第 36 卷 第 4 期
2021 年 8 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol. 36 No. 4
Aug. 2021

DOI:10.13364/j.issn.1672-6510.20200215

一种基于改进二进制粒子群的工作流云调度算法

熊聪聪，高 萌，赵 青，徐丹滢
（天津科技大学人工智能学院，天津 300457）

摘 要：针对批处理科学工作流这一类应用，解决云环境中任务分配问题，从而有效降低成本，提高资源利用率，提出了一种改进的二进制粒子群算法. 尽管传统的二进制粒子群算法具有很强的全局探测能力，但难以收敛于全局最优位置，而且随着迭代次数的不断增加，后期的搜索能力差. 本文对粒子的更新公式进行修改，改善原始二进制粒子群算法的收敛性，提高了最优解的探测能力. 实验结果表明，该算法所得最优解具有更好的实际调度时间和更少的资源租赁成本.

关键词：云计算；任务调度；二进制粒子群算法；批处理科学工作流
中图分类号：TP311 **文献标志码**：A **文章编号**：1672-6510(2021)04-0061-06

A Workflow Cloud Scheduling Algorithm Based on Improved Binary Particle Swarm Optimization

XIONG Congcong，GAO Meng，ZHAO Qing，XU Danying
(College of Artificial Intelligence, Tianjin University of Science & Technology, Tianjin 300457, China)

Abstract: An improved binary particle swarm optimization algorithm is proposed in this article to solve the problem of task allocation in cloud environment so as to effectively reduce cost and improve resource utilization. Although the traditional binary particle swarm optimization algorithm has strong global detection capability, it is difficult to converge to the global optimal location. Moreover, with the increase of iteration times, the search capability in the later stage is poor. In the article, the particle updating formula was modified to improve the convergence of the original binary particle swarm optimization algorithm and improve the detection ability of the optimal solution. The experimental results showed that the optimal solution obtained by the algorithm had better actual scheduling time and less resource leasing cost.

Key words: cloud computing; tasks scheduling; binary particle swarm optimization algorithm; batch science workflow

随着数据规模日渐庞大、计算日趋复杂，科学工作流已经不能满足数据处理的需求，批处理科学工作流应运而生. 批处理科学工作流较科学工作流运算更为复杂，数据处理能力更强，它是包含大量批处理任务组的科学工作流. 批处理科学工作流广泛应用于高能物理学、气象学、生物信息学等不同领域，是建模和管理数据密集型数据的有效手段.

云计算为批处理科学工作流的处理提供了技术手段支持. 云计算在 2006 年由搜索引擎服务提供商 Google 率先提出[1]. 目前，云计算的用户数量已经超过 40 亿. 云计算是依靠虚拟技术进行使用，按需访问是云计算的特点. 数据密集型应用是云计算环境下最普遍的应用之一，数据通信往往是数据密集型应用的性能瓶颈. 面对庞大的数据规模，怎样对其进行合理的任务分配，使得云计算能够高效完成任务和给出合理调度方案显得尤为重要.

作为云计算的核心技术，任务调度是云计算处理任务过程中的重要环节之一，因此优化任务调度机制是提高云计算综合性能的重要方法[2]. 在云环境中，任务调度是一个非确定性多项式问题(NP 难题)的优

收稿日期：2020-12-14；**修回日期**：2021-04-14
基金项目：国家自然科学基金青年项目(11803022)；天津市科委应用基础与前沿技术研究计划项目(18JCQNJC69800)
作者简介：熊聪聪（1961—），女，四川泸州人，教授；**通信作者**：赵 青，副教授，zhaoqing@tust.edu.cn

化问题[3]. 任务调度可以合理地分配计算资源任务，并且对计算的任务进行高效率的调度，主要目标是将不同的用户任务分配合适的资源，例如主机或虚拟机(VM)，并找出可以执行任务的适当顺序在分配资源中执行. 任务调度促进了云数据中心的可持续发展，高效率的任务调度策略是非常必要的.

近年来，随着应用领域的不断扩大，批处理科学工作流的调度问题引起了国内外学术界的重视，Cai 等[4]提出的方法只考虑了批量任务组的整体同步调度，没有考虑批处理数据组中任务的单独调度问题. 此后，Cai 等[5]又对该方法进行改进，提出了一种基于多规则调度的算法，通过提高剩余时间间隙的利用率以最小化租赁成本，可以用于批处理数据组中单个任务的调度，考虑到了任务转移、浪费预测，但没有考虑任务间数据的传输成本增加的问题. 本文希望考虑批处理工作量中子任务的调度问题以及数据密集型任务中最为关键的数据传输成本的变化，将采用元启发式算法加以解决. 这是因为元启发式算法通常可以在有限的时间里高效、智能地探索搜索空间，以找到接近最佳的解决方案[6-7]. 而粒子群算法是近年来受到广泛关注的一种元启发式算法，具有效率高、收敛好的特点.

粒子群算法是美国的两位学者 Eberhart 和 Kennedy 源于对鸟群捕食行为的启发提出的一种元启发式算法，主要用来解决寻找最优解问题. Rodriguez 等[8]提出基于粒子群算法的动态时间片资源分配方法，虽然为了避免调度失败，采用了最大化任务执行时间假设的方法，但缺乏对任务间的数据依赖的考虑. 马亮等[9]提出一种改进的粒子群调度算法，以任务的完成时间为出发点进行考虑，对算法进行变异和修改，改进后的算法具有很好的性能，但着重考虑了任务的完成时间，未对其他影响因素进行考量. Saeedi 等[10]提出一种改进的多目标粒子群优化算法求解工作流调度问题，该算法考虑了 4 个影响因素，但未对粒子群算法的收敛性进行优化.

粒子群算法一般用于连续空间上的求解问题，而二进制粒子群算法更适合于求解 0/1 离散空间上的求解问题，这与任务的调度模型非常相似. 但已有研究成果显示，二进制粒子群算法存在收敛性不好的问题[11-12]，所以本文对粒子的更新公式进行了修改. 随着速度的逐渐降低(趋近于 0)，发生跳转的概率增大，改进的算法有效避免了这个问题. 实验结果表明，该算法提高了最优解的探测能力，所得最优解具有更好的实际调度时间和成本，提高了资源的利用率. 在相同的条件设置下，该算法优于未经改进的二进制粒子群算法，当迭代次数增多时，其综合调度性能优点明显.

1　批处理科学工作流任务调度模型描述

工作流应用程序被建模为 DAG 图，即 $E=\{T, U\}$，其中 $T=\{t_1, t_2, \cdots, t_n\}$ 表示所有任务的集，任务间的偏序关系集合表示为 $U=\left\{\left(v_i, v_j\right) \mid i<j\right\}$，$\left(v_i, v_j\right)$ 表示先执行任务 v_i，然后再执行任务 v_j. 图 1 为批处理科学工作流 DAG 图.

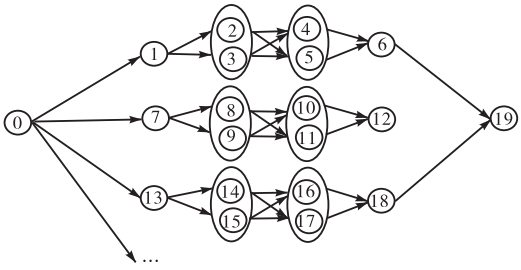

图 1　批处理科学工作流 DAG 图

Fig. 1　DAG diagram of batch scientific workflow

在云环境中进行任务调度，假设任务集合为 $T=\{t_1, t_2, \cdots, t_n\}$，虚拟机的集合为 $VM=\{vm_1, vm_2, \cdots, vm_m\}$，有 n 个相互独立的子任务将它们分配给 m 个虚拟机执行，其中 $m<n$. 第 j 个子任务用 $t_j=(j=1, 2, \cdots, n\}$ 表示，第 i 个虚拟机用 $vm_i(i=1, 2, \cdots, m)$ 表示. 每个任务只能由一个虚拟机运行. 任务集合与虚拟机集合的调度关系用矩阵 $\boldsymbol{X}$ 表示为

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{1}$$

其中 $x_{ij} \in \{0,1\}$，$\sum_{i=1}^{m} x_{ij}=1$，$i \in 1, 2, \cdots, n\}$，$j \in 1, 2, \cdots, n\}$. 如果子任务 t_j 在资源 vm_i 上执行，则 $x_{ij}=1$，否则 $x_{ij}=0$.

参考给出的矩阵 $\boldsymbol{X}$，每一列代表任务分配，每一行代表分配给虚拟机的任务. 在每一列中，数值 1 表示虚拟机被分配给一个任务时，每个任务只能由一个虚拟机来执行，数值 0 表示没有被分配任务. 例如，表 1 为 3 个虚拟机上有 5 个任务的调度方案. 与位置矩阵相似，每个粒子速度也以 $m \times n$ 矩阵的形式表示，其元素的范围为[0, 1]. 每行都只有一个 1，表示每个任务只能分配给一个虚拟机进行计算，但一个虚

拟机可能不只计算一个任务.

表 1　调度方案图示例

Tab. 1　Sample scheduling scheme diagram

	vm_0	vm_1	vm_2
t_0	1	0	0
t_1	0	1	0
t_2	0	0	1
t_3	1	0	0
t_4	0	1	0

2　改进二进制粒子群的批处理科学工作流任务调度算法

2.1　生成初始种群

初始种群的生成关系到算法能否快速找到最优解，确定合理的初始种群对二进制粒子群算法至关重要. 本文的初始种群采用计算机随机生成，它的优点是具有充分的随机性，分布比较均匀，同时也保证了粒子的丰富性.

2.2　适应度函数

处理优化问题的关键在于构造出合理的适应度函数，它是判断一个解好坏的标准. 适应度函数的选取对二进制粒子群算法的效率尤为重要. 本文以时间和成本为优化目标.

当任务节点为 y，虚拟机使用类型为 i 类时，j 个虚拟机执行任务需要的时间为

$$t_y=\frac{\mathrm{etc}_{yi}-t_b}{j}+t_b \tag{2}$$

其中：etc_{yi} 表示当虚拟机类型为 i 时，在第 y 个任务节点执行任务时所需的估测时间，t_b 为虚拟机启动的时间与执行软件安装的时间之和.

任务调度总时间是在关键路径进行任务调度时所需要的时间总和.

当任务节点为 y，虚拟机使用类型为 i 类时，j 个虚拟机执行任务需要的总费用为

$$C_y=\left\lceil\frac{t_y}{a}\right\rceil\cdot c_i\cdot j \tag{3}$$

其中：a 为虚拟机的租用周期，c_i 为虚拟机使用类型为 i 类时单个租用周期的单价.

总费用 C_{total} 为每个任务节点的费用的累加之和.

$$C_{\mathrm{total}}=\sum_{i=1}^{n}c_i \tag{4}$$

由上所述，适应度函数 fitness 定义为

$$\mathrm{fitness}=w_1\cdot C_{\mathrm{total}}+w_2\cdot\mathrm{Max}(0,T_{\mathrm{total}}-T_{\mathrm{deadline}}) \tag{5}$$

其中：$w_1+w_2=1$，w_1 与 w_2 为实数；T_{deadline} 为用户所提供的调度该批任务所要求的截止时间与任务批开始进行调度的时间差值. Max 代表取括号内的两数中较大的数.

2.3　二进制粒子群算法及其改进

在粒子群算法中，每个优化问题的最优解可以通过搜索粒子的位置得到，由优化的函数决定它们的适应度函数值，粒子们依据当前的最优粒子状态进行更新. 标准粒子群算法根据式(6)进行解的迭代更新，在每一次迭代过程中，每个粒子根据两个值更新它们的位置. 其中一个值是粒子自身所找到的最优解，这个解称为个体极值；另一个极值是整个群体找到的最优解，这个解称为全局极值.

$$\begin{cases}v^i(t+1)=w\times v^i(t)+c_1\times\mathrm{rand}()\times\\ \qquad\left(y^i(t)-x^i(t)\right)+\\ \qquad c_2\times\mathrm{rand}()\times\left(\hat{y}^i(t)-x^i(t)\right)\\ x^i(t+1)=x^i(t)+v^i(t)\end{cases} \tag{6}$$

其中：w 为惯性权重，rand() 为均匀分布在(0，1)之间的随机数，c_1 和 c_2 为学习因子. 粒子的速度 v_i 被最大速度 $v_{\max}$ 所限制，即若 $v_i>v_{\max}$，则令 $v_i=v_{\max}$，而若 $v_i<-v_{\max}$，也令 $v_i=v_{\max}$. 为了处理离散型优化问题，需对公式进行修改，提出了二进制粒子群算法. 它的速度公式保持不变，位置的迭代公式修正为

$$x_{i+1}=\begin{cases}1, & \text{if}\quad \mathrm{rand}()<\mathrm{sigm}(v_{i+1})\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

$$\mathrm{sigm}(v_{i+1})=1/[1+\exp(-v_{i+1})] \tag{8}$$

为了防止 sigm 函数饱和，通常给 v_{i+1} 的取值规定一个区间范围为{－4.0，4.0]，sigm 函数与速度 v_i 的关系如图 2 所示. 需指出的是：sigm 函数值不表示某位变化的概率，而是表示某位取 1 的概率.

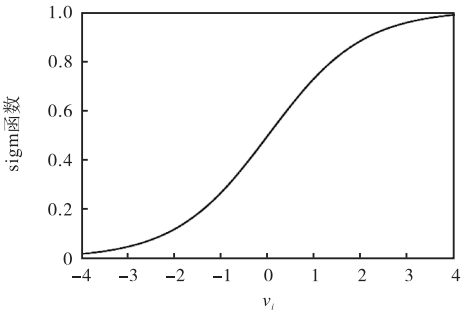

图 2　sigm 函数与 v_i 关系

Fig. 2　Relationship between sigm function and v_i

根据式(7)，粒子改变是一种根据概率进行的改变. 位为 1 的概率为 $\mathrm{sigm}(v_{id})$，而位为 0 的概率为

$1-\mathrm{sigm}(v_{id})$. 如果位已经是 0，则表示位发生变化的概率为 $\mathrm{sigm}(v_{id})$；如果位已经是 1，则表示位发生变化的概率为 $1-\mathrm{sigm}(v_{id})$. 在速度 v_{id} 值已知的前提下，位发生改变的绝对概率为

$$p(\Delta) = \mathrm{sigm}(v_{id}) - \mathrm{sigm}(v_{id})^2 \tag{9}$$

结合式(8)和式(9)，得

$$p(\Delta) = (1/1+\exp(-v_{id})) - (1/[1+\exp(-v_{id})])^2 \tag{10}$$

式(7)表示位速度与位改变的绝对概率之间关系，其关系如图 3 所示. 由图 3 可以看出，当位速度是 0 时，位的绝对改变概率最大. 经分析得出以下结论：当二进制粒子群算法的粒子速度是 0 时，位最有可能发生改变，不利于全局最优解的搜索，此时的算法具有更强的随机性，缺乏方向性. 二进制粒子群算法随着迭代次数的增多，其随机性更强了，不能收敛到全局最优解. 本文针对二进制粒子群算法的这种情况进行了改进.

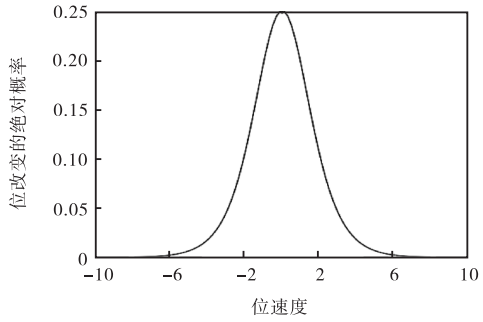

图 3　位速度与位改变的绝对概率之间关系

Fig. 3　Relationship between bit velocity and bit change absolute probability

在粒子群算法寻优的过程中用 v_i 表示速度，能够对粒子的位置和方向有一定影响，使算法在指定的搜索范围内进行搜索. 假设算法的进化迭代视为一个自适应过程，则不断的会有新粒子替代粒子 x_i 的位置，并且根据 v_i 进行自我调节. 在进行每一次迭代时，群体经验的最优解是粒子移动的方向，粒子群算法是有目的的寻找最优解. 而二进制粒子群算法中，v_i 表示一种可能的概率性，粒子的每维分量的位置取值可能以 $\mathrm{sigm}(v_{i+1})$ 的概率取 1，或者以 $1-\mathrm{sigm}(v_{i+1})$ 的概率取 0，其位发生变化概率为

$$p(\Delta) = \mathrm{sigm}(v_{i+1})\,(1-\mathrm{sigm}(v_{i+1})) \tag{11}$$

分析得出，当速度越趋近于 0 时，其位发生改变的概率越高. 粒子在寻找最优解的过程中，粒子越来越靠近最优粒子，即速度在不断趋近于 0 时，此时位发生改变的概率应该越小，而二进制粒子群算法反而越来越大，如此必然影响算法的寻优能力.

本文提出的算法基于二进制粒子群算法的寻找最优解模式进行改进，粒子速度作为粒子位置的修正项，应当充分考虑粒子之间的此种导向作用，所以在对二进制粒子群算法进行改进中，速度迭代公式维持不变，速度归一化公式 sigm 函数以及位置迭代公式改进为

$$\begin{cases} \mathrm{sigm}(v_{i+1}) = \dfrac{v_{i+1}^2}{1+v_{i+1}^2} \\ x_{i+1} = \begin{cases} 1-x_i, & \text{if rand()} < \mathrm{sigm}(v_{i+1}) \\ x_i, & \text{otherwise} \end{cases} \\ x_i \in \{0,1\} \end{cases} \tag{12}$$

改进后的新 sigm 函数与 v_i 关系如图 4 所示，当速度的绝对值越来越大时，位改变的概率也越来越大；反之，当速度趋近于 0 时，位改变的概率越来越小. 由此得出，改进后的函数遵循二进制粒子群算法的寻优模式，更有利于全局最优解的搜索.

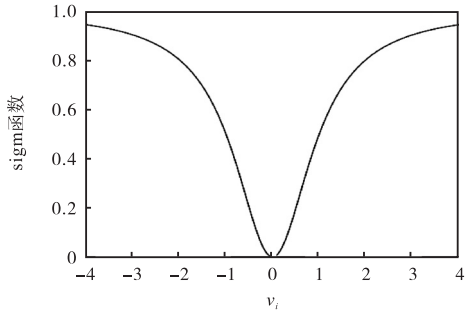

图 4　新 sigm 函数与 v_i 的关系

Fig. 4　New relationship between sigm function and v_i

3　仿真实验

为了验证本文所提模型以及算法的有效性，选择开源的云仿真 CloudSim 平台对优化算法进行仿真验证. 采用 Montage 和 CyberShake 作为测试用例. 软件安装时间为 10 s，带宽 B 为 10 MB/s，虚拟机的加载时间为 30 s，采用 Amazon EC2 提供的虚拟机价格，它是基于按小时计费原则. 本文提出的改进二进制粒子群算法与基本二进制粒子群算法分别进行任务完成时间和花费两方面的比较与分析.

Montage 100 测试用例随迭代次数增加完成时间的变化趋势测试结果如图 5 所示. 测试用例为 100 时，与未经改进的算法相比，改进的二进制粒子群算法的趋势平缓，时间变化比较稳定，且所用时间相对少. 当迭代次数为 90 次时，开始收敛.

Montage 100 测试用例随迭代次数增加所需费用的变化趋势测试结果如图 6 所示. 当任务数为 100

时，与未经改进的算法相比，改进的二进制粒子群算法的趋势较平缓，且费用相对少.

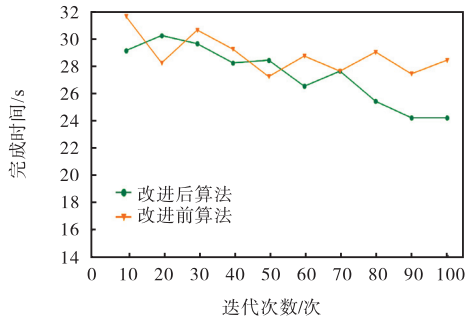

图 5　Montage 测试用例任务数为 100 时算法的完成时间

Fig. 5　Algorithm completion time for the Montage test case with 100 tasks

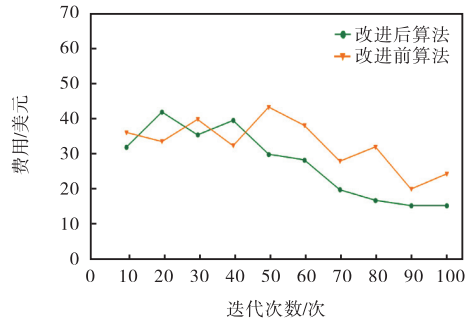

图 6　Montage 测试用例任务数为 100 时算法所需费用

Fig. 6　Algorithm completion cost for the Montage test case with 100 tasks

CyberShake 100 测试用例随迭代次数增加完成时间的变化趋势和随迭代次数增加所需费用的变化趋势测试结果如图 7 和图 8 所示. 与未经改进的算法相比，改进的二进制粒子群算法所需的完成时间和费用都优于未改进的算法.

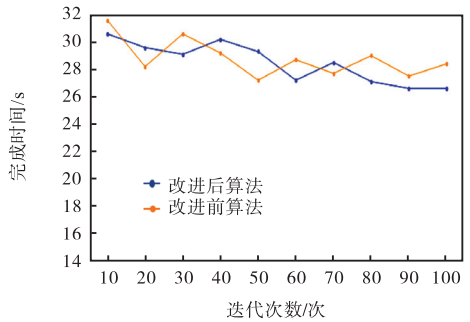

图 7　CyberShake 测试用例任务数为 100 时算法的完成时间

Fig. 7　Algorithm completion time for the CyberShake test case with 100 tasks

综合分析，本算法对不同的工作流具有普适性，在不同的工作流上都体现了优化作用. 本算法对租赁成本的优化效果更为明显，而对于完成时间的优化略微不明显. 分析原因是由于本算法对租赁成本的优化是越低越好，而对完成时间的优化是尽量小于截止时间.

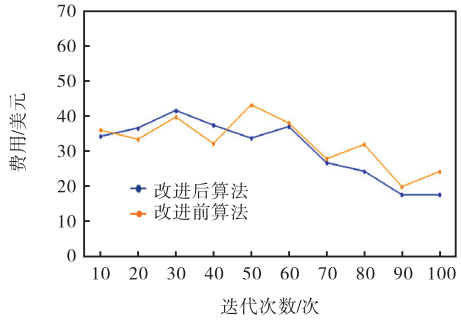

图 8　CyberShake 测试用例任务数为 100 时算法所需费用

Fig. 8　Algorithm completion cost for the CyberShake test case with 100 tasks

Montage 100 测试用例在迭代次数相同且任务数不同时，时间和费用的变化趋势如图 9 和图 10 所示. 当迭代次数相同时，改进的二进制粒子群算法随着任务数量的增加，完成时间和费用都优于未经改进的二进制粒子群算法.

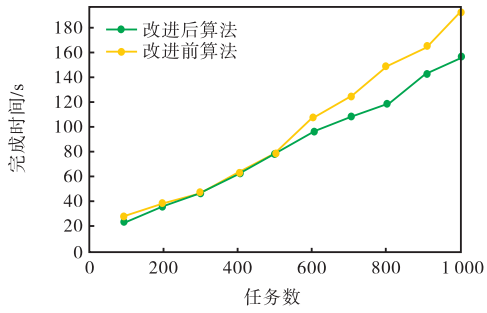

图 9　不同任务数对应的完成时间

Fig. 9　Completion time for different tasks

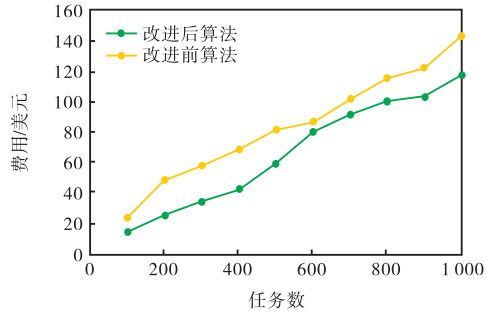

图 10　不同任务数对应的所需费用

Fig. 10　Costs of different tasks

4　结　语

针对二进制粒子群算法不能收敛于粒子的全局

最优位置的问题，本文对粒子的更新公式进行了修改，提出了一种基于改进的二进制粒子群的任务调度算法. 实验结果表明，经过改进的二进制粒子群算法能得到很好的收敛，具有普适性，所得最优解具有更低的调度时间和费用，提高了资源利用率. 在相同的条件设置下，该算法优于传统的二进制粒子群算法.

致谢：本研究获得天津教委项目（2017KJ035，2018KJ105）资助.

参考文献：

[1] BEDRA A. Getting started with google APP engine and clojure[J]. IEEE Internet computing，2010，14(4)：85–88.

[2] GARG S，CHAURASIA P K. Application of genetic algorithms task scheduling in cloud computing[J]. International journal of computer sciences and enginerring，2019，7(6)：782–787.

[3] MASDARI M，SALEHI F，JALALI M，et al. A survey of PSO-based scheduling algorithms in cloud computing[J]. Journal of network & systems management，2017，25(1)：122–158.

[4] CAI Z，LI X，GUPTA J N D. Heuristics for provisioning services to workflows in XaaS clouds[J]. IEEE Transactions on services computing，2016，9(2)：250–263.

[5] CAI Z，LI X，RUIZ R. Resource provisioning for task-batch based workflows with deadlines in public clouds[J]. IEEE Transactions on cloud computing，2019，7(3)：57–73.

[6] 熊聪聪，陈长博，赵青，等. 基于遗传算法的批处理科学工作流任务调度算法的改进[J]. 天津科技大学学报，2020，35(2)：74–80.

[7] DEVARAJ A F S，ELHOSENY M，DHANASEKARANA S，et al. Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments[J]. Journal of parallel and distributed computing，2020，142：36–45.

[8] RODRIGUEZ M A，BUYYA R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds[J]. IEEE Transactions on cloud computing，2014，2(2)：222–235.

[9] 马亮，李晓. 基于改进粒子群算法的云计算任务调度策略[J]. 计算机与现代化，2013，11(9)：78–81.

[10] SAEEDI S，KHORSAND R，BIDGOLI S G，et al. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing[J]. Computers & industrial engineering，2020，147：106649.

[11] 刘建华，杨荣华，孙水华. 离散二进制粒子群算法分析[J]. 南京大学学报（自然科学版），2011，41(5)：504–514.

[12] 罗金炎. 改进的二进制粒子群算法求解车辆路径问题[J]. 数学的实践与认识，2014，44(23)：205–211.

责任编辑：郎婧