基于改进细菌觅食算法的云计算资源调度策略

赵宏伟 田力威
(沈阳大学信息工程学院 沈阳 110044)

摘 要 资源调度是云计算的核心问题之一,调度算法的好坏直接影响着系统的处理能力。生物群体智能算法是一类模仿群体生物在自然界进化过程中表现出的群体智能性的算法,具有良好的协调性和整体稳定性。将菌群觅食算法应用到云计算资源调度的计算方法中,可以利用菌群算法对节点进行复制和消亡,对云计算资源调度节点的分配情况进行控制。针对传统菌群算法中随机选择趋化过程所造成的资源变化区间过大的问题,文中提出了改进的基于群体感应交流机制的细菌觅食CBFO优化算法和在群体协作过程中引入细菌趋化动作的MPSOBS优化算法,根据节点周围的节点情况和整个菌群的情况选取趋化因子,使趋化的过程更加准确。仿真实验结果表明,所提算法在任务的执行时间、系统负载均衡和资源服务质量方面均优于BFO算法,在提高资源利用率的同时能保证云应用的服务质量。

关键词 云计算,资源调度,群体智能,细菌觅食

中图法分类号 TP331 **文献标识码** A **DOI** 10.11896/jsjkx.181002000

Cloud Computing Resource Scheduling Strategy Based on Improved Bacterial Foraging Algorithm

ZHAO Hong-wei TIAN Li-wei
(School of Information Engineering, Shenyang University, Shenyang 110044, China)

Abstract As one of the core problems of cloud computing, the efficiency of scheduling algorithm has a direct impact on the operation capacity of the system. Swarm intelligence algorithm, with good coordination and overall stability, is one kind of swarm intelligence algorithms which imitates swarm intelligence in the process of evolution swarm. This paper presented a calculation method of bacteria foraging algorithm applied to cloud computing resource scheduling algorithm, which can be used to control the node allocation of cloud computing resource scheduling by using bacterial swarm algorithm to copy and perish the nodes. According to the problem of too much resource change interval caused by the random selection chemotax in the traditional flora swarm algorithm, the bacteria foraging CBFO optimization algorithm based on Quorum Sensing mechanism and the MPSOBS optimization algorithm introducing bacteria chemotaxis action in the process of group collaboration were proposed in this paper. According to the environment around the nodes and the situation of the whole flora, the chemotaxis factor is selected to make the process of chemotaxis more accurate, which is implemented on the cloud computing platform. The simulation results show that the proposed algorithm is more efficient than the BFO algorithm in terms of task execution time, system load balancing and resource service quality, and can improve the service quality of cloud applications while improving resource utilization.

Keywords Cloud computing, Resource scheduling, Swarm intelligence, Bacterial foraging

菌群觅食算法(BFO)是一种群体智能算法。虽然菌群算法在食物选择过程中的行为相比人工鱼群算法更不容易控制和分析,但是菌群算法本身可以对计算单元进行复制和消亡,使得在使用菌群算法对程序进行控制时,可以更好地根据实际运行的需要控制资源管理和程序的运行。在进行云计算资源调度的过程中,一方面可以通过将任务合理地分配到各个节点中对资源的调度进行优化,另一方面也可以通过对资源节点的选择来优化资源调度的过程[1-2]。菌群算法可以通过对周围环境的感知来拟制进化机制,计算出资源配置的合理情况。

1 问题的提出

云计算是一个商业计算模型,利用互联网将网络资源提供给用户。计算任务通常被分配到一个资源池中,使得所有的应用和请求都可以根据实际情况及实际需要得到相应的计

到稿日期:2018-09-28 返修日期:2018-11-27 本文受国家自然科学基金面上项目(71672117),辽宁省自然科学基金(20170540646,20180550658)资助。

赵宏伟 男,教授,主要研究方向为云计算、大数据、群体智能算法优化,E-mail:zhw30@163.com;**田力威** 教授,主要研究方向为数据智能处理技术,E-mail:tianliwei@163.com(通信作者)。

算资源、存储空间和与之相关的大量软件信息的支持。通常，云计算系统可以使用虚拟化技术对资源进行封装，并将封装好的资源提供给用户[3]。

在云计算系统中，云计算和云资源具有复杂性及不确定性，云资源的分配和分类通常是一个动态的过程，类别和分配不是固定的，并且经常变化。为了提高资源的可利用率，通常从资源和任务的计划出发，利用资源分配的核心机制来增强资源分配和调度的合理性。早期的静态调度算法只是简单地将 M 个任务分配到 N 个资源上，而云资源调度过程中存在资源的动态申请与释放，并且云计算是对外提供服务的，必须考虑资源消耗的成本。静态算法难以满足云资源调度要求，容易造成资源的浪费，因此当前主要采用动态调度算法[4]。为了实现对云计算资源的动态调度，本文从资源计划分配的角度设计了一种资源调度模型，可以通过对整个计算资源环境的理解来设计应用的实际效果情况、服务质量情况、资源数量、虚拟机的负载平衡和多因素的混合问题。

目前关于菌群觅食算法，有研究通过调整方法的参数来对算法进行优化。Liu 等[5]利用大肠杆菌之间的相互作用增加菌群的紧凑度，并以此来调节菌群对食物位置定位的准确性，进而提高算法的收敛性。Mishra 等[6]将模糊集合和模糊规则的概念与菌群觅食行为结合在一起，提出了一种模糊集合菌群觅食的改进方法。Datta 等[7]在菌群觅食和复制的过程中，使用了一种自适应方法，通过对环境的感知来调节自适应的适应度和趋化步长，使算法在贴近食物时对食物的定位更加准确。Majhi 等[8]将细菌觅食方法和神经网络结合，使算法在融合和环境感知的训练过程中更加准确。因为传统的菌群觅食算法在对周围环境感知的过程中比较弱化，所以本文对菌群觅食算法进行了改进，使其可以对周围环境进行感知和计算。

针对云计算资源分配的节点选择问题，本文通过对菌群觅食算法进行改进，使菌群算法对周围环境的感知能力更强，从而形成一种协作的觅食行为，并使云计算节点的选择和调用过程更加合理，同时将改进方法成功应用于云计算资源调度中。

2 相关工作

本文主要研究生物智能算法在云计算虚拟资源调度上的应用，本节将从菌群觅食模型、菌群感应机制等相关方面进行介绍[9-10]。

2.1 菌群觅食模型

菌群优化方法模拟最多的微生物对象为大肠杆菌(Escherichia Coli，E. Coli)。E. Coli 细菌个体的生命周期分为：自由觅食－获取营养阶段－缺乏营养死亡阶段。比较成功的基于细菌行为的优化算法是由 Passino 提出的细菌觅食优化算法(Bacteria Foraging Optimization，BFO)。此算法是基于群体的，一些学者已着手从算法的理论及应用两方面开展了初步的研究[11]。BFO 算法模型主要包括趋化操作、繁殖和消亡操作以及迁徙操作，具体表述如下。

(1)趋化算子描述

大肠肝菌在觅食过程中会记住以前某个时刻的状态，并通过与当前状态进行比较来做出一种决策判断，这就是细菌对环境的信息反馈机制。这种机制使得细菌表现出了对环境的多种适应性行为，如前进、停止、翻转等。

(2)繁殖算子描述

在细菌的趋化循环完成后，即认为细菌能量已达到一定程度，细菌可以进行下一步繁殖算子。在 BFO 算法中，能量高的细菌获得繁殖机会，能量低的细菌被自然淘汰掉而变成消亡状态。细菌的繁殖能量为细菌的适应度之和。为了保持细菌种群规模不变和繁殖算子操作的有效性，种群定义能量高的一半细菌繁殖，以此替代能量低的另一半细菌。另外，在 BFO 的繁殖算子中，一般需要设定细菌的规模为偶数[12]。

(3)迁移算子描述

温度升高和食物消耗在改变细菌生存环境的同时，可能使细菌个体的生活区域发生变化，从而导致细菌群体消亡。更重要的是，由于受到水冲刷，部分细菌群体可能进行集体迁移。为了提高细菌觅食算法的全局寻优能力，设计了迁移算子来模拟这一功能。

从以上各个算子的缺点可以看出，在 BFO 算法优化的过程中，细菌趋化算子的效率较低；繁殖算子使得细菌种群的多样性缺失；迁移算子采用固定迁移概率，导致精英细菌消亡，从而影响了 BFO 算法的全局寻优能力。因此，进一步深入研究细菌觅食优化的生物学原理，分析仿生算法表达的真正机理(群体感应机制等)，简化和抽象算法的模型，为我们研究细菌觅食优化和仿生优化提供了较为宽阔的平台及空间。

2.2 群体感应机制

细菌群体间的感应机制[13]是极为复杂的。在算法的实现过程中，首先，将细菌群体划分为 M 个簇，每个簇记为 C_m；其次，在每一步算法的迭代过程中，趋化算子的游动执行细菌群体感应操作式(1)，即每个细菌个体向自己所在的簇前进一步。

$$B_i(t+1)=B_i(t)+rand(\ )\cdot Step\cdot(C_m^i-B_i(t)) \quad (1)$$

其中，$rand(\ )$为(0，1)的随机数，$Step$ 为趋化的步长，C_m^i表示簇的位置。C_m^i通过细菌的当前位置和所有簇的平均中心点 V_j 的误差之和获得：

$$C_m^i=\sum_{B_i\in C_i}|B_i-V_j| \quad (2)$$

这种群体感应机制有利于细菌朝着拥有丰富食物的方向前进，快速找到食物源区域，而且可以避免处于较优位置的细菌个体得不到有效利用。因此，在 BFO 算法中引入群体感应机制，让细菌之间相互协作，并把这种机制应用到云虚拟资源调度系统的调度寻优方案中，可以增强对调度节点分配情况的控制。

3 云计算资源虚拟调度模型

到目前为止，大量学者的研究表明，云计算的资源调度问题是一个 NP 完全问题。同时，一些传统的精确解法的时间复杂度为 $O(n^2)$，其中，n 为需要调度的任务数量，m 为分配的虚拟机的数量[14-15]。

接下来,为了解决本文研究的云计算资源调度的菌群算法应用问题,本节将对使用的云计算虚拟资源调度模型进行介绍。假设云计算平台需要调度的作业集合为 $J(j_1,j_2,\cdots,j_k)$,每一个作业包含的任务数集合是 $T_j(t_{j1},t_{j2},\cdots,t_{ji})$,给每一个工程分配的虚拟资源向量为 $\boldsymbol{V}_n(v_1,v_2,\cdots,v_n)$,任务所分配的虚拟机为 $H(h_{11}^1,h_{12}^2,\cdots,h_{ik}^n)$,$h_{ik}^n=1$ 表示第 i 个任务被分配到作业 j 上并且在虚拟资源 n 上运行,$h_{ik}^n=0$ 则表示任务 i 没有在虚拟资源 n 上运行。

根据云计算模型中资源的分配位置,任务列表主要考虑任务执行的最终时间(式(3))和虚拟机的负载平衡(式(6)),以实现资源利用效果的最大化。

$$f_1(x)=\min[(F(J)+E(J))/2] \tag{3}$$

$$F(J)=\min_{j=1}^{k}\sum_{v=1}^{n}\sum_{i=1}^{m}t_{ijv}\cdot h_{ij}^v \tag{4}$$

$$E(J)=(\sum_{j=1}^{k}t_j)/m \tag{5}$$

其中,$F(J)$用来计算作业的完成时间,$E(J)$是用户期望的平均执行时间,t_{ijv}表示作业 j 中的子任务 i 的执行时间。

虚拟资源的负载平衡也是本文关注的一个重要方面,并且作为云计算资源调度的关键指标备受关注。因此,式(6)给出云计算中心负载平衡的计算标准。

$$f_2(x)=\min(\sum_{v=1}^{n}(t_{ijv}\cdot h_{ij}^v-\sum_{v=1}^{n}t_{ijv}\cdot h_{ij}^v/n)^2/n) \tag{6}$$

在云计算的资源调度过程中,虚拟资源调度的负载平衡可以被看作每一个虚拟机执行任务的时间与云计算中心整体执行任务的平均时间之差达到最小。因此,需要把以上两方面因素综合在一起进行考虑,据此提出了云计算资源虚拟调度模型,如式(7)所示。模型的约束关系和虚拟机资源情况满足式(8)－式(10)的要求。

$$F(X)=\min(\lambda_1 f_1(x)+\lambda_2 f_2(x)) \tag{7}$$

$$h_{ij}^v\leqslant g_{ij}^v \tag{8}$$

$$h_{ij}^v\geqslant 0,g_{ij}^v\geqslant 0,t_{ijk}\geqslant 0 \tag{9}$$

$$1-P_l\geqslant P_{\text{succ}} \tag{10}$$

其中,h_{ik}^v是一个判定系数,表示当前任务是否在虚拟资源上运行;g_{ik}^v表示虚拟机 v 是否可以承载任务 i 的情况;t_{ijk}表示任务的实际运行时间;在云计算环境中,资源计算失败的概率为 P_l,它可以从资源监视器中得到;P_{succ}就是用户要求的执行成功的概率。

在每个虚拟机资源调度中,当虚拟机故障率过高时,会启动虚拟机放置的物理资源调度,重新布置故障率高的虚拟机器。虚拟资源调度的优化目标是在保证既定目标的情况下,使任务完成时间最短且虚拟机负载均衡,以达到资源调度额优化配置的目标。

4　改进的菌群觅食算法

4.1　CBFO 算法

粒子群算法有非常明显的群体感应协作机制。与粒子群算法一样,菌群觅食算法也应具有一定的群体感应和协作能力,其基本思想是通过趋化、繁殖消亡以及迁移过程来进行自我优化,具体优化的过程如下所示。对于经典的菌群觅食算法的趋化过程,使用式(11)和式(12)进行实现。

$$P(i,j+1,k,l)=P(i,j,k,l)+C(i)\phi(i) \tag{11}$$

$$\phi(i)=\Delta_i/\sqrt{\Delta_i^{\mathrm{T}}\Delta_i} \tag{12}$$

其中,$C(i)$表示菌群提前移动的步数,$\phi(i)$表示菌群的翻转度。

针对目前菌群算法存在的问题,本节将周围环境信息与趋化过程相结合,提出了一种群体感应交流机制的菌群觅食算法。改进后的菌群觅食算法主要是在菌群趋化过程中考虑了菌群的整体环境信息,具体通过菌群个体之间的关联情况、个体细菌和子群之间的情况、影响菌群和子群信息因素的情况来提升算法的性能。

具体实现上,在智能优化算法的基础上,菌群觅食算法加入了个体之间的协作关系和通信关系的预判,同时在子群信息中寻找优化的觅食区域信息。改进后的趋化函数如式(13)所示:

$$\begin{aligned}\Delta(i)=&(\theta_{\text{gbest}}(i)-\theta_{\text{individual}})+(\theta_{\text{subbest}}(i)-\theta_{\text{individual}})+\\&(\theta_{\text{ibest}}(i)-\theta_{\text{individual}})\end{aligned} \tag{13}$$

算法 1 给出了改进后的菌群觅食算法 CBFO 的具体执行过程。其中,适应度由目标函数计算获得,根据菌群环境进行子群划分,通过式(13)获得细菌协同趋化翻转的角度,并根据式(1)的群体感应机制完成细菌趋化直行,向食物能量较高的区域前进。如果一个细菌的能量值较高,在本算法中该细菌将以一定的概率进行分裂和迁徙操作,当细菌进行繁殖或迁徙操作后,新生细菌进入初始状态,重启觅食过程,当达到最大次数或小于预定收敛精度要求时,停止迭代,输出最优解。

算法 1　群体感知交流机制的菌群觅食算法 CBFO

输入:细菌种群数量、相关迭代次数、收敛精度等参数的初始值

输出:最优细菌个体

初始化细菌种群数量、迭代次数 T=1、最大迭代次数 maxgen 等参数;

计算细菌适应度;

WHILE(T<=maxgen)

　计算当前的菌群环境信息,形成当前的子群信息;

　FOR (每个细菌)

　　计算细菌的随机矢量 $\Delta(i)\in R^D$(式(13)),表示当前细菌的方位角;

　　根据当前位置和环境信息确定翻转方向;

　　进行趋化翻转操作,若有改进,则继续在该方向上前进一步(式(1));

　　判断是否分裂,若分裂,则复制该细菌进行下一个个体操作;

　　判断是否消亡,若消亡,则从种群中移除,进行下一个个体操作;

　　通过计算概率判断是否迁徙,迁徙后随机初始化细菌位置,进行下一个个体操作;

　END FOR

　T=T+1;

END WHILE

4.2　MPSOBS 算法

本节将 BFO 算法中细菌个体趋化过程中的翻转和直行机制引入到多群体协作的 PSO 算法中,提出了一种 BFO 和 PSO 的混合算法,即 MPSOBS 算法。该算法将细菌向任意方向移动的随机角度和直行步长加入到 PSO 算法速度和位置的更新策略中:

$$v_{id}(t)=wv_{id}(t-1)+R_1c_1(p_{id}-x_{id}(t-1))+R_2c_2(p_{gd}-x_{id}(t-1)) \quad (14)$$

$$x_i^{t+1}(c,d)=x_i^t(c,d)+\Phi(i)v_i^{t+1} \quad (15)$$

其中，c 表示趋化翻转的次数，d 表示趋化直行的次数；$\Phi(i)$表示一个符合随机分布的随机角度，在趋化翻转时通过 $rand(\ )$ 取随机角度，但在趋化直行时 $\Phi(i)=1$。

通过引入 BFO 细菌的趋化机制，MPSOBS 算法可以得到较好的全局探测与局部开发能力。算法 2 给出了引入 BFO 细菌的趋化机制的多种群协同粒子群算法的执行过程。

算法 2 细菌趋化的多种群协同算法 MPSOBS

输入：粒子种群数量、相关迭代次数、收敛精度等参数的初始值

输出：最优粒子个体

初始化细菌种群数量 S、迭代次数 T=1、最大迭代次数 maxgen 等参数；

将种群 S 随机划分为 M 个子群；

计算各个粒子的适应度；

```
WHILE(N<=maxgen)
  WHILE(趋化翻转没达到次数)
    WHILE(趋化直行没达到次数 and 没达到个体最优)
      FOR 每个粒子
        计算粒子的适应度；
        根据式(14)进行粒子速度的更新；
        根据式(15)进行粒子位置的更新；
      End FOR
      更新粒子的最佳位置；记录全局最佳位置；
    END WHILE
END WHILE
  T=T+1；
END WHILE
```

算法 2 把细菌的觅食趋化行为引入到粒子群优化算法中，在速度更新阶段，通过式(15)将趋化翻转和趋化直行引入到优化算法中，趋化翻转次数和趋化直行次数可以自行设定，当达到最大次数 T_{max} 或小于预定收敛精度 ξ 要求时，停止迭代，输出最优解。

5 改进细菌觅食算法的云计算资源调度的实现

改进的细菌觅食算法 CBFO 和 MPSOBS 都考虑了周围环境对整体觅食效果的影响，这类方法将有助于对云计算资源分配的估计，可以更好地确定应选择多少资源节点以及如何选取资源节点。本节将对如何在第 3 节中提出的云计算资源调度模型上运用改进的细菌觅食算法进行介绍。

用改进的细菌觅食算法 CBFO 计算如何进行云计算资源的任务调度问题时，每一个细菌都可以代表一种实际的调度方案。具体地，在云计算资源调度中运用改进的菌群觅食算法的过程如下所示：

Step 1 根据云计算资源的类型和数量情况，对云计算资源的虚拟化资源配置池进行初始化，资源池被分为 n 个不同的资源子群。

Step 2 初始化菌群集合 $X_k=\{x_1^k,x_2^k,\cdots,x_N^k\}$，$X_k$ 表示关于任务 K 的子群，并且为每个子群分配 N 个独立的细菌资源。

Step 3 根据菌群趋化优化算法，计算出每个细菌的适应值，然后根据式(7)选出优化算法得到的细菌个体。

Step 4 根据式(11)和式(12)对当前细菌的位置和方向进行调整，同时对比同阶段子群和其他单体细菌的实际情况，然后根据计算结果对邻居子群和单体细菌的情况进行更新。

Step 5 为了进一步计算群体变化的趋势、菌群的繁殖过程和迁移过程，使用式(13)并根据菌群的实际情况来更新适应值的情况。

Step 6 对迭代终止条件进行判断，在不满足迭代次数上界的情况下，为了计算当前迭代得到的最优位置情况，找到云计算资源分配节点部署的最优化策略。否则，待达到迭代的终止次数时结束程序。

MPSOBS 算法在云环境中的实现过程与 CBFO 算法基本类似，但由于算法的偏重不同，因此操作步骤也有所不同。MPSOBS 算法的 Step 4 是根据式(13)和式(14)更新共生种群的速度，选择个体极值、种群极值和全局极值，并引入细菌趋化动作。MPSOBS 算法的优化结果更加倾向于保持解的多样性。

6 实验及分析

6.1 CBFO 和 MPSOBS 算法仿真

本节对菌群感应算法 CBFO、菌群-粒子群算法的混合算法 MPSOBS，以及 BFO 和 PSO 进行了比较实验。

(1)实验设置

在比较的过程中使用了 5 类函数进行区分，具体情况如表 1 所列。

表 1 5 类测试函数

Table 1 Five test fucntions

测试函数	具体形式
f_1=Sphere	$f_1(x)=\sum_{i=1}^{D}x_2^i$
f_2=Rosenbrock	$f_2(x)=\sum_{i=1}^{D}100\times(x_{i+1}-x_i^2)^2+(1-x_i)^2$
f_3=Rastrigin	$f_3(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10$
f_4=Griewank	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$
f_5=Ackley	$f_5(x)=-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{30}\sum_{i=1}^{D}\cos2\pi x_i\right)+20+e$

在实验中，BFO 的参数设置如下：算法中细菌总数为 100，趋向次数为 100，游动次数为 3，复制次数为 4，迁徙次数为 2，游动步长为 0.001，迁徙概率为 0.25。CBFO 和 MPSOBS 算法的参数设置与 BFO 一致，只是将细菌划分为 5 个子群。PSO 选定的学习系数为 $c_1=c_2=2.0$；初始的迭代权重为 $\omega=0.9$，并且终止值为 0.5。

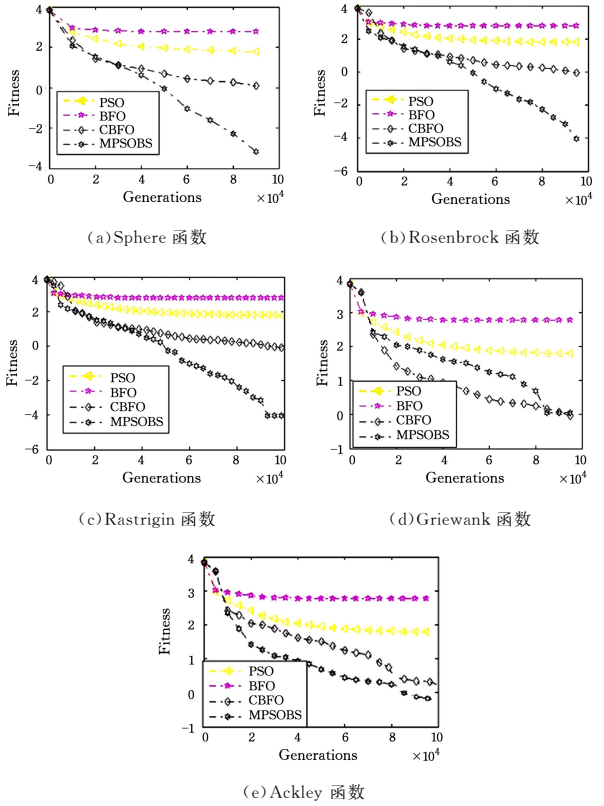
实验运行中，将迭代次数的上限设置为 10000 次，如果收敛满足要求，则算法执行结束；如果收敛未达到预期的效果，那么执行 10000 次或达到预设精度以后，算法自动终止。

(2)实验结果及分析

对 4 种方法运行的适应值进行观察，分别考查几种方法的最优情况、最差情况、平均情况和标准差情况。

图 1 给出了 4 种算法关于 5 种函数在数据集上运行的收敛情况。图 1(a)显示了 4 种方法关于 Sphere 函数的收敛情况。所有算法在实验测试的迭代范围内都可以实现有效收

敛，说明了各种算法的有效性。同时，基础菌群觅食算法 BFO 的收敛速度和基本情况最差，算法收敛的趋势比较平缓，收敛效果一般。改进的 MPSOBS 方法表现出了很强的收敛效果，平均不到 1000 次迭代就开始收敛，并且可以保证极好的收敛状态。



(a)Sphere 函数　(b)Rosenbrock 函数

(c)Rastrigin 函数　(d)Griewank 函数

(e)Ackley 函数

图 1　5 个函数上的收敛情况

Fig. 1　Covergence on 5 functions

从图 1(b)以及图 1(c)表现出的运行结果可知，对于 Rosenbrock 和 Rastrigin 函数，BFO，PSO 和 CBFO 的表现均比较差，但是 MPSOBS 取得了不错的效果。

从图 1(d)以及图 1(e)可以看出，基础菌群觅食算法 BFO 和粒子群算法 PSO 在函数测试上的表现比较差，但是 CBFO 和 MPSOBS 得到了较好的收敛效果。由算法在 Griewank 和 Ackley 函数性能测试上的优越表现可知，CBFO 和 MPSOBS 算法比较适合处理分布式问题。从图 1(e)可以看出，MPSOBS 算法在 Ackley 函数上的收敛表现最好，说明该算法取得了最好的结果，也说明了改进算法的正确性。

从图 1 的整体情况来分析，PSO 在所有函数上的表现居中，CBFO 和 MPSOBS 的表现比较好，BFO 的表现最差，但是所有算法都可以在 10000 次迭代以内收敛，因此以上方法对 5 种函数均适应。同时，通过比较也发现 MPSOBS 方法在各种函数上都体现出了较好的适应能力和性能。在算法的时间复杂度上，CBFO 算法在经典 BFO 的趋化操作中引入群体感应机制，减少了算法游动和翻转循环，在降低时间复杂度的同时更有利于实现算法的快速收敛。MPSOBS 算法在多群体协作的 PSO 算法中引入了趋化过程的翻转和直行机制，改进粒子位置和速度的更新策略，在提高粒子更新精度的同时降低了算法的时间复杂度。

6.2　CBFO 和 MPSOBS 云环境资源调度的模拟仿真

(1)实验设置

本文使用云计算模拟平台 CloudSim 对算法进行仿真测试(CloudSim 的优势在于它比真实的云环境测试的速度快，且成本低)，以验证和检测 CBFO 及 MPSOBS 算法在云计算环境中的可行性。在云计算资源调度策略中，通过提出的云计算资源调度模型的策略，可以评价 CBFO 和 MPSOBS 算法的准确性和收敛性。下面对改进的 CBFO 和 MPSOBS 算法及基础 BFO 算法进行了比较。

本节实验算法的参数设置如下：种群规模为 100，子群规模为 5，迭代次数均为 100，但是 MPSOBS 的迭代次数为进化次数、趋化次数和游动次数之和的乘积，学习因子 $c_1=c_2=2$，惯性权重的大小设置为 0.8。此外，在云计算的中心部署上，实验设置了 500 个虚拟资源；为了减小虚拟环境之间的差异，将虚拟资源数的子群数量设置为 5，并将最大迭代次数设为 100。

(2)实验结果及分析

从图 2 中可以看出，在迭代 100 次的情况下，本文提出的 CBFO 和 MPSOBS 算法在研究云资源调度问题时，相比 BFO 算法具有更快的收敛速度。MPSOBS 算法在改进群体协作模式的同时引入了细菌趋化动作，增加了个体的多样性，因此 MPSOBS 算法能够在较快收敛的同时得到更好的解。

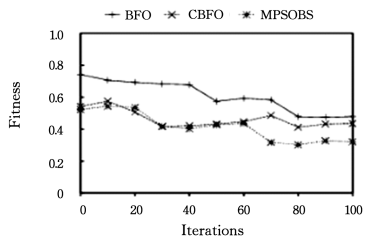


图 2　3 种云资源调度算法的收敛结果

Fig. 2　Convergence result of three algorithms

图 3、图 4 给出了 CBFO 算法、MPSOBS 算法和 BFO 算法在完成 100 个任务量时虚拟资源调度过程中的任务完成时间和资源负载平衡的运行情况。CBFO 和 MPSOBS 算法在任务总完成时间上的优势越来越明显，并且随着任务数量的增多，CBFO 和 MPSOBS 算法的优势越来越明显，因此，改进的细菌觅食算法 CBFO 和 MPSOBS 算法相对于 BFO 算法更具有优势。同时，从图 4 还可以看出，相对于 MPSOBS 和 BFO 算法，CBFO 算法具有更好的负载平衡性优势。这是因为 MPSOBS 算法为了保证粒子个体的多样性，防止陷入局部最优，引入了细菌趋化动作，从而弱化了负载平衡。

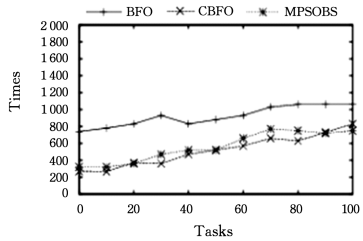


图 3　3 种云资源调度算法的任务完成时间的比较

Fig. 3　Execution time of three algorithms

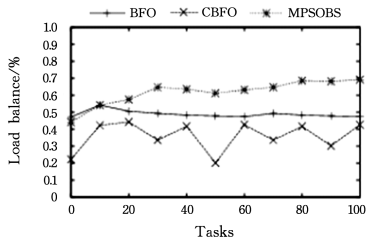


图 4 3 种算法负载平衡度的比较

Fig. 4 Load balance comparison of three algorithms

此外，采用 CBFO 和 MPSOBS 算法进行优化可以有效地实现云计算资源调度的优化和交换，这些针对服务目标水平的计算可以满足多节点和冲突节点之间的需求，应用服务的满意度水平相比 BFO 都有较大的提升。图 5 给出了 3 种算法的服务满意度情况，可以看出 MPSOBS 和 CBFO 算法具有更高的服务满意度(接近 80%)。

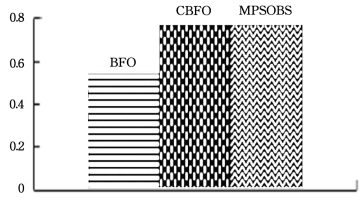


图 5 资源服务满意度的比较

Fig. 5 Service satisfaction comparison

通过以上实验可以看出，改进的 CBFO 和 MPSOBS 算法在云计算资源调度的过程中具有更好的性能，运行更加稳定，迭代次数更少，服务满意度的比率更好。以上说明这两种方法特别适合用于解决云计算的资源调度问题；同时，算法在资源调度的过程中可以提高云计算资源的利用率，保证云计算资源应用的服务质量。

结束语 本文从微观生物的角度研究了一类简单、易描述的 E. coli 细菌的典型行为，并引入了细菌群体感应机制，建立了群体感应的数学模型，提出了菌群优化算法 CBFO；同时，把细菌的趋化动作引入到多协同粒子群算法中，提出了 MPSOBS 优化算法，以保证粒子自身进化的多样性，避免算法早熟收敛。为了验证改进算法的性能，将 CBFO 和 MPSOBS 算法应用于函数优化问题，并将其与其他传统的群体智能优化算法进行性能对比。结果表明，改进的算法是一种高效、准确的群体智能优化算法，并且适用于求解云计算的资源调度问题。本文重点研究了使用细菌觅食智能算法来解决云计算资源调度的问题。目前，大数据和云计算的相关算法越来越火热，相关话题的安全、质量、智能方向的研究也越来越重要，因此，后期的工作会将云计算资源调度算法与大数据和云计算的安全、质量、智能相结合来进行研究。

参 考 文 献

[1] HOSSAIN S. Infrastructure as a service[J]. Cloud Computing Service & Deployment Models Layers & Management, 2013, 22(7):26-49.

[2] TSAFRIR D, SCHUSTER A, BENYEHUDA M, et al. Deconstructing Amazon EC2 Spot Instance Pricing[C]// Third International Conference on Cloud Computing Technology and Science. Washington, DC, USA, 2012:304-311.

[3] KIPPENBROCK T, HOLLOWAY E, MOORE D D. Google Docs[J]. CIN: Computers, Informatics, Nursing, 2010, 28(3): 138-140.

[4] ERLYKIN A D, HARPER D A T, SLOAN T, et al. Data from: Mass extinctions over the last 500 myr: an astronomical cause? [J]. Palaeontology, 2017, 60(2):365-372.

[5] GAO S, LIU X, ZHANG R, et al. Analysis of Block OMP using Block RIP[J]. 微生物学报, 1999, 97(7):162-171.

[6] BROWNE M C, CLARKE E M, MISHRA B. Automatic Verification of Sequential Circuits Using Temporal Logic[J]. IEEE Transactions on Computers, 2006, 35(12):1035-1044.

[7] MITRA S, DATTA S, AND T P. Introduction to Physical Polymer Science[J]. Macromolecular Chemistry & Physics, 2010, 207(8):787-787.

[8] MAJHI J, SMID M. Multi-criteria geometric optimization problems in layered manufacturing[C]// The Fourteenth Symposium on Computational Geometry. ACM, 1998:19-28.

[9] WANG G H, LI Q H, LIU A F. Multi-objective optimization cloud workflow scheduling evolutionary genetic algorithm [J]. Computer Sciences, 2018, 45(5):31-37. (in Chinese)
王国豪，李庆华，刘安丰，多目标最优化云工作流调度进化遗传算法[J]. 计算机科学，2018，45(5)：31-37.

[10] WEI X R, WANG F. A Reliability-Driven Cloud Workflow Scheduling Genetic Algorithm [J]. Application Research of Computers, 2018, 35(5):1390-1394. (in Chinese)
魏秀然，王峰. 一种可靠性驱动的云工作流调度遗传算法[J]. 计算机应用研究，2018，35(5)：1390-1394.

[11] TIAN G H, MENG D, ZHAN J F. Resource dynamic provisioning strategy based on failure rules in cloud computing environment[J]. Journal of Computer, 2010, 33(10):1859-1872. (in Chinese)
田冠华，孟丹，詹剑锋. 云计算环境下基于失效规则的资源动态提供策略[J]. 计算机学报，2010，33(10)：1859-1872.

[12] WANG Z J, CHEN Y J. Research on I/O Resource Utility Optimization Scheduling Algorithm for Cloud Storage [J]. Computer Research and Development, 2013, 50(8): 1657-1666. (in Chinese)
王健宗，谌炎俊. 面向云存储的 I/O 资源效用优化调度算法研究[J]. 计算机研究与发展，2013，50(8)：1657-1666.

[13] VAQUERO L M, RODERO-MERINO L, MORÁN D. Locking the sky: a survey on IaaS cloud security[J]. Computing, 2011, 91(1):93-118.

[14] ARABNEJAD H, BARBOSA J. A Budget Constrained Scheduling Algorithm for Workflow Applications[J]. Journal of Grid Computing, 2014, 12(4):665-679.

[15] XU Z J, CHEN S X. Research on Fusion Algorithm Based on Membrane Computing and Ant Colony Algorithm in Cloud Computing Resource Scheduling [J]. Computer Measurement and Control, 2017, 25(1):120-127. (in Chinese)
徐浙君，陈善雄. 基于膜计算和蚁群算法的融合算法在云计算资源调度中的研究[J]. 计算机测量与控制，2017，25(1)：120-127.