文章编号:1671-4598(2022)11-0233-07 DOI:10.16526/j.cnki.11—4762/tp.2022.11.034 中图分类号:TP393.4 文献标识码:A

基于混合蚁群优化的边缘计算细粒度任务调度方法

陈 刚,王志坚
(广州华商学院 数据科学学院,广州 511300)

摘要:为缓解中心服务器的压力,制定合理的调度方案,基于混合蚁群优化算法提出了边缘计算细粒度任务调度方法;描述边缘计算任务调度问题,并设置假设条件,简化调度求解难度;通过计算任务的优先指数,按照从大到小的顺序排列后组成任务队列;分析边缘服务器性能特征,明确边缘服务器处理能力;构建能耗以及时延多目标函数,并设置约束条件,利用混合蚁群优化算法求解多目标函数,完成边缘计算细粒度任务调度方案设计;仿真结果表明:在5 000 m×5 000 m的仿真区域内,该方法应用下的任务调度能耗可以控制在150 kW/h以内,任务调度时延在5 s以内,说明该方法性能更优,所获得的调度方案更合理。

关键词:蚁群算法;遗传算法;边缘计算;细粒度任务;边缘服务器

Method of Fine Grained Task Scheduling for Edge Computing Based on Hybrid Ant Colony Optimization

CHEN Gang, WANG Zhijian
(School of Data Science, Guangzhou Huashang College, Guangzhou 511300, China)

Abstract: In order to alleviate the pressure of the central server and formulate a reasonable scheduling scheme, a fine-grained task scheduling method based on edge computing is proposed based on hybrid ant colony optimization algorithm. The scheduling problem of edge computing tasks is described, and the assumptions are set to simplify the difficulty of scheduling. By calculating the priority index of tasks, the task queue is formed after being arranged in the order from large to small. The performance characteristics of edge server are analyzed, and the processing capacity of edge server is clarified. The multi-objective function of energy consumption and delay is constructed, and some constraints are set. The multi-objective function is solved by the hybrid ant colony optimization algorithm to complete the design of fine-grained task scheduling scheme for edge computing. In a simulation area of 5 000 m×5 000 m, the experimental results show that the task scheduling energy consumption of the method may control within 160 Kw/h, the delay of task scheduling is 4 s, which shows that the performance of the proposed method is better, and the scheduling scheme is more reasonable.

Keywords: ant colony optimization; genetic algorithm; edge calculation; fine grained tasks; edge server

0 引言

云计算为客户端提供了海量的可以访问的数据资源以及计算资源,因此每时每刻都有大量的用户向云平台提交任务申请,而云平台也需要根据申请来处理这些任务。面对用户申请的大规模任务,云平台原有的网络架构出现了较为严重的时延问题[1]。为此,近年来边缘计算的观点被提出来。边缘计算旨在分担云计算中心的任务处理压力,提高服务质量。然而,由于边缘计算节点的资源和计算能力都是有限的,它只能处理少量的部分任务。面对这种情况,如何合理地将任务调度给对应的边缘节点[2],成为了新的难题问题。造成任务调度困难的原因有二个,一是边缘节点距离云计算中心的距离不同,距离越远,任务调度所耗费的时间和能耗就越多;二是不同任务所需要的计算资源是不同的,若是将小任务调度给计算能力很强的边缘节点上,将造成资源浪费;反之,边缘计算因为能力不够,将无法处理任务,服务质量受到严重限制[3]。所以如何根据任务的特征以及边缘节点的计算资源、距离,将任务合理地调度给合适的边缘计算节点至关重要。

国内外针对边缘计算的任务调度都进行相关研究。文献[4]首先将计算任务描述为一个有向无环图,然后将系统的延迟作为优化目标,并在截止期限、优先级、节点完成期限等三个约束条件下,利用改进的NSGA-Ⅱ算法求解获取了任务卸载调度方案。文献[5]以系统总开销为优化问题,并将其细化分为三个子问题,以方便求解,利用自适应遗传算法对其逐一求解,得到综合任务调度卸载方案。文献[6]以任务满意度最大化为目标,将多个任务调度到

收稿日期:2022-07-27; **修回日期**:2022-08-29。
基金项目:广州华商学院校级导师制科研项目(2022HSDS16)。
作者简介:陈 刚(1973-),男,湖南长沙人,硕士,讲师,主要从事信息安全与云计算方向的研究。
王志坚(1970-),男,湖南长沙人,博士,教授,硕士生导师,主要从事控制理论与控制工程方向的研究。
引用格式:陈 刚,王志坚.基于混合蚁群优化的边缘计算细粒度任务调度方法[J].计算机测量与控制,2022,30(11):233-239.

边缘服务器上配置的虚拟机上，利用深度强化学习解决了时间调度和资源分配的问题。文献［7］提出了一种基于烟花模型的调度方法，首先采用焰火视图来搜索资源，以满足任务的功能需求，然后根据任务相关性和资源对任务进行分包，最后通过构建三维空间距离来计算任务包与资源之间的匹配度。上述调度算法在时间延迟和调度能耗上都不是很理想，需要采用新型智能算法对边缘计算的任务调度进行改进与优化。

边缘计算任务调度是一个大规模且复杂的问题，单一的蚁群算法的优化能力并不能很好地应对，因此，本文将基础蚁群算法与遗传算法相结合，实现边缘计算细粒度任务高效率调度，以期为边缘计算任务调度问题提供参考和借鉴。

1 边缘计算细粒度任务调度的原理

边缘计算细粒度任务调度属于一种连续性问题，使用单一的蚁群算法进行调度，容易产生收敛速度慢，计算时间长，易于过早陷入局部最优的问题。为此，本文引入遗传算法，构建混合蚁群算法对细粒度任务调度进行优化，解决单一蚁群算法易于过早陷入局部最优的问题，提高边缘计算细粒度任务调度性能，具体描述如下。

1.1 问题描述

边缘计算任务调度是指将原本应由本地服务器或中心云端处理的任务合理地分配给网络边缘端的服务器节点，以缓解本地服务器或中心云端的任务处理压力，减少任务处理延迟[8-9]。边缘计算任务调度传统模型如图 1 所示，本文边缘计算细粒度任务调度模型如图 2 所示。本文调度模型分为底部的终端设备层和上端的边缘层。终端设备层主要包括客户端中用到的手机、平板电脑等，其中一部分会产生计算密集型任务，这些任务无法在本机设备提供的资源下完成任务，需要为其制定合理的调度策略将任务迁移到合适的地方，可以在满足任务时延要求的情况下完成任务。终端设备层的另一部分设备由没有任务处理的闲置终端用户组成。在某些时刻其 CPU 处于闲置状态，可以为其他 CPU 过载的终端用户提供计算资源。

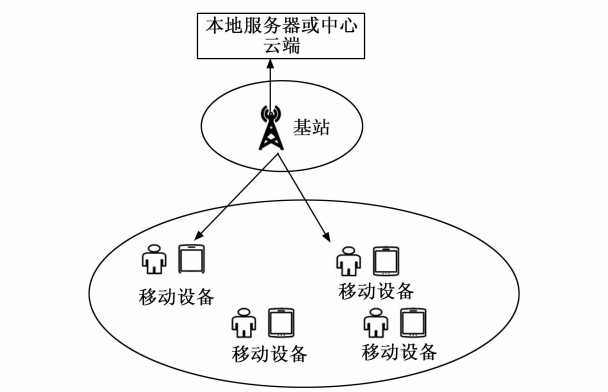

图 1 移动计算任务调度的传统模型

由图 2 可知，每个边缘服务器都带有一个基站，其作用是接收发送过来的任务，当任务处理完成后再将处理结果通过基站发送给移动用户，完成任务调度[10]。

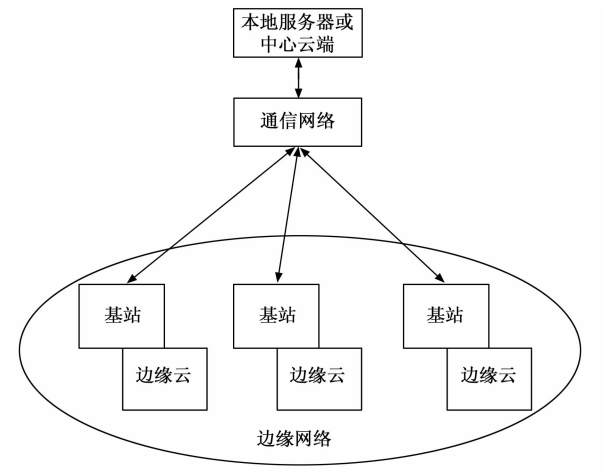

图 2 现代移动边缘计算任务调度模型

1.2 移动边缘计算任务调度的总体架构

移动边缘计算是一种新型移动通信网络技术，在移动网络边缘提供网络环境，将网络大量的任务进行缓存，该技术可以处理比较复杂的网络边缘任务，移动边缘计算具有较好的任务计算能力，可以分析、处理海量网络任务。该移动边缘计算节点在地理位置方面与信息源比较接近，用户发送请求时，网络能快速响应用户的请求，极大地减小了响应的时延，使网络中的数据传输网和核心网络不会发生网络拥塞，移动边缘计算技术还可以实时获取各大网络基站 ID、用户的网络数据、用户请求等信息，对网络中的链路可进行有效的感知自适应，为用户在网络边缘处部署位置应用，从而提升网络用户的服务质量。本文采用的基于移动边缘计算总体组成结构如图 3 所示。

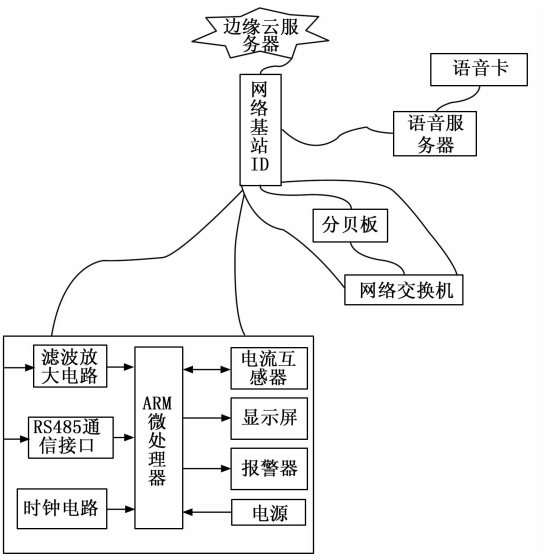

图 3 基于移动边缘计算任务调度的总体结构

1.3 细粒度任务调度的原理

移动边缘计算是将网络控制、数据处理和相关存储迁移到网络边缘，为覆盖范围内的移动用户提供较好的密集

型计算服务。任务调度方法就是要实现基站与用户之间的互动，在用户密集且流动性大的情况下实现任务合理分配，其多基站协同合作的示意图如图 4 所示。

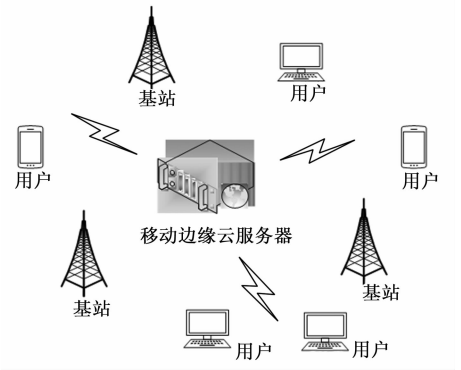

图 4　移动边缘云服务器任务调度示意图

移动边缘计算任务调度质量取决于边缘基站计算资源的分配。要想实现边缘服务器中的资源公平合理的分配，就要对智能边缘基站进行研究。边缘计算节点为基站调度任务时，主要考虑两个性能指标，分别为数据传输速率和任务执行能耗。通过任务分类合理化的方式提高数据传输速率，将优先级高的任务调度到边缘基站中，优化资源调度程序。除此之外，基站的能耗是一定有的，应在提供能量一定的前提下，让基站能源的总消耗小于所有任务执行任务结束时的总和。

2　边缘计算细粒度任务调度的设计

2.1　假设条件设置

粒度指的是根据项目模块划分的细致程度区分的标准，一个边缘计算任务调度模块划分得越多，每个子模块越小，负责的工作越细致，即所属认为的粒度越细，因此在调度细粒度任务时的难度和能耗比较于粗粒度任务调度更高。通常来说，在细粒度任务调度时，其可以根据自身的需要选择合适的边缘节点进行卸载，但是如何选择合适的边缘节点成为细粒度任务调度时的关键问题[11]。按照以往的调度方案，会选择能耗最少或者延迟时间最短作为目标，选择距离最短的边缘节点进行任务调度。然而，这种调度方案并没有将边缘节点的资源容量考虑在内，会发生任务拥塞问题，导致任务调度失败，而与此同时，边缘网络中有可能存在部分边缘服务器节点闲置的状态，造成了计算资源的浪费[12]。针对上述问题，在任务资源调度前，需要作出几项假设条件。

假设条件 1：任务调度时不能只考虑一个边缘节点作为调度对象；

假设条件 2：边缘节点的计算资源存在能源耗尽的情况；

假设条件 3：边缘网络拓扑已知，且拓扑中每个服务器节点与本地服务器、中心云端的距离是已知的；

假设条件 4：每个任务对计算资源的需求都是已知且存在差异的；

假设条件 5：边缘节点内部数个虚拟机工作模式为并行；

假设条件 6：任务调度过程中为连续执行，不存在间断执行；

假设条件 7：任务传输信道的信息是已知的。

2.2　任务调度顺序设计

任务调度器可以获取任务调度的优先级，即完成任务调度顺序设计工作[13]。任务调度器对任务的排序规则是通过计算任务的优先指数，然后从大到小的顺序排列待调度任务[14]。优先指数计算公式如下：

$$S_i = A_i - (A_i - d_i) \cdot \alpha = \frac{t_i}{b_i - c_i} - \left(\frac{t_i}{b_i - c_i} - \frac{w_i}{W}\right) \cdot n \tag{1}$$

式中，S_i 代表细粒度任务 i 的优先指数；A_i 代表细粒度任务 i 的饱和度；d_i 代表相对权重比；α 代表平衡指数；b_i 代表细粒度任务 i 执行的截止时刻；c_i 代表细粒度任务 i 执行的开始时刻；w_i 代表细粒度任务 i 执行价值；W 代表细粒度任务总价值；t_i 代表细粒度任务 i 执行最长耗时；n 代表细粒度任务数量。

基于计算出来的任务优先指数[15]，设计出任务调度顺序方案。

2.3　边缘服务器性能特征分析

边缘服务器受到其自身硬件资源的影响，其计算资源性能具有不同的特征，这就直接影响了对任务的处理能力[16]。为此，通过了解边缘服务器性能特征对于细粒度任务调度至关重要，为调度可靠性和成功率判断提供了重要依据。边缘服务器的性能可以通过静态指标和动态指标两类指标表示，考虑到静态指标对边缘计算细粒度任务调度性能的影响更高，因此，本文重点分析了边缘服务器静态指标。

静态指标表示服务器硬件资源情况的基本性能指标，其主要包括 CPU 中央处理单元的 CPU 频率、计算能力等、存储设备的任务数据量、任务达到率等。根据静态指标能够直接明确边缘服务器性能，为细粒度任务调度中边缘服务器节点的选择提供了重要的参考[17]。

2.4　任务调度方案设计

在上述各研究成果的基础上，本章节基于混合蚁群优化设计细粒度任务调度方案，该部分分为目标函数构建，约束条件设置[18]以及混合蚁群优化算法求解三部分。

2.4.1　目标函数构建

细粒度任务调度旨在解决任务调度所需要的能量（能耗）以及任务调度所需要的时间（时延）两个问题。针对这两个问题，本研究中设置的目标函数为多目标函数[19]。函数描述如下：

$$\begin{aligned} \min Y &= y_1 \cup y_2 \\ y_1 &= Q_i(1) - P_i\left[Q_i(1) - \sum_{j=1}^{m} q_{ij} Q_i(2)\right] \\ y_2 &= T_i(1) - P_i\left[T_i(1) - \sum_{j=1}^{m} q_{ij} T_i(2)\right] \end{aligned} \tag{2}$$

式中，$\min Y$ 代表综合目标最小值；y_1 代表细粒度任务调度能

耗；y_2 代表细粒度任务调度时延；$Q_i(1)$、$T_i(1)$ 代表任务 i 在本地处理时的能耗、时延；P_i 代表调度决策因子，当等于 0 时，认为任务 i 在本地处理，当等于 1 时，任务 i 执行调度处理；q_{ij} 代表边缘服务器分配因子，当等于 0 时，任务 i 没有被分配给边缘服务器 j 进行调度，当等于 1 时，任务 i 被分配给边缘服务器 j 进行调度。$Q_i(2)$、$T_i(2)$ 代表任务 i 在边缘节点上处理的能耗、时延；m 代表边缘服务器数量[20]。

2.4.2 约束条件设置

为上述多目标函数设置的约束条件如下。

约束条件 1：细粒度任务调度时延 y_2 小于任务最迟截止处理时间（最大时延限值）。

约束条件 2：边缘服务器性能约束，主要指的是 1.4 部分的静态指标。

约束条件 3：任务只能调度到一个边缘服务器处理，即细粒度任务调度只能在一个边缘服务器 j 上完成。

3 混合蚁群算法完成细粒度任务调度实现

虽然基础蚁群算法具有并行计算、扩展能力好和较好全局搜索能力等优点，但当达到一定迭代次数后，会出现解趋于一致现象，使得基础蚁群算法呈现收敛速度慢和容易陷入局部最优解的缺点。本文提出的混合蚁群算法采用了遗传算法对数据特征进行预选择，一定程度上克服了基础蚁群算法的上述缺点，可以进一步解决该问题，提高基础蚁群算法的性能，对其初始解的选择方法和信息素更新策略进行改进。

混合蚁群算法是在原有蚁群算法的基础上结合遗传算法，以弥补基础蚁群算法存在的缺陷[21]。本文先利用遗传算法求出多目标函数的初始解，并将其转换为蚁群算法的初始路径信息素分布，最后再执行蚁群算法寻优程序，完成进行任务调度方案求解[22]。针对前面章节描述的目标函数，设置相应的约束条件后，将遗传算法与基础蚁群算法相结合构成混合蚁群算法，求出最优解，即任务调度方案。具体过程如图 5（a）和图 5（b）所示：

针对于图 5（a），可以按照下面的步骤与文字进行详细描述。

步骤 1：初始化蚁群算法，并设置相关参数；

步骤 2：对边缘计算细粒度任务调度问题进行编码处理；

步骤 3：定义适应度函数；

步骤 4：初始化种群；

步骤 5：种群个体解码；

步骤 6：计算种群个体的适应度函数值，筛选出优良父代个体；

步骤 7：执行遗传算法的操作，得到新一代群体；

步骤 8：将新一代群体与父代个体进行优劣替换；

步骤 9：得到最终子代个体（多目标函数的初始解）；

步骤 10：判断当前的进化代数是否位于最小和最大进化代数之间，且连续进化停滞代数的进化率大于等于最小进化率，若满足上述条件，则进入下一步；否则返回步骤 6；

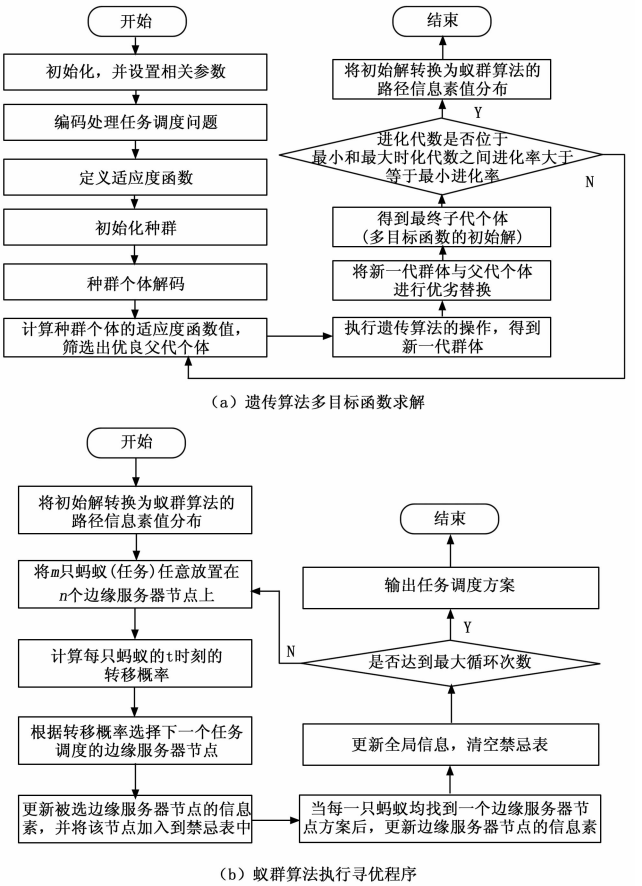

图 5 混合蚁群优化算法求解任务调度流程

步骤 11：将初始解转换为蚁群算法的路径信息素值分布；

至此完成第一部分的多目标函数求解。为了避免使用基础蚁群算法陷入局部最优，因此引入蚁群算法执行边缘计算细粒度任务调度寻优程序，具体流程如图 5（b），可以按照下面的步骤与文字进行详细描述。

步骤 1：将 m 只蚂蚁（任务）任意放置在 n 个边缘服务器节点上；

步骤 2：计算每只蚂蚁的 t 时刻的转移概率；

步骤 3：根据转移概率选择下一个边缘计算细粒度任务调度的边缘服务器节点；

步骤 4：更新被选边缘服务器节点的信息素，并将该节点加入到禁忌表中；

步骤 5：当每一只蚂蚁均找到一个边缘服务器节点方案后，更新边缘服务器节点的信息素；

步骤 6：更新全局信息，清空禁忌表；

步骤 7：判断是否达到最大循环次数[23]，若是，输出边缘计算细粒度任务调度方案；否则，回到步骤 1，直至达到最大循环次数。至此完成基于混合蚁群优化的边缘计算细粒度任务调度。

4 仿真测试与分析

4.1 仿真测试参数设置

在 Intel Corei7 CPU @ 2.80 GHz，NVIDIA Ge G Force

GTX1050Ti 和 8GB RAM 配置的工作站中进行调度仿真测试。

目标工作站在图 2 的调度模型中完成细粒度任务调度，此次测试的应用场景为某地区的交通视频监控。使用边缘计算可以让服务器能够在网络边缘完成对视频流的采集、压缩、存储、检测、显示以及最终的控制等整个监控流程，同时可以解决核心网压力过大无法对其及时转发而造成不连续等问题。

考虑到移动性是边缘计算服务器的固有属性。当用户在小区间切换时，可能会导致服务器的切换，而且不同服务器的属性与配置也存在差异，因此本文通过边缘计算系统与归属位置寄存器的配合实现移动性管理。

具体测试参数如表 1 所示。

表 1　仿真试验参数表

名称	参数
仿真区域大小	5 000 m×5 000 m
边缘服务器数量	10 个
基站数量	10 个
基站覆盖半径	100 m
细粒度任务数量	100 个
任务达到率	0.5 个/s
边缘服务器 CPU 频率	20 GHz
边缘服务器的计算能力	5 GHz/周期
时隙长度	0.5 s
信道带宽	5 MHz
信道数目	10
任务数据量	10～100 MB
最大时延限值	12 s
传输功率	0.8 W

4.2　细粒度任务优先指数计算

按照公式（1）计算 100 个细粒度任务的优先指数，并组成任务调度队列。其中，前 20 个任务的优先指数如表 2 所示。

4.3　混合蚁群算法参数设置

在表 1 所示的参数下，利用文章提出的混合蚁群算法对 1 000 个细粒度任务进行调度，测试其收敛性能。该测试通过适应度函数最优值体现，适应度函数越快达到最低点，表明算法性能更好，测试结果如图 6 所示。

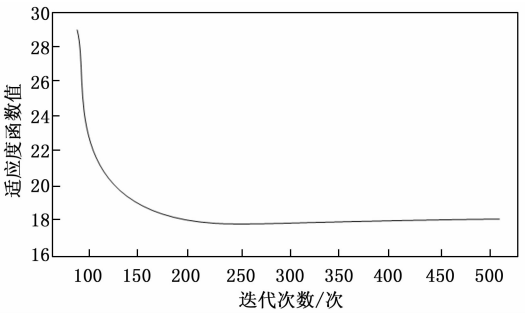

图 6　算法收敛速度测试

由图 6 可知，本文方法适应度函数值在迭代次数为 200 次时达到最低点，即此时本文方法实现收敛。表明本文方法细粒度任务调度能力较强，能够有效提高细粒度任务调度的可靠性。此时混合蚁群算法的参数情况如表 3 所示。

表 2　前 20 个细粒度任务优先指数表

任务序号	优先指数	队列位置
12	85.512	1
8	82.450	2
30	80.547	3
48	78.920	4
22	78.214	5
7	77.625	6
5	76.871	7
68	75.542	8
74	75.471	9
80	74.443	10
42	74.125	11
35	74.014	12
88	73.921	13
87	73.912	14
92	73.017	15
4	73.014	16
70	73.012	17
73	72.987	18
6	73.982	19
18	79.974	20

表 3　混合蚁群算法参数表

算法	名称	数值
蚁群算法	蚂蚁数量	20
	初始信息素	100
	信息素增加量	10
	信息素挥发系数	0.50
	信息素影响因子	1.50
	启发函数影响因子	3.5
遗传算法	转移概率	0.52
	交叉概率	0.55
	变异概率	0.20
其他	最小进化代数	40
	最大进化代数	200
	最小进化率	0.10%

4.4　结果分析

4.4.1　能耗和时延分析

相同仿真测试条件下，利用基于改进 NSGA-Ⅱ的调度方法、基于自适应遗传算法的调度方法、基于 DRL 的调度方法、基于烟花模型的调度方法求解调度方案，并与本文的混合蚁群优化算法做比较，然后同时模拟运行 5 个任务调度方案，将 100 个任务调度给 10 个边缘服务器节点，统计其调度的能耗以及时延。结果如图 7 和图 8 所示。

由图 7 可知，在同一基站带宽下，本文方法的任务调

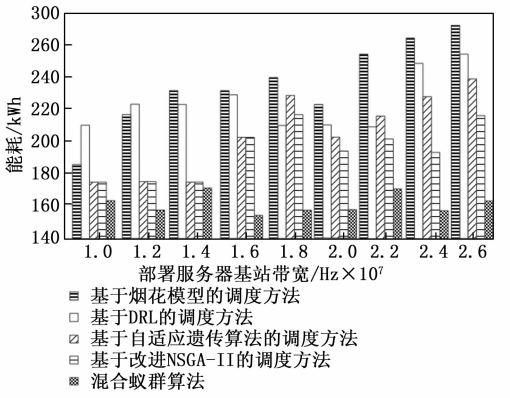

图 7 任务调度的能耗

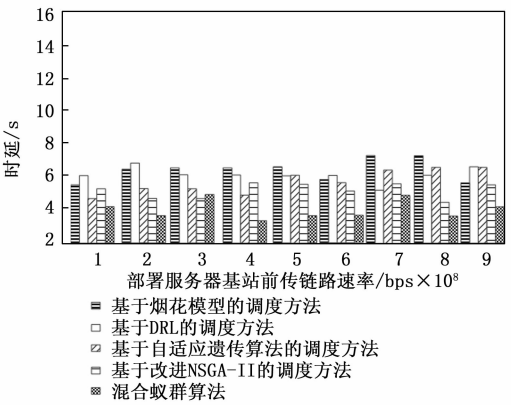

图 8 任务调度的时延

度能耗更低，最高调度能耗为基站带宽 1.4 kHz 时的 150 kWh，本文方法调度能耗更低的原因是在调度过程中利用混合蚁群算法根据转移概率选择下一个任务调度的边缘服务器节点，减少了选择带宽时干扰因素的影响。

由图 8 可知，在前传链路速率一致时，本文方法的细粒度任务调度时延更小，最高时延为 5 s，其主要原因是本文方法利用遗传算法优化了蚁群算法，提高了算法的调度性能，避免陷入局部最优。因此，综合图 7 和图 8 中可以看出，与其他 4 种调度方法相比，混合蚁群任务调度能耗和时延均要更小，这也说明本文的调度方法表现更好，所求出的调度方案更为合理。

4.4.2 用户碰撞概率测试

为了进一步验证本文方法的有效性，以不同数量的竞争用户碰撞的概率为测试指标，测试五种方法的冲突概率，如图 9 所示。

由图 9 可知，五种方法的冲突概率随着用户数量的增加而增加，其主要原因是随着用户数量的增加，用户选择同样的子载波的概率变得更高，因此提高了用户冲突概率。而本文方法在五种方法中，用户冲突概率更低，最高用户冲突概率为 0.27%，本文方法用户冲突概率较低的原因是在求解目标函数时，设定了细粒度任务只能调度到一个边缘服务器上处理的约束条件，降低了用户数量增加对边缘

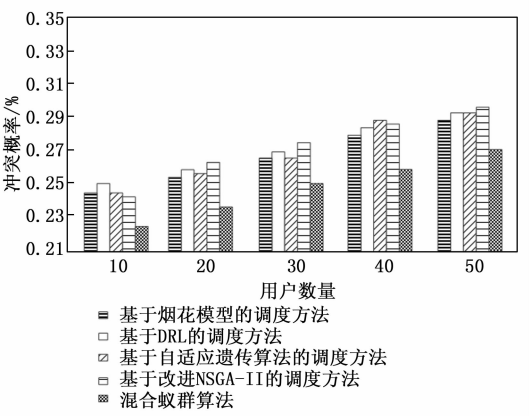

图 9 用户碰撞概率测试

服务器带宽造成的影响，减少了用户冲突。同时，当发生较高概率的用户碰撞时，可以增加子信道带宽来减少多用户之间的碰撞概率。通过增加子信道带宽，为任务调度带来了足够大的竞争空间，可以为用户提供更多的选择机会，去选择不同的竞争子载波，从而提供更好的完成细粒度任务调度。

5 结束语

边缘计算支持在网络边缘提供低延迟服务，以缓解中心服务器的压力。然而，边缘服务器上计算资源的有限容量给调度应用程序任务带来了巨大挑战。针对上述问题，本文将遗传算法与蚁群算法相结合，通过混合算法实现边缘计算细粒度任务调度。仿真测试结果证明了其有效性，求出的调度方案更为合理。下一步研究中，将针对不同类型的任务调度、任务卸载等问题进行分析，以提高本文方法的综合性能。

参考文献：

[1] AMER D A, ATTIYA G, ZEIDAN I, et al. Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing [J]. The Journal of Supercomputing, 2021, 78 (3): 2793 - 2818.

[2] SMYS S, RANGANATHAN G. Performance Evaluation of Game Theory Based Efficient Task Scheduling For Edge Computing [J]. Journal of ISMAC, 2020, 2 (1): 50 - 61.

[3] QIAN Y, SHI L, LI J, et al. An Edge-Computing Paradigm for Internet of Things over Power Line Communication Networks [J]. IEEE Network, 2020, 34 (2): 262 - 269.

[4] 崔玉亚，张德干，张 婷，等. 一种面向移动边缘计算的多用户细粒度任务卸载调度方法 [J]. 电子学报，2021，49 (11)：2202 - 2207.

[5] 闫 伟，申 滨，刘笑笑. 基于自适应遗传算法的 MEC 任务卸载及资源分配 [J]. 电子技术应用，2020，46 (8)：95 - 100.

[6] SHENG S, CHEN P, CHEN Z, et al. Deep Reinforcement Learning-Based Task Scheduling in IoT Edge Computing [J]. Sensors, 2021, 21 (5): 1666.

[7] ZHOU E, ZHANG J, DAI K. Research on Task and Resource

Matching Mechanism in the Edge Computing Network [J]. International Core Journal of Engineering, 2020, 6 (4): 94-104.

[8] 刘　明，龚　伟. 基于联合决策模型的物联网边缘计算资源分配 [J]. 计算机仿真，2021，38 (12)：299-303.

[9] NIE L, WANG X, SUN W, et al. Imitation-Learning-Enabled Vehicular Edge Computing: Toward Online Task Scheduling [J]. IEEE Network, 2021, 35 (3): 102-108.

[10] WANG Y, RU Z Y, WANG K, et al. Joint Deployment and Task Scheduling Optimization for Large-Scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing [J]. IEEE Transactions on Cybernetics, 2020, 50 (9): 3984-3997.

[11] JIN H, WU W, SHI X, et al. TurboDL: Improving CNN Training on GPU with Fine-grained Multi-streaming Scheduling [J]. IEEE Transactions on Computers, 2020, 70 (4): 552-565.

[12] JIN H, WU W, SHI X, et al. TurboDL: Improving CNN Training on GPU with Fine-grained Multi-streaming Scheduling [J]. IEEE Transactions on Computers, 2020, 70 (4): 552-565.

[13] MIROBI G J, AROCKIAM L. DAVmS: Distance Aware Virtual Machine Scheduling approach for reducing the response time in cloud computing [J]. The Journal of Supercomputing, 2021, 77 (6): 6664-6675.

[14] DHAL K, RAI S C, PATTNAIK P K, et al. CEMAR: a fine grained access control with revocation mechanism for centralized multi-authority cloud storage [J]. The Journal of Supercomputing, 2021, 78 (2): 987-1009.

[15] MEMARI P, MOHAMMADI S S, JOLAI F, et al. A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture [J]. The Journal of Supercomputing, 2021, 78 (1): 93-122.

[16] SHERMIN T, TENG S W, SOHEL F, et al. Integrated Generalized Zero-Shot Learning for Fine-Grained Classification [J]. Pattern Recognition, 2021, 122 (9): 108246.

[17] TANG H, LI C, ZHANG Y, et al. Optimal multilevel media stream caching in cloud-edge environment [J]. The Journal of Supercomputing, 2021, 77 (10): 10357-10376.

[18] EGASHIRA S, WANG Y, TANAKA K. Fine-Grained Cryptography Revisited [J]. Journal of Cryptology, 2021, 34 (3): 1-43.

[19] MA Y, LU C, SINOPOLI B, et al. Exploring Edge Computing for Multi-Tier Industrial Control [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39 (11): 3506-3518.

[20] BASLAMISLI A S, DAS P, LE H A, et al. ShadingNet: Image Intrinsics by Fine-Grained Shading Decomposition [J]. International Journal of Computer Vision, 2021, 129 (8): 2445-2473.

[21] NAIK B B, SINGH D, SAMADDAR A B. FHCS: Hybridized Optimization for Virtual Machine migration and task scheduling in cloud data center [J]. IET Communications, 2020, 14 (12): 1942-1948.

[22] WANG K, ZHOU Y, LIU Z, et al. Online Task Scheduling and Resource Allocation for Intelligent NOMA-Based Industrial Internet of Things [J]. IEEE Journal on Selected Areas in Communications, 2020, 38 (5): 803-815.

[23] SAKOWSKY R A. Disentangling the welfarism/extra-welfarism distinction: Towards a more fine-grained categorization [J]. Health Economics, 2021, 30 (9): 2307-2311.

(上接第 232 页)

[8] CHU P, VU H, YEO D, et al. Robot reinforcement leaning for automatically avoiding a dynamic obstacle in a virtual environment [J]. Lecture Notes in Electrical Engineering, 2015, 352: 157-164.

[9] 刘志荣，姜树海，袁雯雯，等. 基于深度 Q 学习的移动机器人路径规划 [J]. 测控技术，2019，38 (7)：24-28.

[10] MNIH V, KAVUKEUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529.

[11] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized Experience Replay [J/OL]. Computer Science, 2015. [2022-09-11]. https://doi.org/10.48550/arXiv.1511.05952.

[12] XIN J, ZHAO H, LIU D, et al. Application of deep reinforcement learning in mobile robot path planning [C] //Proc of Chinese Automation Congress. Piscataway, NJ: IEEE Press, 2017: 7112-7116.

[13] KHATIB O. Real-Time Obstacle Avoidance System for Manipulators and Mobile Robots [J]. The International Journal of Robotics Research, 1986, 5 (1): 90-98.

[14] 孙传禹，张　雷，辛　山，等. 结合 APF 和改进 DDQN 的动态环境机器人路径规划方法 [J/OL]. 小型微型计算机系统，2022：1-8. [2022-09-11]. https://kns.cnki.net/kcms/detail/21.1106.TP.20220509.1303.012.html.

[15] WATHINS C J C H, DAYAN P. Q-learning [J]. Machine Learning, 1992, 8 (3-4): 279-292.

[16] 周飞燕，金林鹏，董　军. 卷积神经网络研究综述 [J]. 计算机学报，2017，40 (6)：23.

[17] LECUN Y, BOTTOU L. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86 (11): 2278-2324.

[18] 闫皎洁，张锲石，胡希平. 基于强化学习的路径规划技术综述 [J]. 计算机工程，2021，47 (10)：10.

[19] DAS P K, BEHERA H S, PANIGRAHI B K. Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity [J]. Engineering Science and Technology, an International Journal, 2016, 19 (1): 651-669.

[20] 徐晓苏，袁　杰. 基于改进强化学习的移动机器人路径规划方法 [J]. 中国惯性技术学报，2019，27 (3)：7.

[21] 于红斌，李孝安. 基于栅格法的机器人快速路径规划 [J]. 微电子学与计算机，2005，22 (6)：3.