能汽车的遗传算法。本论文评估通过仿真实验验证了算法的性能实验。实验结果表明，本论文的算法可以达到节能的目的。在未来的研究中，本论文计划建立在数据大小和指令数量之间函数关系并考虑最后期限限制，同时，也将考虑卸载过程的真实性。

3.2 系统模型

本节主要介绍了系统模型，分为两个部分，系统场景，车辆计算能力模型，以及车辆能耗模型。

3.2.1 系统场景

在本文中，如图 4 所示，层次结构由车辆云、边缘云和中心云组成。中心云负责全局调度，它执行的任务包括执行复杂的计算任务和进行全局的决策。边缘云是车辆云的控制器，负责创建、维护和删除车辆云。车辆云由一定范围内的、参与共享其计算资源的智能车辆组成。基站、边缘服务器和远程的中心云服器通过有线来连接，基站和边缘服务器物理上距离十分接近，因此可以看作一体，后文中以边缘云来称呼这两者的总和。

每辆车都可以访问边缘云并与其进行通讯。车辆可以通过将任务卸载到其他车辆的方式来节省能源。这样可以提高整体资源利用率。

任务卸载过程如下：首先，实时流量信息被传输到云服务器进行分析。实时信息包括目的地、当前位置、时间、车速等。云服务器在汇总数据后，进行大数据分析以获取道路拥堵情况，然后推断未来某个时间段内车辆的位置信息，并将结果返回到车辆附近的边缘云。

这些车辆在未来一段时间内将会参与资源共享。然后，边缘云根据这些信息分配任务，并将车辆分为提供者和请求者，他们在 T 时间段共享资源。

图：系统场景层次结构

3.2.2 车辆计算能力模型

如何定义车辆的计算资源是建立起问题描述非常重要的一步，在此，我使用计算每秒执行的指令条数来刻画计算资源，本论文将资源共享时间定义为 $\mathcal{T}=\{1,\ldots,T\}$.。车辆数量定义为 N 。

本论文使用元组 $J_{it}=\{r_{it},r'_{it},d_{it}\}$ 来表示时间为 t 时，车辆 i 的任务大小，d_{it} 是任务的数据大小。

任务分为必须在本地执行的任务，以及可以分配出去的任务。这一点并不抽象， 例如有些涉及隐私的任务，或者是要求实时性的任务，就必须在本地执行。

在时间 t 时，r_{it} 是车辆 i 必须要在本地执行的任务，r'_{it} 是车辆 i 能够分配给别人的任务，并且它们的值是由需要的指令条数来表示的。

为了更好的描述这个模型，本论文定义 $\mathbf{X}_t=\left\{x_{ij}\right\}\in\{0,1\}^{N\times N}$ 为分配矩阵，如果 $x_{ij}=1$，代表车辆i执行车辆j所卸载的任务。注意，如果$x_{ii}=1$，车辆 i 所有的任务都在本地执行。

本论文将车辆i在时间为t 的容量定义为C_{it}时。在时间t，当车辆i被选为提供者(P)，所有需求者需要的资源需要少于这辆车的容量。公式化表达如下：

$$\sum_{j=1}^{N}x_{ij}^{t}\cdot r'_{jt}\le C_{it},i=1,\ldots,N \tag{3.1}$$

一个关键的步骤是怎样衡量车辆的计算资源，在本篇文章中，计算资源定义在每秒能够执行的指令条数上。

公式化表达如下：

$$C_{it}=M_{it}\cdot\Delta T-r_{it} \tag{3.2}$$

其中，ΔT 是资源共享的持续时间，并且它是一个比建立车辆边缘网络的更小的时间粒度，在ΔT 秒后，分配矩阵会发生变化。

对于一个单核的 CPU，每秒能够执行的指令条数(m_{it}) 和 CPU 的频率(f_{it})有如下的关系:

$$M_{it} = v_i \cdot f_{it} + \theta_i \tag{3.3}$$

其中，v_i 和 θ_i 是待估计的参数。

最后，公式（2）中的 CPU 的容量 C_{it} 在能被下面的公式计算出来：

$$C_{it} = (v_i \cdot f_{it} + \theta_i) \times \Delta T - r_{it} \tag{3.4}$$

3.2.3 车辆能耗模型

当本论文考虑车辆的能源消耗时，本论文将它分为两个部分（1）计算所需要的能量以及（2）传输所需要的能量。

根据[9][10][11][12]，计算所需要的能耗能够被下列公式计算：

$$E = \lambda_i \cdot f_{it}^3 \cdot \Delta T \tag{3.}$$

其中 f_{it} 是 CPU 在 t 时刻的频率。如果一辆车被选为了需求者（R），这辆车的频率会下降 $f_{it}^{'}$。因为任务被分配出去了，所以他消耗的能量会减少。消耗的能量能够被以下公式计算：

$$E_i^{save} = (\lambda_i \cdot f_{it}^3 - \lambda_i \cdot f'^3_i) \times \Delta T, \forall i \in R \tag{3.}$$

传输能耗和传输的时间有线性关系，传输的时间取决于数据大小和传输速率的比值(b_{ij})：

$$E = P_0 \cdot \frac{d_{it}}{b_{ij}} \tag{3.}$$

其中，P_0 是传输率。尽管功率可以变化 [18]、[19]、[20]，但为了简单起见，本论文在本文中使用固定值。

最大的传输速率（b_{ij}）可以通过香农公式计算：

$$b_{ij} = W \log(1 + \mathrm{SNR}) \tag{3.}$$

其中，SNR 是信噪比，W 是频道的带宽。因为它和每一个场景相关，本论文将它考虑为一个常数。

对于每一辆车，接收信息所消耗的能量是：

$$E_i^{rec} = \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})}, \quad i = 1, \ldots, N \tag{3.9}$$

对于每一辆车，发送信息所需要的能量是：

$$E_i^{send} = \sum_{i=1, i\neq j}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+SNR)}, \quad j = 1, \ldots, N \tag{3.}$$

定义 E_t^{blnc} 是车辆 i 在时刻 t 所消耗的能量总和，它可以被这么计算

$$E_{it}^{blnc} = \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})} + \sum_{j=1,, j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\mathrm{SNR})} + \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N \tag{1}$$

本论文算法的设计目标是:最小化所有车辆能量消耗的平方。平方和和普通的和相比，平方和更容易受到极端值的影响，因为平方项会使得较大或较小的值对平方和的贡献更加显著。

这个目标既能够考虑最小化能量消耗，又能考虑公平，也就是说平衡了能量消耗和公平。公式表示如下文：

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2$$

3.3 问题建模

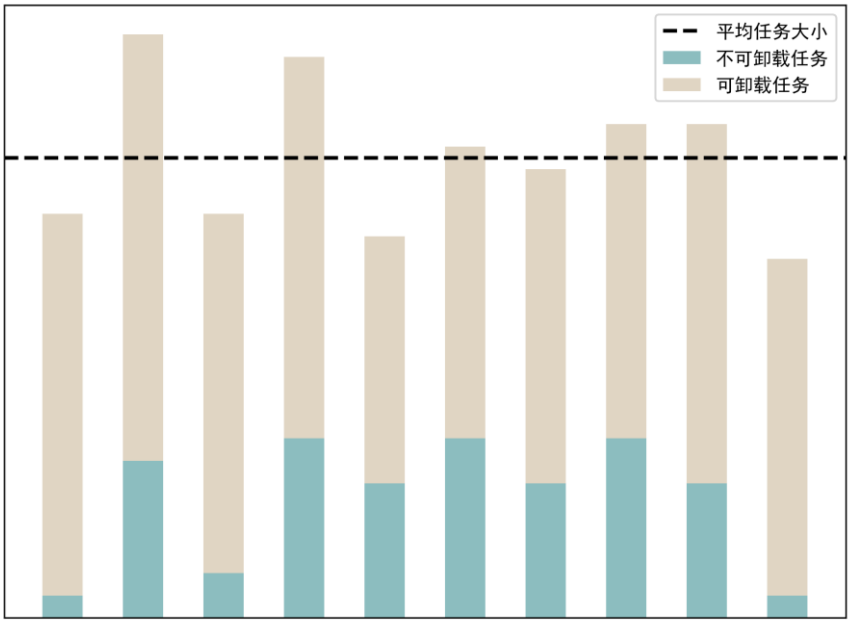
在本节中，本论文将问题分为两种情况来建模，一种是任务可分情况（3.3.1 节），另一种是任务不可分情况（3.3.2 节）。当任务可分的情况下，问题能够直接通过规划的方式解决；当任务不可分时，不能通过规划的方式来解决。

3.3.1 任务可分情况

当车辆在进行某些同质任务时，可以将大任务任意的拆成一些小任务，因为任务的粒度太小，因此可以近似看成任何一个任务可以随意拆分然后分配到任意的车辆上，也就是任务可分的情况。

在任务可分时，根据本地留存的高优先级任务的大小又分为：高于平均任务（大任务），和低于平均任务大小（小任务）这两种情况。

（1）小任务情况



图：情况 1 小任务情况

当本地任务低于平均任务大小（小任务）时，该问题较为简单，直接将所有任务平均分配到车辆上就能解决。

该问题等价于：首先将固定值 C 切制成 n 个数，是的他们的三次方之和最小。设切分的数分别为$a_1,a_2,\cdots,a_n$，根据均值不等式：$\sqrt[3]{\frac{\sum_{i=1}^{n}a_i^3}{n}}\geq\frac{\sum_{i=1}^{n}a_i}{n}$，可得

$$\sqrt[3]{\frac{\sum_{i=1}^{n}a_i^3}{n}}\geq\frac{\sum_{i=1}^{n}a_i}{n}$$

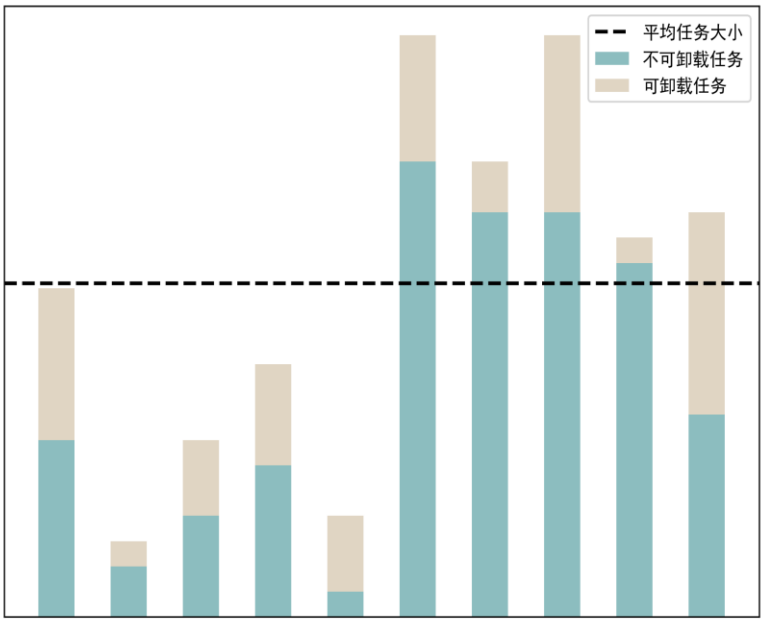
即

$$\sum_{i=1}^{n} a_i^3 \geq \frac{n(\sum_{i=1}^{n} a_i)^3}{n^3} = \frac{C^3}{n^3}$$

因此，要使$\sum_{i=1}^{n} a_i^3$最小，需要让$\sum_{i=1}^{n} a_i$的值尽可能平均分配给 n 个数，即令$a_1 = a_2 = \cdots = a_n = \frac{C}{n}$。

当本地任务是小任务时，能搞保证参与资源共享的车辆，任务的大小可以卸载为平均值。

（2）大任务情况



图：情况 2 大任务情况

但是当任务高于平均任务大小（大任务）时，该问题需要进行分类的讨论，是否还需要将再给有大任务的汽车分配任务。

该问题等价于：有一个固定值 C，将其切割为 n 个数，要求这 n 个数的三次方最小，已知有 m 个数必须大于$\frac{C}{n}$，如何切割。

在有限制条件的情况下，该问题可以使用拉格朗日乘数法来求解。设切分后的数为$a_1,a_2,\cdots,a_n$，且有 m 个数大于$\frac{C}{n}$。

有以下约束条件：

$$\begin{cases}\sum_{i=1}^{n} a_i = C, \\ \sum_{i=1}^{n} \text{count}(a_i, \frac{C}{n}) = m\end{cases}$$

其中，count 表示指示函数，当$a_i > \frac{C}{n}$时，取值为 1，否则为 0。

目标为：

$$\min \sum_{i=1}^{n} a_i^3 \tag{3.}$$

根据拉格朗日乘数法，构造带拉格朗日乘数$\lambda_1,\lambda_2,\cdots,\lambda_n$的拉格朗日函数：

$$L\left(\{a_i\}_{i=1}^{n},\{\lambda_i\}_{i=1}^{n}\right) = \sum_{i=1}^{n} a_i^3 + \sum_{i=1}^{n} \lambda_i \left(a_i - \frac{C}{n}\right) + \mu \sum_{i=1}^{n} \text{count}\left(a_i, \frac{C}{n}\right) \tag{3.}$$

其中，μ 是 Lagrange 乘子。

对L 求导并令其等于 0，可得：$3a_i^2 + \lambda_i - \mu\,\text{count}\left(a_i, \frac{C}{n}\right) = 0$。将上式分为两种情况：

当$a_i \le \frac{C}{n}$时，μ=0，则有$a_i = -\frac{1}{3}\lambda_i$。

当$a_i > \frac{C}{n}$时，则有$\lambda_i = -3a_i^2$，μ 取任意值。

将a_i代入约束条件中可得：

$$m = \sum_{i=1}^{n} \text{count}(a_i, \frac{C}{n}) = \sum_{i=1}^{n} \text{count}(\lambda_i, 3(\frac{C}{n})^2) \tag{3.}$$

即有m个λ_i满足$\lambda_i > 3(\frac{C}{n})^2$。假设它们的下标$i_1,i_2,\cdots,i_m$，则有：

$$\sum_{j=1, j\neq i_k}^{n} a_j = C - \frac{\sum_{k=1}^{m} \lambda_{i_k}}{n}, \quad k = 1,2,\cdots,m \tag{3.}$$

进一步代入目标函数 $\sum_{i=1}^{n} a_i^3$ 中，可以得到：

$$\sum_{i=1}^{n} a_i^3 = \left(\frac{C - \sum_{k=1}^{m} \lambda_{i_k}}{n}\right)^3 \cdot (n-m) + \sum_{k=1}^{m} \lambda_{i_k}^3 \tag{3.}$$

要求 $\sum_{i=1}^{n} a_i^3$ 最小，就要使 $\sum_{k=1}^{m} \lambda_{i_k}^3$ 最小。该问题是一个无约束优化问题，可以使用导数来求解。可得，最优的分配方案是，将那几个本地任务是大任务的车辆将他们所有可分得任务分出去，然后再均分剩下的任务。

3.3.2 任务不可分情况

在任务不可分时，该问题没有多项式时间内的解决方法，因此，在下一节中，本文打算通过智能算法来获得一个近似解，来满足靠近最优解和卸载方案计算时间的平衡。

该问题的优化目标是通过调度各个车辆的任务，最小化所有能耗的平方，该问题最终被建模为如下形式：

$$\min \quad \sum_{t=1}^{T}\sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\text{SNR})} + \sum_{j=1, j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\text{SNR})} + \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N$$

$$\min \quad \sum_{t=1}^{T}\sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\text{SNR})} + \sum_{j=1, j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\text{SNR})} + \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N$$

(3)

$$\sum_{j=1}^{N} x_{ij}^t \cdot r'_{jt} \leq C_{it} \tag{4}$$

$$\sum_{i=1}^{N} x_{ij}^{t} \geq 1 \tag{5}$$

$$f_{it} \geq 0 \tag{6}$$

$$x_{it}^{t} \in \{0,1\} \tag{7}$$

正如前文所提到的，公式（14）说明了对于每一辆车辆 i，需求的总和不能超过他的容量。公式（15）说明了车辆 i 的任务，必须要被执行，无论是本地执行还是分配给其他车辆。

。

3.4 基于基因算法的问题求解

因为[公式]这是一个 NP 难（NP-Hard） 问题，多项式时间内求解不出该问题的精确解，直接计算法求解是不现实的，因此采用了基因算法求解该问题

基因算法是一种典型的启发式智能优化算法，其鲁棒性来源主要有以下三点：首先来源于个体多样性，通过引进交叉、变异等操作，使得不同个体之间进行信息的交流，有助于保持种群的多样性，从而不易陷入问题的局部最优解。其次来源于选择策略。基因算法中的选择策略来自于达尔文自然选择学说中的“适者生存”原则，通过对适应度高的个体进行选择和繁殖，逐步提高整个种群的适应性，这提高了算法的鲁棒性。最后来源于参数设置。基因算法中的参数设置，例如交叉的概率、变异的概率，这对算法的表现具有较大影响。通过对参数的合理设置，可以提高算法的鲁棒性，并且使其更容易达到全局最优解。因此本文选择了能适用于离散问题的基因算法来求解该问题。

以下是基因算法的实现步骤：

(1) 使用初始化参数来初始化种群：设置种群的大小，每个个体的基因编码方式和以及初值范围，设置循环最大次数为$it_{\max}$，然后根据这些参数生成初始种群。同时设置目标函数$f(x)$以及约束函数族$G(x)$。

(2) 评估适应度：将每个个体的基因编码字符串解码成相应的解，然后计算其适应度值。适应度值是个体在解空间中执行任务的优劣程度的量化指标。

基因算法二进制解码公式如下：

$$\delta = \frac{U_x - L_x}{2^l - 1} \tag{3.}$$

$$x = L_x + \delta \cdot \sum_{i=1}^{l} A_i 2^{i-1}$$

其中，δ是偏移量，l是二进制码的长度，A_i是第i位二进制位的值，x是解码后的值。例如，给定$\mathrm{U_x} = 5$，$\mathrm{L_x} = -5$以及$l = 10$，那么$\delta = \frac{5-(-5)}{2^{10}-1} \approx 0.009775$。

基因算法二进制编码二进制长度公式如下：

$$l = \left\lceil \log_2 \left(\frac{U_x - L_x}{\mathrm{SearchAccuracy}} \right) \right\rceil \tag{3.}$$

其中，$\mathrm{U_x}$和$\mathrm{L_x}$分别是解的最大值和最小值，SearchrAccuracy 是求解的精度。例如，给定$\mathrm{U_x} = 5$和$\mathrm{L_x} = -5$，精度是 0.01，那么$\mathrm{l} = \left\lceil \log_2 \left(\frac{5-(-5)}{0.01} \right) \right\rceil = 10$，也就是说，需要十个二进制位来表示解。

适应度计算公式如下：

$$fit_i = e^{-\frac{y_i}{mean(y)}} \tag{3.}$$

其中，$y_i = f(x_i)$，$mean(y)$是种群中所有的个体的平均值。

(3) 选择操作：按照轮盘赌选择的规则，从种群中选择一部分较好的个体进行进化操作，以期望得到更加优秀的后代个体。在进行轮盘赌选择时，标准化处理的公式为：

$$p_i = \frac{fit_i}{\sum_{i=1}^{n} fit_i} \tag{3.}$$

其中，n 是种群中的个体数量。

(4) 进行遗传操作：包括交叉操作和变异操作。交叉操作是将两个个体的染色体进行配对，然后按照一定的概率产生新的后代染色体；变异操作是随机改变个体染色体的一个或多个基因，以增加种群的多样性。

(5) 更新种群：这是“适者生存”法则在基因算法中的应用。将新生成的后代个体插入到种群中，也就是按照一定的比例，某些适应度高的个体替换掉适应度低的个体。

(6) 判断遗传算法是否满足停止准则：停止的准则可以设置最大迭代次数或者在种群中出现一定数量的相似个体等条件，决定是否终止遗传算法。如果不满足条件则跳转到第（2）步。

3.5 仿真实验

3.5.1 试验环境

本章实验环境为：处理器为：Intel(R) Core(TM) i7-7700HQ；显卡的型号为：NVIDIA GeForce RTX 1050；机带 RAM 为 8GB；操作系统为 Windows10 专业版；Python 版本为 3.8，Numpy 版本为 1.19.2，matplotlib 版本为 3.3.2。

3.5.2 仿真参数设计

表 1：仿真参数

参数名称	值/分布	参数名称	值/分布
f_i	[700, 1900]	ΔT	1
v_i	7.683	r_{it}	U[820, 10000]
θ	-4558.52	r'_{it}	U[200, r_{it}]
λ	0.00125	P_0	0.2
W	10 MHz	SNR	677

本文将变异概率设置为 0.01，参数用于控制每个个体发生变异的概率。该概率越高，个体发生变异的可能性就越大。一般情况下，变异概率会设置为较小的值，以保持种群的多样性，并防止算法陷入局部最优解。

本文选择了 7 种种类的参与资源共享的车辆数目，分别是{10，15，20，25，30，35，40}。在算法中，种群数量被设置为 40。而当车辆数量为 35 和 40 时，种群数量被设置为 150。因为当车辆数量为 35 和 40 时，种群的规模太小，算法不会收敛或者不能很好的收敛到最优值附近。

3.5.3 评价标准

本论文设置了个指标来衡量本论文的实验，分别是，性能指标，公平指标，种群离散程度。

在实验中，本论文的性能指标是指节能的百分比。节能的百分比，其定义如下：

$$P=\min\left(100\cdot\left(1-\frac{\sum_{i=1}^{N}E_{it}^{blnc}}{E_{loc}}\right)\right),t=1,\ldots,T \tag{3.}$$

其中，E_{loc} 是本地执行任务所需要消耗的能量，

为了反映本论文任务分配的公平性，本论文在标准差的基础上定义了公平系数（FC）。与标准差一样，数值越小，公平性越高。标准差定义如下：

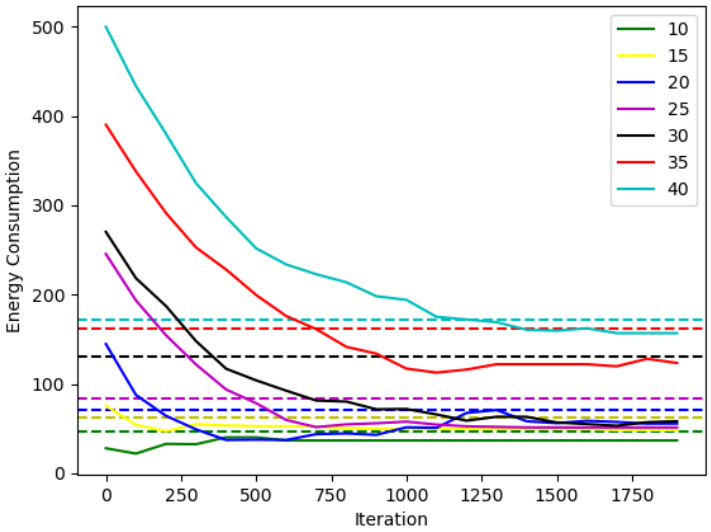
$$\sigma=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}} \tag{3.}$$

FC 的定义如下：

$$FC=\max\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(E_{it}^{\text{blnc}}-\bar{E}\right)^2}}{\bar{E}},t=1,\ldots,T \tag{3.}$$

其中，t 是迭代的次数，$\bar{E}$ 是所有车辆能耗的平均值。

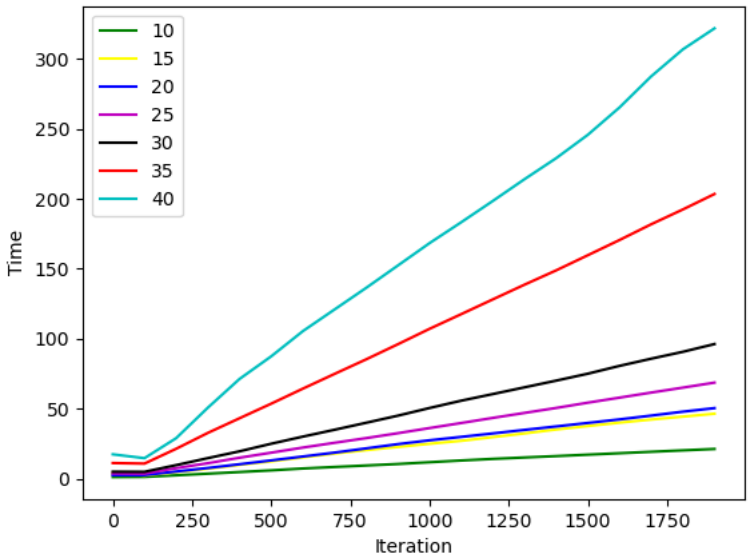
3.5.4 实验结果



图：能耗图

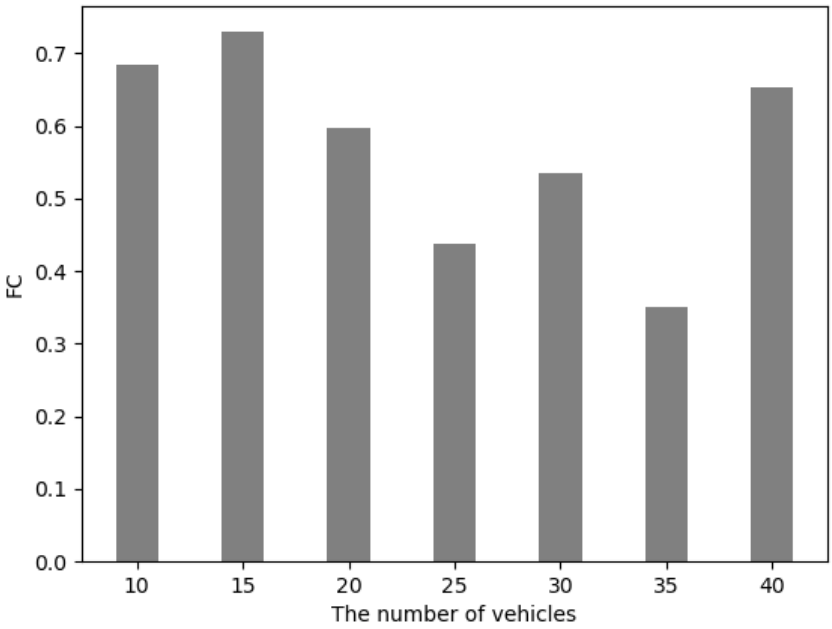
在图 2 中，虚线是任务没有负载出去而在本地执行所需要消耗的能量。本论文可以看到，当车辆数为 10 时，该算法迭代几十轮就可以得到很好的结果。当车辆数为 20 时，需要 350 次迭代才能得到一个较好的结果。而当车辆数量为 30 时，能耗在 750 次迭代之前变化较大，而在 750 次迭代之后变化就会明显变小。也就是说，能源消耗的减少率在 750 次迭代前较快，而在 750 次迭代后，能源消耗的减少率就会降低。当车辆数为 40 时，几乎没有节能效果。

本论文可以得出结论，随着车辆数量的增加，迭代的次数也在增加。因此，当本论文在边缘服务器上运行该算法时，本论文需要合理地调整迭代次数，因为车辆数量的变化会显著的影响算法的运行时间。



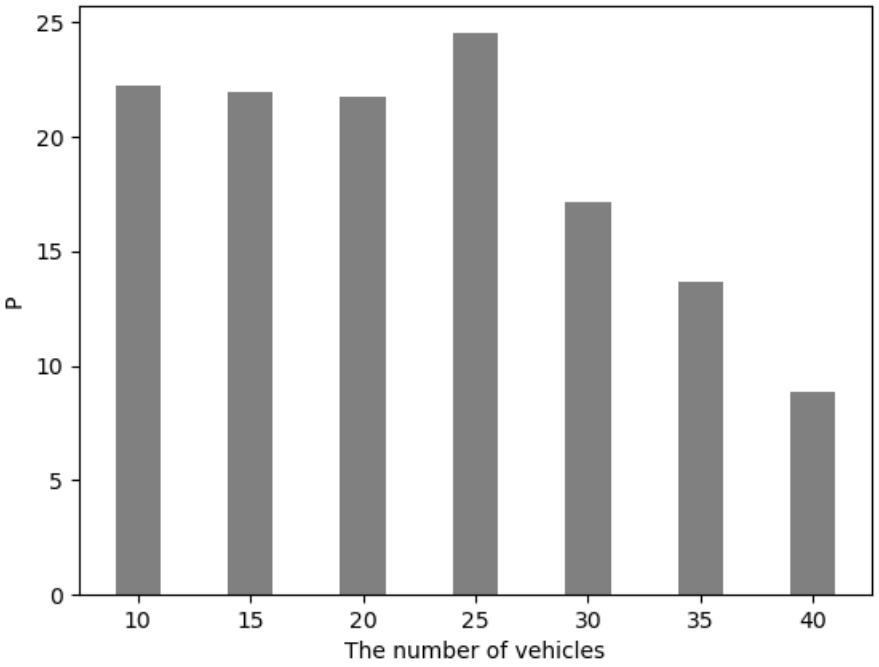
图：

在图 3 中，经过 100 次迭代以后，所有数目运行时间和迭代次数呈线性关系。然而，当迭代次数固定时，算法的运行时间和车辆之间的关系不是线性的，而是非线性的，参与资源共享的车辆的数越多，运行时间增加的越多。



图：平衡因子 FC 的值

如图 4(a)所示，随着车辆的增加，车辆能量消耗的平衡首先增加，然后减少。当车辆数目为 30 和 40 时，能耗的平衡系数反而增加。



性能度量 P 的值

如图4(b)所示，当车辆数量在10到25辆之间时，本论文的系统可以节省20%以上的能量消耗，但是当车辆数量为30辆时，只有17%的能量消耗。但当车辆数为30时，只能节省17%，而当车辆数为40时，几乎可以节省20%。当车辆数量为40辆时，几乎没有节省能量。

综合考虑性能测量、算法运行时间以及平衡因子，选择25辆汽车参与资源共享是非常合适的。

3.6 本章小结

本章所研究的是车联网领域的任务卸载和分配的方案，通过能耗优先的原则设计出一套卸载算法。首先，本章详细介绍了车联网中任务卸载的系统模型，并对其中的要素进行了说明，包括系统场景、车辆计算模型及车辆能耗模型。接着，作者将问题建模为数学规划形式，提出了一种基于粒子群算法的卸载与分配策略，然后设计了相关评价标准，并对实验结果进行了分析。评价标准表明，该算法能够很好地提高资源利用率，同时还兼顾了公平性。

然而，在研究过程中也发现了一些缺陷，比如，基因算法存在着编码和解码的过程，需要进行交叉和变异操作，这导致了计算量太大和计算时间过长等问题。因此，在下一步的研究中，本论文将引入新的智能算法，并将其改进为适用于离散问题的算法，以加快算法的运行时间。

本章研究了一种车联网环境下能耗优先的计算卸载和分配的方案。本章首先介绍了车联网中任务卸载的系统模型，包括，系统场景，车辆计算模型、车辆能耗模型，然后将问题建模为了数学规划形式，提出了一种基于基因算法的任务卸载分配算法，然后设计了相关评价标准及分析了实验结果。评价标准显示，该算法能够很好的提高资源的利用率，同时也很好的兼顾公平。

但在本章的研究过程中发现了一些不足：基因算法存在着编码和解码的过程，以便于进行交叉和变异操作，这带来了计算量太大以及计算时间过长的问题。因此，

下一张会引出较新的智能算法并将其修改为能解决离散问题的算法，加快算法的运行时间。

第4章 基于贪心法调整的离散侏儒猫鼬算法的任务卸载算法

通过上一章节的研究，本论文完成了车联网中对任务卸载算法研究的任务，但也指出了存在计算量太大以及计算时间过长等问题。因此本章将通过使用新的智能算法并将其修改为能解决离散问题的算法来解决该问题。

4.1 侏儒猫鼬算法

和粒子群、基因算法类似，侏儒猫鼬算法（Dwarf Mongoose Optimization Algorithm，DMOA）也是一种基于动物行为的启发式优化算法，能够用来解决线性以及非线性的优化问题。该算法是由 Jeffrey O. Agushaka 等科学家于 2022 年提出的，算法的灵感源于侏儒猫鼬的捕猎行为，通过模拟侏儒猫鼬在觅食和捕猎时的行为特征来进行对问题的求解。

侏儒猫鼬算法在搜索过程中，不断尝试各种不同的位置和状态组合，计算适应度，并据此进行调整，最终找到最优解或次优解。

像其他智能算法一样，侏儒猫鼬算法借鉴了侏儒猫鼬在觅食和捕猎时的行为特征，同时也吸收了其他算法的特征。

侏儒猫鼬是一种小型哺乳动物，以体型较小昆虫以及蚂蚁，蚁蛇等为食，它们具有机敏的行动能力和快速适应环境的能力。

侏儒猫鼬通常生活在在一个母权制的家庭群体中，由一对阿尔法夫妻（leader）领导种群生活在一起。无论在每个年龄组中，雌性都比雄性地位高，而幼崽比它们的哥哥姐姐地位高。这些群体中的劳动分工和利他主义是哺乳动物中最高的，他们根据年龄和性别，不同的猫鼬充当警卫、保姆、攻击捕食者和攻击同种入侵者。

为了模仿它们的觅食行为，优化过程建立了三个社会结构：阿尔法小组，侦察兵小组，保姆小组。

表 4.2 常用算法

算法名称	主要思想
粒子群算法	鸟群或鱼群等生物群体的行为
模拟退火算法	固体材料退火过程中的原子结构调整
遗传算法	生物遗传和进化理论
蚁群算法	蚂蚁在寻找食物时的行为

4.1.1 阿尔法小组

阿尔法小组是又进行出发去觅食（探索新的解）的行为的种群，食物的位置由下面的公式给出：

$$X_i^{t+1} = X_i^t + \text{phi} \times \text{peep} \times (X_i^t - X_{peep})$$

其中，X_i^{t+1} 是新产生的解，phi 服从于在[-1,1]之间的均匀分布。peep 是雌性首领的位置数量，X_{peep} 是选出的雌性首领的位置。

雌性首领在阿尔法小组中产生，每个个体都有可能成为雌性首领，适应度高的个体成为雌性首领的可能性更大，适应度差的个体成为雌性首领的可能性小，每个个体成为雌性首领的概率计算方式如下：

$$\alpha = \frac{fit_i}{\sum_{i=1}^{n} fit_i}$$

其中，fit_i 是第 i 个个体的适应度，n 是种群数量。

4.1.1 保姆小组

保姆通常是附属的群体成员，和幼仔呆在一起，定期轮换，让母亲带领其他成员进行日常觅食。她通常在中午和晚上回来给幼崽喂奶。也就是说，保姆小组的成员不进行觅食行为，同时，保姆的数量取决于种群规模，种群越大，保姆数量越多。

在算法中，当一个个体很久没有进行觅食（更新）时，就说明不是陷入了局部最优，就是这个解太差了，更新的可能比较小，将其放入保姆小组中，并重新进行初始化。

4.1.1 侦察兵小组

侦察兵寻找下一个睡觉的土堆（睡眠丘），因为猫鼬不会回到先前睡觉的土堆，这保证了探险的有效性。在该算法中，侦察兵小组和阿尔法小组是一个小组，首先会进行阿尔法小组的跟新行为，然后再进行侦察兵小组的更新行为。

睡眠丘的计算公式如下：

$$sm_i = \frac{fit_{i+1} - fit_i}{\max\left\{\left|fit_{i+1}, fit_i\right|\right\}}$$

睡眠丘的平均值反映了该轮迭代的移动范围，φ 计算公式如下：

$$\varphi = \frac{\sum_{i=1}^{n} sm_i}{n}$$

CF 是表示侦察兵小组移动能力的参数：

$$CF = \left(1 - \frac{iter}{T}\right)^{\left(2\frac{iter}{T}\right)}$$

其中，$iter$ 为当前迭代次数，T 为最大迭代次数，当迭代次数增加时，其值会减小，也就是说，随着迭代次数增加，侦察兵小组移动能力会减小。

侦察兵小组的更新方式如下：

$$X_{i+1}=\begin{cases}X_i - CF * \text{ phi } * \text{ rand } * \left[X_i - \overrightarrow{\mathbf{M}}\right] & \text{if } \varphi_{i+1} > \varphi_i \\ X_i + CF * \text{ phi } * \text{ rand} * \left[X_i - \overrightarrow{\mathbf{M}}\right] & \text{else}\end{cases}$$

其中，phi 服从于[0,1]之间的均匀分布，$\overrightarrow{\mathbf{M}}=\sum_{i=1}^{n}\frac{X_i \times sm_i}{X_i}$ 。

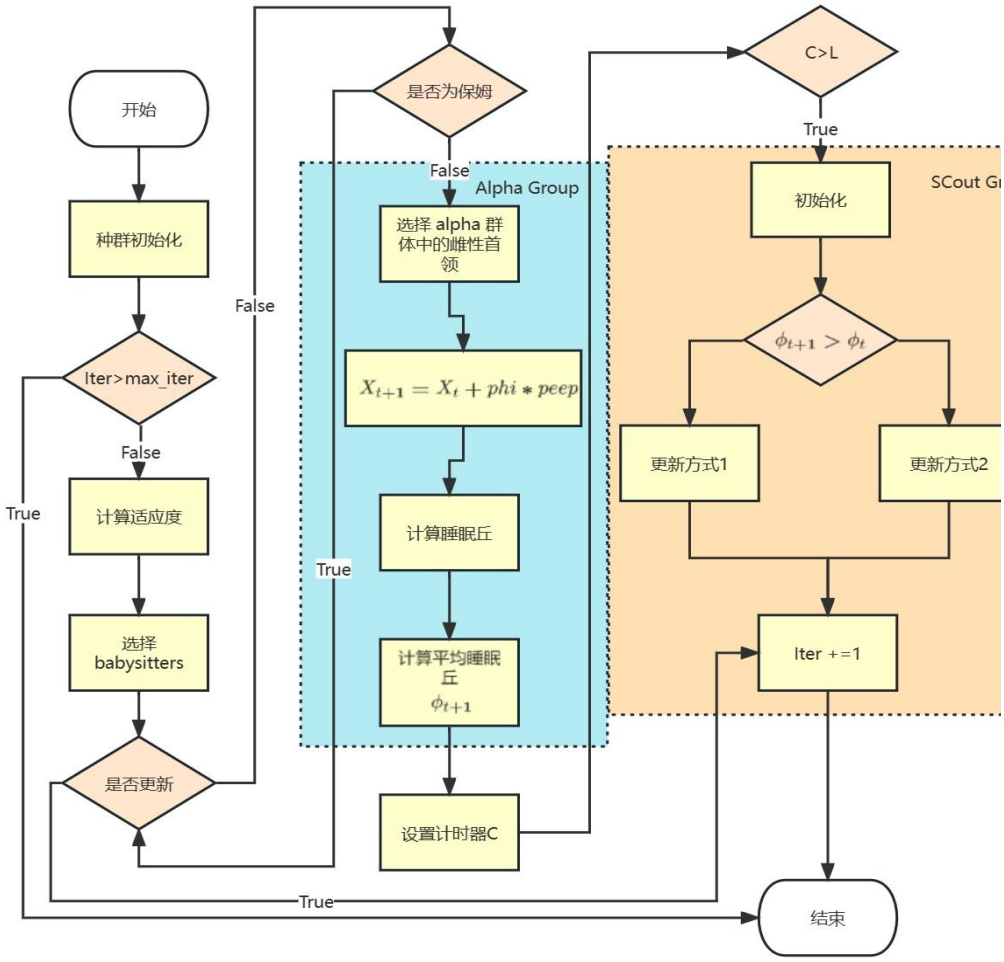
4.1.1 算法流程

优化从阿尔法小组出发(探索空间)觅食开始，将保姆小组留在巢穴中。一旦找到觅食地点，阿尔法小组就一直觅食到中午（保留更新值）。

当他们返回交换保姆时，按照保姆交换标准，很久没觅食的成员会成为保姆。一旦保姆被交换，他们就不会回到先前觅食的地方，以避免过度放牧。

侦察兵会发现一个新的觅食地点，并通知雌性首领，带领家族到达新的地点。

可以看出，侏儒猫鼬算法中种群的主体部分（阿尔法小组或者侦察兵小组）一次迭代中会进行两次更新行为。



图：侏儒猫鼬算法流程图

显而易见，侏儒猫鼬算法在进行每一步的更新时，使用了类似于梯度的更新方式：$X_i^{t+1} = X_i^t + \text{phi} \times \text{peep} \times (X_i^t - X_{peep})$，因此该算法只能解决连续形式的问题，不能解决离散形式的问题，需要本论文将其改造为像基因算法一样的离散型算法。

4.2 离散算法相关的知识

本节介绍了想要将算法修改为离散算法的相关知识。

4.2.1 优化问题的描述形式

本论文的问题和并行机器调度问题（parallel machines scheduling problem）有同质性，并行机器调度问题是指在一些并行处理的机器上调度执行一组作业的问题。该问题与日常生产生活紧密联系，通常出现在工业车间生产、物流快递配送、计算机任务分配等领域，可以帮助企业有效地安排生产计划、降低成本、提高效率，也可以在计算机系统中动态地调度任务，提高系统总体性能。

该问题的形式化描述如下：有一个工件（任务）集合 $J=\{J_1, J_2,\ldots J_n\}$，其中，任务共有 n 个。还有一个机器的集合即 $M=\{M_1, M_2,\ldots,M_m\}$，共有 m 个机器，所有的机器都是一样的，并且每台机器一次只能处理一个任务。每一项任务应该必须在其中一台机器上进行，在所有机器上运行的时间不会改变。任何一台机器处理任务i所需的时间由 p_i 给出。调度中分配给机器 M_j 的作业的子集用 S_j 表示。作业一旦开始处理，就必须在不中断作业的情况下完成，即，任务一旦开始做那就必须做完。

此外，作业之间没有优先关系，也就是说，在任何时间，作业可以不受限制地随意分配到任意一台机器上。研究了以最小化最后一个工件离开系统的时间为目标的最优工件分配问题，即最大完工时间准则。

最小最大完工时间的混合整数规划公式如下:

$$\min \max_{1\le j\le m} \sum_{i=1}^{n} p_i x_{ij}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij}=1 \quad i=1,\ldots,n$$

$$x_{ij} \in \{0,1\}$$

公式~(ref{pmpmin1})表明，本论文应该耗时最久的那台机器尽可能地小，因为所有的作业完成才算是完成了任务。公式~(ref{pmpst1})表明每个作业必须分配到一台机器上。如果 $x_{ij}=1$，意味着作业 i 被分配到 j 的机器上。

然而，工件问题这个形式的空间复杂度是 $O(mn)$，而且因为每个变量都是二进制变量，所以时间复杂度是 $O(2^{mn})$。本论文将并行机器的调度表述为另一种形式：

$$\min \max_{1\le j\le m} \sum_{1\le i\le n, x_i=j} p_i$$

$$\text{s.t.} \quad x_i \in \{0,1,\ldots,m\}, \quad i=1,\ldots,n$$

其中，工件问题该形式的空间复杂度为 $O(n)$，时间复杂度为 $O(m^n)$。

工件问题每个变量之间的取值有相关性，如几个变量的值交换，很大的概率就会导致最优解变成次优解甚至是劣解。

后文中，称呼每一个 x_i 为变量，所有变量的解集合即 $X=(x_1,x_2,...x_n)$ 称为该问题的一个解。

在后文设计离散形式的算法时，解的表示是一个关键的步骤，解具有与问题域相关的必要的信息。例如上文中使用基因算法解决问题，基因算法为了表示解，将解重新进行了编码，用二进制字符串的形式表示解。为了建立算法和问题之间的联系，对于有 n 个任务的问题，对应的解的维度也是 n。因此，在后文进行离散算法表示时，并行机器调度问题的解是用一个长度等于作业数量的数组表示的。

图：解的表示

如图所示，一共有六个任务，第 1，2，5 号任务放在第一台机器上处理，第 3，

4，6 号任务放在编号为 2 的机器上进行处理。

4.2.2 各变量之间的相关性

本节通过两个实验来验证各个变量之间的相关性。第一个实验是验证解中某些变量的取值发生改变对目标函数的影响，第二个实验是验证两个解进行交叉行为对目标函数的影响。第二个实验实际是第一个实验的更进阶的版本，第一个实验交换的是同一个解中的某些变量，第二个实验交换的是不同解中的连续的变量。

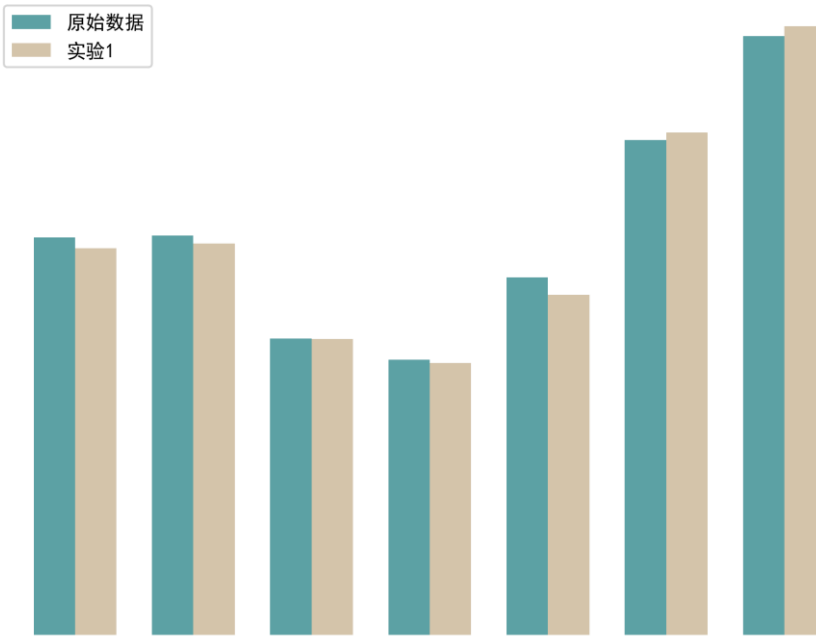
这两个实验都说明了解变量之间的相关性，为了消除变量之间的相关性，本论文生成了解以后，还需要对解的变量进行调整。

实验一：验证解中某些变量的取值发生改变对目标函数的影响。

（1） 随机生成初始解，且保证解的合法性，计算目标函数。

（2） 对每个解的某些位置的变量，交换他们的取值。

（3） 计算完成操作后的解所对应的目标函数，生成图像。



图：

在图中，绿色是没有更新的解所对应的函数值，黄色柱状图是更新解以后所对应的变量值，可以看出，如果仅仅只是交换两个变量值的更新方式，更新和不更新没有什么显著的区别。

实验二：两个解进行交叉行为对目标函数的影响。

（1） 随机生成初始解，且保证解的合法性，计算目标函数。

（2） 解两两配对，进行交叉操作。

（3） 计算完成操作后的解所对应的目标函数，生成图像。

乘法运算法($\dot{O}$)，解中每一个等于变量进行，值为另一个解相对应变量的值，然后进行交叉操作。

下面将解释什么是交叉操作。交叉操作是基因算法的一个解更新行为，是指在两个父代个体（解）之间交换染色体上的一部分基因信息（变量）并生成新的子代个体（解）。作为进化算法的重要步骤之一，用于产生更好的后代解，并增加群体的多样性。

交叉操作通常需要满足如下两个条件：首先，确定交叉方式，即交叉方式有多种形式，例如单点交叉、多点交叉、均匀交叉和线性交叉等，本文选择了双点交叉的方式。其次，保证基因信息的完整性和正确性，即交叉操作需要保证新生成的解是有效的解，因此需要保证每个基因在子代中都不缺失或重复。

下面讲解双点交叉的操作，在双点交叉（Two-point Crossover）中，需要随机选择两个交叉点的位置，然后将两个父代个体的这两个交叉点之间的基因序列进行互换，从而产生新的子代个体。

下面以一个长度为 6 的染色体为例，说明双点交叉的具体操作过程：假设有两个父代个体分别是：

$$S_1 = [0,1,0,1,0,1]$$

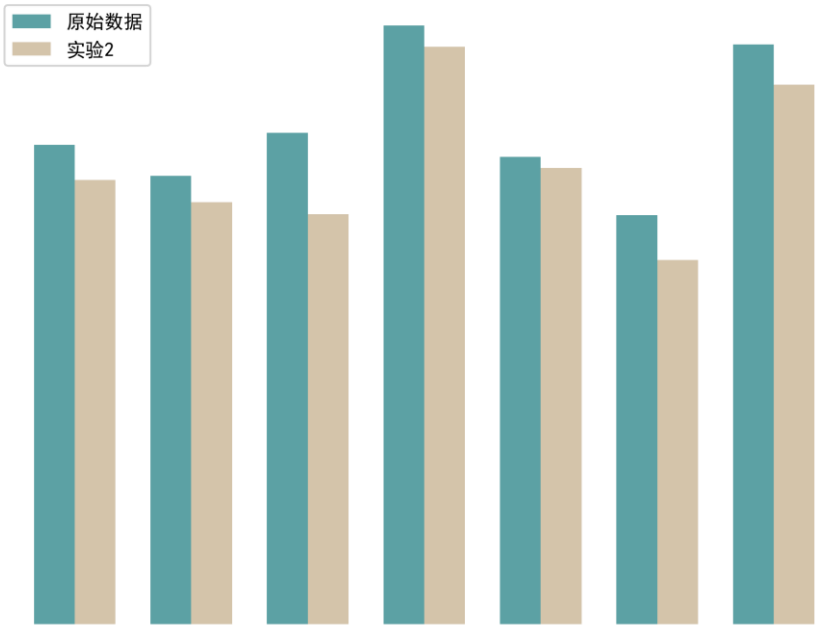
$$S_2 = [1,0,1,0,1,0]$$

本论文随机选择两个交叉点位置 k1=2 和 k2=5，然后将两个父代个体在这两个位置之间的基因信息进行互换，得到新的子代个体 S_3 和 S_4：

$$S_3 = [0,0,1,0,1,1]$$

$$S_4 = [1,1,0,1,0,0]$$

子代个体中来自父代个体的基因信息（变量）在交叉点之前或之后是完全一致的，但在交叉点之间则发生了互换。



图：实验二实验结果

在图中，绿色是没有本地执行所对应的函数值，黄色柱状图是进行交叉操作以后所对应的变量值，可以看出，虽然比实验一效果要好，但是这种交叉这种更新策略效果并不明显，且这几次实验之间的差值比较大，更加说明了变量之间有相关性。

4.3 基于贪心法调整的离散侏儒猫鼬算法

本节提出了修改后的基于贪心法调整的离散侏儒猫鼬算法。

4.3.1 基于贪心法的调整策略

为了克服变量之间存在的相关性，基因算法有着解码和编码两个过程，来配合交叉操作，使得交叉操作更具有随机性，克服了变量之间的相关性。

本论文不使用基因算法，而是想将侏儒猫鼬算法改造为能够求解离散形式问题的原因是，基因算法的解码和编码两个过程十分浪费时间，所以，为了加快侏儒猫鼬算法的运行时间，本论文在此使用基于贪心法的调整策略。

本论文采用的贪心法是处理时间最短法（Shortest Processing Time，SPT），该算法是将作业调整到当前剩余处理时间最短的机器上。SPT 算法的执行步骤如下：首先，将所有待处理的任务按照所需处理时间从小到大排序。其次，依次将任务分配给选择当前剩余处理时间最短的机器。直到所有任务都被分配并完成。

下面是 SPT 的一个实例：假设有 6 个任务需要调度，它们的处理时间分别为 3，4，5，4，3 和 7，有两台机器可用。首先，按照所需处理时间从小到大对任务进行排序：time=[3, 3, 4, 4, 5, 7]。

接下来，依次将任务分配给可用的处理器，使得处理时间最短的任务尽量早地被处理。第一批任务的处理时间分别为 3、3，将这两个任务分配给两台处理器中处理时间较短的那台。此时，两台处理器的处理时间分别为 3 和 3。现在剩余的任务列表为 [4, 4, 5, 7]。

接着，在两台处理器中选择处理时间最短的处理器（即第一个处理器），并为其分配下一个处理时间最短的任务，即处理时间为 4 的任务。第一个处理器的处理时间增加到了 7。现在剩余的任务列表为 [4, 5, 7]。

然后，选择处理时间最短的处理器（即第二个处理器），并为其分配下一个处理时间最短的任务，即处理时间为 4 的任务。第二个处理器的处理时间增加到了 7。现在剩余的任务列表为 [5, 7]。

继续选择处理时间最短的处理器（即第一个处理器），并为其分配下一个处理时间最短的任务，即处理时间为 5 的任务。第一个处理器的处理时间增加到了 12。现在剩余的任务列表为 [7]。

最后，选择处理时间最短的处理器（即第二个处理器），并为其分配最后一个任务，即处理时间为 7 的任务。第二个处理器的处理时间增加到了 14，此时所有任务都被分配并完成。

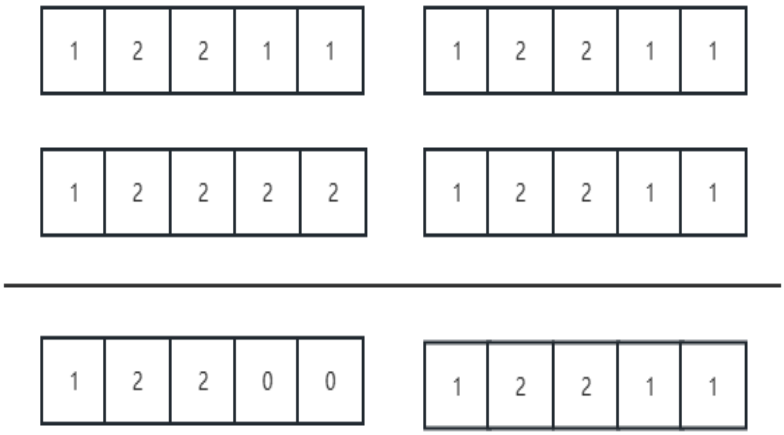
因此，使用 SPT 算法进行调度后，两台处理器的处理时间分别为 12 和 14。

4.3.2 重定义运算规则

为了更好的用数学语言描述每个解更新的过程，以及适应离散形式的更新形式，重新定义运算规则如下：

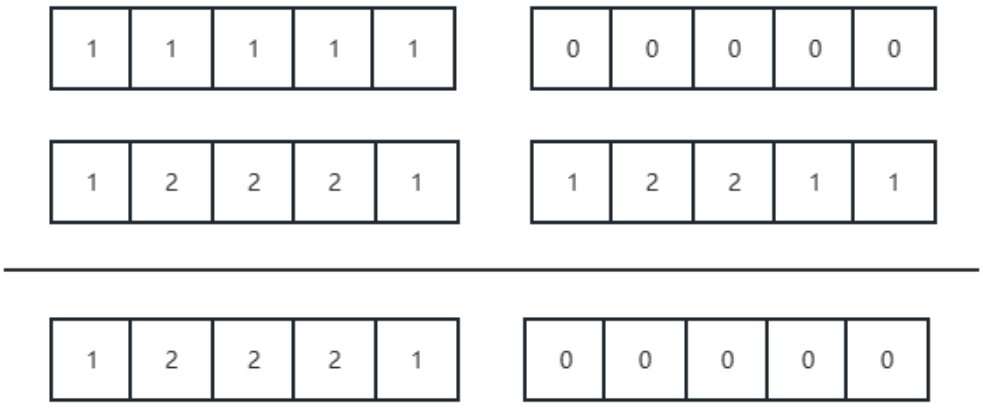
双目减法运算符($\overline{\mathrm{O}}$)，解中每一个变量相应元素的值是否不同。如果是，则获得其值，如果不是，则置为 0。也就是说，获得两个解中相同的部分，不同的部分置为 0。

$$X_1 \overline{\mathrm{O}} X_2 = \begin{cases} X_{1i} & X_{1i} = X_{2i} \\ 0 & X_{1i} != X_{2i}, i = 1,2...n \end{cases}$$



图：减法运算符号

双目乘法运算法($\overset{\times}{O}$)，每个变量的取值分别相乘，前一个变量的取值是 n 重伯努利分布，其值只有 0 和 1。



图：乘法运算符号

单目运算符$\odot X_1$，对X_1中所有为 0 的元素进行 SPT 调整，调整步骤如下：

一共两台机器，编号分别为 1 和 2，$time=[3,4,5,4,3,7]$，$X_1=[0,1,2,2,0,0]$，对值为 0 的元素进行 SPT 调整，值为 k（$k\neq 0$）的元素的意思：该任务的调整到第 k 台机器上。调整步骤如下：

（1）首先，将两个数组打包，得到$\begin{matrix}[3,4,5,4,3,7]\\ [0,1,2,2,0,0]\end{matrix}$

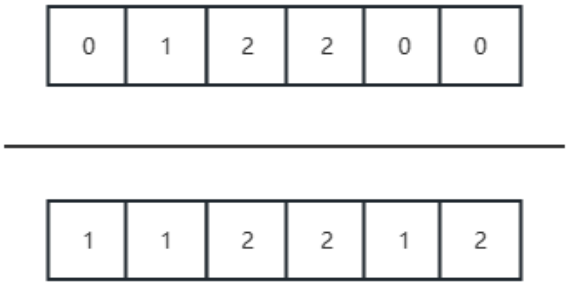
（2）将任务按照所需处理时间从小到大进行排序，得到$\begin{matrix}[3,3,4,4,5,7]\\ [0,0,1,2,2,0]\end{matrix}$。

（3）1 号机器的处理时间为 4，2 号机器的处理时间为 9，将所需处理时间最短的且对应分配机器值为 0 的任务，即任务处理时间为 3 的任务调度到 1 号机器上，1 号机器处理时间变成 7。

（4）1 号机器的处理时间为 7，2 号机器的处理时间为 9，将所需处理时间最短的且对应分配机器值为 0 任务，即任务处理时间为 3 的任务调度到 1 号机器上，1 号机器处理时间变成 10。

（5）1 号机器的处理时间为 10，2 号机器的处理时间为 9，将所需处理时间最短的且对应分配机器值为 0 任务，即任务处理时间为 7 的任务调度到 2 号机器上，1 号机器处理时间变成 16。

（6）将分配的方案映射回到没有排序的数组中得到$[1,1,2,2,1,2]$。即$\odot X_1=[1,1,2,2,1,2]$。



图：单目运算符

离散侏儒猫鼬算法伪代码

Algorithm 1: Genetic Algorithm

输入: 种群数量 n, 目标函数 $f(x)$, 最大迭代次数max_iter, 变量的维度 ndim

输出: x_{best}

初始化种群数量 n
初始化目标函数 $f(x)$
初始化种群 X
初始化最大迭代次数max_iter
初始化保姆的数量 bs
n=n-bs
设置保姆交换参数 L
for *iter=1:max_iter* **do**
    计算种群的适应度
    设置计时器 C
    通过下列公式, 找到 alpha 群体中的雌性首领 X_α

$$\alpha = \frac{fit_i}{\sum_{i=1}^{n} fit_i}$$

    对种群中的每一个个体, 通过下列公式, 计算候选的觅食位置

$$X_i^{iter+1} = \mathrm{RVS} \overset{\times}{\mathrm{O}} X_i^{iter} \bar{\mathrm{O}} X_\alpha$$

    其中, $\mathrm{RVS} = \prod_{i=1}^{ndim} Bernoulli(a_i; p)$
    评估每个新值 X_i^{iter+1}
    用下列公式计算睡眠丘陵

$$sm_i = \frac{fit_i^{iter+1} - fit_i^{iter}}{\max\{|fit_i^{iter+1}, fit_i^{iter}|\}}$$

    用下列公式计算睡眠丘的平均值

$$\varphi = \frac{\sum_{i=1}^{n} sm_i}{n}$$

    用下列公式计算移动向量

$$\vec{M} = \sum_{i=1}^{n} \frac{X_i \times sm_i}{X_i}$$

    如果 $C \geq L$, 交换保姆并重新初始化
    用如下公式进行更新

$$\boldsymbol{X}_{i+1} = \begin{cases} X_i - CF * \text{ rand } * \left[X_i - \vec{M}\right] & \text{if } \varphi_{i+1} > \varphi_i \\ X_i + CF * \text{ rand } * \left[X_i - \vec{M}\right] & \text{else} \end{cases}$$

    更新 X_{bset}
end
return X_{best}

图：基于贪心法调整的离散侏儒猫鼬算法流程图

4.4 仿真实验

4.4.1 实验环境及评价标准

名称	型号或版本
处理器	Intel(R) Core(TM) i7-7700HQ
显卡	NVIDIA GeForce RTX 1050
RAM	8.0GB
操作系统	Windows10 专业版
Python	3.8
Numpy	1.19
Matplotlib	3.3

4.4.2 评价标准

（1）种群的多样性：种群多样性无疑是启发式算法性能的一个关键问题，主要有两种方法能够衡量种群的多样性。

第一类方法是基于距离的度量方法（distance-based measurement，DMB），通过计算个体之间的欧氏距离来衡量种群间的多样性。DBMs 有几种不同的形式，如个体间的平均空间距离、个体与种群中心位置之间的距离的平均值、个体间最

大距离等[16,17]。但是相同的是，距离越远代表着种群多样性越丰富，反之，距离越小则代表着的种群多样性越差。

第二种方法是基于频率的测量方法（frequency-based measurement，FBM)。在 FBM 中，通过统计具有代表性、有优势的个体的频率[18]或者平均基因频率[19]来评价种群多样性。此外，熵度量是属于 FBM 的另一种方法，因为熵（公式 \ref{shang}）使用了基因出现的概率/适应度。

$$S = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

其中，(S) 表示熵，(n) 表示系统中可能的状态数，(p_i) 表示第 (i) 个状态发生的概率。

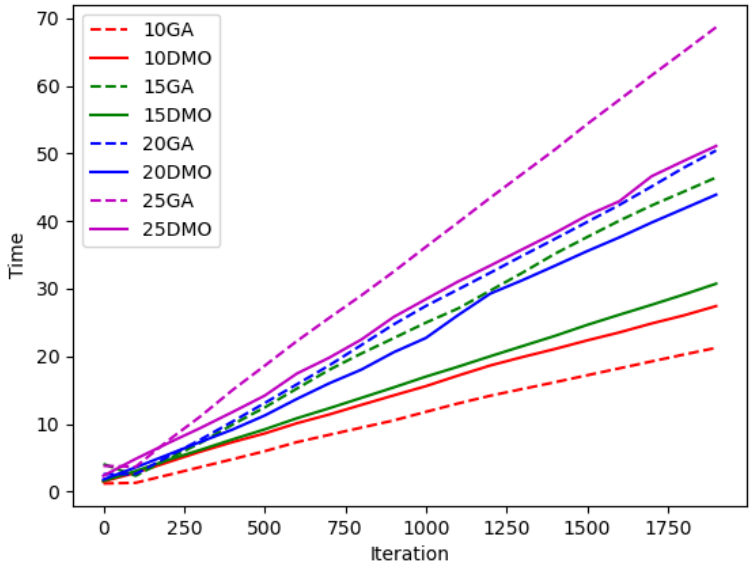
在本文中，本论文使用了一种度量方法，即将种群中每个解的各个维度的方差之和的二分之一次方作为多样性 D 的衡量标准。下面是计算多样性 D 的公式：

$$D = \sqrt{\sum_{j=1}^{m} D_j^2},\ D_j = \frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \bar{x}_j\right)^2$$

其中，m 是解的维度，n 是种群的数量，x_{ij} 是是第 i 个粒子的第 j 个维数。

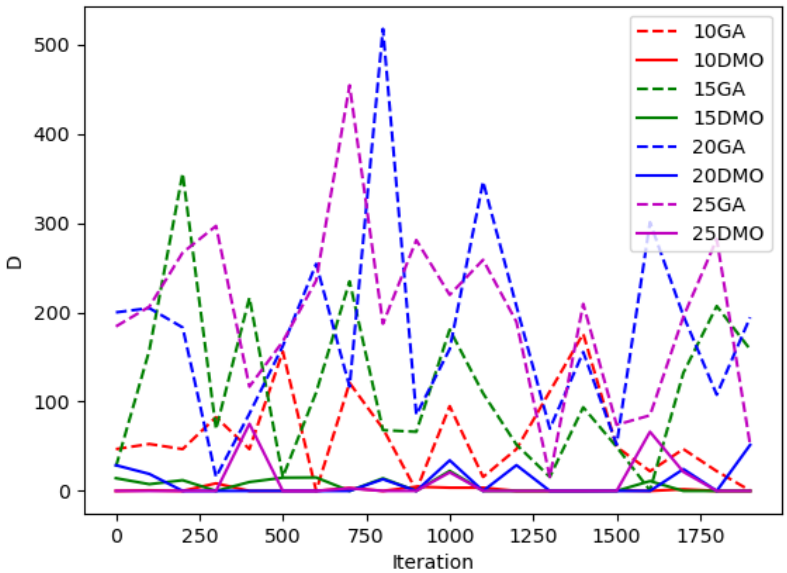
（2）其他评价标准（例如 FC，P）和第三章一样。

4.5 实验结果及分析



图：

如图所示，虚线是基因算法求解该问题的时间，实线是侏儒猫鼬算法的运行时间，标签前面的数字代表着车辆的数目，同样颜色的线线代表着同样的车辆的数量。可以看出，侏儒猫鼬算法的运行时间明显比基因算法的运行时间要短，且当车辆数目为 15 和 20 的时候，运行时间大约为原算法的三分之二，当车辆数目为 10 和 25 时，运行时间能节省约四分之一。



图：

可以看出，基因算法的种群差异度变化幅度很大，这就说明，每次算法的运行过程中还在产生新的解，且种群差异度很大，这可以充分保证侏儒猫鼬算法不会跌入到很差解中且能够产生更优的新解。

4.6 本章小结

本章在上一章的基础上，探索了新的方法来解决第三章提出的问题。本论文解决了基因算法中编码解码过程耗时较长的问题，并通过对侏儒猫鼬算法的改进，将其转变为离散侏儒猫鼬算法。实验结果表明，与基因算法相比，本论文的算法在运行时间上有显著的缩短。

第5章 总结与展望

5.1 论文的主要工作及贡献

随着国产新能源汽车公司的崛起，电动汽车越来越受到人们的追捧，其中很重要的一个原因就是可玩的智能系统。虽然电动汽车的电池续航里程越来越大，但是对比人们对于里程的渴望而言是远远不够的，如何在有限的容量中让汽车的行驶里程变得更大，仍然需要解决，而任务卸载是其中一个有希望能够解决该问题的方法，将任务卸载到边缘端或者是其他的汽车上，通过合理的调度来降低所有车辆的能耗和。这就需要本论文来设计合理的算法来进行调度，但是该问题的模型复杂，且并不能通过凸优化的方法来解决，因此需要其他的方法来解决。智能算法是作为求解复杂问题的佼佼者，本论文通过智能算法来求解该问题。本文采用基因算法以及基于贪心法调整的侏儒猫鼬算法来解决本文的复杂问题。本文的总结如下：

（1）研究了车联网中任务卸载的问题模型，将能耗模型分为两部分来描述，将其描述为数学规划的形式。

（2）本文使用了基因算法来求解该问题，但是从本质上来讲，并没有对基因算法进行创新。

（3）本文在侏儒猫鼬算法的基础上提出了基于贪心法调整的侏儒猫鼬算法，核心思想是通过贪心法（处理时间最短法）来实现对种群进行更新。

5.2 未来的工作

针对车联网中能耗优先的任务调度问题，本文考虑的目标是最小化能耗的平方之和，这能有效兼顾分配方案的公平性。但是该场景下依旧有许多问题需要解决，未来仍需要进行探索。

（1）本文第三章使用了基因算法来解决该问题，但是基因算法存在编码解码过程导致计算速率偏慢。因此，下一步的研究可以考虑加入一些其他的近似算法来加快这个过程。

（2）本文第四章使用了离散侏儒猫鼬算法来解决该建模问题，虽然离散侏儒猫鼬算法能够比基因算法更快速的解决问题，但是其算法的不确定性也比基因算法更大，没有一个比较稳定的迭代次数。因此，下一步的研究可以考虑后期动态的不确定性，迭代前期加大不确定性，迭代后期减小不确定性，进一步提高问题的解决效率。

参考文献

[1] 余文科程媛, WENKE YU Y C. 物联网技术发展分析与建议[J/OL]. 物联网学报, 2020, 4(4): 105-109. DOI:10.11959/j.issn.2096-3750.2020.00195.

[2] 施巍松, 张星洲, 王一帆, 等. 边缘计算：现状与展望[J/OL]. 计算机研究与发展, 2019, 56(1): 69-89. DOI:10.7544/issn1000-1239.2019.20180760.

[3] CANO Z P, BANHAM D, YE S, 等. Batteries and fuel cells for emerging electric vehicle markets[J]. Nature Energy, 2018, 3(4): 279-289.

[4] HE Y, SI P B, SUN E C, 等. Joint Task Management in Connected Vehicle Networks by Software-Defined Networking, Computing and Caching[C/OL]//2017 International Conference on Network and Information Systems for Computers (ICNISC). 2017: 47-50[2024-02-02]. https://ieeexplore.ieee.org/abstract/document/8842717. DOI:10.1109/ICNISC.2017.00018.

[5] LIU X Y, DING Z, BORST S, et al. Deep Reinforcement Learning for Intelligent Transportation Systems[EB/OL]. (2018-12-03)[2024-02-02]. https://arxiv.org/abs/1812.00979v1.

[6] LIU N, LI Z, XU J, 等. A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning[C/OL]//2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). 2017: 372-382[2024-02-02]. https://ieeexplore.ieee.org/abstract/document/7979983. DOI:10.1109/ICDCS.2017.123.

[7] ZHANG Y, YAO J, GUAN H. Intelligent Cloud Resource Management with Deep Reinforcement Learning[J/OL]. IEEE Cloud Computing, 2017, 4(6): 60-69. DOI:10.1109/MCC.2018.1081063.

[8] CHENG M, LI J, NAZARIAN S. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers[C/OL]//2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). 2018: 129-134[2024-02-02]. https://ieeexplore.ieee.org/abstract/document/8297294. DOI:10.1109/ASPDAC.2018.8297294.

[9] HE Q, CUI G, ZHANG X, 等. A Game-Theoretical Approach for User Allocation in Edge Computing Environment[J/OL]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(3): 515-529. DOI:10.1109/TPDS.2019.2938944.

[10] Distributed Multiuser Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach | IEEE Journals & Magazine | IEEE Xplore[EB/OL]. [2024-02-02]. https://ieeexplore.ieee.org/abstract/document/8012473.

[11] GAD A G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review[J/OL]. Archives of Computational Methods in Engineering, 2022, 29(5): 2531-2561. DOI:10.1007/s11831-021-09694-4.

[12] BELOGAEV A, ELOKHIN A, KRASILOV A, 等. Cost-Effective V2X Task Offloading in MEC-Assisted Intelligent Transportation Systems[J/OL]. IEEE Access, 2020, 8: 169010-169023. DOI:10.1109/ACCESS.2020.3023263.

[13] CHEN M H, DONG M, LIANG B. Resource Sharing of a Computing Access Point for Multi-User Mobile Cloud Offloading with Delay Constraints[J/OL]. IEEE Transactions on Mobile Computing, 2018, 17(12): 2868-2881. DOI:10.1109/TMC.2018.2815533.

[14] HUYNH L N T, PHAM Q V, PHAM X Q, et al. Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach[J/OL]. Applied Sciences, 2020, 10(1): 203. DOI:10.3390/app10010203.

[15] BUI K H N, JUNG J J. ACO-Based Dynamic Decision Making for Connected Vehicles in IoT System[J/OL]. IEEE Transactions on Industrial Informatics, 2019, 15(10): 5648-5655. DOI:10.1109/TII.2019.2906886.

[16] URSEM R K. Diversity-Guided Evolutionary Algorithms[C/OL]//GUERVÓS J J M, ADAMIDIS P, BEYER H G, et al. Parallel Problem Solving from Nature — PPSN VII. Berlin, Heidelberg: Springer, 2002: 462-471. DOI:10.1007/3-540-45712-7_45.

[17] MORRISON R W, DE JONG K A. Measurement of Population Diversity[C/OL]//COLLET P, FONLUPT C, HAO J K, et al. Artificial Evolution. Berlin, Heidelberg: Springer, 2002: 31-41. DOI:10.1007/3-540-46033-0_3.

[18] GOUVEA M M, ARAUJO A F R. Diversity control based on population heterozygosity dynamics[C/OL]//2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). 2008: 3671-3678[2024-02-04]. https://ieeexplore.ieee.org/abstract/document/4631295. DOI:10.1109/CEC.2008.4631295.

[19] COLLINS R J, JEFFERSON D R. Selection in massively parallel genetic algorithms[M]. University of California (Los Angeles). Computer Science Department, 1991.

致谢

时光飞逝，转眼间即将结束硕士阶段的学习。在毕业论文完成之际，我想借此机会向所有帮助过我的人们表示最真挚的感谢。

首先，我要感谢我的导师，在硕士学习阶段，张老师给予我耐心、全面和细致的指导，让我在研究生生涯中获得了宝贵的学习机会和成长经历。在研究学习中，您一直以严谨的态度、卓越的专业知识和宽广的学术视野给予我悉心的指导和帮助。您的言传身教、精益求精的精神和不断探索创新的敬业精神都深深地影响和感染着我。在此，我要向您表达我最衷心的感谢和敬意。

其次，我要感谢我的同门们。在这段共同求知的旅程中，本论文相互鼓励、相互学习，共同成长。每一次的讨论和交流，都让我受益匪浅。感谢你们在我困难时给予的支持与帮助，本论文一起度过了难忘的时光。

最后，我要由衷地感谢我的家长。是你们一直以来的支持和鼓励，让我有信心、有勇气去追求自己的梦想。你们对我的无私奉献和默默付出都让我感到无比的幸福和感激。没有你们的支持与理解，我无法顺利完成这段学业之旅。

愿未来能够永怀初心，砥砺前行。