DOI: 10.13382/j.jemi.B2306401

面向云-雾计算系统中的遗传算法任务调度研究*

王 昊[1] 李 晖[1,2] 宋端正[1] 诸锦涛[1]
(1. 南京信息工程大学电子与信息工程学院 南京 210044;2. 无锡学院电子信息工程学院 无锡 214105)

摘 要:近年来,不断发展物联网产生出海量数据,这给网络云等基础设施带来巨大压力。对此学者们提出的“云-雾”计算的架构模型,其中雾计算的障碍之一是如何分配计算资源以最小化网络资源。研究提出了一种基于启发式算法的 TCC(time cost computing-power)算法,以优化该异构系统中基于遗传算法的“云-雾”计算中的任务调度问题,包括执行时间、操作成本以及总算力资源。算法以“云-雾-端-网”混合计算任务调度为研究对象,以进化遗传算法为研究手段,结合云计算、雾计算、遗传算法等技术优势,实现时延、代价以及算力之间的平衡。在混合计算任务调度中,对于仅考量单一指标的 TCaS 算法,本算法具有更好的均衡性;本算法适应度值分别比 BLA 算法 0.93%以及 RR 算法 26.02%。本算法还可以灵活地匹配用户对高性能-成本-算力多方面的需求,提升系统的有效性。

关键词:云-雾计算;任务调度;遗传算法;熵值法
中图分类号:TN929.5 **文献标识码**:A **国家标准学科分类代码**:510.4

Research on genetic algorithm task scheduling in cloud-fog oriented computing systems

Wang Hao[1] Li Hui[1,2] Song Duanzheng[1] Zhu Jintao[1]
(1. College of Electronics and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China; 2. College of Electronics and Information Engineering, Wuxi University, Wuxi 214015, China)

Abstract: In recent years, the growing Internet of Things (IoT) has generated huge amounts of data, which has put enormous pressure on infrastructures such as the network cloud. One of the obstacles to fog computing is how to allocate computing resources in a way that minimizes network resources. A heuristic-based TCC (time cost computing-power) algorithm is proposed to optimise the task scheduling problem in genetic algorithm-based “cloud-fog” computing in this heterogeneous system, including execution time, operational cost and total computing power resources. The algorithm is based on “cloud-fog-end-net” hybrid computing task scheduling, and uses evolutionary genetic algorithms as a research tool, combining the advantages of cloud computing, fog computing and genetic algorithms to achieve a balance between latency, cost and computing power. In the hybrid computing task scheduling, this algorithm has a better balance performance than TCaS algorithm which only considers a single metric; the adaptation value of this algorithm is 0.93% and 26.02% higher than BLA algorithm and RR algorithm respectively. The algorithm is also flexible enough to match the user's needs in terms of high performance-cost-computing power, enhancing the effectiveness of the system.

Keywords: cloud-fog computing; task scheduling; genetic algorithm; entropy method

0 引 言

随着第 5 代通信技术(5th generation,5G)的飞速发展,目前通信网络不再是由单一网络所组成,而是由多种不同属性参数的网络异构融合而成的通信系统[1-2]。B5G(beyond 5G)通信系统中应用不同的无线接入技术,系统可以共享相同的无线资源从而构建成异构无线网

收稿日期:2023-04-03 Received Date:2023-04-03
*基金项目:国家自然科学基金项目(61661018)、江苏省基础研究计划青年基金项目(BK20210064)、江苏省双创博士人才项目(JSSCBS20210863)、南京信息工程大学滨江学院科研启动项目(2021r006)资助

络。而随着机器类型通信的需求越来越大(如智慧城市以及交通管理等),产生了大量的待处理的数据,这便需要网络设备具有处理大数据的运算能力。

云环境是支持设备算力网络发展的平台[3],可以利用云计算强大的计算能力来计算复杂度高、精确度高的任务来最小化资源的消耗。而随着连接设备数量的日益增加,传统的集中式的云计算架构模型难以完成如此大规模任务量的需求,主要原因在于云节点和设备之间的过于遥远,使得大规模设备产生的大量数据容易造成信道堵塞[4],降低了网络性能和服务质量(quality of service,QoS)。另外,为了降低成本,这些设备并不是时时刻刻都连接在云节点上。

云-雾计算系统具有巨大优势,例如降低时延、减小网络负担、降低成本以及提高处理任务效率,但也面临诸多挑战,主要挑战便是在处理任务中资源的分配以及任务的调度。云-雾计算系统中的任务调度目标是为了用户或服务商的利益。在用户方面,他们关注一些标准,如执行时间、预算上限、安全性保证、私密性保护和最小成本消耗。另一方面,服务商的目的是负载间平衡、资源利用率和能源效率最大化。为了保证 QoS,响应执行时间在直接影响用户体验时起着重要作用。此外,实施成本也是用户非常感兴趣的一个方面。一个任务时间表,它可以最小化完成时间并节省金钱成本以及算力等资源。算力也是一个人们常常忽视的重要网络资源,本文还添加算力资源考量,使得云-雾计算任务调度更具均衡性。

关于云计算与雾计算的研究,不少国内外学者都提出自己的见解。例如,文献[5]提出了当前物联网环境中遇到的困难,提出云计算与雾计算想结合的必要性。有学者提出利用强化学习方法,例如,Fang 等[6]提出利用深度强化学习的资源分配方法来优化云层与雾层的内容分配问题,用排队论将内容传输过程转化为延迟最小化问题,实现了低延迟的内容传输。Fan 等[7]提出利用强化学习方法,动态地将资源分配给有需求的雾辅助物联网任务,用来改善任务延迟、提高用户的 QoS。Khumalo 等[8]提出利用强化学习方法,在 5G 中引入雾计算,优化雾结构中的延迟、能耗以及成本问题,以在最大约束条件下用最小的能耗处理数据。Abdulazeez 等[9]提出利用强化学习来雾计算计算卸载优化,进一步提升雾计算能力。Guevara 等[10]提出利用强化学习在云-雾系统中的多目标任务调度,在保证用户的 QoS 前提下,最大限度地降低成本完成任务调度。

也有学者利用其他方法,例如,Bitam 等[11]提出利用蜜蜂生命算法(bee life algorithm,BLA)进行任务调度,BLA 算法主要关注执行时间以及需要内存参数,缺点便是只能在小型任务中使用,大型任务以及精确度要求高的任务并未进行准确测试。Binh 等[12]提出利用进化遗传算法(genetic algorithm,GA)进行任务调度,但是只在小型任务中完成了测试,并且只关注时间与成本消耗之间的权衡,并未考虑其他可能影响用户 QoS 的参数。Shen 等[13]提出利用对称平衡不完备设计来设计可以多用户参与的密钥协议,从而拓宽云计算中用户的数量,提升云计算系统中的计算能力,缺点是未在用户安全性以及隐私方便提供保护。Sun 等[14]提出针对云计算中的可量化安全评估方法,提高了整个云平台的安全性但未能将安全评估方法拓展到其他平台。Rabay'a 等[15]提出点对点雾计算模型,通过添加点对点机制来增强雾计算能力,减少对云平台的负担,同时还能提高带宽吞吐量,缺点是只在小型数据中进行验证,并且对延迟的容忍度比较低。Liu 等[16]提出多层雾计算网络,考虑成本问题使其最小化完成用户调度。文献[17]中提出了基于遗传进化算法的任务调度研究,但仅提及影响用户 QoS 的两个基本指标:执行时间及成本消耗,其应用范围有较大局限性。

在大多数研究中,通常算法仅部署在处理节点具有类似能力的云计算或雾计算系统中。而在混合云-雾计算系统中,在处理速度和资源使用成本方面,云节点和雾节点之间存在显著差异,因此任务不应该均匀分布到处理节点,因此一些经典的基于云的调度算法可能并不有效[18]。本课题主要提出了多属性算法,一种异构网络中基于遗传算法的云-边计算研究,以获得执行时间、处理成本以及算力资源之间的最优权衡。

本文主要贡献在于:用于处理大规模应用任务的高度分布式计算平台云-雾计算系统中的任务调度问题。提出了一种多属性感知调度算法 TCC 来解决这个问题,包括时间、成本以及常常被人们忽视的算力。算法的主要目标是在上述 3 种属性之间取得良好的平衡,以在云-雾计算系统中完成一系列任务。算法可以灵活地适应各种用户的需求,其中有人希望在当前时间优先考虑执行时间,而或有人希望在预算紧张的情况下完成任务,又或者有人希望在充足的算力资源下完成任务。同时,为在异构网络环境中实现更好的 QoS,本课题使用熵值法计算客观权重,帮助拥有更好的决策效果。本算法所得效果在时间、成本以及算力之间的平衡上更为优化。

1　云-雾计算系统模型

雾计算与云计算主要的区别在于:1)架构方面:雾计算实现分布式计算基础设备将边缘化的设备充分利用[19],云计算为传统的集中式架构;2)网络延迟方面:雾节点可以在网络边缘快速生成大数据、减少延迟;可以实现数据短暂缓存,减少网络压力,雾计算有时也被称为“边缘计算”,通过将数据存储在离移动用户与基站更近

的设备或者本地计算机中,而不是所有数据都是通过云端来运行,从而减小网络延迟,而设所有数据都在云端执行的话,将会产生非常大的延迟,影响用户的 QoS。3)系统响应时间方面:雾计算支持快速响应时间,云计算的响应时间却是比较慢的。4)工作环境与设备方面:雾节点可以安置于任何一个有网络的位置,例如公园、车俩或者写字办公楼中,例如交换机、路由器这种有处理能力的网络设备都可以作为雾节点[20],而云计算只能处于特定机房内。5)成本方面:雾计算这种可以安置于任何一个有网络的设备,这比专门用来建设云计算特定机房所需成本更低,云计算所需成本更加的高昂。6)安全性方面:雾计算的安全性可以自定义,用户隐私保障性更好,云计算则安全性欠佳。7)服务器节点方面:雾计算有很多存储能力计算能力有限的节点,云计算虽计算能力很大,但是可扩展存储能力的节点数量很少。但是云计算并不是一无是处,8)在计算容量方面:云计算相较于雾计算,计算容量更大。9)在保护数据方面:云计算在运行过程中,不会有数据损耗,保护数据的完整性,而雾计算会产生数据损耗。雾计算可以将大型数据中用户附近的网络设备上产生的任务进行处理,对云服务器的计算量进行分担。需要注意的是,雾节点的处理能力有限,对于任务数量过于庞大或者对时延精确度敏感的任务,仍需要在云节点处理[21]。表 1 总结了云计算与雾计算的区别。

表 1 云计算和雾计算的区别

Table 1 Difference between cloud and fog computing

参数	云计算	雾计算
延迟	高	低
系统响应时间	低	高
成本	高	低
架构	集中式	分布式
设备间通信	通过互联网从远处通信	通过边缘设备利用各种协议通信
距离信息来源	长期远离信息来源	短期内靠近信息来源
计算容量	更高	更低
数据损耗	不会减少数据	有数据损耗
服务器节点	具有可扩展存储和计算能力的节点很少	存储和计算能力有限的非常大的节点
安全性	安全性欠佳,未定义	更安全,可定义
工作环境	特定机房	室外(如公园)或室内
服务器节点位置	互联网内	本地网络边缘

如图 1 所示,在本系统模型中各个移动设备可与微基站通信或者也可以直接和宏基站通信,通过雾节点与云之间的相互配合,达到更好的任务调度。其中大部分移动用户距离微基站以及雾节点距离较近,所需传输资源较少,而距离云节点距离较远,所需的传输资源较多。每个宏基站都配备有一个雾节点,在网络边缘提供计算以及存储资源。雾节点根据网络状态以及候选网络参数进行任务调度,宏基站可以检测到各个移动设备的状态并将其发送到雾节点上,一般情况下,会有多个移动设备在多个位置、不同基站上传输数据。在每个位置上,宏基站都会将移动用户所需任务数据采集传输给雾节点以及云节点进行任务处理。雾节点与云节点相互配合,雾节点有响应延迟小,执行速度快等优势,云计算有计算容量大等优势,二者相互结合,发挥出最大性能。移动用户的任务需求具体传输路径为:通过移动设备访问微基站,由微基站传输到宏基站以及互联网内的云服务端中,而后由雾节点、雾代理和云节点按照系统模型任务执行表运行计算执行任务调度。

本文目标是优化执行时间、处理成本以及算力资源,这意味着找到一个解决方案,使时间、成本以及算力资源最小,并分别接近所有任务完成最短时间、完成一组任务最低成本以及所有任务完成所需算力资源最小。因此,效用函数 F 越接近 1,所获得的解越最优。

2 任务调度问题

2.1 系统架构

在云-雾系统中,雾节点在执行由智能设备所产生的任务时,由于处理能力有限,一些雾节点在区域内协同云节点共同工作,来满足用户的需求。设系统由 f 个雾节点和 c 个云节点以及 1 个雾代理组成。雾节点直接与移动用户通信。来自移动用户的所有请求都会立即转发给雾代理,该代理负责分析、估计并调度要在云-雾系统中执行的所有任务。雾代理靠近雾节点,两者数据通信所消耗的时间可以忽略不计。

为了保证系统的性能,安装在雾代理上的多属性算法旨在找到实现时间、成本与算力之间效率最优的任务执行时间表。该系统模型的运行步骤如图 2 所示。

步骤 1)移动用户发送请求,该请求由其所连接的雾节点处理;

步骤 2)此请求(或作业)立即转发给雾代理;

步骤 3)为了在分布式系统中进行处理,将每个作业计算并评估任务所需的资源使用量;

步骤 4)处理任务和节点的所有信息,雾代理运行调度算法以找到良好的任务分配;

步骤 5)输出之后,任务被发送到相应的云节点和雾节点;

步骤 6)每个节点中利用熵值法处理分配的所有任务并得到权重值,然后将结果发送回雾代理(即步骤 5));

步骤 7)完成所有任务后,雾代理将合并作业结果;

步骤 8)随后,通过用户连接的雾节点将响应发送给移动用户。

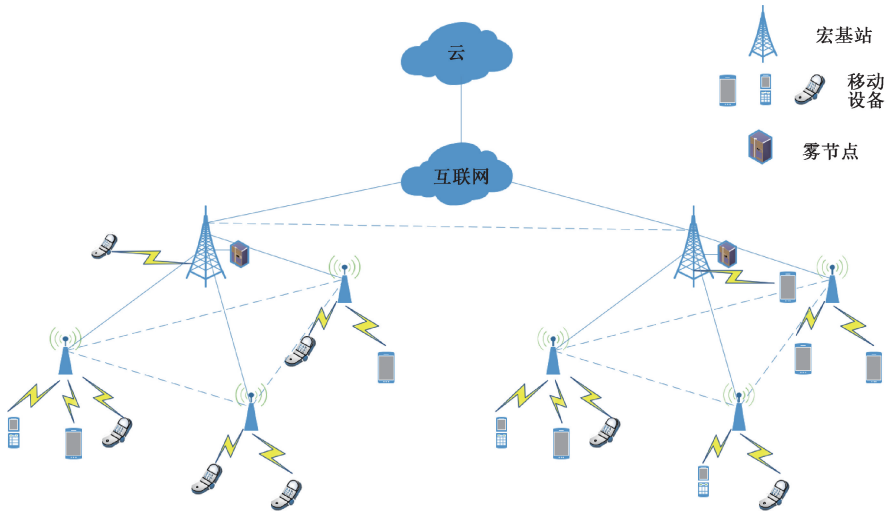

图 1　云-雾计算系统模型

Fig. 1　Cloud-fog computing system model

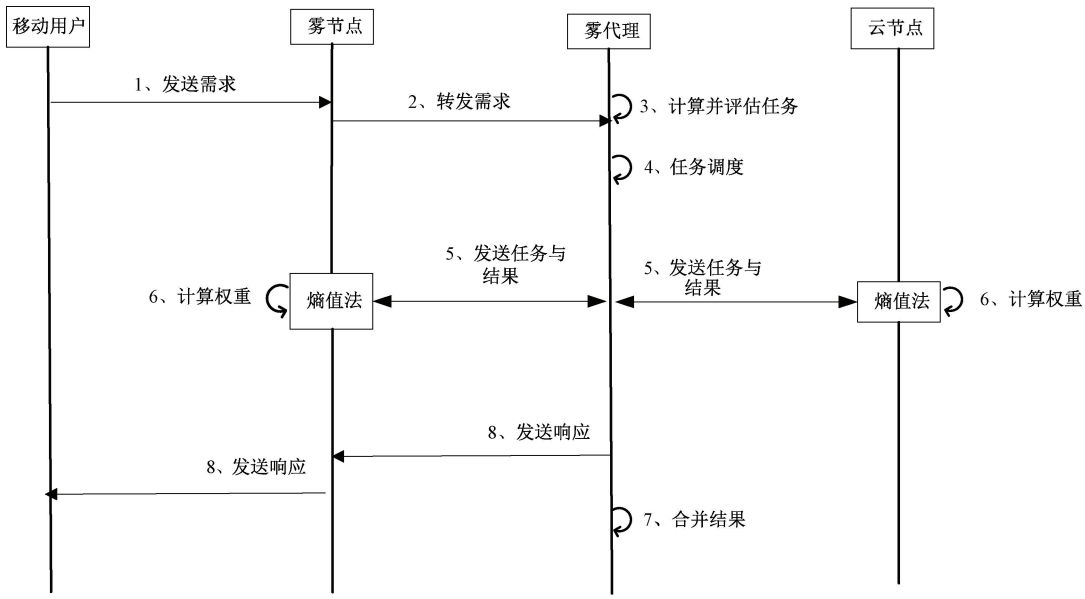

图 2　云-雾计算系统的操作流程

Fig. 2　The operation of the cloud-fog computing system

2.2　系统架构

当移动用户请求被转发到雾节点中，再由雾节点分解成各个小任务转发到各个雾代理中，每个小任务都有各自的需求与输入输出参数，比如所需内存大小以及输出文件大小等，所有小任务的需求与输入输出的组合又为成本、时间以及算力资源中重要的组成部分。设 M_n 为第 n 个任务，则每次分解都会有 n 个独立的小任务发送反馈回系统中，任务集合表示为：$M=\{M_1, M_2, M_3, \cdots, M_n\}$。

本云-雾计算系统中有云节点与雾节点相互合作工作，它们具有相同的属性，比如说 CPU 速率、CPU 使用费等各种使用费用，也有比如算力资源等不同的属性。系统中假设有 c 个云节点以及 f 个雾节点的共 t 个处理器的集合表示为：$N=\{N_1, N_2, N_3, \cdots, N_t\}$。其中 N_i 表示的是第 i 个处理器节点。

每一个任务 M_n 都会被分配到各自的处理器 N_i 中，用 M_n^i 来表示。从而可以通过一个处理器来处理一个或多个任务，用集合表示为 $N_iM=\{M_a^i, M_b^i, M_c^i, \cdots, M_z^i\}$。在本云 - 雾计算系统中，关于任务调度问题，集合表示为：$N_dM=\{M_x^1, M_y^2, M_j^3, \cdots, M_n^p\}$。

1)运行时间的计算

对于集合中的 N_iM 中来说,节点 N_i 完成分配到的任务所需的运行时间(running time,RT)可以表示为:

$$RT(N_i)=\sum_{M_n^i\in N_iM}RunTime(M_n^i)=\frac{\sum_{M_n^i\in N_iM}l(M_n)}{CPUr(N_i)} \tag{1}$$

其中,$RunTime(M_n^i)$ 表示为节点 N_i 处理任务 M_n 所需的运行时间,通过如下公式计算:

$$RunTime(M_n^i)=\frac{l(M_n^i)}{CPUr(N_i)} \tag{2}$$

其中,$l(M_n)$ 表示任务 M_n 的指令数量,$CPUr(N_i)$ 表示节点 N_i 的 CPU 速率。总运行时间(total running time,TRT)定义为从收到指令到完成最后一个任务所运行总时间:

$$TRT=\max_{1\leqslant i\leqslant t}[RT(N_i)] \tag{3}$$

设定 $mTRT$(minial TRT)为 TRT 的最小值,即系统完成所有任务所需时间最小值。在最理想情况下,所有的工作节点同时以最短时间完成分配任务,得到 $mTRT$:

$$mTRT=RT(N_1)=RT(N_2)=\cdots=RT(N_t) \tag{4}$$

2)消耗成本的计算

在云-雾计算系统中,处理一项任务则必须支付一定的成本费用,即执行成本、内存使用成本以及带宽使用成本。当处理节点 N_i 执行某一项任务时,需要支付的成本为:

$$C(M_n^i)=c_1(M_n^i)+c_2(M_n^i)+c_3(M_n^i) \tag{5}$$

执行成本由如下公式得:

$$c_1(M_n^i)=c_1'(M_n^i)\times RunTime(M_n^i) \tag{6}$$

其中,$c_1'(M_n^i)$ 是节点 N_i 在处理任务 M_n 时所需的单位时间 CPU 成本。内存成本由如下公式得:

$$c_2(M_n^i)=c_2'(M_n^i)\times Memory(M_n^i) \tag{7}$$

其中,$c_2'(M_n^i)$ 是节点 N_i 在处理任务 M_n 时所需的内存成本。$Memory(M_n^i)$ 为执行任务 M_n 时所需的内存大小。带宽使用成本由如下公式得:

$$c_3(M_n^i)=c_3'(M_n^i)\times B(M_n^i) \tag{8}$$

其中,$c_3'(M_n^i)$ 是节点 N_i 在处理任务 M_n 时所需的内存成本。$B(M_n^i)$ 为执行任务 M_n 时所需的内存大小。总的使用成本由如下公式得:

$$TC=\sum_{M_n^i\in N_dM}C(M_n^i) \tag{9}$$

而最小总成本(minimal total cost,mTC)是在云-雾计算系统中完成一整组任务所需的成本:

$$mTC=\sum_{M_n\in M}mC(M_n^i) \tag{10}$$

其中,mC(minimal cost)为完成单组任务所需的单组最小成本。

3)资源算力的计算

设 $Comp$(computing power)为节点 N_i 执行任务所需的算力资源,由于在云-雾计算系统中每个子系统所具有的算力资源不尽相同,对于节点的满足度也不相同,所以本文用相对贴近度 $\eta(X_i)$ 来映射出所需算力资源值,如表 2 所示。

表 2 相对贴进度 $\eta(X_i)$ 对应 *Comp* 值

Table 2 Relative posting rate $\eta(X_i)$ corresponding to *Comp* value

相对贴近度 $\eta(X_i)$	*Comp* 值
0~0.2	0.2
0.2~0.4	0.4
0.4~0.5	0.5
0.5~0.7	0.7
0.7~1.0	1.0

在此基础上定义一个效用函数,用来计算时间、成本以及总算力资源这三者之间的相互权衡:

$$F=\alpha\times\frac{mTRT}{TRT}+\chi\times\frac{mTC}{TC}+\delta\times Comp \tag{11}$$

其中,α χ 以及 δ 这 3 个参数分别对应着时间、成本以及总算力之间的平衡。当这 3 个参数都相等时,则意味着时间、成本以及总算力之间在优化过程中具有相同的优先级。当需要着重优化某一参数时,则将其对应的系数增大使其具有更高的权重。

设计的目标是优化时间、成本以及总算力之间,即找到一种方案使得 TRT 以及 TC 最小。因此,若效用函数 F 越接近 1,则所设计的方案越优。

3 算力计算模型

3.1 算力指标

算力是指输入数据在通过服务器时,服务器对其进行处理输出的一种计算能力,是评估服务器计算能力的一种综合指标,算力越大说明服务器计算能力就越强。主要将总算力划分为 5 大算力/能力进行评估:通用算力、智能算力、算效能力、网络能力以及存储能力[22]。

通用算力:通用算力主要是指服务器的通用计算能力,单位设定为 TFLOPS,本文使用“每秒浮点运算次数”来评估服务器的通用算力能力。

智能算力:智能算力主要是指智能计算能力,单位设定为 TFLOPS,本文同样使用“每秒浮点运算次数”来评估服务器的智能算力能力。

算效能力:服务器的算效能力设定为输入数据算力与所有设备功耗和的比值,即可以理解为所有设备每在消耗多少能量时输出多少的算力。

网络能力:本文主要采用网络带宽速度来评估服务器的网络能力。

存储能力:服务器的存储能力,即服务器每秒能够读

写的次数,主要是由存储容量、存储性能以及存储安全这3个方面来组成,本文使用每秒读写次数来衡量存储能力。

本课题所需要的数值为算力相对贴近度 $\eta(X_i)$,其计算步骤为:

步骤 1)根据排队论将所给数据进行划分优先级。将数据分为最大值与最小值,从而在其之间划分优先级,越接近理想值的数据所划分的优先级越高。

步骤 2)将优先级转化为优先数(0~1 之间的实数)。将步骤 1)中定出的优先级与数据一一对应,而后将不同的数据之间相互比较,从而得出的数据在指标下优于或劣于另一数据的优先级数,通过优先数算法将各个优先级数转变到 0~1 之间。

步骤 3)求解正、负计算值,根据定义 1,计算其所对应的向量。

步骤 4)根据式(18)和(19)求解出每个数值所对应的投影值。

步骤 5)根据式(20)求解出每个数据所对应的相对贴进度 $\eta(X_i)$。

步骤 6)用所得出的相对贴进度 $\eta(X_i)$ 映射出 *Comp* 值。

定义 1:假定 $\boldsymbol{X}_p=[r'_p,r''_p]$ 与 $\boldsymbol{X}_q=[r'_q,r''_q]$ 为集合 X 上的不确定等级变量,若将 X_p 与 X_q 通过广义优先数转化为 $\boldsymbol{X}_p=[\varsigma(r'_p),\varsigma(r''_p)]$ 与 $\boldsymbol{X}_q=[\varsigma(r'_q),\varsigma(r''_q)]$。因此 $\boldsymbol{X}_p$ 与 $\boldsymbol{X}_q$ 形成的向量定义为:

$$\boldsymbol{X}_p\boldsymbol{X}_q=[\min\varsigma(r'_p),\max\varsigma(r''_p)] \tag{12}$$

其中:

$$\min\varsigma(r'_p)=\min(|\varsigma(r'_p)-\varsigma(r'_q)|,|\varsigma(r''_p)-\varsigma(r''_q)|) \tag{13}$$

$$\max\varsigma(r''_p)=\max(|\varsigma(r'_p)-\varsigma(r'_q)|,|\varsigma(r''_p)-\varsigma(r''_q)|) \tag{14}$$

式中:$\varsigma(r'_q)$ 为 r'_p 通过广义优先数转化的结果,其他同理。

3.2　双向投影模型

双向投影法是一种经常被用于处理决策问题的方法,不仅可以考的被评价物体之间的距离,还可以考虑被评价物体之间的夹角[23]。同时,双向投影法还具有普通投影法所不具有的科学合理以及易实现等特性,因此在决策中广泛适用。本文采用双向投影模型[24],具体计算过程如下。

设定是第 m 个属性下,备选方案 $\boldsymbol{X}_i$ 等级转换所得到的信息。

首先,在第 m 个属性下,将正、负优先理想方案选定如下:$\boldsymbol{X}^+=[\max\varsigma(r'_{pm}),\max\varsigma(r''_{pm})]$ 与 $X^-=[\min\varsigma(r'_{pm}),\min\varsigma(r''_{pm})]$,其中 $1<m<n$,n 为备选方案的总数;根据定义 1 得出正、负优先理想方案解的向量值:$\boldsymbol{X}^-\boldsymbol{X}^+$。同理同样可以得到 $\boldsymbol{X}_i$ 与正、负优先理想方案形成的向量表示:$\boldsymbol{X}^-\boldsymbol{X}_i$ 以及 $\boldsymbol{X}_i\boldsymbol{X}^+$,其中:

$$\boldsymbol{X}^-\ \boldsymbol{X}^+=[\min((\max_{1\leqslant m\leqslant n}\varsigma(r'_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r'_{pm}),(\max_{1\leqslant m\leqslant n}\varsigma(r''_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r''_{pm})),\max((\max_{1\leqslant m\leqslant n}\varsigma(r'_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r'_{pm}),(\max_{1\leqslant m\leqslant n}\varsigma(r''_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r''_{pm}))] \tag{15}$$

$$\boldsymbol{X}^-\ \boldsymbol{X}_i=[\min((\varsigma(r'_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r'_{pm}),(\varsigma(r''_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r''_{pm})),\max((\varsigma(r'_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r'_{pm}),(\varsigma(r''_{pm})-\min_{1\leqslant m\leqslant n}\varsigma(r''_{pm}))] \tag{16}$$

$$\boldsymbol{X}_i\boldsymbol{X}^+=[\min((\max_{1\leqslant m\leqslant n}\varsigma(r'_{pm})-\varsigma(r'_{pm}),(\max_{1\leqslant m\leqslant n}\varsigma(r''_{pm})-\varsigma(r''_{pm})),\max((\max_{1\leqslant m\leqslant n}\varsigma(r'_{pm})-\varsigma(r'_{pm}),(\max_{1\leqslant m\leqslant n}\varsigma(r''_{pm})-\varsigma(r''_{pm}))] \tag{17}$$

从而得到向量 $\boldsymbol{X}^-\boldsymbol{X}_i$ 到向量 $\boldsymbol{X}^-\boldsymbol{X}^+$,以及向量 $\boldsymbol{X}^-\boldsymbol{X}^+$ 到向量 $\boldsymbol{X}_i\boldsymbol{X}^+$ 之间的投影值分别为:

$$\upsilon_{X^-X^+}(\boldsymbol{X}^-\boldsymbol{X}_i)=|\boldsymbol{X}^-\boldsymbol{X}_i|\cdot\cos(\boldsymbol{X}^-\boldsymbol{X}_i,\boldsymbol{X}^-\boldsymbol{X}^+)=\frac{\sum_{m=1}^{n}(\varsigma(\mu r'_{pm})\cdot\varsigma(\mu^-r'_{pm})+\varsigma(\mu r''_{pm})\cdot\varsigma(\mu^-r''_{pm}))}{|\boldsymbol{X}_i\boldsymbol{X}^+|} \tag{18}$$

$$\upsilon_{X_iX^+}(\boldsymbol{X}^-\boldsymbol{X}^+)=|\boldsymbol{X}^-\boldsymbol{X}^+|\cdot\cos(\boldsymbol{X}_i\boldsymbol{X}^+,\boldsymbol{X}^-\boldsymbol{X}^+)=\frac{\sum_{m=1}^{n}(\varsigma(\mu r'_{pm})\cdot\varsigma(\mu^+r'_{pm})+\varsigma(\mu r''_{pm})\cdot\varsigma(\mu^+r''_{pm}))}{|\boldsymbol{X}_i\boldsymbol{X}^+|} \tag{19}$$

当 $\upsilon_{X^-X^+}(\boldsymbol{X}^-\boldsymbol{X}_i)$ 的值越大,说明备选方案 $\boldsymbol{X}_i$ 的值越接近正优先理想方案解 $\boldsymbol{X}^+$,值越小,说明备选方案 $\boldsymbol{X}_i$ 的值越接近负优先理想方案解 $\boldsymbol{X}^-$。同理,可得 $\upsilon_{X_iX^+}(\boldsymbol{X}^-\boldsymbol{X}^+)$ 的目标是实现值的最大化,从而达到最接近正优先理想方案解 $\boldsymbol{X}^+$。

为了使结果更加准确合理,将备选方案和正、负优先理想解 $\boldsymbol{X}^-\boldsymbol{X}^+$ 的投影值相结合,用相对贴近度 $\eta(\boldsymbol{X}_i)$ 表示,通过如下公式来计算:

$$\eta(\boldsymbol{X}_i)=\frac{\upsilon_{X^-X^+}(\boldsymbol{X}^-\boldsymbol{X}_i)}{\upsilon_{X^-X^+}(\boldsymbol{X}^-\boldsymbol{X}_i)+\upsilon_{X_iX^+}(\boldsymbol{X}^-\boldsymbol{X}^+)} \tag{20}$$

4　熵值法计算权重

在基于效用函数的基础上,在权衡时间、成本以及总算力资源上的各自权重系数,即 α、χ 以及 δ,为保证更加的公平性以及更好的均衡性及用户体验,本文使用熵值法来计算各自的权重系数,以达到用户感知最优的目的。

熵值法来对异构网络中不同网络的同一属性进行分析,因为不同网络之间无法直接进行比较,通过熵值法对不同网络的同一属性分析得到的数据,属性值的波动越大、熵就越大;相反,属性值的波动越稳定、则熵就越小[25]。计算步骤如下。

首先对网络属性进行归一化处理,得到标准化矩阵 $\boldsymbol{R}$:

$$\boldsymbol{R}=\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1j} \\ r_{21} & r_{22} & \cdots & r_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} \end{bmatrix} \tag{21}$$

r_{ij} 计算公式为：

$$r_{ij}=x_{ij}\Big/\sqrt{\sum_{i=1}^{n}x_{ij}^2} \tag{22}$$

若网络属性属于代价属性，则标准化公式为：

$$q_{ij}=1-x_{ij}\Big/\sqrt{\sum_{i=1}^{n}x_{ij}^2} \tag{23}$$

则备选网络属性的熵值为：

$$E(m)=-\frac{1}{\ln n}\sum_{i=1}^{n}r_{ij}\times\ln r_{ij} \tag{24}$$

每个属性的客观权重：

$$w_m=\frac{1-E(m)}{\sum_{m=1}^{M}[1-E(m)]} \tag{25}$$

其中，$1-E(m)$ 为第 m 个属性的偏差度，并且每个属性的客观权重计算公式为 w_m 满足 $\sum_{m=1}^{M}w_m=1$。由此可得网络属性客观权重集为 $W=\{w_1,w_2,\cdots,w_n\}$。

5 遗传算法

在云-雾计算系统中，为了更好地完成任务调度问题，提出基于遗传算法的 TCC 算法，具体计算过程如下所示。

5.1 染色体编码

在遗传算法里，个体通常为某个问题的一个解。染色体所代表的个体会体现出遗传算法中任务调用的解决方案，使用 n 个基因的 n 维阵列来表示染色体的编码。任务的序列号由基因的序列表示。每一列基因都是范围为[1, t]的整数点，其中 t 为节点的数量。染色体编码可以灵活的选择遗传操作，方便创建更加新型的个体，同时还能从父代继承良好的基因。其不用考量任务的顺序的优势能够不影响云-雾计算中执行时间、成本以及算力资源。

5.2 种群初始化

假设一个种群的大小为 *nPop*，为了保证种群的多样性，这 *nPop* 的种群将被随机初始化。从最初的个体中选择进行一些操作进行遗传给下一代。

5.3 适应函数

在得到一个种群后，根据适者生存规则把优秀的个体保存下来，同时淘汰掉不适应环境的劣质个体。本文中适应函数即为式(11)，用来衡量本个体在种群中是否占据优势，F 越高，证明个体越好、遗传效果越成功。根据 F 的值同样可以用来衡量每个个体的适应度，从而进行一些生存与死亡操作。

5.4 遗传算子

通过选择可以得到较好的基因，但这并不是最好的基因，需要通过繁殖后代(交叉与变异)来得到更好的基因组，然而繁殖并不能保证后代完全优于父代，于是便需要接着通过选择使得更加适应环境的个体保存下来，从而完成进化，整个个体进化过程就是不断进行上述过程。

具体的繁殖过程包括交叉与变异。交叉是指个体由父亲和母亲两个个体繁殖所产生的，子代的个体 DNA 中有一半是父亲的 DNA 有一半是母亲的 DNA，不过此“一半”并非真正意义上的一半，而是随机产生的一个交配点，这个交配点可以是染色体中的任意位置。通过交叉得到的 DNA 可能会发送变异，使其 DNA 既不来自父亲也不来自母亲，在某个位置发送随机改变。

同时，交叉与变异并不是必然发生的，同样也是一个随机概率事件。交叉情况中，最坏的情况便是子代的 DNA 都要比父代的要差，如果交叉以一个较低的概率发生，则可以保证子代有部分基因和父代水平基本一样。对于变异来说，变异本质上是想让算法跳出局部最优解，如果变异的概率过大，那么算法迭代到最优解时还会不稳定，因此不管是交叉概率还是变异概率，都应选择一个适当范围。

6 仿真与数据分析

6.1 系统参数设置

本文云-雾计算系统中，由于每个云节点与雾节点在处理任务时都会产生不同的时间以及成本参数，设定有 10 个雾节点以及 3 个云节点共 13 个节点共同组建成云-雾计算系统，设定其中节点与基站中的坐标都为 $x\in[1,100]$ 和 $y\in[1,100]$，任务数为随机的 1 300~1 500 之间，CPU 速率：*CPUr*、CPU 使用成本 c_1'、内存使用成本 c_2' 以及带宽使用成本 c_3' 如表 3 所示。对于雾节点来说，雾节点处理任务速率有限，但是成本相对低廉。对于云节点来说，计算处理任务能力比雾节点要强，但是要付出更多的成本和代价。

表 3 云-雾计算系统中各节点参数

Table 3 Parameters of each node in a cloud-fog computing system

参数	雾计算	云计算	单位
节点数	10	3	node
CPUr	[1 300,1 500]	[3 000,5 000]	MIPS
c_1'	[0.1,0.4]	[0.7,1.0]	G $/s
c_2'	[0.01,0.03]	[0.02,0.05]	G $/MB
c_3'	[0.01,0.02]	[0.05,0.1]	G $/MB

注：$ 为美元，代表计算成本

本文设定 6 个网络组成云-雾计算系统,通用算力参数设定为[10,150]内随机选取;智能算力参数设定为[50,650]内随机选取;算效能力参数设定为[10,100]内随机选取;网络传输能力参数设定为[450,1 800]内随机选取;网络存储能力参数设定为[3,16]内随机选取[23]。则网络算力/能力的输入参数如表 4 所示。

表 4　5 大算力/能力参数

Table 4　Five computing power/ability parameters

通用算力/TFLOPS	智能算力/TFLOPS	算效/GFLOPSW	网络/Mbit/s	存储/IOPS
61.26	547.24	72.03	900.89	8.08
137.30	545.00	40.30	1 699.31	12.68
23.92	50.62	23.91	1 218.41	3.63
20.20	350.33	100.00	964.89	15.38
127.09	640.65	80.45	376.28	3.32
16.66	88.34	13.25	467.61	4.49

对于适应函数 F 来说,时间、成本以及总算力资源中各自的权重比 α、χ 以及 δ 占据重要地位。因此,使这 3 个参数设置得更加合理也是本文工作重点之一。设定 6 个网络有各自的时间、成本以及总算力资源,其中有的网络处理速度较快但是成本昂贵,有的网络消耗资源较少但是处理时间较长,为方便计算,这 6 个网络中的总算力资源设定为 5 大网络资源之和。候选网络属性参数如表 5 所示。

表 5　候选网络属性参数

Table 5　Candidate network attribute parameters

时间/s	成本/$	总算力
70	120	1 589.5
80	155	2 434.59
90	110	1 320.78
100	115	1 450.8
110	110	1 227.79
120	85	590.35

由此利用熵值法,即式(22)~(25),得到权重比参数值为:$\alpha=0.360\,3$、$\chi=0.325\,8$ 和 $\delta=0.313\,9$。

由于所提出的 TCC 算法是基于遗传算法得出的,便与一些其他进化算法进行比较:同样基于遗传算法的 TCas 算法[20];粒子群优化算法 MPSO[20];蜜蜂生命算法 BLA[14];简单循环调度算法 RR。表 6 给出了这几种算法的参数设置。

本算法所采用的数学符号说明表如表 7 所示。

6.2　仿真结果

对于本算法的可行性验证,本文场景设定为雾节点与多个云节点相互写作来共同处理用户请求任务。在本场景中,雾节点具有相似的属性值,任务会平均分配给节点,而云节点又比雾节点能力更强,因此找到最佳方案显得更为困难。

表 6　BLA、TCaS、MPSO 以及 TCC 算法参数设置

Table 6　Parameters used in BLA, TCaS, MPSO and TCC

参数		BLA	TCaS	MPSO	TCC
运行次数		30	30	30	30
种群大小	蜂王	1			
	雄蜂	30	100	100	100
	工蜂	69			
交叉率		90%	90%	$c_1=c_2=1.5$	90%
突变率		1%	1%	$w=1$	1%
子代数		500	500	[0.05,0.1]	500

表 7　数学符号说明

Table 7　Mathematical symbol description

符号	定义
节点 N_i 处理任务 M_n 所需的运行时间	*RunTime*
运行完所有任务所需的总运行时间	*TRT*
TRT 的最小值	*mTRT*
处理节点 N_i 执行某一项任务时,需要支付的成本	*C*
CPU 执行成本	c_1
内存使用成本	c_2
带宽使用成本	c_3
总的使用成本	*TC*
最小总成本	*mTC*
节点 N_i 执行任务所需的算力资源	*Comp*
映射算力资源所需的相对贴近度	$\eta(X_i)$
衡量算法优劣的效用函数	*F*
效用函数中时间权重系数	α
效用函数中成本权重系数	χ
效用函数中算力权重系数	δ

由于云-雾计算系统中不同网络的属性参数都有所不同,所具有的总算力资源也不尽相同,所以本文使用相对贴近度 $\eta(X_i)$ 来进行映射出本文所需的 *Comp* 值,将表 4 中数值代入式(18)~(20)中,计算得出相对贴近度 $\eta(X_i)$。再将计算出的相对贴近度代入表 2 得出 *Comp* 值。

图 3 给出在云-雾计算系统中 6 个网络随着任务数量的增加各自的对应的适应度值,由于设定 6 个网络中主要区别便在于总算力资源的相对贴近度 $\eta(X_i)$ 之间的差异,所以变化趋势相似。网络 1 与网络 5 之间计算出的总算力相对贴近度相近即 *Comp* 值相近,则在图中高度重合。且显而易见,TCC_2 的适应度值最高,这便得益于网络 2 的总算力资源相对贴近度 $\eta(X_2)$ 最高,使得 *Comp* 值最高。照成这样一结果原因在于图 3 实验是处于算力资源是唯一变量情况下,其余候选网络属性参数一致,这也说明了图 3 中 6 个网络适应度值变化趋势相一致的原因,由效用函数 F 公式(11)可知,当时间、成本以及加上算力资源对应的权重 α、χ、δ 这些参数一定时,唯一影响 F 值的只有 *Comp* 值。由表 2 可知,Comp 值又

由 $\eta(X_i)$ 映射得出，即在图 3 实验中，影响 TCC_1、TCC_2 到 TCC_6 适应度值的重要参数便是 $\eta(X_i)$。根据式(12)~(20)代入表 4 数值计算得出 6 个网络对应的总算力资源相对贴近度 $\eta(X_i)$ 值：$\eta(X_1)\approx 0.518\ 0$、$\eta(X_2)\approx 0.700\ 6$、$\eta(X_3)\approx 0.200\ 5$、$\eta(X_4)\approx 0.478\ 9$、$\eta(X_5)\approx 0.522\ 3$ 以及 $\eta(X_6)\approx 0.023\ 3$。根据表 2 可映射得出 *Comp* 值分别为：*Comp*_1 = 0.7、*Comp*_2 = 1、*Comp*_3 = 0.4、*Comp*_4 = 0.5、*Comp*_5 = 0.7 和 *Comp*_6 = 0.2。在熵值法计算出的权重值：$\alpha = 0.360\ 3$、$\chi = 0.325\ 8$ 和 $\delta = 0.313\ 9$ 基础上将 6 个 *Comp* 值代入式(11)可得 TCC_2 适应度在平均值上是最高的，适应度平均值约比 TCC_1 和 TCC_5 高 0.094 1、约比 TCC_3 高 0.188 3、约比 TCC_4 高 0.156 9、约比 TCC_6 高 0.251 1。在 5 大算力资源为唯一变量情况下，随着任务增加 6 个候选网络得适应度值如图 3 所示。

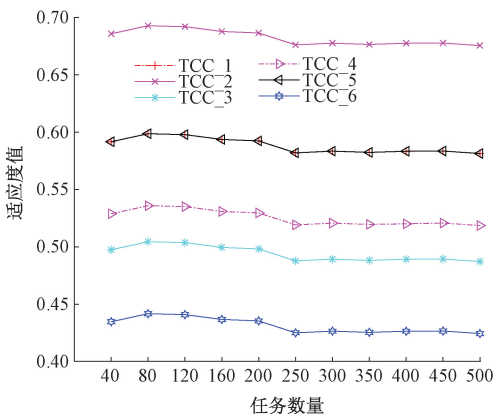

图 3 随着任务数量的增加 6 个网络的适应度值

Fig. 3 Fitness values for six networks as the number of tasks increases

图 4 表示出随着任务数量的增加，本文所提出的 TCC 算法与其他 4 种算法在 *TRT* 上的比较，图 4 显示出 TCC 与 TCaS 算法的 *TRT* 值相贴近，这是由于本文的 TCC 算法与 TCaS 算法架构相似，所以 *TRT* 值也相近，但优于架构简单的 MPSO 算法、BLA 算法以及 RR 算法。TCC 和 TCaS 是基于遗传算法理论，BLA 以及 MPSO 是基于进化算法理论，所以在所需时间 *TRT* 方面，当任务数量和数据规模不是很高时，四者都是相近的，由图 4 可以看出，当任务数量到达 450 和 500 时，BLA 和 MPSO 所需 *TRT* 值显然比本 TCC 算法高很多，体现出本 TCC 算法在高任务数量以及复杂任务环境依旧有很好的时间优势。简单循环调度算法 RR 是传统的任务调度算法，在算法架构上便是属于简单的将输入参数进行循环计算进行任务调度，因此在同样的任务数量时产生 *TRT* 值是 5 个算法中最大的。当任务数量更大，问题更加复杂时，RR 算法会需很大的 *TRT* 值，从而会影响用户的 QoS。

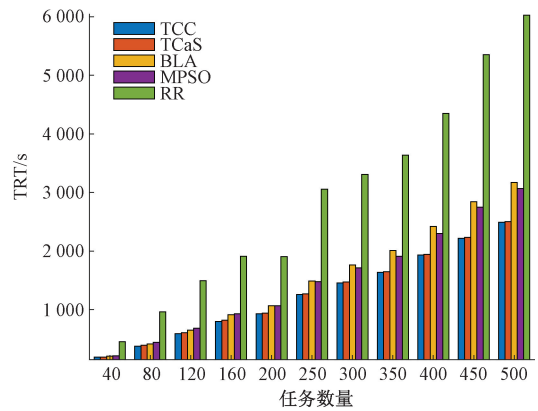

图 4 随着任务数量的增加不同算法所对应 *TRT* 值

Fig. 4 *TRT* values for different algorithms as the number of tasks increases

图 5 表示出随着任务数量的增加，本文所提出的 TCC 算法与其他 4 种算法在适应度值的比较。需要强调的是，本算法选择的是总算力资源贴合度更高的网络 2，则用 TCC_2 来表示且更加强调的是时间、成本以及总算力资源之间相互的权衡，因此会更加具有均衡性与公平性，更能满足大众的需求，更好地完成任务调度。图 5 中 TCC 的适应度值虽然没有 TCaS 算法高，但是 TCaS 算法只考量了时间与成本参数，并未涉及到算力资源，在均衡性上还略有欠缺。而本 TCC 算法在适应度值上平均下来高于 BLA 算法 0.93%以及 RR 算法 26.02%。

基于对以上场景的具体分析，本文所设的 TCC 算法与其他 4 种比较算法相比，TCC 算法可以在执行时间、消耗成本以及所需总算力资源上达到比以往其他算法都未能达到的均衡，同时显示任务调度的优势，而其中只具有简单机制的 RR 算法表现出最差的结果。

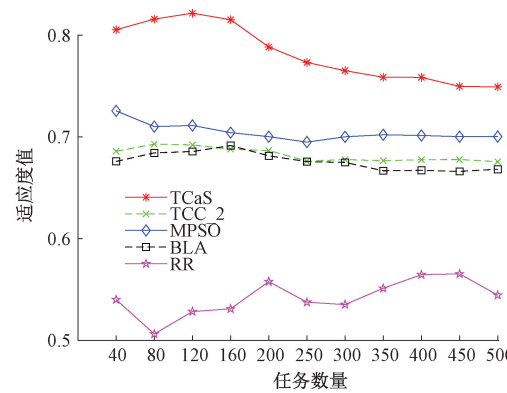

图 5 随着任务数量的增加不同算法对应的适应度值

Fig. 5 Fitness values for different algorithms as the number of tasks increases

在云-雾计算系统中由于权衡参数的不同，可能会导致适应度值的不同，当 TCC 算法中 $\alpha = 1$，其他权重值为 0

时,即对执行时间实现权重最大化,图 6 显示了当执行时间所占有权重最大时,随着任务数量的增加,系统的适应度值的变化。图 6 中显示出 TCC 算法的适应度值是最高的,平均下来高于 MPSO 算法 18.02%、BLA 算法 22.99%以及 RR 算法 53.43%。

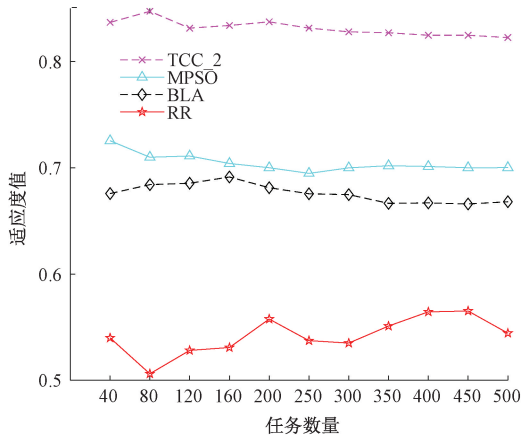

图 6　当 TCC 算法 $\alpha=1$ 时随着任务数量的增加对比各算法对应的适应度值

Fig. 6　Comparing the corresponding fitness values of each algorithm as the number of tasks increases when the TCC algorithm $\alpha = 1$

图 7 显示了随着系统任务数量的增加,云-雾计算系统中节点处理任务所需的总成本与执行任务所消耗时间的比值。TCC 的数值与 TCaS 数值一直想接近,这是由于二者在都是云-雾架构,完成任务所执行时间与消耗成本都相近,但是本文 TCC 算法还是要略高于 TCaS 算法,这是由于本文还考虑了总算力资源因素,造成单位时间内消耗成本要略高于 TCaS 算法。其中 TCaS 的适应度函数中的时间与成本都设定的为 0.5,设定时间与成本的优先值相同。关于 RR 算法在任务数量为 200 波动上升,这是因为此时 RR 算法的执行时间较小,只有 1 905.70 s,因此在任务数量 200 时 *TC/RunTime* 值较高与而本文的 TCC 算法设定的是基于熵值法计算出的适合时间、成本以及总算力资源的系数权重,具有更好的均衡性。

在云-雾计算系统中,雾计算的 *CPUr* 是影响成本的一个重要参数,图 8 显示了系统在不同的雾计算的 *CPUr* 下随着任务数量的增加所产生的适应度值,由图 8 中可以见到,当雾计算的 *CPUr*=[1 300,1 500]时,系统所产生的适应度值是最大的。而雾计算的 *CPUr*=[500,1 500]时,系统的适应度值是最低的。当雾计算的 *CPUr*=[1 500,2 000]时,系统所产生的适应度值并不是最高的,由此可见,并不是当雾计算的 *CPUr* 值越高适应度值就越高。在图 8 中显示出当雾计算的 *CPUr*=[1 300,1 500]时,系统在任务调度问题上能力最佳。

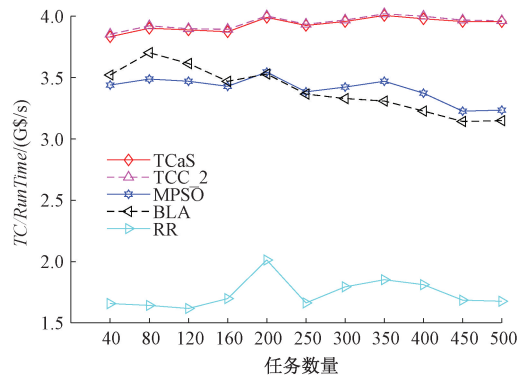

图 7　随着任务数量的增加不同算法的 *TC/RunTime* 值

Fig. 7　*TC/RunTime* values for different algorithms as the number of tasks increases

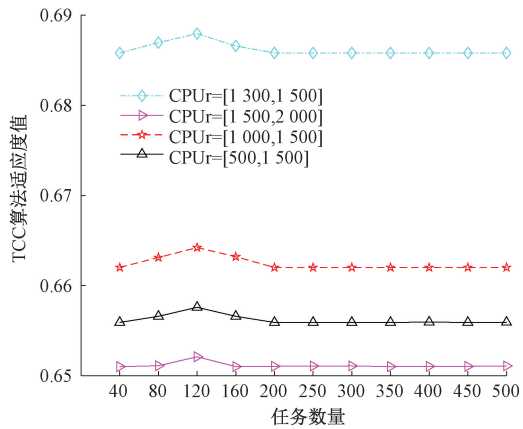

图 8　随着任务数量的增加不同雾计算 *CPUr* 所对应的适应度值

Fig. 8　Fitness values corresponding to the *CPUr* of different fog computing as the number of tasks increases

7　结　论

本文主要针对遗传算法在云-雾计算系统中的任务调度上的性能优化开展研究,以启发式算法为研究手段,提出基于遗传算法的 TCC 算法。TCC 算法考虑时间、成本以及最容易忽视的总算力资源上的任务调度方面,具有更好的均衡性更加能满足用户的需求,给用户更好的体验。

本文采用熵值法进行权重计算,以获得更加客观的权重值。考虑特殊的需求,如有限制时间、有限的预算或者有限的总算力资源,本算法可灵活改变权重值。在云-雾计算系统中,本算法在适应度值上平均下来高于 BLA 算法 0.93%以及 RR 算法 26.02%。

本文缺点在于系统计算量大且吞吐量较小,当任务数量过大时,所需执行时间过长,这对时间有限的移动用

户来说并不友好。在未来,本文将研究与改进更加优秀的算法来应用于云计算、雾计算以及任务调度问题。此外,还可以增加截止时间、预算上限以及算力极限这些限制,使得算法具有更高的实用性。同时,可以引入几何深度学习技术,使得计算性能得到进一步提升。

参考文献

[1] FANG F, YE G, ZHANG H, et al. Energy-efficient joint user association and power allocation in a heterogeneous network [J]. IEEE Transactions on Wireless Communications, 2020, 19(11): 7008-7020.

[2] LIU W, YU J, MU X. Research on power control technology based on LTE-A heterogeneous network[J]. Electronic Measurement Technology, 2016, 39 (12): 155-158.

[3] FALFUSHINSKY V, SKARLAT O, TULCHINSKY V. Cloud computing platform within Grid Infrastructure[C]. 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). IEEE, 2013, 2: 717-720.

[4] MARBUKH V. Systemic risks in the cloud computing model: Complex systems perspective[C]. 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). IEEE, 2016: 863-866.

[5] LINTHICUM D S. Connecting fog and cloud computing[J]. IEEE Cloud Computing, 2017, 4(2): 18-20.

[6] FANG C, XU H, YANG Y, et al. Deep-reinforcement-learning-based resource allocation for content distribution in fog radio access networks[J]. IEEE Internet of Things Journal, 2022, 9(18): 16874-16883.

[7] FAN Q, BAI J, ZHANG H, et al. Delay-aware resource allocation in fog-assisted IoT networks through reinforcement learning [J]. IEEE Internet of Things Journal, 2021, 9(7): 5189-5199.

[8] KHUMALO N, OYERINDE O, MFUPE L. Reinforcement learning-based computation resource allocation scheme for 5G fog-radio access network[C]. 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, 2020: 353-355.

[9] ABDULAZEEZ D H, ASKAR S K. Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment[J]. IEEE Access, 2023,11: 12555-12586.

[10] GUEVARA J C, TORRES R S, BITTENCOURT L F, et al. QoS-aware task scheduling based on reinforcement learning for the cloud-fog continuum[C]. GLOBECOM 2022-2022 IEEE Global Communications Conference. IEEE, 2022: 2328-2333.

[11] BITAM S, ZEADALLY S, MELLOUK A. Fog computing job scheduling optimization based on bees-swarm[J]. Enterprise Information Systems, 2018, 12(4): 373-397.

[12] BINH H T T, ANH T T, SON D B, et al. An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment[C]. Proceedings of the 9th International Symposium on Information and Communication Technology, 2018: 397-404.

[13] SHEN J, ZHOU T, HE D, et al. Block design-based key agreement for group data sharing in cloud computing[J]. IEEE Transactions on Dependable and Secure Computing, 2017, 16(6): 996-1010.

[14] SUN A, GAO G, JI T, et al. One quantifiable security evaluation model for cloud computing platform[C]. 2018 Sixth International Conference on Advanced Cloud and Big Data (CBD). IEEE, 2018: 197-201.

[15] RABAY'A A, SCHLEICHER E, GRAFFI K. Fog computing with p2p: Enhancing fog computing bandwidth for iot scenarios[C]. 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2019: 82-89.

[16] LIU Z, YANG Y, CHEN Y, et al. A multi-tier cost model for effective user scheduling in fog computing networks[C]. IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2019: 1-6.

[17] NGUYEN B M, THI THANH BINH H, THE ANH T, et al. Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud-fog computing environment[J]. Applied Sciences, 2019, 9(9): 1730.

[18] YANG R, YU F R, SI P, et al. Integrated blockchain and edge computing systems: A survey, some research issues and challenges[J]. IEEE Communications Surveys & Tutorials, 2019, 21(2): 1508-1532.

[19] ATLAM H F, WALTERS R J, WILLS G B. Fog computing and the internet of things: A review[J]. Big Data and Cognitive Computing, 2018, 2(2): 10.

[20] TUN K N, PAING A M M. Resource aware placement of IoT devices in fog computing[C]. 2020 International Conference on Advanced Information Technologies (ICAIT). IEEE, 2020: 176-181.

[21] LEE Y C, LIAN B. Cloud bursting scheduler for cost efficiency[C]. 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). IEEE, 2017: 774-777.

[22] 新型数据中心发展三年行动计划(2021-2023 年)解读[J]. 中国信息化,2021(9):21-23.

Interpretation of the 3-year action plan for the development of new data centers (2021-2023)[J]. China Informatization, 2021(9): 21-23.

[23] YE J. Bidirectional projection method for multiple attribute group decision making with neutrosophic numbers[J]. Neural Computing and Applications, 2017, 28: 1021-1029.

[24] 吴美希, 杨晓彤. 算力五力模型: 一种衡量算力的综合方法[J]. 信息通信技术与政策, 2022, 48(3): 13-21.
WU M X, YANG X T. Five-force model of computational power: A comprehensive method to measure computational power[J]. Information and Communication Technology and Policy, 2022, 48(3): 13-21.

[25] 陈发堂, 杨玲, 贾俊文, 等. 5G 异构网络中基于熵权 TOPSIS 的小区预切换方法[J]. 电讯技术, 2022, 62(3): 299-304.
CHEN F T, YANG L, JIA J W, et al. Cell pre-handover method based on entropy weight TOPSIS in 5G heterogeneous network[J]. Telecom Technology, 2022, 62(3): 299-304.

作者简介

王昊, 2021 年于新乡学院获得学士学位, 现为南京信息工程大学硕士生, 主要研究方向为异构无线网络融合中的多资源动态分配与优化研究。
E-mail: stud_wh@163.com

Wang Hao received his B. Sc. degree from Xinxiang University in 2021. Now he is a M. Sc. candidate in Nanjing University of Information Science and Technology. His main research interests include dynamic allocation and optimization of multiple resources in heterogeneous wireless network convergence.

李晖(通信作者), 于哈尔滨工业大学获得博士学位, 现为无锡学院电子信息工程学院教授, 主要研究方向为星地互联网络、6G 移动通信、工业互联网。
E-mail: hitlihui1112@163.com

Li Hui (Corresponding author) received his Ph. D. degree from Harbin Institute of Technology. Now he is a professor in the College of Electronics and Information Engineering, Wuxi University. His main research interests include satellite Internet, 6G mobile communication and industrial Internet.

宋端正, 2021 年于南京信息工程大学滨江学院获得学士学位, 现为南京信息工程大学硕士生, 主要研究方向为无线通信、强化学习。
E-mail: 2736404990@qq.com

Song Duanzheng received his B. Sc. degree from Binjiang College of Nanjing University of Information Science and Technology in 2021. Now he is a M. Sc. candidate in Nanjing University of Information Science and Technology. His main research interests include wireless communication and reinforcement learning.

诸锦涛, 2021 年于南京邮电大学通达学院获得学士学位, 现为南京信息工程大学硕士生, 主要研究方向为异构网络的资源优化。
E-mail: 1042807641@qq.com

Zhu Jintao received his B. Sc. degree from Tongda College of Nanjing University of Posts and Telecommunications in 2021. Now he is a M. Sc. candidate in Nanjing University of Information Science and Technology. His main research interests include resource optimization of heterogeneous networks.