一种面向移动云计算的多目标任务卸载算法

宋富洪，邢焕来，潘炜
（西南交通大学信息科学与技术学院，四川 成都 611756）

摘　要：计算能力和资源受限的移动设备可将待处理的密集型任务卸载到云端执行，从而增强移动设备的计算能力并减少电池能源消耗（EC）。然而，现有研究在卸载任务时不能较好地均衡移动端的应用完成时间（FT）和 EC。提出了基于分解的多目标进化算法（MOEA/D）来同时优化应用 FT 和 EC，并将动态电压频率调整技术引入 MOEA/D 中，在不增加应用 FT 的前提下，调节移动设备的 CPU 时钟频率以进一步降低移动设备的 EC。仿真结果表明，与多个算法相比，所提出的算法在多目标性能上更优。

关键词：移动云计算；移动设备；多目标进化算法；任务卸载；完成时间；能源消耗
中图分类号：TN929.5
文献标识码：A
doi: 10.11959/j.issn.2096−3750.2019.00118

Multi-objective task offloading algorithm for mobile cloud computing

SONG Fuhong, XING Huanlai, PAN Wei
School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China

Abstract: Mobile devices with limited computing power and resources can offload intensive tasks to the cloud for execution, thus improving the computing capacity of mobile devices and reducing battery energy consumption. However, the existing researches cannot properly balance the application finish time and energy consumption of the mobile terminal when offloading tasks. An MOEA/D based algorithm was proposed to optimize the application finish time and energy consumption, and dynamic voltage frequency scaling technology was introduced into the MOEA/D to adjust the CPU clock frequency of mobile devices to further decrease the energy consumption without increasing the application finish time. The simulation results demonstrate that the proposed algorithm outperforms a number of existing algorithm in terms of the multi-objective performance.

Key words: mobile cloud computing, mobile device, multi-objective evolutionary algorithm, task offloading, finish time, energy consumption

1　引言

当今，随着移动互联网技术的迅猛发展，诸如平板电脑、智能手机和便携式穿戴设备等智能移动设备已经成为用户的主要计算平台[1]。虽然智能终端和无线通信技术正在日益演进，使得移动设备的计算性能大幅度提升，但由于移动设备受电池容量、计算和存储能力等方面的限制，仍然无法满足计算密集型任务的需求。移动设备较低的计算能力可能会延迟应用的完成时间（FT，finish time），从而很难保证服务质量（QoS，quality of service）。同时，密集型任务会加速消耗移动设备有限的电池能量[2-3]。

无线网络和云计算的不断融合促使了移动云计算（MCC，mobile cloud computing）[4]的产生和发展，MCC 用户可将应用程序的部分任务卸载到

收稿日期：2019−07−15；**修回日期：**2019−08−05
基金项目：国家自然科学基金资助项目（No.61401374）；中央高校基本科研业务费创新基金资助项目（No.2682017CX099）
Foundation Items: The National Natural Science Foundation of China (No.61401374), Innovation Fund Project of Basic Scientific Research Operating Expenses of Central Universities (No.2682017CX099)

具有丰富计算资源的远端云上执行，云端再将任务的执行结果返回到移动设备。一般来说，MCC 的大型云端服务器距离移动用户几十千米甚至更远，用户卸载任务时会产生大量的传输时延，因此，在 MCC 环境下的任务卸载适用于时延容忍的大规模计算密集型应用，如在线社交网络、移动电子商务、移动医疗和移动在线学习等。相反地，移动边缘计算（MEC，mobile edge computing）[5]是将轻量级服务器部署在网络边缘，一般距离移动用户 1 km 以内，传输时延较短，因此，MEC 适用于时延敏感的小规模计算密集型应用，如虚拟现实、自动驾驶和交互式在线游戏等。当前，随着信息技术的快速发展和移动用户应用的极大丰富，移动端应用程序越来越复杂，规模也相继增大。本文旨在研究时延容忍的计算密集型应用的任务卸载，即在 MCC 环境下的任务卸载问题。

移动设备利用 MCC 技术获取云端服务器的丰富资源和强大的计算能力，从而扩展自身受限的资源以提升移动设备的计算能力。移动设备卸载任务到云端执行，能提高任务的执行效率并大幅度降低移动设备的能源消耗（EC，energy consumption）[6]。但是不合理的任务卸载策略会产生大量数据传输，不仅会增加执行应用的时延，还会大量消耗移动设备的能量。此外，随着移动设备应用任务数的增加，已有的卸载策略不能较好地权衡移动设备的任务 FT 和 EC。因此，如何合理地在移动设备和云端之间卸载任务，在减少应用 FT 的同时，尽可能降低移动设备的 EC，是当今 MCC 领域的重要问题，也称为任务卸载问题。

鉴于此，本文主要研究 MCC 环境下移动设备的应用 FT 和 EC 的多目标任务卸载优化问题，提出了基于分解的多目标进化算法（MOEA/D，multi-objective evolutionary algorithm based on decomposition），该算法是分解一个多目标优化问题（MOP，multi-objective optimization problem）以同时优化多个标量的子问题。此外，将移动设备的动态电压频率调整（DVFS，dynamic voltage frequency scaling）技术引入 MOEA/D 中，采用遗传算子生成一个新解后，对该解实施动态电压调节机制，保证在不增加 FT 的情况下，进一步降低移动设备的 EC。仿真结果表明，本文提出的算法具有可行性和优越性，在多目标性能上优于 NSGA-Ⅱ算法、DVFS 算法以及原始的 MOEA/D 算法。

2 研究现状

移动设备将资源密集型和计算密集型的应用程序卸载到云端执行，以弥补自身计算资源不足的问题。如何合理地卸载任务使 FT 降到最低，同时减小移动设备的 EC，是 MCC 中的研究热点。目前，国内外学者对该问题进行了大量研究，可分为 3 类：1) 最小化应用的总 FT；2) 最小化移动设备的 EC；3) 同时优化移动设备的应用 FT 和 EC。

当资源受限的移动设备利用 MCC 技术卸载密集型应用程序到云端执行时，任务的 FT 是衡量 MCC 环境下 QoS 的一个重要指标。Cai 等[7]提出了一种需求驱动的任务调度模型，并引入估计方法来预测任务的 FT。该文献从移动用户和服务供应商两个方面来研究任务卸载问题，通过一种基于 2D 染色体遗传算法（GA，genetic algorithm）的目标权重方法来最小化移动用户的 FT 并均衡服务供应商的负载。Yang 等[8]研究了多用户的计算分区和云资源计算卸载的调度问题，与优化单用户的应用 FT 不同，该文献旨在最小化多个用户的平均 FT。Balamurugan 等[9]提出了一种处理器选择算法来有效地卸载应用任务到异构处理器上，该算法通过任务的优先级对移动设备上的应用程序实现高效、快速调度。

另一个衡量 MCC 环境 QoS 的重要指标是移动设备的 EC。Kumar 等[10]根据待卸载的任务量和网络性能分析了任务的卸载，认为卸载任务到云端并不总会节省移动设备的 EC，进一步分析可知，卸载任务是否能节省 EC 取决于卸载的任务量是否超过无线通信的开销。Tong 等[11]旨在权衡移动设备的 EC 和应用 QoS，并提出了基于应用感知的无线传输调度算法，在满足应用截止时间的前提下，最小化移动设备无线传输的 EC。Mahmoodi 等[12]建模任务卸载为一个线性优化问题，从而得到更具适应性和可扩展性的模型。与其他只考虑任务先后次序的方法相比，该方法能得到更优的方案。Lin 等[13]研究了在满足应用截止时间的基础上，尽可能最小化移动设备的 EC。首先计算任务优先级以确定任务的执行顺序，再根据此顺序得到具有最小 FT 的初始调度，最后通过启发式算法和 DVFS 技术来进一步降低移动设备的 EC。但该方法只考虑在满足应用截止时间的基础上降低 EC，而未对应用 FT 进行优化。

一般来说，应用 FT 和移动设备的 EC 之间是相互关联的，减少 FT 势必会导致移动设备的 EC 过多，反之亦然，因此，一些研究致力于联合优化这两者。Deng 等[14]提出了基于 GA 的任务卸载策略来同时优化工作流的 FT 和移动设备的 EC，该策略还设计了设备的移动性管理和卸载系统的容错管理。Guo 等[15]提出了一种能效动态卸载和资源调度算法，在任务截止时间的硬性条件下，降低卸载过程的 EC。Zhou 等[16]研究了 MCC 环境下基于时延传输的多目标工作流调度，并提出了基于 GA 的算法，在 GA 的编码中同时考虑了任务的执行位置和执行顺序。通过延长非关键任务的执行时间，以进一步降低移动设备的 EC，该算法虽然能获得大量非支配解以供决策者按需选择，但随着应用任务数的增加，所得解的质量越来越低。

3　模型及问题描述

本节首先介绍 MCC 环境下移动设备的应用表示，包括本地计算模型和云计算模型，然后详细阐述本文的多目标任务卸载问题[13]。

移动设备执行的应用可表示为一个有向无环图（DAG，directed acyclic graph），记为 $\boldsymbol{G}=(V,E)$，节点 $v_i \in V$ 表示一个任务；边 $e(v_i,v_j)\in E$ 表示任务 v_i 和任务 v_j 之间的先序限制关系，即任务 v_i 必须在任务 v_j 开始执行之前完成执行。用 pre(v_i)表示节点 v_i 的前驱节点集，suc(v_i)表示节点 v_i 的后继节点集。$N=|V|$表示应用的总任务数，一般不超过 100 个。无任何前驱节点的任务节点为开始任务，记为 v_{start}，无任何后继节点的任务节点为结束任务，记为 v_{end}。

3.1　本地执行模型

假定移动设备有 K 个异构处理器，每个处理器都可使用 DVFS 技术。本文用一个二元组 $(f_k^{\max}, p_l^{\text{txd}})$，$k\in\{1,\cdots,K\}$来表示移动设备，其中，$f_k^{\max}$ 是移动设备第 k 个处理器的最大运行频率，且支持 H 种不同的电压（i.e., $\alpha_{k,1}<\cdots<\alpha_{k,H}=1$），则任务 v_i 运行的实际频率 $f_k(v_i)$ 可由电压 $\alpha_k(v_i)$ 控制，即 $f_k(v_i)=\alpha_k(v_i)\cdot f_k^{\max}$；$p_l^{\text{txd}}$ 表示卸载任务时的传输功率。由于任务的输出数据量较小，因此，忽略移动设备从云端接收数据的 EC。

设 $T_l^{k,\min}(v_i)$ 表示移动设备使用第 k 个处理器的最大运行频率执行任务 v_i 时所用的时间，其中，l 表示在本地执行。所以，任务 v_i 在第 k 个处理器上的实际执行时间 $T_l^k(v_i)$ 可通过式(1)得到

$$T_l^k(v_i)=T_l^{k,\min}(v_i)/\alpha_k(v_i) \tag{1}$$

用 $T_c(v_i)$、$T_{\text{txd}}(v_i)$和 $T_{\text{rxd}}(v_i)$分别表示任务 v_i 在云端的执行时间、移动设备发送任务 v_i 所需时间和移动设备从云端接收任务 v_i 的输出数据的所用时间。

如果任务 v_i 在移动设备的第 k 个处理器上执行，则移动设备执行任务 v_i 的 EC 为

$$E_l^k(v_i)=(\alpha_k(v_i))^{\gamma_k-1}\cdot E_l^{k,\max}(v_i) \tag{2}$$

其中，$\gamma_k\in[2,3]$，$E_l^{k,\max}(v_i)$ 是任务 v_i 在第 k 个处理器上以最大运行频率执行时的 EC。

任务的执行必须满足任务之间的先序限制关系，任务开始执行前，其所有前驱任务已被卸载完成。若任务 v_i 在本地处理器上执行，则执行的准备时间为

$$RT_l^e(v_i)=\max_{v_j\in\text{pre}(v_i)}\max\left\{FT_l^e(v_j),FT_c^{\text{rxd}}(v_j)\right\} \tag{3}$$

其中，$FT_l^e(v_j)$ 为任务 v_j 在本地执行的 FT，$FT_c^{\text{rxd}}(v_j)$ 为移动设备从云端接收任务 v_j 的输出数据的 FT。若任务 v_j 在本地执行，则 $FT_c^{\text{rxd}}(v_j)=0$，反之，$FT_l^e(v_j)=0$。显然，任务 v_i 的执行需在准备时间 $RT_l^e(v_i)$ 后开始，本地处理器在当前时刻有可能正在执行其他任务，因此，任务 v_i 的开始执行时间 $ST_l^e(v_i)$ 有 $ST_l^e(v_i)\geqslant RT_l^e(v_i)$。

3.2　云计算模型

若将任务 v_i 卸载到云端执行，则在无线发送信道上的准备发送时间为

$$RT_c^{\text{txd}}(v_i)=\max_{v_j\in\text{pre}(v_i)}\max\left\{FT_l^e(v_j),FT_c^{\text{txd}}(v_j)\right\} \tag{4}$$

进一步可得移动设备发送数据的 EC 为

$$E_c^{\text{txd}}(v_i)=p_l^{\text{txd}}\cdot T_{\text{txd}}(v_i) \tag{5}$$

$$T_{\text{txd}}(v_i)=FT_c^{\text{txd}}(v_i)-ST_c^{\text{txd}}(v_i) \tag{6}$$

其中，$ST_c^{\text{txd}}(v_i)$ 和 $FT_c^{\text{txd}}(v_i)$ 分别为移动设备在无线发送信道上发送任务 v_i 的开始时间和结束时间。

任务 v_i 在云端服务器上执行的准备时间为

$$RT_c^e(v_i)=\max\{FT_c^{\text{txd}}(v_i),\max_{v_j\in\text{pre}(v_i)}FT_c^e(v_j)\} \tag{7}$$

如果任务 v_i 的前驱任务 v_j 已在本地执行完成，则 $FT_c^e(v_j)=0$。

任务 v_i 在云端服务器上执行完成后，需将执行

结果返回移动设备，则传输返回结果的准备时间为

$$RT_c^{\mathrm{rxd}}(v_i)=FT_c^e(v_i) \tag{8}$$

式(8)表明，任务 v_i 在云端执行完毕后，可立即将 v_i 的输出数据发送给移动设备。

3.3 问题描述

本文在 MCC 环境下多目标任务卸载算法的主要目标是最小化移动设备的应用 FT 和 EC。对于给定的一个应用（有向无环图 $\boldsymbol{G}$），算法的目标是产生多个可行解 $\Omega=(\boldsymbol{L},\boldsymbol{O})$，其中，$\boldsymbol{L}=(\mathrm{loc}_1,\cdots,\mathrm{loc}_i,\cdots,\mathrm{loc}_N)$ 表示应用的任务执行位置向量，其中，$\mathrm{loc}_i\in\{1,\cdots,K,K+1\}$。若 $1\leqslant\mathrm{loc}_i\leqslant K$，则表示任务 v_i 在移动设备的某个处理器上执行，$\mathrm{loc}_i=K+1$ 表示任务 v_i 在云端执行；$\boldsymbol{O}=(\mathrm{ord}_1,\cdots,\mathrm{ord}_i,\cdots,\mathrm{ord}_N)$ 表示应用的任务执行顺序向量，其中，$\mathrm{ord}_1=v_{\mathrm{start}}$，$\mathrm{ord}_N=v_{\mathrm{end}}$，$\mathrm{ord}_i\in\{v_2,...,v_{N-1}\}$。任务的执行必须满足任务间的先序限制关系，即任务的执行必须在所有前驱任务完成执行后才能开始执行。已知 $\boldsymbol{L}$ 和 $\boldsymbol{O}$ 后，就可以确定任务在本地核执行还是在云端执行以及执行顺序，因此，一个可行解 Ω 就是应用的一个卸载策略。

为了计算应用 FT，需计算每个任务的开始时间和结束时间，因此，移动设备的应用 FT 为结束任务 v_{end} 的 FT，即

$$FT(\boldsymbol{G})=\begin{cases}FT_l^e(v_{\mathrm{end}}), & 1\leqslant\mathrm{loc}_{\mathrm{end}}\leqslant K\\ FT_c^{\mathrm{rxd}}(v_{\mathrm{end}}), & \mathrm{loc}_{\mathrm{end}}=K+1\end{cases} \tag{9}$$

其中，$\mathrm{loc}_{\mathrm{end}}$ 为结束任务 v_{end} 的执行位置。

若任务在本地执行，则移动设备的 EC 为在本地核上执行的 EC；若任务在云端执行，则移动设备的 EC 为发送数据的 EC。因此，在整个应用的执行过程中，移动设备的 EC 为

$$EC(\boldsymbol{G})=\sum_{\mathrm{v}_i\in V,1\leqslant\mathrm{loc}_i\leqslant K}E_l^k(v_i)+\sum_{\mathrm{v}_i\in V,\mathrm{loc}_i=K+1}E_{\mathrm{c}}^{\mathrm{txd}}(v_i) \tag{10}$$

基于以上两个方面的计算，本文在 MCC 环境下的多目标任务卸载问题可描述为：找到一种或多种卸载策略，在该策略下，能够最小化移动设备的应用 FT 和 EC，即

$$\min\ (FT(\boldsymbol{G}),EC(\boldsymbol{G})) \tag{11}$$

4 算法设计

MOP 可通过式(12)来定义[19]

$$\begin{cases}\min F(\boldsymbol{x})=(f_1(\boldsymbol{x}),\cdots,f_m(\boldsymbol{x}))\\ \text{s.t.}\quad g_i(\boldsymbol{x})\leqslant 0,\ i=1,\cdots,p\\ \qquad h_j(\boldsymbol{x})=0,\ j=1,\cdots,q\end{cases} \tag{12}$$

式(12)表明，MOP 有 m 个优化目标，且目标之间一般是相互冲突的，其中，$\boldsymbol{x}\in X\subset R^n$ 是 n 维决策向量，$g_i(\boldsymbol{x})$ 和 $h_j(\boldsymbol{x})$ 表示该优化问题的两个约束条件。

定义 1 Pareto 占优

设 $x_1,x_2\in X$，当且仅当满足式(13)时，称 x_1 Pareto 占优 x_2。

$$\begin{aligned}&\forall i=1,\cdots,m,f_i(x_1)\leqslant f_i(x_2)\\ &\wedge\exists j\in\{1,\cdots,m\},f_j(x_1)<f_j(x_2)\end{aligned} \tag{13}$$

也可称为解 x_1 支配解 x_2，记为 $x_1\prec x_2$。

定义 2 Pareto 最优解

对于式(12)描述的 MOP，若 $x^*\in X$ 满足式(14)，则称 x^* 为 Pareto 最优解。

$$\neg\exists x\in X:x\prec x^* \tag{14}$$

定义 3 Pareto 最优解集

对于 MOP，所有 Pareto 最优解组成的集合为 Pareto 最优解集，记为 PS

$$PS=\{x^*\mid\neg\exists x\in X:x\prec x^*\} \tag{15}$$

定义 4 Pareto 前沿

将定义 3 中的 PS 对应的目标函数在空间上形成的曲线称为 Pareto 前沿。

在 MCC 环境下，移动设备可将密集型的任务卸载到云端执行，以降低设备 EC，但需将任务执行的相关数据传输到云端，这势必会增加传输时延，延迟了应用 FT。因此，本文优化的移动设备应用 $FT(\boldsymbol{G})$和 $EC(\boldsymbol{G})$是两个相互冲突的优化目标，不存在同时使两个目标都达到最优的解，取而代之的是找到优化问题的 Pareto 最优解集。

目前，对于 MOP 的求解主要采用多目标进化算法（MOEA，multi-objective evolutionary algorithm）[20]，MOEA 可得到近似最优的 Pareto 解集，其中，每个解表示 MOP 中多个目标之间的权衡。MOEA/D[17]是求解 MOP 的一个全新框架，该框架将 MOP 分解为多个标量形式的单目标优化子问题，再对多个子问题同时进行优化。此外，MOEA/D 与单目标 GA 相融合，利用遗传操作产生新个体，通过多次迭代进化，获得 MOP 最优前沿。理论表明，MOEA/D 比 NSGA-Ⅱ[18]具有更低的计算复杂度，

且收敛速度更快，所以一提出就受到了学术界的广泛关注，已成功应用于一系列复杂 MOP 中[21-24]。因此，本文采用 MOEA/D 求解 MCC 环境下的多目标任务卸载问题，同时优化 $FT(\boldsymbol{G})$和 $EC(\boldsymbol{G})$。

4.1　分解方法

MOEA/D 将 MOP 分解成多个标量形式的单目标优化子问题，再同时优化这些子问题，分解方式有多种，本文采用 Tchebycheff 方法进行 MOP 的分解[17]，其计算如式(16)所示。

$$\begin{cases}\min g^{te}(x\mid\boldsymbol{\lambda},z^*)=\max\limits_{1\leqslant i\leqslant m}\{\lambda_i\mid f_i(x)-z_i^*\mid\}\\ \text{s.t.}\quad x\in X\subset R^n\end{cases}\tag{16}$$

其中，m 为优化目标数；$\boldsymbol{\lambda}=(\lambda_1,\cdots,\lambda_m)$ 为权重向量，且 $\sum\limits_{i=1}^{m}\lambda_i=1$；$z^*=(z_1^*,\cdots,z_m^*)$ 为参考点，$z_i^*=\min\{f_i(x)\mid x\in X\}$，$i$=1,⋯, m。

4.2　初始化

如 3.3 节所述，解 Ω=($\boldsymbol{L}$,$\boldsymbol{O}$)的编码由两部分组成，分别是任务的执行位置和执行顺序。对于执行位置来说，任务可在移动设备的任一处理器上执行或卸载到云端执行。本文随机生成每个任务的执行位置，具体产生方式即任务执行位置初始化伪代码，如算法 1 所示，其中，randInt(1, K+1)表示随机生成[1, K+1]之间的整数。

算法 1　任务执行位置初始化伪代码

1) for i=1 to N do //N 为应用任务数

2) loc_i =randInt(1,K+1)　//随机生成 v_i 的执行位置

3) end for

在任务执行顺序初始化方面，必须满足任务之间的先序限制关系，即任意边 $e(v_i,v_j)\in E$，任务 v_i 必须在任务 v_j 开始执行之前完成其执行，文献[16]给出了任务执行顺序伪代码如算法 2 所示。

算法 2　任务执行顺序初始化伪代码

1) $S=\varnothing$　// 待排序任务集

2) $\boldsymbol{O}=\{v_{\text{start}}\}$ //已排序任务集

3) $T=T-\{v_{\text{start}}\}$

4) while $T\neq\varnothing$　do

5) for i=1 to N do

6) if $\text{pre}(v_i)\subset\boldsymbol{O}$ and $v_i\notin S$ then

7) $S=S+\{v_i\}$

8) end if

9) end for

10) 从 S 中随机选择一个任务 v_s

11) $S=S-\{v_s\}$

12) $T=T-\{v_s\}$

13) $\boldsymbol{O}=\boldsymbol{O}+\{v_s\}$ //在 $\boldsymbol{O}$ 的末尾添加 v_s

14) end while

设种群规模为 P，则需要循环执行 P 次算法 1 和算法 2 的伪代码，实现任务执行位置和执行顺序的初始化，下面给出基于 MOEA/D 的多目标任务卸载算法的初始化过程。

Step1　设置 $EP=\varnothing$。

Step2　初始化 P 个权重向量 $\lambda^1,...,\lambda^P$，对每个 λ^i 计算 $B(i)=\{i_1,...,i_T\}$，其中，$\lambda^{i_1},....,\lambda^{i_T}$ 是与权重 λ^i 的 T 个最近邻居权重向量。

Step3　根据算法 1 和算法 2 的伪代码初始化 P 个个体的种群，记为 $x^1,....,x^P$。

Step4　通过 Step3 产生的种群初始化参考点 $z=(z_1,\cdots,z_m)$。

4.3　基于 MOEA/D 的多目标任务卸载算法

DVFS 能降低移动设备的高性能处理器的执行频率，使得高性能处理器与低性能处理器具有相同的执行性能，那么在低性能处理器上执行任务可降低移动设备的 EC，因此，本文将 DVFS 技术融入 MOEA/D 中，以进一步降低移动设备的 EC。首先给出 DVFS 算法的伪代码[13]，随后描述基于 MOEA/D 的多目标任务卸载算法过程。

算法 3　DVFS 算法伪代码

输入　调度策略

1) for i=1 to N do

2) if $1\leqslant\text{loc}_i\leqslant K$ then　//本地任务

3) flag=0；h=1

4) while flag==0 and $h<H$ do

5) 计算 v_i 用第 h 个频率时新的 FT 即 $FT_l^{e,n}(v_i)$

6) if 该核上 v_i 存在下一个任务 v_j then

7) $\text{limit}_1=ST_l^e(v_j)$

8) else if　v_i 是该核上的最后一个任务 then

9) $\text{limit}_1=FT(\boldsymbol{G})$

10) end if

11) if $v_i\neq v_{\text{end}}$ then

12) $\text{limit}_2=\min_{v_j\in\text{suc}(v_i)}ST_l^e(v_j)$

13) else

14) $\text{limit}_2=FT(\boldsymbol{G})$;

15) end if

16) if $FT_l^{e,n}(v_i) \leqslant \text{limit}_1$ and $FT_l^{e,n}(v_i) \leqslant \text{limit}_2$ then

17) flag=1

18) 分配第 h 个频率来执行 v_i

19) 更新 v_i 的 FT

20) end if

21) $h = h+1$

22) end while

23) end if

24) end for

输出 本地执行任务具有新的 FT 的调度策略

算法 3 中第 9 行和第 14 行表示执行 DVFS 算法后不增加整个应用的 FT，因此，在不增加 FT 的基础上，达到降低移动设备 EC 的目的。

下面描述在 MCC 环境下，基于 MOEA/D 的多目标任务卸载算法过程，记为 MOEA/D-DVFS。在 Step1 中，采用 4.2 节中的初始化过程对算法进行初始化，包括 P 个权重向量和 P 个个体的种群。Step3 中对 x^i 的两个邻居个体 x^k 和 x^l 进行遗传操作，产生新的个体，本文采用文献[16]中的遗传算子（交叉、变异）来进行遗传操作。在 Step4 中，对新个体 y 实施 DVFS 算法，产生新的策略 $\bar{y}$ 以进一步降低移动设备的 EC。

Step1 算法初始化 //参考第 4.2 节

重复

Step2 for i=1 to P do

Step3 Reproduction：随机从 $B(i)$中选择元素 k、l，且 $k \neq l$，对 x^k 和 x^l 执行遗传操作，产生新个体 y

Step4 DVFS 算法：对 y 执行 DVFS 算法，产生新的调度策略 $\bar{y}$

Step5 更新 z：设 j=1,…, m，如果 $f_j(\bar{y}) < z_j$，则 $z_j = f_j(\bar{y})$

Step6 更新邻居解：对 $j \in B(i)$，如果 $g^{te}(\bar{y} \mid \lambda^j, z) \leqslant g^{te}(x^j \mid \lambda^j, z)$，则 $x^j = \bar{y}$

Step7 更新 EP：删除 EP 中被 $F(\bar{y})$ 支配的向量，若 EP 中无向量支配 $F(\bar{y})$，则添加 $F(\bar{y})$ 到 EP 中

终止 若满足终止条件，则算法结束并输出 EP；否则返回 Step2

5 仿真实验

本节首先描述 4 种算法的性能评价指标，然后给出实验场景及参数设置，最后 MOEA/D-DVFS 与 NSGA-Ⅱ、DVFS 以及原始的 MOEA/D 进行完整的算法性能比较。

5.1 性能评价指标

为了综合评估 MOEA/D-DVFS 的算法性能，使用 4 种算法性能度量指标。设 PF_{ref} 为较好接近于真实 Pareto 前沿的参考集，PF_{know} 为算法获得的最优解集。一般情况下，对于高复杂性的 MOP[24]（如本文的多目标任务卸载问题）并不知道真实的 Pareto 前沿，因此，将所有算法运行后得到的最优解组合起来，求出非支配解作为参考集 PF_{ref}。

1) 反向世代距离指标

反向世代距离（IGD，inverted generational distance）可衡量算法非支配解集的收敛性和多样性，IGD 值越小则表明算法整体性能越好，定义如下

$$\begin{cases} \text{IGD} = \dfrac{\sum\limits_{sr \in PF_{\text{ref}}} d(sr, PF_{\text{know}})}{|PF_{\text{ref}}|} \\ d(sr, PF_{\text{know}}) = \min\limits_{sk \in PF_{\text{know}}} \|sr - sk\| \end{cases} \tag{17}$$

其中，$d(sr, PF_{\text{know}})$ 是 sr 与 PF_{know} 中每个解的最小欧式距离。

2) 世代距离指标

世代距离（GD，generational distance）可判断算法获得的最优解集与真实 Pareto 前沿的收敛程度，定义如下

$$\begin{cases} GD = \sqrt{\dfrac{\sum\limits_{sk \in PF_{\text{know}}} d(sk, PF_{\text{ref}})}{|PF_{\text{know}}|}} \\ d(sk, PF_{\text{ref}}) = \min\limits_{sr \in PF_{\text{ref}}} \|sk - sr\| \end{cases} \tag{18}$$

3) 平均计算时间指标

将每种算法运行 50 次所消耗的平均计算时间（ACT，average computation time）作为衡量算法计算复杂度的指标。

4) Pareto 前沿曲线

由于多目标任务卸载问题的优化目标为应用 FT 和移动设备 EC，为典型的双目标优化问题。因此，可将算法得到的 Pareto 解集绘制在二维坐标系中显示出来，Pareto 曲线可以直观展示算法的 Pareto 解集的分布性和收敛性。

5.2 实验结果

采用随机生成应用（即 DAG）的方式来产生实

验场景[13]，模拟 10 种具有不同任务数的 DAG，任务数 N 分别为 10,20,…,100。

假定移动设备具有 K=3 个异构处理核，核 1 为低功率处理器，核 3 为高功率处理器。3 个处理器在最大运行频率下的功率消耗 P_k 分别为 P_1=1、P_2=2、P_3=4。移动设备卸载任务时的传输功率 p_l^{txd}=0.5。每个处理器具有 H=4 种可调节的频率，频率调节因子分别为 $\alpha_{k,1}$=0.2、$\alpha_{k,2}$=0.5、$\alpha_{k,3}$=0.8、$\alpha_{k,4}$=1；且 γ_k=2，k=1,2,3。

为了验证本文提出的 MOEA/D-DVFS 在 4 种性能评价指标上的优越性，比较以下 4 种算法，每种算法独立运行 50 次，统计每次运行的 IGD、GD 和 ACT 值，并且计算这 50 次的均值和标准差。

1) NSGA-Ⅱ（$A1$）：Zhou 等[16]针对基于时延传输的多目标工作流调度问题，提出用 NSGA-Ⅱ来解决，通过延长非关键任务的执行时间来降低移动设备的 EC。种群规模和最大迭代次数都设为 100，交叉概率和变异概率为 1。

2) DVFS（$A2$）：Lin 等[13]提出了 DVFS 技术来进一步降低移动设备的 EC。

3) MOEA/D（$A3$）：Zhang 等[17]提出了 MOEA/D，将 MOP 分解为多个标量的子问题。设分解方式为 Tchebycheff，种群规模 P=100 终止条件为达到最大迭代次数 100，邻居大小 T=10，遗传算子的交叉概率和变异概率均为 1。

4) MOEA/D-DVFS（Prop.）：本文提出的基于 MOEA/D 的多目标任务卸载算法，该算法将 DVFS 技术引入原始的 MOEA/D 中来进一步降低移动设备的 EC，算法参数设置与 MOEA/D 相同。

4 种算法在所有测试场景下的 IGD 和 GD 统计结果如表 1 所示，显然本文提出的 MOEA/D-DVFS 的 IGD 和 GD 值均小于其他 3 种算法，表明在 MCC 环境下的多目标任务卸载问题上，本文提出的算法性能最优。如 4.1 节所述，IGD 反映算法的整体性能，可同时评估算法的收敛性和多样性，IGD 值越小，表明算法整体性能越好。同样，提出的算法也得到了最小的 GD，该值反映出值越小则算法的近似 Pareto 解集越接近于真实的 Pareto 前沿。此外，DVFS 在所有实验场景下具有 0 值标准差，由于 DVFS 是一种启发式算法，与其他 3 种随机搜索的进化算法相比，不具有随机性，因此，DVFS 算法运行多次得到的 Pareto 解集一样，且通过比较 IGD 和 GD 值证明解质量较差。

ACT 可评价算法计算复杂度，ACT 统计结果如表 2 所示，统计了所有算法运行 50 次的 ACT，其中，最优值加粗。由表 2 可看出，当 N=10、20、30、40 时，$A2$ 的 ACT 值最小；当 N 大于或等于 50 时，$A3$ 的 ACT 值最小。所以，当应用任务数较大时，$A3$ 的时间复杂度较低，算法更具有实用性。此外，本文所提算法的 ACT 值大于 $A3$，由于所提算法在 $A3$ 的基础上增加了 DVFS 技术，需消耗一定时间，但两者 ACT 值非常接近，因此，所提算法在解决多目标任务卸载问题上同样具有实用性。

表 1　　**4 种算法在所有测试场景下的 IGD 和 GD 统计结果**

N	IGD				GD			
	$A1$	$A2$	$A3$	Prop.	$A1$	$A2$	$A3$	Prop.
10	2.62(0.10)	3.68(0.00)	2.66(0.16)	**0.82**(0.58)	1.81(0.15)	2.02(0.00)	1.73(0.18)	**0.73**(0.31)
20	8.10(0.22)	8.20(0.00)	8.22(0.51)	**3.48**(0.77)	2.82(0.10)	2.87(0.00)	2.79(0.09)	**1.77**(0.20)
30	13.21(0.59)	14.40(0.00)	12.52(0.83)	**5.19**(1.07)	3.55(0.11)	3.80(0.00)	3.45(0.08)	**1.83**(0.22)
40	22.27(2.24)	19.35(0.00)	16.60(1.36)	**8.00**(2.55)	3.84(0.13)	4.29(0.00)	3.81(0.11)	**1.86**(0.21)
50	33.74(3.36)	23.24(0.00)	20.73(1.63)	**8.77**(2.34)	4.28(0.22)	4.63(0.00)	4.09(0.17)	**2.16**(0.22)
60	44.05(4.55)	26.86(0.00)	27.32(2.69)	**14.32(**3.91)	4.53(0.20)	4.96(0.00)	4.53(0.19)	**2.23**(0.24)
70	63.05(6.64)	24.28(0.00)	26.92(2.56)	**12.56**(3.29)	5.14(0.35)	4.83(0.00)	4.47(0.21)	**2.27**(0.28)
80	77.55(6.53)	30.10(0.00)	29.29(2.27)	**11.09**(3.10)	5.71(0.36)	5.14(0.00)	4.98(0.19)	**2.56**(0.27)
90	90.09(8.57)	33.03(0.00)	34.76(2.26)	**13.67**(4.23)	6.43(0.38)	5.69(0.00)	5.38(0.14)	**2.79**(0.43)
100	96.05(7.83)	37.16(0.00)	38.61(2.38)	**16.68**(4.12)	5.53(0.36)	5.31(0.00)	5.22(0.19)	**2.55**(0.33)

注：表中 $x(y)$的 x 和 y 分别表示均值和标准差，最优值加粗。

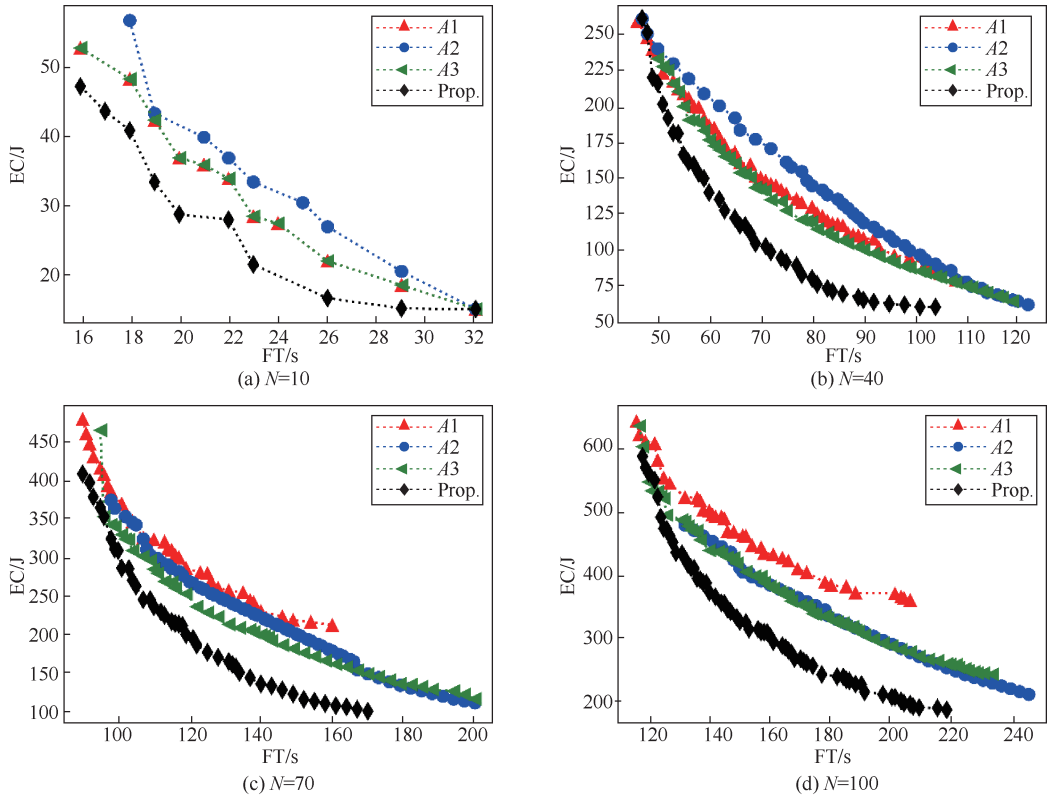


图 1　4 种算法在 4 种场景下的近似 Pareto 曲线

表 2　　ACT 统计结果/s

N	*A*1	*A*2	*A*3	Prop.
10	7.88	**0.17**	4.79	4.96
20	11.50	**1.21**	8.61	8.71
30	15.09	**4.85**	12.08	12.51
40	18.84	**10.65**	16.43	17.09
50	22.57	25.24	**20.67**	21.24
60	26.41	47.51	**24.53**	25.48
70	30.07	73.96	**28.44**	29.02
80	34.02	124.39	**32.78**	33.49
90	37.85	194.31	**36.96**	37.98
100	41.73	323.55	**40.73**	42.49

注：表中最优值加粗。

为了直观地展示算法的 Pareto 解集的分布性和收敛性，4 种算法在 4 种场景下的近似 Pareto 曲线如图 1 所示，绘制了 4 个实验场景（即 *N*=10、40、70、100）中 4 种算法独立运行 50 次得到的近似 Pareto 解集。横坐标为应用 FT，纵坐标为移动设备 EC，且两者均是最小化目标，由图 1 可知，本文所提算法更接近于真实的 Pareto 前沿。

综上所述，本文提出的算法在 4 种算法性能评估指标上更优，能较好地解决多目标任务卸载问题。

6　结束语

任务卸载问题是 MCC 环境中的重要研究内容，设计卸载策略时的一个关键问题是怎样均衡移动设备的应用 FT 和 EC。以往的研究工作主要关注降低移动设备 EC 或者减少应用 FT，较少同时优化两者，即使联合优化两个目标，也不能在 FT 和 EC 之间取得较好的平衡。而且，当应用任务数量较大时，传统方法也不能较好地解决本文问题。

鉴于此，针对 MCC 环境下的多目标任务卸载问题，本文提出了基于 MOEA/D 的多目标进化算法，该算法将 DVFS 技术引入 MOEA/D 中，来进一步降低移动设备的 EC。仿真结果表明，在多个实验场景下，与 NSGA-Ⅱ、DVFS、原始 MOEA/D 算法相比，本文所提算法在 IGD、GD、Pareto 前沿曲线以及 ACT 指标上更优，从而能较好地均衡移动设备的应用 FT 和 EC。

参考文献：

[1] ATAT R, LIU L, CHEN H, et al. Enabling cyber-physical communication in 5G cellular networks: challenges, spatial spectrum sensing, and

cyber-security[J]. IET Cyber-Physical Systems: Theory & Applications, 2017, 2(1): 49-54.

[2] LI C, ZHU L, TANG H, et al. Mobile user behavior based topology formation and optimization in Ad Hoc mobile cloud[J]. The Journal of Systems and Software, 2019, 148: 132-147.

[3] LI L, LIU Z, TSENG M, et al. Enhancing the lithium-ion battery life predictability using a hybrid method[J]. Applied Soft Computing, 2019, 74: 110-121.

[4] NOOR T, ZEADALLY S, ALFAZI A, et al. Mobile cloud computing: challenges and future research directions[J]. Journal of Network and Computer Application, 2018, 115: 70-85.

[5] MAO Y, YOU C, ZHANG J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[6] DENG S, HUANG L, TAHERI J, et al. Computation offloading for service workflow in mobile cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(12): 3317-3329.

[7] CAI Z, CHEN C. Demand-driven task scheduling using 2D chromosome genetic algorithm in mobile cloud[C]//IEEE International Conference on Progress in Informatics and Computing. IEEE, 2014: 539-545.

[8] YANG L, CAO J, CHENG H, et al. Multi-user computation partitioning for latency sensitive mobile cloud applications[J]. IEEE Transactions on Computers, 2015, 64(8): 2253-2266.

[9] BALAMURUGAN M, AKILA V. Effective processor selection on heterogeneous computing[C]//Proceedings of IEEE Second International Conference on Science Technology Engineering and Management. IEEE, 2016: 13-16.

[10] KUMAR K, LU Y H. Cloud computing for mobile users: can offloading computation save energy?[J]. Computer, 2010, 43(4): 51-56.

[11] TONG L, GAO W. Application-aware traffic scheduling for workload offloading in mobile clouds[C]//IEEE International Conference on Computer Communications. IEEE, 2016: 1-9.

[12] MAHMOODI S E, UMA R N, SUBBALAKSHMI K P. Optimal joint scheduling and cloud offloading for mobile applications[J]. IEEE Transactions on Cloud Computing, 2019, 7(2): 301-313.

[13] LIN X, WANG Y, XIE Q, et al. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment[J]. IEEE Transactions on Services Computing, 2015, 8(2):175-186.

[14] DENG S, HUANG L, TAHERI J, et al. Computation offloading for service workflow in mobile cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(12): 1.

[15] GUO S, LIU J, YANG Y, et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing[J]. IEEE Transactions on Mobile Computing, 2019, 18(2): 319-333.

[16] ZHOU Y, LI Z, GE J, et al. Multi-objective workflow scheduling based on delay transmission in mobile cloud computing[J]. Chinese Journal of Computer, 2018, 29(11): 72-91.

[17] ZHAN F Q, LI H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[18] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[19] MIETTINEN K. Nonlinear multiobjective optimization[M]. Holland: Kluwer, 1999.

[20] COELLO C A, LAMONT G B. Applications of multi-objective evolutionary algorithms[M]. California: World Scientific, 2004.

[21] TRIVEDI A, SRINIVASAN D, SANYAL K, et al. A survey of multiobjective evolutionary algorithms based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(3): 440-462.

[22] YU X, CHE N, GU T, et al. Set-based discrete particle swarm optimization based on decomposition for permutation-based multiobjectivecombinatorial optimization problems[J]. IEEE Transactions on Cyber netics, 2018, 44(8): 2139-2153.

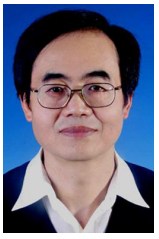
[23] ZHU Y, WANG J, QU B. Multi-objective economic emission dispatch considering wind power using evolutionary algorithm based on decomposition[J]. International Journal of Electrical Power & Energy Systems, 2014, 63: 434-445.

[24] XING H, WANG Z, LI T, et al. An improved MOEA/D algorithm for multi-objective multicast routing with network coding[J]. Applied Soft Computing, 2017, 59: 88-103.

[作者简介]

宋富洪（1992− ），男，贵州遵义人，西南交通大学博士生，主要研究方向为移动边缘计算和多目标进化算法。

邢焕来（1983− ），男，河北唐山人，西南交通大学副教授，主要研究方向为计算机网络、进化计算和边缘计算。



潘炜（1959− ），男，湖南岳阳人，西南交通大学教授、博士生导师，主要研究方向为通信与信息系统、微波光子学等。