针对移动云计算任务迁移的快速高效调度算法

史雯隽[1] 武继刚[2] 罗裕春[2]

(天津工业大学计算机科学与软件学院 天津 300387)[1] (广东工业大学计算机学院 广州 510006)[2]

摘 要 计算量较大的应用程序由于需要大量的能耗,因此在电池容量有限的移动设备上运行时十分受限。云计算迁移技术是保证此类应用程序在资源有限的设备上运行的主流方法。针对无线网络中应用程序任务图的调度和迁移问题,提出了一种快速高效的启发式算法。该算法将能够迁移到云端的任务都安排在云端完成这种策略作为初始解,通过逐次计算可迁移任务在移动端运行的能耗节省量,依次将节省量最大的任务迁移到移动端,并依据任务间的通讯时间及时更新各个任务的能耗节省量。为了寻找全局最优解,构造了适用于此问题的禁忌搜索算法,给出了相应的编码方法、禁忌表、邻域解以及算法终止准则。构造的禁忌搜索算法以提出的启发式解为初始解进行全局搜索,并实现对启发解的进一步优化。通过实验将所提方法与无迁移、随机迁移、饱和迁移3类算法进行对比,结果表明提出的启发式算法能够快速有效地给出能耗更小的解。例如,在宽度为10的任务图上,当深度为8时,无迁移、随机迁移与饱和迁移的能耗分别为5461、3357和2271能量单位,而给出的启发解对应的能耗仅为2111。在此基础上禁忌搜索算法又将其能耗降低到1942,这进一步说明了提出的启发式算法能够产生高质量的近似解。

关键词 移动云计算,任务迁移,调度,启发式算法

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.014

Fast and Efficient Scheduling Algorithms for Mobile Cloud Offloading

SHI Wen-jun[1] WU Ji-gang[2] LUO Yu-chun[2]

(School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)[1]

(School of Computer, Guangdong University of Technology, Guangzhou 510006, China)[2]

Abstract Running applications of high computation on mobile devices is constrained by limited battery capacity and energy consumption of the devices. Cloud offloading is a main solution for supporting computationally demanding applications on these resource-constrainted devices. This paper proposed a fast and efficient heuristic approach for scheduling and offloading problems of the application task graph in the wireless network. The proposed heuristic approach initially moves the tasks that can be offloaded to the cloud, and then iteratively moves the tasks with highest benefit value to the mobile device. The benefit values are updated in each iteration to cater for the task concurrence. In addition, this paper also constructed a tabu search approach to search for the global optimization solution. It presented and implemented the encoding method, tabu list, neighborhood solutions and the stopping criterion of the proposed tabu search algorithm. The customized tabu search algorithm is with the initial solution generated by the proposed heuristic algorithm. By comparing three algorithms based on non-offloading, full offloading, and random offloading, experimental results show that the proposed heuristic algorithm runs very fast, and the generated heuristic solutions are efficient. For the case of the task graphs with width of 10 and depth of 8, the energy consumption of non-offloading, full offloading, and random offloading are 5461, 3357 and 2271 respectively, while the proposed heuristic solution is 2111. It is further reduced to 1942 by the customized tabu search. The results confirm that the proposed heuristic algorithm can generate high quality approximate solution for the scheduling and offloading problem in mobile computing.

Keywords Mobile cloud computing, Task offloading, Scheduling, Heuristic algorithm

1 引言

随着用户对网络数据和服务质量(QoS)的需求呈指数增长,智能手机、笔记本电脑和平板电脑技术的发展也对各种网络服务和应用产生了更高需求。虽然移动设备的性能因自身中央处理器(CPU)的发展而越来越高,但是仍不能满足应用

到稿日期:2017-05-19 返修日期:2017-06-18 本文受国家自然科学基金资助项目(61672171),广东省教育厅重大科研项目(2016KZDXM052),广东省应用型科技研发专项(重点)(2015B010129014)资助。

史雯隽(1988-),女,博士生,主要研究方向为云计算、数据中心网络;**武继刚**(1963-),男,博士,教授,CCF会员,主要研究方向为高性能计算,E-mail:asjgwucn@outlook.com(通信作者);**罗裕春**(1990-),男,硕士生,主要研究方向为云计算。

程序在短时间内处理大量数据的计算需求。此外，电池持续的高能耗始终是限制用户在自己的设备上充分使用高数据需求应用的障碍。移动用户计算与电量的高需求和低存储量之间的矛盾，促使了移动云计算的发展，使得云计算资源可以为移动用户提供服务[1]。在移动云计算中，用户的设备基于移动运营商和网络可以使用距离较远的云平台的丰富的计算和存储资源。移动云计算的优势[2]是：1）通过将计算密集型的高能耗任务迁移到云端来延长电池的使用时间；2）移动用户可以使用复杂的应用程序；3）为用户提供高效的数据存储功能。

将网络应用的数据流和计算密集型应用的任务迁移到云端或者小型的云服务器上执行已经成为了一种公认的解决移动设备资源不足问题的方案[3-4]，比如视频游戏、基于计算机视觉的应用程序等[5]。本文中，迁移后的任务将被放在远程的云端完成。现有的计算迁移方法分为 3 类：1）整个应用都在云端执行[6-7]；2）根据省电情况来决定将整个应用迁移到云端执行还是迁移到移动端执行[8]；3）将部分可迁移的任务迁移到云端执行，其余的在移动端完成[9-13]。在实施部分迁移的过程中，可将应用程序粗略地划分为各个组件[9-10,14]，或者用 MAUI[11]，ThinkAir[12] 和 CloneCloud[15] 划分方法将应用划分为更细粒度的迁移。

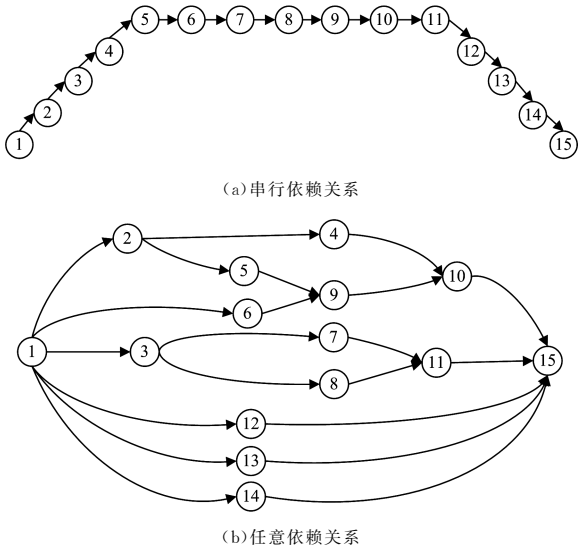
在迁移策略的实施中完成任务时所消耗的电量和时间，是对服务质量产生决定性影响的两个因素，尤其是在无线网络中更需要考虑这两方面[16]。因此在一定的时间约束下，使得移动端的能耗最少是一个挑战性的课题。计算迁移方案时将耗费较多的计算和存储资源，由于它不在移动端处理能力的范围内，因此不能将其放在移动端计算。如果将其放在云端，则要消耗过多的传输延时和能耗。因此将这类复杂问题放在移动端所在区域的网络接入基站中的小型处理器上进行处理是主流做法[1]，这样既能保证足够的处理能力也能保证快速通讯。文献[17]针对无线网络中多个计算任务的迁移策略的任务调度问题，利用 IBM CPLEX 优化器求解最优调度方案。但是文献[17]提供的优化器解决方案在内存为 60GB 的服务器上需要运行两个小时，因此在确保服务质量的实际应用中并不适用。针对以上不足，本文给出了计算模型和公式化描述，并设计了两种快速启发式算法。

本文第 2 节给出了问题的形式化描述；第 3 节展示了启发式算法和禁忌搜索算法；第 4 节为实验结果和分析；最后总结全文。

2　相关工作

2.1　任务迁移模型

在应用的任务图中任务之间的依赖关系可以是串行的，或者可以通过处理转化成串行的，如图 1(a)所示。更普遍任务间的依赖关系是任意的，如图 1(b)所示。在任务图中每一个节点代表一个任务，任务 i，j 之间的连线表示 i，j 之间的依赖关系。任务 i 输出的数据是任务 j 输入的数据。任务图必须满足如下 3 个特性：1）每个节点必须有一个入度和一个出度（除了第一个起始节点只有一个出度，最后一个节点只有一个入度外）；2）每一个节点和第一个节点（最后一个节点）之间存在一条直接或间接的路径；3）任务图的邻接矩阵对角线上的元素是 0，即不能存在自身到自身的节点。为了简化问题，假定任务图中不能存在环，即不能出现从某个点经过若干步骤再到自身的通路。如果原始任务图出现了环，则可以通过一些图规约等预处理方式将环代替。特别地，在这些任务中有部分任务仅能在移动端完成，其中第一个任务和最后一个任务必须在移动端完成。



(a)串行依赖关系

(b)任意依赖关系

图 1　有依赖关系的任务图

Fig. 1　Tast graph with dependency relationship

由于云服务器具有强大的计算能力和存储能力，迁移到云服务器上的任务在云端可以并行完成。沿用文献[17]中的两个假设：1）移动用户所使用的多组件应用也同样安装在云服务器上；2）尽管通过无线接口获得的网络传输速率和延迟会随时间变化，但是在移动应用进程中认为移动连接带宽是恒定的[11-12,14,19-21]。对于相互之间有通讯的两个任务，若它们同在移动端或云端完成，那么通讯时间和功耗可以忽略不计；若它们不在同一端完成，那么则需要通过移动端上传或下载，此时则需要消耗移动端的能量，并且需要耗费一定的时间。

本文要解决的移动云计算任务迁移的调度问题是指在一定的时间约束下，决定迁移哪些任务到云端完成，并确定任务执行的先后顺序，使得移动设备消耗的能量最小。然而移动端在单位时间内由于计算能力的限制，仅能执行一个任务，那么同时进行多个任务的调度问题就变得尤为重要。下一节将给出这个调度问题的数学模型。

2.2　数学模型

本节将利用文献[17]的整数线性规划模型对上述移动任务迁移问题进行描述。本节未在正文中解释的参数及其含义如表 1 所列。

表 1　参数定义

Table 1　Definition of parameters

参数	定义
N	总的任务数
T	总的时间限制
c_j	任务 j 在云端完成
m_j	任务 j 在移动端完成
$\tau_j^m(\tau_j^c)$	任务 j 在移动端（云端）完成需要的时间
P	单位时间内移动端执行一个任务的能耗
w_k	k 在移动端或者云端完成的时间
$P_{Tx}(P_{Rx})$	移动端传递（接收）单位数据的能耗

在单位时间$(t-1,t]$(用t表示)内,定义变量x_{jt}^{l}表示任务j在t时刻在移动端完成($l=1$)或在云端完成($l=0$)。因此用变量m_j来表示任意一个任务j是在移动端完成($m_j=1$)还是在云端($c_j=1$)完成,计算式为:

$$m_j=\sum_{t=1}^{T}x_{jt}^{0} \tag{1}$$

$$c_j=\sum_{t=1}^{T}x_{jt}^{1} \tag{2}$$

其中,T是完成整个任务图的时间约束。当任务i是任务j的前驱节点并且i在云端完成,j在移动端完成时,用τ_{ij}^{cm}表示任务i向任务j传递数据所需要的时间,它也同时包含了乘积m_jc_i。为了使这个优化问题线性化,将m_jc_i替换成二次项z_{ji}表示。$z_{ji}=1$表示j在移动端完成,i在云端完成。在其他情况下,$z_{ji}=0$。这个变量还需符合4个约束[22]:$z_{ji}\leqslant m_j$,$z_{ji}\geqslant 0$,$z_{ji}\leqslant c_i$,$z_{ji}\geqslant c_i-(1-m_j)$,$\forall i,j$。那么,这两个决策变量相乘的二次项就转化成了一个新决策变量,这个优化问题依然是线性的。同样,λ_{ij}^{mc}表示当i在移动端完成且j在云端完成时任务i向任务j传递数据所需要的时间,它所包含的m_ic_j用z_{ij}来表示。对于$\forall i,j$,移动端向云端及云端向移动端进行传输的时间的计算式如下:

$$\lambda_{ij}^{cm}=\alpha_{ij}z_{ij}\frac{d_{ij}}{R_d} \tag{3}$$

$$\lambda_{ij}^{mc}=\alpha_{ij}z_{ij}\frac{d_{ij}}{R_u} \tag{4}$$

其中,$\alpha_{ij}=1$表示i必须在j之前执行,反之为0。d_{ij}是j从i节点获得的数据量,R_d和R_u分别是无线网络接口下载和上传的平均速率。值得注意的是,当$i=j$,或者i不是j的前驱,或者i和j同时在云端或移动端完成时,τ_{ij}^{cm}和τ_{ij}^{mc}都等于0。因此,任务迁移所消耗的能量为:

$$E_{com}=P_T\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{ij}^{mc}+P_R\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{ij}^{cm} \tag{5}$$

建立在决策变量x_{jt}^{l},z_{ij}($l\in\{0,1\}$,$i,j=1,\cdots,N$,$t=1,\cdots,T$)上的最小化移动端能量消耗的目标函数可定义为:

$$\min E=\{\sum_{j=1}^{N}Pm_j\tau_j^m+E_{com}\} \tag{6}$$

同时,需要满足以下5个约束条件。

1)总时间限制:完成整个任务图的所有任务的总时间在T内,即最后一个任务N的完成时刻小于或等于T:

$$0<\sum_{t=1}^{T}t\cdot x_{Nt}^{0}\leqslant T \tag{7}$$

2)任务的执行顺序限制:若k要在j之前完成,令$\upsilon(w,\lambda)=t+w_k+\lambda_{jk}^{cm}+\lambda_{jk}^{mc}$,那么

$$\sum_{l=0}^{1}\sum_{s=1}^{\upsilon(w,\lambda)}x_{ks}^{l}\leqslant\sum_{l=0}^{1}\sum_{s=1}^{t}x_{js}^{l},\text{if } t=w_j,\cdots,T-w_k-\lambda_{jk}^{cm}-\lambda_{jk}^{mc} \tag{8}$$

3)每个任务无论在云端还是移动端最多且仅能执行一次,即:$m_j+c_j=1$,$\forall j$。

4)在移动端任务间是串行的:在移动端多个任务串行完成,即在每个单位时间内只有一个任务在运行,也即对于任意t:

$$\sum_{j=1}^{N}\sum_{s=t}^{\min\{t+w_j-1,T\}}x_{js}^{0}\leqslant 1 \tag{9}$$

5)每一个任务的完成时间限制:对于任意一个任务k,在k执行前都要等它前面的任务执行完成,k的完成时间就是它前面任务的完成时间加上它自身任务的完成时间,如果j是k之前的任务,则:

$$\sum_{l=0}^{1}\sum_{t=1}^{T}t\cdot x_{jt}^{l}+\lambda_{jk}^{cm}+\lambda_{jk}^{mc}+w_k\leqslant\sum_{l=0}^{1}\sum_{t=1}^{T}t\cdot x_{kt}^{l} \tag{10}$$

值得注意的是,所有的决策变量都是0-1变量$x_{jt}^{l}\in\{0,1\}$,$l\in\{0,1\}$,$\forall j,t$。

3 调度算法

3.1 先来先服务过程

本文采用先来先服务过程(Procedure FCFS)来计算整个任务图的完成时间。由于在云端完成的任务可以并行,划分到云端执行的任务只有当其前驱节点完成这个任务后才能够执行。

由于移动端任务间是串行的,因此对于移动端的多个可执行任务,需要确定先后执行顺序,以保证所有任务在时间限制内完成。先来先服务过程以移动端任务能够执行的时间先后顺序建立一个先进先出的任务队列,以此来确定任务执行的先后顺序。步骤如下:

1)建立一个空的先进先出队列(激活队列),用来存储具有以下条件的任务:

①该任务被划分到移动端执行。

②该任务的所有前驱节点任务已经执行完成,称为激活状态。

2)将所有激活状态的任务建立为一个激活队列,那么任务图中的任意一个节点均存在4种状态:激活、就绪、可运行、已完成。

①对已迁移到云端的任务,如果已处于激活状态,那么该任务可执行,也称其为可运行状态。

②在移动端执行的任务如果处于激活状态,且该任务已在激活队列的队头,则该任务可执行,也称该任务是可运行状态。

③在移动端执行的任务如果处于激活状态,但是未处在激活队列的队头,则还不能直接执行该任务,称该任务处于就绪状态。

④若一个任务的父节点没有全部完成,则该任务不能执行,称该任务处在非激活状态。

⑤若一个任务已被完成,则称它处于已完成状态。

3)初始化移动端执行时间为0,在云端的总消耗时间为0。

4)从任务图的第一个节点开始,按广度优先搜索遍历,对每一个节点,按照是否在激活状态或可运行状态执行该任务,即:

①若遍历到的节点是云端的可运行任务,则执行该任务。

②若遍历到的节点是移动端的可运行任务,则执行该任务。

③若遍历到的节点是移动端的就绪任务,则将其插入激活队列的队尾。

若任务图中一个任务完成,则执行步骤4),直到所有任务完成。先来先服务过程的伪代码描述如算法1所示。

算法1 Procedure FCFS

输入:任务图,任务图中每个任务在移动端的功耗、通信数据量、在云端的执行时间

输出:任务图总的时间

begin

1. 初始化激活队列为空,即 active_ queue←Ø;

2. while 任务图中仍有任务未执行 do

2.1. 对所有任务进行广度优先排序;

2.2. for i=1to N do

```
    if 节点 i 为可运行状态 then
      update active_ queue;
/ * 一个任务执行完毕，将该任务标记为已完成状态并记录执行完该
 任务的时间 * /
    else if i 为就绪状态 then
      插入 active_ queue 的队尾;
    else if i 为非激活状态或者已经执行完的状态 then
    跳过
    end if
    end if
    end if
  end for
end while
3. return 最后一个被执行的任务的完成时间;
end
```

由于 Procedure FCFS 在节点数为 N、边数为 E 时采用的是广度优先遍历，因此其复杂度为 $O(N+E)$。

3.2　启发式算法

对于每个任务，无论该任务是从云端划分到移动端，还是从移动端划分到云端，都会造成与之相连接的节点间的通信能耗和通信时间的变化。当一个任务从移动端划分到云端时，对移动设备来说，节省了执行该任务的能耗；而当一个任务从云端划分到移动端时，移动端执行该任务的能量比原来消耗得更多。启发式算法的思想是：每次从可迁移的任务中挑选节省能量最大的任务进行迁移，用 Procedure FCFS 来计算总的完成时间，直到时间约束不能满足或任务迁移不能够再降低能耗。启发式算法的步骤如下：

1）初始化：将所有可以迁移的任务都先假定在云端，而将仅能在移动端执行的任务放到移动端。计算当前的总能量消耗，并将所有可以迁移的任务标记为“未划分”。

2）每一次任务迁移的过程如下：在符合时间约束条件下按广度优先搜索，从标记为“未划分”的任务中依次挑选节省能耗最大的节点，将其迁移到移动端完成。若挑选出的节点使得迁移后的总能耗比迁移之前小，则迁移该节点，并将其标记为“已划分”，更新移动端总能耗；反之，整个算法结束。

3）进行步骤 2）中的迁移过程，直到算法停止。

算法的伪代码描述如算法 2 所示。

算法 2　Heuristic Algorithm(HA)

```
输入：任务图，任务图中任务在移动端的功耗，通信需要的数据量，在
      云端执行需要的时间，整个任务执行完的时间限制 time_con-
      straint
输出：移动端消耗的总能量
begin
1. 所有可迁移的任务都在云端执行，计算当前能耗 current_cost;
2. 标记可迁移的任务为“可卸载且未迁移”;
3. while 标记为“可迁移且未迁移”的任务数≠0 do
  3.1. for i=1to N do
    if 遍历到标记为“可迁移且未迁移”的节点
    用 Procedure FCFS 计算该节点从云端迁移到移动端的 temp_
    time 和 temp_cost
    / * temp_time 是总任务执行时间，temp_cost 是移动端总消耗能
    量 * /
      if temp_time ≤ time_constraint
      then
      存储此节点
        end if
      end if
      end for
    从存储节点中选取最小的 temp_cost
  3.2. if temp_cost≤current_cost then
      将该节点迁移到移动端，并将该节点标记为“可迁移且已迁
      移”，同时“可卸载且未迁移”的节点数量减 1;
        current_cost:=temp_cost;
        else
        停止 while 循环
        end if
      end while
4. return current_cost
end
```

启发式算法的 while 步骤最多循环 N 次。在每次循环中，都要进行广度优先遍历，且每遍历一个节点都要调用 Procedure FCFS 计算总任务图的执行时间。因此，启发式算的最坏时间复杂度为 $O((N+E)*(N+E)*N)$。

3.3　禁忌搜索算法

禁忌搜索算法是一种解决难解问题的有效的元启发式算法，本文用禁忌搜索算法来优化算法 HA 的解。禁忌搜索算法的各个要素的定义如下。

1）编码方法

任务数为 N，构造一个 N 维向量表示任务的完成方式。向量的每一个分量代表相应任务的完成方式。若该分量的值为 1，表示该分量对应的任务在移动端执行；若该分量的值为 0，表示该分量对应的任务在云端执行，如：[1，0，0，1，1，0，0，…，0，0，1]。

2）适值函数

移动端消耗的总能量为适值函数值。

3）初始解

以启发式算法 HA 的解作为初始解。

4）移动与邻域移动

随机地选取当前解的一个分量进行变化，即 0 变成 1，或者 1 变成 0。

5）禁忌表

禁忌表保存已经搜索过的解，长度设为 $10\% * N$。

6）选择策略

①邻域产生方法：随机选择当前解的一个分量，若该分量对应的任务仅能在移动端完成，则再随机选择一个分量，直到对应任务能够进行迁移为止，改变该分量的值。计算改变后的任务图的总时间，若不超过总时间限制，则计算当前任务图的适值，并保存；否则，重新按照上述方法产生邻域。重复上述过程，直到产生足够数目的邻域解为止。

②寻找邻域最优解：若邻域最优解在禁忌表中，且该解劣于历史最优解，则选择次优邻域解，直到非禁忌表中的解作为当前的最优解。若邻域最优解优于历史最优解，则该解将不受禁忌的约束，作为历史最优解。

③停止准则：最大迭代步数为停止准则。

禁忌搜索算法的伪代码描述如算法 3 所示。

算法 3 Tabu Search Algorithm(TS)

输入:任务图 G,整个任务执行完的时间限制 time_constraint

输出:移动端消耗的总能量

```
begin
1. 以启发解作为禁忌搜索的初始解,相应的能耗 HA_cost 作为初始适值;
   当前适值 current_cost:=HA_cost;
   全局适值 global_cost:=HA_cost;
2. for i =0 to iteration_number then
   /* iteration_number 是迭代次数 */
   2.1. for j=0 to neighbor_size then
        /* neighbor_size 表示领域大小 */,
      2.1.1. 用 Procedure 1 FCFS 计算邻域解对应的任务完成总时间
             temp_time;
      2.1.2. if temp_time ≤time_constraint then
        计算该解对应的移动端消耗的能量 temp_cost;
      end if
   end for
   2.2. 选出最小适值 min_cost;
   2.3. current_cost:=min_cost;
   2.4. if min_cost<=global_cost then
          global_cost:=min_cost;
        end if
   2.5. 更新当前解向量和禁忌表
      end for
3. return global_cost
end
```

4 实验结果

为了检验算法 HA 和 TS 解决任务迁移问题的性能,本节通过仿真实验将算法 HA 和 TS 的性能与引言中提到的 3 种迁移方案进行比较。分别选取将整个任务图都放在移动端完成的无迁移(non-offloading)算法、可迁移的任务全部在云端完成的饱和迁移(full offloading)算法以及在能迁移的任务中随机挑选完成方式的随机迁移(random offloading)算法。选取移动端总能量消耗(total energy consumption)和算法的运行时间两个性能指标来展示不同算法的解的性能和运行速度。

4.1 实验设置

实验使用 Intel(R) Xeon(R) CPU E5-2670 v3 处理器、主频 2.30 GHz、50 GB 内存、64 位 Windows 7 旗舰版操作系统的计算机,采用 C++编程,所得数据均是 100 次相同设置实验的平均值。为了贴合实际应用的真实情况,参照文献[17],将实验数据设置如下:

1)任务在移动端完成的时间

20%在区间[0,50]随机产生;

20%在区间[50,100]随机产生;

15%在区间[100,150]随机产生;

10%在区间[150,200]随机产生;

15%在区间[200,500]随机产生;

20%在区间[500,1000]随机产生。

2)禁忌搜索参数设置

Tabu Search 禁忌表的长度和邻域都设置为 0.2 * N,迭代次数为 1000。

4.2 算法性能评估

图 2 给出了任务图宽度对算法性能的影响。各参数设定如下:任务间传输的数据量为 50 kB,上传/下载速率为 640/1408 Kbps,只有第一个节点和最后一个节点仅能在移动端完成,时间限制为全部在移动端执行的总时间的 85%,任务图深度固定为 10,宽度从 5 变化到 8。图 2 显示了总能量消耗随任务图宽度变大呈总体平稳且稍有上升的趋势。能耗最大的是将所有任务放在移动端执行的 non-offloading 算法,其次依次是 random offloading 算法、full offloading 算法、HA 和 TS 算法,TS 算法的解为最优。启发解和禁忌搜索算法的解较为接近,这也说明相比于禁忌搜索这种全局优化算法的解,启发解是高质量的近似解。

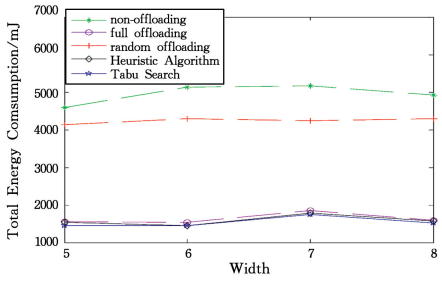


图 2 总能量消耗随任务图宽度的变化

Fig. 2 Changing of total energy amsumption with the width

图 3 给出了任务图深度对算法性能的影响。各参数设定如下:任务间传输的数据量为 50 kB,上传/下载速率为 640/1408 Kbps,只有第一个节点和最后一个节点仅能在移动端完成,时间限制为全部在本地执行的总时间的 85%,任务图宽度固定为 10,深度为 5~8。图 3 显示总能量消耗随任务图深度变大,呈总体平稳且稍有上升的趋势,这与宽度变化时的情况相同。能耗最大的是将所有任务放在移动端执行的 non-offloading 算法,其次依次是 random offloading 算法、full offloading 算法、HA 和 TS 算法,TS 算法的解为最优。TS 算法的解仍然是最优的,而 HA 的解与 TS 的解的能耗非常相近。当深度为 6 时,random offloading 算法的能耗大于 non-offloading 算法,这是由于当任务传输的能耗大于在移动端自身完成的能耗时迁移反而会增加能耗损失。

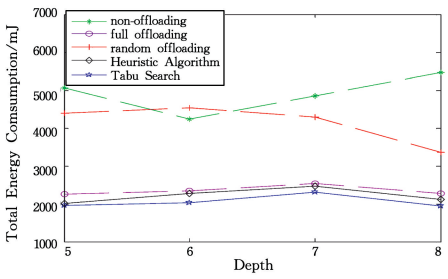


图 3 总能量消耗随任务图深度的变化

Fig. 3 Changing of total energy comsumption with task graph depth

图 4 给出了仅能在移动端执行的任务比例对算法性能的影响。各参数设定如下:任务间传输的数据量为 50 kB,上传/下载速率为 640/1408 Kbps,时间限制为全部在本地执行的总时间的 85%,任务图宽度固定为 15,深度为 10,仅能在移动

端完成的任务的比例为 0%，10%，…，100%。随着这一比例的增大，移动端的总能耗有所上升，这是因为可以迁移的任务减少，能耗无法转移，只能由移动端来完成更多的任务。

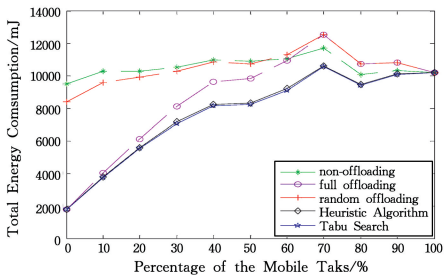


图 4　总能量消耗随移动端任务比例的变化

Fig. 4　Changing of total energy comsumption with percentage of mobile tasks

图 5 给出了任务数目（问题规模）变化对算法性能的影响。各参数设定如下：任务间传输的数据量为 50 kB，上传/下载速率为 640/1408 Kbps，仅能在移动端执行的任务的比例为 30%，时间限制为全部在移动端完成的总时间的 85%。随着任务数目的增加，移动端的能耗也快速增大，其中 non-offloading 和 random offloading 算法增长最快，HA 和 TS 的解非常接近，是所有算法中表现最优的。

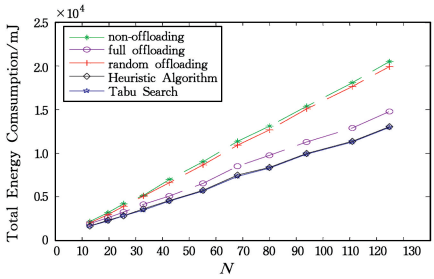


图 5　总能量消耗随任务数量的变化

Fig. 5　Changing of total energy comsumption with task N amount

为了测试各算法在运行时间方面的性能，在表 2 中列出图 4 的实验中各算法的运行时间。可以看出 non-offloading，full offloading 和 random offloading 算法几乎不花费任何时间，但其解是最差的。HA 和 TS 的解相近，根据本文之前的实验数据，TS 仅比 HA 提升了 1.5%，但 HA 的运行速度最多只是 TS 的 1/50。

表 2　各种算法运行时间的比较

Table 2　Comparison of running time of algorithms

（单位：ms）

N \ Algorithm	non-offloading	full offloading	random offloading	HA	TS
13	0.00	0.00	0.01	0.26	30.12
20	0.00	0.01	0.01	0.83	150.34
26	0.00	0.01	0.05	2.10	302.20
33	0.00	0.00	0.11	4.05	526.44
43	0.00	0.00	0.06	7.54	835.84
55	0.00	0.02	0.09	12.26	1217.08
68	0.00	0.00	0.21	19.09	1579.94
80	0.00	0.01	0.34	34.28	2286.67
94	0.01	0.00	0.35	48.40	3024.20
111	0.00	0.01	0.59	63.30	3923.05
125	0.00	0.02	0.68	104.69	5285.92

结束语　本文针对无线网络中应用程序任务图的调度和迁移问题，提出了一种快速高效的启发式算法与一种适用于此问题的禁忌搜索算法。构造的禁忌搜索算法以本文提出的启发式解为初始解，并对启发解进行进一步优化。雾计算和边缘服务器的迅猛发展，使得迁移的任务除了在较远的云端完成以外还可以在传送速度更快的边缘服务器上完成，如何更好地利用边缘服务加快任务处理的速度是我们未来的研究方向。

参 考 文 献

[1] DINH H T, LEE C, NIYATO D, et al. A survey of mobile cloud computing: architecture, applications, and approaches [J]. Wireless Communications & Mobile Computing, 2013, 13(18): 1587-1611.

[2] BARBAROSSA S, SARDELLITTI S, LORENZO P D. Communicating While Computing: Distributed mobile cloud computing over 5G heterogeneous networks [J]. IEEE Signal Processing Magazine, 2014, 31(6): 45-55.

[3] KUMAR K, LU Y H. Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? [M]. IEEE Computer Society Press, 2010.

[4] VALLINA-RODRIGUEZ N, CROWCROFT J. Energy Management Techniques in Modern Mobile Handsets [J]. IEEE Communications Surveys & Tutorials, 2013, 15(1): 179-198.

[5] KEPHART J O, CHESS D M. The Vision of Autonomic Computing [J]. Computer, 2003, 36(1): 41-50.

[6] SHU P, LIU F, JIN H, et al. eTime: Energy-efficient transmission between cloud and mobile devices[C]// INFOCOM, 2013 Proceedings IEEE. IEEE, 2013: 195-199.

[7] LIN Y D, CHU T H, LAI Y C, et al. Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds [J]. IEEE Systems Journal, 2015, 9(2): 393-405.

[8] ZHANG W, WEN Y, GUAN K, et al. Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel [J]. IEEE Transactions on Wireless Communications, 2013, 12(9): 4569-4581.

[9] MAHMOODI S E, SUBBALAKSHMI K P, SAGAR V. Cloud offloading for multi-radio enabled mobile devices[C]// IEEE International Conference on Communications. IEEE, 2015: 5473-5478.

[10] WU H, WANG Q, WOLTER K. Tradeoff between performance improvement and energy saving in mobile cloud offloading systems[C]// IEEE Conference on Communications Workshops. IEEE, 2013: 728-732.

[11] CUERVO E, BALASUBRAMANIAN A, CHO D K, et al. MAUI: making last longer with code offload[C]// International Conference on Mobile Systems, Applications, and Services. DBLP, 2010: 49-62.

[12] KOSTA S, AUCINAS A, HUI P, et al. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading [C] // INFOCOM, 2012 Proceedings IEEE. IEEE, 2012: 945-953.

（下转第 116 页）

一种近似算法,下一步将从理论上分析该算法与最优解之间的差距。

参 考 文 献

[1] The Climate Group. Smart 2020:Enabling the low carbon economy in the information age[R]. London,2008.

[2] LANGE C,KOSIANKOWSKI D,WEIDMANN R,et al. Energy Consumption of Telecommunication Networks and Related Improvement Options[J]. IEEE Journal of Selected Topics inQuantum Electronics,2011,17(2):285-295.

[3] CLARK D. The design philosophy of the DARPA internet protocols [C]// ACM SIGCOMM Computer Communication Review. 1988:106-114.

[4] FISHER W,SUCHARA M,REXFORD J. Greening backbone networks:reducing energy consumption by shutting off cables in bundled links[C]// Proceedings of the First ACM SIGCOMM Workshop on Green Networking. 2010:29-34.

[5] MINERAUD J,WANG L,BALASUB RAMANI A M S,et al. Hybrid renewable energy routing for ISP networks[C]// IEEE Conference on Computer Communications. 2016:1-9.

[6] YANG Y,WANG D,PAN D,et al. Wind Blows,Traffic Flows:Green Internet Routing under Renewable Energy[C]// IEEE Conference on Computer Communications. 2016.

[7] CHIARAVIGLIO,LUCA,MELLIA M,et al. Minimizing ISP Network Energy Cost:Formulation and Solutions[J]. IEEE/ACM Transactions on Networking,2012,20(2):463-476.

[8] LI Q,XU M,YANG Y,et al. Safe and Practical Energy-Efficient Detour Routing in IP Networks [J]. IEEE/ACM Transactions on Networking,2014,22(6):1925-1937.

[9] ZHANG M,YI C,LIU B,et al. GreenTE:Power-aware traffic-engineering[C]// IEEE International Conference on Network Protocols. IEEE Computer Society,2010:21-30.

[10] KRIST P. Scalable and Efficient Multipath Routing:Complexity and Algorithms[C]// 2015 IEEE 23rd International Conference on Network Protocols(ICNP). IEEE,2015:376-385.

[11] OHARA Y,IMAHORI S,METER R V. MARA:Maximum Alternative Routing Algorithm[C]// Proceedings of IEEE INFOCOM. 2009:298-306.

[12] KWONG K W,GAO L,ZHANG Z L,et al. On the feasibility and efficacy of protection routing in IP networks[J]. IEEE/ACM Transactions on Networking,2011,19(5):1543-1556.

[13] SPRING N,MAHAJAN R,WETHERALL D,et al. Measuring isp topologies with rocketfuel [J]. IEEE/ACM Transactions on Networking,2016,12(1):2-16.

[14] http://www.cs.bu.edu/brite.

[15] Power Management for the Cisco 12000 Series Router[OL]. http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12spower.html.

[16] CHABAREK J,SOMMERS J,BARFORD P,et al. Power-Awareness in Network Design and Routing[C]// INFOCOM 2008. 2008.

[17] RAHMAN M M,SAHA S,CHENGAN U,et al. IP Traffic Matrix Estimation Methods:Comparisons and Improvements[C]// ICC 2006. 2006:90-96.

[18] XU D H,CHIANG M,REXFORD J. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering [C]// INFOCOM 2008. 2008:1717-1730.

(上接第 99 页)

[13] ZHANG W,WEN Y,WU D O. Collaborative Task Execution in Mobile Cloud Computing Under a Stochastic Wireless Channel [J]. IEEE Transactions on Wireless Communications, 2015, 14(1):81-93.

[14] HUANG D,WANG P,NIYATO D. ADynamic Offloading Algorithm for Mobile Computing [J]. IEEE Transactions on Wireless Communications,2012,11(6):1991-1995.

[15] CHUN B G,IHM S,MANIATIS P,et al. Clone Cloud:elastic execution between mobile device and cloud[C]// Conference on Computer Systems. ACM,2011:301-314.

[16] MAHMOODI S E,SUBBALAKSHMI K P,SAGAR V. Cloud offloading for multi-radio enabled mobile devices[C]// IEEE International Conference on Communications. IEEE, 2015:5473-5478.

[17] MAHMOODI S E,UMA R N,SUBBALAKSHMI K P. Optimal Joint Scheduling and Cloud Offloading for Mobile Applications [J]. IEEE Transactions on Cloud Computing,2016,PP(99):1.

[18] BALAKRISHNAN P,THAM C K. Energy-Efficient Mapping and Scheduling of Task Interaction Graphs for Code Offloading in Mobile Cloud Computing [C] // IEEE/ACM International Conference on Utility and Cloud Computing. IEEE,2014:34-41.

[19] KOVACHEV D,YU T,KLAMMA R. Adaptive Computation Offloading from Mobile Devices into the Cloud[C] // International Symposium on Parallel and Distributed Processing with Applications. IEEE,2012:784-791.

[20] NIR M,MATRAWY A,ST-HILAIRE M. An energy optimizing scheduler for mobile cloud computing environments[C]// IEEE INFOCOM 2014-IEEE Conference on Computer Communications Workshops. IEEE,2014:404-409.

[21] BARBAROSSA S,SARDELLITTI S,LORENZO P D. Computation offloading for mobile cloud computing based on wide cross-layer optimization[C]// Future Network and Mobile Summit. IEEE,2013:1-10.

[22] RUBIN P. [OL]. http://orinanobworld.blogspot.de/2010/10/binary-variables-and-quadratic-terms.html.