移动云环境中数据流应用的 Cloudlet 选择策略研究

刘伟[1,2]，熊曙[1]，杜薇[1,2]，王伟[3]

（1. 武汉理工大学计算机科学与技术学院，湖北 武汉 430070；
2. 交通物联网技术湖北省重点实验室，湖北 武汉 430070；
3. 同济大学计算机科学与技术系，上海 200092）

摘 要：现有的 Cloudlet 选择策略大多只使用单个 Cloudlet 资源进行计算卸载，对于拥有较多可并行执行组件的移动数据流应用程序，性能提升有限。针对这一问题，提出一种基于化学反应优化算法的 Cloudlet 选择策略。该策略以减少应用的完成时间和移动设备能耗为目的，在满足应用程序组件间依赖关系的前提下，充分利用多 Cloudlet 的计算资源使移动数据流应用的并行组件同时执行，提升了应用执行效率的同时降低了移动设备能耗。仿真实验表明，在多 Cloudlet 环境中应用程序的性能相较于 POCSS 策略平均提升了 18.2%。

关键词：微云选择；能耗；完成时间；移动数据流应用；化学反应算法

中图分类号：TP319

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2019020

Research on Cloudlet selection strategy for data streaming applications in mobile cloud environment

LIU Wei[1,2], XIONG Shu[1], DU Wei[1,2], WANG Wei[3]

1. School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China
2. Hubei Key Laboratory of Transportation Internet of Things, Wuhan 430070, China
3. Department of Computer Science and Technology, Tongji University, Shanghai 200092, China

Abstract: Most existing Cloudlet selection strategies only used the resources of one Cloudlet to compute offloading, which couldn't obtain the superior performance improvement for mobile data streaming application with many parallel components. To address this issue, a Cloudlet selection strategy based on chemical reaction optimization algorithm was proposed. The strategy aims to reduce application's completion time and energy consumption of mobile device. When the dependencies among application's components was satisfied, the strategy can take full advantage of the computing capability of multi-cloudlet to execute the parallel components of mobile data stream application simultaneously. Therefore the strategy can improve the execution efficiency and reduce the energy consumption of mobile device. The simulation results reveal that the proposed strategy can achieves 18.3% on average performance improvement than POCSS strategy does in multi-Cloudlet environment.

Key words: Cloudlet selection, energy consumption, completion time, mobile data streaming application, chemical reaction optimization algorithm

收稿日期：2018−02−05；**修回日期**：2018−07−16

基金项目：国家自然科学面上基金资助项目（No.61672384）；教育部人文社科基金资助项目（No.16YJCZH014）；中央高校基本科研业务基金资助项目（No.2016III028, No.2017III028-005）

Foundation Items: The Chinese National Natural Science Foundation (No.61672384), The Ministry of Education Project of Humanities and Social Sciences (No.16YJCZH014), The Fundamental Research Funds for the Central Universities (No.2016III028, No.2017III028-005)

1 引言

随着移动互联网技术的迅猛发展，以智能手机为代表的移动设备越来越普及，据国际电信联盟数据显示，在 2017 年全球智能手机拥有量预计达到 43 亿[1]。利用种类繁多的传感器，移动设备上产生了诸多如手势识别[2]、人脸识别[3]等移动数据流应用程序，这类应用对延迟比较敏感，运行时需要大量的计算资源，且会导致移动设备大量消耗电量。然而移动设备的电池容量、计算能力、存储空间等资源比较有限，在运行此类应用时，移动设备的电量面临着严峻的考验。移动云计算（MCC，mobile cloud computing）的出现为突破移动设备资源限制提供了可能[4]，将应用的部分计算密集型任务通过计算卸载技术[5]卸载到远程的云计算中心上执行，有效地降低了移动设备的电量消耗，如 MAUI（mobile assistance using infrastructure）[6]、CloneCloud[7]、Odessa[8]等。

然而，远程云计算中心距离移动用户较远，通过广域网连接将会带来较高的网络延迟[9]，很难保证移动数据流应用的服务质量要求。为此，美国卡内基梅隆大学 Satayanarayanan 教授[10]首次提出 Cloudlet（微云）这一概念，它是距离移动设备较近且资源丰富的一台或多台机器组成的计算机集群。移动设备可以通过局域网与 Cloudlet 直接相连，拥有带宽大和延迟低的优势[11]。因此，这种计算模式将移动数据流应用卸载到 Cloudlet 上执行，能够获得良好的用户体验。

随着移动云计算的发展和普及，各商家或单位部署有不同计算、存储和网络等资源的 Cloudlet，可以同时为区域内的用户提供服务。用户在运行诸如人脸识别等移动数据流应用时，如何依据周围 Cloudlet 收集的移动设备、应用特征和网络状况等信息，为应用选择合适的 Cloudlet 进行网络连接和计算卸载，以延长移动设备的续航时间和提高应用程序的性能成为亟待解决的问题。

针对这一问题，研究者们从不同的角度展开了工作。然而，这些工作大多只为移动设备上的应用选择单个 Cloudlet 进行计算卸载，对于拥有较多并行组件的移动数据流应用，如光学字符识别、QR 二维码应用[12]、对象识别[13]等，性能提升有限。为此，本文提出了一种可以充分利用多个 Cloudlet 资源来最大化移动数据流应用并行执行效率的 Cloudlet 选择策略，为用户带来更短的应用完成时间和更低的移动设备能量消耗。本文主要贡献如下。

1) 建立了移动数据流应用的 Cloudlet 选择问题的系统模型，旨在最小化应用完成时间和移动设备能耗，同时将问题定义为双目标的组合优化问题，并证明该问题是 NP-hard 问题。

2) 设计了一个动态调整的适应度函数，根据移动设备的剩余电量调整应用完成时间与移动设备能耗的权值，进而采用线性加权法将该双目标问题归一成单目标问题。

3) 提出了一种基于化学反应优化算法的 Cloudlet 选择策略 CROCS（chemical reaction optimization Cloudlet selection strategy）。该策略能够在满足组件间依赖关系的前提下，充分利用移动数据流应用中可以并行执行的组件，提高了应用性能。

2 研究动机

本文所提出的移动数据流应用的 Cloudlet 选择策略的主要思想是将移动数据流应用的并行组件分散到多个 Cloudlet 上执行，以提高应用的执行效率，减少完成时间。然后，将通过一个具有一般拓扑结构的移动数据流应用程序——光学字符识别（OCR，optical character recognition）在多 Cloudlet 环境中运行的例子验证本文思想的正确性。

光学字符识别应用程序通过检查输入图片中的手写体或者印刷体字符，经过明暗处理后确定文字的形状，然后采用字符识别方法将其翻译成计算机文字。其应用程序结构如图 1 所示，其中组件 v_0 为开始组件，表示获取输入图片；组件 v_1 表示一维最大熵法计算阈值；组件 v_2 表示最大类间方差法计算阈值；组件 v_3 表示迭代法计算阈值；组件 v_4 表示选择二值化；组件 v_5 表示识别过程组件 v_6 为结束组件，表示输出结果。组件 v_1、v_2 和 v_3 为可以同时在不同 Cloudlet 或者移动设备上执行的并行组件。其中各个组件的计算量分别为 0、3 600、3 800、3 400、1 800、14 200 和 0（单位为百万条指令），边的权值表示组件之间的数据传输量大小，单位是 KB。

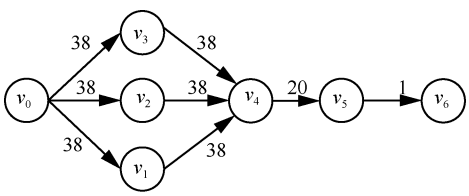

图 1 光学字符识别应用程序结构

如图 2 所示，当用户运行该应用程序时，周围有 3 个具有相同计算能力的 Cloudlet（用 C_1、C_2 和 C_3 表示）可作为计算节点，其中 C_1 作为距离用户最近的代理 Cloudlet，用户会向其发送计算卸载请求，同时将移动设备的硬件信息发送给 C_1 以备其做出选择策略。然后代理 Cloudlet 依据周围 Cloudlet 之间的带宽与距离、移动设备与各 Cloudlet 的计算能力与最早开始时间、代理 Cloudlet 与移动设备之间的距离和带宽等信息做出合适的选择策略[14]，Cloudlet 端和移动设备依据选择策略协同执行计算任务，最后将应用程序的结果返回到移动设备。设置移动设备与 Cloudlet 的计算能力分别为 40 000 和 80 000，单位是每秒钟执行百万指令数，与不同 Cloudlet 之间的带宽为 8 Mbit/s，Cloudlet 之间的带宽为 240 Mbit/s。采用不同选择方案对应的应用完成时间如表 1 所示。

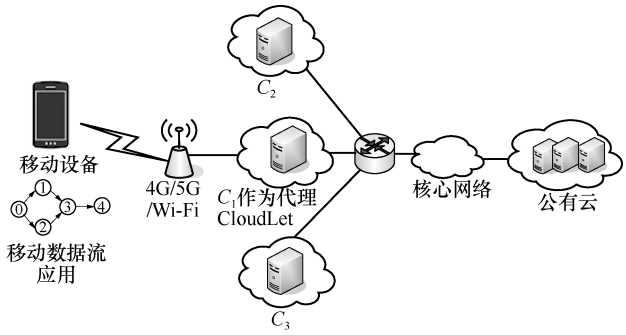

图 2　移动数据流应用 Cloudlet 选择场景

表 1　不同选择方案对应的应用完成时间

方案编号	移动设备	C_1	C_2	C_3	完成时间/s
1	0,6	1,4,5	2	3	0.286
2	0,1,6	2	3	4,5	0.324
3	0,1,2,3,6	4,5	—	—	0.498
4	0,6	2,3,4,5	1	—	0.329
5	0,6	1,2,3,4,5	—	—	0.370
6	0,1,2,3,4,5,6	null	—	—	0.660

通过以上例子表明，应用的完成时间随着不同的选择方案而不同，并且其中并行组件较为分散的方案（如方案 1、2 和 4）相较于其他并行组件集中的方案（如方案 3、5 和 6）对应的完成时间较低。因此可以通过选取合适的选择方案加快应用执行速度。下面基于这一思想建立相应模型、设计相应算法，找出合适的选择策略，并通过实验验证本文思想的正确性。

3　相关工作

随着人脸识别等移动数据流应用越来越流行和移动云计算的普及，用户在运行移动数据流应用时将常处于多 Cloudlet 环境中，如何选择 Cloudlet 为用户提供良好的服务是一个亟需解决的问题。在已有的工作中，文献[15]针对非集中式部署的多 Cloudlet 环境，提出了一种应用资源限制的选择策略，选择满足应用对资源需求的 Cloudlet 进行计算卸载。文献[16]依据距离进行选择，为应用优先选择距离用户最近的 Cloudlet 卸载应用。文献[17]考虑到离用户最近的 Cloudlet 未必能提供最好的计算能力和良好的网络质量，提出选择 2 个 Cloudlet，其中一个拥有较好的网络连接，另一个拥有较好的计算能力来保证执行应用的性能。文献[18]则基于不同应用对资源需求具有差异化的特点，对已发现的 Cloudlet 按照满足应用特点的服务质量属性进行排名，依据该排名选择合适的 Cloudlet 进行卸载应用。文献[19]则考虑应用卸载过程中移动设备的能量消耗，选择移动设备能量消耗最低的 Cloudlet。Mukherjee 等[14]为了避免文献[16]中某些任务可能需要卸载到远程云带来较高延迟的情况，提出将最近的 Cloudlet 作为代理服务器，代理服务器通过广播请求获得周围 Cloudlet 的信息，依据获得的信息选择最低移动设备能耗或最低完成时间的 Cloudlet 。文献[20]则将不同的应用部署到不同的 Cloudlet 上，而后依据应用种类选择 Cloudlet，系统中的应用性能得到了进一步的提升。

以上这些工作都没有考虑具有一般拓扑结构的移动数据流应用有许多并行部分，可以在多个计算节点上执行。文献[21]利用这些并行部分，使应用的部分组件在移动设备上执行，部分组件在 Cloudlet 上执行，提升了应用执行效率，进而减小了应用的完成时间。中国科学技术大学的刘炜清等[22]针对单个 Cloudlet 与移动设备的带宽有限从而限制了移动数据流应用吞吐量的问题，提出利用将部分组件的实例放置到其他 Cloudlet 上执行，提高了应用的吞吐量。与之对比，本文工作主要是充分利用多个 Cloudlet 的资源使移动数据流应用的并行组件同时执行，提升了应用执行效率，使移动数据流应用的完成时间最小化，降低移动设备能耗。

4　系统模型

本文的目标是寻找一种适合移动数据流应用

的 Cloudlet 选择策略以最小化应用的完成时间和移动设备能量消耗。为解决这一问题，首先需要对相关概念和移动数据流应用进行建模，进而建立相应的完成时间和移动设备能耗建立模型。

4.1 相关概念模型

1) 移动设备

移动设备可以用一个八元组表示，mobile={power, capacity, P_{idle}, P_{comp}, P_{ts}, P_{tr}, J, D_J}，其中，power 表示移动设备的剩余电量，capacity 表示移动设备的计算能力（单位是 MIPS），P_{idle} 表示移动设备 CPU 处于空闲状态时的功率，P_{comp} 表示移动设备 CPU 处于计算状态时的功率，P_{ts} 表示移动设备发送数据的功率，P_{tr} 表示移动设备接收数据的功率，J 表示代理 Cloudlet 编号，D_J 表示移动设备与代理 Cloudlet 的距离。

2) Cloudlet

系统中的 Cloudlet 可以通过 Cloudlets=(C,D)表示，C 表示 Cloudlet 的集合，D 表示 Cloudlet 之间的关系。C={Cloudlet$_1$, Cloudlet$_2$,…, Cloudlet$_m$}，$|C|=m$。对于 C 中每一个 Cloudlet 可以使用一个三元组表示，Cloudlet$_j$={j, capacity$_j$, Q_j}，其中 j 表示编号，capacity$_j$ 表示计算能力大小（单位是 MIPS），Q_j 表示最早开始执行时间。

由于系统中存在多个 Cloudlet，它们之间的关系主要通过距离体现，所以采用一个矩阵 $\boldsymbol{D}$ 表示其距离关系，$\boldsymbol{D}=\begin{bmatrix} d_{1,1} & \cdots & d_{1,m} \\ \vdots & \ddots & \vdots \\ d_{m,1} & \cdots & d_{m,m} \end{bmatrix}$，其中，$d_{i,j}$ 表示 Cloudlet$_i$ 与 Cloudlet$_j$ 之间的距离（单位是 m），有 $d_{i,j}=d_{j,i}$，且当 $i=j$ 时，$d_{i,j}=0$。

3) 网络连接

网络连接主要由 2 个部分组成，分别是移动设备与代理 Cloudlet 之间的网络，以及 Cloudlet 之间的网络。移动设备与代理 Cloudlet 之间的上传和下载带宽分别为 B^u 和 B^d；Cloudlet 之间通过带宽相同的有线网络连接，带宽单位是 B。

4) Cloudlet 选择策略

移动数据流应用中的每一个组件可以在 Cloudlet 或移动设备上执行，对于一个由 n 个组件组成的移动数据流应用，其 Cloudlet 选择策略可以使用一个 n 维向量表示，$\boldsymbol{X}=\{x_1, x_2, \cdots, x_n\}$，$x_i \in [0,m]$，其中 x_i 表示第 i 个组件的执行位置。当 $x_i=0$ 时，表示该组件在移动设备上执行，当 $x_i \in [1,m]$ 时，表示该组件在 Cloudlet$_{x_i}$ 上执行。

4.2 移动数据流应用模型

移动数据流应用可以建模成有向无环图 $G(V,E)$，其中 $V=\{v_1, v_2, \cdots, v_n\}$ 表示组件的集合，E 表示组件之间的依赖关系。对于 V 中每一个组件 v_i 可以用一个三元组表示，$v_i=\{i, x_i, I_i\}$，其中 i 为组件编号，x_i 为组件 v_i 选择的 Cloudlet 位置，I_i 为组件 v_i 的指令数。E 中每条边的权重 data$_{j,k}$ 表示组件 v_j 和 v_k 之间传输的数据量大小。如图 1 所示，入口组件 v_0 和出口组件 v_6 表示虚拟组件，且这 2 个组件必须在移动设备上执行。

4.3 应用完成时间模型

完成时间[8]是移动数据流应用一个重要的性能指标，表示应用程序从数据输入到结果输出的时间。完成时间越低表示应用的响应速度越快，用户体验越好。

对于 Cloudlet 选择策略 X，因为应用的组件可以在不同的位置执行，执行时组件之间会产生数据传输时间和传播时间，其中包含并行组件在多个 Cloudlet 或移动设备上并行执行产生的额外通信时间开销。对于组件 v_k 和 v_i 之间的数据传输时间 trans$_{k,i}$ 可以由式(1)计算。

$$\text{trans}_{k,i}=\begin{cases} \dfrac{\text{data}_{k,i}}{B^u}+\dfrac{\text{data}_{k,i}}{B}, & x_k=0 \text{且} x_i \neq 0, J \\ \dfrac{\text{data}_{k,i}}{B^u}, & x_k=0 \text{且} x_i=J \\ \dfrac{\text{data}_{k,i}}{B}, & x_k, x_i \in [1,m] \text{且} x_i \neq x_k \\ \dfrac{\text{data}_{k,i}}{B}+\dfrac{\text{data}_{k,i}}{B^d}, & x_k \neq 0, J \text{且} x_i=0 \\ \dfrac{\text{data}_{k,i}}{B^d}, & x_k=J \text{且} x_i=0 \\ 0, & x_k=x_i \end{cases} \tag{1}$$

其中，当 $x_k=0$ 且 $x_i \neq 0, J$ 时，表示组件 v_k 在移动端执行，v_i 在除代理 Cloudlet$_J$ 之外的其他 CloudLet$_{x_i}$ 上执行，其数据传输时间由 2 个部分组成：1) 数据从移动设备传输到代理 Cloudlet$_J$ 的时间，2) 数据由代理 Cloudlet$_J$ 传输到 Cloudlet$_{x_i}$ 上的时间。当 $x_k=0$ 且 $x_i=J$ 时，即组件 v_k 在移动设备上执行，组件 v_i 在代理 Cloudlet$_J$ 上执行，其数据传输时间为组件间的数据量 data$_{k,i}$ 与移动设备上传数据带宽 B^u 的比值。当 $x_k, x_i \in [1,m]$ 且 $x_i \neq x_k$

时，即组件 v_k 和组件 v_i 在不同的 Cloudlet 上执行，其数据传输时间为组件间的数据量 $\text{data}_{k,i}$ 与 Cloudlet 之间的带宽 B 的比值。当 $x_k \neq 0, J$ 且 $x_i=0$ 时，即组件 v_k 在除了代理 Cloudlet_J 之外的 CloudLet_{x_k} 上执行，组件 v_i 在移动设备上执行，其数据传输时间包括数据从 Cloudlet_{x_k} 传输到代理 Cloudlet_J 的时间和数据从代理 Cloudlet_J 传输到移动设备的时间。当 $x_k = J$ 且 $x_i=0$ 时，表示组件 v_k 在代理 Cloudlet_J 上执行，组件 v_i 在移动设备上执行，其数据传输时间为组件间的数据传输量 $\text{data}_{k,i}$ 与移动设备下载数据的带宽的比值。当 $x_i=x_k$ 时，表示组件 v_k 和组件 v_i 在同一位置执行，其数据传输时间为 0。

组件 v_k 和 v_i 之间的传播时间 $\text{tprop}_{k,i}$ 可以由式(2)得出。

$$\text{tprop}_{k,i}=\begin{cases}\dfrac{D_J}{S}+\dfrac{d_{J,x_i}}{S}, & x_k=0, x_i\in[1,m]\\ \dfrac{d_{k,i}}{S}, & x_k, x_i\in[1,m]\text{且}x_i\neq x_k\\ \dfrac{d_{x_k,J}}{S}+\dfrac{D_J}{S}, & x_k\in[1,m], x_i=0\\ 0, & x_k=x_i\end{cases} \tag{2}$$

其中，S 为传播速度，大小为 $2\times10^8\text{m/s}$[23]。当 $x_k=0$，$x_i\in[1,m]$ 时，其数据传播时间包括 2 个部分：1) 组件 v_k 所在的移动设备与代理 Cloudlet_J 之间的传播时间;2) 代理 Cloudlet_J 与组件 v_i 所在 Cloudlet_{x_i} 之间的传播时间。当 x_k , $x_i\in[1,m]$ 且 $x_i \neq x_k$，即组件 v_k 和 v_i 在不同 Cloudlet 上执行，其传播时间为两组件所选择执行位置的距离与传播速度比值。当 $x_k\in[1,m]$, $x_i=0$ 时，其数据传播时间包括组件 v_k 所选择的 Cloudlet_{x_k} 与代理 Cloudlet_J 之间的传播时间和代理 Cloudlet_J 与组件 v_i 所在移动设备的传播时间。当 $x_k=x_i$ 时，即组件 v_k 和 v_i 在同一个位置执行，其传播时间为 0。

由此可以得到组件 v_i 的开始执行时间 ST_i 和完成时间 FT_i 的计算式，如式(3)和式(4)所示。

$$\text{ST}_i=\begin{cases}0, & \text{pred}(i)=\varnothing\\ \max\left(\max\limits_{k\in\text{pred}(i)}(\text{trans}_{k,i}+\text{tprop}_{k,i}+\text{FT}_k), \text{RT}_{x_i}\right), & x_i\in[1,m]\\ \max\left(\max\limits_{k\in\text{pred}(i)}(\text{trans}_{k,i}+\text{tprop}_{k,i}+\text{FT}_k), \text{RT}_m\right), & x_i=0\end{cases} \tag{3}$$

其中，使用 $\text{pred}(i)$ 表示 v_i 的前驱组件的集合。$\text{trans}_{k,i}$ 和 $\text{tprop}_{k,i}$ 分别表示组件 v_k 和 v_i 之间的数据传输时间和数据传播时间。RT_{x_i} 表示 Cloudlet_{x_i} 的最早开始执行时间，初始为 Q^{x_i}，在组件选择执行位置后对 RT_{x_i} 进行更新。RT_m 表示移动设备的最早开始执行时间，初值为 0，在组件选择执行位置后对 RT_m 进行更新。当组件 v_i 的前驱组件集合为空时，即 v_i 为开始组件，其开始执行时间为 0；当 $x_i\in[1,m]$，即组件 v_i 的执行位置在 Cloudlet_{x_i}，其开始执行时间为前驱组件中最晚将数据传输到 Cloudlet_{x_i} 的时间与其最早开始执行时间 RT_{x_i} 的最大值；当 $x_i=0$ 时，即组件 v_i 的执行位置为移动设备，其开始执行时间为前驱组件中最晚将数据传输到移动设备的时间与移动设备最早能开始执行时间 RT_m 的最大值。

组件 v_i 的完成时间可以由式(4)计算。

$$\text{FT}_i=\begin{cases}\text{ST}_i+\dfrac{I_i}{\text{capacity}_{x_i}}, & x_i\in[1,m]\\ \text{ST}_i, & i=1\text{或}i=n\\ \text{ST}_i+\dfrac{I_i}{\text{capacity}}, & x_i=0\end{cases} \tag{4}$$

当 $i=1$ 或 $i=n$ 时，即组件 v_i 为开始组件和结束组件时，其完成时间为其开始执行时间；当组件 v_i 不是开始组件或者结束组件时，完成时间为其开始执行时间加上其所在执行位置上的执行时间。

由此，可以得到应用的完成时间为

$$\text{FT}=\text{FT}_n \tag{5}$$

4.4 移动设备能耗模型

移动设备能耗主要由无线网络接口、屏幕、CPU、GPS 等部件产生的能耗[24]组成，本文主要考虑无线网络接口数据传输和 CPU 能耗。

4.4.1 数据传输能耗

本文中数据传输能耗包括数据发送和接收时移动设备能量消耗。

1) 数据发送能耗

$$E_{\text{send}}=P_{\text{ts}}\sum_{k\in\text{succ}(i),x_i=0}\text{trans}_{i,k} \tag{6}$$

其中，$\text{succ}(i)$ 表示组件 v_i 后继组件的集合，$\sum\limits_{k\in\text{succ}(i),x_i=0}\text{trans}_{i,k}$ 表示移动设备无线网络发送数据所用时间的总和。

2) 数据接收能耗

$$E_{\text{recev}} = P_{\text{tr}} \sum_{k\in\text{pred}(i),x_i=0} \text{trans}_{k,i} \tag{7}$$

其中，$\sum_{k\in\text{pred}(i),x_i=0} \text{trans}_{k,i}$ 表示移动设备无线网络接收数据所用时间的总和。

4.4.2 CPU 能耗

CPU 能耗包括计算能耗和空闲能耗。计算能耗表示有组件在移动设备上执行时 CPU 的能量消耗，空闲能耗为没有组件在移动设备上执行时 CPU 处于空闲状态的能量消耗。

1) 计算能耗

$$E_{\text{comp}} = P_{\text{comp}} \sum_{x_i=0} \frac{I_i}{\text{capacity}} \tag{8}$$

其中，$\sum_{x_i=0} \frac{I_i}{\text{capacity}}$ 表示在移动设备上执行组件运行时间总和，即移动设备 CPU 处于运行状态的时间。

2) 空闲能耗

空闲时间可以通过完成时间减去数据发送时间、数据接收时间和计算时间计算。

$$T_{\text{idle}} = \text{FT} - \sum_{k\in\text{succ}(i),x_i=0} \text{trans}_{i,k} - \sum_{k\in\text{pred}(i),x_i=0} \text{trans}_{k,i} - \sum_{x_i=0} \frac{I_i}{\text{capacity}} \tag{9}$$

由此可以计算出空闲能耗

$$E_{\text{wait}} = P_{\text{idle}} T_{\text{idle}} \tag{10}$$

则移动设备能耗可以表示为

$$E_{\text{total}} = E_{\text{send}} + E_{\text{recev}} + E_{\text{comp}} + E_{\text{wait}} \tag{11}$$

4.5 问题定义

在 4.3 节和 4.4 节分别介绍了移动数据流应用的完成时间模型和移动设备能耗模型。对于给定的应用、移动设备和 Cloudlets 等信息，可以得到任意 Cloudlet 选择策略 X 对应的应用完成时间 FT 和移动设备能耗 E_{total}。本文的 Cloudlet 选择问题是寻找合适的 Cloudlet 选择策略 X，最小化应用的完成时间和移动设备能耗，可以由式(12)定义。

$$\min_X f(\text{FT}, E_{\text{total}}) \tag{12}$$

约束条件如下。

(a) $x_0 = x_n = 0$

(b) $x_i \in [0,m], i \neq 0,n$

(c) $\text{trans}_{k,i} + \text{tprop}_{k,i} + \text{FT}_k \leqslant \text{ST}_i, \forall k \in \text{pred}(i)$

限制条件(a)表示入口组件 v_0 和出口组件 v_n 必须在移动设备上执行，限制条件(b)表示中间组件的执行位置为移动设备或者 Cloudlet；限制条件(c)表示组件之间的依赖关系，即组件 v_i 开始执行之前必须等待其前驱组件执行完成并将所需数据传输到该组件 v_i 所在执行位置。

命题 1 式(12)是一个 NP-hard 问题。

证明 假设不考虑移动设备能耗，该问题就变成了最小化应用完成时间的单目标组合优化问题，并且认为开始组件和结束组件可以在任何计算节点上执行，则该问题可规约成式(13)形式。

$$\min_X \text{FT} \tag{13}$$

约束条件如下。

(a) $x_i \in [0,m]$

(b) $\text{trans}_{k,i} + \text{tprop}_{k,i} + \text{FT}_k \leqslant \text{ST}_i, \forall k \in \text{pred}(i)$

而式(13)与并行程序在并行计算机系统最小化完成时间的调度问题[25]等价。依据文献[26]，该问题的特殊情况是 NP-hard 问题，所以式(12)的问题也是 NP-hard 问题。证毕。

5 算法设计

由 4.5 节可知该 Cloudlet 选择问题是一个双目标的组合优化问题[27]，且属于 NP-hard 问题。而化学反应优化算法是通过模拟分子运动这一自然现象得到的一种通用元启发式算法[28]，可以用于求解此类组合优化问题，并且该 Cloudlet 选择问题与网格计算中多目标任务调度和人工神经网络中参数优化问题相似，在文献[29-30]中已经用实验表明了化学反应优化算法相较于其他启发式搜索算法在求解该类问题上的性能优势。因此,本文采用基于化学反应优化的启发式算法求解。在化学反应优化算法中包括分子和基本化学反应两大因素，其中每个分子包含 3 个重要组成部分：分子结构、势能和动能。本文中分子结构对应着 Cloudlet 选择问题的解，势能决定分子结构的稳定程度，其在本文为 Cloudlet 选择策略的目标函数值，动能是系统中判断是否发生反应的量化值。基本化学反应包括单分子碰撞、单分子分解、分子间碰撞和分子合成 4 种，通过这 4 种化学反应操作使分子结构不断变得稳定的过程，同样对应着在问题求解空间内不断寻求更优 Cloudlet 选择策略的过程。

5.1　问题编码

针对本文移动数据流应用的 Cloudlet 选择问题，采用十进制编码方式对分子结构进行编码，分子中的原子对应于组件的选择位置，则其整个的分子结构对应于一个 Cloudlet 选择策略。例如对于图 1 中光学字符识别应用对应的一种分子结构 X={0,1,2,3,1,1,0}，表示组件 v_0 和 v_6 在移动设备上执行，组件 v_1、v_4 和 v_5 在 Cloudlet$_1$ 上执行，组件 v_2 在 Cloudlet$_2$ 上执行，组件 v_3 在 Cloudlet$_3$ 上执行。

5.2　适应度函数

本文 Cloudlet 选择策略的目标是最小化移动数据流应用的完成时间和移动设备能耗。这里通过简单的线性加权将 2 个目标归一为单目标，所以对于该问题的适应度函数 $f(X)$可以使用式(14)表示。

$$f(X)=\alpha\frac{f_1(X)}{f_1(\text{allinmobile})}+\beta\frac{f_2(X)}{f_2(\text{allinmobile})} \tag{14}$$

其中，函数 $f_1(X)$和 $f_2(X)$分别表示 Cloudlet 选择策略 X 对应的应用完成时间和移动设备能耗。f_1(allinmobile)表示应用程序的所有组件都在移动设备上执行的能耗，f_2(allinmobile)表示应用程序的所有组件都在移动设备上执行的完成时间。α 和 β 分别表示用户依据移动设备剩余电量对完成时间和移动设备能耗的权重要求，其中 α, $\beta\in(0,1)$且满足 $\alpha+\beta=1$。当移动设备剩余电量较为充足时，应用的完成时间作为主要优化目标，可以设置 $\alpha>\beta$；当剩余电量不足时，移动设备能耗作为主要优化目标，可以设置 $\alpha<\beta$。

5.3　分子操作设计

1) 单分子碰撞

单分子碰撞 ineff_coll(X, x_1)：是指单个分子碰撞容器壁改变其势能和动能的过程。分子的结构 X 发生轻微变化，产生新的分子结构 x_1，利用这一特点进行问题求解空间的领域搜索。

本文具体设计采用在原分子（原 Cloudlet 选择策略）X 中，随机选择一个原子，随机变换其值（组件的选择位置），产生新的分子（新 Cloudlet 选择策略）x_1。

2) 单分子分解

单分子分解 decompose(X, x_1, x_2)：是指动能较大的分子与容器壁发生碰撞后，分解成 2 个分子的过程。分子 X 发生单分子分解后，将产生 2 个新的分子 x_1 和 x_2，单分子分解反应过程中对分子结构的改变较大，用于提高算法的全局搜索能力。

本文具体设计采用原分子（原 Cloudlet 选择策略）X 随机产生一个分解点，新分子（新 Cloudlet 选择策略）x_1 获得原分子 X 分解点之前的所有原子（组件的选择位置），新分子（新 Cloudlet 选择策略）x_2 获得原分子 X 分解点之后的所有原子，其余部分随机产生。

3) 分子间碰撞

分子间碰撞 iter_ineff_cole(X_1, X_2, x_1, x_2)：是指 2 个分子发生碰撞后产生 2 个新分子的过程，即原分子 X_1 和 X_2 发生分子间碰撞之后产生新分子 x_1 和 x_2。分子间碰撞与单分子碰撞类似，反应剧烈程度都不大，对分子结构的改变较小，用于在邻域空间搜索问题的解。

本文具体设计采用类似遗传算法中单点交叉的方式，随机选择一个位置作为碰撞点，新分子（新 Cloudlet 选择策略）x_1 获得原分子（原 Cloudlet 选择策略）X_1 碰撞点之前的原子（组件的选择位置）和（原 Cloudlet 选择策略）X_2 碰撞点之后的原子，新分子（新 Cloudlet 选择策略）x_2 获得原分子 X_1 碰撞点之后的原子和原分子 X_2 碰撞点之前的原子。

4) 分子合成

分子合成反应 synthesis(X_1, X_2, x)：是指 2 个分子碰撞后合成一个新分子的过程，即原分子 X_1 和 X_2 发生分子合成碰撞后产生 x。由于分子合成反应对分子结构的改变很大，提高分子搜索新区域能力，用于增强算法全局搜索能力。

本文具体设计采用随机生成一个划分点，碰撞后的新分子（新 Cloudlet 选择策略）x 获得原分子（原 Cloudlet 选择策略）X_1 的划分点之前原子（组件的选择位置）和获得原分子（原 Cloudlet 选择策略）X_2 的划分点之后原子。

5.4　算法描述

算法 1 给出了基于化学反应优化算法的 Cloudlet 选择策略（CROCS），其对应的流程图如图 3 所示。对于给定的移动设备 mobile、Cloudlet*s*、应用程序 $G(V,E)$等信息，CROCS 算法会得到一个近似最优的选择策略 X。

该算法分为 3 个阶段：初始阶段、迭代过程和最后阶段。在初始阶段（算法 1 中 1)~11)），初始化系统参数并随机产生 PopSize 个选择策略分子组成的分子群，计算每一个分子的适应度值，初始化其势能和动能。随后进入迭代阶段（算法 1 中 12) ~ 27)），迭代阶段结束条件由初始设置的迭代次数

MaxIter 决定。在每一次迭代过程中，首先由在[0,1]之间的随机数 t 与系统参数 MoleColl 值的大小关系决定发生单分子反应还是发生分子间的反应。当随机数 t 大于系统参数 MoleColl，随机从分子群中选择一个分子，如果分解反应条件满足，则发生单分子分解反应，否则发生单分子碰撞反应。当随机数 t 小于系统参数 MoleColl，随机从分子群中选择 2 个分子，如果合成反应条件满足，则发生分子合成反应，否则发生分子间碰撞反应。在最后阶段（算法 1 中 28)~29)），从分子群中得到所含势能最小的分子作为最终的 Cloudlet 选择策略。

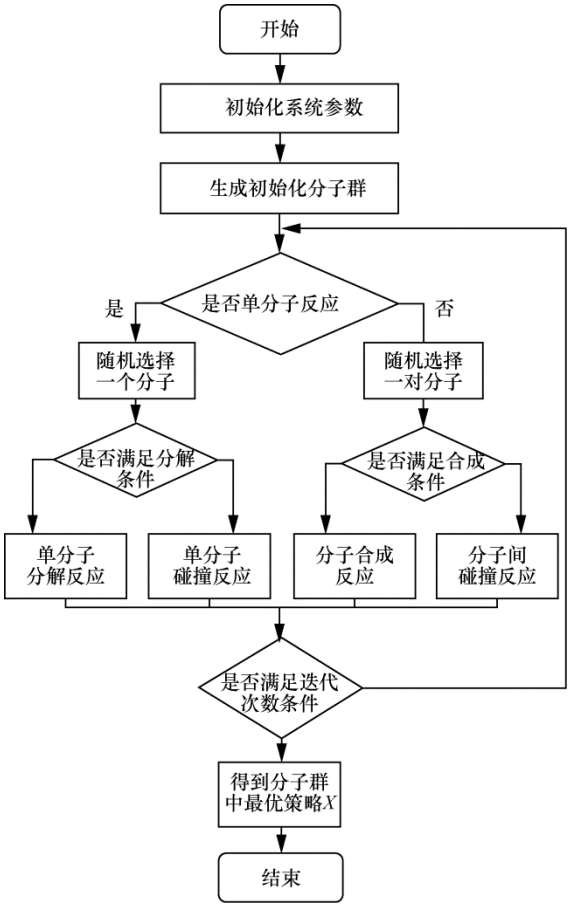

图 3 CROCS 算法流程

算法 1 基于化学反应优化算法的 Cloudlet 选择策略（CROCS）

输入 Mobile，Cloudlet，$G(V,E)$，迭代次数 MaxIter，权重参数 α 和 β

输出 选择策略 X

1) 初始化 PopSize, Molecules, MoleColl, InitialKE /*PopSize 为分子群初始规模，Molecules 为分子群, MoleColl 为分子反应决定类型因子，InitialKE 为初始动能*/

2) i←1

3) while i≤PopSize

4) 随机生成一个 Cloudlet 选择策略分子 X

5) $f_2(X)$←computfinishtime(X)/*依据式(5)计算选择策略分子 X 对应的应用完成时间*/

6) $f_1(X)$←computtotalenergy(X)/*依据式(11)计算选择策略分子 X 对应的移动设备能耗*/

7) PE[X]←computeFitness(X)/*依据式(14)计算选择策略分子 X 的适应度并初始化势能*/

8) KE[X]←InitialKE /*初始化策略分子 X 的动能*/

9) 将分子 X 加入到分子群 Molecules 中

10) i←i+1

11) end while

12) j←1

13) while j≤MaxIter /*迭代 MaxIter 次*/

14) t=random(0,1)

15) if (t > MoleColl)

16) 随机从分子群 Molecules 中选择一个分子 X

17) if (decomposition criterion met)then

18) decompose(X, x_1, x_2)/*发生单分子分解反应*/

19) else

20) ineff_coll(X , x_1) /*发生单分子碰撞反应*/

21) else

22) 随机从分子群 Molecules 中选择 2 个分子 X_1 和 X_2

23) if (synthesis criterion met) then

synthesis(X_1, X_2, x)/*发生分子合成反应*/

25) else

26) iter_ineff_cole(X_1, X_2, x_1, x_2)/*发生分子间碰撞反应*/

27) end while

28) X←findthebest(Molecules)/*从分子群中找到最优的策略*/

return X

5.5 解空间分析

本节将分析问题的解空间，并给出选择化学反

应算法求解该问题的原因。

1) 解空间大小

如上文模型所述，对于一个拥有 n 个组件的应用。除开始和结束组件外，其每一个组件的选择位置为 m 个 Cloudlet 或移动设备，即可选择位置个数为 m+1，故解空间的大小为$(m+1)^{n-2}$。

2) 可选算法分析

求解这类组合优化问题的算法可以分为确定性算法（如线性规划等）和启发式算法（如化学反应优化算法、模拟退火算法和遗传算法[31]等）。然而确定性算法在这种指数级的解空间大小下，将面临组合爆炸问题。在启发式算法中，化学反应优化算法能够通过 4 种基本化学反应实现启发式搜索。其中单分子碰撞和分子间碰撞反应作用于邻域搜索；分子合成和单分子分解反应作用于全局搜索。因而化学反应算法相较于其他启发式算法，能够避免过早地陷入局部最优。因此，在本文中采用化学反应优化算法求解该问题。

5.6 时间复杂度分析

假设初始分子群的规模为 PopSize，应用的组件个数为 n，迭代次数为 MaxIter。在初始阶段生成 PopSize 个分子的分子群，并初始化每一个分子的动能和势能，计算势能时需要计算一次适应度，而计算适应度函数的时间复杂度为 $O(n)$，故在初始阶段的时间复杂度为 O(PopSize×n)。在迭代阶段，单分子分解反应的时间复杂度为 $O(n)$，单分子碰撞反应的时间复杂度为 $O(1)$，分子合成反应的时间复杂度为 $O(n)$，分子间碰撞反应的时间复杂度为 $O(1)$，所以在 4 种分子操作过程中的平均复杂度为 $O(n)$，同时还需要计算新分子的适应度函数。迭代次数为 MaxIter，所以迭代过程中的时间复杂度为 O(MaxIter×n×n)。综上所述，整个算法的时间复杂度为 O((MaxIter×n+Pop*S*ize)×n)。

5.7 算法收敛性分析

化学反应算法收敛性主要取决于化学反应操作算子[32],经过基本的化学反应之后，令选择策略 X_1 到选择策略 X_2 的转换概率 $P(X_1 \to X_2)>0$, 选择策略 X_1 和 X_2 属于选择策略任意元素。由于每一次迭代的独立性，经过 MaxIter 次迭代过程后，算法不能从次优解达到全局最优解的概率是 $1-(1-p)^{\text{MaxIter}}$。因此算法经过 MaxIter 次迭代后能够达到最优解的概率是 $1-(1-p)^{\text{MaxIter}}$，当 MaxIter→∞时，可以得到最优解的概率为 $\lim\limits_{\text{MaxIter}\to\infty} 1-(1-p)^{\text{MaxIter}}=1$。所以当迭代次数足够多时，算法总会收敛于全局最优解。

6 仿真实验与结果分析

6.1 仿真实验环境配置

1) Cloudlet 和移动设备配置

实验中的 Cloudlet 由以下 3 台服务器模拟，配置信息如表 2 所示。使用 Cloudlet_1 作为代理 Cloudlet，Cloudlet_2、Cloudlet_3 作为其他计算服务器。其中 Cloudlet_2 计算能力最强，Cloudlet_3 其次，作为代理的 Cloudlet_1 计算能力最弱。

表 2　　Cloudlet 配置

Cloudlet	内存/GB	CPU
Cloudlet_1	16	Intel(R) Xeon(R) E5-2407 0@ 2.20 GHz
Cloudlet_2	32	Intel(R) Xeon(R)E5-2620v3@ 2.40 GHz
Cloudlet_3	32	Intel(R) Xeon(R)E5-2620v2@ 2.10 GHz

实验采用的移动设备为智能手机，其配置、CPU 和无线接口在不同状态下功率（单位是 mw）信息如表 3 所示，其中 CPU 功率和无线网络接口功率等信息是依据 Android 操作系统应用程序框架提供的能耗分析配置文件（PowerProfile.xml）获得。

表 3　　移动设备配置

移动设备	内存/GB	CPU	CPU 功率		Wi-Fi 功率	
			运行/mw	空闲/mw	发送/mw	接收/mw
MI 5	3	MSM8996@1.8 GHz	610	8	120	120

2) 网络配置

实验中的网络配置分为 2 个方面：Cloudlet 与移动设备之间的网络，移动设备的默认上传带宽 B^u 和下载带宽 B^d 设置为 8 Mbit/s；Cloudlet 之间的网络，本文 Cloudlet 之间的带宽 B 大小相同，同时 Cloudlet 选择策略的结果与 Cloudlet 之间的带宽大小有关，所以采用 4 种不同的 Cloudlet 之间的带宽，分别是 10 Mbit/s、30 Mbit/s、50 Mbit/s 和 210 Mbit/s。

3) 移动数据流应用配置

为了不失一般性，实验采用应用模型如图 1 所示的光学字符识别移动数据流应用程序进行实验。

6.2 实验结果分析

在本节中，将 CROCS 策略与已有的 2 种策略：H-PSP（heuristic based on partial stochastic path）[21]

和 POCSS（power and latency aware optimum Cloudlet selection strategy）[14]进行对比，其中 H-PSP 策略联合利用移动设备和单个 Cloudlet 的计算能力，并以最小化应用的完成时间为目标，其中单个 Cloudlet 选取计算能力最强的 Cloudlet 作为组件候选的执行位置；POCSS 策略是选择一个应用的完成时间最小或移动设备能耗最低的 Cloudlet，并将应用的所有可卸载的组件全都卸载到一个 Cloudlet 上进行执行。

为了验证本文算法有效性，首先，以应用的完成时间和移动设备能量消耗作为性能指标，采用控制变量法，分别改变 Cloudlet 之间的带宽、Cloudlet 个数、应用程序并行组件指令数在总指令数中的占比和参数 α 等影响因素，然后统计分析应用的完成时间和移动设备能耗随这些因素的变化规律。其次，通过引入已经广泛应用的遗传算法进行对比，考察其收敛特性。

1) Cloudlet 之间的带宽的影响

本组实验研究 Cloudlet 之间的带宽对应用的完成时间和移动设备能耗的影响，实验参数的配置如表 4 所示。

表 4　　实验 1 环境配置

实验环境	参数值
Cloudlet 个数/个	3
移动设备与 Cloudlet 之间带宽/（Mbit·s^{-1}）	8
并行组件指令数占比	40%
参数 α	0.5
Cloudlet 之间的带宽/（Mbit·s^{-1}）	10,30,50

本实验中设置 Cloudlet 个数为 3 个，分别为 $Cloudlet_1$、$Cloudlet_2$ 和 $Cloudlet_3$，移动设备与 Cloudlet 之间的带宽为 8 Mbit/s，并行组件指令数在总指令数中占比为初始的 40%，Cloudlet 之间的带宽分别取 10 Mbit/s、30 Mbit/s、50 Mbit/s。具体实验结果如图 4 所示。

如图 4(a)所示，随着 Cloudlet 之间带宽的增大，CROCS 策略和 POCSS 策略的应用完成时间随之减小，而 H-PSP 策略没有变化。CROCS 策略和 POCSS 策略都需要通过代理 Cloudlet 与其他 Cloudlet 进行交互，所以 Cloudlet 之间的带宽增加降低了组件间数据传输成本，减小了应用的完成时间。在 Cloudlet 之间的带宽较低的情况下，较高的数据传输时间成本使得 CROCS 策略没有发挥其充分利用应用的组件间并行性优势，使得 CROCS 策略在 Cloudlet 之间的带宽较小的条件下并没有优势。随着 Cloudlet 之间的带宽的增加，组件之间的传输时间成本降低，CROCS 策略可以充分利用应用的并行性，进而与其他 2 种策略相比在完成时间上有明显的优势,相较于 H-PSP 策略和 POCSS 策略在完成时间上平均有 3%和 14.8%的性能提升。

如图 4(b)所示，随着 Cloudlet 之间带宽的增大，CROCS 策略和 POCSS 策略的移动设备能耗都越来越小，且都处于较低的水平。由于两者都没有任何组件在移动设备上执行，移动设备能耗主要由数据传输能耗和空闲能耗组成，Cloudlet 之间的带宽增加使这两者策略的应用完成时间都有所降低（如图 4(a)所示），由此降低了一定的空闲能耗，但空闲能耗在总能耗中占比太低，对于移动设备能耗的降低并不明显。而 H-PSP 策略的移动设备能耗相较于其他 2 种策略较高，其原因是 H-PSP 策略会使用移动设备资源执行一定的并行组件以减小应用的完成时间，从而带来了较高的计算能耗。综合来看，相较于 H-PSP 策略和 POCSS 策略在移动设备的能耗上平均有 70.5%和 6%的能耗降低。

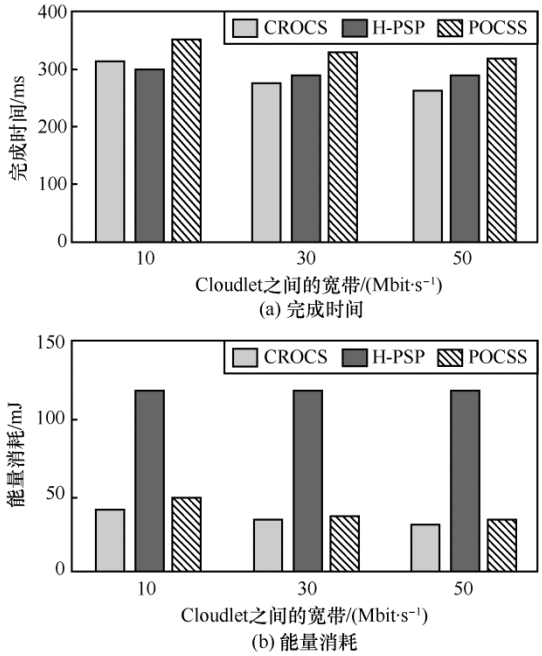

图 4　Cloudlet 之间的带宽对应用完成时间和移动设备能耗的影响

2) Cloudlet 个数的影响

本组实验研究 Cloudlet 个数对应用完成时间和移动设备能耗的影响，实验参数的配置如表 5 所示。

表 5　　实验 2 环境配置

实验环境	参数值
Cloudlet 之间的带宽/(Mbit·s^{-1})	210
移动设备与 Cloudlet 之间的带宽/(Mbit·s^{-1})	8
并行组件指令数占比	40%
参数 α	0.5
Cloudlet 个数/个	1,2,3,4

本实验中，为了减小 Cloudlet 之间的带宽对本实验的影响，将其设置为 210 Mbit/s，移动设备与 Cloudlet 之间的带宽为 8 Mbit/s，并行组件指令数在总指令数中占比为 40%，Cloudlet 个数依次为 1 个、2 个、3 个和 4 个，当 Cloudlet 个数为超过 3 个时，超过部分采用 $Cloudlet_3$ 模拟。具体实验结果如图 5 所示。

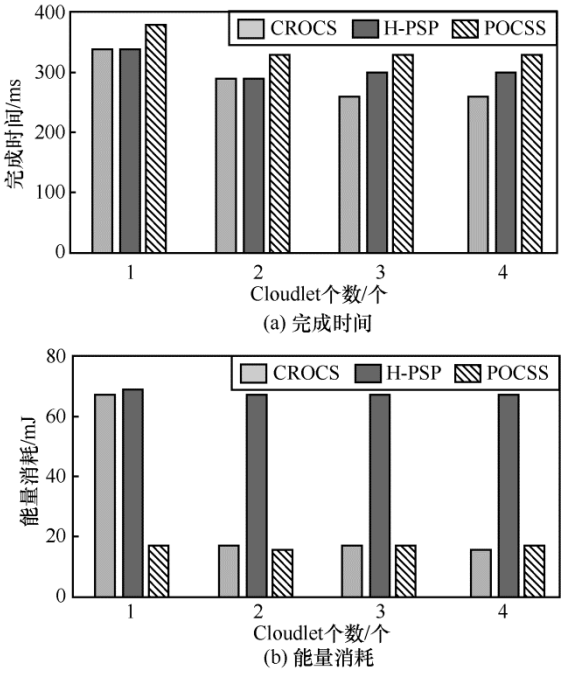

图 5　Cloudlet 个数对应用完成时间和移动设备能耗的影响

如图 5(a)所示，随着 Cloudlet 个数的增加，CROCS 策略的应用完成时间逐渐减小。当 Cloudlet 个数从一个增长到 3 个时，主要是由于 CROCS 策略会使应用中更多的并行组件能够分散到多个 Cloudlet 上并行执行，从而降低了应用的完成时间，而当 Cloudlet 个数与光学字符识别并行组件达到相同的 3 个时，应用的并行程度达到最大，此时进一步增加 Cloudlet 规模，应用完成时间的降低是通过引入了计算能力较强的 Cloudlet 加快应用效率实现的。对于 H-PSP 策略和 POCSS 策略，Cloudlet 个数从一个增加到 2 个的时候，完成时间有所降低，其后则保持不变。原因是引入了计算能力更强的 $Cloudlet_2$，H-PSP 策略和 POCSS 策略会主要使用计算能力更强的 $Cloudlet_2$ 进行计算卸载，从而降低了应用的完成时间，当 Cloudlet 个数进一步增加，这 2 种策略仍然只使用该计算能力较强的 $Cloudlet_2$，所以并不会影响应用的完成时间。综合来看，相较于 H-PSP 策略和 POCSS 策略，CROCS 策略在完成时间上平均有 4.5%和 12%的性能提升，最好情况下分别能达到 12.1%和 22.3%。

如图 5(b)所示，随着 Cloudlet 个数的增加，CROCS 策略的移动设备能耗也随着降低。当 Cloudlet 个数从一个增加到 2 个的时候，CROCS 策略的移动设备能耗显著降低，这是由于当 Cloudlet 个数只有一个 Cloudlet 时，CROCS 策略会利用移动设备的计算能力，将部分组件留在移动设备上执行，会产生较高的移动设备计算能耗。随着 Cloudlet 个数进一步增加，应用的并行组件能够分散到其他 Cloudlet 上执行，从而显著降低了移动设备的能耗。H-PSP 策略在 Cloudlet 个数较多的时候依然需要利用移动设备的计算能力，从而会一直产生较高的移动设备能耗。相比之下，CROCS 策略的移动设备能耗在 Cloudlet 个数足够多时相较于 H-PSP 和 POCSS 策略分别减小了 77.1%和 5.9%。

3) 应用程序并行组件指令数在总指令数中占比的影响

本组实验为了进一步验证 CROCS 策略对于组件之间可并行部分充分利用的特性，在保持应用的总指令数不变前提下，将光学字符识别中的 3 个并行组件指令数与全部组件的指令数的比例逐步提高，探究应用的并行性对选择策略的影响。具体实验配置如表 6 所示。

表 6　　实验 3 环境配置

实验环境	参数值
Cloudlet 之间的带宽/（Mbit·s^{-1}）	210
移动设备与 Cloudlet 之间的带宽/（Mbit·s^{-1}）	8
Cloudlet 个数/个	3
参数 α	0.5
并行组件指令数占比	40%,60%,80%

本组实验中，Cloudlet 之间带宽设置为 210 Mbit/s，移动设备与 Cloudlet 之间的带宽为 8 Mbit/s，Cloudlet 个数为 3 个，应用中并行组件指令数占比为 40%、60%、

80%。具体实验结果如图 6(a)和图 6(b)所示。

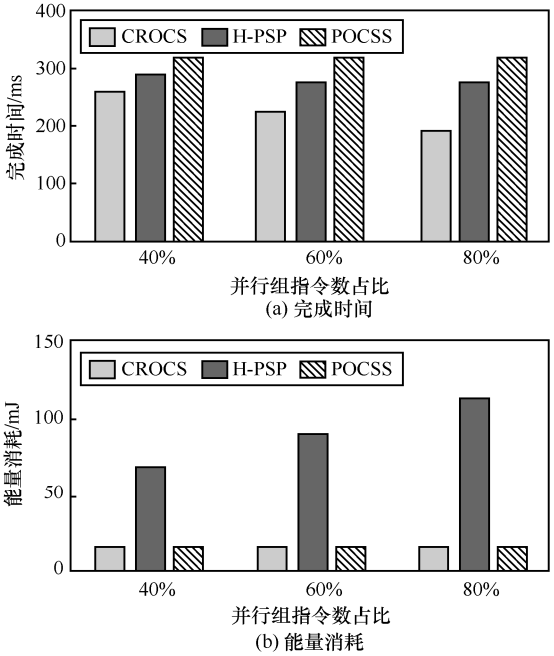

图 6 并行组件指令数占比对应用完成时间和移动设备能耗的影响

如图 6(a)所示，随着并行组件指令数占比越来越高，CROCS 策略和 H-PSP 策略的应用完成时间会相应减小，而 POCSS 策略没有变化。主要是因为在 CROCS 策略和 H-PSP 策略中，可以并行执行组件的指令数增多，同时执行的效率更高，加快了应用的执行速度，从而降低了应用的完成时间。但 CROCS 策略相较于 H-PSP 策略，其应用完成时间减小的幅度更为明显，这是由于 CROCS 策略对于并行组件的并行程度利用得更加充分。而 POCSS 策略中应用程序的所有组件都只在同一个 Cloudlet 上执行，并没有考虑到组件间的并行性，所以并行组件指令数占比对其没有影响。综合来看，随着并行组件的指令数占比的增加，CROCS 策略的应用完成时间相较于其他 2 种策略优势更加明显。具体来说，相较于 H-PSP 策略和 POCSS 策略在完成时间上有 19.3%和 28.2%的性能提升。

如图 6(b)所示，随着并行组件指令数占比越来越高，CROCS 策略的移动设备能耗相较于 H-PSP 策略优势越来越明显。由于 H-PSP 策略中需要在移动设备上执行的指令数的增加，导致移动设备 CPU 处于计算状态的时间相应增长，且 CPU 计算功率相较于网络接口的数据传输功率和 CPU 空闲功率较高，从而显著增加了移动设备能耗，所以相较于 H-PSP 策略，CROCS 策略在移动设备能耗上有较大优势。其次，CROCS 策略和 POCSS 策略的所有组件都是在 Cloudlet 上执行，其移动设备能耗主要只由两部分组成，网络接口的数据传输能耗和 CPU 空闲能耗。虽然 CROCS 策略的应用完成时间降低了不少（如图 6(a)所示），但是 CPU 空闲功率太小，并没有大幅降低移动设备能耗。所以 CROCS 策略和 POCSS 策略相比，移动设备能耗相差不大。综合来看，相较于 H-PSP 策略和 POCSS 策略在移动设备的能耗上平均有 83.7%和 4%的能耗减小。

4) 移动设备与 Cloudlet 之间带宽的影响

本组实验为了验证移动设备与 Cloudlet 之间的带宽对应用完成时间和移动设备能耗的影响，实验参数配置如表 7 所示。

表 7 实验 4 环境配置

实验环境	参数值
Cloudlet 之间的带宽/（Mbit·s^{-1}）	210
Cloudlet 个数/个	3
参数 α	0.5
并行组件指令数占比	40%
移动设备与 Cloudlet 之间的带宽/（Mbit·s^{-1}）	4,8,12

本实验中设置 Cloudlet 之间的带宽为 210 Mbit/s，Cloudlet 个数为 3，参数 α 为 0.5，并行组件指令数占比为 40%，移动设备与 Cloudlet 之间的带宽分别设置为 4 Mbit/s、8 Mbit/ss 和 12 Mbit/s。具体实验结果如图 7 所示。

如图 7(a)所示，随着移动设备与 Cloudlet 之间的带宽增大，CROCS、H-PSP 和 POCSS 策略的应用完成时间都随之降低，这是由于随着移动设备与 Cloudlet 之间带宽的增大提高了移动设备与 Cloudlet 端的数据传输效率，从而减小了应用完成时间。与此同时，CROCS 策略的应用完成时间相较于 H-PSP 和 CROCS 策略在完成时间相对较小，这是因为 CROCS 策略能够更加充分的利用多个 Cloudlet 计算能力加快应用执行效率，进而 CROCS 策略相较于其他 2 种策略在完成时间较优。

图 7(b)展示了不同移动设备与 Cloudlet 之间的带宽对移动能耗的影响。从中可以看出随着移动设备与 Cloudlet 之间带宽的增大，3 种选择策略的移动设备能耗都相应降低。这是由于移动设备与 Cloudlet 之间的带宽的增加导致移动设备的网络接口处于数据传输时间降低，从而移动设备的数据传输能耗随之降低，并最终降低了移动设备能量消耗。

同时还可以看出 CROCS 和 POCSS 策略相较于 H-PSP 策略，移动设备能耗处于较低水平，这是由于 H-PSP 策略优化目标主要是应用的完成时间，其对移动设备计算能力的使用产生了较多能耗。

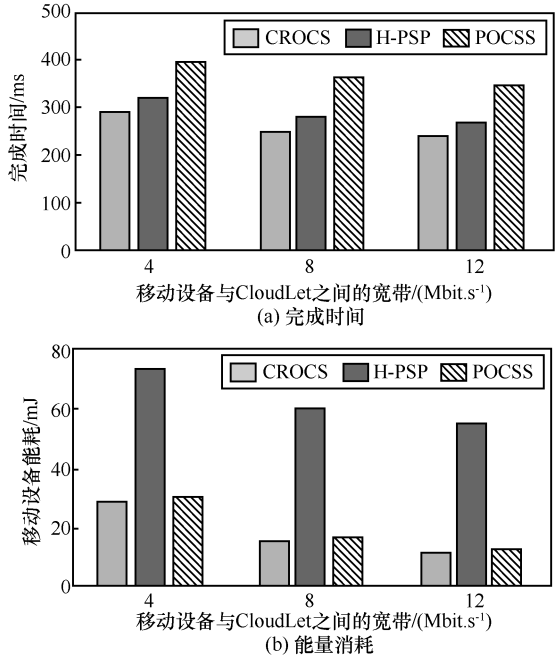

图 7　移动设备与 Cloudlet 之间的距离对应用完成时间和移动设备能耗的影响

5) 参数 α 的影响

本组实验为了探究参数 α 对应用完成时间和移动设备能耗的影响，实验参数的配置如表 8 所示。

表 8　　实验 5 环境配置

实验环境	参数值
Cloudlet 之间的带宽/（Mbit·s^{-1}）	210
移动设备与 Cloudlet 之间的带宽/（Mbit·s^{-1}）	8
并行组件指令数占比	40%
Cloudlet 个数	2
参数 α	0.1,0.5,0.9

本组实验中，Cloudlet 之间带宽设置为 210 Mbit/s，移动设备与 Cloudlet 之间的带宽为 8 Mbit/s，Cloudlet 个数设置为 2 个，应用中并行组件指令数占比为 40%，参数 α 分别设置为 0.1、0.5 和 0.9。具体实验结果如图 8 所示。

图 8(a)表示参数 α 对应用的完成时间的影响。随着参数 α 的增大，本文提出的 CROCS 策略的应用完成时间相应变小。这是由于参数 α 的增大表示主要优化目标越来越偏向于应用的完成时间，从而导致 CROCS 策略会选择使用移动设备的计算资源执行部分并行组件，减小了应用的完成时间。而 H-PSP 和 POCSS 策略不受参数 α 的影响，所以其应用的完成时间没有变化。

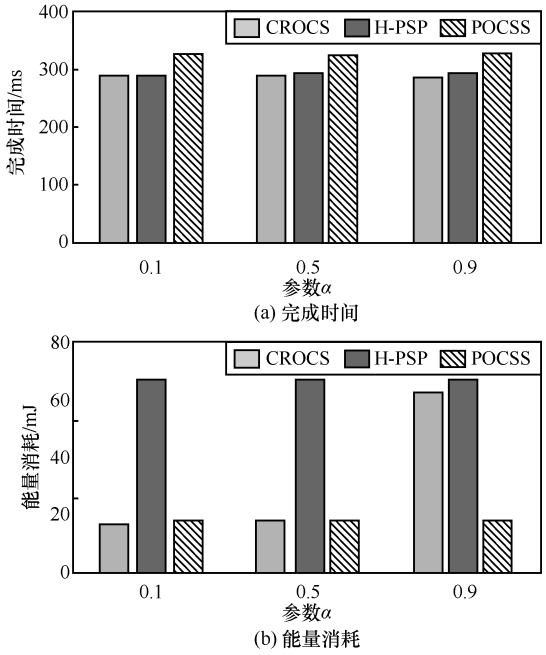

图 8　参数 α 对应用完成时间和移动设备能耗的影响

图 8(b)表示参数 α 对移动设备能耗的影响。随着参数 α 的增大， CROCS 策略的移动设备能耗随之增大。这是由于参数 α 的增大导致 CROCS 策略会选择使用移动设备的计算资源执行部分并行组件减小应用完成时间，从而移动设备将产生较大的计算能耗，因此移动设备的能耗相应增加。同样地，参数 α 对 POCSS 和 H-PSP 策略没有影响，所以其移动设备能耗没有变化。综上所述，实验结果表明 CROCS 策略可以通过调整应用的完成时间和移动设备能耗的权重参数 α 动态地给出选择策略，达到了预期的效果。

6) 收敛特性

为了考察化学反应优化算法在求解此类组合优化问题时的收敛特性，选取已经被广泛应用的遗传算法（GA，genetic algorithm）和离散粒子群优化算法（DPSO，discrete particle swarm optimization）[33] 与之进行对比。

其中 CROCS 参数设置为：分子群规模 PopSize 为 40，分子反应决定因子 MoleColl 为 0.2，初始动能 InitialKE 为 500，迭代次数 MaxIter 为 500。GA 参数设置为：种群规模为 40，变异概率为 0.02，交

叉概率为 0.8，迭代次数为 500。DPSO 的参数设置为：粒子群规模为 40，惯性常数为 0.8，加速度系数为 0.5，迭代次数为 500。分别独立运行 30 次，CROCS 与 GA、DPSO 算法运行的结果如表 9 所示，收敛曲线如图 9 所示。

表 9　CROCS 与 GA、DPSO 运行结果

算法	寻找到全局最优解的概率	平均适应度	最低适应度	最大适应度	标准差
GA	13.33%	0.466 9	0.455 666	0.527 732	0.163 5
DPSO	43.33%	0.458 7	0.455 666	0.465 812	0.003 2
CROCS	60%	0.452 7	0.455 666	0. 462 26	0.002 4

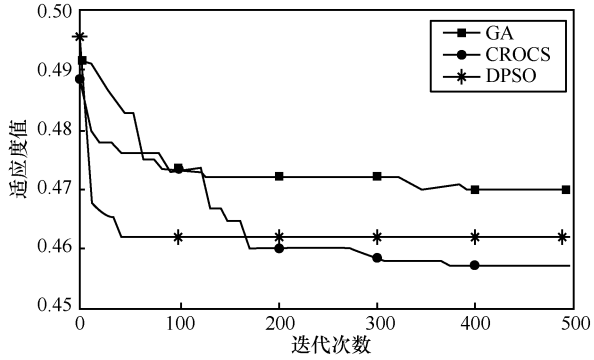

图 9　收敛曲线

如表 9 所示，CROCS 相较于 GA 和 DPSO 策略得到全局最优解的概率更高。其次，对应的平均适应度和最大适应度值也相对于 GA 和 DPSO 策略更优，并且从标准差看 CROCS 策略有更高的求解稳定性。因此，CROCS 相较于 GA 和 DPSO 策略有更好的全局收敛性。

图 9 为 CROCS 与 GA、DPSO 策略的收敛曲线。CROCS 的收敛曲线相较于 GA 策略更接近与底线，并且收敛的速度相对较快；而 DPSO 虽然在计算初期收敛速度较快，但很快陷入局部最优解，无法跳出。因此，CORCS 策略可以更快地跳出局部最优解，而 GA 和 DPSO 更易于陷入局部最优，无法找到全局最优解。主要因为化学反应优化算法中单分子分解和分子合成的分子操作对分子结构改变较大，具有很强的全局搜索能力。综上所述，CROCS 相较于 GA 策略的收敛速度更快且全局收敛性更好，相较于 DPSO 策略的全局收敛性更好。

综合以上 6 组实验，本文提出的 CROCS 策略与 H-PSP 策略和 POCSS 策略在应用性能上平均分别提升了 8.9%和 18.2%，在移动设备能耗方面则平均分别减少了 77.1%和 5.3%，并且在收敛特性上也有较优的效果。

7　结束语

本文针对在多 Cloudlet 环境中移动数据流应用的 Cloudlet 选择问题，提出了一种基于化学反应优化算法的 Cloudlet 选择策略 CROCS，该策略通过充分利用多个 Cloudlet 资源最大化移动数据流应用的并行执行效率，在提升移动数据流应用性能的同时兼顾移动设备的能量消耗。仿真实验表明，本文提出的选择策略相较于其他没有充分考虑移动数据流应用中拥有大量可以并行执行的组件的策略，在应用的完成时间和移动设备的能量消耗上有较为明显的优势。在未来工作方面，考虑从用户的移动性方面展开研究。用户的移动会导致用户周围的网络和可用的 Cloudlet 资源发生变化，需要实时基于周围环境选择合适的 Cloudlet 才能满足应用的服务质量需求。另外，本文仿真实验环境设置与实际环境还有一定的差距，未来考虑更加贴近实际运行环境的因素，探究其对算法性能的影响。

参考文献：

[1] ITU-T. ICT facts and figures 2017[R]. International Telecommunication Union, 2017.

[2] SHAHZAD M,LIU A X,SAMUEL A. Secure unlocking of mobile touch screen devices by simple gestures:you can see it but you can not do it[C]//International Conference on Mobile Computing & NETWORKING. 2013:39-50.

[3] AKARIMAN Q, JATI A N, NOVIANTY A. Face recognition based on the Android device using LBP algorithm[C]//International Conference on Control, Electronics, Renewable Energy and Communications. 2015:166-170.

[4] 曹洋, 江涛, 杨世永, 等. 移动云计算网络中的最优资源分配研究[J]. 通信学报, 2011, 32(9A):42-48.
CAO Y, JIANG T, YANG S Y, et al. Optimal resource allocation in mobile cloud computing network[J]. Journal on Communications, 2011, 32(9A):42-48.

[5] 崔勇, 宋健, 缪葱葱, 等. 移动云计算研究进展与趋势[J]. 计算机学报, 2017, 40(2):273-295.
CUI Y, SONG J, MIAO C C, et al. Mobile cloud computing research progress and trends[J]. Chinese Journal of Computers, 2017, 40(2): 273-295.

[6] CUERVO E, BALASUBRAMANLAN A, Cho D, et al. MAUI: making smartphones last longer with code offload[C]//Proceedings of the 8th international conference on Mobile systems, applications, and services. 2010: 49-62.

[7] CHUN B G, IHM S, MANIATIS P, et al. Clonecloud: elastic execution between mobile device and cloud[C]//Proceedings of the sixth conference on Computer systems. 2011: 301-314.

[8] RA M R, SHETH A, Mummert L, et al. Odessa: enabling interactive

perception applications on mobile devices[C]//Proceedings of the 9th international conference on Mobile systems, applications, and services. 2011: 43-56.

[9] 张文丽, 郭兵, 沈艳, 等. 智能移动终端计算迁移研究[J]. 计算机学报, 2016, 39(5):1021-1038.
ZHANG W L, GUO B, SHEN Y, et al. Computation offloading on intelligent mobile terminal[J]. Chinese Journal of Computers, 2016, 39(5): 1021-1038.

[10] SATYANARAYANAN M, BAHL P, CACERES R, et al. The case for VM-based Cloudlets in mobile computing[J]. IEEE Pervasive Computing, 2009, 8(4):14-23.

[11] 华夏进, 董瑞志, 彭鑫, 等. 基于统计预测的 Cloudlet 调度机制研究[J]. 小型微型计算机系统, 2016, 37(3):406-411.
HUA X J, DONG R Z, PENG X, et al. Cloudlet scheduling mechanism research based on the statistical forecasting[J]. Journal of Chinese Mini-Micro Computer Systems, 2016, 37(3):406-411.

[12] YANG L, CAO J, TANG S, et al. Run time application repartitioning in dynamic mobile cloud environments[J]. IEEE Transactions on Cloud Computing, 2016, 4(3):336-348.

[13] PILLAI P S, MUMMERT L B, SCHLOSSER S W, et al. SLIPstream: scalable low-latency interactive perception on streaming data[C]// International Workshop on Network and Operating Systems Support for Digital Audio and Video. 2009:43-48.

[14] MUKHERJEE A, DE D, ROY D. A power and latency aware Cloudlet selection strategy for multi-Cloudlet environment[J]. IEEE Transactions on Cloud Computing, 2016: 1.

[15] PARMAR D, KUMAR A S, NIVANGUNE A, et al. Discovery and selection mechanism of Cloudlets in a decentralized MCC environment[C]//IEEE/ACM International Conference on Mobile Software Engineering and Systems. 2016:15-16.

[16] TAWALBEH L, JARARWEH Y, ABABNEH F, et al. Large scale Cloudlets deployment for efficient mobile cloud computing[J]. Journal of Networks, 2015, 10(1):70-76.

[17] SAAD H B, KASSAR M, SETHOM K. Utility-based Cloudlet selection in mobile cloud computing[C]//2016 Global Summit on Computer & Information Technology, 2016: 91-96.

[18] CHILUKURI S, BOLLAPRAGADA S, KOMMINENI S, et al. Rain-Cloud-Cloudlet selection for effective cyber foraging[C]//Wireless Communications and NETWORKING Conference. 2017.

[19] GAI K, QIU M, ZHAO H, et al. Dynamic energy-aware Cloudlet-based mobile cloud computing model for green computing[J]. Journal of Network & Computer Applications, 2016, 59(C):46-54.

[20] ROY D G, DE D, MUKHERJEE A, et al. Application-aware Cloudlet selection for computation offloading in multi-Cloudlet environment[J]. Journal of Supercomputing, 2017, 73(4):1-19.

[21] SHU G, ZHENG X, XU H, et al. Cloudlet-assisted heuristic offloading for mobile interactive applications[C]//IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. 2017:66-73.

[22] LIU W, CAO J, QIU X, et al. Improving performance of mobile interactive data-streaming applications with multiple Cloudlets[C]//IEEE International Conference on Cloud Computing Technology and Science. IEEE Computer Society, 2014:46-53.

[23] RAVI A, PEDDOJU S K. Mobility managed energy efficient android mobile devices using Cloudlet[C]//Students' Technology Symposium. 2014:402-407.

[24] PERRUCCI G P, FITZEK F H P, WIDMER J. Survey on energy consumption entities on the smartphone platform[C]//Vehicular Technology Conference. 2011:1-6.

[25] 蒋廷耀, 李庆华. DAG 任务图的一种调度算法[J]. 小型微型计算机系统, 2003, 24(10):1796-1799.
JIANG T Y, LI Q H. A scheduling algorithm for dag task graphs[J]. Journal of Chinese Mini-Micro Computer Systems, 2003, 24(10): 1796-1799.

[26] GAREY M R, JOHNSON D S. Computers and intractability: a guide to the theory of NP-completeness[M]. W. H. Freeman, 1986.

[27] 陈乃金, 江建慧. 融合面积估算和多目标优化的硬件任务划分算法[J]. 通信学报, 2013(2):40-55.
CHEN N J, JIANG J H. Hardware-task partitioning algorithm merged area estimation with multi-objective optimization [J]. Journal on Communications, 2013(2):40-55.

[28] LAM A Y S, LI V O K. Chemical-reaction-inspired metaheuristic for optimization[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(3):381-399.

[29] XU J, LAM A Y S, LI V O K. Chemical reaction optimization for task scheduling in grid computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(10):1624-1631.

[30] JAMES J Q, LAM A Y S, LI V O K. Evolutionary artificial neural network based on chemical reaction optimization[C]// IEEE Congress on Evolutionary Computation. 2011: 2083-2090.

[31] 刘全, 王晓燕, 傅启明, 等. 双精英协同进化遗传算法[J]. 软件学报, 2012, 23(4):765-775.
LIU Q, WANG X Y, FU Q M, et al. Double elite coevolutionary genetic algorithm[J]. Journal of Software, 2012, 23(4):765-775.

[32] LAM A Y S, LI V O K, XU J. On the convergence of chemical reaction optimization for combinatorial optimization[J]. IEEE Transactions on Evolutionary Computation, 2013, 17(5): 605-62

[33] PAN Q K, TASGETIREN M F, LIANG Y C. A discrete particle swarm optimization algorithm for the permutation flowshop sequencing problem with makespan criterion[M]. London: Springer, 2007: 19-31.

[作者简介]

刘伟（1978− ），男，湖北襄阳人，博士，武汉理工大学副教授，主要研究方向为云计算、边缘计算、绿色计算。

熊曙（1994− ），男，湖北黄冈人，武汉理工大学硕士生，主要研究方向为云计算。

杜薇（1978− ），女，湖北武汉人，博士，武汉理工大学副教授，主要研究方向为云计算、边缘计算。

王伟（1979− ），男，湖北武汉人，博士，同济大学副教授，主要研究方向为云计算、大数据。