南 开 大 学

本 科 生 毕 业 论 文（设 计）

中文题目：基于移动边缘计算的计算卸载启发式算法研究

外文题目：Research on Computational Offload Heuristic Algorithm Based on Mobile Edge Computing

学　　号：1811361

姓　　名：郭宇

年　　级：2018 级

学　　院：网络空间安全学院

系　　别：物联网工程

专　　业：物联网工程

完成日期：2022 年 4 月

指导教师：吴英 副教授

关于南开大学本科生毕业论文（设计）的声明

本人郑重声明：所呈交的学位论文，是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或没有公开发表的作品内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：郭宇

2022 年 4 月 29 日

本人声明：该学位论文是本人指导学生完成的研究成果，已经审阅过论文的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

学位论文指导教师签名：吴英

2022 年 4 月 29 日

摘 要

随着物联网的快速发展和 5G 技术的迅速推广，移动边缘计算(Mobile Edge Computing，MEC)逐渐成为研究热点。MEC 是一种基于移动通信网络的全新分布式计算方式，可以向用户提供超低时延和高带宽的网络服务。在 MEC 中，计算卸载是指完全或部分地将设备的计算任务卸载到服务器，是移动边缘计算的关键问题之一。目前，计算卸载的性能通常以时间延迟和能量消耗的综合指标作为衡量指标。论文将目光聚焦于用户-边缘云-中心云三层模型的计算卸载决策问题，提出一种名为定向选择性遗传算法(Targeted Selective Genetic Algorithm，TSGA)的启发式算法用以解决上述问题。通过仿真实验的结果对比可知，该方案相较于基准算法降低了最高 42%的系统消耗代价，在只考虑时延和只考虑能耗的情况下也能够降低 21%和 19%系统代价。因此针对计算卸载决策问题，基于 TSGA 的卸载方案对于系统代价有一定的降低效果。

关键词：移动边缘计算；计算卸载；定向选择性遗传算法；最小系统消耗

Abstract

With the rapid development of the Internet of things and the rapid promotion of 5g technology, mobile edge computing (MEC) has gradually become a research hotspot. MEC is a new distributed computing method based on mobile communication network, which can provide users with network services with ultra-low delay and high bandwidth. In MEC, computing offload refers to completely or partially unloading the computing tasks of the device to the server. It is one of the key problems of mobile edge computing. At present, the performance of computing offload is usually measured by the comprehensive index of time delay and energy consumption. This paper focuses on the computing offload decision problem of user edge cloud center cloud three-tier model, and proposes a heuristic algorithm called targeted selective genetic algorithm (TSGA) to solve the above problems. Through the comparison of simulation results, it can be seen that this scheme reduces the system consumption cost by up to 42% compared with the benchmark algorithm, and can also reduce the system cost by 21% and 19% when only considering delay and energy consumption. Therefore, for the decision-making problem of computing unloading, the unloading scheme based on TSGA has a certain effect on reducing the system cost.

Key Words: mobile edge computing (MEC) ; computing offload; targeted selective genetic algorithm (TSGA) ; minimum system consumption

目 录

第一章 绪论

第一节 研究背景

近几年来，互联网技术，大数据，物联网等不断发展，日新月异[1]。5G 技术的产生和发展也促生了许多新的技术应用场景，例如虚拟现实（Virtual Real，VR）、自动驾驶等，在这些场景中，用户设备会产生出大量的数据和计算任务，这些计算任务大多数都是高计算量低时延需求的工作[2]。对于这些任务，传统的云计算方式是将任务上传到云服务器进行计算，计算完成后再将结果返回给原设备。这样做虽然能解决计算量大和本地处理速度慢的问题，但由于设备距离云服务器较远，传输过程中，难免需要考虑传输速率、网络扰动、能量消耗、数据安全等问题[3]，尤其是像 VR 这种需要实时交互的计算任务，时延过大会导致用户的体验及其恶劣。对于这种情况，移动边缘计算（Mobile Edge Computing，MEC）的提出[4]无疑为解决此类问题给出了一个明确的方向。

MEC 系统中将有计算和缓存资源能力的服务器部署到距离用户较近的网络边缘作为边缘云服务器，用户可以将对延迟要求高或计算量庞大的任务卸载到边缘云进行计算。MEC 不仅减轻了用户设备的计算压力，降低了用户与云中心服务器的交互频率，还可以明显减少信息交换时的等待时间[5-6]。但 MEC 中边缘服务器相对于中心云服务器来说毕竟属于轻量化服务器，具有有限的计算资源和计算能力，而用户的数量相对服务器数量而言是较多的，如何将用户所产生的任务合理的进行资源分配并进行计算卸载就成了边缘计算中研究的重点和难点[7]。

第二节 国内外研究现状

现有研究对于计算卸载决策目标主要集中在优化能耗和时间成本上[8]。文献[9]中使用马尔可夫决策过程的方法对于任务的卸载策略进行分析，提出了一种高效的一维搜索算法，算法根据每个任务的平均延迟和移动设备的平均功耗寻找任务调度最优策略。文献[10]提出了一种面向多用户的联合计算卸载和资源分配策略，其中，基于贪心算法的计算卸载算法获得最佳卸载决策，而给定策略的计算资源最佳分配则使用拉格朗日乘子法获得，相较于基准算法，其实验结果可以

大幅降低系统成本。

文献[11]中考虑单个移动设备进行计算卸载的情况，目标为联合计算卸载和移动设备的CPU频率来最小化任务时延和设备能耗并证明了其目标为NP问题，其针对固定CPU频率和弹性CPU频率，分别提出了两种基于半正定松弛（Semidefinite Relaxation，SDR）的方法来进行优化，使得在能耗和时延方面都有明显改进效果。文献[12]中则致力于在满足在应用程序给定的执行时间内完成任务同时节省能耗，其基于Lyapunov优化提出了一种动态卸载算法（Dynamic Offloading Algorithm，DOA），其算法结果比现有算法更加节能。

综上所述，现有研究大多是以任务完成时延和系统能耗的综合指标作为评判标准，将计算卸载问题建模成为相关的数学问题进行处理，并提出相关的启发式算法进行处理，从而降低系统消耗并找到更好的计算卸载决策。

第三节 论文工作及结构

论文将优化计算卸载策略问题建模为优化任务时延和设备能耗的综合模型，从而计算出最小系统代价。基于此模型，对经典遗传算法进行改进，设计出定向选择性遗传算法，根据此算法寻找最小系统代价并获得最佳卸载策略。同时进行了仿真实验，将论文提出的算法与其他基准算法进行了比对，实验结果证明了定向选择性遗传算法在计算最小系统消耗和计算卸载决策方面相较于其他基准算法具有一定优势。

论文结构组织安排如下：

第一章，主要介绍了研究背景和国内外研究现状，阐述了相关问题的研究价值，并介绍了相关问题已有的研究成果，说明了论文的主要工作。

第二章，主要介绍了系统场景模型的搭建和一些基准算法。通过分析基准算法的优缺点介绍了其对应适用的工作场景。

第三章，详细介绍了定向选择性遗传算法。阐述了定向选择性遗传算法工作流程，并细致的规定了评价标准的计算方式。

第四章，通过仿真实验进行结果比对。介绍了实验数据设置，通过对基准算法和论文提出算法输出的结果进行对比，分析出论文算法的优势。

第五章，总结了论文工作和不足，展望未来工作。

第二章　系统场景和相关算法概述

第一节 系统场景

论文中使用的场景为最经典的三层计算卸载整体布局：第一层为能产生任务的物联网设备终端；第二层为边缘云服务器及其基站，其中认为边缘服务器部署在离基站不远的地方，通过有线方式连接起来，基站负责接收数据和传输数据，其与边缘服务器发生数据传输时认为速度极快，其时延不计入总体时延计算；第三层为中心云服务器及其基站，中心云服务器计算能力较强，其与其基站设置与边缘云相同。三层结构之间进行任务传输采用无线传输的方式。系统场景配置如图 2.1 所示。设备终端上产生的任务可以选择在本地执行，卸载到边缘云执行或者卸载到中心云执行。

论文场景中认为本地设备产生了一组任务，这组任务中包含若干个小任务，对于这些小任务每个都可以采取不同的卸载策略，但其无法再被继续分割为更小的任务，具体示意图如图 2.2 所示，其中策略组中的 L 代表本地计算，E 代表卸载到边缘云，C 代表卸载到中心云。任务选择卸载到中心云执行必须要先将任务上传到边缘云，再通过边缘云基站上传到中心云，从而模拟本地设备距离中心云服务器较远的情况。边缘云服务器和中心云服务器的计算能力不是无穷大的，其能同时计算的任务数量有限。

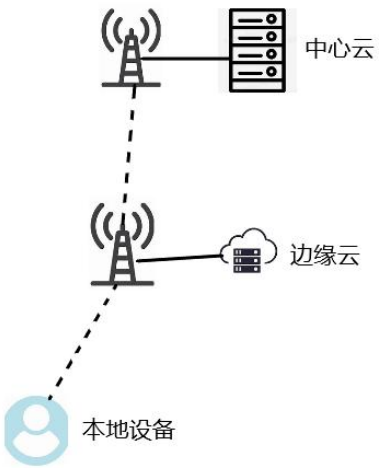

图 2.1 本地设备、边缘云、中心云的三层结构

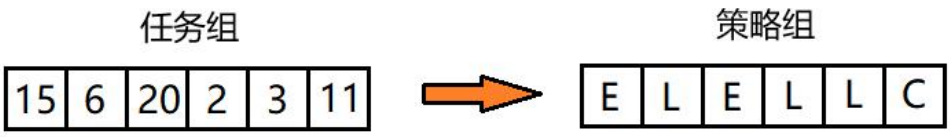

图 2.2 任务组对应卸载策略组

第二节 全部本地计算

全部本地计算是一种最基础的任务处理方式。当任务在本地设备产生后，无论其是否可以被卸载到其他服务器，都选择在本地进行计算。这种处理方式的好处是不用为其他服务器增加负担，且不需要进行数据传输，无需担心在数据传输途中数据丢失或被他人窃听等安全问题。但随着网络技术不断发展，这种处理方式的缺点很快就暴露出来。随着产生的任务计算量愈发增大，本地设备的计算能力的增长速度已经不再跟得上任务计算量的增长速度，这会导致当所有任务仍然在本地计算时，本地计算能力不足，处理一个计算量较大的任务时延较长，而其它任务需要等当前任务完成才能被计算从而不得不处于等待状态，形成恶性循环，最后导致一组任务完成的整体时延过长。且由于任务计算量大，本地处理器以全工作能耗进行工作，而完成任务的时延又较长，导致处理一组任务本地的能量消耗也较大。

在 5G 通信模式下，产生的任务数据量都较大，全部本地计算可能不再是一个好的卸载策略，但其仍有可用的应用场景。举例来说，在小型传感器网络中，每个部署下去的传感器就可以认为是一种本地设备，这些设备根据环境变化产生了不同的响应，对于这些响应需要将其转换为数据，这种任务所需的计算量和能耗都是很小的，可以选择全部在本地设备进行计算，传感器将收集到的变化转化为对应的数据，然后将其存储起来或上传至数据收集端。

第三节 随机卸载策略

随机卸载策略[13]，顾名思义对于本地产生的任务，无论其任务计算量大小，随机选择其处理方式，即对于一个任务，其选择的策略在以下三种策略中随机选

择一个：本地计算，卸载到边缘云服务器计算，卸载到中心云服务器计算。其优点在于引进了边缘云计算和中心云计算，对于计算量稍大的任务来说有机会卸载到计算能力更强的云服务器上进行计算，相对于本地计算减少了任务时延和能量消耗。其缺点为：不稳定性太高。由于任务都是实时生成的，对这些任务采取随机策略，很有可能发生策略不适合任务的情况，例如：产生了一个计算量较大的任务，而随机策略为其分配了全部本地计算的策略，那么对于一组任务整体来说这无疑增加了整体时延和设备能耗。又或者产生了一个计算量很小的任务，而随机策略为其分配了卸载到中心云服务器计算，那么这就会浪费中心服务器的计算资源，还有可能发生传输数据的时间大于在本地计算时间的情况，增加整体任务的延时。

虽然随机策略卸载具有不稳定性，但有的场景也可以使用这种方法进行任务卸载的策略决定。例如当本地设备产生的任务大小较为固定，其所需计算量也较为稳定时，本地设备又具有一定的计算能力，设备之间传输数据速度较快。那么这时采取随机策略计算可以相对的减轻全部本地计算给本地设备带来的负担，且对于本地设备来说云服务器具有更高的计算速度，进行计算卸载可以减少整体的任务时延，同时将部分任务卸载到服务器还可以减少本地设备的能耗，使得本地设备减少电量消耗。

第四节 原始遗传算法

遗传算法（Genetic Algorithm，GA）是一个不断发展，不断进步的搜索启发式算法。遗传算法是模仿达尔文进化理论提出的一种随机全局搜索优化算法，其起源于计算机对生物系统进行的模拟研究。1975 年，John H. Holland 教授在其专著《Adaptation in Natural and Artificial Systems》中提出了模式定理，作为遗传算法的基本定理[14]。这本书被认为是第一本系统论述了遗传算法的专著，而 1975 年也被认为是遗传算法的诞生元年。随着后续不断对于遗传算法的研究，遗传算法不断被改进完善，并运用到越来越多的领域中。遗传算法的优点在于：具有快速且随机的搜索能力，无论问题规模大小；根据概率采用随机机制执行迭代进程，使得算法具有一定的随机性；遗传算法本身较为简单，具有丰富的修改空间，其改进方法可以采用与其他算法或技术相结合的方式。但遗传算法的缺点也是限制

其使用场景的关键问题：遗传算法中初始种群的选择的好坏会一定程度影响最后的输出结果；对于某些常用参数的设置，经常是依靠以往经验和不断尝试进行的，而这些参数的设置直接关系到最后输出解的品质；由于遗传算法中的很多步骤是随机过程，最后要得到精确度较高的优秀解需要的时间较长且容易过早的输出局部最优解而忽略了全局最优解的存在[15]。

运用遗传算法的场景有很多，一般来说，传统穷举法很难求出最优解的问题都可以使用遗传算法来求出一个最优解或接近最优解的结果。例如对于求解背包问题，图形划分问题，旅行商问题[16]等，遗传算法能输出一个很好的结果；在人工智能领域方面，机器人轨迹生成的问题也可以使用遗传算法来生成一个对于机器人来说较为安全的路径[17]；遗传算法还可以被用于训练神经网络，遗传算法的思想也可以被运用到机器学习中，例如基于遗传学的机器学习模型。

第三章　定向选择性遗传算法

第一节 定向选择性遗传算法

根据系统场景和系统消耗模型，论文所要完成的目标为：对于本地设备所产生的一组任务，规划出一种对于这组任务中每个小任务对应的卸载策略，使得这组任务的系统消耗最小。此目标问题被证明为 NP-Hard 问题[18]，直接求解得到最小的系统消耗较难。而使用启发式算法对其进行求解，可以在有限的时间内输出一个较好的结果，其搜索方式也比普通算法的搜索方式更高效。论文提出一种定向选择性遗传算法（Targeted Selective Genetic Algorithm，TSGA），基于遗传算法的基础对其进行改进，用来解决此模型下的计算卸载策略问题。

定向选择性遗传算法的主要步骤包括：定向选择、交叉、突变、自然选择等，针对论文中的系统场景，需要对算法中的参数编码、染色体生成规则、适应度函数、遗传操作控制等主要步骤进行特定性的设置，从而使得算法满足系统模型要求。

3.1.1 参数编码和初始染色体组生成

在论文提出的系统场景模型中，任务有三种卸载方案选择：在本地设备计算，卸载到边缘云服务器计算，卸载到中心云服务器计算。对其分别编号为 0、1、2，从而使用数字代表任务选择的计算卸载方式。根据一组任务所分成的小任务的数量，对应的使用数字代表其小任务的卸载方式，这些数字组成一个数组，作为这组任务的卸载策略。一个卸载策略即作为一条染色体参与到算法中，每条染色体上不同位置所承载的卸载方式作为染色体上的一个基因。

初始染色体组中包含随机生成的初始染色体，初始染色体的长度等同于一组任务所包含的小任务的数量，每一条染色体作为一条可能的卸载策略。初始染色体组有染色体最大数量限制，而生成初始染色体组时默认生成最多数量的染色体，以便于后面进行自然选择时将劣质的初始染色体淘汰替换成适应度更高的子代染色体，促使染色体组进化。

3.1.2 染色体生成规则

染色体及其所携带的基因需要符合一定的规则才可以认为这条染色体可以作为“正常”染色体参与后续过程，否则需要重新生成符合规则的染色体。在本模型中为了方便进行计算，设置边缘云服务器和中心云服务器只能同时处理固定数量的任务，而任务大小不进行限制；任务是在本地设备终端产生的，所以选择本地设备处理的任务数量不设限制。所以在一条染色体中，最多能同时存在固定数量的“1”基因，同理最多能存在固定数量的“2”基因，而染色体中的“0”基因则不受数量限制。另外，染色体长度必须和一组任务中包含的小任务的数量一致，从而保证每个基因和小任务能够一一对应，便于最后计算出对于这组任务采取的卸载策略所对应系统消耗值。

3.1.3 适应度函数

对于染色体组中的每条染色体都会有一个适应度，适应度越大代表着这条染色体在此环境下更占优势，这条染色体越能保留下来。适应度需要使用适应度函数进行计算，在论文所设置的系统场景中，取系统消耗的倒数作为适应度函数，这样处理能保证对于一条染色体所代表的卸载策略，这组任务采取此策略造成的系统消耗越小，其染色体适应度越大，此条染色体越能被保留下来。适应度函数具体计算方法如公式 3.1 所示：

$$\text{fitness} = \frac{1}{\alpha \times T_{total} + (1-\alpha) \times E_{total}} \tag{3.1}$$

3.1.4 定向性选择

对于后续过程，需要在染色体组中选择两条染色体作为父母染色体参与。传统的遗传算法是采用随机序号方式进行选取。在论文的模型下采用此种办法会发生抽取的父母染色体均为适应度较小的染色体的特殊情况，这样其生成的子代染色体的适应度相对较差，使得此次交叉过程结果无效，增加了无效迭代的次数，直接影响到最后的输出结果。

论文采用确定一条适应度最大的染色体与随机一条染色体作为父母染色体

的方法作为选择的结果。在染色体组中每一条染色体都有其对应的适应度，适应度越大的染色体则其对应的卸载策略越好，选择适应度最大的染色体作为父母染色体中的一条可以保证后续过程中生成的子代染色体可以携带有部分此优秀染色体的基因组，为生成优秀子代染色体奠定基础。另外一条染色体采用“轮盘赌”方式[19]进行选择：每条染色体被选中的概率为其适应度在适应度总和中所占比例（计算公式如下公式 3.2），随机一个 0 到 1 的小数作为随机选择百分值，其落到的区域即为所选择染色体。“轮盘赌”的选择方式秉持着“适应度越大，被选择概率越大”的准则，使得选择出的父母染色体大部分情况均为适应度较大的染色体，以利于更优子代染色体的生成。

$$P(i) = \frac{fitness(i)}{\sum_{i=0}^{n} fitness(i)} \tag{3.2}$$

3.1.5 选择性交叉

染色体交叉生成新一代染色体是遗传算法中一个重要的步骤。父母染色体根据交叉概率进入交叉过程从而生成子代染色体，随着不断进行迭代，适应度高的子代染色体使得染色体组整体不断进化，同时也会生成新的适应度更高的子代染色体。原始遗传算法中常使用单点交叉算子进行交叉过程[20]。父母染色体在某一点断开后重新组合，使得子染色体在断开点前一部分为父染色体的基因组，后一部分为母染色体的基因组，具体流程如图 3.1 所示。单点交叉算子在论文的系统环境中劣势在于：当存在染色体生成规则的情况下，父母染色体可能找不到一个好的断开点进行交叉重组，这样会使得这次迭代过程无法生成子代染色体，浪费一次迭代进化种群的机会。还有可能发生父母染色体中的优秀基因无法全部遗传到子代染色体中，使得生成的子代染色体适应度反而不如父母染色体的情况。

图 3.1 单点交叉算子

论文采用选择性交叉算子。对于父母染色体，其中的优秀基因直接遗传到子代染色体中，其余部分在保证符合染色体生成规则的前提下随机进行填充。当父母染色体上的优秀基因处于同一位置时选择其中任意一个进行直接遗传。具体流程如图 3.2 所示。这样进行交叉能够保证父母染色体中的优良基因能够遗传到下一代，而下一代染色体在作为父母染色体时又可以将新生成的优良基因和前一代的优良基因组遗传下去，以利于最好染色体的生成。同时在交叉过程中一定能生成一条子代染色体，保证此次迭代的有效性。

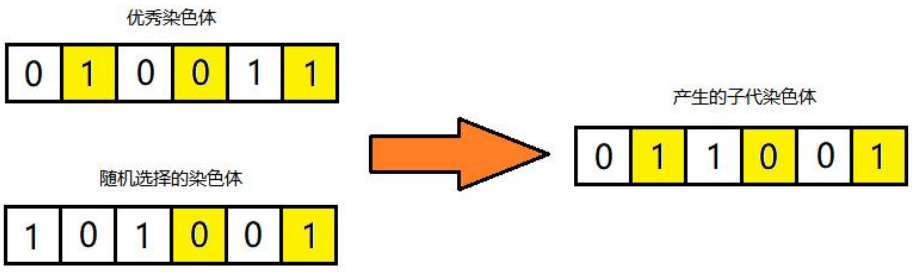

图 3.2 选择性交叉算子

优良基因的评价标准为：当任务组中数据量较大的任务选择了卸载到边缘计算即染色体对应位置基因为“1”时；当任务组中数据量中等的任务选了卸载到中心云计算即染色体对应位置基因为“2”时；当任务组中数据量较小的任务选择了本地计算即染色体对应位置基因为“0”时。以上情况则认为此基因为优秀基因。

3.1.6 染色体突变过程

交叉生成的子代染色体根据突变概率有机会发生突变过程。突变是指染色体上的某处基因发生改变，在本模型中，突变位置为随机位置，改变成为的基因也为 0、1、2 中的随机数，但保证不与原基因相同。突变后生成的染色体要保证其符合染色体生成规则，否则就重新进行突变过程直到生成的染色体符合规则。染色体突变过程参考下图 3.3。

突变过程的意义在于排除局部最优解对整体算法的影响。在进行多次迭代后，适应度最大的染色体会逐渐趋向于一个稳定的策略，也就是局部最优解，为

了寻找更好的全局最优解，突变过程可以给染色体组中带来新的基因，突破局部最优解带来的限制，有利于算法继续进化。

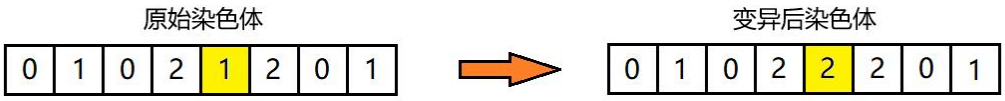

图 3.3 突变过程

3.1.7 自然选择过程和终止条件

经过交叉变异过程生成的子代染色体如果其适应度大于初代染色体组中的适应度最小染色体，则将初代染色体组中的适应度最小的染色体替换为适应度更高的子代染色体，以此模拟“优胜劣汰”的自然选择步骤，使得染色体组不断进化，变得更适应环境。

当算法进行到一定迭代次数或者得到的最大染色体适应度变化很小时，则认为已经找到了全局最优解，输出最大适应度的染色体所对应的卸载策略所造成的最小系统消耗，否则继续算法迭代。完整的算法流程图如图 3.4 所示。

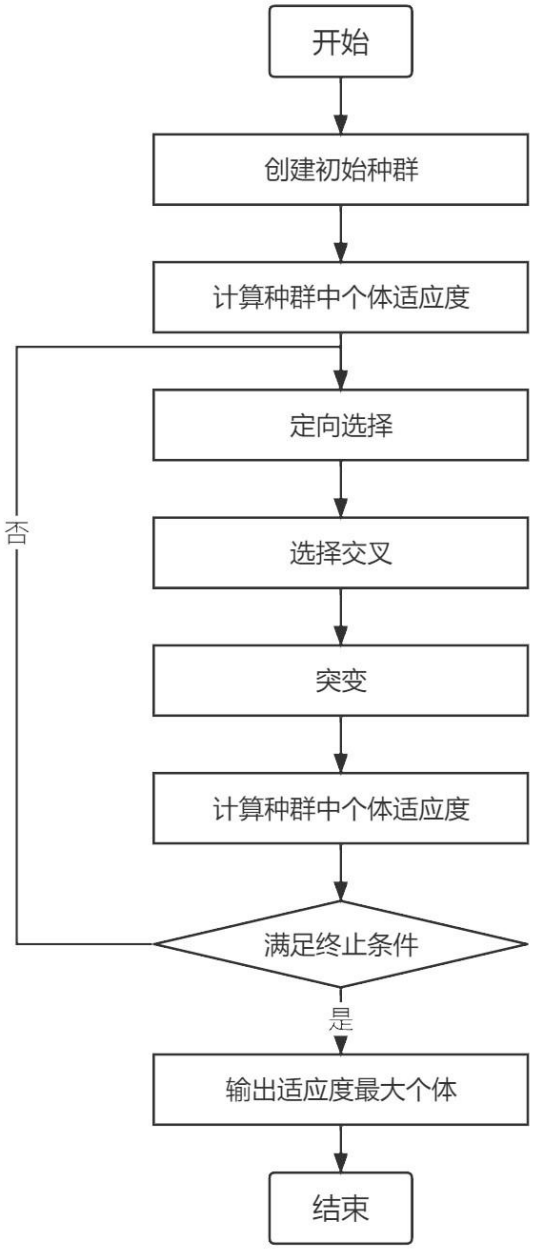

图 3.4 定向选择性遗传算法流程图

第二节 系统消耗参数

评价计算卸载策略的好坏，使用的评判标准为使用此策略造成的系统消耗多少。在论文提出的模型下，系统消耗主要由任务完成时延和系统能量消耗组成，其中任务时延认为是从任务出现到任务计算结果返回到本地设备终端的时间。由于任务时延无法直接和系统能耗耦合作为系统代价的衡量标准，为了计算方便，取两者的数值进行线性加权相加作为系统消耗，即系统消耗的计算公式如下公式

3.3：其中 T_{total} 是任务总时延，E_{total} 是系统为了完成此任务整体消耗的能量。α为权重因子，表示其在总消耗中所占比重。

$$cost = \alpha \times T_{total} + (1-\alpha) \times E_{total} \tag{3.3}$$

3.2.1 时延模型

对于任务时延的计算，根据其卸载策略不同，对应的时延计算方法也不同。选择本地计算时总时延与任务计算量大小和本地计算能力有关，而选择卸载到服务器计算时总时延由三部分组成：任务上传时延，计算时延，结果下发时延。为了方便计算，在此模型中认为所有设备的计算能力与其 CPU 频率呈线性正相关关系；本地、边缘云服务器、中心云服务器互相之间传输带宽均设置为 B。部分参数设置如表 3.1 所示。

表 3.1 时延相关参数列表

参数含义	参数设置	参数含义	参数设置
本地设备 CPU 频率	f_{local}	本地消耗时延	T_{local}
边缘云 CPU 频率	f_{edge}	边缘云消耗时延	T_{edge}
中心云 CPU 频率	f_{center}	中心云消耗时延	T_{center}

(1) 本地计算

当任务选择本地计算时，因为计算工作在本地完成，所以得出计算结果后认为时延计时完毕。所以任务总时延只与任务计算量大小 W 和本地计算能力有关。所以任务总时延的公式为：

$$T_{total} = T_{local} = \frac{W}{f_{local}} \tag{3.4}$$

(2) 卸载到边缘云计算

当任务选择卸载到边缘云进行计算时，任务要完全传递到边缘云服务器才能开始计算，则需要计算其上传时延 T_{upload}，为了方便计算，在此模型中认为上传时延只和传输带宽 B 和任务大小 S_t 相关；当任务完全上传至边缘服务器，边缘服务器立即开始进行任务计算，此时任务计算时延 T_{work} 与任务计算量 W 和边缘服务器计算能力有关；计算完成后，边缘云服务器马上将计算结果返回给本地设备，此时下发时延 T_{down} 与传输带宽 B 和结果大小 S_r 相关。所以综上所述，任务

总时延计算公式为：

$$T_{total} = T_{edge} = T_{upload} + T_{work} + T_{down} \tag{3.5}$$

$$T_{upload} = \frac{S_t}{B} \tag{3.6}$$

$$T_{work} = \frac{W}{f_{edge}} \tag{3.7}$$

$$T_{down} = \frac{S_r}{B} \tag{3.8}$$

(3) 卸载到中心云计算

当任务选择卸载到中心云服务器进行计算时，根据第一节中的系统场景可知，任务首先要上传到边缘云，再由边缘云传输给中心云服务器，其计算结果也要同样经由“边缘云之手”下发给本地设备。所以卸载到中心云进行计算总时延的构成是相同的，T_{total} 和 T_{center} 计算方式同公式 3.5。而其中不同点在于每一个分步时延的计算：上传时延分为本地上传到边缘云的时间 T_{L2Eup} 和边缘云上传到中心云的时间 T_{E2Cup}。计算时延与任务计算量 W 和中心云服务器计算能力有关。下发时延由中心云下发到边缘云的时延 T_{C2Ed} 和边缘云下发到本地的时延 T_{E2Ld} 组成。所以相应的计算公式为：

$$T_{L2Eup} = T_{E2Cup} = \frac{S_t}{B} \tag{3.9}$$

$$T_{C2Ed} = T_{E2Ld} = \frac{S_r}{B} \tag{3.10}$$

$$T_{upload} = T_{L2Eup} + T_{E2Cup} \tag{3.11}$$

$$T_{work} = \frac{W}{f_{center}} \tag{3.12}$$

$$T_{down} = T_{C2Ed} + T_{E2Ld} \tag{3.13}$$

3.2.2 能耗模型

对于系统能耗的计算，分为计算能耗和传输能耗两种能耗类型。在计算能耗时，认为执行任务计算的设备以工作功耗进行工作，而其它设备以静止能耗处于待机状态。同样的在计算传输能耗时，发起传输任务的设备以工作功耗进行工作直到任务被完全传输，而包括接收端设备在内的其他设备则以静止能耗处于待机状态。计算的系统总能耗为所有设备产生的能耗的总和，根据系统场景设置总能耗为本地设备的能耗 E_{local}、边缘云服务器设备的能耗 E_{edge} 和中心云服务器设备

的能耗 E_{center} 的总和（公式 3.14）。对于不同设备的不同功耗的符号表示如表 3.2 所示。

$$E_{total} = E_{local} + E_{edge} + E_{center} \tag{3.14}$$

表 3.2 功耗相关参数列表

参数名称	参数含义	参数名称	参数含义
K_{local}	本地设备工作功耗	Q_{local}	本地设备静止功耗
K_{edge}	边缘云工作功耗	Q_{edge}	边缘云静止功耗
K_{center}	中心云工作功耗	Q_{center}	中心云静止功耗

(1) 本地计算

当任务选择本地计算时，由于不产生任务传输的工作，传输功耗不列入总功耗计算。总功耗只包括本地设备计算产生的能耗和其他设备待机的能耗，而总时延为公式 3.4 中计算时延，则计算公式如下：

$$E_{total} = T_{total} \times K_{local} + T_{total} \times Q_{edge} + T_{total} \times Q_{center} \tag{3.15}$$

(2) 卸载到边缘云计算

任务选择卸载到边缘云计算，任务需要先进行上传，上传完成后才能进行计算，随后计算结果再下发给本地设备。在此过程中不涉及中心云服务器参与，认为中心云服务器始终处于待机状态。根据公式 3.6，3.7 和 3.8 可以求出每一步的时延，根据分步时延相应的计算每个设备的能耗，进而根据公式 3.14 求出总能耗。

$$E_{local} = T_{upload} \times K_{local} + \left(T_{total} - T_{upload}\right) \times Q_{local} \tag{3.16}$$

$$E_{edge} = T_{work} \times K_{edge} + T_{down} \times K_{edge} + T_{upload} \times Q_{edge} \tag{3.17}$$

$$E_{center} = T_{total} \times Q_{center} \tag{3.18}$$

(3) 卸载到中心云计算

当选择卸载到中心云计算时，卸载过程需要边缘云“推波助澜”，所有设备均有工作状态和待机状态。本地设备只有上传一次任务为工作状态，其余时间均为待机；边缘云服务器需要上传一次任务和下发一次结果，除此之外为待机状态；

中心云服务器需要进行任务计算，且要将结果下发给边缘云服务器，其余时间为待机状态。因为每个设备的工作功耗不同，所以需要更加细化上传时间和下发时间，所以需要采用公式 3.9 和公式 3.10 中的“步骤时间”进行能耗的计算。所以具体计算公式如下：

$$E_{local} = T_{L2Eup} \times K_{local} + \left(T_{total} - T_{L2Eup}\right) \times Q_{local} \tag{3.19}$$

$$E_{edge} = \left(T_{E2Cup} + T_{E2Ld}\right) \times K_{edge} + \left(T_{total} - T_{E2Cup} - T_{E2Ld}\right) \times Q_{edge} \tag{3.20}$$

$$E_{center} = \left(T_{work} + T_{C2Ed}\right) \times K_{center} + \left(T_{total} - T_{work} - T_{C2Ed}\right) \times Q_{center} \tag{3.21}$$

3.2.3 限制条件

根据系统场景中的一些初始条件，需要对于整个计算卸载场景附加一些约束条件以符合系统场景的设置。

$$\alpha \in [0,1] \tag{3.22}$$

$$T_{total} = a \times T_{local} + b \times T_{edge} + c \times T_{center} \tag{3.23}$$

$$a \in \{0,1\}, b \in \{0,1\}, c \in \{0,1\} \tag{3.24}$$

$$a + b + c = 1 \tag{3.25}$$

公式 3.22 是对公式 3.3 的限制，表示时延权重和能耗权重只能取包含 0 到 1 之间的数值且两项权重相加必须为 1，保证系统消耗为任务时延和系统能耗的有效线性加权和。公式 3.23 到公式 3.25 为对于卸载策略的限制，其中公式 3.24 使用 1 和 0 来表示是否选择此种卸载策略，*a* 代表本地设备计算，*b* 代表卸载到边缘云计算，*c* 代表卸载到中心云计算，而公式 3.25 限制了一种任务只能选择一种卸载策略进行处理。公式 3.23 表示对于此任务时延的计算只能取其卸载策略所对应的时延计算方式，而能耗的计算限制与实验的计算限制类似，所以不再列出。

第四章 仿真结果分析

第一节 仿真实验参数设置

论文采用 Python 作为实验平台进行仿真实验。系统场景的部分参数参考文献[21,22]，具体实验参数如表 4.1 所列。

表 4.1 仿真实验参数设置

参数意义	参数符号	参数取值
本地终端静态 CPU 功耗/mW	Q_{local}	10
边缘云节点静态 CPU 功耗/mW	Q_{edge}	20
中心云节点静态 CPU 功耗/mW	Q_{center}	40
本地终端工作 CPU 功耗/mW	K_{local}	90
边缘云节点工作 CPU 功耗/mW	K_{edge}	200
中心云节点静态 CPU 功耗/mW	K_{center}	400
本地终端 CPU 频率	f_{local}	1GHz
边缘云节点 CPU 频率	f_{edge}	4GHz
中心云节点 CPU 频率	f_{center}	8GHz
设备之间传输速率/(MB/s)	B	5
一组任务中包含任务数量	N_{task}	10
一组任务总数据大小	S_{total}	~U(140,160)
服务器最多同时处理任务数量	N_{max}	4
任务大小取值范围/MB	S_t	~U(5,20)
任务计算量系数	γ	1000
染色体种群中染色体数量	n	50
一次算法中迭代次数	cycle	200
计算结果大小系数	δ	0.1

任务计算量 W 为任务大小 S_t 和任务计算量系数γ的乘积，任务计算量系数选择 1000 的原因是任务大小单位为 MB，而 CPU 频率单位为 GHz，当任务量系数为 1 时就会导致计算时延的单位为μs，而上传时延和结果下发时延单位均为 ms，

这种情况下会导致计算时延在时延计算中所占比例低，从而选择使用较大的任务计算量系数使得计算时延单位为 ms。计算结果相对于计算任务来说数据量较小，而过小的计算结果大小会导致结果下发时延极短，所以将原任务大小 S_t 与计算结果大小系数δ=0.1 相乘作为计算结果的数据大小 S_r。

定向选择性遗传算法中部分相关参数根据经验进行设置,其中交叉概率设置为 80%，变异概率设置为 3%[21]。这样设置的原因是当交叉概率过低时会导致生成子代染色体数量较少，迭代一定次数后初始染色体组进化结果较差；变异率通常在 0.1%到 10%中设置，当变异率设置较大时会使子代染色体有很大机率发生突变，影响优秀基因的遗传。仿真实验采用 10 组随机任务进行循环测试，目的是多收集数据从而避免一些随机过程导致的特殊实验结果影响实验结果分析。

第二节 全部本地计算

首先测试的算法为全部本地计算，实验结果如柱状图 4.1 所示，其中浅红色代表全部本地计算的系统消耗，紫色为标准对比值，横坐标为实验次数，纵坐标为系统消耗代价（单位：1），系统消耗计算方式为公式 3.3，α取值 0.5。

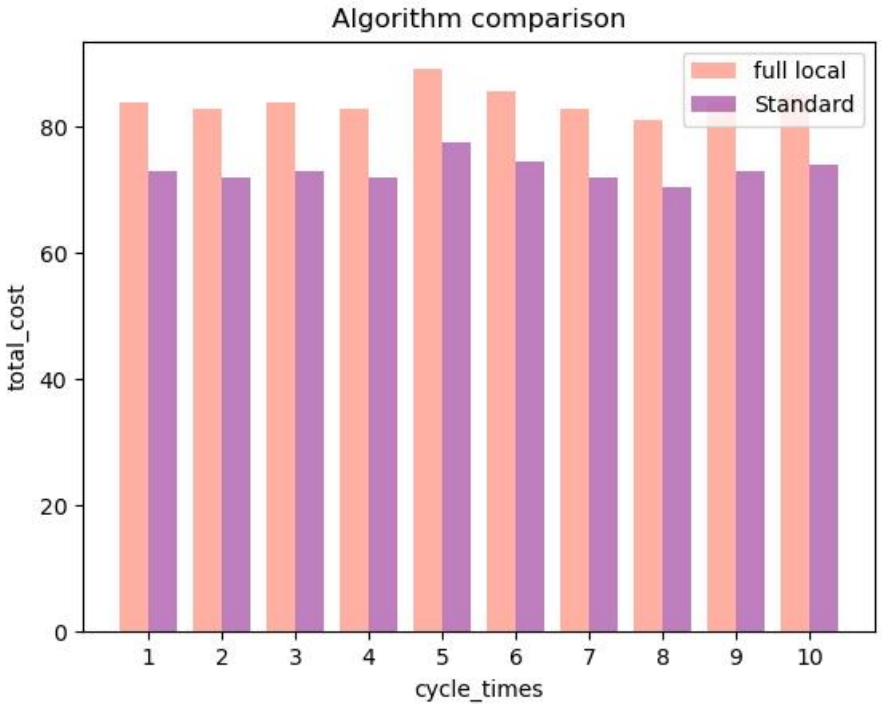

图 4.1 全部本地计算与标准消耗值对比

图中的标准对比值取值方法为当次实验的组任务数据总大小的一半，认为系统消耗小于此值时策略为良好策略。根据图中对比可以发现，全部本地计算处理策略所造成的系统代价全部大于所设置的标准对比值，其平均值比标准值高出15%，由此可知对于此系统场景，全部本地计算不是一个良好策略，其原因为本地计算能力较差，仿真实验中设置的任务大小及任务计算量参数较大，对于本地终端设备来说，每个任务所需要计算的时间都相当大，导致整体任务的时延相对较大；对于能耗方面，虽然本地设备的运行功耗较低，但相对应的运行时间较长，造成本地设备终端的能耗较大，同时由于整体任务的时延较大，对于处于待机状态的边缘云服务器和中心云服务器来说，也造成了一定的能耗，所以综合上述情况能耗方面全部本地计算表现也较差，综合所有因素导致系统消耗普遍较高。

第三节 随机策略卸载

随机策略卸载和标准对比值的对比结果如图 4.2 所示，其中紫色柱为标准对比值，橙色柱为随即策略卸载的系统消耗，横坐标为实验次数，纵坐标为系统消耗（单位：1），系统消耗计算方式为公式 3.3，α取值 0.5。

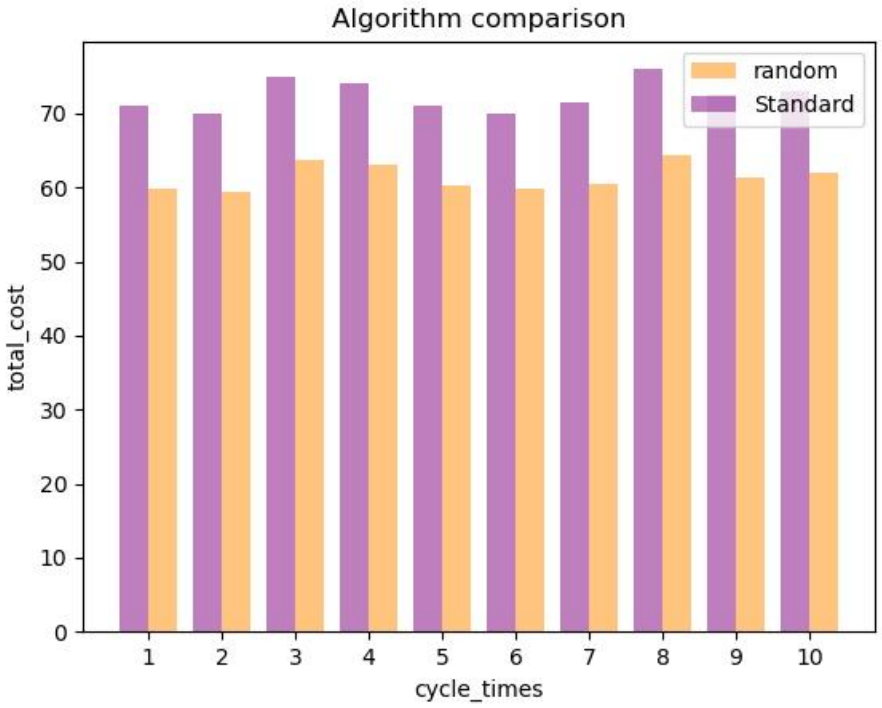

图 4.2 随机卸载策略和标准消耗值对比

由于随机策略卸载的随机性过强，有可能在随机时出现针对随机生成的任务组恰好生成了这组任务对应的最佳卸载策略，或者出现全部本地计算的最坏的结果。为了方便其结果与其他算法结果做对比，选择使用针对一组任务，随机生成1000 种可能的策略，取其对于此任务造成的系统消耗的平均值作为随机策略卸载的结果。仿真参数设置了一组任务中包含 10 个小任务，则对应的一种策略包含 10 个卸载方法，而在随机策略卸载中每一个卸载方法的取值是随机的，总共会有 3^{10}=59049 种不同的卸载策略，因此 1000 种策略里出现重复策略的可能性较低。

由图 4.2 可以看到随机策略卸载造成的系统消耗较低，且均小于标准对比值，其平均值比标准值平均减少了 15.2%。但这个对比结果无法说明随机卸载策略是一种良好的卸载策略算法，其原因为：图中显示的是 1000 个策略造成的系统消耗的平均值，并不能代表随机出一种策略就能达到此消耗。在实际应用中，随机策略卸载不会像仿真实验一样取平均值或取其中的最好策略进行计算卸载，而是随机一种策略直接执行，这样做不能保证每次生成的卸载策略对于这组任务来说都是好的卸载策略，且有可能遇到最坏的情况：生成的策略使得任务全部在本地执行。综上所述随机卸载策略是不稳定的一种“赌博”性质的卸载策略算法。

第四节 原始遗传算法

原始遗传算法作为一种启发性算法本身对于求解 NP-Hard 问题就已经是一种不错的解决方案，在仿真实验中，实验的大部分条件也都支持原始遗传算法进行最优解的寻找。但是，由于有染色体生成规则的限制，在交叉生成新染色体过程中原版算法可能会遇到问题，导致当次迭代过程对于染色体组进化过程无效，使得在当前系统场景下原版遗传算法由于迭代次数的限制输出的结果不是所想要找的全局最优解。但相对于其他基准算法来说，其输出结果是最接近于全局最优解的。

图 4.3 是全部本地计算策略算法和原始遗传算法输出的系统最小消耗值的对比，浅红色为全部本地计算策略，浅绿色为原始遗传算法，横坐标为实验次数，纵坐标为系统消耗（单位：1），系统消耗计算方式为公式 3.3，α取值 0.5。

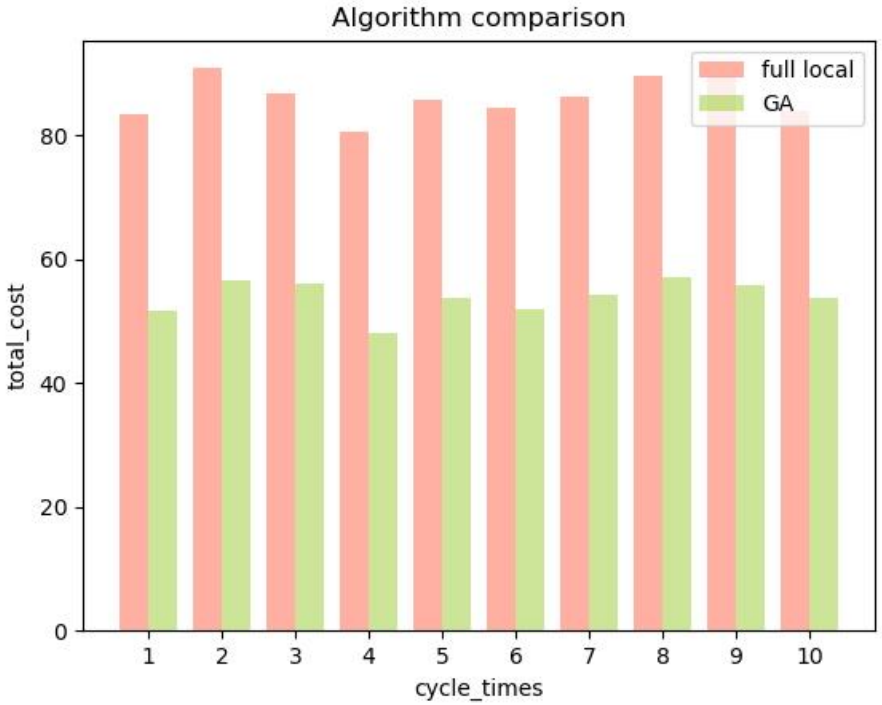

图 4.3 原始遗传算法和全部本地计算对比

根据图 4.3 可以看出相对于全部本地计算，原始遗传算法计算出的的系统消耗比全部本地计算的系统消耗平均降低了 37.7%，且相对于不稳定的随机卸载策略能够较为稳定的输出较低的系统消耗值，在当前系统场景下，原始遗传算法可以作为一个解决问题的办法。但由于缺少对比，原始遗传算法的输出结果是否已经是当前所能找到的最优解仍是未知的。

第五节 定向选择性遗传算法

在论文的系统场景下，全部本地计算策略造成的系统消耗较大，随机卸载策略输出结果不稳定，原始遗传算法虽然能输出不错的结果，但有可能因为其中的某些步骤受到限制，导致在有限的迭代次数中，输出结果是局部最优解而不是所需的全局最优解。相比之下，根据系统场景进行改进的定向选择性遗传算法理论上可以在有限的迭代次数中输出全局最优解或最接近全局最优解的局部最优解。

将定向选择性遗传算法与其他三种基准算法的输出结果进行联合对比，输出结果如图 4.4 所示，其中浅红色为全部本地计算策略，橙色为随机卸载策略算法，浅绿色为原版遗传算法，蓝色为定向选择性遗传算法，横坐标为实验次数，纵坐

标为系统消耗（单位：1），系统消耗计算方式为公式 3.3，α取值 0.5。

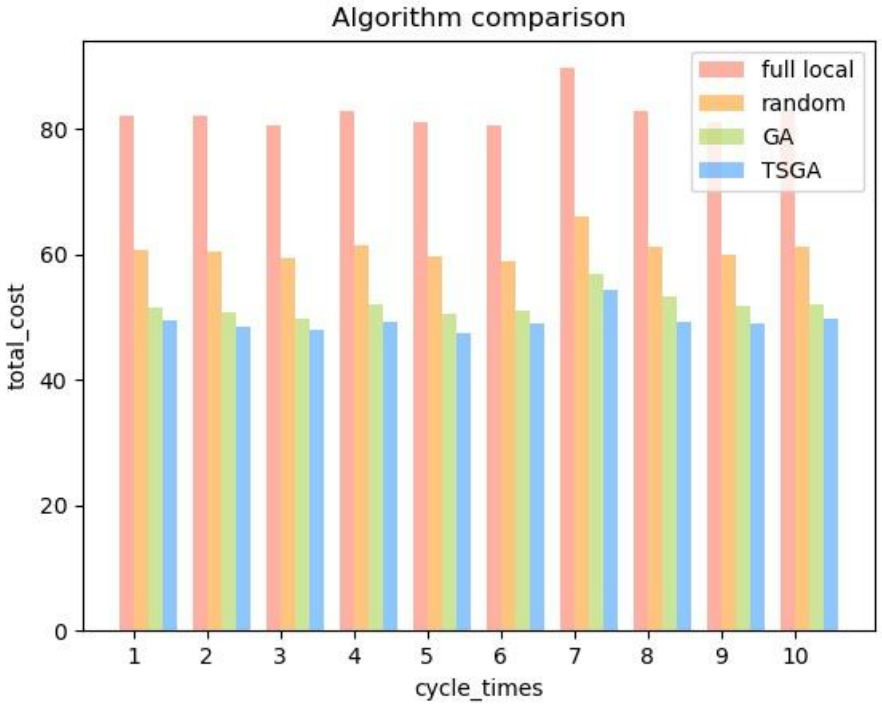

图 4.4 定向选择性遗传算法与基准算法对比

根据图 4.4 可以看到定向选择性遗传算法在当前系统环境设置下相比于其他三种基准算法输出了较低的系统消耗。相比于全部本地计算，系统消耗降低了 39.9%；相对于随机卸载策略，不仅输出的系统消耗低 18.6%，且输出结果稳定；相对于原版遗传算法，定向选择性遗传算法能够在有限的迭代次数中输出更接近全局最优解的局部最优解，输出的系统消耗也下降了原版算法的 3.5%。

上述测试中，系统消耗计算公式中取的时延权重为 0.5，能耗权重为 0.5。在实际生活中，由于不同场景中使用的设备不同，对于时延和能耗的要求也不尽相同，例如：当产生任务的本地设备为用户的手机时，其对于时延的要求一般较低，但是由于现在手机普遍电池容量较小，用户对于电量消耗极为敏感，不能因为请求了一个计算服务就消耗了手机 20%的电量，因此对于本地设备的能耗要求较高，这就需要将计算公式 3.3 中的时延权重降低而能耗权重升高；当产生任务的本地设备为 VR 游戏的设备时，由于 VR 游戏需要给玩家及时的环境反馈，且对于玩家的行动需要进行大量的计算并投影到虚拟环境中并即时反馈给玩家，其对于任务的时延要求极为严格，而 VR 设备大多是直接连接电源的，其对于计算任务所需的能耗并不敏感，相应地在计算公式中时延权重增加而能耗权重降低。

为了测试不同时延权重和能耗权重对于几种算法的输出结果的影响，选择了时延权重为 1、能耗权重为 0 和时延权重为 0、能耗权重为 1 的两种情况分别进行仿真模拟。仿真模拟结果分别如图 4.5，图 4.6 所示，其中在图 4.5 中，横坐标为实验次数，纵坐标为任务完成时延（单位：ms）；在图 4.6 中，横坐标为实验次数，纵坐标为所有设备能耗（单位：J）。

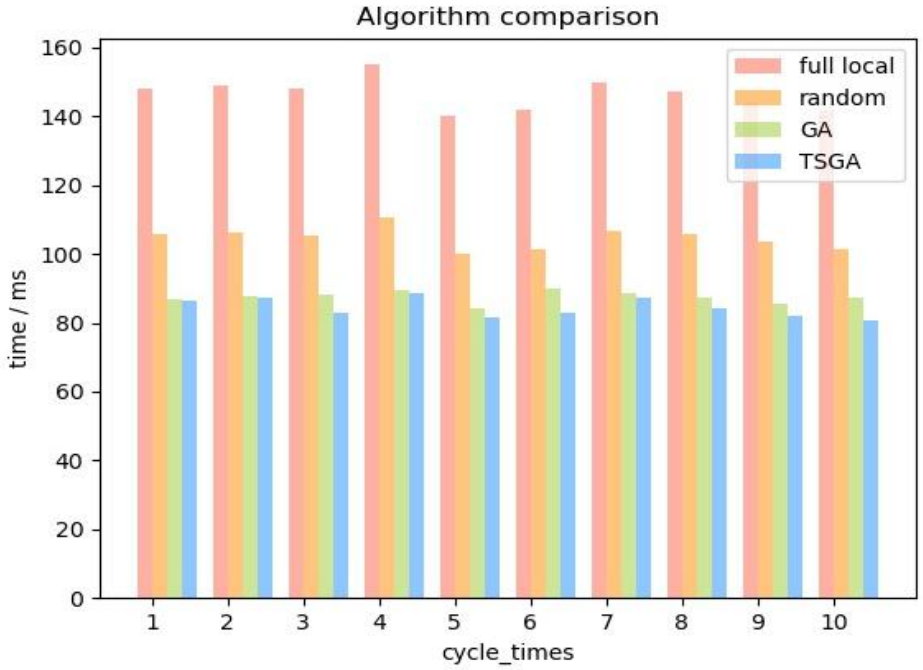

图 4.5 不同算法输出的最小时延

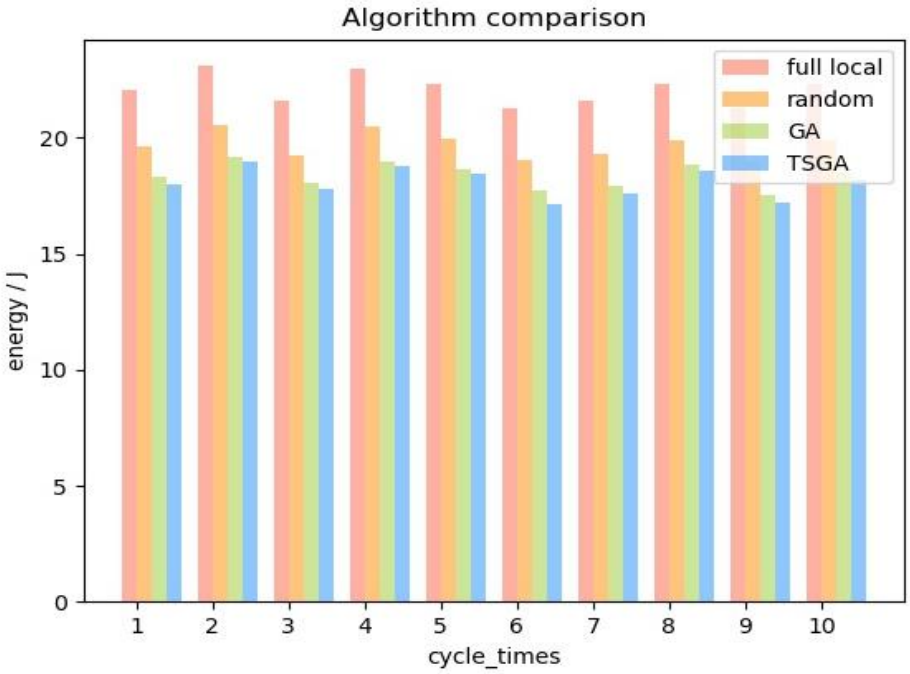

图 4.6 不同算法输出的最小能耗

可以看到在两种权重取值情况下，定向选择性遗传算法输出的系统消耗相比于原版算法平均分别下降了 4%和 2%，证明在当前系统环境下，定向选择性遗传算法能够更好的输出全局最优解。有的实验结果中可能会出现原版遗传算法结果和定向选择性遗传算法结果相同的情况，这是由于原版遗传算法所使用的初始染色体组相较于定向选择性遗传算法较为优秀，而初始染色体组一定程度影响最后的结果，因此原版遗传算法得到了和定向选择性遗传算法相近或相同的全局最优解，但这种情况属于特殊情况，出现概率较低，整体上来说定向选择性遗传算法输出结果比原版遗传算法输出结果要好。

第五章　工作总结及未来展望

第一节 本文工作总结

论文对于边缘计算中的计算卸载问题进行了研究，并提出了一种定向选择性遗传算法来解决计算卸载决策问题。首先对于系统环境进行建模，建立了用户-边缘云-中心云的三层模型；然后详细介绍了定向选择性遗传算法的工作流程，并与原始遗传算法进行了对比讨论，介绍了本算法的改进方法及改进意义；随后对于计算卸载策略的评判标准进行了数学建模，使用任务完成时延和系统能量消耗的线性加权和作为系统消耗的计算方式，并对于时延和系统能耗在不同卸载策略下的不同计算方式进行了讨论；最后进行了仿真实验以对比定向选择性遗传算法和其他基准算法的输出结果，根据仿真实验结果得出结论：改进后的算法更加适应当前系统场景的计算卸载决策问题。

由于时间、工作量和本人能力等原因，论文尚有不足之处。对于改进后的算法其中的交叉概率和变异概率，未能测试其变化对于最后输出结果的影响，只是根据以往经验选择了较为中立的数值进行仿真实验；在仿真实验结果对比环节，只是寻找了几个基准算法进行了结果对比，并未寻找更多相关启发式算法及其结果进行对比。

第二节 未来展望

移动边缘计算作为云计算的一种延伸形式[23-25]，是解决传统云计算中设备距离中心云较远带来的延迟和能耗问题的一种新兴技术，而计算卸载作为 MEC 中的关键问题，如何高效的进行计算卸载是近些年的研究热点。论文对于边缘计算中的三层模型中的计算卸载问题提出了一种定向选择性遗传算法，通过对于遗传算法中的某些随机过程改进为定向选择过程从而改进算法性能，能够很好的解决当前模型的计算卸载决策问题。

未来考虑将算法中的交叉概率和变异概率改为动态调整的数值，使得算法在未得出最优解的迭代中增加交叉概率而减少突变概率，使得算法能够有充分生成子代染色体的环境，而在算法得到的子代染色体适应度逐渐稳定时减少交叉概率

增加变异概率，使得算法突破局部最优解，为染色体组增加新的变化；论文方案只考虑了单边缘节点的场景，而在现实生活中，边缘网络中的边缘云节点多可以进行联合卸载[26]，这种情况下计算卸载的方式就多了一种选择，可以在论文方案加入考虑卸载到异地边缘的情况，进一步完善算法；由于论文提出的算法其中也有进化过程，而机器学习相关算法在迭代进化方面具有传统算法所不具有的优势，在未来可以考虑将机器学习的相关算法与论文算法结合起来进行进一步改进以适应更多的系统场景。

参考文献

[1] Marz N, Warren J. Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Greenwich, USA: Manning Publications Co. , 2015

[2] Shi Weisong, Sun Hui, Cao Jie,*et al*. Edge Computing- An Emerging Computing Model for the Internet of Everything Era[J]. Journal of Computer Research and Development, 2017, 54(5): 907-924.

[3] Zhao J, Liu Y, Gong Y, *et al.* A dual-link soft handover scheme for C/U plane split network in high-speed railway. IEEE Access, 2018, 6: 12473-12482

[4] Patel M, Naughton B, Chan C, *et al*. Mobile-edge computing introductory technical white paper[J]. White paper, mobile-edge computing (MEC) industry initiative, 2014, 29: 854-864.

[5] 宁振宇, 张峰巍, 施巍松. 基于边缘计算的可信执行环境研究, 计算机研究与发展, 2019, 56(7): 1441-1453

[6] Wang T, Luo H, Zheng X, *et al*. Crowdsourcing mechanism for trust evaluation in CPCS based on intelligent mobile edge computing[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(6): 1-19.

[7] 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.

[8] 张依琳, 梁玉珠, 尹沐君, 等. 移动边缘计算中计算卸载方案研究综述[J]. 计算机学报, 2021.

[9] Liu J, Mao Y, Zhang J, *et al*. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE international symposium on information theory (ISIT). IEEE, 2016: 1451-1455.

[10] 黄冬晴, 俞黎阳, 陈珏, 等. 面向移动边缘计算的联合计算卸载和资源分配策略研究[J]. 华东师范大学学报 (自然科学版), 2021, 2021(6): 88.

[11] Dinh T Q, Tang J, La Q D, et al. Offloading in mobile edge computing: Task allocation and computational frequency scaling[J]. IEEE Transactions on Communications, 2017, 65(8): 3571-3584..

[12] Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing[J]. IEEE Transactions on Wireless Communications, 2012, 11(6): 1991-1995.

[13] Cerutti G, Prasad R, Brutti A, et al. Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms[J]. IEEE Journal of Selected Topics in Signal Processing, 2020, 14(4): 654-664.

[14] John H H. Adaptation in Natural and Artificial Systems. Cambridge:The MIT Press, 1992

[15] 唐文琦, 曾干敏, 刘泽宇. 浅谈遗传算法及其部分改进算法[J]. 科技风, 2019, 12.

[16] 陈斌, 徐华中. 一种改进遗传算法及其在 TSP 问题中的应用[J]. 计算机工程, 2002, 28(9): 90-92.

[17] Tian L, Collins C. An effective robot trajectory planning method using a genetic algorithm[J]. Mechatronics, 2004, 14(5): 455-470.

[18] Chen B, Quan G. NP-hard problems of learning from examples[C]//2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery. IEEE, 2008, 2: 182-186.

[19] Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A: Statistical Mechanics and its Applications, 2012, 391(6): 2193-2196.

[20] Poli R, Langdon W B. Genetic programming with one-point crossover[M]//Soft Computing in Engineering Design and Manufacturing. Springer, London, 1998: 180-189.

[21] 高基旭, 王珺. 一种基于遗传算法的多边缘协同计算卸载方案[J]. 计算机科学, 2021, 48(1): 72-80.

[22] 唐宁. 边缘计算的物联网任务处理策略研究: [硕士学位论文]. 北京邮电大学, 2021.

[23] Zhou Y Z, Zhang D. Near-end cloud computing: opportunities and challenges in the post-cloud computing era[J]. Chinese Journal of Computers, 2019, 42(4): 677-700.

[24] 曹芷晗, 卢煜成, 赖思思, 等. 基于边缘计算的传感云研究进展[J]. Journal of Software, 2019, 30(11): 40-50.

[25] LIU Z, LI S, LI B, *et al*. New elastic collision optimization algorithm and its application in sensor cloud resource scheduling[J]. Journal of Zhejiang University (Engineering Science), 2018, 52(8): 1431-1443.

[26] Dinh T Q, La Q D, Quek T Q S, et al. Learning for computation offloading in mobile edge computing[J]. IEEE Transactions on Communications, 2018, 66(12): 6353-6367.

致 谢

在这里，感谢吴英老师在我本科毕设和论文书写过程中对我的悉心指导和无私关怀。是吴老师在我遇到困难时给我提供帮助，帮助我解决论文主题的问题，并深入探讨了研究方向和实现难度；在实验期间，吴老师也及时对我的工作进行了关注和纠正，使我少做了很多无用功；论文书写方面，吴老师对我的书写规范和格式要求严格细致，在论文书写的每个阶段都进行了详尽的指导并及时给出修改意见。在这里再次感谢吴老师给予我的极大帮助，您的教诲和指导我受益匪浅。

感谢实验室的李君龙学长在我的实验期间为我提供的帮助，学长在我进行期间为我提供了大量的改进思路和改进方法，使我的实验结果在原来的基础上有了更好的提升，同时也感谢学长为我提供的实验室的优秀实验环境和毕设的设计经验，再次表示感谢。

还要感谢各位大学期间担任我的课程的各位老师，是您们的努力让我不仅学会了知识，还学会了获取知识的能力。正是有您们为我打下的优秀基础使我能够完成这次的本科毕业设计，十分感谢各位老师们。

感谢我的舍友们在这四年来对我的包含和照顾，没有你们的支持，我很难能有现在这样的成绩。还感谢各位校园内的工作人员们，为我们提供了良好的校园环境和生活便利。感谢南开大学让我成为了更好的自己，也让我遇到更多非常好的朋友们。

最后感谢我的父母，对你们的感恩之情是语言远远无法表达出来的，我会永远肩负着你们的希望，铭记对我的恩情，继续前进下去的！