Software Engineering and Applications 软件工程与应用, 2023, 12(6), 851-865
Published Online December 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.126083

Hans汉斯

云边端协同下的多云机器人卸载策略研究

杨　朝，李志聪

哈尔滨师范大学计算机科学与信息工程学院，黑龙江　哈尔滨

收稿日期：2023年10月28日；录用日期：2023年12月5日；发布日期：2023年12月14日

摘　要

如何充分利用中心云、边缘云和本地设备的计算资源，使得系统总代价尽可能降低，是云机器人任务卸载领域亟须解决的主要问题之一。针对该问题，提出一种基于遗传博弈的部分任务卸载算法(Genetic Game Theory-Partial Task Offloading, GGT-PTO)。将云机器人任务完成时间与系统能耗作为衡量系统总代价的两个指标，通过设置任务指标权重来模拟实际任务需求偏好，使用两个阈值对任务进行粒化分割。在每一轮对系统总代价的博弈中，使用遗传算法代替常规设置步长的方式，选出能使系统总代价降低量最大化的单个任务，并将该任务此时对应的卸载阈值更新入系统卸载策略集合中。不断重复以上过程，寻找该算法下的纳什平衡状态，确定整个系统的最佳卸载策略集合，进而降低整个系统的总代价。仿真结果表明，相比于常见二进制卸载算法和基于传统博弈的部分卸载算法，该算法在任务完成时间和能耗上均有明显降低，可使云边端协同服务的整体性能显著提升。

关键词

云边协同，任务卸载，云机器人，博弈论，遗传算法

Research on Multi-Cloud Robot Offloading Strategy under Cloud-Edge-End Collaboration

Zhao Yang, Zhicong Li

College of Computer Science and Information Engineering, Harbin Normal University, Harbin Heilongjiang

Received: Oct. 28th, 2023; accepted: Dec. 5th, 2023; published: Dec. 14th, 2023

Abstract

How to make full use of the computing resources of the central cloud, edge cloud and local devices to reduce the total cost of the system as much as possible is one of the main problems that need to

文章引用: 杨朝, 李志聪. 云边端协同下的多云机器人卸载策略研究[J]. 软件工程与应用, 2023, 12(6): 851-865.
DOI: 10.12677/sea.2023.126083

be solved in the field of cloud robot task unloading. To solve this problem, a genetic game-based partial task offloading algorithm (GGT-PTO) is proposed. The cloud robot task completion time and system energy consumption are taken as two indicators to measure the total cost of the system. The weight of the task indicator is set to simulate the actual task demand preference, and two thresholds are used to granulate the task. In the game of the total cost of the system in each round, the genetic algorithm is used instead of the conventional method of setting the step size to select a single task that can maximize the total cost reduction of the system, and the corresponding unloading threshold of the task at this time is updated into the system unloading strategy set. Repeat the above process, find the Nash equilibrium state under the algorithm, determine the best unloading strategy set of the whole system, and then reduce the total cost of the whole system. The simulation results show that compared with other binary offloading algorithms and partial offloading algorithms based on traditional games, the algorithm significantly reduces the task completion time and energy consumption, and can significantly improve the overall performance of cloud-edge collaborative services.

Keywords

Cloud-Edge Collaboration, Task Offloading, Cloud Robots, Game Theory, Genetic Algorithms

Copyright © 2023 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着大规模存储技术与网络通信技术的飞速发展，云计算这一概念早已不再流于表面，而是不断与各行业不同技术领域进行交叉、融合、深耕，云计算领域与机器人领域的结合自然也不例外。“云机器人”这一概念，是由卡耐基梅隆大学的 James Kuffner 教授在 Humanoids 2010 会议上首次提出[1]，是云计算领域与机器人领域相结合的重要成果。相比于普通机器人，云机器人能够利用云端资源，并通过将计算任务卸载到云端的方式，摆脱机器人自身资源有限性的弊端，大大提升云机器人的自身性能，能够更好的满足对实时性要求较强领域的需求。目前，云机器人技术已经被广泛应用在以下多个领域：感知和计算视觉领域、导航领域、抓取和操纵领域、制造或服务领域、社会、农业、医疗和灾难救助领域、众包应用等众多领域[2]。

随着第五代通信技术(5G)的快速发展，承载人脸识别、虚拟现实、车联网等高复杂度功能的应用程序正在大规模的进入人们的生活。伴随而来的是庞大用户量的激增以及海量计算的需求，基于云端的卸载策略显然已经无法满足当前人们的需求。于是，移动边缘计算[3] (Mobile Edge Computing, MEC)这一概念被提出，用来解决中心云服务器无法满足当前大规模计算量和用户激增而导致时延不稳定的窘境。由于 MEC 更接近用户端，时延大幅度缩短，更适配当前对实时性要求较高的应用。因此，MEC 技术被欧洲 5G 基础设施公司伙伴关系研究机构认为是实现 5G 严格关键性能指标的新兴技术之一[4]。

移动边缘计算的提出丰富了云机器人所能使用的计算资源，给寻找卸载策略提供了新的解决思路。目前，云机器人能使用的资源包含了中心云服务器、边缘云服务器和本地设备三种资源。当需要处理的任务计算量较小或任务安全性要求比较高时，只使用本地资源就能满足需求；当需要处理的任务计算量巨大或对任务安全性要求并不高甚至没有时，可以考虑将任务卸载到中心云服务器上，通过利用云服务器的强大计算能力来满足任务要求；当任务量较大或对任务安全性有一定要求，尤其对任务实时性要求

极高时，可以考虑将任务卸载到边缘服务器中，这样既利用了边缘服务较强的计算能力，也充分利用了边缘云距离用户较近，时延较低的优点。除此之外，当大量云机器人接收到任务时，可能存在需要通过同一信道将各自任务卸载到中心云服务器或者边缘云服务器的情况。此时，会出现传输信道的堵塞，进而影响卸载任务执行时间，产生额外的因信道阻塞而导致的运行成本[5]。

综上所述，决定云机器人系统整体性能优劣的主要因素是云机器人任务卸载策略的选择。当多云机器人系统短时间接收到庞大的高密集型任务时，如何选择出最佳的任务卸载策略，合理的利用中心云、边缘云和本地设备的计算资源，使得系统在实时性和能耗上满足甚至超出工程项目的基本预期，是任务卸载领域最主要的研究问题。

2. 相关工作

针对云边协同下任务卸载策略的设计和问题研究，诸多学者都在不断地拓宽该领域的上限。Dai [6]提出一种用于异构网络多任务处理的双层计算任务卸载框架。将问题分解成用户关联和计算卸载两个子问题，聚焦计算卸载子问题，核心是共同优化计算资源、传输功率和卸载分区，最后结合用户关联。该算法能优化多个 MEC 服务器之间的负载平衡，并以降低能耗为目标。局限性是未能使用中心云服务器的强大计算能力。Li 等[7]提出一种结合缓存与非正交多址接入(Non-orthogonal Multiple Access, NOMA)的联合方案。通过考虑多 MEC 服务器情况下对任务执行结果进行缓存，从而避免重复卸载，降低任务执行时延。局限性仍然是未能使用中心云服务器的强大计算能力。Guo [8]提出一种近似协同计算卸载方案(ACCO)和一种分布式计算卸载算法(DCCA)。该研究基于光纤无线混合网络，以降低设备总能耗为目标。局限性是未将任务完成时间作为系统代价的指标。

随着云计算领域不断散发出其强大的生命力，并在人们日常生活中提供愈加坚实而强有力的保障，该领域逐渐吸引了越来越多其它领域的学者。由此，博弈论被引入云机器人任务卸载问题中，并逐渐受到更多关注，各种改进博弈论算法在任务卸载领域亦大放光彩。Zhou [9]提出了一种基于博弈论的低时间复杂度的部分任务卸载算法，该论文基于多用户多无线信道的 5G 无线边缘计算系统，以系统能耗和时延的加权为目标。局限性是未使用中心云服务器的强大计算能力。Wu [10]提出了一种基于融合近似因子和博弈论的低时间复杂度的资源卸载分配算法，该论文联合考虑了云选择和路由优化，引入近似因子减少总路由的次数，并通过 Lyapumov 函数证明算法的收敛性。局限性是未使用中心云服务器的强大计算能力。Liang [11]提出了一种近似合作计算卸载方案和一种基于博弈模型的合作计算卸载方案，该论文基于传统的云边协同环境，以系统的时延和能耗为目标，通过合作的方式，降低时延和能耗，从而提升系统的整体性能。Wang [12]提出了一种基于定价机制和联合博弈理论的联合卸载方案，该论文基于多 MEC 的 5G-D2D 网络系统。局限性是未使用中心云服务器的强大计算能力。

目前，基于博弈论原理进行改进，来处理任务卸载问题是当前比较常用的处理策略。除此之外，随着“算力”时代的到来，深度学习和启发式算法的优势也逐渐明显。然而它们均有各自的不足，深度学习和启发式算法虽然能获得比较接近最优卸载的策略，但本身不具备可解释性；博弈论具有可解释性，但博弈论在任务卸载领域的使用过程中，需要将连续的卸载空间通过设置步长的方式，使连续值转变成离散值，这样一来就极其容易丢失最优卸载策略，进而导致所获取到的最佳卸载策略的保真性较差、系统性能表现不够良好。

基于上述论文的不足和对现有常用策略的分析，为了在保证可解释性的同时，进一步提升系统的整体性能，本文设计了一种云边端协同下基于遗传博弈的多云机器人部分任务卸载策略。将传统博弈论和一种十分常见的启发式算法(遗传算法)进行结合。在遗传博弈算法中，博弈论的使用保证了该算法是可解释的，而遗传算法的使用降低系统的总代价，进一步提升系统的整体性能。在中心云－单边缘云－本地

协同条件下，通过模拟多个云机器人任务特征，将任务通过单一信道进行卸载，使用遗传博弈的方法，将任务时间和能耗的加权作为代价函数，当卸载策略更新后，任务的卸载量和信道传输速率随之改变，进而影响整个系统的总代价。另外需要注意的是：由于本算法中增加了任务安全等级，因此系统在任务卸载的安全性上具有一定的保障。

具体算法流程为：(1) 根据任务安全等级高低，将所有任务分成两种类型。安全等级高的任务强制二进制卸载到本地运行且不可更改。(2) 针对安全需求低的任务，使用两个阈值将任务粒化分割，运用遗传算法寻找单个任务的最优卸载策略。(3) 通过设计的遗传博弈问题模型，以不断博弈的方法寻找所有任务卸载策略的纳什平衡状态，确定整个系统的最佳任务卸载策略。

3. 系统模型和代价函数

3.1. 系统环境模型

图 1 为本文所使用的中心云－单边缘云－本地协同系统环境模型[13]。系统环境模型中包含单个大型中心云服务器、单个边缘云服务器、单个基站(Base Station, BS)以及多个云机器人(Cloud Robot, CR)端。单边缘云服务器放置在基站上，基站能够和中心云以及云机器人进行信息交互。

Figure 1. System environment model
图 1. 系统环境模型

云机器人集合为 $CR=\{1,2,...,i,...,N\}$，除云机器人 i 以外的其他云机器人集合为 $CR=\{1,2,...,i-1,i+1,...,N\}$，全部云机器人任务卸载集合为 $\theta=\{\theta_i,i\in CR\}$，除云机器人 i 以外的其他云机器人任务卸载策略为 $\theta_{-i}=\{\theta_1,\theta_2,...,\theta_{i-1},\theta_{i+1},...\theta_N\}$。针对每一个云机器人 i，都对应一个云机器人计算任务 τ_i。从 τ_i 中提取出三个可划分的任务特征构成三元组 (C_i,D_i,S_i) [13], 其中 C_i 表示任务的计算量，D_i 表示任务的数据量，S_i 表示任务的安全需求等级。

当任务安全等级 $S_i=1$ 时，表示该任务对安全有一定要求,只能将任务卸载到本地计算；当任务安全等级 $S_i=0$ 时，表示该任务对安全没有要求或者对安全要求较低，可以将任务分割成子任务，并卸载到中心云、边缘云或本地的其中任何一个地方。

当任务安全等级 $S_i=0$ 时，针对每一个云机器人 $i\in CR$ 的任务，均可通过一对阈值 $(\varphi_i,\phi_i)\in TS_i$ 将任务划分为三部分，TS_i 是任务卸载阈值策略空间，具体如下：

$$TS_i=\{(\varphi_i,\phi_i)\mid 0\le\varphi_i\le\phi_i\le 1,\ i\in CR\}。$$

另外，设置云机器人的卸载策略为 ts_i，$ts_i\in TS_i$，云机器人 i 根据卸载策略将由阈值划分出的子任务分别在中心云服务器、边缘云服务器或本地运行，其中 $(0,\varphi_i)$ 表示的是本地处理的子任务，(φ_i,ϕ_i) 表示

的是卸载到 MEC 服务器上运行的子任务，$(\phi_i,1)$ 表示的是卸载到中心云服务器上运行的子任务。此时存在三种极端情况，若 $\varphi_i=\phi_i=0$，则表示云机器人 i 的任务全部卸载到中心云服务器；若 $\varphi_i=0,\phi_i=1$，则表示云机器人 i 的任务全部卸载到边缘云服务器；若 $\varphi_i=\phi_i=1$，则表示云机器人 i 的任务全部卸载到本地计算。将所有云机器人 i 的阈值划分策略放入一个集合中，将每一个云机器人的阈值全部都放入策略集合 $\theta_i=\{ts_i,i\in CR\}$ 中。

3.2. 不同卸载策略下的代价函数

3.2.1. 本地任务代价函数

当任务在本地直接执行时，计算任务产生的代价主要考虑的是在本地上运行时间 T_i^{loc} 以及本地能耗 E_i^{loc} 两个因素。按照云机器人任务 i 所对应的安全等级 S_i，将所有云机器人任务分成两种类型，第一种类型：当云机器人任务的安全等级 S_i 为 1 时，说明任务对安全要求较高，只允许任务在本地执行。此时，整个任务全部直接本地运行；第二种类型：当云机器人任务的安全等级 S_i 为 0 时，说明任务对安全要求不高甚至没有安全要求。此时，任务可以以子任务的方式卸载到中心云服务器、边缘云服务器或者本地三者中的任意一个地方执行。

本地计算时间主要是由本地机器人的自身 CPU 频率决定的。F_i^{loc} 为云机器人的自身 CPU 频率[14]，则云机器人任务在本地的计算时间如下：

$$T_i^{loc}=\begin{cases}\dfrac{C_i\varphi_i}{F_i^{loc}} & S_i=0\\ \dfrac{C_i}{F_i^{loc}} & S_i=1\end{cases} \tag{1}$$

云机器人任务在本地的计算能耗通常可以用 $\varepsilon=\eta f^2$ [14]表示，其中 η 是能耗系数，f 是 CPU 频率。则云机器人任务在本地的能耗如下：

$$E_i^{loc}=\begin{cases}C_i\eta\left(F_i^{loc}\right)^2\varphi_i & S_i=0\\ C_i\eta\left(F_i^{loc}\right)^2 & S_i=1\end{cases} \tag{2}$$

云机器人 i 在计算时间和能耗两者不同权重下的本地代价函数为：

$$J_i^{loc}\left(ts_i\right)=\lambda_i^eE_i^{loc}+\lambda_i^tT_i^{loc} \tag{3}$$

约束条件为：$0\le\lambda_i^e\le1$，$0\le\lambda_i^t\le1$，$\lambda_i^e+\lambda_i^t=1$，$\lambda_i^e$ 表示云机器人 i 对能耗的权重，λ_i^{t} 表示云机器人 i 对时间的权重。根据实际情况，当用户对能耗要求较高，或不希望能耗过多时，适当提高参数 λ_i^e 的数值占比，使得能耗对代价函数影响较大；当用户在处理对实时性要求较高的任务时，适当提高参数 λ_i^{t} 的数值占比，使得时间对代价函数影响较大。这里假定所有云机器人在处理当前任务 τ_i 时，整个处理过程中，权重设置一经开始就已固定，并不会因受到外部影响而中途发生更改。

3.2.2. MEC 服务器任务代价函数

当任务卸载到边缘云服务器上执行时，计算任务产生的代价主要考虑的是在边缘云服务器上运行时间 T_i^{mec}、上行链路的传输时延 T_i^{trans} 以及传输过程中产生的能耗 E_i^{loc} 三个因素。由于下行链路传输所返回的数据量远远小于上行链路传输输入的数据量，所以一般不考虑下行链路的时延[9] [15]。

为了更好的描述和构建代价函数，以下对上行传输速率 r_i、传输干扰 I_i 进行定义。本文采用非正交多址技术进行数据传输，构建了一个允许多云机器人在单一信道下同时进行数据传输的环境。由香农定理可知，当多云机器人任务在同时经过单一信道时，信道是上行传输速率 r_i [16]如下：

$$r_i = B \cdot \log_2\left(1+\frac{P_i h_i}{I_i+\omega_i}\right) \tag{4}$$

其中，B 为传输信道(BS)的带宽，P_i 为云机器人的上行传输功率，h_i 为云机器人与信道 BS 之间的信息增益，ω_i 是信道内部的高斯噪声功率。

针对云机器人与信道 BS 之间的信息增益 h_i，令 d_i 为云机器人 i 到边缘云服务器之间的距离，σ 为衰减系数，则有如下公式[17]：

$$h_i = d_i^{-\sigma} \tag{5}$$

当多个云机器人同时都需要卸载到非本地端时，云机器人之间会出现冲突问题，由于使用的信道数据传输技术是 NOMA 技术，故云机器人 i 的信道传输速率 r_i 会受到其他云机器人任务卸载策略的影响。对于云机器人 i 而言，令 ts_i^{off} 为有子任务在非本地端执行的决策集合，其他云机器人 j 产生的干扰 I_i [9] 如下所示：

$$I_i = \sum_{\left\{j \in HCR | ts_i \in ts_j^{off}\right\}} p_j h_j \tag{6}$$

将公式(6)代入公式(4)中，形成云机器人 i 的上行传输速率为：

$$r_i = B \cdot \log_2\left(1+\frac{P_i h_i}{\sum_{\left\{j \in HCR | ts_i \in ts_j^{off}\right\}} p_j h_j + \omega_i}\right) \tag{7}$$

云机器人任务卸载到边缘云服务器时的运行时间为：

$$T_i^{mec} = \begin{cases} \dfrac{C_i\left(\phi_i-\varphi_i\right)}{F_{mec}} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{8}$$

在公式(8)中，F_{mec} 为边缘云服务器的 CPU 频率。

云机器人任务卸载到边缘服务器的上行传输时延为：

$$T_i^{trans} = \begin{cases} \dfrac{D_i\left(\phi_i-\varphi_i\right)}{r_i} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{9}$$

云机器人任务卸载到边缘服务器所需要的能耗为：

$$E_i^{mec} = \begin{cases} \dfrac{P_i D_i\left(\phi_i-\varphi_i\right)}{r_i} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{10}$$

云机器人 i 在对应卸载决策 ts_i 下的边缘云代价函数为：

$$J_i^{mec}\left(ts_i\right) = \lambda_i^e E_i^{mec} + \lambda_i^t\left(T_i^{trans} + T_i^{mec}\right) \tag{11}$$

3.2.3. 中心云服务器任务代价函数

当任务卸载到中心云服务器上执行时，计算任务产生的代价主要考虑的是在中心云服务器上运行时间 T_i^c、上行链路的传输时延 T_i^{trans}、基站与云服务器之间的传输时延 T_i^{bc} 以及传输过程中产生的能耗 E_i^{loc} 四个因素。

设置中心云服务器的 CPU 频率为 F_c，则云机器人任务在中心云执行时间为：

$$T_i^c = \begin{cases} \dfrac{C_i(1-\phi_i)}{F_c} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{12}$$

云机器人任务卸载到中心云服务器的上行传输时延为：

$$T_i^{trans} = \begin{cases} \dfrac{D_i(1-\phi_i)}{r_c} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{13}$$

令 D_i^{back} 为云端返回到 BS 的数据量，ν 为单位时间内的数据传输时延，则云机器人在云端和基站 BS 之间的往返时延和为：

$$T_i^{bc} = \begin{cases} \left(D_i + D_i^{back}\right)(1-\phi_i)\nu & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{14}$$

云机器人任务卸载到边缘服务器所需要的能耗为：

$$E_i^c = P_i T_i^{trans} = \begin{cases} \dfrac{P_i D_i(1-\phi_i)}{r_i} & S_i = 0 \\ 0 & S_i = 1 \end{cases} \tag{15}$$

云机器人 i 在对应卸载决策 ts_i 下的中心云代价函数为：

$$J_i^c(ts_i) = \lambda_i^e E_i^c + \lambda_i^t \left(T_i^{trans} + T_i^{bc} + T_i^c\right) \tag{16}$$

3.3. 系统目标模型

系统目标模型为 $G = \{CR, \{\theta_i\}, \{J_i\}\}$，$CR$ 为云机器人集合，$\{\theta_i\}$ 为云机器人 i 的卸载策略集合，$\{J_i\}$ 为云机器人 i 总成本集合，其中 $i \in CR$。结合本地任务代价函数、MEC 服务器代价函数是和中心云服务器代价函数，得单个云机器人任务总代价模型函数为：

$$J_i(\theta_i, \theta_{-i}) = J_i^{loc}(\theta_i) + J_i^{mec}(\theta_i) + J_i^c(\theta_i) \tag{17}$$

保持其他云机器人卸载策略集合 θ_{-i} 不变，云机器人 i 的最佳卸载策略为：

$$\theta_i^{\min} = \arg\min J_i(\theta_i, \theta_{-i}) \tag{18}$$

若对于 $\forall \theta_i \in TS_i, i \in CR$，系统中总存在：$J_i\left(\theta_i^{\min}, \theta_{-i}^{\min}\right) \le J_i\left(\theta_i, \theta_{-i}^{\min}\right)$，则系统达到最小总代价此时对应的最优卸载策略集合为：

$$\theta^{\min} = \left\{\theta_1^{\min}, \theta_2^{\min}, ..., \theta_i^{\min}, ..., \theta_N^{\min}\right\} \tag{19}$$

4. 基于遗传博弈的部分任务卸载策略

基于中心云－单边缘云－本地协同的准静态场景下(准静态场景：在云机器人任务 i 执行期间，λ_i^e 和 λ_i^t 不发生变化)云机器人任务如何进行部分任务卸载是本文研究的主要问题。可以考虑使用遗传算法确定单个云机器人的最优卸载决策 $\theta_i^{\min}$，进而获得单个云机器人总代价的最小化。另外，由于文中所使用的信道传输技术是 NOMA 技术，在多任务信息传输交互时，各任务间冲突无法避免，可以使用博弈论来处理问题冲突，确定系统的卸载策略阈值集合 $\theta^{\min}$，进而降低整个系统总代价。因此，提出一种基于遗传博弈的部分任务卸载策略。通过达到纳什平衡来寻找最优卸载策略[18]。

4.1. GGT-PTO 中的遗传算法

遗传算法是一种求解复杂系统优化问题的启发式算法，其主要组成部分包括选择操作，交叉操作，变异操作。由于遗传算法不依赖于问题本身的具体领域，对不同问题都有较强的鲁棒性，因此被广泛应用于各种学科和研究领域。

遗传算法的经典应用领域是函数优化。基于本实验模拟场景下的公式(18)本质上是一个带约束的函数优化问题，对于函数优化问题，考虑采用经典遗传算法进行处理：

定义 1：经典遗传算法的染色体编码方式采用二进制编码，并将单个云机器人的系统总代价的倒数作为适应度函数：

$$\begin{aligned} &\max\ 1/J_i\left(\theta_i,\theta_{-i}\right) \\ &subject\ to\ \varphi_i-\phi_i\le 0 \\ &0\le\varphi_i\le 1 \\ &0\le\phi_i\le 1 \end{aligned} \tag{20}$$

遗传算法的选择操作采用轮盘赌方式，交叉操作采用两点交叉方式，交叉概率设置为 0.7。对于算法终止条件，当本实验达到迭代卸载代数或者当适应度函数前后两轮误差小于所设置的允许误差值时，则认为算法终止。另外，本实验中设置的约束为 $\varphi_i-\phi_i\le 0$ 以及云机器人两个阈值 ϕ_i、φ_i 的取值范围均为 $[0,1]$。

4.2. GGT-PTO 中的博弈性质

已知势博弈具有有限改进特性[19]，并且作为博弈的一种，势博弈拥有纳什平衡状态，因此，给出本实验下的势博弈概念。

定义 2：若云机器人系统目标模型 $G=\{CR,\{\theta_i\},\{J_i\}\}$ 模型中存在一个势函数 $\psi(i)$ 满足当 $\theta\to\theta^{\min}$ 时，对 $\forall i\in CR$ 并且 $\theta_i,\theta_i^{\min}\in\{\theta_i\}$，若有 $J_i\left(\theta_i^{\min},\theta_{-i}\right)\le J_i\left(\theta_i,\theta_{-i}\right)$，则有 $\psi\left(\theta_i^{'},\theta_{-i}\right)\le\psi\left(\theta_i,\theta_{-i}\right)$，则称这个博弈为势博弈[19]。

定义 3：考虑到势博弈具有有限改进特性，多玩家博弈游戏参与者策略由 θ_i 更新为 $\theta_i^{\min}$ 是代价减小策略优化的过程，即 $J_i\left(\theta_i^{\min},\theta_{-i}\right)\le J_i\left(\theta_i,\theta_{-i}\right)$，则可将有限的势函数描述为：

$$\psi\left(\theta_i\right)=\sum_{i=1}^{N}J_i\left(\theta_i,\theta_{-i}\right) \tag{21}$$

系统的卸载更新策略公式为

$$\Delta\psi(i)=\psi\left(\theta_i\right)-\psi\left(\theta_i^{\min}\right) \tag{22}$$

$\Delta\psi(i)$ 的值越大则说明云机器人更新策略的机会越大，因此需找出 $argmax\Delta\psi(i),i\in CR$ 的云机器人 i，并将最佳卸载策略更新为对应的策略 $\theta_i^{\min}$，在经过有限的更新迭代后博弈达到 NE 状态，并可以找到纳什平衡策略集合 $\theta^{\min}$。

4.3. GGT-PTO 卸载策略的设计

本文设计实现了云边端协同下基于遗传博弈论的部分任务卸载策略算法。为了充分考虑云机器人任务细粒度划分后部分卸载所带来的成本优化，研究选用两个阈值对任务进行粒化分割；为了获得单个云机器人的最优卸载策略阈值，研究选用遗传算法进行随机搜索；最后通过模拟多云机器人任务经过同一信道卸载的博弈，找寻卸载策略的纳什平衡状态，进而找到最小的系统代价。

GGT-PTO 算法核心设计流程：第一步，遍历所有云机器人任务的安全等级，将安全等级较高的任务强制设置为直接本地执行(后期不可更改)，以此保证系统的安全性，并将安全等级较低的任务初始化为本

地执行，即将云机器人任务卸载策略集合初始化为$\theta=\{\theta_i=(1,1),i\in CR\}$；第二步，在其他云机器人任务卸载策略$\theta_{-i}$不变时，使用遗传算法依次更新云机器人$i$的卸载策略$\theta_i^{\min}$，通过对比，寻找出差值变化最大的$\arg\max\Delta\psi(i)$所对应的云机器人$i$，并将找到的云机器人$i$的最优卸载策略$\theta_i^{\min}$更新入云机器人卸载策略集合$\theta$中；第三步，根据更新后的云机器人任务卸载策略集合$\theta$更新所有云机器人任务卸载上行传输速率$r_i$。不断重复以上三个步骤，依次找到每个云机器人的最佳卸载策略，最终更新的云机器人卸载策略集合θ就是纳什平衡策略集合$\theta^{\min}$。具体GGT-PTO算法设计如下所示：

输入：云机器人CR，云机器人任务τ_i，信道带宽B，边缘云服务器CPU频率F_{mec}，中心云服务器CPU频率F_c，上行链路传输功率P_i，BS与中心云间的单位数据传输时延ν，BS与中心云间任务响应的数据量D_i^{back}等初始数据。

输出：云机器人最优卸载策略集合$\theta^{\min}$，系统最小的总成本代价$\sum_{i=1}^{N}J_i\left(\theta^{\min}\right)$

初始化：每个云机器人均本地执行，云机器人策略集合$\theta=\{\theta_i=(1,1),i\in CR\}$。

while：

for $i=1$ to N do:

使用遗传算法所有$\arg\min J_i(\theta_i,\theta_{-i})$；

end for

将所有云机器人任务总代价求和得到系统总代价：$\psi(\theta_i)=\sum_{i=1}^{N}J_i(\theta_i,\theta_{-i})$；

计算所有云机器人任务改变策略阈值前后的差值：$\Delta\psi(i)=\psi(\theta_i)-\psi(\theta_i^{\min})$；

寻找值最大的$\arg\max\Delta\psi(i)$所对应的云机器人任务i，将云机器人任务i此时的卸载策略更新入卸载策略集合中；

根据新卸载策略集合θ更新信道传输速率r_i；

continue;

if $\arg\max\Delta\psi(i)\le 0$:

当前策略就是最优策略集合，根据最优策略集合计算系统总成本代价$\sum_{i=1}^{N}J_i\left(\theta^{\min}\right)$；

return $\theta^{\min}$；

end if

每进行一轮循环，就会使用N次遗传算法，将其中一个云机器人i对应的最佳卸载策略$\theta_i^{\min}$确定，经过有限次循环后，定能达到纳什平衡，找到最佳的云机器人卸载策略集合$\theta^{\min}$。

5. 实验仿真

5.1. 实验设置

本实验利用Python语言进行仿真设计，模拟的系统环境采用的是光纤无线混合网络下中心云－单边缘云－本地协同服务系统，针对光纤无线通信参数和服务器CPU频率、信道等参数的设置，本文实验主要参考[9] [16] [20]，具体可参考下表1。

Table 1. Simulation experiment parameters
表1. 仿真实验参数

序号	主要参数	量值
1	信道带宽B	10MHz
2	上行传输功率P_i	100 mW

Continued

3	能耗系数 η	5×10^{-26}
4	信道的噪声功率 ω_0	−100 dBm
5	边缘 CPU 频率 F_{mec}	5GHz
6	中心 CPU 频率 F_c	20GHz
7	BS 与中心云间的单位数据传输时延 ν	1 μs/bit
8	BS 与中心云间任务响应的数据量比 D_i^{back}	5%

为了保证模拟实际情况中，每个云机器人的功能不同，所以云机器人所需要解决的任务自然也是不相同的，本文通过提取任务可划分的特征[20]：任务计算量 C_i 的随机分配从 200 MC 到 500 MC，任务数据量 D_i 的随机分配从 500 KB 到 1000 KB，任务安全等级 S_i 的随即数值是 0 或 1。

5.2. 实验分析

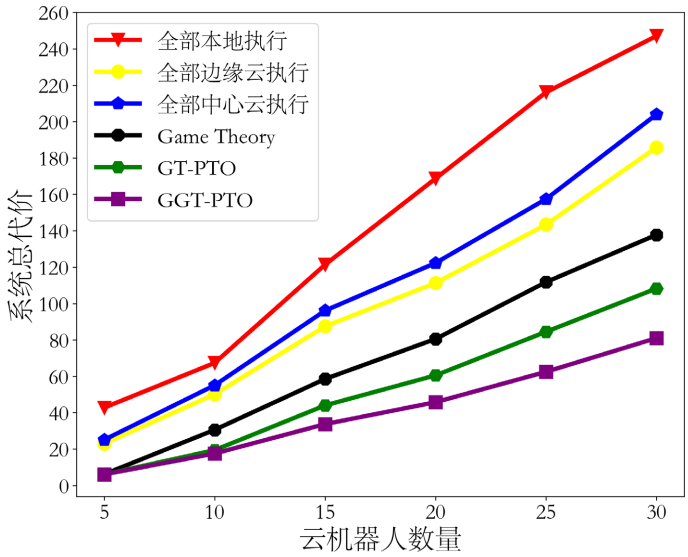

Figure 2. Comparison diagram of total system cost under different unloading algorithms
图 2. 不同卸载算法下的系统总代价比较图

图 2 是多云机器人在云边端协同下不同卸载算法的系统总代价随云机器人个数的变化对比图。首先，从总体趋势上看，随着云机器人数量的增加，所有卸载算法的系统总代价都会随之增加。其次，对比不同的卸载算法，针对三种全部卸载方式，性能由低到高依次为：任务全部本地执行、任务全部卸载到中心服务器、任务全部卸载到边缘服务器。可以看出，将任务卸载到远端，能有效的降低系统的总代价，并且将任务全部卸载到边缘云服务器上的总代价优于全部卸载到中心云服务器上的总代价，究其原因，是因为相比中心云，边缘云低延时、更节能、高效率等特点更具有优势。前面三种方法都是直接将任务进行全部卸载，每种方法仅利用中心云、边缘云或本地设备的其中一种计算资源，无法充分发挥三种计算资源的全部优势，所以引入传统的 Game Theory 算法，通过该算法将任务卸载到指定位置的方式，有效降低系统代价。前面四种方法都是二进制卸载模式，没有考虑将任务细粒化而带来的成本优化，所以姜春茂等[21]提出的一种基于博弈论的部分任务卸载算法(GT-PTO)，该算法有效的降低系统的总代价，但亦存在一定的局限，由于卸载策略阈值取值范围是一个连续区间，而博弈论的使用需要通过设置步长的方式将连续区间转变成一系列离散值，这样一来，极有可能丢失真正最优的卸载策略阈值，使得最后

找到的最优卸载策略集合保真性较低。因此，本文提出了 GGT-PTO 算法，明确引入两个阈值对任务进行粒化分割，并融合遗传算法来改进博弈过程，试图在保证算法可解释性不下降的前提下，提高最终所找到的最佳卸载策略阈值集合的保真性，进而使得云边端协同服务的整体性能显著提升。如图 2 可知，GT-PTO 算法明显优于各种二进制卸载方式，完美体现了 GT-PTO 算法粒化分割对成本的优化是十分显著的，而相比于 GT-PTO 算法，本文 GGT-PTO 算法进一步引入遗传算法，提高了所找到的最佳卸载策略阈值集合的保真性，进一步的优化、无限逼近最优策略，使得系统的整体性能进一步提升。

综上所述，相比原有的全部二进制任务卸载算法、基于传统 Game Theory 的二进制任务卸载算法以及基于传统 Game Theory 的部分任务卸载算法(GT-PTO)，本文提出的 GGT-PTO 算法，能显著降低系统总代价，极大提升云边协同的整体性能。

图 3 是 GGT-PTO 算法中引入不同的遗传算法下产生的代价在个数不同的云机器人条件下的对比图。图 3 有 6 个子图，按照从左到右从上到下的顺序，每个子图的云机器人个数依次是 5 个、10 个、15 个、20 个、25 个和 30 个。每一个子图对比了 GGT-PTO 算法中引入五种不同遗传算法下而产生的系统代价。子图间对比可知，随着云机器人任务个数增加，无论引入哪种遗传算法，系统的总成本都随之上升。另外，根据引入五种不同遗传算法后 GGT-PTO 算法的性能表现，可知引入经典遗传算法可使得 GGT-PTO 算法的性能最好，其结果恰恰体现了 Occam’s Razor 教授提出的奥卡姆剃刀原则的精髓，因此，在后续实验中，为使得 GGT-PTO 算法表现更好，GGT-PTO 算法均选择引入经典遗传算法。

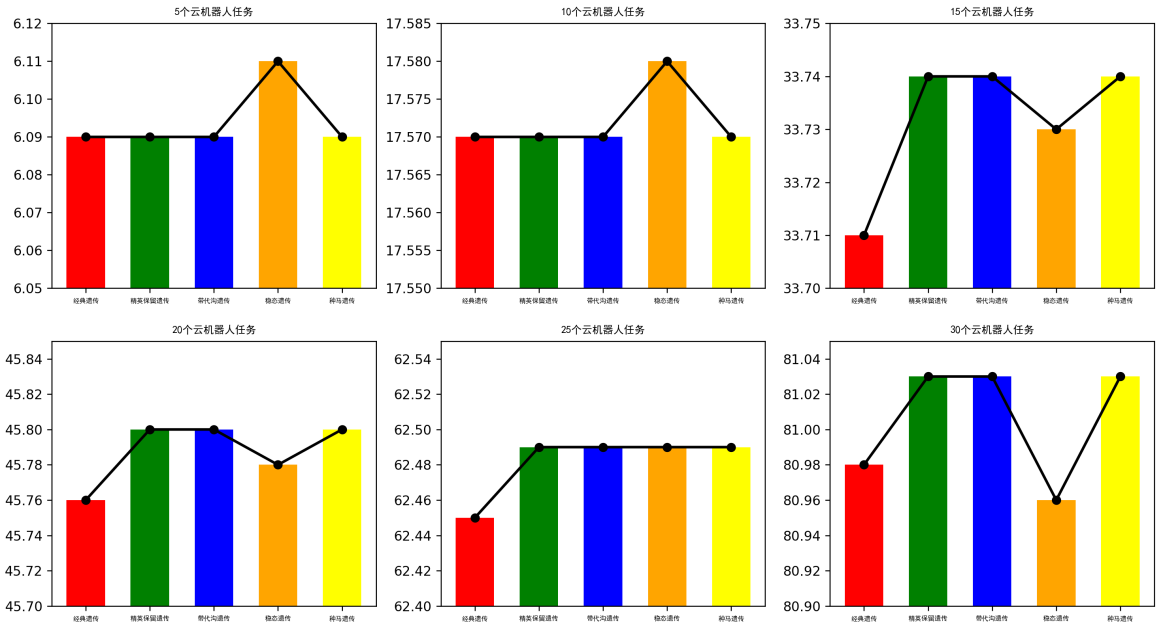

Figure 3. The total cost of GGT-PTO algorithm with different genetic algorithms under different tasks
图 3. 不同任务数量下引入不同遗传算法的 GGT-PTO 算法的总成本

图 4、图 5 和图 6 是 GGT-PTO 算法中本地 CPU 频率、边缘云服务器 CPU 频率、中心云服务器 CPU 频率的变化对系统总代价的影响实验。图 4 是不同本地 CPU 频率下对系统总代价的影响，设置本地 CPU 频率依次为 0.5 GHz、0.7 GHz、1.0 GHz。在其他条件不发生改变时，改变本地 CPU 频率，可以发现，随着本地 CPU 频率的不断提升，系统的总代价也随之变大。究其原因，应是随着本地 CPU 频率的不断提升，达到纳什平衡状态时所得到的阈值卸载策略会因为本地 CPU 频率变强而向本地卸载更多的任务量，而随着本地卸载任务量增多，进而就会导致总代价的增加。

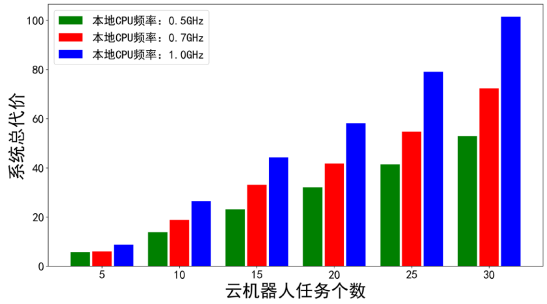

Figure 4. The impact of different local CPU frequencies on the total cost of the system
图 4. 不同本地 CPU 频率下对系统总代价的影响

图 5 是不同中心云 CPU 频率对系统总代价的影响，设置中心云服务器的 CPU 频率依次为 15 GHz、20 GHz、25 GHz。可以发现，随着中心云服务器 CPU 频率的不断提升，系统的总代价逐渐降低。究其原因，应是随着中心云服务器 CPU 频率的不断提升，达到 NE 状态时所得到的阈值卸载策略集合会因为中心云服务器的 CPU 频率变强而向中心云服务器卸载更多的任务量，而随着中心云服务器上卸载的任务量增多，就会导致总代价的有所降低。

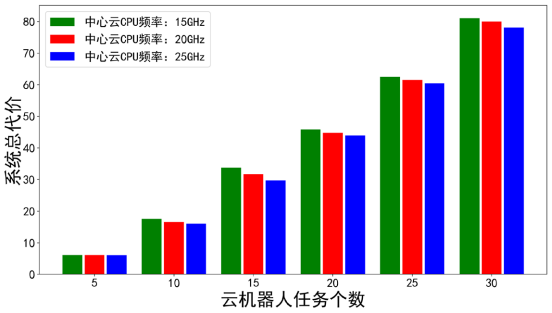

Figure 5. The impact of different central cloud CPU frequencies on the total system cost
图 5. 不同中心云 CPU 频率下对系统总代价的影响

图 6 是不同边缘云 CPU 频率对系统总代价的影响，设置边缘云服务器的 CPU 频率依次为 5 GHz、7 GHz、10 GHz。可以发现，边缘云和中心云比较相似，随着边缘云服务器 CPU 频率的不断提升，系统的总代价逐渐降低。究其原因，应是随着边缘云服务器 CPU 频率的不断提升，达到 NE 状态时所得到的阈值卸载策略集合会因为边缘云服务器的 CPU 频率变强而向中心云服务器卸载更多的任务量，而随着边缘云服务器上卸载的任务量增多，就会导致总代价的有所降低。

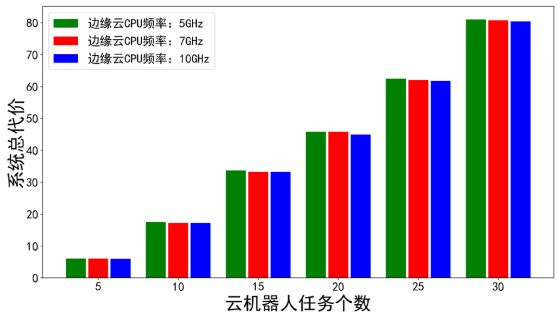

Figure 6. The impact of different edge cloud CPU frequencies on the total system cost
图 6. 不同边缘云 CPU 频率下对系统总代价的影响

综合图 4、图 5 和图 6 所展示的结果，在实验参数条件下，本地 CPU 频率的降低、中心云和边缘云服务器 CPU 频率的提升均可以使得系统总代价降低，相反，本地 CPU 频率的提升、中心云和边缘云服务器 CPU 频率的下降均可以使得系统总代价上升。

图 7、图 8、图 9 分别是 TGGT-PTO 算法下不同时间与能耗权重分别对任务完成时间、能耗、总的对比图。图 7 中观察的是权重对任务完成时间的影响，设置了五种情况进行对比，分别是 0.1:0.9、0.3:0.7、0.5:0.5、0.7:0.3、0.9:0.1。从整体上观察，在本实验参数条件下，随着云机器人数量增加，系统时延总代价也随之增加。从相同云机器人数量角度上看，随着能耗权重的减少、时间权重的增加，系统任务完成时间会增加。相反，随着能耗权重的增加、时间权重的减少，系统任务完成时间会减少。

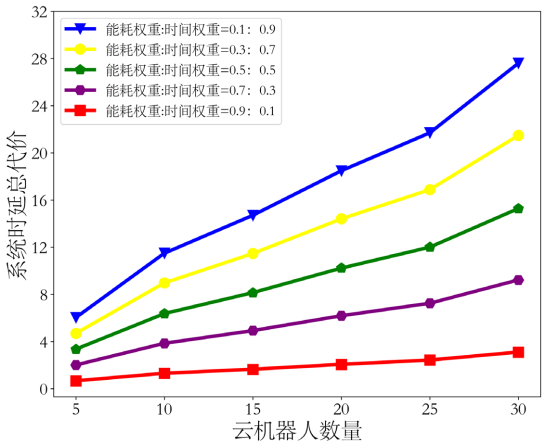

Figure 7. The effect of the energy/time weights on task completion time
图 7. 能耗/时间权重对任务完成时间的影响

图 8 中观察的是权重对系统能耗的影响，设置了五种情况进行对比，分别是 0.1:0.9、0.3:0.7、0.5:0.5、0.7:0.3、0.9:0.1。从整体上观察，在本实验参数条件下，随着云机器人数量的增加，系统能耗总代价也随之增加。从相同云机器人数量角度上看，随着能耗权重的增加、时间权重的减少，系统任务完成时间会增加，相反，随着能耗权重的减少、时间权重的增加，系统任务完成时间会减少。

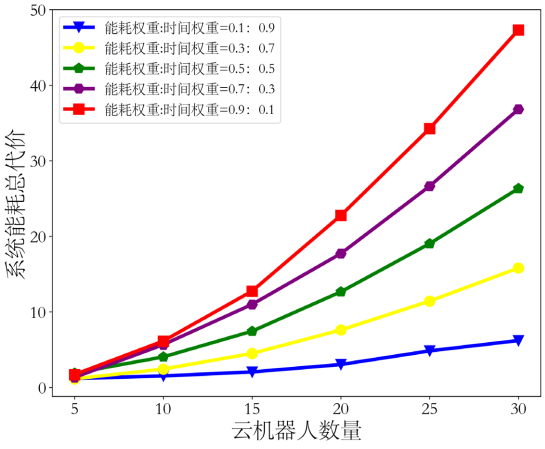

Figure 8. The effect of energy/time weighting on energy consumption
图 8. 能耗/时间权重对能耗的影响

图 9 中观察的是权重对系统总代价的影响，仍然是设置了五种情况进行对比，分别是 0.1:0.9、0.3:0.7、

0.5:0.5、0.7:0.3、0.9:0.1。从整体上观察，在本实验参数条件下，随着云机器人数量的增加，系统总代价也随之增加。在云机器人任务数量在 17 到 18 个时，系统存在临界值，当云机器人任务数量小于临界值时，能耗权重越大、时间权重越小，系统总代价越小；当云机器人任务数量大于临界值时，能耗权重越大、时间权重越小，系统总代价越大。

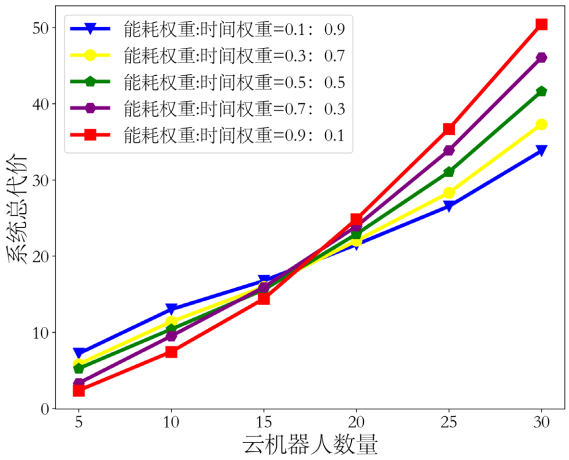

Figure 9. The effect of energy consumption time weighting on the total cost of the system
图 9. 能耗时间权重对系统总代价的影响

6. 结束语

本文研究了多云机器人在云边端协同环境下的任务卸载策略问题，将任务完成时间和系统能耗作为衡量系统代价的两个指标，根据云机器人接收到的实际任务需求偏好，设置任务完成时间和系统能耗的权重值，将寻找最佳卸载策略作为核心问题。为了最大程度降低系统总代价，本文提出了一种云边端协同下基于遗传博弈的部分任务卸载策略算法(GGT-PTO)。该算法使用两个阈值对实际任务进行粒化分割，将二进制任务卸载方式转变为部分任务卸载方式，充分考虑云机器人任务细粒度划分后，部分卸载方式所带来的成本优化。另外，将遗传算法引入传统博弈论中，避免传统博弈中，因设置步长而导致的最优值丢失问题，利用遗传算法无限逼近最优值，极大的降低了系统总代价。仿真实验表明，在当前中心云－单边缘云－本地协同环境下，相比原有的全部二进制任务卸载算法、基于传统 Game Theory 的二进制任务卸载算法以及基于传统Game Theory的部分任务卸载算法(GT-PTO)，GGT-PTO算法的表现更加良好，能够进一步降低系统的总代价。同时，本算法仍存在一定的局限性。环境方面的局限：基于单信道、单边缘云服务器的研究虽然具有一定的研究意义，但显然不够复杂，难以应对模拟真实的应用场景。总代价最优的保真性局限：使用遗传算法，只能无限逼近最优解，但仍然不是真正的最优解。未来研究的方向可以考虑在更加复杂的多信道、多 MEC 环境下使用该算法。或者尝试继续优化当前算法，使结果进一步逼近系统最优解，以此降低系统总代价，进而提高云边端协同服务的整体性能。

参考文献

[1] Miratabzadeh, S.A., Gallardo, N., Gamez, N., *et al*. (2016) Cloud Robotics: A Software Architecture: For Heterogeneous Large-Scale Autonomous Robots. 2016 *World Automation Congress* (*WAC*), Puerto Rico, 31 July-4 August 2016, 1-6. https://doi.org/10.1109/WAC.2016.7583017

[2] Saha, O. and Dasgupta, P. (2018) A Comprehensive Survey of Recent Trends in Cloud Robotics Architectures and Applications. *Robotics*, **7**, Article 47. https://doi.org/10.3390/robotics7030047

[3] Abbas, N., Zhang, Y., Taherkordi, A. and Skeie, T. (2017) Mobile Edge Computing: A Survey. *IEEE Internet of*

Things Journal, **5**, 450-465. https://doi.org/10.1109/JIOT.2017.2750180

[4] Mao, Y., You, C., Zhang, J., *et al*. (2017) A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, **19**, 2322-2358. https://doi.org/10.1109/COMST.2017.2745201

[5] Ahmed, E. and Rehmani, M.H. (2016) Mobile Edge Computing: Opportunities, Solutions, and Challenges. *Future Generation Computer Systems*, **70**, 59-63. https://doi.org/10.1016/j.future.2016.09.015

[6] Dai, Y., Xu, D., Maharjan, S. and Zhang, Y. (2018) Joint Computation Offloading and User Association in Multi-Task Mobile Edge Computing. *IEEE Transactions on Vehicular Technology*, **67**, 12313-12325. https://doi.org/10.1109/TVT.2018.2876804

[7] Li, S., Li, B. and Zhao, W. (2020) Joint Optimization of Caching and Computation in Multi-Server NOMA-MEC System via Reinforcement Learning. *IEEE Access*, **8**, 112762-112771. https://doi.org/10.1109/ACCESS.2020.3002895

[8] Guo, H. and Liu, J. (2018) Collaborative Computation Offloading for Multiaccess Edge Computing over Fiber-Wireless Networks. *IEEE Transactions on Vehicular Technology*, **67**, 4514-4526. https://doi.org/10.1109/TVT.2018.2790421

[9] Zhou, S. and Waqas, J. (2020) The Partial Computation Offloading Strategy Based on Game Theory for Multi-User in Mobile Edge Computing Environment. *Computer Networks*, **178**, Article ID: 107334. https://doi.org/10.1016/j.comnet.2020.107334

[10] Wu, B., Zeng, J., Ge, L., *et al*. (2019) A Game-Theoretical Approach for Energy-Efficient Resource Allocation in MEC Network. *ICC* 2019: 2019 *IEEE International Conference on Communications* (*ICC*), Shanghai, 20-24 May 2019, 1-6. https://doi.org/10.1109/ICC.2019.8761727

[11] Liang, Z., Liut, Y., Huang, K. and Lok, T.M. (2019) I/O Interference Aware Multiuser Computation Offloading for Virtualized Edge Computing. *ICC* 2019: 2019 *IEEE International Conference on Communications* (*ICC*), Shanghai, 20-24 May 2019, 1-6. https://doi.org/10.1109/ICC.2019.8762042

[12] Balasubramanian, V., Wang, M., Reisslein, M., et al. (2019) Edge-Boost: Enhancing Multimedia Delivery with Mobile Edge Caching in 5G-D2D Networks. 2019 *IEEE International Conference on Multimedia and Expo* (*ICME*), Shanghai, 8-12 July 2019, 1684-1689. https://doi.org/10.1109/ICME.2019.00290

[13] 吴学文, 廖婧贤. 云边协同系统中基于博弈论的资源分配与任务卸载方案[J]. 系统仿真学报, 2022, 34(7): 1468-1481.

[14] Chen, X. (2014) Decentralized Computation Offloading Game for Mobile Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, **26**, 974-983. https://doi.org/10.1109/TPDS.2014.2316834

[15] Hu, J., Li, K., Liu, C. and Li, K.Q. (2020) Game-Based Task Offloading of Multiple Mobile Devices with QoS in Mobile Edge Computing Systems of Limited Computation Capacity. *ACM Transactions on Embedded Computing Systems*, **19**, 1-21. https://doi.org/10.1145/3398038

[16] Fei, X., Wy, A. and He, L.A. (2020) Computation Offloading Algorithm for Cloud Robot Based on Improved Game Theory. *Computers & Electrical Engineering*, **87**, Article ID: 106764. https://doi.org/10.1016/j.compeleceng.2020.106764

[17] Yang, L., Cao, J., Cheng, H. and Ji, Y.S. (2015) Multi-User Computation Partitioning for Latency Sensitive Mobile Cloud Applications. *IEEE Transactions on Computers*, **64**, 2253-2266. https://doi.org/10.1109/TC.2014.2366735

[18] 杨卫霞. 基于博弈论的云机器人计算卸载策略研究[D]: [硕士学位论文]. 西安: 西安工业大学, 2020.

[19] 王艺洁, 凡佳飞, 王陈宇. 云边环境下基于博弈论的两阶段任务迁移策略[J]. 计算机应用, 2021, 41(5): 1392-1398.

[20] Ren, J., Yu, G., He, Y. and Li, G.Y. (2019) Collaborative cloud and Edge Computing for Latency Minimization. *IEEE Transactions on Vehicular Technology*, **68**, 5031-5044. https://doi.org/10.1109/TVT.2019.2904244

[21] 姜春茂, 杨振兴. 云边协同下基于博弈论的云机器人部分任务卸载策略[J]. 系统仿真学报, 2023, 35(5): 987-997.