移动云计算中的一种任务联合执行策略

柳 兴[1),2),3)] 李建彬[2)] 杨 震[4)] 李振军[3)]

[1)](中国联合网络通信有限公司 湖南省分公司 长沙 410014)

[2)](中南大学信息安全与大数据研究院 长沙 410083)

[3)](北京邮电大学信息与通信工程学院 北京 100876)

[4)](北京邮电大学计算机学院 北京 100876)

摘 要 针对移动用户将应用迁移至云端处理会引起大量的数据传输导致高能耗的问题,提出了一种任务联合执行策略(Task Collaborative Execution Policy,TCEP).首先,在云端和移动终端联合执行移动应用的前提下,将应用考虑为一系列的串行任务,根据任务的计算负荷、输入和输出数据量,把云端与移动终端联合移动应用的优化问题建模为最小化移动终端的能耗问题,并得出结论该优化问题属于 NPC(Nondeterministic Polynomial Complete)问题.接着,按移动终端需向云端迁移任务的次数来划分迁移策略集,并利用串行任务仅能一个接一个地执行的特点,给出了一次迁移最优特性.然后,通过对比串行任务与染色体的相似点,采用遗传算法来处理文中优化问题,并在简单遗传算法(Simple Genetic Algorithm,SGA)的基础上,利用一次迁移最优特性来设计交叉操作和变异操作,以便进一步提高算法性能.最后,通过仿真验证了所提策略及算法的性能,仿真结果表明,改进后的遗传算法具有良好的收敛性能,能够保证新个体具有仅向云端迁移一次的特性,与现有方法相比,所提策略可有效地减少搜索最优解的运算时间,能在满足应用执行时间要求的同时最小化移动终端的能耗.

关键词 云计算;移动互联网;应用迁移;能效;遗传算法

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2017.00364

A Task Collaborative Execution Policy in Mobile Cloud Computing

LIU Xing[1),2),3)] LI Jian-Bin[2)] YANG Zhen[4)] LI Zhen-Jun[3)]

[1)](*Hunan Branch of China UNICOM*, *Changsha* 410014)

[2)](*Institute of Information Safety and Big Data*, *Central South University*, *Changsha* 410083)

[3)](*School of Information and Communication Engineering*, *Beijing University of Post and Telecommunications*, *Beijing* 100876)

[4)](*School of Computer Science*, *Beijing University of Post and Telecommunications*, *Beijing* 100876)

Abstract The user can save the energy consumption on Mobile Devices (MDs) by offloaded mobile application to cloud, but it adds the transmission energy consumption of MDs. Focusing on this problem, a Task Collaborative Execution Policy (TCEP) is proposed in this paper. Firstly, mobile applications are considered as a series of tasks which can be executed by cloud or MD. Using the calculated load, the amount of input and output data of the task, we model the optimization problem as energy consumption minimization of MD for scene which the cloud and the MD jointly carry outa mobile application. Based ontheanalysis,onecandrawaconclusionthat the optimumenergy consumption of MD belong to NPC(Nondeterministic Polynomial Complete) problem. Next, considering the mobile application be jointly executed by the cloud and the MD, we divide the policies class into various set of the migration policy based on the number of offloading.

收稿日期:2016-01-20;在线出版日期:2016-10-04.本课题得到国家自然科学基金(61173017,61563038)、国家"八六三"高技术研究发展计划项目基金(2014AA01A701)、内蒙古自治区高等学校创新团队发展计划支持项目(NMGIRT-A1609)和 2013 年度湖南省教育厅科学研究项目(13C613)资助.**柳 兴**,男,1984 年生,博士,主要研究方向为云计算、移动云计算和移动互联网.E-mail:liuxing_bupt@qq.com.**李建彬**(通信作者),男,1968 年生,教授,主要研究领域为智慧医疗、大数据应用和信息安全.E-mail:jiaweihuang@csu.edu.cn.**杨 震**,男,1975 年生,博士,副教授,主要研究方向为云计算和移动多媒体.**李振军**,男,1975 年生,副教授,主要研究方向为云计算和移动教育.

According to the serial task just be executed one by one, we obtain the optimal feature of application offloading. Then, considering the serial task similar to chromosome, we turn to genetic algorithm solving the optimumenergy consumption. Using this offloading property, we improved the performance of genetic algorithm by designed the crossover operation and the mutation operation. At last the simulation validates the feasibility of the proposed policy and algorithm. Simulation results show that the proposed algorithm can ensure the new individual executing offloading no more than one time. Compared with existing methods, the proposed method can effectively improve the rate of convergence and reduce the operation time. Moreover, it can significantly save the energy consumption on the MT while meeting the application deadline.

Keywords　cloud computing; mobile internet; application offloading; energy efficiency; genetic algorithm

1　引　言

随着信息科技的发展、生活节奏的加快，移动化应用软件（如移动办公、移动支付、手机淘宝、微信等）逐渐被人们接受，并成为人们生产生活中不可缺少的一部分. 与此同时，电子科技的发展极大地提高了移动终端硬件方面的性能，使得移动终端可以承载智慧化程度更高的应用软件，进一步增加了人们对移动应用的依赖.

不难预测，在未来很长一段时间内，移动应用的智慧化程度会不断增强，以满足人们对智慧化应用的需求. 然而，硬件性能的改善往往落后于智慧化应用的发展，且移动终端受其尺寸的限制，始终存在计算能力弱、存储空间小以及电池续航时间短等问题[1]. 因此，移动终端的硬件性能必然成为移动应用智慧化发展的瓶颈.

为了弥补移动终端的不足，快速提高移动应用的智慧化程度，移动云计算技术应运而生. 移动云计算是一种将云计算与移动互联网相融合的新技术. 在移动云计算中，用户可以通过将应用迁移至云端处理的方式来弥补其移动终端的计算能力弱和存储空间小的缺陷，提高应用的服务质量[2-3]. 此外，在云端处理移动应用可极大地减少移动终端的能耗. 然而，应用迁移的引入必然会增加数据的传输量[4]，动态变化的无线信道会增加数据传输的不确定性. 当移动应用在较差的无线信道中进行迁移时会造成较高的数据丢包率，为完成应用迁移系统必须对丢弃的数据包进行重传，这不仅会导致较高的传输能耗而且还会导致较长的传输时间[5-6]，不仅影响移动终端的续航时间而且还会影响移动应用的服务质量. 因此，如何合理的进行应用迁移，既能保障移动应用的服务质量，又能减少移动终端的能耗是移动云计算急需解决的问题之一.

为减少移动终端的能耗，改善移动应用的服务质量，一些学者对移动云计算中的应用迁移问题进行了研究. 文献[7]将移动应用分为计算密集型和通信密集型两种，并指出计算密集型应用适合在云端执行，通信密集型应用适合在本地执行，而介于计算密集型与通信密集型之间的应用需要根据带宽情况来确定适合在哪里执行. 文献[8-9]进一步考虑移动应用可分解为一系列串行任务的场景，以计算密集型任务安排在云端执行和通信密集型任务安排在本地执行为设计原则，在无线随机信道中研究云端与移动终端联合执行移动应用的能效问题，并根据动态规划的逆推解法提出了一种调度算法，该算法获取的最优迁移策略能够在满足应用的执行时间要求下最小化移动终端的能耗. 然而，文献[8-9]是从最后一任务开始到第一个任务截止，根据信道状态聚个计算每个任务的执行位置，其计算时间随着任务数目成倍数增加. 当移动应用的任务较多时，难以在较短的时间内获取最优迁移策略.

本文将移动应用考虑为线性拓扑结构的任务模型，在动态变化的无线环境下，充分利用云端和移动终端的计算资源，提出了一种任务联合执行策略（Task Collaborative Execution Policy，TCEP）. 具体贡献如下：

（1）将云端与移动终端联合移动应用的优化问题建模为最小化移动终端的能耗问题，并通过分析判定该优化问题属于 NPC（Nondeterministic polynomial Complete）问题；

（2）推导了一次迁移可使应用执行时间最小的

特性,并结合文献[8]的研究结果得出一次迁移最优的结论;

(3)通过分析串行任务的特点选用遗传算法来计算全局最优解,并利用一次迁移最优特性来设计遗传算法的交叉操作和变异操作,减少算法的运算时间;

(4)通过分析一次迁移策略集合的策略总数和子个体的产生过程得出结论,所提算法具有较好的收敛性能,都能在较短的时间内计算出最优解;

(5)通过仿真验证了所提算法和策略的有效性.仿真结果表明,所提算法的收敛速度较快,所提策略能在满足应用执行时间限制的前提下最小化移动终端的能耗.

本文第 2 节概述相关工作;第 3 节详细描述移动应用的任务模型,给出研究目标;第 4 节分析移动应用的迁移特性,并给出一次迁移最优的结论;第 5 节利用一次迁移最优的结论来设计遗传算法搜索最优解;第 6 节分析所提算法的性能;第 7 节给出仿真结果及性能分析;第 8 节是结束语.

2 相关工作

目前针对移动云计算中的应用迁移问题研究很多,这些研究大致可以分为 3 类.第 1 类是在理想的信道情况下,研究移动应用的整体迁移问题.文献[7]首次将移动应用考虑为计算密集型和通信密集型两种,通过研究得出结论,计算密集型应用适合在云端执行,通信密集型应用适合在本地执行.文献[10]针对移动终端和云服务器的最小化能耗问题,根据博弈论提出了一种能效算法,该算法能在线性时间内获得纳什均衡解.文献[11]基于博弈论提出了一种分布式计算迁移策略,该策略能够根据应用规模有效地进行应用迁移.文献[12]针对移动云计算中的任务分配能效优化问题,提出了一种贪婪自动卸载算法来获得次优解.文献[13]考虑远端云数据中心、近端云片和移动终端组成的 3 层架构移动云计算系统,根据网络情况和任务执行时的能耗提出了一种迁移判决算法,该算法能在满足应用延时要求的前提下提高移动终端的能效.文献[14]提出了一种移动云计算的应用执行框架,该框架采用移动代理在应用层面支持应用在云上的执行,并提出了一种带宽分配机制确保计算资源能更好的得到利用.文献[15]考虑云端提供虚拟机执行移动应用的场景,针对云端大量执行移动应用会引起网络拥塞影响服务质量的问题,基于 Stackelberg 博弈提出了虚拟机定价与分配方案,该方案可在减少网络拥塞的同时减少用户和云提供商的开销.上述文献所考虑的场景和模型过于理想,脱离了实际,虽然在理论研究上具有一定的贡献,但是缺乏应用价值.

第 2 类是在动态变化的无线信道下,研究移动应用的整体迁移问题.文献[16]考虑移动应用的计算负荷和数据传输量等因素,根据信道的变化情况,提出了一种自动迁移模型,并采用分类法设计遗传算法来执行迁移判决.文献[17]针对无线信道的随机特性,利用信道条件较好时传输数据可提高能效的特点,基于最优停止理论提出了一钟信道感知的传输调度策略,该策略可在满足应用延时可接受的情况下最优化能效.文献[18-19]在随机信道条件下研究了一种迁移判决策略,该策略可为云端执行优化数据传输调度,为本地执行调整时钟频率,并通过阈值判决法来决定应用是在云端执行还是在本地执行,实现优化移动终端能耗的目标.文献[20]考虑在动态变化的无线信道场景下以用户当前连接条件为依据,并采用整数规划方法来设计迁移算法.文献[21]根据李雅普诺夫优化提出了一种动态迁移算法,该算法能够根据无线信道条件和应用的执行时间要求来优化移动终端的能耗.文献[22]在时断时续的无线信道中研究延时容忍型应用的数据传输问题,并基于李雅普诺夫优化提出了一种新的数据传输能效策略,该策略根据信道状态和数据排队长度来作出传输决策,可有效地改善移动终端的能效.文献[23]针对具有 3G、EDGE、WiFi 等多种接口的智能手机向云端上传图片、视频等延时容忍型应用的场景,基于基于李雅普诺夫优化提出了一种自适应连接选择算法,该算法可根据信道条件和上次数据排队情况自动作出数据上传决策.文献[24]针对移动云计算中的延时容忍型和数据密集型应用在多连接环境下进行传输调度的场景,提出了一种近似动态规划算法,该算法能优化系统的吞吐量和能效.文献[25]针对移动数据迁移的网络选择问题,针对网络条件不可预测的场景,基于李雅普诺夫优化提出提出了一种延时感知选择算法,针对网络条件可预测的场景,将网络条件与李雅普诺夫优化结合提出了一种网络感知选择算法.文献[26]针对当前 3G/4G 蜂窝小区的容量难以满足应用带宽需求的问题,在移动用户使用阅览服务场景下提出了一种服务推拉策略,该策略能根据用户订阅的多媒体内容进行预取,并能通过对连接质量的估计在适当的时间向用

户推送内容. 文献[27]考虑迁移数据的解码开销，提出了一种动态迁移方法，该方法能够高效地执行应用迁移，减少应用的执行时间和能耗. 然而，这些文献均只考虑了应用整体迁移的情况，在实际生活中一个移动应用往往由多个任务组成，且每个任务的计算负荷、输入和输出数据量都不同，如果将计算负荷小、输入和输出数据量大的任务迁移至云端执行同样会导致较长的执行时间和较高的能耗.

第 3 类是在动态变化的无线信道下，研究云端与移动终端联合执行移动应用的问题. 文献[28]考虑用户的需求和任务组织管理的变化，提出了一种联合任务管理模型. 文献[29]考虑用户志愿将其计算资源对外提供服务的场景，设计了一种自适应任务联合执行策略，并采用蚁群算法来获取最优解. 文献[30]考虑由并行任务构成的移动应用，在无线随机信道场景下研究联合执行的能效优化问题，并提出了一种联合任务执行调度算法，该算法能较快地获取最优解可最小化移动终端的能耗. 文献[31]考虑在无线链路不稳定时用户可提供计算资源的场景，提出了一种用户提供计算的平台，并在此基础上提出了一种自组织临界调度算法，该算法能在满足应用执行时间的前提下最小化移动终端的能耗. 文献[8-9]针对由串行任务构成的移动应用，考虑在无线随机信道中研究云端与移动终端联合执行移动应用的能效问题，并根据动态规划的逆推解法提出了一种调度算法，该算法获取的最优迁移策略能够在满足应用的执行时间要求下最小化移动终端的能耗. 上述研究虽然在一定程度上提高了应用的服务质量和移动终端的能效，但是不论是在应用场景上还是在设计方法上还有很多问题亟待研究.

本文是在分析文献[8-9]所研究的云端与移动终端联合执行移动应用的基础上，针对移动应用所组成的任务较多时，文献[8-9]所提方法难以在较短时间内获取最优迁移策略的缺点，考虑采用启发式算法来解决这一问题. 通过对串行任务的特点进行分析，选用遗传算法来计算最优解，并利用串行任务的迁移特性来设计遗传算法的交叉操作和变异操作，实现快速获取最优迁移策略的目标.

3　系统模型

与文献[9]类似，本文假设移动应用由一系列线性拓扑结构的任务所构成，研究云端与移动终端联合执行这类应用的能效问题. 本文所考虑的应用模型在实际生活有广泛的应用. 如人机对战中的国际象棋或中国象棋，机器每走一步其后台的计算都可以简单分解为几个串联的任务：首先计算每颗棋子可以移动的位置，然后计算每个棋子的最优移动位置，最后计算最优的移动棋子；又如循环迭代算法，每次迭代都可看作一个任务，前一次迭代运算的输出数据就是后一次迭代运算的输入数据，若 n 次迭代后满足算法的结束条件，便输出最后一次迭代运算的结果.

图 1 给出了移动应用的任务模型，应用由 n 个任务组成，整个应用的完成时间限制为 T_d. 对于任意任务 k，其计算负荷为 w_k，输入数据量为 α_k，输出数据量为 β_k. 所有任务只能按顺序逐个执行，任务在执行过程中没有外部数据输入，因此任务 $k-1$ 的输出数据为任务 k 的输入数据，即 $\beta_{k-1}=\alpha_k$[9]. 移动终端是移动应用的发起者，当用户需要使用移动应用时，会在本地通过触发该应用的开始状态 S 来. 开始状态 S 被触发后，应用被初始化并产生任务 1 的输入数据 α_1，应用开始执行. 对于任意中间任务 k，其执行者可能是移动终端也可能是云端，当任务 k 的执行者收到其输入数据 α_k 后便开始运算，任务 k 运算结束后执行者输出数据 β_k. 移动终端收到任务 n 的输出数据 β_n 后进入结束状态 D，应用结束. 若任务 n 在移动终端上执行，则完成任务 n 的执行便进入结束状态 D. 假设移动终端的时钟频率为 f_m，计算功率为 P_m，则移动终端执行任务 k 的时间为 $d_m(k)=w_k f_m^{-1}$，移动终端的能耗为 $e_m(k)=d_m(k)P_m$；同样假设云端的时钟频率为 f_c，移动终端的空闲功率为 P_i，则云端执行任务 k 的时间为 $d_c(k)=w_c f_c^{-1}$，此时移动终端的能耗为 $e_c(k)=d_c(k)P_i$[9].

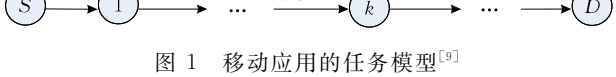


图 1　移动应用的任务模型[9]

假设移动终端仅通过一个无线信道与云端进行通信，并采用 Gilbert-Elliott(GE)信道模型来拟合该无线信道. 假设系统可侦测当前时隙 t 的信道状态 g_t，$g_t=g_G$ 和 $g_t=g_B$ 分别表示当前信道状态为“好”和“坏”. 假设移动终端的传输功率固定，因此传输速率由信道状态 g_t 唯一决定. 设信道在当前时隙 t 内的传输速率为 R_t，当 $g_t=g_G$ 时，信道的传输速率为 R_G，即 $R_t=R_G$；当 $g_t=g_B$ 时，信道的传输速率为 R_B，即 $R_t=R_B$. 进一步假设信道从状态 g_G 到状态 g_B 的转移概率为 p_{GB}，从状态 g_B 到状态 g_G 的转移概

率为 p_{BG}，则信道在状态 g_G 和状态 g_B 的稳态概率分别为 $\frac{p_{BG}}{p_{BG}+p_{GB}}$ 和 $\frac{p_{GB}}{p_{BG}+p_{GB}}$. 根据上述假设，可以推导出信道的平均传输速率为 $\mathbb{E}(R)=\frac{p_{BG}}{p_{BG}+p_{GB}}R_G+\frac{p_{GB}}{p_{BG}+p_{GB}}R_B$. 对于任意任务 k，若移动终端在分别在信道状态 $g_t=g_G$ 和 $g_t=g_B$ 时发送数据，则发送数据的期望时间分别为 $\mathbb{E}(d_s(k))=1+\max\left[0,\frac{\alpha_k-R_G}{\mathbb{E}(R)}\right]$ 和 $\mathbb{E}(d_s(k))=1+\max\left[0,\frac{\alpha_k-R_B}{\mathbb{E}(R)}\right]$. 若移动终端分别在信道状态 $g_t=g_G$ 和 $g_t=g_B$ 时接收数据，则接收数据的期望时间分别为 $\mathbb{E}(d_r(k))=1+\max\left[0,\frac{\beta_k-R_G}{\mathbb{E}(R)}\right]$ 和 $\mathbb{E}(d_r(k))=1+\max\left[0,\frac{\beta_k-R_B}{\mathbb{E}(R)}\right]$. 假设移动终端的发送功率和接收功率分别为 P_s 和 P_r，则移动终端的数据发送能耗和数据接收能耗分别为 $e_s(k)=\mathbb{E}(d_s(k))P_s$ 和 $e_r(k)=\mathbb{E}(d_r(k))P_r$.

本文的优化目标是在满足应用执行时间限制的同时最小化移动终端的能耗. 应用的执行时间和移动终端的能耗受任务的执行位置和信道状态的影响，根据上述假设和分析，可推导出完成一个移动应用的能耗和执行时间分别如下：

$$\mathbb{E}(e(p))=\sum_{k=1}^{n}\mathbb{E}(e_x(k)),\ p\in\mathbb{P} \tag{1}$$

$$\mathbb{E}(d(p))=\sum_{k=1}^{n}\mathbb{E}(d_x(k))\leq T_d \tag{2}$$

其中：$\mathbb{P}$ 为应用迁移策略集合；p 为应用迁移策略决定每个任务的执行位置；T_d 为执行时间要求；$e_x(k)$ 为执行执行任务 k 时移动终端产生的能耗；$d_x(k)$ 为执行任务 k 需要的时间. 若任意任务 k 在移动终端上执行，则有 $e_x(k)=e_m(k)+e_s(k)+e_r(k)$ 和 $d_x(k)=d_m(k)+d_s(k)+d_r(k)$，如果任务 $k-1$ 和 $k+1$ 都在移动终端上执行，则 $e_s(k)$、$e_r(k)$、$d_s(k)$ 和 $d_r(k)$ 均为 0；若任意任务 k 在云端执行，则有 $e_x(k)=e_c(k)+e_s(k)+e_r(k)$ 和 $d_x(k)=d_c(k)+d_s(k)+d_r(k)$. 如果任务 $k-1$ 和 $k+1$ 都在云端执行，则 $e_s(k)$、$e_r(k)$、$d_s(k)$ 和 $d_r(k)$ 均为 0.

从式(1)和(2)可以发现，每个任务均可在云端或移动终端上执行，即每个任务执行位置都有两种选择. 又因为所考虑的移动应用由 n 个任务组成，则应用迁移策略集合 $\mathbb{P}$ 中存在 2^n 种迁移策略，根据文献[32]可得出结论，式(1)和(2)所示优化问题属于 NPC 问题. NPC 问题计算最优解的时间是随着输入规模 n 成指数级增加，即计算式(1)和(2)所示优化问题的时间随任务数 n 的增加而迅速增加. 为此，本文接下来的工作就是设计一种搜索算法快速地获取最优的迁移策略，在满足应用执行时间限制的同时最小化移动终端的能耗.

4 迁移特性分析

对于由 n 个任务组成的移动应用，最坏情况下会出现云端与移动终端依次交替执行这 n 个任务，即任务 1,3,5,…在云端执行，任务 2,4,6,…在移动终端上执行，因此其迁移次数最多为 $\frac{n}{2}$ 次. 为了便于表述，本文称完成一个移动应用需要进行 k 次迁移的策略为 k 次迁移策略，其中 $1\leq k\leq\frac{n}{2}$. 文献[8]已证明了 1 次迁移策略可使移动终端的能耗最小，本文用引理给出该结论如下：

引理 1. 在马尔科夫随机信道中，若存在迁移策略 $p_e\in\mathbb{P}$，可最小化移动终端的能耗，则 p_e 中的迁移次数不大于 1.

根据引理 1 可分析出，能使移动终端能耗最小的迁移策略 p_e，若无需将任务迁移至云端执行，则所有任务都在移动终端上执行；若需要将部分任务迁移至云端执行，则 p_e 中只有一次任务迁移. 对于 p_e 中有任务迁移的情况，存在一个迁移至云端执行的任务和返回移动终端执行的任务，为便于表述，称该迁移至云端执行的任务为最佳迁移任务，返回移动终端执行的任务为最佳返回任务. 文献[8]给出了寻找最佳迁移任务和最佳返回任务的条件，分别用引理给出如下：

引理 2. 在马尔科夫随机信道中，假设任务 $i-1$ 已在移动终端上执行完成，当前信道状态为 g_t，若把任务 i 迁移至云端执行可最小化移动终端的能耗，则任务 i 满足：

$$\max\left[0,\frac{\alpha_i-R_t}{w_i\mathbb{E}[R]}\right]-\frac{\beta_i}{w_i\mathbb{E}[R]}+\frac{1}{w_i}<\frac{1}{P_s}\left(\frac{P_m}{f_m}-\frac{P_i}{f_c}\right) \tag{3}$$

引理 3. 在马尔科夫随机信道中，假设任务 $j-1$ 已在云端执行完成，当前信道状态为 g_t，若把任务 j 返回至移动终端执行可最小化移动终端的能耗，则任务 j 满足：

$$\max\left[0,\frac{\alpha_j-R_t}{w_j\mathbb{E}[R]}\right]-\frac{\beta_j}{w_j\mathbb{E}[R]}+\frac{1}{w_j}<\frac{1}{P_r}\left(\frac{P_i}{f_c}-\frac{P_m}{f_m}\right) \tag{4}$$

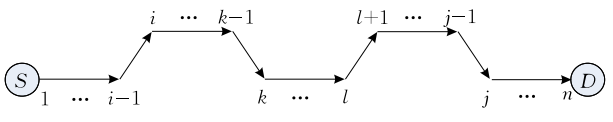
受引理 1 的启发，云端与移动终端联合执行移动应用的时间也可能存在类似的性质，下面对这一假设进行分析. 在本文考虑的串行任务模型中，若将相邻的两个任务均放在移动终端或云端执行，可减

少任务的输入数据在无线信道中传输，相应的数据传输时间为 0. 通常情况下，云端的时钟频率要快于移动终端，将任务迁移至云端执行可减少执行时间. 根据这些特点，可得出最小执行时间迁移策略的特性，用定理给出如下.

定理 1.　在马尔科夫随机信道中，若存在迁移策略 $p_d \in \mathbb{P}$，可最小化移动应用的执行时间，则 p_d 中的迁移次数不大于 1.

证明.　由于移动终端在开始状态 S 产生数据 α_1 后，应用开始执行；收到任务 n 的输出数据 β_n 后进入结束状态 D，应用结束. 因此，应用的开始状态 S 和结束状态 D 都在移动终端上执行. 如果任务 1 在移动终端上执行的时间小于其输出数据的上传时间，则将任务 1 安排在移动终端上执行更有利于减少执行时间；任务 1 和 2 在移动终端上执行的时间小于任务 2 输出数据的上传时间，则将任务 1 和 2 安排在移动终端上执行更有利于减少执行时间. 如此类推，不妨设任务 1 到 $i-1$ 安排在移动终端上执行更有利于减少执行时间. 如果任务 n 在移动终端上执行的时间与其输入数据的下载时间之和小于其输出数据的下载时间，则将任务 n 安排在移动终端上执行更有利于减少执行时间. 同样不妨设任务 j 到 n 安排在移动终端上执行更有利于减少执行时间. 根据上述假设，任务 i 是第一次迁移至云端执行的任务，任务 j 是最后一次返回移动终端执行的任务. 下面采用反证法证明一次迁移策略可使移动应用的执行时间最小.

假设一次迁移策略不能使得移动应用的执行时间最小，图 2(a)所示两次迁移中，任务 i 到 $k-1$ 在云端执行，任务 k 到 l 在移动终端执行，任务 $l+1$ 到 $j-1$ 在云端执行. 图 2(b)所示两次迁移中，任务 i 到 $j-1$ 在云端执行. 与图 2(a)不同，图 2(b)所示一次迁移中，任务 k 到 l 也在云端执行，即任务 $l+1$ 到 $j-1$ 均在云端执行.



(a) 两次迁移

(b) 一次迁移

图 2　两次迁移和一次迁移

对于任务 k，由于 $f_c > f_m$，则有

$$w_k f_c^{-1} < w_k f_m^{-1} < w_k f_m^{-1} + \mathbb{E}[d_r(k-1)] \quad (5)$$

其中，$\mathbb{E}[d_r(k-1)]$ 为移动终端接收任务 $k-1$ 输出数据的时间.

同理，对于任务 l，则有

$$w_k f_c^{-1} < w_l f_m^{-1} < w_l f_m^{-1} + \mathbb{E}[d_s(l)] \quad (6)$$

对于任务 h，则有

$$w_h f_c^{-1} < w_h f_m^{-1},\ h = k+1, \cdots, l-1 \quad (7)$$

由式(5)、(6)和式(7)可推得，图 2(b)所示迁移策略的执行时间小于图 2(a)所示迁移策略. 又因为，由 n 个任务组成的移动应用的迁移次数最多为 $\frac{n}{2}$ 次. 同理，可以证明 $k-1$ 次迁移策略的执行时间小于 k 次迁移策略，其中 $3 \le k \le \frac{n}{2}$. 根据递归性可得出结论，一次迁移策略能使得移动应用的执行时间最小，与假设矛盾. 因此，最小执行时间迁移策略的迁移次数不大于 1.　　证毕.

由定理 1 可知，迁移策略集合 $\mathbb{P}$ 中存在某迁移策略可使得移动应用的执行时间最小，若该迁移策略需要将部分任务迁移至云端处理，则存在最佳迁移任务和最佳返回任务，分别用定理给出如下.

定理 2.　在马尔科夫随机信道中，假设任务 $i-1$ 已在移动终端上执行完成，当前时隙 t 的信道状态为 g_t，若把任务 i 迁移至云端执行可最小化应用的执行时间，则任务 i 满足：

$$\max\left[0, \frac{\alpha_i - R_t}{w_i \mathbb{E}[R]}\right] - \frac{\beta_i}{w_i \mathbb{E}[R]} + \frac{1}{w_i} < \frac{1}{f_m} - \frac{1}{f_c} \quad (8)$$

证明.　根据假设，移动终端已完成任务 $i-1$ 的执行，为辅助证明继续假设任务 j 返回移动终端执行可最小化应用的执行时间. 若将任务 i 迁移至云端执行，则整个应用的执行时间为

$$\begin{aligned} D_i &= \sum_{k=1}^{i-1} \frac{w_k}{f_m} + \mathbb{E}(d_s(i)) + \sum_{k=i}^{j-1} \frac{w_k}{f_c} + \\ &\quad \mathbb{E}(d_r(j-1)) + \sum_{k=j}^{n} \frac{w_k}{f_m} \\ &= \begin{cases} \sum_{k=1}^{i-1} \frac{w_k}{f_m} + 1 + \max\left[0, \frac{\alpha_i - R_G}{\mathbb{E}(R)}\right] + \\ \quad \sum_{k=i}^{j-1} \frac{w_k}{f_c} + \frac{\beta_{j-1}}{\mathbb{E}[R]} + \sum_{k=j}^{n} \frac{w_k}{f_m}, & R_t = R_G \\ \sum_{k=1}^{i-1} \frac{w_k}{f_m} + 1 + \max\left[0, \frac{\alpha_i - R_B}{\mathbb{E}(R)}\right] + \\ \quad \sum_{k=i}^{j-1} \frac{w_k}{f_c} + \frac{\beta_{j-1}}{\mathbb{E}[R]} + \sum_{k=j}^{n} \frac{w_k}{f_m}, & R_t = R_B \end{cases} \end{aligned} \quad (9)$$

若仍将任务 i 放在移动终端上执行，而任务 $i+1$ 迁

移至云端执行，则整个应用的执行时间为

$$D_i'=\sum_{k=1}^{i}\frac{w_k}{f_m}+\frac{\alpha_{i+1}}{\mathbb{E}[R]}+\sum_{k=i+1}^{j-1}\frac{w_k}{f_c}+\frac{\beta_{j-1}}{\mathbb{E}[R]}+\sum_{k=j}^{n}\frac{w_k}{f_m} \quad (10)$$

当 $R_t=R_G$ 时，若将任务 i 迁移至云端执行可使移动应用的执行时间最小，则有 $D_i<D_i'$，用公式表达如下：

$$D_i-D_i'=$$

$$\frac{w_i}{f_c}-\frac{w_i}{f_m}+1+\max\left[0,\frac{\alpha_i-R_G}{\mathbb{E}(R)}\right]-\frac{\alpha_{i+1}}{\mathbb{E}(R)}<0 \quad (11)$$

又因为，任务 i 的输出数据是任务 $i+1$ 的收入数据，即 $\alpha_{i+1}=\beta_i$，则式(11)可写为

$$\max\left[0,\frac{\alpha_i-R_G}{w_i\mathbb{E}[R]}\right]-\frac{\beta_i}{w_i\mathbb{E}[R]}+\frac{1}{w_i}<\frac{1}{f_m}-\frac{1}{f_c} \quad (12)$$

同理，当 $R_t=R_B$ 时，根据式(9)和(10)可推出：

$$\max\left[0,\frac{\alpha_i-R_B}{w_i\mathbb{E}[R]}\right]-\frac{\beta_i}{w_i\mathbb{E}[R]}+\frac{1}{w_i}<\frac{1}{f_m}-\frac{1}{f_c} \quad (13)$$

结合式(12)和(13)，可推得式(8)． 证毕．

定理 3. 在马尔科夫随机信道中，假设任务 $j-1$ 已在云端执行完成，当前时隙 t 的信道状态为 g_t，若把任务 j 返回至移动终端执行可最小化应用的执行时间，则任务 j 满足：

$$\max\left[0,\frac{\alpha_j-R_t}{w_j\mathbb{E}[R]}\right]-\frac{\beta_j}{w_j\mathbb{E}[R]}+\frac{1}{w_j}<\frac{1}{f_c}-\frac{1}{f_m} \quad (14)$$

证明． 根据假设，云端已完成任务 $j-1$ 的执行，为辅助证明继续假设任务 i 迁移至云端执行可最小化应用的执行时间．若将任务 j 返回至移动终端执行，则整个应用的执行时间为

$$D_j=\sum_{k=1}^{i-1}\frac{w_k}{f_m}+\mathbb{E}(d_s(i))+\sum_{k=i}^{j-1}\frac{w_k}{f_c}+\mathbb{E}(d_r(j))+\sum_{k=j}^{n}\frac{w_k}{f_m}$$

$$=\begin{cases}\sum_{k=1}^{i-1}\frac{w_k}{f_m}+\frac{\alpha_i}{\mathbb{E}(R)}+\sum_{k=i}^{j-1}\frac{w_k}{f_c}P_i+1+\\ \quad\max\left[0,\frac{\beta_{j-1}-R_G}{\mathbb{E}(R)}\right]+\sum_{k=j}^{n}\frac{w_k}{f_m},\ R_t=R_G\\ \sum_{k=1}^{i-1}\frac{w_k}{f_m}+\frac{\alpha_i}{\mathbb{E}(R)}+\sum_{k=i}^{j-1}\frac{w_k}{f_c}P_i+1+\\ \quad\max\left[0,\frac{\beta_{j-1}-R_B}{\mathbb{E}(R)}\right]+\sum_{k=j}^{n}\frac{w_k}{f_m},\ R_t=R_B\end{cases} \quad (15)$$

若仍将任务 j 放在云端执行，而任务 $j+1$ 返回至移动终端执行，则整个应用的执行时间为

$$D_j'=\sum_{k=1}^{i-1}\frac{w_k}{f_m}+\frac{\alpha_i}{\mathbb{E}[R]}+\sum_{k=i+1}^{j}\frac{w_k}{f_c}+\frac{\beta_j}{\mathbb{E}[R]}+\sum_{k=j+1}^{n}\frac{w_k}{f_m} \quad (16)$$

当 $R_t=R_G$ 时，若将任务 j 返回至移动终端执行可使移动应用的执行时间最小，则有 $D_i<D_i'$，用公式表达如下：

$$D_i-D_i'=$$

$$\frac{w_j}{f_m}-\frac{w_j}{f_c}+1+\max\left[0,\frac{\beta_{j-1}-R_G}{\mathbb{E}(R)}\right]-\frac{\beta_j}{\mathbb{E}(R)}<0 \quad (17)$$

又因为，任务 j 的输出数据是任务 $j+1$ 的收入数据，即 $\alpha_j=\beta_{j-1}$，则式(17)可写为

$$\max\left[0,\frac{\alpha_j-R_G}{w_j\mathbb{E}[R]}\right]-\frac{\beta_j}{w_j\mathbb{E}[R]}+\frac{1}{w_j}<\frac{1}{f_c}-\frac{1}{f_m} \quad (18)$$

同理，当 $R_t=R_B$ 时，根据式(15)和(16)可推出：

$$\max\left[0,\frac{\alpha_j-R_B}{w_j\mathbb{E}[R]}\right]-\frac{\beta_j}{w_j\mathbb{E}[R]}+\frac{1}{w_j}<\frac{1}{f_c}-\frac{1}{f_m} \quad (19)$$

结合式(18)和(19)，可推得式(14)成立． 证毕．

根据文献[8]证明引理 2 和引理 3 的过程可获得最小能耗策略 p_e 的执行时间 $d(p_e)$．同样，根据定理 2 和定理 3 的证明过程可获得最小执行时间策略 p_d 的执行时间 $d(p_d)$．比较 $d(p_e)$、$d(p_d)$ 和 T_d 可发现，当 $d(p_d)>T_d$ 时，最优策略不存在；当 $d(p_e)\leq T_d$ 时，p_e 为最优策略；当 $d(p_d)<T_d<d(p_e)$ 时，最优策略存在，但 p_e 不是最优策略．前两种情况属于特殊情况，而第 3 种情况属于一般性情况，难以通过条件判决方式来获取最优策略．考虑到本文研究的优化问题属于 NPC 问题，因此接下来的主要工作就是利用上述引理和定理的结论，通过设计一种启发式算法快速地获得最优策略．

5 算法设计

根据上述分析，如果根据定理 1 获得的最小执行时间策略 p_d，满足 $d(p_d)\leq T_d$，则在一次迁移策略集合中就能找到最优迁移策略，因此，算法的设计思想就是围绕着一次迁移特性在一次迁移策略集合中搜索最优迁移策略．本文所考虑的串行任务模型类似于染色体，故此我们采用遗传算法来搜索最优迁移策略．遗传算法是一种借鉴生物界的进化规律而设计的随机化搜索方法[33]．根据生物进化规律，通过设计交叉和变异操作不断获得新个体，并对新个体进行筛选，如此反复迭代，直到获得最优解为止．为了便于区分，本文称这种未考虑应用场景及问题特性而设计的遗传算法为简单遗传算法(Simple Genetic Algorithm，SGA)．SGA 算法并非针对本文所研究问题的具体应用场景及特性而设计，存在运算时间长和局部性问题．为此，本文根据一次迁移最优特性，在 SGA 算法的基础上加入一些改进方案来设计一种新的遗传算法．由于每个任务均可在云端或移动终端上执行，为了便于表述，在染色体的基因

座中引入“$^-$”加以区别，例如：$\overline{u_k}$表示任务 k 在云端执行；u_k表示任务 k 在移动终端上执行.

5.1　初始群体选择

本算法是利用一次迁移特性来进行设计，为提高算法的收敛速度，初始种群也应是一次迁移策略，其选择方法如下：(1) 加入所有任务均在云端执行的个体，即 $\boldsymbol{u}_{cl}=[\overline{u_1},\overline{u_2},\cdots,\overline{u_n}]$，以确保移动终端在计算能力非常弱时或者计算负荷非常高时有迁移策略输出；(2) 加入所有任务都在移动终端上执行的个体，即 $\boldsymbol{u}_{lo}=[u_1,u_2,\cdots,u_n]$，以确保算法在信道状态非常糟糕时有迁移策略输出；(3) 为快速获取一批最佳任务迁移的子个体和最佳任务返回的子个体，选择一些仅一个任务在云端执行的个体，且这些个体的能耗不大于个体 $\boldsymbol{u}_{cl}$ 和个体 $\boldsymbol{u}_{lo}$. 例如：选取第 k 个任务在云端执行的个体，即 $\boldsymbol{u}_{\overline{k}}=[u_1,u_2,\cdots,u_{k-1},\overline{u_k},u_{k+1},\cdots,u_n]$，则该个体必须满足：$e(\boldsymbol{u}_{\overline{k}})\leq\min[e(\boldsymbol{u}_{lo}),e(\boldsymbol{u}_{cl})]$，其中 $e(\boldsymbol{u}_{cl})=e_s(1)+\sum_{k=1}^{n}e_c(k)+e_r(n)$，$e(\boldsymbol{u}_{lo})=\sum_{k=1}^{n}e_m(k)$，$e(\boldsymbol{u}_{\overline{k}})=e_s(k)+\sum_{i\neq k}e_m(i)+e_r(k)+e_c(k)$.

5.2　适应度函数

为便于在选择过程中对个体进行选择，我们设计能直接反应个体的性能的适应度函数. 性能好(满足 T_d 约束且能耗较小)的个体适应度大，性能差(不满足 T_d 约束或能耗较大)的个体适应度小. 在选择个体时需要考虑能耗和执行时间两个因素，因此采用加权的方式来定义适应度函数，用公式表达如下：

$$f(T)=Af_e+Bf_d \tag{20}$$

其中：A 和 B 为正加权系数；f_e 和 f_d 的数学表达式分别为

$$f_e=\Phi_e(e(p)-\min[e(\boldsymbol{u}_{lo}),e(\boldsymbol{u}_{cl})]) \tag{21}$$

$$f_d=\Phi_d[d(p)-T_d] \tag{22}$$

其中：$e(p)$和 $d(p)$分别表述迁移策略 p 的能耗和执行时间；Φ_e和 Φ_d分别表述迁移策略 p 在能耗和执行时间方面的惩罚函数，其数学表达式分别为

$$\Phi_e=\begin{cases}1, & e(p)\leq\min[e(\boldsymbol{u}_{lo}),e(\boldsymbol{u}_{cl})]\\ r_e, & e(p)>\min[e(\boldsymbol{u}_{lo}),e(\boldsymbol{u}_{cl})]\end{cases} \tag{23}$$

$$\Phi_d=\begin{cases}1, & e(p)\leq T_d\\ r_d, & e(p)>T_d\end{cases} \tag{24}$$

根据式(23)和(24)，当个体的能耗小于$\min[e(\boldsymbol{u}_{lo}),e(\boldsymbol{u}_{cl})]$时，个体在能耗方面的性能较好，其能耗惩罚函数 Φ_e的值为 1，否则个体在能耗方面的性能较差，其能耗惩罚函数 Φ_e 的值为 $r_e(0<r_e<1)$. 同样，当 $e(p)\leq T_d$时，执行时间惩罚函数 Φ_d 的值为 1，否则为 $r_d(0<r_d<1)$. 如果 r_e和 r_d取值较大，新个体的适应性较强，在运算初期能快速增加种群的多样性，但在运算后期完成每代计算的时间也会增长；如果 r_e和 r_d取值较小，新个体的适应性较弱，增加种群多样性的速度较慢，需要通过多代繁衍才能丰富种群的多样性，同样会增加运算时间. 目前没有较好的方法，通过选取合适的 r_e和 r_d来最小化算法的运算时间，简单起见，本文参考文献[33]，选取 $r_e=r_d=0.5$.

5.3　选择方法

本算法采用最佳个体保存的方式作为选择方法，即选出最佳个体，直接遗传到子代群体. 其余个体采用比例选择法进行选择，个体被选择的概率与其适应值大小成正比，个体的适应值可通过式(20)获得. 对于第 i 个个体，被选择的概率为

$$sp_i=\frac{f(T_i)}{\sum_{i=1}^{N_p}f(T_i)} \tag{25}$$

其中，N_p为群体中的个体总数.

5.4　交叉操作

本算法的交叉操作是在 SGA 算法的基础上进行设计. 在 SGA 算法的交叉操作中，仅对被选个体的部分基因进行交互，而不对新个体做进一步处理. 具体操作如下：

首先，随机选择两个即将进行交叉操作的个体，即父代个体；接着，在任意一个父代个体上随机选取一个交叉点，同时将该个体交叉点的位置标记为另一个父代个体的交叉点；然后，交换这两个父代个体在交叉点之前的基因，并得到两个新个体，即子代个体.

例如，随机选择两个即将配对的父代个体 $\boldsymbol{u}_{f1}$ 和 $\boldsymbol{u}_{f2}$，并随机选取交叉点：

$\boldsymbol{u}_{f1}=[u_1,u_2,\cdots,\overline{u_{k-2}},\overline{u_{k-1}},|\overline{u_k},\overline{u_{k+1}},u_{k+2},\cdots,u_n]$，

$\boldsymbol{u}_{f2}=[u_1,u_2,\cdots,u_{k-2},u_{k-1},|u_k,\overline{u_{k+1}},\overline{u_{k+2}},\cdots,u_n]$，

其中“|”表示交叉点，对“|”之前的基因进行交叉后得到两个新个体如下：

$\boldsymbol{u}_{s1}^*=[u_1,u_2,\cdots,u_{k-2},u_{k-1},|\overline{u_k},\overline{u_{k+1}},u_{k+2},\cdots,u_n]$，

$\boldsymbol{u}_{s2}^*=[u_1,u_2,\cdots,\overline{u_{k-2}},\overline{u_{k-1}},|u_k,\overline{u_{k+1}},\overline{u_{k+2}},\cdots,u_n]$.

显然，采用 SGA 算法得到的新个体可能不是一次迁移策略，如 $\boldsymbol{u}_{s2}^*$. 这会导致算法的收敛速度慢，甚至出现局部性问题.

为此，本文在上述过程中增加一个改进方案. 该方案在获得新个体后，对其染色体进行检测，并通过修改基因的方式确保每个新个体都是一次迁移策略. 具体操作如下：

首先检测两个新个体的染色体,判断是否满足一次迁移特性,例如检测 $\boldsymbol{u}_{s2}^{*}$ 会发现存在一段染色体 $\overline{u_{k-1}},u_k,\overline{u_{k+1}}$,可判断 $\boldsymbol{u}_{s2}^{*}$ 不是一次迁移策略;然后通过修正不满足一次迁移特性的个体的染色体,将该个体变为一次迁移策略,例如将 $\boldsymbol{u}_{s2}^{*}$ 中的基因 u_k 修正为 $\overline{u_k}$,并最终得到满足一次迁移特性的子个体:

$$\boldsymbol{u}_{s1}=[u_1,u_2,\cdots,u_{k-2},u_{k-1},\overline{u_k},\overline{u_{k+1}},u_{k+2},\cdots,u_n],$$

$$\boldsymbol{u}_{s2}=[u_1,u_2,\cdots,\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},\cdots,u_n].$$

采用改进后的交叉操作获得的子个体均为一次迁移策略,具有较好的适应性,能有效地提高了算法的收敛速度.

5.5 变异操作

本算法的交叉操作同样是在 SGA 算法的基础上进行设计.在 SGA 算法的变异操作中,仅对被选个体的某个基因进行变异,且变异概率为 p_m.

例如,随机选取个体 $\boldsymbol{u}_{f3}$ 作为即将变异的父代个体:

$$\boldsymbol{u}_{f3}=[u_1,\cdots,u_{k-4},u_{k-3},\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},u_{k+3},\cdots,u_n].$$

个体 $\boldsymbol{u}_{f3}$ 上的任意基因发送变异的概率为 p_m,若基因 u_i 发生变异,且有 $1<i<k-3$,则得到的新个体为

$$\boldsymbol{u}_{s3}^{*}=[u_1,\cdots,\overline{u_i},\cdots,u_{k-3},\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},u_{k+3},\cdots,u_n].$$

显然,$\boldsymbol{u}_{s3}^{*}$ 不是一次迁移策略,可以得出结论,SGA 算法变异操作能够增加群体的多样性,但是很难保证新个体是一次迁移策略,增加了算法获取最优解的难度.

为此,本文对上述变异操作进行重新设计,具体规则如下:(1)需要通过无线传输数据的任务对应的基因发生变异的概率 $P(A)=p_m$;(2)需要通过无线传输数据的任务对应的基因发生变异时,与之相距 k 个位置的基因也发生变异的概率 $P(B|A)=p_m^k$;(3)需要通过无线传输数据的任务对应的基因,以及与之相距 k 个位置的基因都发生变异时,则它们之间的基因也会发生变异.

例如,同样选取个体 $\boldsymbol{u}_{f3}$ 作为即将变异的父代个体:

$$\boldsymbol{u}_{f3}=[u_1,\cdots,u_{k-4},u_{k-3},\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},u_{k+3},\cdots,u_n].$$

个体 $\boldsymbol{u}_{f3}$ 中的基因 $u_{k-3},\overline{u_{k-2}},\overline{u_{k+2}}$ 和 u_{k+3} 均需要通过无线传输数据,它们发生变异的概率均为 p_m.如果基因 u_{k-3} 发生变异,则变异后的新个体为

$$\boldsymbol{u}_{s3}=[u_1,\cdots,u_{k-4},\overline{u_{k-3}},\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},u_{k+3},\cdots,u_n].$$

基因 u_{k-3} 发生变异时,与之相距 2 个位置的基因 u_{k-5} 也发生变异的概率为 p_m^2,即得到新个体为

$$\boldsymbol{u}_{s4}=[u_1,\cdots,\overline{u_{k-5}},\overline{u_{k-4}},\overline{u_{k-3}},\overline{u_{k-2}},\overline{u_{k-1}},\overline{u_k},\overline{u_{k+1}},\overline{u_{k+2}},\cdots,u_n].$$

由于基因 u_{k-3} 与 u_{k-5} 同时发送变异,因此通过变异操作获得 $\boldsymbol{u}_{s4}$ 的概率为 p_m^3.不难发现,改进后的变异操作获得的新个体为一次迁移策略,能较好的解决局部性问题.

6 算法性能分析

根据第 5 节的算法设计,初始群体选择了所有任务都在云端执行的个体和一些仅一个任务在云端执行的个体,这些个体都是一次迁移策略.为了保证在信道状态很差时有迁移策略输出,初始群体还选取了所有任务都在移动终端上执行的个体.交叉操作获得新生个体后会对其染色体进行检测,并修正不合格个体的染色体以保证每个新生个体都具有一次迁移特性.变异操作是采用条件变异方式,能保证新个体具有一次迁移特性.为了区别 SGA 算法,称本文所设计的算法为一次迁移遗传算法(Once Migration Genetic Algorithm,OM-GA).

根据引理 1 和定理 1 可以得出结论,全局最优解位于一次迁移策略集合中.OM-GA 算法的初始群里以及交叉操作和变异操作产生的子个体均属于一次迁移策略,因此 OM-GA 算法是在一次迁移策略集合中搜索全局最优解,而 SGA 算法是在应用迁移策略集合中搜索全局最优解.一次迁移策略集合和应用迁移策略集合中的策略数分别为 $\frac{n(n-1)}{2}$ 和 2^n,OM-GA 算法的计算量要明显低于 SGA 算法,且其优势随着任务数 n 的增加而愈发明显.

当初始群体较大时,只需一次变异操作就能获得多个具有最佳迁移任务或最佳返回任务的个体.然后,再通过交叉操作将具有最佳迁移任务的个体与具有最佳返回任务的个体进行交叉便可得到最优迁移策略.因此,OM-GA 算法最快可通过 2 代进化就能获得最优迁移策略,任务数 n 对进化代数的影响不大.

7 仿真结果与性能分析

本文仿真分两部分,首先通过 3 个实验从算法的收敛性、运算时间以及输出策略的任务执行位置

对本文 OM-GA 算法的自身性能进行了分析，并与文献[20]提出的基于聚合开销的拉格朗日松弛(Lagrangian Relaxation Based Aggregated Cost, LARAC)算法进行比较，验证 OM-GA 算法在收敛性能和运算时间上的优势. 然后通过 2 个实验将本文 TCEP 策略与本地执行策略、云端执行策略、LARAC 策略[20]以及阈值判决策略进行比较，其中阈值判决策略是根据带宽和计算量来判决整个应用是在云端执行还是在本地执行，验证在不同的$\mathbb{E}[R]$和 f_m下，TCEP 策略对移动终端能耗的改善情况.

仿真中，考虑云端尽可能靠近用户的场景，假设数据在有线网络中的传输时间非常小，仿真时不考虑该传输时间. 移动终端只需完成数据在无线信道中的传输，云端便可开始执行任务. 与系统模型一致，无线信道采用 GE 信道模型来拟合. 假设信道的上下行传输速率相同，信道在 g_G 和 g_B 状态的传输速率分别为 200 Kbit/s 和 10 Kbit/s，信道从 g_G 到 g_B 的转移概率 $p_{GB}=0.005$，从 g_B 到 g_G 的转移概率为 $p_{BG}=0.005$.

第 1 个实验比较 OM-GA 算法与 SGA 算法的收敛性. 假设移动应用由 10 个任务组成，输入数据为 $\alpha=\{10,4,2,3,20,10,2,1,5,40\}$ Kbit，输出数据为 $\beta=\{4,2,3,3,20,10,2,15,40,30\}$ Kbit，计算负荷为 $w=\{2,2,50,30,20,40,55,15,20\}$ Mcycles. 采用移动终端的能耗作为输出策略是否收敛的衡量指标，随着进化代数的递进，当输出策略不能使移动终端的能耗继续降低时，则认定算法收敛，输出策略为最优策略. 仿真结果如图 3 所示，从图中可以发现：(1) OM-GA 算法的收敛速度要快于 SGA 算法，验证了采用一次迁移最优特性来设计遗传算法可提高算法的收敛速度；(2) OM-GA 算法在第 3 代就能获得最优策略，这主要是因为采用一次迁移最优特性来设计遗传算法的交叉过程和变异过程后，一次变异操作就能得到多个最优迁移任务或者最优返回任务的子个体，随后通过将最优迁移任务的个体与最优返回任务的个体进行交叉操作就能获得最优策略，因此通过 3 代进化就能获得全局最优解.

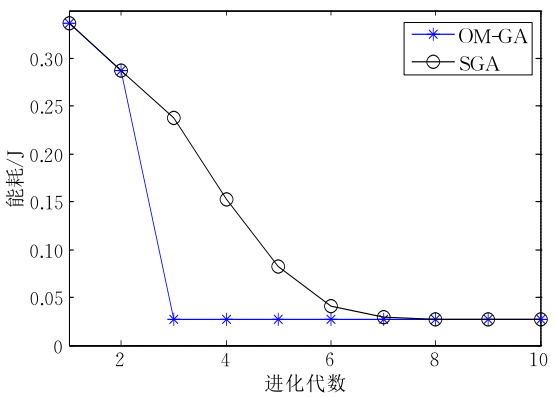


图 3　收敛性能比较

第 2 个实验比较 OM-GA 算法、SGA 算法和 LARAC 算法的运算时间. 假设由 10 个任务组成的移动应用，其输入数据为 $\alpha=\{10,4,2,3,20,10,2,1,5,40\}$ Kbit，输出数据为 $\beta=\{4,2,3,3,20,10,2,15,40,30\}$ Kbit，计算负荷为 $w=\{2,2,50,30,20,40,55,15,20\}$ Mcycles. 由 20 个任务组成的移动应用，其输入数据为 $\alpha=\{10,4,2,3,20,10,2,1,5,40,10,4,2,3,20,10,2,1,5,40\}$ Kbit，输出数据为 $\beta=\{4,2,3,3,20,10,2,15,40,30,4,2,3,3,20,10,2,15,40,30\}$ Kbit，计算负荷为 $w=\{2,2,50,30,20,40,55,15,20,2,2,50,30,20,40,55,15,20\}$ Mcycles. 仿真结果如表 1 所示，从表中可以发现：(1) OM-GA 算法的运算时间远低于 SGA 算法，这是因为 OM-GA 算法在 SGA 算法基础上采用一次迁移最优特性来设计交叉和变异操作，提高了算法的收敛速度；(2) OM-GA 算法的运算时间要明显低于 LARAC 算法，这是因为 LARAC 算法是采用动态规划来计算最短路径，需要在每个任务执行前进行一次执行位置判定，而 OM-GA 算法只需在第一个任务执行前计算最优迁移策略；(3) 随着任务数的增加，3 种算法的运算时间都有所增加，但是 OM-GA 算法的运算时间仅有微量增加，而 SGA 算法和 LARAC 算法的增加幅度都比较大，这是因为 OM-GA 算法利用一次迁移特性来进行设计，只需计算最佳迁移任务和最佳返回任务便可完成最优迁移策略的计算，这也验证了任务数 n 对 OM-GA 算法运算时间的影响不大；(4) 当任务数 n 从 10 增加到 20 时，SGA 算法的运算时间时间增加幅度高于 LARAC 算法，这主要是 LARAC 算法采用拉格朗日优化可减少了迭代次数.

表 1　算法的运算时间比较

任务数 n	OM-GA 算法/s	SGA 算法/s	LARAC 算法/s
10	1.311075	3.433471	3.200318
20	1.718365	7.815627	6.081186

第 3 个实验验证 OM-GA 算法的一次迁移特性，其参数设置与第 1 个实验相同. 图 4 给出了 OM-GA 算法下不同任务的执行位置. 图中分别采用内嵌斜杠、白和黑色的柱状图描述每个任务的输入数据、计算负荷和输出数据，采用红线表示任务的

执行位置，上和下两个位置分别表示任务在云端执行和移动终端执行.从图中可以发现：(1)任务1和2在移动终端执行，任务3迁移至云端执行后任务4到8均在云端执行，任务9返回至移动终端上执行后任务10也在移动终端上执行，验证了采用OM-GA算法获得的最优迁移策略属于一次迁移迁移策略；(2)迁移至云端执行的任务3，其计算负荷较高、输入和输出数据较少，而返回至云端执行的任务9，其计算负荷较低、输入和输出数据较多，验证了定理2和3的正确性，这也验证了本文TCEP策略的设计原则是将计算密集型任务放在云端执行；(3)任务6的计算负荷相对较低、输入和输出数据相对较多，但是任务6仍安置在云端执行，其原因是任务5和任务7都在云端执行，将任务6放在云端执行不仅可以减少传输能耗而且还能减少移动终端的执行能耗，这也验证了OM-GA算法的输出策略具有一次迁移特性.

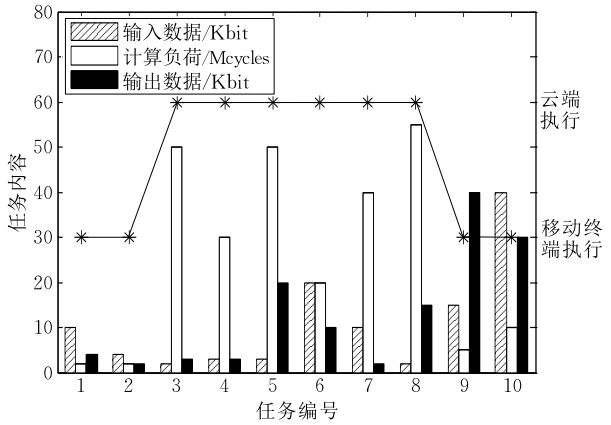


图4 任务的执行位置

第4个实验将本文TCEP策略与本地执行策略、云端执行策略、LARAC策略以及阈值判决策略进行比较.比较5种策略的能耗与$\mathbb{E}[R]$的关系，假设移动应用由10个任务组成，$\alpha=\{10,4,2,3,20,10,2,1,5,40\}$Kbit，$\beta=\{4,2,3,3,20,10,2,15,40,30\}$Kbit，$w=\{2,2,50,30,20,40,55,15,20\}$Mcycles.其他参数设置如下：$P_s=0.1$ W，$P_r=0.05$ W，$P_m=0.5$ W，$P_i=0.001$ W，$f_m=500$ MHz，$f_c=5000$ MHz，$T_d=4.0$ s.仿真结果如图5所示，从图中可以发现：(1)随着传输速率的增加，本地执行策略的能耗固定不变，这主要是本地执行策略中所有任务都在移动终端上执行，没有数据在信道中传输，不会产生传输能耗，因此信道传输速率不会对能耗造成影响；(2)除本地执行策略外其他4种策略的能耗随$\mathbb{E}[R]$的增加而减少，这是移动终端的发送功率和接受功率一定，$\mathbb{E}[R]$增加则任务迁移和返回的时间都会随之减少，故此导致传输能耗减少；(3)TCEP策略和LARAC策略的能耗要小于阈值判决策略，这是因为阈值判决策略是将应用进行整体迁移，而TCEP策略和LARAC策略都是采用任务联合执行，将计算负荷小、输入和输出数据大的任务放在移动终端上执行更有利于减少能耗；(4)TCEP策略的能耗略低于LARAC策略，这主要是因为TCEP策略采用了OM-GA算法，其计算最优迁移策略的时间要小于LARAC策略，使得移动终端在触发移动应用后的等待时间减少所造成的结果.

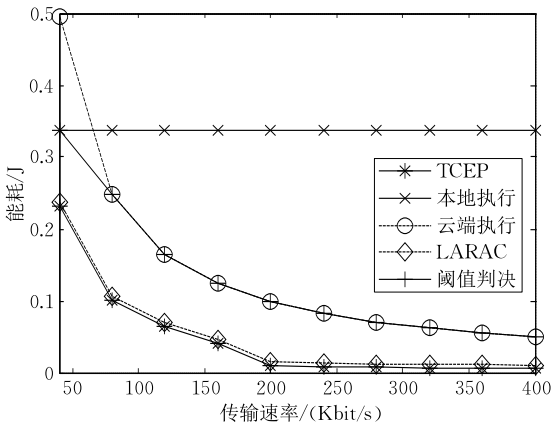


图5 能耗与$\mathbb{E}[R]$的关系

第5个实验比较5种策略的能耗与f_m的关系，假设$\mathbb{E}[R]=400$ Kbit/s，其他参数设置与第4个实验一致.仿真结果如图6所示，从图中可以发现：(1)随着f_m的增加，云端执行策略的能耗保持固定不变，这是因为云端执行策略的所有任务都在云端执行，移动终端不处理任何任务，其时钟频率f_m的变化不会响应云端任务处理业务的速度，因此云端执行策略的能耗不会随着f_m的增加而增加；(2)本地执行策略、阈值判决策略和TCEP策略的能耗随f_m的增加而减少，主要原因是这4种策略均有任务安置在移动终端上执行，f_m的增加会减少移动终端执行任务的时间，仿真中假设P_m一定，因此增加f_m可减少任务在移动终端上的执行能耗；(3)当$f_m>800$ MHz时，本地执行策略的能耗与阈值判决策略一样且低于云端执行策略，这主要是因为移动终端处理任务的能耗会随f_m的增加而减少，f_m增加到某一值后，阈值判决策略会选择将所有任务放在移动终端上执行，因此本地执行策略与阈值判决策略具有相同的能耗，云端执行策略的主要能耗是传输能

耗且不受 f_m 变化的影响，而本地执行策略没有传输能耗且计算能耗随 f_m 的增加而减少，当 f_m 增加到足够大时，必然会造成本地执行策略的能耗会低于云端执行策略的现象；(4) 当 f_m 较低时，TCEP 策略的能耗与云端执行策略和阈值判决策略相同，这是因为 TCEP 策略和阈值判决策略会在 f_m 较低时将整个应用迁移至云端执行所造成的结果；(5) 整体上，TCEP 策略的能耗低于其他 4 种策略，这可以解释为 TCEP 策略能在较短的运算时间内根据信道的传输速率、任务的计算负荷、输入和输出数据量来确定每个任务的执行位置，因此能更好的减少移动终端的能耗；(6) TCEP 策略的能耗与 LARAC 策略几乎相同，这是因为两种策略都是采用云端和移动终端联合执行移动应用的方式，且输出策略相同所造成的结果，但是 TCEP 策略的运算时间要优于 LARAC 策略.

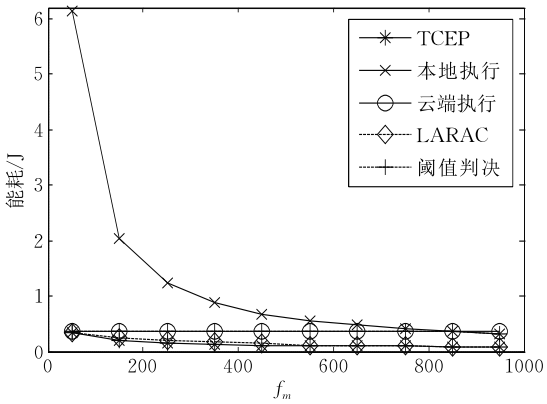


图 6　能耗与 f_m 的关系

8　结束语

本文对移动云计算中的应用迁移问题进行了研究，提出了一种 TCEP 策略，该策略能充分利用云端和移动终端的计算资源来联合执行移动应用，并提出了一种 OM-GA 算法来搜索全局最优解. 仿真结果表明，本文 OM-GA 算法能在较短的时间内获得最优解，与现有的策略相比，本文 TCEP 策略能有效地减少移动终端的能耗.

文中所考虑的任务模型具有一定的合理性和有效性，在某种程度上可以反映部分移动应用的特点. 然而，在实际应用中，还存在树形结构和网状结构的任务模型，任务在执行过程中可能还需要彼此间进行信息交互，即一般性的情况. 后续研究工作的重点是针对这种一般性的任务模型设计联合执行策略，以提高策略的实用性.

参　考　文　献

[1] Yang Z, Liu X, Hu Z, et al. Seamless service handoff based on delaunay triangulation for mobile cloud computing. Computers and Mathematics with Applications Wireless Personal Communications, 2013, 8770(6): 1-6

[2] Liu F, Shu P, Jin H, et al. Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications. IEEE Wireless Communications, 2013, 20(3): 1334-1354

[3] Ge C, Sun Z, Wang N. A survey of power-saving techniques on data centers and content delivery networks. IEEE Communications Surveys & Tutorials, 2013, 15(3): 1334-1354

[4] Xiang X D, Lin C, Chen X. Energy-efficient link selection and transmission scheduling in mobile cloud computing. IEEE Wireless Communications Letters, 2014, 3(2): 153-156

[5] Niyato D, Wang P, Hossain E, et al. Game theoretic modeling of cooperation among service providers in mobile cloud computing environments//Proceedings of the IEEE Wireless Communications and Networking Conference: Services, Applications, and Business. Shanghai, China, 2012: 3128-3133

[6] Kaewpuang R, Niyato D, Wang P, et al. A framework for cooperative resource management in mobile cloud computing. IEEE Journal on Selected Areas in Communications, 2013, 31(12): 2685-2700

[7] Kumar K, Lu Y H. Cloud computing for mobile users: Can offloading computation save energy. Computer, 2010, 43(4): 51-56

[8] Zhang W, Wen Y, Wu D. Collaborative task execution in mobile cloud computing under stochastic wireless channel. IEEE Transactions on Wireless Communications, 2015, 13(1): 1536-1276

[9] Zhang W, Wen Y, Wu D. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing// Proceedings of the IEEE INFOCOM. Turin, Italy, 2013: 190-194

[10] Ge Y, Zhang Y, Qiu Q, et al. A game theoretic resource allocation for overall energy minimization in mobile cloud computing system//Proceedings of the IEEE International Symposium on Low Power Electronics and Design. California, USA, 2012: 279-284

[11] Chen X. Decentralized computation offloading game for mobile cloud computing. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(4): 974-983

[12] Gao B, He L, Lu X, et al. Developing energy-aware task allocation schemes in cloud-assisted mobile workflows// Proceedings of the IEEE International Conference on Computer and Information Technology. California, USA, 2015: 1-6

[13] Magurawalage C M S, Yang K, Hu L, et al. Energy-efficient and network-aware offloading algorithm for mobile cloud computing. Computer Network, 2014, 74(9): 22-33

[14] Huang S H, Shih C S, Shieh J P, et al. Executing mobile applications on the cloud: Framework and issues. Computers and Mathematics with Applications, 2012, 63: 573-587

[15] Liu Xing, Yuan Chao-Wei, Yang Zhen, et al. A scheme of pricing and virtual machine allocation in mobile cloud computing. Journal of University of Electronic Science and Technology of China, 2016, 45(2): 197-201(in Chinese)
(柳兴, 袁超伟, 杨震等. 移动云计算中一种虚拟机定价与分配方案. 电子科技大学学报, 2016, 45(2): 197-201)

[16] Folino G, Pisani F S. Automatic offloading of mobile applications into the cloud by means of genetic programming. Applied Soft Computing, 2014, 25: 253-265

[17] Poulakis M I, Panagopoulos A D, Constantinou P. Channel-aware opportunistic transmission scheduling for energy-efficient wireless links. IEEE Transactions on Vehicular Technology, 2013, 62(1): 192-204

[18] Wen Y, Zhang W, Luo H. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones//Proceedings of the IEEE International Conference on Computer Communications. Orlando, USA, 2012: 2716-2720

[19] Zhang W, Wen Y, Guan K, et al. Energy-optimal mobile cloud computing under stochastic wireless channel. IEEE Transactions on Wireless Communications, 2013, 12(1): 4569-4581

[20] Chandra R, Bahl P. MAUI: Making smart phones last longer with code offload//Proceedings of the International Conference on Mobile Systems, Applications, and Services. New York, USA, 2010: 49-62

[21] Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing. IEEE Transactions on Wireless Communications, 2012, 11(6): 1991-1995

[22] She P, Liu F, Jin H, et al. eTime: Energy-efficient transmission between cloud and mobile devices//Proceedings of the IEEE INFOCOM. Turin, Italy, 2013: 195-199

[23] Ra M, Paek J, Aharma A, et al. Energy-delay tradeoffs in smartphone application//Proceedings of the 2010 ACM MobiSys. California, USA, 2010: 2555-270

[24] Xiang X, Lin C, Chen X, et al. Energy-efficient link selection and transmission scheduling in mobile cloud computing. IEEE Wireless Communications Letters, 2014, 3(2): 153-156

[25] Yu H, Cheung M H, Huang L, et al. Delay-aware predictive network selection in data offloading//Proceedings of the IEEE INFOCOM. Toronto, Canada, 2014: 173-174

[26] Wang X, Chen M. PreFeed: Cloud-based content prefetching of feed subscriptions for mobile users. IEEE Systems Journal, 2014, 8(1): 201-207

[27] Yang S, Kwon D, Yi H, et al. Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing. IEEE Transactions on Mobile Computing, 2014, 13(11): 2648-2660

[28] Stoitsev T, Scheidl S, Flentge F, et al. Enabling end-user driven business process composition through programming by example in a collaborative task management system//Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing. New York, USA, 2008: 157-165

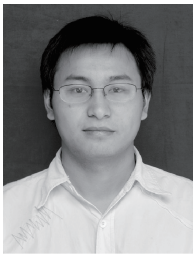
[29] Sebastio S, Amoretti M, Luch L A. A computational field framework for collaborative task execution in volunteer clouds//Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Hyderabad, India, 2014: 105-114

[30] Liu X, Yuan C W, Li Y, et al. A lightweight algorithm for collaborative task execution in mobile cloud computing. Wireless Personal Communications, 2015, 86(2): 579-599

[31] Liu X, Yuan C, Yang Z, et al. An energy saving algorithm based on user-provided resources in mobile cloud computing //Proceedings of the VTC'13-Fall. Las Vegas, USA, 2013: 1-5

[32] Stephen C. The complexity of theorem proving procedures//Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. Shaker Heights, USA, 1971: 151-158

[33] Ye Miao, Wang Yu-Ping, Wei Jing-Xuan. Coverage repair strategies for wireless sensor networks based on muti-mobile nodes and genetic algorithm. Journal on Communications, 2014, 35(12): 45-52(in Chinese)
(叶苗, 王宇平, 魏静萱. 基于多移动节点和遗传算法的传感器网络覆盖修复策略. 通信学报, 2014, 35(12): 45-52)



LIU Xing, born in 1984, Ph. D. His research interests include cloud computing, mobile cloud computing and mobile internet.

LI Jian-Bin, born in 1968, professor. His research interests include smart healthcare, big data applications and information safety.

YANG Zhen, born in 1975, Ph. D., associate professor. His research interests include cloud computing and mobile multimedia.

LI Zhen-Jun, born in 1975, associate professor. His research interests include cloud computing and mobile education.

Background

Mobile Cloud Computing (MCC) has emerged as one of the most promising techniques in mobile services. MCC can improve the performance of mobile applications, which are offered by the mobile cloud service providers, by offloading data processing from mobile devices (MDs) to remote cloud. As a result, MDs do not need a powerful configuration since all the complicated computing can be processed in the cloud. Moreover, the calculation energy of MDs can be significantly reduced. However, after MD implemented offloading, the data of mobile application, which is transmitted on wireless networks, is increasing rapidly since user' mobile applications have to get support from the remote cloud. Moreover, the energy consumption for transmitting data in a bad wireless channel is more than that in a good wireless channel. As a result, energy saving from application offloading is not guaranteed on the MD via a stochastic wireless channel. So far, the limited battery life has been found by market research as the biggest complaint for MDs. Thus, designing a good offloading policy is very important for MCC.

In this paper, we aim at the problem of the energy consumption on MD by migrated mobile application to cloud. Firstly, mobile applications are considered as a series of tasks which can be executed by cloud or MT, and we utilize the calculated load, the amount of input and output data of the task to obtain the optimal feature of application migration. Then, using this migration property, we designed the cross-over operation and the mutation operation of genetic algorithm.

This paper is supported by the project of the National High Technology Research and Development Program (863 Program) of China under Grant No. 2014AA01A701, the National Natural Science Foundation of China under Grant No. 61173017 and No. 61563038, the Program for Innovative Research Team in Universities of Inner Mongolia Autonomous Region No. NMGIRT-A1609 and the Educational Commission Fund of Hunan Province No. 13C613. These projects aim to energy optimization in cloud computing. The authors have been working on mobile cloud computing and mobile application offloading. Some methods have been produced to analyze offloading threshold value. Many papers have been published in international conferences and journals.