浙江理工大学学报，2019，41(3)：354-359
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673－3851(n).2019.03.012

基于改进萤火虫算法的云计算任务调度算法

李成辉[a]，李仁旺[b]，杨强光[b]，贾江鸣[b]
（浙江理工大学，a.信息学院；b.机械与自动控制学院，杭州 310018）

摘　要：针对云计算环境中的大量任务，为提高资源利用率，缩短任务完成时间，提出了一种基于改进萤火虫算法的任务调度算法。首先用每只萤火虫的位置表示一种可行的调度方案，利用自然数对萤火虫进行编码表示其所在位置，并随机初始化种群；然后在搜索过程中利用混沌扰动对适应度函数值较低的萤火虫进行激活，保持种群活性，利用真实物理反弹理论对飞出搜索区域的萤火虫进行控制，维护种群多样性，降低陷入局部最优的概率。在CloudSim平台进行仿真测试，结果表明，该算法能够有效缩短任务完成时间，且寻优结果更佳。

关键词：云计算；任务调度；萤火虫算法；混沌；多样性
中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1673-3851 (2019) 05-0354-06

Cloud computing task scheduling algorithm based on improved firefly algorithm

LI Chenghui[a], *LI Renwang*[b], *YANG Qiangguang*[b], *JIA Jiangming*[b]
(a. School of Information Science and Technology, b. Faculty of Mechanical Engineering & Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: In order to improve the utilization rate of resources and shorten the task completion time, a task scheduling algorithm based on improved firefly algorithm was proposed to deal with a large number of tasks in the cloud computing environment. Firstly, the location of each firefly was used to express a feasible scheduling scheme, and natural numbers were applied to encode fireflies and represent their locations. Meanwhile, the population was randomly initialized. Secondly, in the search process, chaos disturbance was used to activate the firefly with low fitness function value and maintain the population activity. Real physical rebound theory was utilized to control the firefly flying out of the search area so as to maintain the population diversity and reduce the probability of falling into local optimal. Finally, the simulation test of CloudSim platform was conducted. The test results show that the algorithm can shorten the task completion time and the optimization result is better.

Key words: cloud computing; task scheduling; firefly algorithm; chaos; diversity

0　引　言

云计算是计算机技术发展中的焦点领域，它是利用互联网将物理上分散分布的资源节点整合成一个逻辑上为一体的数据服务系统。随着对云计算研究的不断深入，云服务已经从单纯的分布式计算框架发展成并行计算、分布式计算、虚拟化、网络存储、负载均衡、热备份冗余等互联网技术综合运用的产物[1]，是已经投入商业运营的基于互联网的资源虚拟化的服务模式。这种服务模式将计算资源（包括

收稿日期：2018－10－07　　**网络出版日期**：2018－12－28
基金项目：国家自然科学基金项目(51475434)；浙江省自然科学基金项目(LY14G010007)
作者简介：李成辉(1992－　)，男，安徽颍上人，硕士研究生，主要从事计算机应用技术方面的研究。
通信作者：李仁旺，E-mail：renwangli@zstu.edu.cn

但不限于计算、存储、应用软件、网络、服务等）作为可配置的共享资源按需对外提供服务。目前，云计算提供的服务可划分为三种类型：基础设施即服务（IaaS）、平台即服务（PaaS）以及软件即服务（SaaS）[2]。

在商业领域，企业运营管理所涉及的产品供应链包含了采购、加工、库存以及销售等多个环节，各环节的生产数据反映了企业运行的实际状况。现代企业为了在激烈的市场竞争中占据一定的优势，势必要在尽可能短的时间内对生产数据进行分析，辅助生产监管、采购计划、库存管理等各项工作，这就要求云计算系统能够及时有效地完成用户提交的任务[3]，云计算提供的强大的计算服务满足了企业对数据快速处理的需求。在云计算中，硬件和软件都是可分配的资源，这些资源接受云环境中资源管理器的统一管理。伴随着云计算环境的不断扩大和任务需求的不断增长，资源管理器对任务合理的调度和资源高效的使用显得越来越重要。传统的经典调度算法如优先级调度法、时间片轮转法、多级反馈队列调度法等，并不能满足云环境下大规模任务调度的准确性和实时性要求，容易造成服务器负载不均问题。因此，设计出性能更优的云计算任务调度算法成为目前云计算领域研究的热点[4]。

云计算中的任务调度问题属于 NP 完全问题[5]，天牛须搜索、遗传、蚁群、粒子群等智能算法可用于求解此类组合优化问题。目前，很多学者在此方面取得了一些研究成果。葛君伟等[6]通过分析已有调度算法存在的缺陷，以总任务完成时间和负载均衡为调度目标，提出了基于遗传加差分算法的调度算法。娄建峰等[7]针对粒子群算法易过早收敛的缺陷，提出了基于改进粒子群算法的任务调度算法，通过引入混沌扰动使种群跳出局部最优。杨单等[8]提出了基于改进混沌萤火虫算法的资源调度算法，通过使用拉格朗日松弛函数对调度模型进行化简，较好地解决了负载不均的问题。

目前，企业的管理方式、业务处理趋向于标准化，即使存在一定的个性化管理，其任务的范围和粒度也都容易控制。对于成熟的产品供应链，其生产数据的产生速度和数据量是可预期的，即使出现波动一般也在可控范围内，计算资源也是可以预估的。云计算系统大多采用分布式集群搭建，扩展性可以得到保障。本文根据云计算中计算资源和任务量的特点，提出了基于改进萤火虫算法的任务调度算法，针对传统萤火虫算法容易早熟陷入局部最优的问题，引入混沌扰动策略对目标值较低的部分进行扰动，同时在萤火虫位置更新中依据真实物理反弹理论将飞出搜索空间的萤火虫反弹回来，维持种群的寻优能力。

1　云计算中的任务调度描述

在云计算中，任务执行不仅要考虑总的完成时间，还要兼顾虚拟资源的负载均衡。云计算系统通常会将复杂庞大的计算任务切分为若干个子任务，然后依据任务调度策略将其分配到不同的资源节点上运行。目前，已经用于生产环境的云计算服务采用的是 Google 提出的 MapReduce 编程模式[9]。MapReduce 是一种用于处理分布式海量数据集的并行编程模式，它充分借鉴了分而治之的理念，用于大规模数据集的并行计算。MapReduce 将一个较大的待处理对象切分成若干个较小的块，每个块都有一个 Map 任务对其加工，Map 任务之间相互独立、并发执行。Reduce 任务就是对 Map 任务产生的中间结果进行合并、排序。所有 Reduce 任务输出的结果集中起来便是完整的结果集。MapReduce 运行模式如图 1 所示。

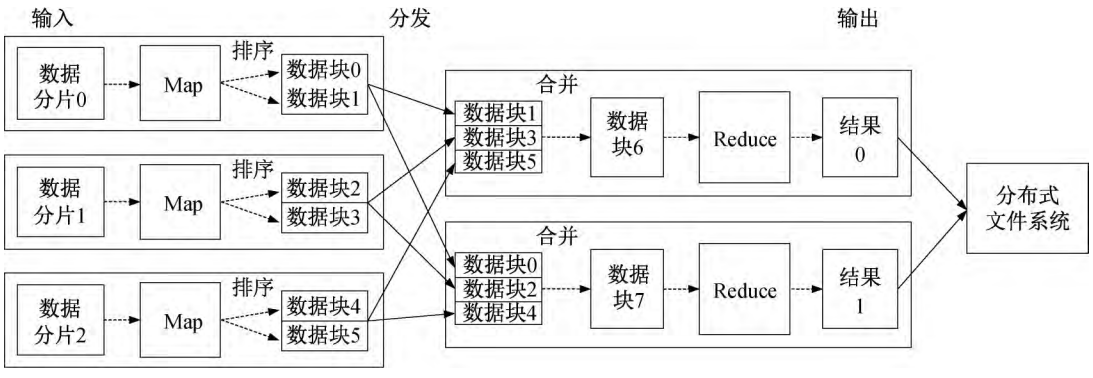

图 1　MapReduce 运行模式

2 标准萤火虫算法

2.1 算法机理

萤火虫算法(Firefly algorithm, FA)是由剑桥大学学者 Yang 提出的一种启发式群智能优化算法[10-11]。算法思想源自于自然界中萤火虫在夜晚进行集群活动时出现的行为现象。萤火虫算法将搜索空间中的点描述为自然界中的萤火虫个体,将搜索和迭代过程看作成萤火虫个体间相互吸引和位移的过程,用适应度函数的适应值表示萤火虫的荧光亮度,荧光亮度体现了萤火虫所处位置的优劣。种群中亮度较低的萤火虫不断地向比自己更亮的萤火虫靠拢,如果两个萤火虫的荧光亮度相同,则各自随机移动。萤火虫的位置在移动的过程中完成迭代,最终聚集在荧光亮度最高的萤火虫周围,即最优位置附近,至此完成了寻优过程。

萤火虫算法涉及两个关键因素:萤火虫的荧光亮度和个体间的吸引度。萤火虫的荧光亮度受自身位置影响,位置越好则适应值越佳,萤火虫的荧光亮度就越高。萤火虫是彼此吸引的,吸引度的高低只与萤火虫发出荧光的亮度和萤火虫之间的距离有关。其中,亮度会随着萤火虫之间距离的增加而减小,这是模拟光在介质中传播逐渐衰竭的现象。吸引度与萤火虫之间的距离成反比,距离越大则吸引度越小。

2.2 算法的数学描述

萤火虫 i 的荧光亮度

$$I_i = f(\boldsymbol{X}_i) \tag{1}$$

其中:$\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{id})$为萤火虫 i 在 d 维空间中的位置;f 为萤火虫荧光亮度的计算函数;I_i 为萤火虫 i 在位置 $\boldsymbol{X}_i$ 处的荧光亮度值。本文直接使用适应度函数值来表示萤火虫的荧光亮度,以此建立萤火虫 i 的荧光亮度与适应度函数值之间的联系。

萤火虫 i 对萤火虫 j 的吸引度

$$\beta_{ij}=\beta_0 \mathrm{e}^{-\lambda r_{ij}^2} \tag{2}$$

其中:β_0 为光源处的吸引度,即最大吸引度,通常为1;λ 为光吸收系数,$\lambda\in[0.01,100]$;r_{ij} 为萤火虫 i 与萤火虫 j 之间的笛卡尔距离,即:

$$r_{ij}=\|\boldsymbol{X}_i-\boldsymbol{X}_j\|=\sqrt{\sum_{k=1}^{d}(x_{ik}-x_{jk})^2} \tag{3}$$

萤火虫 j 在 t 时刻被萤火虫 i 吸引,则萤火虫 j 的位置更新公式为:

$$\boldsymbol{X}_j^{t+1}=\boldsymbol{X}_j^t+\beta_{ij}(\boldsymbol{X}_i^t-\boldsymbol{X}_j^t)+\alpha^t\varepsilon^t \tag{4}$$

其中:$\boldsymbol{X}_i{}^t$、$\boldsymbol{X}_j{}^t$ 为萤火虫 i 和萤火虫 j 在 t 时刻的位置;α^t 为 t 时刻的步长因子,通常是取[0,1]的常数;ε^t 为 t 时刻服从高斯分布的随机数向量。

3 基于改进萤火虫算法的云计算任务调度

3.1 萤火虫编码

将萤火虫算法应用到云计算任务调度系统中,首先需要对萤火虫进行编码,将云计算任务调度与萤火虫的荧光亮度、吸引度等结合起来。由于任务数量是离散的整型数值,故对萤火虫的编码也是取离散的整型数值。本文采用自然数对萤火虫编码,萤火虫位置的维数等于接受调度的云计算任务数量。假设有 m 个任务,云环境中有 n 个可分配资源,则萤火虫可以编码成 m 维向量:

$$\boldsymbol{Y}=(y_1,y_2,\cdots,y_i,\cdots,y_m) \tag{5}$$

其中:$1\leqslant y_i\leqslant n$,$1\leqslant i\leqslant m$,萤火虫每一维下标代表一个任务编号,每个一维分量 y_i 表示云环境分配给此任务的资源编号。假设集群中有10个任务,有5个可使用资源,则粒子可编码为(3, 4, 1, 2, 3, 5, 2, 1, 3, 5),如表1所示。因为每只萤火虫代表一种可行的调度方案,对萤火虫解码便可获知任务分配状况,如表2所示。

表1 萤火虫编码示例

任务编号	萤火虫编码(资源编号)
1	3
2	4
3	1
4	2
5	3
6	5
7	2
8	1
9	3
10	5

表2 萤火虫解码示例

资源编号	任务编号
1	3,8
2	4,7
3	1,5,9
4	2
5	6,8

3.2 初始化种群的产生

假设对 m 个任务在资源数量为 n 的环境下进行调度,种群依据问题规模初始化时系统随机生成 NP 只萤火虫,记第 i 只萤火虫的位置向量为:

$$\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{im}),$$

其中：$1 \leqslant x_{ij} \leqslant n$，$x_{ij}$ 表示第 i 号资源分配给第 j 号任务，初始化时 x_{ij} 取[1，n]区间上的随机整数。

3.3　适应度函数

适应度函数是用来评估调度方案优劣的标准。萤火虫的位置代表一种任务调度方案，萤火虫的荧光亮度可以表示萤火虫此刻所在位置的优劣，故可以建立萤火虫位置与荧光亮度的关联关系，将萤火虫的荧光亮度作为适应度函数值来评估任务调度方案的优劣，适应度函数值越大表示萤光亮度越高，萤火虫的位置也就越好，调度方案也就越好。本文以适应值代表萤火虫的荧光亮度，萤火虫以最优适应值为目标，迭代更新位置和吸引度，直至满足停止条件。本文采用 $m \times n$ 阶矩阵 $\boldsymbol{ETC}$ 表示任务在云环境中的运行时间，其中 $\boldsymbol{ETC}(i,j)$ 表示任务 i 占用资源 j 的时间。根据 $\boldsymbol{ETC}$ 和对萤火虫解码结果，可以得到云环境中每个资源被任务占用的时长 $Time(j)$：

$$Time(j)=\sum_{i=1}^{m}\boldsymbol{ETC}(i,j),\quad j=1,2,\cdots,n \tag{6}$$

当云环境中所有的资源均被释放，表示所有任务都执行完毕，任务整体完成时间 $FTime$：

$$FTime=\max(Time(j)),\quad j=1,2,\cdots,n \tag{7}$$

定义适应度函数：

$$Fitness=\frac{1}{FTime} \tag{8}$$

3.4　混沌扰动策略

混沌是一种存在于自然界中的较为普遍的现象，大多数非线性系统会出现不同程度的混沌运动[12]。混沌运动看似混乱，其内部却具有规律性、遍历性及随机性等重要特征。混沌优化已经作为仿生智能算法的优化策略被广泛的应用[13]。为防止迭代后期由于种群多样性不足导致收敛速度放缓，引入混沌序列对萤火虫进行混沌扰动，增强萤火虫的搜索能力。本文选择 Tent 映射[14] 生成混沌序列，其公式为：

$$x_{n+1}=\begin{cases}2x_n, & x\in[0,0.5)\\ 2(1-x_n), & x\in[0.5,1]\end{cases} \tag{9}$$

当种群收敛速度开始减缓时，根据适应度函数值对萤火虫进行排序，用 Tent 混沌映射对当前种群中适应值较低的 20% 的萤火虫进行扰动。首先将萤火虫的每一维分量映射到(0，1)区间，生成一个新的向量 $\boldsymbol{D}=(d_1,d_2,\cdots,d_m)$，$d_i=(x_i-l_i)/(u_i-l_i)$，其中 x_i 表示该萤火虫的第 i 维分量，l_i 和 u_i 分别表示此维度值域的下边界和上边界。然后以新向量 $\boldsymbol{D}$ 为初始值，利用 Tent 映射生成新的序列，以新序列作为萤火虫的新位置，计算出其适应值。若萤火虫在新位置上的适应值优于此前萤火虫的适应值，则替换原来的位置，与适应值较高的 80% 部分重新组合进入下一轮搜索。

3.5　种群多样性控制策略

利用 3.4 的混沌扰动策略对萤火虫的位置进行调整，由于混沌运动的随机性，可能会出现萤火虫飞出搜索空间的情况，导致种群因多样性的降低而无法快速跳出局部最优。在标准萤火虫算法中，对于飞出搜索空间的萤火虫都作了限制，限制操作会使得飞到搜索空间外的萤火虫停留在边界上。这种处理机制较为粗糙，对种群多样性的维护并没有显著成效。为此，引入文献[15]的真实物理反弹理论将飞出搜索空间的萤火虫反弹回空间内，维护种群的多样性。

位置更新公式为：

$$x'_{id}(t)=\begin{cases}lb_{id}+\gamma[lb_{id}-x_{id}(t)], & lb_{id}-a<x_{id}(t)<lb_{id}\\ x_{id}(t)+v_{id}(t+1), & lb_{id}<x_{id}(t)<ub_{id}\\ ub_{id}-\gamma[x_{id}(t)-ub_{id}], & ub_{id}<x_{id}(t)<ub_{id}+a\\ lb_{id}+\gamma[ub_{id}-lb_{id}], & lb_{id}-a>x_{id}(t)\ \text{或}\\ \quad x_{id}(t)>ub_{id}+a\end{cases} \tag{10}$$

其中：lb_{id} 和 ub_{id} 为搜索空间内第 i 个萤火虫在第 d 维空间上的下界和上界；γ 为满足[0，1]内的随机数；a 为用户自定义常量。

3.6　算法描述

基于改进的萤火虫算法的云计算任务调度算法的基本执行步骤如下：

Step1　初始化萤火虫种群，依据问题规模随机初始化 NP 个萤火虫的位置，设定算法各参数：最大吸引度、步长因子、触发混沌扰动的迭代阈值等。

Step2　判断迭代次数是达到最大迭代次数，若达到最大迭代次数，执行 Step10，否则执行 Step3。

Step3　对编码萤火虫进行解码，求得资源与云计算任务间的分配关系，计算出矩阵 $\boldsymbol{ETC}$。

Step4　依据矩阵 $\boldsymbol{ETC}$ 计算出萤火虫的适应度函数值，即萤火虫的荧光亮度。

Step5　所有萤火虫开始趋光移位：首先对于每只萤火虫，搜索其附近亮度比它高的萤火虫，依据距离公式计算两者之间的距离；然后根据公式计算萤火虫的相对吸引度，萤火虫依此决定下一步的移动

方向,由公式更新位置。

Step6 检测萤火虫的位置,运用公式控制种群多样性。

Step7 种群迭代次数达到触发混沌扰动的迭代阈值且最优位置连续 5 次没有更新时执行 Setp8,否则执行 Step2。

Step8 依据亮度对萤火虫进行排序,根据本文的扰动策略对排在后 20% 的萤火虫采取混沌扰动。

Step9 计算被扰动的萤火虫在新位置上的适应值,若优于之前的适应值则替换之前的萤火虫,否则不变。将被扰动后的萤火虫与其余 80% 的萤火虫重新组合,执行 Step2。

Step10 解码全局最优萤火虫,解码结果作为最佳调度方案输出。

4 实验分析

实验使用 CloudSim-3.0.3 模拟云计算环境,对文章提出的任务调度算法进行测试实验。CloudSim[16] 是由澳大利亚墨尔本大学 RajkumarBuyya 教授主导研发的云计算仿真环境。CloudSim 是基于 GridSim 模型发展而来,提供了云计算的特征,主要应用于云计算服务和任务调度策略的性能测试。CloudSim 平台的 DatacenterBroker 类提供虚拟机代理功能,用户通过对其添加实现了自定义调度算法的方法来扩展代理类的功能,从而完成测试实验。

实验中只考虑云环境中资源的计算速度和任务的大小,调度能力以所有任务的总完成时间作为评估指标。为验证本文调度算法的性能,在相同条件下与粒子群算法、蚁群算法进行对比实验。

4.1 算法改进前后的收敛值对比

在 5 个云计算资源、1000 个云计算任务的调度规模下,用本文算法和基于传统萤火虫算法的调度算法分别求解调度问题,结果曲线如图 2 所示。从图 2 中可见,传统萤火虫算法在快速收敛的过程中由于搜索能力的衰减,收敛速度由快变慢,在全局寻优的过程中陷入了早熟;相比于传统萤火虫算法,本文算法的收敛速度和寻优速度明显较快。这主要得益于在算法中引入了混沌干扰以及基于真实物理反弹理论的种群多样性控制策略,在全局寻优的过程中减缓了搜索能力的衰减,降低陷入局部最优的概率。

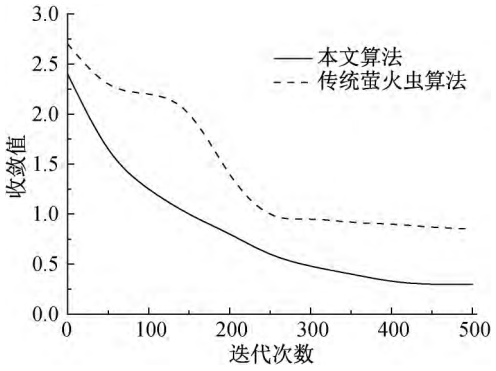

图 2 算法改进前后的收敛值曲线

4.2 任务完成总时间对比

为对比验证本文算法在任务完成总时间上的优化,在拥有 5 个计算资源的云环境中,分别测试云计算任务数量为 50 个和 500 个时,不同迭代次数下任务完成的总时间,并利用粒子群传算法、蚁群算法与本文提出的算法进行对比实验。实验中各云计算资源的计算能力由函数在[400, 1200] MPIs 区间内随机生成,云计算任务的任务长度在[400,1000] MI 区间内。遗传算法、蚁群算法的种群规模、迭代次数与本文算法的种群规模、迭代次数相同,其它参数依照相应文献设置。各算法重复运行 20 次取平均值。实验结果如图 3—图 4 所示。

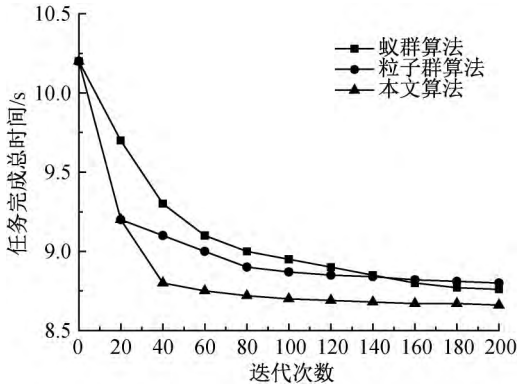

图 3 任务量为 50 的总完成时间

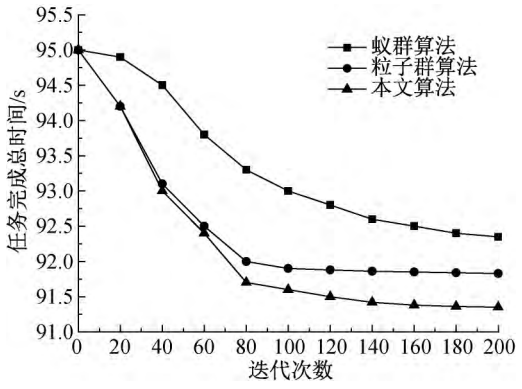

图 4 任务量为 500 的总完成时间

对比实验结果可知，本文所提的云计算任务调度算法继承了萤火虫算法快速收敛的能力，任务完成时间相比其他算法明显减小，引进混沌扰动策略和种群多样性控制策略，维护了种群的寻优能力，减小算法陷入局部最优解的概率，更加适合解答云计算调度问题。

4.3　资源数量对比

将 10000 个云计算任务分配到 5～10 个资源节点上运行，实验结果如图 5 所示。随着计算资源数量的增加，任务完成总时间逐渐减少，这是由于资源数量的增多导致任务竞争资源的激烈程度变小。通过图 5 对比可见，本文算法可以缩短云计算任务的总执行时间。

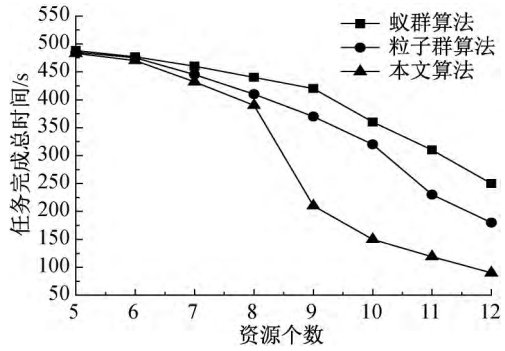

图 5　任务完成总时间随资源数量的变化曲线

5　结　语

本文针对云计算过程中的任务调度问题，采用萤火虫算法对云环境中的资源做合理分配，并在传统萤火虫算法的基础上进行了改进。针对传统萤火虫算法容易早熟陷入局部最优的问题，引入混沌扰动策略对目标值较低的部分进行扰动，同时基于真实物理反弹理论将搜索过程中飞出搜索空间的萤火虫反弹回来，有效地维持了种群的搜索能力。该算法与传统萤火虫算法以及其他智能算法的对比测试结果表明，具有较好的寻优能力和实时性。本文仅考虑了资源数量和任务数量等因素，而实际应用中还需考虑计算成本、传输时延、任务优先级、耗能等其他因素的影响，还有待进一步研究。

参考文献：

[1] 孟必平，王腾蛟，李红燕，等.分片位图索引：一种适用于云数据管理的辅助索引机制[J].计算机学报，2012，35(11)：2306-2316.

[2] 刘鹏.云计算[M].2 版.北京：电子工业出版社，2011：3-15.

[3] 钟鸣.基于云计算模式的 ERP 企业管理信息系统分析[J].通讯世界，2018(9)：20-21.

[4] 陈全，邓倩妮.云计算及其关键技术[J].计算机应用，2009，29(9)：2562-2567.

[5] Arfeen M A，Pawlikowski K，Willig A.A framework for resource allocation strategies in cloud computing environment[C]// Computer Software and Applications Conference Workshops(COMPSACW)，2011IEEE 35th Annual.IEEE，2011：261-266.

[6] 葛君伟，孙方方，方义秋.基于遗传加差分算法的云计算任务调度[J].微电子学与计算机，2016，33(11)：5-9.

[7] 娄建峰，高岳林，李飞，等.基于改进粒子群算法的云计算任务调度算法[J].微电子学与计算机，2016，33(8)：12-116.

[8] 杨单，李超锋，杨健.基于改进混沌萤火虫算法的云计算资源调度[J].计算机工程，2015，41(2)：17-20.

[9] 杜江，张铮，张杰鑫，等.MapReduce 并行编程模型研究综述[J].计算机科学，2015，42(S1)：537-541.

[10] Yang X S.Firefly algorithm，stochastic test functions and design optimization[J].International Journal of Bio－Inspired Computation，2010，2(2)：78-84.

[11] 程美英，倪志伟，朱旭辉.萤火虫优化算法理论研究综述[J].计算机科学，2015，42(4)：19-24.

[12] 李宗岳，陈志军，李名远.基于混沌扰动策略的果蝇优化算法[J].微电子学与计算机，2016，33(7)：64-68.

[13] 张燕.基于混沌优化的最优运输路径问题研究[J].计算机科学，2017，44(S1)：133-135.

[14] 匡芳君，金忠，徐蔚鸿，等.Tent 混沌人工蜂群与粒子群混合算法[J].控制与决策，2015，30(5)：839-847.

[15] 徐刚，杨玉群，刘斌斌，等.一种基于多样性策略的粒子群算法[J].南昌大学学报(理科版)，2013，37(1)：17-21.

[16] 何婧媛.云计算仿真工具 CloudSim 的研究与应用[J].科技资讯，2016，14(2)：32-33.

(责任编辑：康　锋)