基于动态势博弈的边缘算力网络任务调度算法

张 晶,关建峰*,刘科显,申 奥
(北京邮电大学网络与交换技术全国重点实验室,北京 100876)

摘 要: 随着个性化、多样化的新型网络应用和业务的不断发展和成熟,数据量和计算需求呈指数型增长趋势,而云计算、边缘计算、智能终端设备等也得到了快速发展,计算资源呈现出泛在、分散部署的趋势,如何高效协同地利用这些泛在计算资源以满足日益增长的计算需求,成为当前网络领域研究的一项重要新课题. 边缘算力网络集中在网络边缘,在靠近数据源的位置,将异构的计算资源和网络资源结合起来,通过资源感知、服务定位、任务调度等来提高资源利用率和任务执行效率,在保持低延迟和低成本的同时,实现对分布式计算资源的最优配置. 边缘算力网络通常采用分布式的任务调度方式,各节点基于局部范围内的信息进行本地决策,具有决策时间短、能有效缓解中心控制器计算和通信压力等优势. 然而,信息的局部、不对称特征限制了分布式任务调度的全局优化性能,导致计算任务的覆盖率无法得到保障. 本文以边缘算力网络分布式任务调度为核心,依托博弈理论及多目标优化方法,设计基于最佳动态响应的分布式任务调度算法,引入两跳范围内的通信和共识消除机制,在最小化交互开销和决策延迟的情况下,最大限度地提升了分布式任务调度的任务覆盖率,实现向纳什均衡点的收敛;将两跳范围内的共识消除作为优化目标之一,建立基于分布式决策优化性和一致性双目标的动态势博弈模型,通过理论推导证明了局部决策和全局决策的渐进等价性,为纳什均衡的存在性及分布式任务调度的收敛性提供了有效的理论依据;最后,通过仿真与经典分布式决策算法和全局最优解进行了对比,验证了所提出算法的有效性和优化收益.

关键词: 边缘算力网络;任务调度;拍卖算法;动态势博弈;分布式决策
基金项目: 国家自然科学基金(No.62394323)
中图分类号: TP393.1 **文献标识码:** A **文章编号:** 0372-2112(2025)01-0221-17
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240401

Task Scheduling Algorithm Based on Dynamic Potential Game for Edge Compute First Networking

ZHANG Jing, GUAN Jian-feng*, LIU Ke-xian, SHEN Ao
(*State Key Laboratory of Networking and Switching Technology*, *Beijing University of Posts and Telecommunications*, *Beijing* 100876, *China*)

Abstract: With the continuous development and maturity of personalized and diversified new network applications and services, the amount of data and computing demands are experiencing an exponential growth trend. Cloud computing, edge computing and intelligent terminal devices have been developed rapidly, and computing resources have shown a trend of ubiquitous and decentralized deployment. How to use these ubiquitous computing resources efficiently and collaboratively to meet the increasing computational demands has become an important new topic in the current network field. Edge compute first network focus on the edge of the network, near the location of the data source, combining heterogeneous computing resources and network resources to improve resource utilization and task execution efficiency through resource awareness, service positioning, task scheduling, maintaining low latency and low cost and realizing the optimal configuration of distributed computing resources at the same time. Edge compute first network usually adopts the distributed task scheduling mode. In distributed task scheduling, each node makes local decision based on local information, which has the advantages of short decision time and effective relief of calculation and communication pressure of central controller. However, the local and asymmetric nature of information limits the global optimization performance of distributed scheduling, resulting in an inadequate task coverage. This paper focuses on distributed task scheduling in edge compute first network. With the support of game theory and multi-objective optimization methods, a distributed task scheduling algorithm based on optimal dy-

收稿日期:2024-05-06;修回日期:2024-09-26;责任编辑:朱梅玉
*通讯作者:关建峰

namic response is designed, which introduces communication and consensus elimination mechanisms within a two-hop range. Under the conditions of minimizing interaction costs and scheduling delays, it maximizes the task coverage of distributed scheduling and achieves convergence to the Nash equilibrium point. A dynamic game model based on the optimization and consistency of distributed decision-making, with consensus elimination within a two-hop range as one of the optimization objectives, is established. The theoretical derivation demonstrates the asymptotic equivalence between local decisions and global decisions, providing an effective theoretical basis for the existence of Nash equilibria and the convergence of distributed scheduling. Finally, the effectiveness and optimization benefits of the proposed algorithm are validated through simulations and comparisons with a classical distributed decision-making algorithm and global optimal solutions.

Key words: edge compute first networking; task scheduling; auction algorithm; dynamic potential game; distributed decision making

Foundation Item(s): National Natural Science Foundation of China (No.62394323)

1 引言

随着无人驾驶、元宇宙等未来新业务的出现,以及人工智能、大数据等技术的不断发展,数据量和计算需求呈指数级增长,算力已成为一种关键生产要素. 然而,在计算时延、带宽成本、可扩展性等算力处理方面,传统集中式数据中心无法满足新应用、新技术等对泛在异质算力资源的高效协同利用需求[1]. 聚焦“计算”和“网络”,通过边缘算力网络将动态分布的计算与存储资源互联,使海量的应用能够按需、实时调用泛在分布的计算资源,提升各类算力资源的整体运行效率、系统能耗和服务能力,已成为新型网络基础设施演进发展的重要方向[2,3].

算力任务调度是边缘算力网络动态互联和高效服务的关键环节. 随着计算设备呈现日益增多且泛在、分散部署的特征,海量计算设备的高效调度成为困扰算力网络发展的核心问题之一. 由于在应急救援、智慧工厂、空间和水下探测等领域的广泛应用,多智能体协同早已成为一个活跃的研究领域. 多智能体系统具有许多单代理系统所没有的特性,因此它更能够以有效、高效的方式完成困难和复杂的任务,而且还更能容错[4]. 多智能体协同调度通常是指在无人机、机器人等智能体和打击、探测等任务之间找到一对一或一对多的配对,同时最大限度地提升配对带来的收益[5]. 而边缘算力网络中的算力调度问题是指在最小开销的情况下找到计算资源和计算任务之间的一对一或一对多的配对. 因此,边缘算力网络中海量设备的协同调度问题与无人机蜂群、多机器人等多智能体任务调度问题本质上具有相同的模型特征,进而可以借鉴现有方法开拓解决新型算网中的计算任务调度问题.

作为组合优化问题的重要组成部分,许多技术被开发以解决上述任务调度问题[6],包括一些启发式算法被用来提升优化速度[7]. 为了解决同时具有多个优化目标的调度问题,快速非支配排序遗传算法(Non-dominated Sorting Genetic Algorithm,NSGA)等多目标的优化算法也被提出[8]. 然而,以上所有都假设全网信息的可用性,或以集中式方式规划. 在现实条件下,全网信息的收集、全局决策,会给集中控制器带来巨大的通信和计算压力. 在未来网络万物智联趋势下,海量、泛在计算设备的集中调度面临巨大的挑战.

相比于集中式调度模式,分布式任务调度模式能够克服上述限制. 基于共识的拍卖算法CBAA(Consensus-Based Auction Algorithm)、分布式应用部署算法EdgeDecAp等均是基于分布式调度模式设计的算法,成功用于解决分布式的任务调度和部署等问题[9,10]. 分布式任务调度模式下任务调度算法在每个智能体上运行,每个智能体仅收集局部范围内的节点信息和任务信息,完成本节点局部范围内的任务调度,并通过直接连通的节点间的信息交互来降低调度结果的冲突和资源的浪费. 由于不需要获取全局信息,分布式调度模式更加适用于未来网络海量设备的高效调度问题,可扩展性更强. 然而,局部信息约束下各智能体对于计算资源、任务等的认知存在偏差,导致资源和任务闲置,进而可能导致计算任务的覆盖率受到限制,设备空闲、利用率低的情况频繁发生. 传统的三次握手、媒介共享等方式仅能够避免任务的冲突问题,但无法进一步提升任务的覆盖率. 例如,CBAA算法中一跳邻居之间的交互过程、EdgeDecAp算法中借助拍卖方的握手过程等仅保障了任意2个代理之间不存在任务冲突,但仍存在大量的闲置任务和闲置资源没有得到充分的利用和覆盖.

如图1所示,红色星形表示计算资源,蓝色圆形表示计算任务,虚线表示资源和任务之间的连通关系,虚线上的数字是计算资源完成对应任务的代价值. 资源节点同时也是任务调度的决策者. 在各资源节点仅获知直接连通的资源和任务信息的情况下,C_2节点优先选择具有最小代价值5的任务T_2,C_0则选择任务T_1. 在仅依赖于局部信息的分布式任务调度中,由于C_1不知道T_0的存在,导致二者均处于闲置状态.

据此,本文以未来万物智联趋势下的边缘算网任

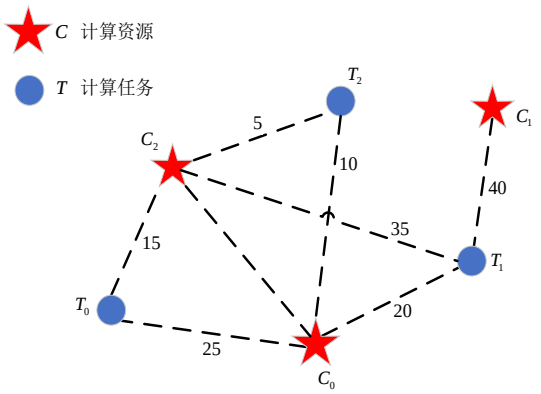


图1　基于随机拓扑的闲置资源问题示例图

务调度问题作为出发点,聚焦于解决分布式调度模式下任务覆盖率的优化提升问题,以实现海量泛在异构算力的高效调度.具体研究内容和贡献如下:

(1)搭建了融合转发、存储、计算等多维资源和需求的边缘算网任务调度系统模型,建立考虑任务覆盖率、实时性以及节点资源限制的多目标优化任务调度模型.

(2)设计了基于最佳动态响应的分布式共识调度算法(关键词为 Consensus-Based 和 Game Theory,简称 CBGT 算法),算法以最优动态响应策略为基础,融合了多目标优化方法.CBGT 通过一跳范围内的任务调度以及两跳邻居节点之间的交互和博弈,实现了分布式任务调度,避免了任务的调度冲突,降低了资源的浪费.此外,设计了智能体之间的任务调度和交互流程、调度结果表项以及表项更新规则.

(3)面向分布式决策的收敛性需求,将本文所考虑问题抽象为博弈模型,建立符合动态势博弈模型(Dynamic Potential Game,DPG)的局部效益函数和全局势函数,借助势博弈理论的收敛特性,从理论上证明了所设计 CBGT 算法的收敛性能.

(4)通过仿真证明了 CBGT 算法比 CBAA、EdgeDecAp 等经典算法更接近全局最优结果,具有更高的任务覆盖率和更低的资源浪费,并且对链路故障具有鲁棒性.

2　相关工作

边缘算力网络作为新型网络架构,目前与其直接相关的任务调度研究成果相对较少,但多智能体、边缘计算、微电网等领域均有相关的研究成果,可为本文的研究提供参考.从调度架构角度可以将现有研究划分为自主指派、第三方集中调度、分布式调度 3 种模式.分布式调度模式下每个代理拥有的任务感知信息不同,需要通过漫长的一致性过程才能够达成任务信息的共识.在任务信息不一致的情况下,也可以通过决策结果的一致性过程保障调度结果的共识.因此,根据共识过程所处阶段的不同,可以将分布式调度模式进一步划分为基于信息共识的分布式调度算法和基于局部信息的分布式自由交易.

自主指派是当某节点本地计算任务量过大时,由该节点将本地计算任务进行划分并指派给不同的邻居节点执行,最终返回给本节点.文献[11]针对大规模机器学习的分布式计算调度进行分析研究,主要考虑由单个主服务器进行集中式统筹调度的模式,采用非编码计算针对存在非持续掉队者的情况进行设计优化.文献[12]面向绿色碳中和场景需求探索绿色计算网络任务调度策略,将碳排放强度作为调度的因素之一,考虑一个边缘服务器连接多个云服务器的场景,从理论上建立优化模型并求解.自主指派模式下各节点之间没有协商过程,容易导致冲突和拥塞.

第三方集中调度是所有用户将自身的计算需求发送给一个第三方调度器,所有计算节点也将各自的计算资源情况发送给第三方调度器,后者根据供需双方的情况进行统一规划和指派.文献[13]采用一个第三方调度器进行任务调度,各用户将计算需求上报给第三方调度器,其根据供需双方的状态进行任务规划和指派,考虑了分散计算过程中的通信时间,设计更加符合现实条件的调度策略.文献[14]设计了一种高效的移动云计算一对一资源交易机制,保证了买卖双方较高的信任度,需要一个集中的拍卖商.组合拍卖可以满足不同类型资源的不同需求,同时给出满足多项式时间计算效率的解.文献[15]和文献[16]分别利用组合双拍卖和基于契约理论的拍卖,考虑了计算节点的覆盖范围约束,但需要一个集中式、接收全局信息的拍卖商.集中式的任务调度模式难以适应网络规模的日益增大,面临决策时间长、通信开销大、容量受限等难题.

基于信息共识的分布式调度算法是在决策之前先进行全网信息的交互,使得每个节点都获得全网信息,之后再基于一致的全网信息各自完成独立的决策.分布式版本的匈牙利方法就是一种基于信息共识的分布式调度算法,在本地计算和通信资源有限的点对点网络中,所有的机器人都协同计算一个共同的调度,以优化给定的全局目标,如行驶的总距离等[5].文献[17]也尝试过基于共识的方法,要求机器人在执行任务之前收敛于一致的任务感知信息.这种算法相当于集中式调度的分布式版本,仍然需要全局信息的交互,各节点的计算开销和交互开销较大,对各节点的本地资源带来较大挑战.

相比于基于信息共识的分布式调度,分布式自由交易模式下不需要在决策之前进行全网信息的交互,而是先基于局部信息进行决策,再基于决策的结果完成一致性共识过程.分布式自由交易模式下用户将计算需求发布给局部范围内的多个计算节点,各计算节点根据自身资源情况对接收到的任务进行竞拍,同时

与邻居节点间进行协调，通过一个高效的分布式机制完成任务的调度，再将竞拍结果返回给成功竞拍的用户[18]. 这种调度方式能够克服上述限制，任务调度算法在每个智能体上运行，仅考虑局部范围内的任务，再通过局部的信息交互即可得到解决方案，极大降低了调度开销和时间成本.

多机器人、无人机群等多智能体系统具有高度自治、动态性强等特征，对灵活、自治的任务调度机制具有较高需求，率先在分布式自由交易模式下积累了丰硕的研究成果. 基于市场的多代理协调方法首先受到广泛的关注，并在多机器人研究界越来越受欢迎[19]. 拍卖是一种常用的基于市场的任务调度方法. 该过程包括几轮投标，代理对每个任务进行投标，其中一个任务的投标价值等于代理访问该任务的估计成本，代理获胜并被分配那些出价低于任何其他代理的任务. 作为多智能体协同领域的经典算法，基于迭代的分布式拍卖算法 CBAA 被提出[9]. CBAA 算法中每个节点通过一跳范围内的竞价和共识交互来获得任务，每个节点只考虑自己的任务，再通过共识消除机制有效避免了冲突调度，充分保障了计算和交互的轻量化. 文献[20]在 CBAA 的基础上考虑了每个代理的任务序列. 文献[21]为每个任务定义了一个新的重要性概念，并通过对车辆产生的本地成本的贡献来衡量. 文献[22]则添加了 2 个易于与文献[21]中的 PI 算法集成的附加模块. 第一种扩展了算法，允许实时动态在线重新调度，第二种通过引入额外的 softmax 动作选择程序来提高算法的探索属性，进而提高性能. 文献[23,24]结果表明，分布式贪婪算法可以异步收敛，且收敛于有限步数. 文献[25]提出了一种协同神经动力学优化方法来解决任务调度问题. 但上述研究都没有考虑其他节点的信息来进行全局优化，这与理想的优化模型有很大的不同. 这种机制具有计算和通信效率，但不可避免地导致解决方案高度次优，因为其不考虑代理之间的任何协同效应. 为了实现面向任务的全局优化，文献[26]将任务的意义作为全局传递参数不断传递，但节点间的协同作用没有得到充分利用. 此外，虽然这些方法是有效的，但它们通常收敛速度很慢，并且需要传输大量的数据.

边缘计算等领域也在网络和业务规模日益增长的趋势下引入了分布式自由交易模式，以保障系统的可扩展性需求. 作为算力网络的先驱技术之一，边缘计算和云计算场景中对利用拍卖、契约理论等方法设计资源交易方案进行了广泛的研究. 文献[27]在移动云计算中提出了一种分布式诚实拍卖机制，在一定程度上防止买家的不真实信息，但每个节点仅考虑自己的资源情况进行决策，可能导致计算任务的覆盖率受限. 文献[28]针对物联网场景下的无服务器边缘计算设计了分布式的任务调度方法，引入了多智能体深度强化学习模型. 深度学习等新型算法的引入为分布式调度提供了不同维度的解决方案，但又不同程度地带来了新的计算压力. 未来万物智联场景下海量泛在设备通常是异构的，其续航能力各有不同，无法充分保障，需要设计绿色、节能的调度算法，以最小的通信和调度计算开销，来获得更大的任务响应率. Beraldi 等人[29]提出了一个分布式机制来选择最佳雾节点，该机制是受到博弈论启发的. 同样利用博弈论，Chen 等人[30]定义了一个针对密集小型细胞网络的分布式算法，该算法迭代地合作进行任务卸载决策. Smolka 等人[10]定义了一个分散式算法 EdgeDecAp，该算法基于拍卖来评估应用程序执行时间，用于实现网络应用的部署. 此外，分布式任务卸载也可以利用设备到设备通信在远端边缘进行[31,32]，其资源共享可能基于区块链并服务于不同的用例，如工业物联网.

从上述相关工作可以看出，在未来网络规模、业务量激增的趋势下，分布式且局部的自由交易模式更加符合任务快速响应需求，但如何在信息受限、通信受限的情况下保障尽可能覆盖更多的任务，尚未得到充分的研究和保障，这正是本文致力于解决的问题.

3 问题描述

本节首先搭建边缘算力调度场景模型，如图 2 所示. 算力调度需要明确调度者、资源的提供者以及资源请求者，图中统一用服务器标志表示算力服务的提供者 S，以用户标志表示服务的请求者 R.

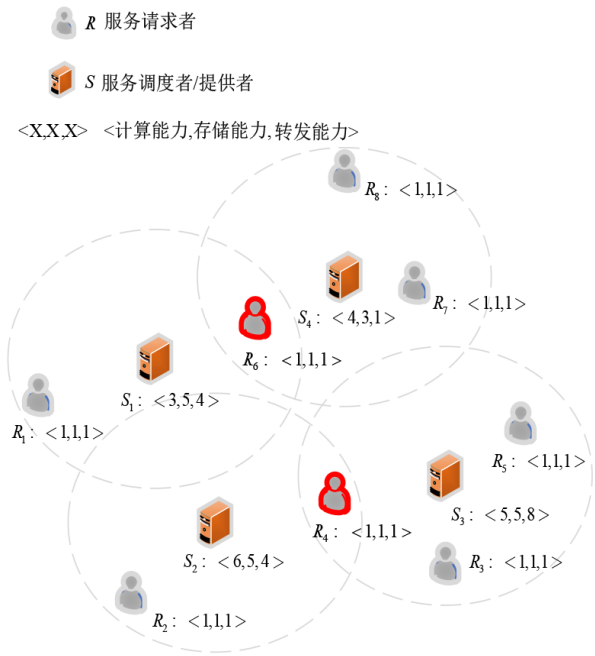


图 2 边缘算力任务调度场景模型

为了减少额外的交互开销,本文中算力提供者同时也是算力的调度者,算力提供者S在接收到请求者R发来的请求消息后,直接在本地完成资源的调度,而不需要浪费额外的交互成本把请求信息汇报给单独的第三方调度器. 网元设备计算能力的不断提升,为这种调度计算和服务计算集成于同一节点完成的模式提供了充分的硬件支撑. 对于算力服务的提供者S,$<X,X,X>$表示其能够提供的计算、存储和网络转发能力;而对于算力服务的请求者R,$<X,X,X>$则表示其对各维度资源的需求. 在无线网络中,图中虚线部分表示算力服务提供者S的一跳通信范围,而在有线网络中,虚线部分表示S的所有直接连通设备.

在未来,算力网络的算力多样泛在、算网服务极简一体化趋势下,调度者可以为云服务器、边缘服务器、核心网路由器、基站等多种具有计算能力的网元节点,通过调度能力的泛在化,可以缓解集中式控制中心的调度压力,大幅提升闲散算力资源的利用率和任务响应时效性. 同时,在数据驱动网络、知识中心网络等新型网络架构和技术的发展趋势下,网元节点自身的功能也需要人工智能、大数据等技术的加持,进而需要大算力的支撑. 由此,网元节点也可能成为算力的请求者. 未来算力网络将可能呈现算力泛在、调度泛在、交易泛在的趋势,而分布式、对等的自由调度和交易模式,为泛在算网提供了可扩展的、快速响应的快捷交易模式.

在分布式调度模式下,各调度者通常仅知道局部范围内的资源情况和请求情况,各调度者之间存在信息的偏差,进而可能导致隐藏提供者和隐藏请求者的出现,造成调度冲突等问题. 在真实边缘算力网络场景下,服务提供者的任务覆盖范围的取值涉及多方面因素. 文献[33]面向车联网、智慧工厂等延迟敏感、数据隐私的应用需求,文献[34]面向卫星网络的带宽压力,均将边缘服务器的任务服务范围设置为一跳,以避免更多的转发带来额外的延迟、带宽压力和敏感数据泄露等问题. 因此,本文假设调度者仅调度一跳范围内的资源和任务,即一跳通信范围同时也是一跳任务服务范围,以保障服务的实时性、数据安全以及缓解带宽压力. 以图2中红色线条特殊标识的2个用户R_4和R_6为例,二者分别处于S_2和S_3、S_1和S_4两个调度者的交叉通信范围内,在各调度者仅能够获得一跳范围内的资源和任务信息,且仅基于上述信息进行局部调度的情况下,交叉区域用户R_4的服务请求可能同时被S_2和S_3两个调度者响应,R_6的请求则可能被S_1和S_4两个调度者响应,进而造成资源的重复使用和浪费. 此外,当S_1和S_4均响应R_6时,用户R_1、R_7和R_8的服务请求则得不到响应,任务覆盖率则由此受到了限制.

据此,在上述边缘算力网络场景中,如何尽可能地在较低交互开销的情况下,通过高效的分布式算力调度,保障任务覆盖率,使得更多请求者R的计算服务请求能够得到满足,使闲置的计算资源能够得到充分的利用,是本文要解决的问题. 本文中用到的符号及其含义汇总在表1中.

本节进一步将上述问题抽象为数学模型. 首先定

表1　符号及含义

符号	含义与解释
S_i	调度者和资源提供者的标志
R_i	服务请求者的标志
$\boldsymbol{U}=[U_1,U_2,\cdots,U_n]$, $\boldsymbol{M}=[M_1,M_2,\cdots,M_n]$	自治代理及其任务能力集合,共有n个自治代理
$\boldsymbol{T}=[T_1,T_2,\cdots,T_m]$, $\boldsymbol{N}=[N_1,N_2,\cdots,N_m]$	任务及其能力需求集合,共有m个任务
C^i_{comp}, C^{ij}_{trans}, C^i_{mem}	代理U_i的计算能力、与代理U_j之间的链路传输能力以及其存储能力
N^i_{comp}, N^i_{trans}, N^i_{mem}	任务T_i的计算能力、传输能力及其存储能力需求
t_{ij}	代理i执行任务j的时间成本
x_{ij}	表示代理i与任务j之间的配对关系,当其值为1时,表示任务j被分配给代理i
t, $T_{\max}$	算法的迭代次数及其最大值
S^i_{pop}	代理i的所有备选配对集合
S^i_{cost}	S^i_{pop}中各配对的时间代价集合
S^i_{nei2hop}(或U_i^2)	代理i的所有两跳邻居集合
S^i_{cons}	代理i的所有备选配对与邻居调度结果的一致性值集合
$S^{i,t}_{\text{nei}}$(或U_i^1)	t时刻代理i一跳范围内的邻居代理集合
$S^i_{\text{nei\_ta}}$	代理i的所有一跳范围任务集合
d_{ij}	代理i和j针对代理i一跳范围内的任务的决策结果之间的汉明距离
T_{stamp}, C_{total}, C_{cons}	配对结果对应的时间戳、总时间代价以及与邻居的一致性值
H_t	各节点的任务服务范围,单位是跳数
H_c	各节点的通信和共识消除范围,单位是跳数

义一组自治代理集合 $\boldsymbol{U}=[U_1,U_2,\cdots,U_n]$和一组任务集合 $\boldsymbol{T}=[T_1,T_2,\cdots,T_m]$. 每个自治代理都有自己的任务能力,所有代理的任务能力集合用 $\boldsymbol{M}=[M_1,M_2,\cdots,M_n]$表示,其中 $M_i=[C^i_{\text{comp}},C^i_{\text{trans}},C^i_{\text{mem}}]$, 各维度能力的定义如下:(1)CPU计算能力 C^i_{comp},节点 i 在一个时隙内能够处理的指令数(如浮点运算)的最大值 $C^i_{\text{comp\_0}}$、内核数量 $C^i_{\text{comp\_1}}$ 以及内存空间大小 $C^i_{\text{comp\_2}}$;(2)链路传输能力 C^{ij}_{trans},链路(i,j)在一个时隙内能够传输的数据单元的最大数量;(3)存储能力 C^i_{mem},节点 i 能够存储的数据单元的最大数量 $C^i_{\text{mem_0}}$以及写入速度 $C^i_{\text{mem\_1}}$.

每个任务都有自己的能力需求,所有任务的能力需求集合用 $\boldsymbol{N}=[N_1,N_2,\cdots,N_m]$ 表示,其中 $N_i=[N^i_{\text{comp}},N^i_{\text{trans}},N^i_{\text{mem}}]$, 各维度需求的定义如下:(1)计算能力需求 N^i_{comp},任务 T_i 的计算指令数 $N^i_{\text{comp\_0}}$,如浮点运算次数,任务 T_i 需要的内核数量 $N^i_{\text{comp\_1}}$ 和内存空间大小 $N^i_{\text{comp\_2}}$;(2)传输能力需求 N^i_{trans},任务 T_i 的数据单元数量;(3)存储能力需求 N^i_{mem},任务 T_i 需要的存储单元数量.

接下来,给出本文所考虑的多智能体任务调度问题的形式化定义,它是一个具有多约束条件的多目标组合优化问题(MultiObjective Combinatorial Optimization Problem,MOCOP). 在最小化多目标的假设下,将MOCOP用数学表示为

$$\begin{aligned}&\min\ \boldsymbol{Y}=\boldsymbol{F}[\boldsymbol{X}]=\left[f_1(\boldsymbol{X}),f_2(\boldsymbol{X}),\cdots,f_s(\boldsymbol{X})\right],\\&\text{s.t.}\quad \boldsymbol{X}\in\Omega,\end{aligned}$$

其中,Ω 是离散决策空间,$\boldsymbol{X}$ 为 Ω 中的一个可行解,s 表示优化目标的数量,$f_i(\boldsymbol{X})$ 表示第 i 个优化目标,$\boldsymbol{F}(\boldsymbol{X})=(f_1(\boldsymbol{X}),f_2(\boldsymbol{X}),\cdots,f_s(\boldsymbol{X}))$为可行解$\boldsymbol{X}$对应的优化目标向量. 为表示方便,这里对优化目标和约束条件均进行了简化,在实际场景中可以根据具体需求进行补充扩展.

结合实际场景特征,基于上述定义形式将本文所考虑的边缘算力网络任务调度的优化目标正式地表示如下:

$$\min \boldsymbol{F}(\boldsymbol{X})=\left[f_1(\boldsymbol{X}),f_2(\boldsymbol{X}),f_3(\boldsymbol{X})\right] \tag{1}$$

$$f_1(\boldsymbol{X})=\sum_{i=1}^{n}\sum_{j=1}^{m}t_{ij}x_{ij} \tag{2}$$

$$f_2(\boldsymbol{X})=-\sum_{i=1}^{n}\sum_{j=1}^{m}N_j x_{ij} \tag{3}$$

$$f_3(\boldsymbol{X})=\sum_{i=1}^{n}\sum_{j=1}^{m}M_i x_{ij} \tag{4}$$

$$\text{s.t. } x_{ij}\in\{0,1\},\ i=1,2,\cdots,n,\ j=1,2,\cdots,m \tag{5}$$

$$\sum_{i=1}^{n}x_{ij}=1,\ j=1,2,\cdots,m \tag{6}$$

$$\sum_{j=1}^{m}N_j x_{ij}\leqslant M_i,\ \forall i=1,2,\cdots,n \tag{7}$$

总体目标是将所有任务尽可能多地分配给代理,且资源消耗和时间成本最小化,各智能体的任务不存在冲突. 为方便表示,这里简单地用 i、j 等序号指代不同的代理和任务. 用二元决策变量 x_{ij} 表示任务 j 是否被分配给代理 i. t_{ij} 表示代理 i 执行完成任务 j 所需花费的时间成本. 式(2)表示所有任务消耗的总时间之和,用于保障任务执行的时间效率. 式(3)表示所有任务覆盖量的相反数,用于保障尽可能多的任务得到响应. 考虑到隐藏资源和隐藏请求的存在,可能导致多个代理被分配给相同的任务,进而造成资源的重复浪费,因此式(4)表示执行所有任务用到的所有资源总数,用于保证相同的任务覆盖率情况下所使用的资源最少. 为简化分析,这里只考虑一对一的分配情况,即每个任务只能被分配给一个代理执行,约束条件中式(6)表示每个任务只能指派给唯一的一个代理执行,式(7)表示分配给每个代理的任务量不能超过其能力的上限.

当一个优化问题中存在多个优化目标时,会产生一组最优解,称为帕累托最优解集,而不是一个单一的最优解. 帕累托最优解集可以基于帕累托支配关系得到. 为了处理独立且冲突的目标函数,定义了不同解之间的帕累托支配关系,考虑向量 $\boldsymbol{u},\boldsymbol{v}\in\mathbf{R}^s$,$\boldsymbol{u}$ 帕累托支配 $\boldsymbol{v}$,记为 $\boldsymbol{u}>\boldsymbol{v}$,当且仅当:

$$\left\{\begin{array}{l}\forall i\in\{1,2,\cdots,s\},\\ f_i(\boldsymbol{u})\leqslant f_i(\boldsymbol{v})\end{array}\right\}\wedge\left\{\begin{array}{l}\exists j\in\{1,2,\cdots,s\},\\ f_j(\boldsymbol{u})<f_j(\boldsymbol{v})\end{array}\right\}$$

也可称为 $\boldsymbol{u}$ 占优 $\boldsymbol{v}$. 也就是说,在所有的优化目标维度下,$\boldsymbol{v}$ 都不优于 $\boldsymbol{u}$,但至少一个优化目标维度下,$\boldsymbol{u}$ 优于 $\boldsymbol{v}$. 如果在整个参数空间内不存在任何一个决策向量帕累托占优某一个决策向量,就称该决策向量是帕累托最优解. 所有帕累托最优解组成的集合称为帕累托最优解集合,也称为帕累托前沿面. 帕累托前沿面的存在使得优化问题的求解相对复杂,可以通过加权法或NSGAⅡ等多目标启发式算法进行求解. 本文中考虑的优化问题即包含不唯一的3个优化目标,可以借助帕累托最优理论及NSGAⅡ等算法求解.

4 基于最佳动态响应的共识调度机制

博弈论(Game Theory,GT)是研究自私节点如何相互作用和合作的一种自然而强大的工具. 它可以被认为是决策理论的一种推广,包括多个参与者或决策代理[35,36]. 除了经济学和政治科学之外,在过去的10年里,GT被应用于控制、信号处理和无线通信等领域,尤其涉及到网络方面的问题[37]. Cui等人[38]介绍了一种用于任务分配的博弈论方法. 与CBAA一样,任务分配的过程被分为2个阶段. 采用契约网络协议进行初始任务分配,然后采用博弈论方法重新分配任务以满足帕累托最优性[39,40]. 文献[41]考虑了基于GT的多代理系

统的分布式资源分配. 在多智能体系统的任务分配活动中,并不总是清楚“优化一个智能体的收益”是什么含义,因为每个代理产生的利润都取决于其他代理人的策略和选择. 因此,在博弈论中,最佳动态响应是每个参与者不断寻找当前对手策略下的最佳行动来进行策略选择.

本文以最佳动态响应策略和多目标优化算法为基础,设计了分布式共识调度算法CBGT. 在边缘算网分布式调度模式下,各计算节点之间地位对等、不分先后顺序地进行各自独立的策略选择,再基于对手的策略寻找最佳的响应策略,迭代地更新自身的策略,直到收敛. 特别地,由于分布式调度中任务覆盖率低以及不必要的资源浪费是由于邻居节点之间的认知信息不一致导致的,CBGT算法将优化目标中的(3)和(4)转化为提高邻居之间的共识性. 优化目标(3)和(4)分别是提高任务覆盖率和降低资源消耗. 在分布式调度中,导致任务覆盖率低和资源浪费的其中一个原因是各节点收集到的任务信息不全且存在差异、无法共识,进而导致以下的问题:(1)2个节点由于不知道对方的决策结果,进而重复执行同一个任务,浪费了计算资源;(2)在计算任务饱和的情况下,如果因为重复分配造成了资源浪费,会进一步导致其他空闲任务得不到及时处理. 图1已经给出了一个存在闲置资源的示例,在实际部署过程中拓扑情况随机多变,存在闲置资源和空闲任务的情况很多. 因此,解决上述问题的根本方法就是通过邻居之间的信息共享,充分消除节点之间决策结果的冲突、信息差,提高共识. 本文通过在两跳邻居节点之间交换决策结果,引入共识消除机制以及一致性优化目标,来提升邻居节点之间分配结果的共识性,有效避免了节点之间分配结果的冲突,保障了任务覆盖率,避免了计算资源的过度消耗. CBGT算法将任务完成时间和邻居之间的共识性共同作为调度的优化目标,双目标的求解问题借助多目标优化算法实现. CBGT算法作为分布式调度算法,必然面临共识收敛问题,本文进一步借助博弈论中的势博弈理论对分布式调度的收敛性进行了理论证明,见第5节.

在边缘算力网络中,为了在保障任务覆盖率的同时尽可能降低通信开销,当每个算力服务器的任务服务范围H_t是n跳时,其通信和共识消除机制的最佳范围H_c应为$2n$跳. 以下对上述取值原则进行具体分析.

(1)从通信开销的角度进行分析. 为分析方便,本文首先进行如下合理化假设:(a)假设每个节点都采用逐跳广播的方式转发共识消除消息;(b)每一跳的所有节点都同时接收到上一跳节点发来的广播消息并同时进行下一跳的转发处理;(c)任意一次广播的传输延迟D_{trans}、处理延迟D_{proc}和带宽消耗C_{band}相等.

基于上述假设,每个节点的一轮通信与共识消除的交互开销是随着H_c的增加等量递增的. 由此,本文建立节点i完成一轮通信与共识消除的交互开销模型C_{com}如下:

$$C_{\text{com}} = H_c C_{\text{onehop}},$$
$$C_{\text{onehop}} = \alpha D_{\text{trans}} + \beta D_{\text{proc}} + \gamma C_{\text{band}},$$

其中,系数α、β和γ分别表示上述3种开销占总体开销的比重. 需要特殊说明的是,一次调度过程的总交互开销由迭代轮数和一轮通信与共识消除的交互开销二者共同决定,其中迭代次数由算法决定,难以通过模型表示,但可以确定的是,在共识消除范围H_c不小于$2n$的情况下,迭代轮数不随H_c的变化而变化,即在H_c不小于$2n$的情况下一次调度过程的总交互开销的变化情况仅由一轮通信与共识消除的交互开销C_{com}决定. 因此,在H_c不小于$2n$的情况下,随着H_c的增大,节点i完成一轮通信与共识消除的交互开销C_{com}逐渐增大,一次调度过程的总交互开销也会逐渐增大.

(2)从任务覆盖率角度进行分析. 在边缘算力网络中,如果每个边缘计算服务器的任务服务范围是局部范围而不是全网,各服务器只有在各自服务范围之间有共同的任务时才可能会出现任务重复分配的情况,进而降低资源效率和任务覆盖率. 因此,本文建立节点s_1和s_2的任务服务范围之间存在共同任务的概率模型如下:

$$P(s_1,s_2) = \frac{\int_{o(s_1,s_2)} \lambda(x)\,\mathrm{d}x}{\int_{A(s_1)} \lambda(x)\,\mathrm{d}x + \int_{A(s_2)} \lambda(x)\,\mathrm{d}x - \int_{o(s_1,s_2)} \lambda(x)\,\mathrm{d}x},$$

其中,$A(s_1)$表示节点s_1的n跳服务范围,$o(s_1,s_2)$表示邻居节点s_1和s_2之间的任务服务范围的交叉区域,$\lambda(x)$表示给定位置x处的任务生成率,$B(s_1)$表示节点s_1的$2n$跳范围. $o(s_1,s_2)$随着节点s_1和s_2之间关系的变化如下:

$$\begin{cases} o(s_1,s_2) > 0, & s_2 \in B(s_1) \\ o(s_1,s_2) = 0, & s_2 \notin B(s_1), \end{cases}$$

进而

$$\begin{cases} P(s_1,s_2) > 0, & s_2 \in B(s_1) \\ P(s_1,s_2) = 0, & s_2 \notin B(s_1). \end{cases}$$

因此,只有s_2在节点s_1的$2n$跳范围内时,两节点之间才可能存在任务交叉范围和共同任务,才需要进行共识消除,即$H_c \geqslant 2n$即可保障节点之间不存在任务冲突,进而保障任务覆盖率和资源利用率.

综上,H_c应该选择不小于$2n$的值,即在$H_c \geqslant 2n$时,交互开销C_{com}随着通信和共识消除范围H_c的增大而增

大,进而造成不必要的交互开销,却没有提升任务覆盖率和资源利用率. 因此,H_c的最佳取值为$2n$,既保证了任务覆盖率,也不增加额外的交互开销.

CBGT算法的主要流程如图3所示. 当节点仅从一跳范围内的任务中进行策略选择时,只有两跳范围内的节点之间可能存在策略冲突. 因此,为了在保障任务覆盖率的前提下尽可能地降低交互开销,仅两跳邻居节点间交互各自的调度结果. 以节点i及其两跳范围内的其中一个邻居节点j为例,节点i首先根据本地信息和效用函数确定局部策略,然后将其局部策略与两跳范围内的所有邻居进行共享,基于两跳邻居的策略计算二者策略的一致性,进而完成策略的更新;所有节点反复迭代上述过程,直到策略不再更改,达到收敛状态. 效用函数是为了最大化任务效率和与邻居节点之间的一致性.

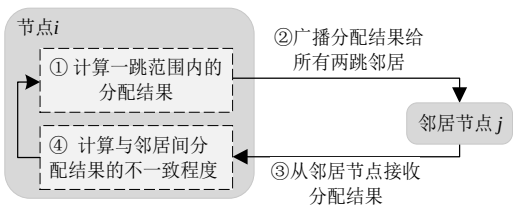


图3 代理i的任务分配流程

在上述步骤①中,代理i形成的分配结果表及其变化过程如图4所示. 第1行表示当前迭代t时刻被分配的邻居代理,第2行表示代理i为各一跳邻居分配的任务,第3行为各配对对应的时间成本,用于与邻居交互时消除碰撞冲突,第4行为该分配结果的时间戳T_{stamp},即其生成的时刻,第5行为该分配结果的总时间代价C_{total},第6行为该分配结果与邻居节点分配结果之间的一致性C_{cons}. 节点i对应表格中的U_1. 随着迭代过程的进行,在接收到邻居节点的分配结果后,节点i会持续调整分配结果,以避免和邻居节点的冲突. 此外,每次迭代都会有代理和任务被成功分配出去,节点i一跳邻居范围内的代理及其数量$\left|S_{\text{nei}}^{i,t}\right|$、任务及其数量$\left|S_{\text{nei_ta}}^{i,t}\right|$会随着迭代$t$不断变化. 因此,节点$i$的分配结果表的长度、内容等均随着时间变化. 在$t_0$时刻,代理$i$一跳范围内共有四个代理$U_0$、$U_1$、$U_2$、$U_3$,以及四个任务$T_0$、$T_1$、$T_2$、$T_3$. 而经过一轮迭代$t_0$以后,由于任务$T_1$被邻居代理$j$抢占,且代理$j$比代理$i$具有更小的任务代价,则代理$i$需要给自己重新分配任务. 此外,在第一轮迭代$t_0$过程中,代理$U_2$成功被分配了任务,代理$i$在收到$U_2$发来的消息后,则将其从自己的分配列表中删除. 由此,在t_1时刻,表格的长度变成了3,U_1的任务也由T_1变成了T_x. 表格的列数由$\left|S_{\text{nei}}^{i,t}\right|$和$\left|S_{\text{nei_ta}}^{i,t}\right|$中的较小值决定. 表格的节点号和任务编号由单独的注册服务器进行统一管理,各用户生成任务时需向注册服务器发送注册请求消息,以获得固定字节长度的任务编号信息. 代理i执行任务j时的时间成本计算方法如下:

$$t_{ij}=\frac{N_{\text{comp\_0}}^{j}}{C_{\text{comp\_0}}^{i}}+\frac{N_{\text{trans}}^{j}}{C_{\text{trans}}^{ij}}+\frac{N_{\text{mem}}^{j}}{C_{\text{mem\_1}}^{i}} \tag{8}$$

其中主要包括计算时间、传输时间和存储的写入时间三部分. 由于计算、传输和存储等维度的量纲不同,所以本文对其进行归一化处理,将计算、传输、存储的成本均转换到时间维度上,保障了度量的统一性. 在保障了任务覆盖率的前提下,任务调度的主要目标是先保障任务的完成时间短. 不同资源在时间维度上的量纲是可以实现统一的,并没有将不同量纲资源直接进行加减运算等不合理或脱离实际的问题.

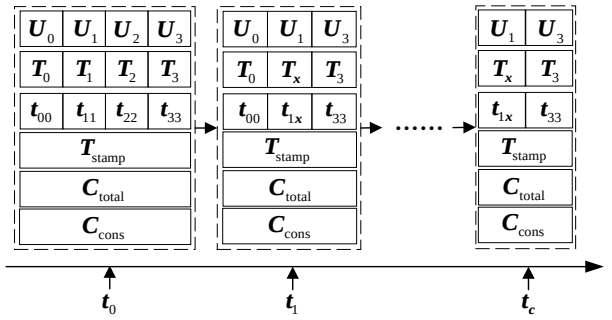


图4 代理i的分配结果及其随着迭代过程的变化情况

在接收到邻居节点的分配结果后,如果代理i当前没有被分配任务,则执行更新过程. 代理i基于收到的邻居节点信息获得尚未被分配任务的邻居节点和任务信息,进一步基于上述信息计算出所有备选的配对组合. 然后,代理i统计各备选配对组合与邻居分配结果之间的一致性,针对每一个配对结果,代理i检查邻居节点j是否有相同的一跳邻居,当发现二者有共同的一跳邻居U_1时,查看二者给U_1分配的任务是否相同,如果相同,则增加该分配结果的一致性值,进而得到所有配对结果与邻居的一致性结果. 最后,代理i获取到一致性值最高的配对结果集合,从中选择代价值最小的结果,作为新的分配结果.

算法1给出了智能体i决策的全过程. 在每次迭代t开始时,智能体i首先获取尚未分配的一跳邻居集合$S_{\text{nei}}^{i,t}$和任务集合$S_{\text{nei_ta}}^{i,t}$. 然后,基于集合$S_{\text{nei}}^{i,t}$和$S_{\text{nei_ta}}^{i,t}$中的元素进行全排列组合,计算得到所有可能配对集合S_{pop}^{i}和对应的S_{cost}^{i}. 集合S_{pop}^{i}和S_{cost}^{i}中的元素数量相同,配对和对应的cost值之间按照集合中的顺序一一对应. 当t为0时,由于尚未获得邻居的分配结果,无法得到一致性值. 因此,需对S_{cost}^{i}进行降序排列,从中选择代价值cost最小的配对结果p_i,然后将其赋值给分配结果列

表 A^i_{allo}. 当 t 大于 0 时,所有节点已经完成一轮初始分配,但这个结果可能与邻居节点间存在冲突和交叉. 因此,将集合 S^i_{pop} 中的所有分配结果与邻居的分配结果 A^j_{allo} 进行比较,如果节点 j 与 i 分配了相同的任务,则从二者中选择时间成本较低的保留,将较高节点的结果置为空,并将其从备选任务集合中删除;统计二者的一致性,得到一致性集合 S^i_{cons}. 其同样与 S^i_{pop} 中的元素数量相同,二者按照集合中的顺序具有一一对应关系. 然后,将 S^i_{pop} 按照 S^i_{cons} 进行降序排列,得到一致性最高的备选分配结果. 进一步,从上述的备选分配结果中选择 cost 值最小的 p_i,将其赋值给列表 A^i_{allo}. 最后,将 A^i_{allo} 发送给所有的两跳邻居,再从所有两跳邻居接收各自的分配结果. 智能体 i 迭代地重复上述过程,直到其策略不再改变. 注意,每个代理的迭代计数可能是不同的,这里允许每个代理有不同的迭代周期.

算法 1　智能体 i 执行算法 CBGT

1. **FOR** $t = 0,1,\cdots,T_{\max}$ **DO**:
2. 获取待分配的节点 $S^{i,t}_{\text{nei}}$ 集合和任务集合 $S^i_{\text{nei\_ta}}$;
3. 计算所有可能的配对结果集合 S^i_{pop} 和对应的 S^i_{cost};
4. 根据各节点的计算能力筛选 S^i_{pop} 和 S^i_{cost},将不满足要求的结果剔除;
5. 对 S^i_{cost} 进行降序排列;
6. **IF** $t = 0$ **DO**:
7. 获得 cost 值最小的配对结果 p_i,将其赋值给分配结果列表 A^i_{allo};
8. **ELSE**:
9. **FOR** $j \in S^i_{\text{nei2hop}}$ **DO**:
10. **IF** $A^{ii}_{\text{allo}} != A^{ji}_{\text{allo}}$ **DO**:
11. 比较二者的时间成本 t_{ii} 和 t_{ji},将时间成本较高的节点的结果置为空;
12. **IF** $t_{ii} > t_{ji}$ **DO**:
13. 将该任务从节点 i 的待分配任务集合 $S^{i,t}_{\text{nei\_ta}}$ 中删除;
14. 统计集合 S^i_{pop} 中各配对结果与 A^i_{allo} 的一致性;
15. 得到一致性集合 S^i_{cons};
16. 将 S^i_{pop} 按照 S^i_{cons} 进行降序排列,得到一致性最高的备选分配结果 A;
17. 从上一步的备选 A 中选择 cost 值最小的 p_i,将其赋值给列表 A^i_{allo};
18. **FOR** $j \in S^i_{\text{nei2hop}}$ **DO**:
19. 发送 A^i_{allo} 到节点 j;
20. 从邻居接收分配结果 A^j_{allo}

算法 1 中的第 14~17 行是基于帕累托最优理论以及 NSGA-II 来设计的. 在每轮迭代 t 过程中,每个节点首先获得了 S^i_{pop} 和 S^i_{cost}(第 3 行),然后又基于与邻居节点的交互,获得了 S^i_{cons}(第 14~15 行),S^i_{pop}、S^i_{cost} 以及 S^i_{cons} 三者是元素数量相同的集合,S^i_{pop} 中包含所有备选的配对,而 S^i_{cost} 和 S^i_{cons} 中分别对应 S^i_{pop} 中各备选配对对应的任务成本和一致性值. 从上述过程可以看出,截至到算法第 15 行,每个备选配对结果都对应 2 个评价指标,且 2 个指标量纲不同,给决策带来了困难,则需要用到帕累托最优理论和 NSGA-II 算法,具体如下:(1)将 S^i_{pop} 分别按照 S^i_{cost} 和 S^i_{cons} 的值进行降序排列(第 5 行和第 16 行),得到 2 个元素值相同但顺序不同的序列 $S^i_{\text{pop\_cost}}$ 和 $S^i_{\text{pop\_cons}}$,两序列各自的最优结果共同组成了帕累托前沿面/帕累托最优解集;(2)确定各个优化目标之间的优先级(如算法 1 中一致性的优先级为 1,时间成本的优先级为 2;(3)从优先级为 1 的目标对应的决策序列中,选择最优的结果 A(第 16 行);(4)如果 A 中存在超过 2 个结果,则再从优先级为 2 的序列中进行筛选,直到选出唯一的结果(第 17 行).

5　基于 DPG 的算法收敛性证明

收敛性是分布式决策问题中的经典属性. 在分布式博弈中,每个代理都有自己的策略空间和目标函数,根据当前的信息和策略各自独立进行决策,而代理之间的协同博弈过程又相互影响,且迭代反复,能否达到收敛状态是一个不确定问题. 对于使用最佳响应动态(Best Response Dynamics,BRD)的一般博弈,没有收敛结果,即基于 BRD 的算法可能会错过一个纳什均衡. 幸运的是,对于一些特殊类型的对策,存在着始终能够保证序列 BRD 收敛到均衡点的充分条件,如势博弈[42]. 势博弈是一种非合作博弈,其中玩家改变自己行为的动机可以用一个单一的函数来表示,称为效用函数. 这里玩家最大化其效用等同于最大化全局目标,全局目标函数称为势函数. 由于效用对齐,势博弈有 2 个重要的特性. 第一个特性是保证了纯策略纳什均衡的存在性. 由于在势博弈中,联合策略空间是有限的,因此势函数总是至少存在一个最大值. 这种局部或全局最大化势函数的策略文件是一个纯策略纳什均衡. 因此,每一个势博弈都至少具有一个纯策略纳什均衡. 第二个特性是势博弈可以作为一种策略设计工具来实现系统级的目标[43]. 文献[44]对 CBAA 进行了基于势博弈理论的解释和分析,证明了其收敛于某些分布式福利博弈的纯策略纳什均衡,无政府状态的代价和该均衡的稳定性代价分别为 1/2 和 1. 然而,经典博弈论的一个主要假设是用户是在静态环境中运行的,不受其他用户行动的影响,只考虑一轮博弈. 网络中各节点之间可以相互通信、相互影响,协同决策是一个具有耦合关系、多轮次反复的过程. 在此基础上,动态博弈考虑了节点之间的影响,更加符合算力网络中协同决策的特征[45].

动态势博弈和最佳动态响应分别为分布式任务调度的收敛性提供了理论和实践方法. 如何建立合理的势函数和本地收益函数,以满足势博弈模型特征,是将

博弈论和本文调度问题进行有机结合的关键. 本节将其建模为博弈问题,引入动态势博弈DPG模型,建立符合条件的局部收益函数和全局势函数,并通过理论推导证明了二者之间的渐进等价性,为模型的收敛性提供了理论支撑. 本文所考虑的基本博弈模型的定义如下.

定义1 一个有限的n人标准式博弈是一个三元组$G=<\boldsymbol{U},\{A_i\}_{i\in U},\{u_i\}_{i\in U}>$,具有以下属性:

(1)$\boldsymbol{U}$是n个代理的有限集;

(2)$\boldsymbol{A}=(A_1,A_2,\cdots,A_n)$,其中$A_i$为智能体$i$的有限行动(或策略)集合,每个向量$\boldsymbol{a}=(a_1,a_2,\cdots,a_n)\in\boldsymbol{A}$是一个行动(或策略)文件;

(3)$\boldsymbol{u}=(u_1,u_2,\cdots,u_n)$,$u_i$是智能体$i$的实值效用(或收益)函数.

一个基本且最被广泛接受的是著名的纳什均衡. 如果智能体做出确定性选择(纯策略),纳什均衡定义如下:如果对于所有的智能体i和所有的策略$a_i'\neq a_i$,满足

$$u_i(a_i,a_{-i})\geqslant u_i(a_i',a_{-i}),$$

则策略文件$\boldsymbol{a}=(a_1,a_2,\cdots,a_n)$是一个纯策略纳什均衡. 其中$a_{-i}=(a_1,\cdots,a_{i-1},a_{i+1},\cdots,a_N)$表示除智能体$i$的策略之外的策略文件,$a_i'$和$a_i$分别表示智能体$i$的2个不同的策略.

求解博弈模型根本上就是求解上述纳什均衡. 一般来说,寻找博弈G的纳什均衡是一项困难的任务,因为各个智能体的效用函数、约束条件等都是耦合的. DPG模型为分布式调度问题提供了一个理论框架,由此可以从理论上分析局部和全局之间的渐进等价关系,为纳什均衡和收敛性的存在提供依据.

定义2 (动态势博弈)如果存在一个势函数ϕ,对每个智能体都满足如下条件,则这个策略博弈被称为动态势博弈:

$$\begin{aligned}&\sum_{t=0}^{\infty}\sigma^t\left[u_i(a_i',a_{-i},t)-u_i(a_i,a_{-i},t)\right]\\&\quad=\sum_{t=0}^{\infty}\sigma^t\left[\phi(a_i',a_{-i},t)-\phi(a_i,a_{-i},t)\right],\end{aligned}$$

其中,σ为折扣因子,$\forall i\in\boldsymbol{U},\forall a_i,a_i'\in A_i,\forall a_{-i}\in A_{-i}$.

这表示智能体i由行动a_i向a_i'变化带来的收益变化和势函数的变化相同. 势博弈中任意代理任意时刻的行动都是沿着改善路径严格增长的. 如果所有代理的联合行动空间是有限的,该势博弈一定存在一个纳什均衡.

本文将智能体i的局部收益定义为

$$u_i=\sum_{t=0}^{\infty}\sigma^t\left[\sum_{j\in S^i_{\text{nei2hop}}}d_{ij}(a_i,a_{-i},t)+\sum_{k\in S^i_{\text{nei}}}\sum_{j\in S^i_{\text{nei\_ta}}}t_{kj}\,x_{kj}(a_i,a_{-i},t)\right]\tag{9}$$

其中,S^i_{nei}和S^i_{nei2hop}分别为智能体i一跳和两跳通信范围内的邻居集合,$S^i_{\text{nei_ta}}$为智能体i一跳通信范围内的任务集合. 为了提高局部决策的全局优化性,定义了节点间决策结果的共识性d_{ij},表示智能体i和j针对智能体i直接连通范围内的任务的决策结果之间的汉明距离,即表示不同智能体关于同一个任务的调度结果的不一致程度,定义如下:

$$d_{ij}=\sum_{k\in S^i_{\text{nei\_ta}}}a_{ik}\oplus a_{jk}\tag{10}$$

其中,a_{ik}为智能体i对任务k的调度结果. 假设节点i和节点j各自的一跳服务范围内都有一个交叉任务k,如果节点i将任务k分配给其一跳范围内的另一个节点q,而节点j将任务k分配给自己,即$a_{ik}=q,a_{jk}=j,a_{ik}\oplus a_{jk}=0$,则$d_{ij}$的值不变;反之,如果节点$i$也将任务$k$分配给任务$j$,则$a_{ik}=j=a_{jk}$,$a_{ik}\oplus a_{jk}=1,d_{ij}$的值随之增加1. 也就是说,基于两跳范围内邻居之间决策结果的交互和协商,逐渐提高节点之间决策结果的一致程度,减少任务分配的冲突和交叉,进而提高全局优化性.

定理 对于上述博弈G,下面的势函数ϕ和本地效用函数u_i定义了一个势博弈:

$$\phi=\sum_{t=0}^{\infty}\sigma^t\left[\sum_{i\in\boldsymbol{U}}\sum_{j\in\boldsymbol{U}}d_{ij}(a_i,a_{-i},t)+\sum_{i\in\boldsymbol{U}}\sum_{j\in\boldsymbol{T}}t_{ij}x_{ij}(a_i,a_{-i},t)\right]$$

证明 基于动态势博弈理论证明分布式决策算法的收敛性,重点在于建立一个局部收益函数和一个全局势函数,通过数学推导,证明在其他节点策略不变的情况下,任意节点i的策略改变,导致局部收益函数的变化和全局势函数的变化相等,即可得到算法收敛的结果. 因此,首先分别给出了智能体i单方面将其行动从a_i变为a_i',而其他智能体的行动a_{-i}保持不变时,代理i本地效用函数的变化式:

$$\triangle u_i(a_{-i})=\sum_{t=0}^{\infty}\sigma^t\left[\begin{array}{l}\left(\sum\limits_{j\in U_i^2}d_{ij}(a_i',t)+\sum\limits_{k\in U_i^1}\sum\limits_{j\in S^i_{\text{nei\_ta}}}t_{kj}\,x_{kj}(a_i',t)\right)\\-\left(\sum\limits_{j\in U_i^2}d_{ij}(a_i,t)+\sum\limits_{k\in U_i^1}\sum\limits_{j\in S^i_{\text{nei\_ta}}}t_{kj}\,x_{kj}(a_i,t)\right)\end{array}\right]$$

以及全局势函数的变化式:

$$\triangle\phi(a_{-i})=\sum_{t=0}^{\infty}\sigma^t\left[\begin{array}{l}\left(\sum\limits_{k\in\boldsymbol{U}}\sum\limits_{j\in\boldsymbol{U}}d_{kj}(a_i')+\sum\limits_{k\in\boldsymbol{U}}\sum\limits_{j\in\boldsymbol{T}}t_{kj}x_{kj}(a_i')\right)\\-\left(\sum\limits_{k\in\boldsymbol{U}}\sum\limits_{j\in\boldsymbol{U}}d_{kj}(a_i)+\sum\limits_{k\in\boldsymbol{U}}\sum\limits_{j\in\boldsymbol{T}}t_{kj}x_{kj}(a_i)\right)\end{array}\right].$$

然后,保持本地效用函数变化式不变,通过对全局势函数进行等价代换和等项相消处理,最终使得全局势函数的变化式与本地效用变化式相等,即完成收敛性证明. 因此,重点在于全局势函数的处理过程. 通过扩展,全局势函数变成了由(1)、(2)和(3)等三部分组成,具体如式(11)所示. 为表示简便,这里分别用U_i^1和

U_i^2表示i的一跳和两跳邻居集合. 其中(1)部分对应的是距离代理i两跳范围之外的代理情况,由于代理i只决策和服务由一跳范围内代理发布的任务,因此,代理i两跳范围之外所有代理之间的任务分配结果与代理i完全无关,因此(1)部分为0;在a_{-i}不变的情况下,(2)部分对应的是节点i两跳范围内节点(除了节点i以外)的决策结果,不受代理i行动从a_i变为a_i'的影响,所以其值为0. 由此,在消除了这2个值为0的部分后,上式进一步演变为式(12). 因此,势函数ϕ和本地效用函数u_i完全相等,所定义的博弈模型为势博弈模型,且A为有限策略空间,则上述博弈存在纳什均衡,通过各节点局部范围内的优化,可以逐渐逼近全局目标,直到收敛. 由此,从理论上证明了CBGT算法能够收敛到纳什均衡点,不会进入到无限死循环状态.

$$\triangle\phi(a_{-i})=\sum_{t=0}^{\infty}\sigma^t\left\{\begin{aligned}&\overbrace{\left(\sum_{k\notin U_i^2}\sum_{j\in \boldsymbol{U}}d_{kj}(a_i')+\sum_{k\notin U_i^2}\sum_{j\in \boldsymbol{T}}t_{kj}x_{kj}(a'_i)\right)-\left(\sum_{k\notin U_i^2}\sum_{j\in \boldsymbol{U}}d_{kj}(a_i)+\sum_{k\notin U_i^2}\sum_{j\in \boldsymbol{T}}t_{kj}x_{kj}(a_i)\right)}^{(1)}\\&+\overbrace{\left(\sum_{k\in U_i^2,k\neq ij\in U_i^2,j\neq i}d_{kj}(a_i')+\sum_{k\in U_i^2,k\neq ij\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i')\right)-\left(\sum_{k\in U_i^2,k\neq ij\in U_i^2,j\neq i}d_{kj}(a_i)+\sum_{k\in U_i^2,k\neq ij\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i)\right)}^{(2)}\\&+\overbrace{\left(\sum_{j\in U_i^2}d_{ij}(a_i',t)+\sum_{k\in U_i^1j\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i',t)\right)-\left(\sum_{j\in U_i^2}d_{ij}(a_i,t)+\sum_{k\in U_i^1j\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i,t)\right)}^{(3)}\end{aligned}\right\}\tag{11}$$

$$\triangle\phi(a_{-i})=\sum_{t=0}^{\infty}\sigma^t\left\{\overbrace{\left(\sum_{j\in U_i^2}d_{ij}(a_i',t)+\sum_{k\in U_i^1j\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i',t)\right)-\left(\sum_{j\in U_i^2}d_{ij}(a_i,t)+\sum_{k\in U_i^1j\in S_{\text{nei\_ta}}^i}t_{kj}x_{kj}(a_i,t)\right)}^{(3)}\right\}=\triangle u_i(a_{-i})\tag{12}$$

6　实验评估

为了验证本文所提算法的性能,本节将所提CBGT算法与CBAA、EdgeDecAp等经典分布式算法进行了仿真实验对比. 首先,介绍了仿真实验的设置. 随后,对比了3种算法的性能. 最后,通过在不同参数下的测试,验证了所提算法的有效性.

6.1　场景和仿真设置

本文假设一个包含n个智能体和m个任务的场景,智能体和任务随机部署在$W\times W$的2维空间中,均处于静止状态,$W=100$ m,通信半径为R. 本文在仿真设置中参考了车辆边缘网络和卫星边缘网络中边缘服务器和任务的参数设置[34,46]. 在仿真过程中,智能体数量n的取值范围为[5~10, 30, 50], 任务数量n的取值范围为[5~10, 500~600]. 代理计算能力M包括代理时钟频率(取值范围为2.5~3.5 GHz)、内存大小(取值范围为32~100 GB)、可用核数(2~128)、存储空间(取值范围为100~500 GB)、存储写入速度(取值范围为80~160 MB/s)等5个维度,任务的计算需求N包括任务内存需求(取值范围8~20 GB)、内核核数需求(取值范围为2~10)、存储空间需求(取值范围为20~100 GB)、任务计算指令数(取值范围为10^9 ~10^{11})等4个维度. 在每轮迭代t初始,在原有拓扑的基础上,按照链路中断概率p随机生成新的邻接矩阵,链路中断概率取值范围是0~0.85. 为了保障所提算法对不同的拓扑具有普适性,本文将随机种子seed的取值范围设置为[0,200],也就是说,对每种n和m的组合,都考虑了200种拓扑情况,充分保障了仿真的样本多样性. 算法的最大迭代次数为200次,为收敛性的验证提供了足够的时间. 当节点之间达到协同一致时,仿真会提前终止. 假设每个代理都正确地知道一跳范围内的代理和任务的位置. 表2对上述参数进行了汇总.

本文考虑CBAA、EdgeDecAp等作为基准算法. CBAA是分布式目标分配中的经典算法,允许智能体在仅知道局部信息的情况下与一跳邻居进行迭代竞拍和冲突消除等过程,在达到一致性的同时实现任务的分配. EdgeDecAp算法是针对边缘计算场景下的应用部署问题设计的分布式算法,有效地为应用部署机制的可扩展性、可靠性等提供了保障. 相比于传统的CBAA算法,EdgeDecAp进一步考虑了应用组件之间的交互成本,将竞拍者的计算时间和组件间的交互成本共同作为竞拍的依据,更加适合算网调度场景特征. 同时,EdgeDecAp引入了拍卖者的角色,拍卖者可以作为不同竞拍者信息的汇集中心,拉近不在通信范围内的竞拍者之间的通信距离,进而在一定程度上降低了竞拍者之间的分配冲突.

此外,为了比较与全局最优解之间的差距,本文对

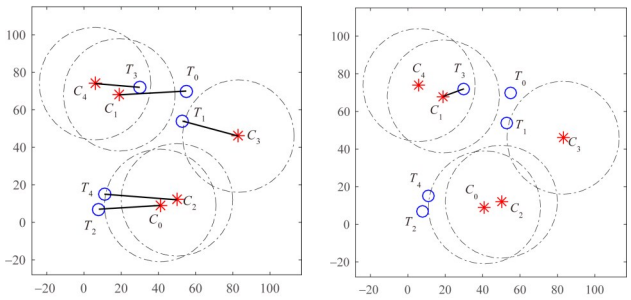
表2 仿真参数

实验参数	取值范围
代理数量n	[5~10, 30, 50]
代理时钟频率$C^i_{\text{comp_0}}$/GHz	2.5~3.5
代理内存大小$C^i_{\text{comp_2}}$/GB	32~100
代理存储空间$C^i_{\text{mem_0}}$/GB	100~500
代理存储写入速度$C^i_{\text{mem_1}}$/(MB/s)	80~160
代理CPU可用核数$C^i_{\text{comp_1}}$	2~128
代理/任务横坐标loc_x	0~100
代理/任务横坐标loc_y	0~100
任务数量m	[5~10, 500~600]
任务内存需求$N^i_{\text{comp_2}}$/GB	8~20
任务CPU核数需求$N^i_{\text{comp_1}}$	2~10
任务计算指令数$N^i_{\text{comp_0}}$	10^9~10^{11}
任务存储能力需求N^i_{mem}/GB	20~100
通信半径R	10~60
链路中断概率p	0~0.85
随机种子seed	0~200
最大迭代次数	200

GS和GS_onehop等全局最优解进行了计算. 这里涉及决策/服务的范围和冲突消除范围2个概念. GS_onehop和CBGT考虑的都是一跳的决策/服务范围,所以将二者对比是合理的. GS考虑的决策/服务范围是全网,所以不适合用来对比. CBGT中的两跳指的是冲突消除范围的两跳,是为了提升一跳的决策/服务效果的,而不是决策/服务的范围变成了两跳. GS是每个节点可以执行全网范围内所有任务情况下的全局最优解. GS_onehop算法是在已知全局所有信息的情况下,借助整数线性规划获得的理论上的理想结果. 请注意,这里考虑的理想结果是考虑了每个节点只执行一跳范围内任务这一特征的. 相比于GS中每个节点执行全局范围内的任务,GS_onehop更加适合作为本文的参考基准. 以图5为例,展示了将全局任务作为各节点备选任务集的GS算法与GS_onehop算法的对比结果. 特别地,为了能够从拓扑图中直观地看出任务配对的优劣,图5和图6将配对之间的代价设置为二者之间的直线距离,比如代理C_3和任务T_1之间的代价为

$$\sqrt{(\text{loc}_{x\_C_3}-\text{loc}_{x\_T_1})^2+(\text{loc}_{y\_C_3}-\text{loc}_{y\_T_1})^2},$$

其他所有仿真中配对之间的代价则是按照第4节中的式(8)进行计算. 图中2种拓扑各有5个CN节点和5个TASK节点,红色星号C代表算力节点,蓝色圆圈T代表任务节点,虚线代表每个算力节点的一跳通信范围,所有的算力节点C和任务节点T均分配了序号. 实线代表算力节点和任务之间的配对关系,如左侧GS中C_1被分配给T_0,所以二者之间用实线相连. 从图5中可以看出,左边GS算法中C_0、C_1、C_2和C_3分别选择了不在其一跳范围内的任务T_2、T_0、T_4和T_1,而右侧GS_onehop算法由于通信范围的限制,仅执行局部范围内的任务,导致C_0、C_2和C_3等3个CN节点均处于空闲状态. 虽然从图中看上去GS具有更高的任务覆盖率,但在网络规模变大的情况下,获得全网信息需要付出很大的开销和时间代价. 因此,本文将GS_onehop算法作为参考的理想基准,更加适用于大规模网络需求.



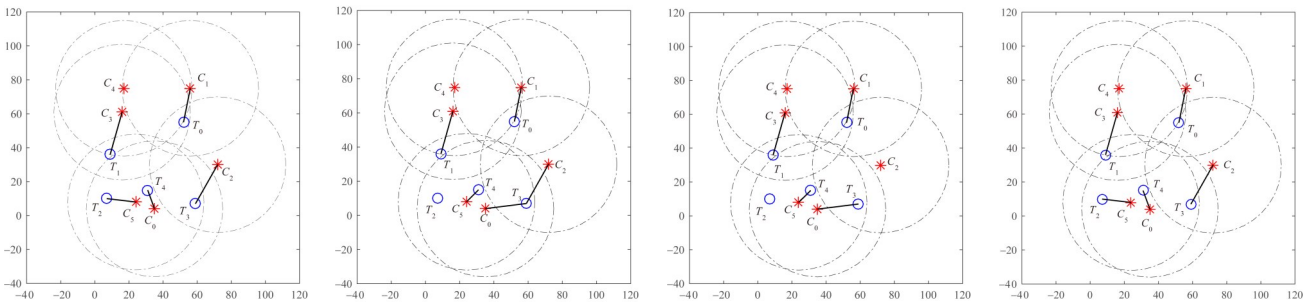
(a) GS (b) GS_onehop

图5 考虑局部和全局2种任务覆盖范围的理想结果对比

6.2 与基准算法对比

图6展示了在相同的随机拓扑中不同算法的分配结果. 图中各元素的含义与图5中保持一致. 为了保证清晰度,这里用了较少的6个代理和5个任务. 在CBAA算法中,由于节点只与一跳邻居节点通信,当2个不相邻节点C_0和C_2选择了同一个任务T_3执行时,由于C_0和C_2无法通信,互相之间无法知道对方的决策,因此会造成任务的重复分配和资源的浪费. 在EdgeDecAp算法中,由于任务发布者T_3同时也充当拍卖者,在T_3处同时得到了C_0和C_2的拍卖信息,且由T_3完成C_0和C_2最终的拍卖决策,将T_3分配给C_0,进而避免了由于C_0和C_2信息不共享带来的调度冲突. 而在CBGT中,节点i可以与两跳范围内的邻居节点通信,即可以与所有可能与节点i存在调度冲突的节点进行交互;此外,CBGT中每个节点都是考虑一跳范围内所有邻居和任务进行的综合决策,而不仅仅考虑自身的情况,进而可以用少量的通信和计算开销充分避免分配的冲突和资源的浪费,在有限的资源下使更多的任务获得计算服务. 在CBGT中,C_0节点综合了一跳范围内C_5节点的情况,认为C_5执行T_2、C_0执行T_4能够获得更大的整体收益;在此基础上,空闲的C_2选择T_3,进一步使任务覆盖率和资源利用率得到了提升.

图7展示了静态任务负载且总任务数变化情况下不同算法的任务覆盖情况. 在本文的设计中,目标首先是保障任务覆盖率,其次是减少资源浪费. 相比于CBAA、EdgeDecAp算法,CBGT算法具有更高的任务覆盖率,更加接近于GS_onehop的理想情况. 当任务覆盖率相同时,则需要进一步比较算力占用情况. 图8展示



(a) GS_onehop　(b) CBAA　(c) EdgeDecAp　(d) CBGT

图6　随机拓扑下不同算法的分配结果可视图

了与图7对应的算力节点分配情况,当总任务数为6或7时,虽然CBAA、EdgeDecAp等算法和CBGT算法具有相同的任务覆盖率,但CBAA占用了很多的算力节点,花费了更多的算力资源. 由此,CBGT具有更高的任务覆盖率和更少的算力消耗.

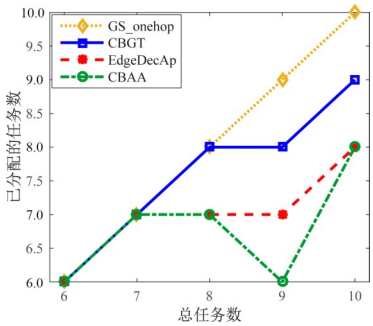


图7　任务覆盖率对比

上述仿真主要是考虑静态负载情况下的任务分配结果,而在系统实际运行过程中,任务流、节点的任务负载通常是动态变化的. 如图9所示,本文进一步考虑了50节点的网络和动态负载的情况,每间隔一段时间执行一次任务分配算法,对该时间段内产生的任务进行分配,总任务数为500~580个,每个节点的任务负载

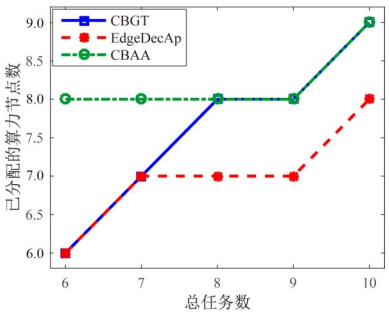
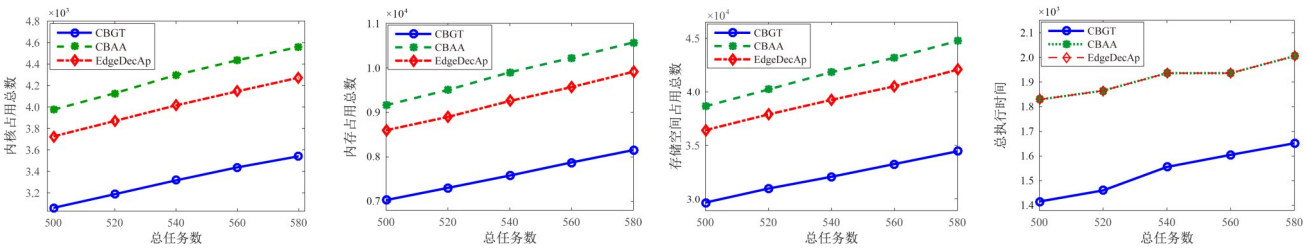


图8　算力占用情况对比

是动态变化的,当任务完成,节点就会将任务从待执行列表中删除,并释放对应的CPU资源. 迭代上述过程,直到所有的任务都分配完成. 在动态负载情况下,由于时间是连续不断的,所有任务都会被最终分配完成,差别主要是执行时间和所消耗的资源的不同. 因此,动态负载情况下,本文主要分析资源消耗和执行时间2个指标. 为了使仿真场景更贴合实际,本文进一步扩展了资源消耗的度量维度,具体包括内核占用总数、内存占用总数和存储空间占用总数等方面. 上述指标是整个动态分配过程中所有任务的指标之和. 相比于CBAA、EdgeDecAp等算法,CBGT算法消耗了更少的资源,具有更快的执行时间.



(a) 内核占用总数　(b) 内存占用总数　(c) 存储空间占用总数　(d) 总执行时间

图9　动态负载情况下的资源消耗情况及任务完成时间对比

6.3　通信半径对任务覆盖率的影响

图10分析了代理通信半径变化对不同算法任务覆盖率的影响. CBGT、CBAA、EdgeDecAp等算法均受到通信半径的影响,在通信半径变小的情况下,任务覆盖率逐渐降低,导致大量的闲置资源无法得到充分利用. 由此,证明了本文所考虑的在低交互开销的情况下提

升任务覆盖率是亟需解决的问题.

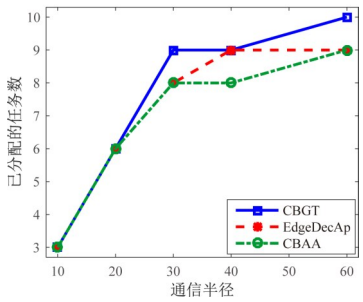


图 10 随着代理通信半径的变化的任务覆盖情况

6.4 链路中断对任务覆盖率的影响

在实际的通信环境中,通信链路通常无法保障100%可达,尤其在节点高速移动的情况下,链路中断概率不容忽视. 本节对存在链路中断情况下的任务调度算法性能进行了仿真分析. 在每个迭代时刻t开始时,根据给定的链路中断概率,在原有连通拓扑的基础上随机添加中断链路. 图11、图12给出了随着链路中断概率变化的任务覆盖率和算法收敛时间变化情况. 由于所考虑问题是离散型的,因此中断概率对算法性能的影响也不是呈现单调线性关系. 由于动态负载情况下时间是连续不间断的,图11考虑的是算法执行300次时的瞬时结果,图12是所有任务分配完成时的结果. 从图11中可以看出,任务覆盖情况虽然随着链路中断概率的变化略有波动,最大的波动是在通信半径$R=50$的时候,已分配的任务数从209降到203,但总体来看影响相对较小,并未呈现明显的下降趋势. 如图12给出了动态负载以及不同的总任务数m情况下,CBGT算法迭代次数随着链路中断概率提升的变化情况,从图中可以看出,随着链路中断概率的升高,虽然CBGT算法的迭代次数有一定的波动,但都是可以在有限时间内收敛的. 需要说明的是,在动态负载情况下,CBGT算法通常被执行不止一次,这里考虑的是算法执行300次之后的结果,因此迭代次数大于表2中的算法执行一次的最大迭代次数200次是合理的. 由此,证明了所设计CBGT算法对于高速移动等链路中断率高的场景具有较好的适应能力.

6.5 共识消除跳数对调度性能的影响

为了更清晰地展示跳数选择的原则,本文进一步增加了交互跳数变化对算法性能影响的仿真,结果如表3所示. 由于动态负载情况下时间是连续不间断的,这里考虑的是算法执行300次时的瞬时结果. 从表中可以看出,当交互跳数为2时,由于有效地消除了任务分配的冲突问题,导致已分配任务数从179个增加到了186个,且对内核、内存空间以及存储空间的消耗得到了有效地降低;当交互跳数进一步提高时,已分配任务

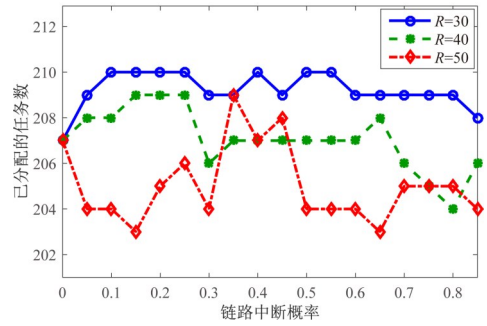


图 11 动态负载情况下随着链路中断概率变化的任务覆盖情况

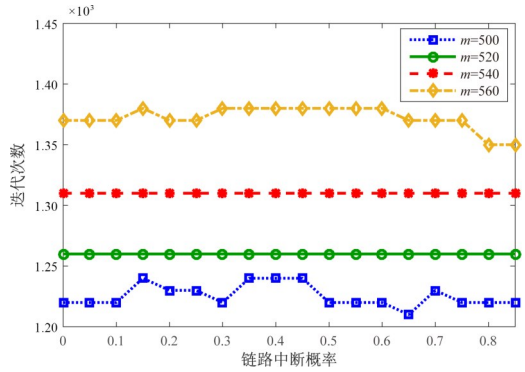


图 12 动态负载情况下随着链路中断概率变化的收敛时间变化情况

数不再增加,对计算资源的消耗也不再降低,但相比于交互跳数为2时,当交互跳数提高到3及以上时,由于交互范围的扩大,带来了额外的交互开销. 由于交互开销随着通信范围的增大而增大是一个显而易见的结果,因此没有在仿真结果中体现. 因此,在任务服务范围是1跳时,通信和共识消除机制的最佳范围是2跳,因为2跳是完全消除了任务的分配冲突情况且交互开销最小的跳数;以此类推,如果任务服务范围是2跳,通信和共识消除机制的最佳范围则应该是4跳.

表 3 交互跳数变化对调度性能的影响

交互跳数	已分配任务数	内核数	内存空间/GB	存储空间/GB
1	179	1 221	2 953	11 803
2	186	1 107	2 675	10 690
3	186	1 107	2 675	10 690
4	186	1 107	2 675	10 690

7 结束语

针对元宇宙、人工智能、大数据等新应用新技术驱动的算网协同需求及未来万物智联趋势带来的规模化挑战,本文从分布式调度模式出发,设计了基于动态势博弈模型的边缘算力网络任务调度算法. 本文融合了网元转发、存储、计算等多维度的能力和需求,搭建了边缘算网任务调度场景模型,并建立了优化目标和约束条件. 本文以最佳动态响应策略和多目标优化算法

为基础,设计了分布式共识调度算法CBGT,以完成对上述优化目标的求解. 最佳动态响应过程实现了动态博弈中节点之间的迭代式最优调度. 多目标优化策略则解决了由于邻居间一致性目标引入导致的可行解不唯一的问题. 特别地,考虑到通过两跳邻居之间的共识交互,可以充分避免节点间对于一跳范围内任务的分配冲突问题,本文巧妙地设计了基于两跳邻居间交互的共识消除策略. 为保障分布式模式下的共识收敛问题,本文依托博弈论中的势博弈理论,建立了符合势博弈模型的局部效益函数和全局势函数,借助势博弈理论从理论上证明了所设计算法的收敛性. 最后,本文通过仿真将所设计的CBGT算法与全局最优、局部最优以及CBAA、EdgeDecAp算法等进行了对比,相比于CBAA、EdgeDecAp等算法,CBGT算法具有更高的任务覆盖率和更少的资源浪费,更加接近于局部最优的情况. 此外,证明了所设计算法对于高速移动等具有较高链路中断概率的情况具有较好的适应性.

参考文献

[1] 中国联通算力网络产业技术联盟. 异构算力统一标识与服务白皮书[R]. 北京, 2021.
China Unicom Arithmetic Network Industry and Technology Alliance. White Paper on Unified Identification and Service of Heterogeneous Computing Power[R]. Beijing, 2021. (in Chinese)

[2] 张宏科, 于成晓, 权伟, 等. 融算网络体系基础研究[J]. 电子学报, 2022, 50(12): 2928-2934.
ZHANG H K, YU C X, QUAN W, et al. Fundamental research on computing integration networking[J]. Acta Electronica Sinica, 2022, 50(12): 2928-2934. (in Chinese)

[3] 贾庆民, 丁瑞, 刘辉, 等. 算力网络研究进展综述[J]. 网络与信息安全学报, 2021, 7(5): 1-12.
JIA Q M, DING R, LIU H, et al. Survey on research progress for compute first networking[J]. Chinese Journal of Network and Information Security, 2021, 7(5): 1-12. (in Chinese)

[4] CAO Y C, YU W W, REN W, et al. An overview of recent progress in the study of distributed multi-agent coordination[J]. IEEE Transactions on Industrial Informatics, 2013, 9(1): 427-438.

[5] CHOPRA S, NOTARSTEFANO G, RICE M, et al. A distributed version of the hungarian method for multirobot assignment[J]. IEEE Transactions on Robotics, 2017, 33(4): 932-947.

[6] BALINSKI M L. Signature methods for the assignment problem[J]. Operations Research, 1985, 33(3): 527-536.

[7] BERTSEKAS D P. A new algorithm for the assignment problem [J]. Mathematical Programming, 1981, 21(1): 152-171.

[8] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[9] CHOI H L, BRUNET L, HOW J P. Consensus-based decentralized auctions for robust task allocation[J]. IEEE Transactions on Robotics, 2009, 25(4): 912-926.

[10] SMOLKA S, WIßENBERG L, MANN Z Á. EdgeDecAp: An auction-based decentralized algorithm for optimizing application placement in edge computing[J]. Journal of Parallel and Distributed Computing, 2023, 175: 22-36.

[11] MOHAMMADI AMIRI M, GÜNDÜZ D. Computation scheduling for distributed machine learning with straggling workers[J]. IEEE Transactions on Signal Processing, 2019, 67(24): 6270-6284.

[12] YANG C S, HUANG-FU C C, FU I K. Carbon-neutralized task scheduling for green computing networks[C]// GLOBECOM 2022 - 2022 IEEE Global Communications Conference. Piscataway: IEEE, 2022: 4824-4829.

[13] YANG C S, AVESTIMEHR A S, PEDARSANI R. Communication-aware scheduling of serial tasks for dispersed computing[C]//2018 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE, 2018: 1226-1230.

[14] JIN A L, SONG W, WANG P, et al. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing[J]. IEEE Transactions on Services Computing, 2016, 9(6): 895-909.

[15] MA L B, WANG X Y, WANG X W, et al. TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things[J]. IEEE Transactions on Mobile Computing, 2022, 21(11): 4125-4138.

[16] DIAMANTI M, PAPAVASSILIOU S. Trading in collaborative mobile edge computing networks: A contract theory-based auction model[C]//2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS). Piscataway: IEEE, 2022: 387-393.

[17] DIONNE D, RABBATH C A. Multi-UAV decentralized task allocation with intermittent communications: The DTC algorithm[C]//2007 American Control Conference. Piscataway: IEEE, 2007: 5406-5411.

[18] CHAKRAA H, GUÉRIN F, LECLERCQ E, et al. Optimization techniques for multi-robot task allocation problems: review on the state-of-the-art[J]. Robotics and Autonomous

Systems, 2023, 168: 104492.

[19] QUINTON F, GRAND C, LESIRE C. Market approaches to the multi-robot task allocation problem: A survey[J]. Journal of Intelligent & Robotic Systems, 2023, 107: 29.

[20] LUO L Z, CHAKRABORTY N, SYCARA K. Distributed algorithms for multirobot task assignment with task deadline constraints[J]. IEEE Transactions on Automation Science and Engineering, 2015, 12(3): 876-888.

[21] ZHAO W Q, MENG Q G, CHUNG P W H. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario[J]. IEEE Transactions on Cybernetics, 2016, 46(4): 902-915.

[22] WHITBROOK A, MENG Q G, CHUNG P W H. Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems[J]. IEEE Transactions on Automation Science and Engineering, 2018, 15(2): 732-747.

[23] QU G N, BROWN D, LI N. Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions[J]. Automatica, 2019, 105: 206-215.

[24] LI T, SHIN H S, TSOURDOS A. Threshold greedy based task allocation for multiple robot operations[EB/OL]. (2019-09-03)[2024-01-26]. https://arxiv.org/abs/1909.01239v1.

[25] WANG J S, WANG J, CHE H J. Task assignment for multivehicle systems based on collaborative neurodynamic optimization[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(4): 1145-1154.

[26] FU X W, FENG P, GAO X G. Swarm UAVs task and resource dynamic assignment algorithm based on task sequence mechanism[J]. IEEE Access, 2019, 7: 41090-41100.

[27] WANG X M, SUI Y, WANG J P, et al. A distributed truthful auction mechanism for task allocation in mobile cloud computing[J]. IEEE Transactions on Services Computing, 2021, 14(3): 628-638.

[28] TANG Q Q, XIE R C, YU F R, et al. Distributed task scheduling in serverless edge computing networks for the internet of things: A learning approach[J]. IEEE Internet of Things Journal, 2022, 9(20): 19634-19648.

[29] BERALDI R, ALNUWEIRI H. Distributed fair randomized (DFR): A resource sharing protocol for fog providers[C]// 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC). Piscataway: IEEE, 2019: 29-36.

[30] CHEN L X, SHEN C, ZHOU P, et al. Collaborative service placement for edge computing in dense small cell networks[J]. IEEE Transactions on Mobile Computing, 2021, 20(2): 377-390.

[31] KARATALAY O, PSAROMILIGKOS I, CHAMPAGNE B. Energy-efficient resource allocation for D2D-assisted fog computing[J]. IEEE Transactions on Green Communications and Networking, 2022, 6(4): 1990-2002.

[32] DAI X X, XIAO Z, JIANG H B, et al. Task co-offloading for D2D-assisted mobile edge computing in industrial internet of things[J]. IEEE Transactions on Industrial Informatics, 2023, 19(1): 480-490.

[33] CHU W B, JIA X M, YU Z W, et al. On incentivizing resource allocation and task offloading for cooperative edge computing[J]. Computer Networks, 2024, 246: 110428.

[34] LI H, YU J Y, CAO L L, et al. Multi-agent reinforcement learning based computation offloading and resource allocation for LEO Satellite edge computing networks[J]. Computer Communications, 2024, 222: 268-276.

[35] ZHANG G P, YANG K, LIU P, et al. Achieving user cooperation diversity in TDMA-based wireless networks using cooperative game theory[J]. IEEE Communications Letters, 2011, 15(2): 154-156.

[36] MONDERER D, SHAPLEY L S. Potential games[J]. Games and Economic Behavior, 1996, 14(1): 124-143.

[37] 康海燕, 冀珊珊. 面向无线边缘网络的分层 Stackelberg 博弈群体激励方法[J]. 电子学报, 2024, 52(7): 2382-2392.
KANG H Y, JI S S. Hierarchical stackelberg game swarm learning incentive method for wireless edge network[J]. Acta Electronica Sinica, 2024, 52(7): 2382-2392. (in Chinese)

[38] CUI R X, GUO J, GAO B. Game theory-based negotiation for multiple robots task allocation[J]. Robotica, 2013, 31(6): 923-934.

[39] ARSLAN G, MARDEN J R, SHAMMA J S. Autonomous vehicle-target assignment: A game-theoretical formulation[J]. Journal of Dynamic Systems, Measurement, and Control, 2007, 129(5): 584-596.

[40] MARDEN J R, WIERMAN A. Distributed welfare games[J]. Operations Research, 2013, 61(1): 155-168.

[41] MARDEN J R. The role of information in distributed resource allocation[J]. IEEE Transactions on Control of Network Systems, 2017, 4(3): 654-664.

[42] HEIKKINEN T. A potential game approach to distributed power control and scheduling[J]. Computer Networks, 2006, 50(13): 2295-2311.

[43] MARDEN J R, ARSLAN G, SHAMMA J S. Cooperative control and potential games[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009,

39(6): 1393-1407.

[44] CHOI H L, KIM K S, JOHNSON L B, et al. Potential game-theoretic analysis of a market-based decentralized task allocation algorithm[M]//Springer Tracts in Advanced Robotics. Tokyo: Springer Japan, 2016: 207-220.

[45] PRASAD A, MOHAPATRA P S, REDDY P V. On the structure of feedback potential difference games[J]. IEEE Transactions on Automatic Control, 2024, 69(1): 637-644.

[46] LIU J X, WANG Y T, PAN D T, et al. QoS-aware task offloading and resource allocation optimization in vehicular edge computing networks via MADDPG[J]. Computer Networks, 2024, 242: 110282.

作者简介

张　晶　女,1994年出生于辽宁省沈阳市. 现为北京邮电大学计算机学院(国家示范性软件学院)、网络与交换技术全国重点实验室博士研究生. 主要研究方向为未来网络架构与智能路由.

E-mail: jzhang2021@bupt.edu.cn

关建峰　男,1982年出生于河南省巩义市. 北京邮电大学副教授、博士生导师. 主要研究方向为网络架构、区块链与网络安全、移动互联网、大数据与人工智能. 中国电子学会会员编号:E190091220M.

E-mail: jfguan@bupt.edu.cn

刘科显　男,1996年出生于河南省周口市. 现为北京邮电大学计算机学院(国家示范性软件学院)、网络与交换技术全国重点实验室博士研究生. 主要研究方向为网络安全.

E-mail: kxliu@bupt.edu.cn

申　奥　男,2002年出生于内蒙古自治区赤峰市. 现为北京邮电大学计算机学院(国家示范性软件学院)、网络与交换技术全国重点实验室博士研究生. 主要研究方向为未来网络架构和容错路由.

E-mail: shenao0128@bupt.edu.cn