计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200900079

移动边缘计算中的动态用户分配方法

唐文君　刘　岳　陈　荣
大连海事大学信息科学技术学院　辽宁 大连 116026
(wjtang@dlmu.edu.cn)

摘　要　在边缘计算环境中,为用户匹配合适的服务器是一个关键问题,可以有效提升服务质量。文中将边缘用户分配问题转换为一个受距离和服务器资源约束的二分图匹配问题,并将其建模为一个0-1整数规划问题进行优化。在离线状态下,基于精确式算法的优化模型可以求得最优分配策略,但其求解时间过长,无法处理规模较大的数据,不适用于现实服务环境。因此,提出了基于启发式策略的在线分配方法,以在时间有限的情况下优化用户-服务器的分配。实验结果显示,基于近邻启发式的在线方法的竞争比能够接近100%,可以在可接受的时间范围内求得较优的分配解。同时,近邻启发式方法比其他基础启发式方法的表现更优秀。

关键词:边缘计算;计算卸载;边缘用户分配;二分图匹配;启发式方法
中图法分类号　TP311.5

User Allocation Approach in Dynamic Mobile Edge Computing

TANG Wen-jun,LIU Yue and CHEN Rong
Department of Information Science and Technology,Dalian Maritime University,Dalian,Liaoning 116026,China

Abstract　In edge computing environment,matching suitable servers for users is a key issue,which can effectively improve the quality of service. In this paper,the edge user assignment (EUA) problem is converted into a bipartite graph matching problem constrained by distance and server resources,and it is modeled as a 0-1 integer programming problem for optimal assignment solution. In the offline state,the optimization model based on exact algorithm can obtain the optimal assignment strategy,but its solution time is too long,and it cannot process large-scale of data,which is not suitable for the real service environment. Therefore,the online user assignment method based on heuristic strategy is proposed to optimize the user-server assignment under limited time. The experimental results show that the competitive ratio obtained by Proximal Heuristic online method (PH) can reach close to 100%,and can obtain a better assignment solution within an acceptable time. At the same time,the online PH method performs better than other basic heuristic methods.

Keywords　Edge computing,Computing offloading,Edge user allocation,Bipartite graph matching,Heuristic method

1　引言

智能手机、穿戴式设备等移动设备终端的普及以及5G通信的出现,促进了物联网的产生和发展。然而,移动设备的计算能力、存储能力都有限,这就产生了一种新的服务提供方法——移动云计算(Mobile Cloud Computing,MCC)。它通过无线网络将计算任务上传到云设置中,来替代或辅助移动设备为用户提供服务。但在物联网中,用户有高带宽、低延迟的服务需求[1],而云端服务器通常设在一个或几个位置,其距离用户较远时,会带来通信延迟问题,影响用户的交互式体验。因此,在2014年出现了移动边缘计算(Mobile Edge Computing,MEC)的概念[2]。MEC通常以分布式方式部署服务器,可将服务和资源移动到网络的边缘来提高服务能力。

边缘计算需要频繁地将用户的需求任务迁移到边缘服务器来计算,且在现实应用环境中,用户的数量较多,因此为用户匹配合适的服务器成为边缘计算中的一个重要研究问题[3]。其研究如何为用户制定合理的分配策略,将其计算任务卸载到合适的边缘服务器中,来低延迟、高效率地达成用户需求。

在MEC环境中,每个边缘服务器都有自己的覆盖范围,

到稿日期:2020-09-09　返修日期:2020-12-07　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61672122,61902050,61602077);中央高校基本科研业务费专项基金(3132019355);赛尔创新项目(NGII20190627);中国博士后科学基金(2020M670736)
This work was supported by the National Natural Science Foundation of China(61672122,61902050,61602077),Fundamental Research Funds for the Central Universities of Ministry of Education of China(3132019355),ERNET Innovation Project (NGII20190627) and China Postdoctoral Science Foundation (2020M670736).
通信作者:陈荣(rchen@dlmu.edu.cn)

只能对其覆盖半径大小内的用户提供服务，因此在分配计算任务时，有必要考虑距离约束；同时，服务器的资源情况（如带宽、内存、CPU 处理能力等）都会影响任务的计算，在卸载中需要考虑服务器提供的资源能否达到用户的工作量需求，也就是容量约束[4]。Lai 等将这种边缘用户分配问题称为 EUA (Edge User Allocation)问题[5]。

在文献[5]中，EUA 问题被建模为一个可变大小的装箱(Variable Sized Vector Bin Packing，VSVBP)问题，其有两个优化目标：1)最大化服务的用户数；2)最小化雇用的服务器数，并使用集中的线性规划来求解该问题。这种分配策略优化方法有较好的效果，但在现实应用环境中用户和服务器的数量庞大，随着问题规模的复杂化，优化问题会变得更加复杂且求解耗时。同时，考虑到用户会随时加入或离开，当用户发生变化时需要及时求得新的分配策略，因此这种优化模型仅适用于离线状态的 EUA。

为了解决以上问题，本文提出了一种考虑用户动态加入或离开当前环境的 EUA 模型，该模型致力于解决更贴合实际的 EUA 问题。本文将 EUA 问题建模为一个二分图匹配问题，在优化两个目标的同时，求解得到能够满足距离约束和容量约束的分配策略。本文同时提出了基于文献[5]中方法的离线版本和基于启发式策略的在线版本的分配方法，并基于现实和仿真的合成数据进行实验，对求解时间进行了对比。本文使用竞争比来评价基于近邻启发式策略的在线方法分配解的可行性。同时，将基于近邻的启发式方法与其他基础启发式方法进行对比，验证了其解的优越性。

2 EUA 问题模型

2.1 EUA 问题描述

图 1 给出了一个 EUA 问题实例，其中包含 4 个服务器和 6 个用户，虚线圆圈为以服务器为中心的服务覆盖范围。假设每个用户的工作量为〈1，1，0.5，4〉，用户 3 可以被分配给服务器 1，也可以被分配给服务器 3，这些都是符合条件的分配策略，但我们无法确定哪种是更优的分配策略。

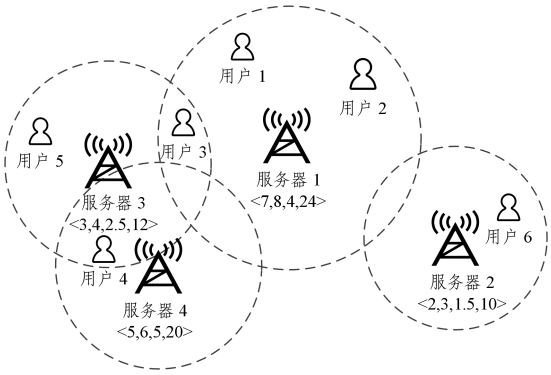


图 1 EUA 问题实例

Fig. 1 Example of EUA problem

为了选择最合适的分配策略，我们对 EUA 问题进行了建模。假设在一个 EUA 环境中，有 m 个边缘服务器 $S=\{s_1, s_2, \cdots, s_m\}$ 和 n 个用户 $U=\{u_1, u_2, \cdots, u_n\}$。用 s_i 和 u_j 分别表示其中的一个边缘服务器和用户。

(1)边缘服务器。每个服务器 s_i 都包含属性$\langle L_i, R_i, \vec{\boldsymbol{C}}_i\rangle$。前两个属性与距离约束相关，分别为服务器的位置和其覆盖半径。$\vec{\boldsymbol{C}}_i$与容量约束有关，为一个表示边缘服务器 s_i 可提供的服务资源情况的向量。我们定义其形式为$\vec{\boldsymbol{C}}_i=\langle \boldsymbol{C}_i^1, \boldsymbol{C}_i^2, \cdots, \boldsymbol{C}_i^d\rangle$，其中的每个元素代表一种计算资源的容量，如 CPU、带宽等，d 为资源的数量。

(2)用户。每个用户 u_j 包含与距离约束相关的属性，即用户位置 l_j。完成用户的计算任务需要占用一定的服务器的资源，我们将其定义为用户任务的工作量，用向量$\vec{\boldsymbol{w}}_i$表示，$\vec{\boldsymbol{w}}_i=\langle \boldsymbol{w}_j^1, \boldsymbol{w}_j^2, \cdots, \boldsymbol{w}_j^d\rangle$，其中的每个元素代表用户需求在某一类资源上的工作资源的占用量。

2.2 EUA 问题的定义

根据 EUA 问题的描述，给出服务器-用户分配的定义。

定义 1(二分图分配问题) 给定一个二分图 $G=(U, S, E)$，U 为所有用户顶点的集合，S 为服务器顶点集合，在每个时刻 t_k，将 U 中的元素 u_j 与 S 中的元素 s_i 进行匹配，这种匹配关系 A_{ij} 用二元组 $A_{ij}=[s_i, u_j]$表示，每个 A_{ij} 构成二分图中的一条边，所有的匹配边构成边集 E。

图 2 给出了一个包含 4 个用户和 3 个服务器的二分图分配实例。其中，用户 u_1 和 u_4 分别分配给服务器 s_1 和 s_3，而 s_2 要为用户 u_2 和 u_3 提供服务。

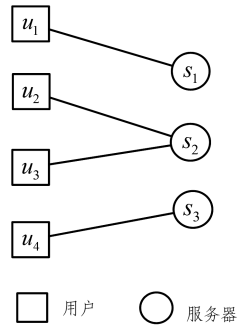


图 2 二分图分配的实例

Fig. 2 Example of bipartite graph

在 EUA 中，我们要尽可能地找到二分图中的最优分配解，即达成计算任务卸载中的两个目标——最大化服务的用户数和最小化雇用的服务器数的分配策略。两个目标的公式如下：

$$\text{maximize} \sum_{u_j \in U} |A_{ij}| \tag{1}$$

$$\text{minimize} \ |s_i \in S| \tag{2}$$

除上述目标外，为了保证提供服务的质量，还要考虑距离和容量约束。

服务器 s_i 和用户 u_j 之间的距离 $D(s_i, u_j)$用欧氏距离表示，其根据用户和服务器的地理位置的经纬度进行计算。由于服务器只会向特定半径内的用户提供服务，因此需要考虑以下距离约束：

$$D(s_i, u_j) \leqslant R_i \tag{3}$$

同时，每个服务器提供的服务资源是有限的，而用户的工作需要占用一定数量的资源，当资源不足时，服务器无法向用户提供服务。因此，需要考虑服务器的容量约束，对于工作量和资源向量中的每个元素 $\forall\, w_j^d \in \vec{w}_i$，$\forall\, C_i^d \in \vec{C}_i$，需要满足式(4)所示的约束：

$$\sum_{u_j \in A_{ij}} w_j^d \leqslant C_i^d \tag{4}$$

本文定义的 EUA 问题如定义 2 所示。

定义 2(EUA 问题) EUA 问题是一个以式(1)和式(2)为分配目标,且满足式(3)、式(4)中约束的二分图匹配问题,它致力于在以用户和边缘服务器为结点的二分图中,找到最符合目标和约束条件的所有分配策略 A_{ij}($\forall i \in [1,m], \forall j \in [1,n]$)。

3 EUA 问题的分配方法

3.1 离线方法

在离线方法中,所有用户和边缘服务器的先验知识都是已知的,我们可以使用精确式算法对分配策略进行优化求解。基于文献[5]中的模型,我们将 EUA 问题规约为一个 0-1 整数规划问题。引入两个二元变量:变量 x_{ij} 表示分配情况,若 $x_{ij}=1$,则表示将用户 u_j 分配给边缘服务器 s_i,$x_{ij}=0$ 则相反;变量 $y_i=1$ 表示服务器 s_i 被雇用,$y_i=0$ 时则相反。

在离线状态下的 EUA 中,根据 EUA 问题的定义,我们的目标为:

$$\text{maximize} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \tag{5}$$

$$\text{minimize} \sum_{i=1}^{m} y_i \tag{6}$$

因为变量 x_{ij} 和 y_i 只能取值 0 和 1,所以式(5)和式(6)能反映服务的用户数量和雇用的服务器数量。

同时,该整数规划模型受式(7)—式(9)的约束。

$$\sum_{i}^{m} x_{ij} \leqslant 1, \forall u_j \in U \tag{7}$$

$$D(s_i, u_j) \cdot x_{ij} \leqslant R_i \cdot y_i \tag{8}$$

$$\sum_{j}^{n} x_{ij} \cdot w_j^d \leqslant C_i^d, \forall w_j^d \in \vec{w}_i, \forall C_i^d \in \vec{C}_i \tag{9}$$

3.2 在线方法

离线方法能够获得全局最优分配解,从而保证用户尽快得到高质量的服务。边缘计算环境并非一成不变,需要获得服务的用户在各个时间段一般是不同的,并非完全已知;在用户、服务器数量庞大时,需要根据这些变化及时得到合适的分配策略,离线方法不方便处理。

因此,对于在线方法,最重要的问题就是在可接受的时间内得到可接受的分配解,以自适应地调整对用户-服务器的分配。

本文的在线分配方法分为两个主要阶段:1)只要扩展的新请求者不会违背约束,就更新当前解;2)当出现一些过期用户或新请求时,重新对用户请求的计算任务进行分配。我们首先使用二分图匹配中的经典算法——匈牙利算法来得到一些候选解,之后分别使用基于基础启发式(Basic Heuristic, BH)和近邻启发式(Proximal Heuristic, PH)的方法得到在线算法的最终分配解。

3.2.1 基础启发式策略(BH)

基础启发式方法会比较每个候选分配解雇用的边缘服务器数量,并将使用服务器数量最少的解作为次优的分配解。它在尽量为更多用户提供服务的情况下,寻找能够使雇用服务器数量更少的策略。

3.2.2 近邻启发式策略(PH)

在 EUA 中,距离服务器近的用户和边缘服务器之间能够保证快速、稳定的数据传输和信息通信,因此我们更希望通过启发式策略进一步保证用户能得到更近的服务器的计算资源,以保证提供服务的质量,提升用户体验。

基于 BH 和 PH 这两个启发式策略,本文提出的在线算法如算法 1 所示,其复杂度为 O(n)。

算法 1 在线 EUA 算法

输入:边缘服务器集合 S,当前时间 t_k 的可用用户集合 U_k,当前分配策略 A_{ij}

输出:新的分配策略 A_{ij}

```
1. FOR s_i 覆盖范围内的每个新出现用户 u_j
2.   IF u_j 的出现并不违反约束 THEN
3.     更新分配策略 A_ij
4. ENDFOR
5. IF 逾期用户的比例大于阈值或仍存在新的请求者 THEN
6.   移除逾期用户 U'_{k-1} ← U_{k-1};
7.   更新当前用户集合 U_k ← U_k ∪ U'_{k-1};
8.     移除没有被任何服务器覆盖到的用户 U_k' ← U_k;
9.       构建当前边缘服务器集合 S_k = {s_i ∈ S | ∀ u_j ∈ A_ij},使雇用的
         服务器数量尽可能少;
10.    对于得到的 U_k 和 S_k,调用匈牙利算法来求得候选解;
11.    使用贪心启发式的方法 BH 或者 PH 来求得解 A_ij;
12. RETURN A_ij
```

4 实验设置

4.1 实验数据

本文以 EUA 问题的公开数据集 EUA Dataset[1)] 中的用户和边缘服务器地理位置数据为基础,并以均匀分布生成的用户和服务器的资源及服务覆盖范围等仿真数据,构建半真实半仿真的数据集。用户的工作量和边缘服务器资源属性包括 CPU、RAM、VRAM 和带宽 4 类。用户和服务器的位置信息都来自于数据集 EUA Dataset。

(1)用户数据。实验中使用 512 条任务数据,CPU 的值在 2 到 9 间随机选取,RAM 大于或等于 CPU 值,VRAM 的值为 CPU～CPU * 2 间的一个值乘以 0.5,带宽平均为 CPU 的 4 倍。本文将 40 个用户划分为一组,每一组用户在同一时间发出服务请求,用户的停留时间在 1 到 10 之间随机生成。

(2)边缘服务器数据。覆盖半径在 450～750 m 之间随机生成;服务器的资源属性值根据服务资源与用户工作量的比值生成。

4.2 参数设置

(1)用户的数量

本文设置不同的用户数量 $n=4,8,16,\cdots,512$。

(2)边缘服务器的数量

设 n 个任务位于 m 个服务器的覆盖范围内,假设其中有 m' 个可用服务器。在实验中,分别设置 $m'=10\%,20\%,\cdots,100\% * m$,这 m' 个服务器目前可以为 n 个用户提供服务。

(3)资源-工作量比

服务器剩余的资源量会影响分配的质量,本文实验中设置不同的服务器资源-用户工作量比值。我们分别设置资源

1) https://github.com/swinedge/eua-dataset

量为工作量的100%，150%，…，300%，然后将其分布到m'个服务器。

表1 实验数据集的设置

Table 1 Settings of experimental data set

数据集	用户数量	可用服务器所占比例	资源-工作量比值
数据集1	4，8，…，512	100%	300%
数据集2	512	10%，20%，…，100%	300%
数据集3	512	100%	100%，150%，…，300%

根据参数设置，本文生成了3个子数据集来进行控制变量的对比实验。表1列出了数据集设置。对于每个子数据集，我们改变一个参数，并保持其他两个参数不变，以观察每个参数对分配效果的影响。

在数据集1中，用户数量从4变化到512(4，8，16，32，64，128，256，512)。在数据集2中，固定用户数量为512，资源-工作量比值为300%，改变可提供服务的服务器占总服务器的比例，即10%，20%，…，100%。在数据集3中，固定用户数量为512，并使所有服务器都可用，变化的因素是资源-工作量比值，分别为100%，150%，…，300%。

4.3 实验评价标准

(1)服务的用户数占总用户数的百分比，该值越大越好。

(2)使用的边缘服务器数占总服务器数量的百分比，该值越小越好。

(3)算法求解的时间(单位为s)，该值越小越好。

(4)竞争比，用于评价在线算法得到的分配解的性能，其计算方法如下：

$$R_X=\frac{\text{在线算法 } X \text{ 得到的目标函数值}}{\text{离线算法得到的最佳目标函数值}} \tag{10}$$

4.4 对比算法

4.4.1 离线算法

离线算法使用3约束求解器CPLEX，Gurobi和Choco，基于文献[5]中的方法进行分配策略的求解。约束求解器基于精确式算法实现规划求解，致力于求得问题的全局最优解。

4.4.2 在线算法

我们比较了3种在线算法，即匈牙利(Hungarian)算法、EUA-BH和EUA-PH算法。匈牙利算法是二分图匹配问题中的常用算法，其实现简单，是处理该类问题的一种基础算法。

为了仿真在线的动态环境，我们根据用户的到达时间和离开时间将用户按一定数量分为多个组。

4.5 实验结果

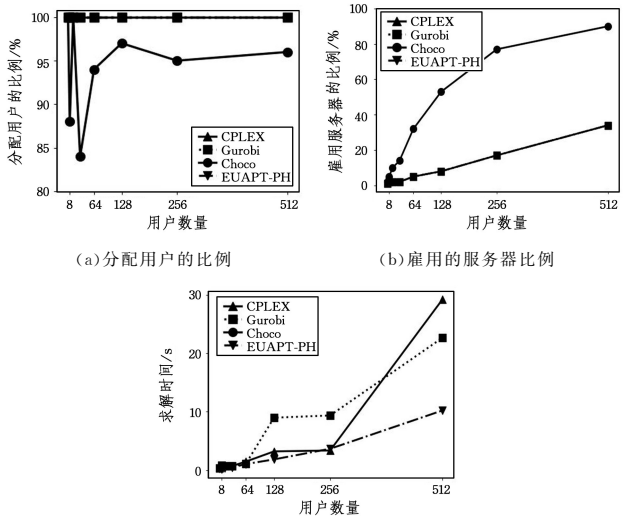
4.5.1 离线算法和在线算法的对比实验

在数据集1到数据集3的实验中，本文对比了不同约束求解器基于文献[5]中的离线方法和在线EUA-PH方法实现的求解效果。

实验中，离线和在线算法使用的用户、边缘服务器数据相同，不同的是在线算法中的用户是动态到达的。

(1)用户数量变化的影响

用户数据变化对分配效果的影响如图3所示。可以看出，随着用户数量增加，雇用服务器数量增多，因为更多的用户需要更多的服务器来提供服务。Choco服务器的使用率最高，CPLEX，Gurobi和EUA-PH都能百分之百地完成任何用户数量的计算任务分配，而Choco仅在用户数量少时可达到100%分配，其他情况下的用户分配比例大都在95%以下。Choco的求解时长为1.12～1 061.34 s，与其他方法不在同一数量级，因此在图中不再标注。当用户数量小于64时，CPLEX和Gurobi的求解时长近似，而用户数量增长到512时，仅Gurobi的求解时长较短。EUA-PH整体上能保证较短的求解时长，且其竞争比可接近100%。



(a)分配用户的比例 (b)雇用的服务器比例

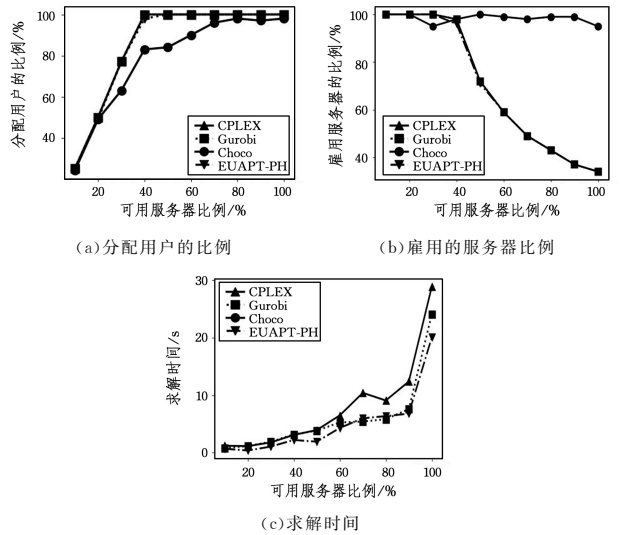
(c)求解时间

图3 用户数量变化对分配效果的影响

Fig. 3 Influence of change of user number on assignment effect

(2)可用边缘服务器数量变化的影响

从图4可以看出，随着可用服务器数量的增加，能得到资源的用户数量也随之增加。因为可选服务器数量变多，所以能选择更合适、可为多个用户提供资源的服务器，从而使服务器雇用率减小。求解时间也随服务器数量的增加而增长，其中Gurobi的求解时长最短，EUA-PH与其结果相似。Choco在所有方面都不尽人意，且其求解耗时过长，为17.74～1 159.52 s，故不再对其进行比较。在该实验中，在线方法EUA-PH的竞争比可以接近100%。



(a)分配用户的比例 (b)雇用的服务器比例

(c)求解时间

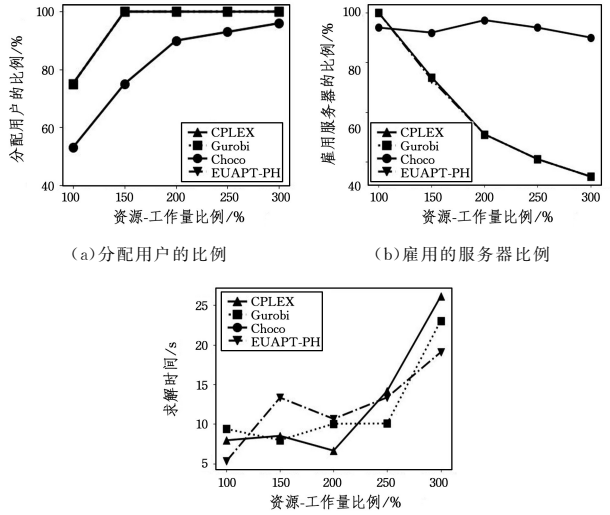
图4 可用服务器数量变化对分配效果的影响

Fig. 4 Influence of changes in number of servers available on assignment effect

(3)资源-工作量比值变化的影响

用户数量变化对分配效果的影响如图 5 所示。随着可用资源增多,现有服务器能为更多用户提供服务,同时需要的服务器减少。Choco 在控制服务器雇用率方面表现较差,当其他求解器将服务器雇用率降低到 40%以下时,其仍能达到 80%的雇用率。

与前两个数据集中的分配效果相同,在该实验中,Gurobi,CPLEX 和 EUA-PH 得到的分配效果相似。在线方法 EUA-PH 的竞争比可接近 100%。CPLEX 的求解时长比其他两种方法都长,Gurobi 和 EUA-PH 在求解时长上难分伯仲,但从趋势上可以看到,Gurobi 的时长变化折线斜率更大。而 Choco 的求解时长过长,我们同样不再对其进行比较。



(a)分配用户的比例 (b)雇用的服务器比例

(c)求解时间

图 5 用户数量变化对分配效果的影响

Fig. 5 Influence of resource-workload ratio change on assignment effect

(4)离线算法和在线算法的求解时长对比

在 3 个求解器中,Gurobi 在控制求解时长方面表现最佳,因此我们将其与在线方法进行对比,来观察离线和在线算法在求解时长方面的表现。在线算法和离线算法在求解时长上的对比如图 6 所示。

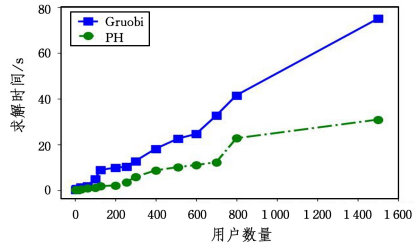


图 6 在线算法和离线算法在求解时长上的对比

Fig. 6 Comparison of solving time of online and offline algorithms

可以看出,在用户数量增长的过程中,Gurobi 的求解时长的增长速度较快,折线斜率迅速增大,这说明文献[5]中的分配方法会耗费较多的时间来求解,影响了 EUA 的质量,而在线方法的求解时长的增长较为平缓。在现实计算任务卸载环境中,用户的出现是动态的,且数据量比实验中的大,因此离线方法具有很大的局限性。同时,将约束求解器投入商用来处理更大规模的数据也需要支付更多的费用,一定程度上会增加分配实现的成本。

4.5.2 在线算法的对比

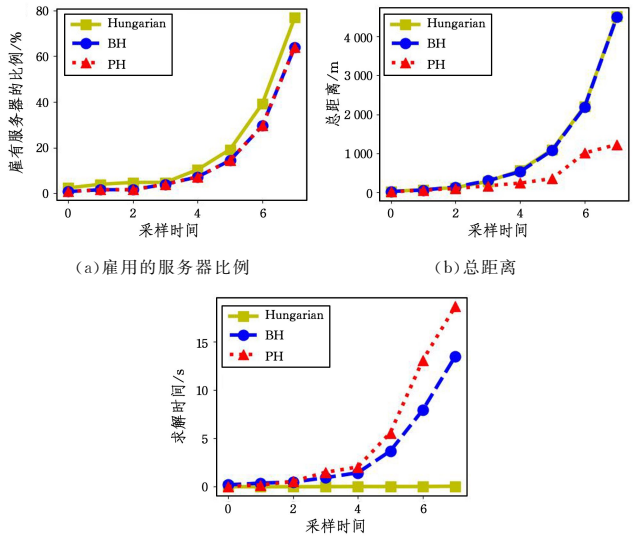
本节对比了 3 种在线算法——匈牙利算法、BH 算法、PH 算法的分配效果。为了对比在线算法在控制距离方面的效果,对分配解中用户与服务器的总距离进行了对比。在下文的结果图中,横坐标表示时间片(将时间划分为多个批次,每个批次为一个时间片)。

(1)用户数量变化的影响

因 3 种算法都能达到近乎 100%的用户分配率,故不再对相关结果进行对比。

从图 7(a)中可看出,在各时间片中,随着用户数量增多,3 种算法都使用了更多的服务器来提供资源,而匈牙利算法使用的服务器较多,无法对式(2)中的目标进行优化,其他两种算法基本持平。

同时,由于 PH 算法是基于最短距离的启发式策略,因此其在控制总距离方面表现最佳。BH 算法的总距离与匈牙利算法类似。在求解时间方面,匈牙利算法实现简单,求解耗时较短,而 BH 方法不需要对距离进行计算,因此求解时间比 PH 方法的短。



(a)雇用的服务器比例 (b)总距离

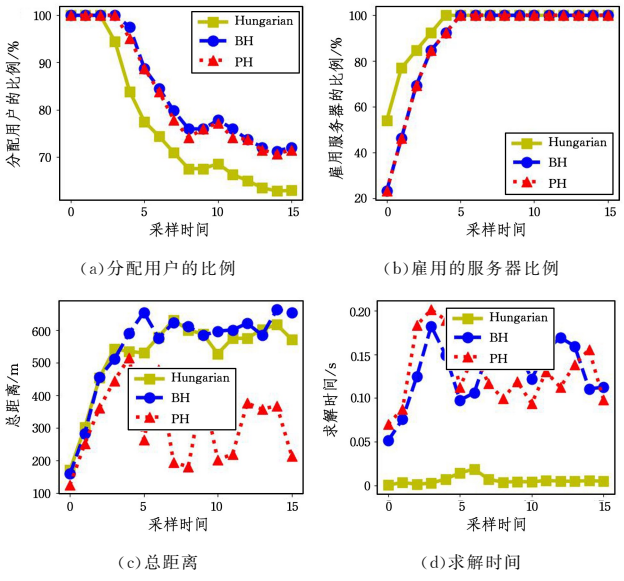
(c)求解时间

图 7 在线算法在数据集 1 上的对比结果

Fig. 7 Comparison of online algorithms on dataset 1

(2)可用边缘服务器数量的影响

可用服务器较多时,3 种算法的表现相似,故本实验讨论在可用服务资源匮乏时的分配效果。从图 8(a)可以看出,随着时间片的变化,出现的用户增多,可用的服务器变少,使得用户分配比例减小,但 BH 和 PH 算法能达到比匈牙利算法更高的分配率。在图 8(b)中,匈牙利算法的服务器使用率在初始时间片中比其他算法高,而因服务器数量较少,最终几种方法的服务器使用率都达到了 100%。同时,PH 算法在服务器较少时仍能控制用户与服务器之间的距离。匈牙利算法的求解时间较短,PH 算法耗费了较长时间来保证距离较短的分配解。BH 因对距离不加以控制,所以用时较短。



(a)分配用户的比例 (b)雇用的服务器比例

(c)总距离 (d)求解时间

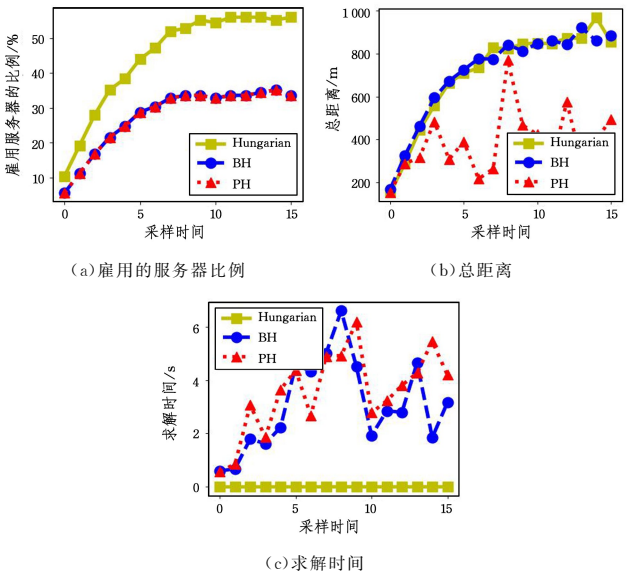
图 8 在线算法在数据集 2 上的对比结果(10%可用服务器)

Fig. 8 Contrasts in the online algorithm in the data set 2 results (10% available servers)

(3)服务器资源-用户工作量比例的影响

在资源丰富时各种算法均能达到 100%的用户分配率，故图 9 只给出了服务器资源较少时极限状态下的分配效果。

在图 9(a)中，BH 和 PH 算法能保证较低的服务器使用率。在图 9(b)中，PH 算法也仅偶尔能保证距离的控制，因为 100%的资源-工作量比例难以实现对距离的要求。同时，PH 算法的求解时间也比其他方法更长。



(a)雇用的服务器比例 (b)总距离

(c)求解时间

图 9 在线算法在数据集 3 上的对比结果(资源-工作量比为 100%)

Fig. 9 Comparison results of online algorithm in data set 3 (Resource-workload ratio is 100%)

综上，匈牙利算法作为基础方法，求解时间较短，但对于目标并不能实现优化。BH 算法的求解时间较短，在优化目标方面表现较好，但其不能控制距离。PH 算法虽求解时长较长，但能优化分配目标和控制距离，避免用户和服务器之间的通信延迟，在一定程度上能够弥补其在求解时长上的劣势，且能解决 EUA 优化模型的离线方法[5]求解时间过长从而难以适用于现实应用的问题，其不失为一种可行性高的 EUA 问题的优化方法。

5 相关工作

移动边缘计算技术被广泛应用于实时性强和带宽密集型业务中，在云计算[6]、区块链[7]等领域都有所应用。在移动边缘计算中，边缘用户分配是云计算和边缘计算中的一个重要环节，它能够将本地无资源执行的计算任务迁移到云端执行，以保证提供高质量的服务[8]。合理的“用户-服务器”分配策略能够提供高质量的服务，不合理的分配策略反而会耗费更多的成本，且会造成延迟，影响服务质量。分配策略的设计聚焦于以下几个目标：降低时间延迟、减小能耗和保证用户 QoE。

Rodrigues 等[9]同时关注了计算和通信两个因素来减小服务延迟。他们通过虚拟机迁移控制处理延迟，并通过传输功率控制来改善传输延迟。文献[10]提出了一种在线任务卸载算法，其使用负载均衡启发式方法将任务卸载到服务器上，最大程度地缩短了移动终端上应用程序的完成时间。

Chen 等[11]将计算任务的分配问题建模为一个联合优化问题，其目标为降低总能量消耗，受时延、传输质量、成本预算和传输功率等因素的约束。You 等[12]将计算资源分配问题规约为一个凸优化问题，以在时间延迟约束下对能量消耗进行最优化。

此外，有很多研究者兼顾时间延迟和能耗对分配策略进行优化，这也是目前的热点研究方向。Kao 等[13]在分配时考虑了计算资源的可用性和链接连接性，并在移动设备的能耗成本和延迟之间找到了平衡。Zhang 等[14]研究了移动边缘计算在多用户和多服务器中的分配策略，它能在保证低延迟的情况下优化策略。文献[15]提出了一种能量敏感的计算任务卸载方法，该方法在有限的能量和等待时间下优化了通信和计算资源的分配。Yang 等[16]提出一种基于用户体验的计算卸载方案，用户希望计算任务完成时满足用户对时延和能耗的要求。Liu[17]提出了一种能够最小化系统总功率开销的双层任务优化策略，在节省任务执行时间的同时减少消耗。

Xiao 等[18]以用户的 QoE 为前提，考虑了计算网络中的服务器节点之间的协作性，减轻了云计算中心的工作量。文献[19]研究了基于 QoE 的边缘计算的计算任务调度问题，并同时考虑了通信和计算资源来处理该问题。在在线调度方面，文献[20]考虑了用户变化对用户分配的影响，但并未将服务器的负载能力考虑在内。

结束语 在边缘计算中，为用户选择合适的边缘服务器来执行计算任务是一个关键问题。用户-边缘服务器匹配要考虑距离和容量两个约束，以保证用户得到就近服务器的服务支持，并且服务器可提供足够的计算资源。本文将 EUA 问题建模为一个二分图匹配问题，并分别使用离线和在线方法对分配策略优化求解。在离线状态下，因对 EUA 模型的优化是基于精确式算法的，所以可以得到最佳的分配结果。但随着数据量的增加，其求解时间过长且增长迅速，并不适用

于现实应用环境。而在线算法的求解时长更短，分配的解虽并非最优但却能达到可接受的效果，更符合现实应用的需要。

本文使用真实数据和仿真数据进行了实验验证，结果表明提出的方法具有可行性和有效性。

参 考 文 献

[1] SHI S, SUN H, CAO J, et al. Edge Computing—An Emerging Computing Model for the Internet of Everything Era [J]. Journal of Computer Research and Development, 2017, 54(5): 907-924.

[2] PATEL M, NAUGHTON B, CHAN C, et al. Mobile-edge computing introductory technical white paper [M]. Mobile-edge Computing (MEC) Industry Initiative. 2014: 1089-7801.

[3] ZHANG W L, GUO B, SHEN Y, et al. Computation offloading on intelligent mobile terminal [J]. Chinese Journal of Computers, 2015, 38(30): 1021-1038.

[4] HE Q, CUI G, ZHANG X, et al. A game-theoretical approach for user allocation in edge computing environment [J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3): 515-529.

[5] LAI P, HE Q, ABDELRAZEK M, et al. Optimal edge user allocation in edge computing with variable sized vector bin packing [C]// International Conference on Service-oriented Computing. 2018: 230-245.

[6] WU J, LIU T, LI J, et al. Research Progress on Blockchain Technology in Mobile Edge Computing[J]. Computer Engineering, 2020, 46(8): 1-13.

[7] WANG Y, GE H, FENG A. Computation Offloading Strategy in Cloud-Assisted Mobile Edge Computing [J]. Computer Engineering, 2020, 46(8): 27-34.

[8] MACH P, BECVAR Z. Mobile edge computing: A survey on architecture and computation offloading [J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[9] RODRIGUES T G, SUTO K, NISHIYAMA H, et al. Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control [J]. IEEE Transactions on Computers, 2016, 66(5): 810-819.

[10] JIA M, CAO J, YANG L. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing [C]// IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2014: 352-357.

[11] CHEN X, CAI Y, LI L, et al. Energy-efficient resource allocation for latency-sensitive mobile edge computing [J]. IEEE Transactions on Vehicular Technology, 2019, 69(2): 2246-2262.

[12] YOU C, HUANG K, CHAE H, et al. Energy-efficient resource allocation for mobile-edge computation offloading [J]. IEEE Transactions on Wireless Communications, 2016, 16(3): 1397-1411.

[13] KAO Y H, KRISHNAMACHARI B, RA M R, et al. Hermes: Latency optimal task assignment for resource-constrained mobile computing [J]. IEEE Transactions on Mobile Computing, 2017, 16(11): 3056-3069.

[14] ZHANG H, GUO J, YANG L, et al. Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC[C]// IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2017: 115-120.

[15] ZHANG J, HU X, NING Z, et al. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks [J]. IEEE Internet of Things Journal, 2017, 5(4): 2633-2645.

[16] YANG T, TIAN L, SUN Q, et al. Computing Offloading Scheme Based on User Experiencein Mobile Edge Computing[J]. Computer Engineering, 2020, 46(10): 33-40.

[17] LIU T. Task Offloading Strategy for Minimizing Power Consumption in Two Layer Edge Computing Architecture[J]. Journal of Chongqing University of Technology (Natural Science), 2019, 33(8): 157-164.

[18] XIAO Y, KRUNZ M. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation [C]// INFOCOM. 2017: 1-9.

[19] LUO J, DENG X, ZHANG H, et al. QoE-driven computation offloading for edge computing [J]. Journal of Systems Architecture, 2019, 97: 34-39.

[20] PENG Q, XIA Y. Mobility-Aware and Migration-Enabled Online Edge User Allocation in Mobile Edge Computing [C]// IEEE International Conference on Web Services. 2019: 91-98.

TANG Wen-jun, born in 1994, Ph. D student. Her main research interests include crowdsourcing workflows, crowd sourcing task assignment and web service testing.

CHEN Rong, born in 1969, Ph. D, professor, is a member of the IEEE and a member of the ACM. His main research interests include software diagnosis, collective intelligence, activity recognition, Internet and mobile computing.