云计算中基于共享机制和群体智能优化算法的任务调度方案

符 晓
(西南石油大学计算机科学学院 成都 610500)

摘 要 为了提高云计算中虚拟机(VM)的利用率并降低任务的完成时间,提出了一种融合共享机制的混合群智能优化算法,实现云任务的动态调度。首先,将虚拟机调度编码为蜜蜂、蚂蚁和遗传个体。然后,利用人工蜂群算法(ABC)、蚁群算法(ACO)和遗传算法(GA)分别在各自邻域内寻找最优解。最后,通过一个共享机制使 3 种算法定期交流各自搜索到的解,并将获得的最佳解作为当前最优解进行下一次迭代过程,以此来加速算法收敛并提高收敛精度。通过 CloudSim 进行了一个云任务调度的仿真实验,结果表明提出的混合算法能够合理有效地调度任务,在任务完成时间和稳定性方面具有优越的性能。

关键词 云计算,任务调度,共享机制,蚁群算法,遗传算法,人工蜂群算法
中图法分类号 TP309 **文献标识码** A

Task Scheduling Scheme Based on Sharing Mechanism and Swarm Intelligence Optimization Algorithm in Cloud Computing

FU Xiao
(School of Computer Science, Southwest Petroleum University, Chengdu 610500, China)

Abstract In order to improve the utilization rate of virtual machine (VM) in cloud computing and reduce the completion time of tasks, a hybrid intelligent optimization algorithm of fusion sharing mechanism was proposed to realize dynamic scheduling of cloud tasks. First, the virtual machine scheduling is encoded as bees, ants and genetic individuals. Then, using artificial bee colony (ABC), ant colony optimization (ACO) and genetic algorithm (GA), the optimal solution is found in each neighborhood. Finally, by a mechanism of sharing , three algorithms regularly exchange their solutions and obtain the optimal solution as the current optimal solution for the next iteration process, in order to accelerate the algorithm convergence and enhance the accuracy of convergence. Through the CloudSim, the results of cloud task scheduling simulation experiment show that the proposed hybrid algorithm can reasonable scheduling tasks effectively, and has the superior performance in the task completion time and stability.

Keywords Cloud computing, Task scheduling, Sharing mechanism, Ant colony optimization, Genetic algorithm, Artificial bee colony

云计算是根据用户的需求动态地提供以数据为中心的计算服务系统。云提供商在数据中心积累大量主机或服务器,其中每个主机可运行一个或多个虚拟机(Virtual Machine, VM)。云提供商应向云用户提供简单快速的应用部署,并提高资源利用率。VM 可以包括以不同速度运行的处理器、内存和处理不同位置的各种存储系统的存储器。此外,应用程序可以独立执行,且不需要任何特定的配置。但是,随着网络用户数量的日益增加,快速、高效地对任务进行调度,提高资源利用率,成为了云计算的核心问题[1]。

文献[2]中实现了基于蚁群算法(Ant Colony Optimization, ACO)的云任务调度,此算法的主要目标是最小化给定任务集的完成时间,它可以根据云中可用的不同 VM 来处理所有任务请求。文献[3]提出了改进的蚁群优化算法(Modified Ant Colony Optimization Algorithm, MACOA),改进的目的是提高基本蚁群优化算法的性能,减少任务执行时间。该方法还介绍了基本蚁群优化控制参数的自适应标准。但以上两种算法搜索最优解决方案的能力尚不能满足实际需求。此外,文献[4]提出使用 ACO 的节点负载平衡来实现负载平衡。此算法中,蚂蚁可以向前和向后移动。如蚂蚁搜索食物称为向前方向,返回巢则是向后方向,这种行为有助于快速平衡节点。该算法可以更好地利用资源,但会消耗更多的功率,而且它具有很高的网络开销。

本文提出了一种新的混合算法,该混合算法基于共享机制和群体智能优化算法,利用了蚁群优化算法、遗传算法(Genetic Algorithm, GA)和人工蜂群算法(Artificial Bee Colony, ABC)的优点,建模蜜蜂、种群中个体和蚁群的行为,以解决云任务调度问题。提出的共享机制模块用来使混合算法成员之间共享解,以此实现快速收敛。最后通过 CloudSim 软件进行仿真评估实验,实验结果证明,相比于 ACO 算法、GA 算法和 ABC 算法,提出的混合算法可以获得更好的 VM 利用率,并且在完成时间和不平衡程度方面显著优于其他算法。

1 相关技术

云计算中,在适当的资源上调度任务至关重要,调度的目

符 晓(1988—),女,硕士生,主要研究领域为计算机网络、云计算、大数据分析。

的是从有限对象集中找到最佳映射[5]。若是对少量对象进行调度的简单问题,则可以通过枚举简单地计算出;若是对复杂问题进行调度,则需要运用启发式和近似方法。启发式是一种接近最优的算法,可以快速找到好的解。它迭代地增强了关于特定候选解的质量,但不能保证找到最优解。启发式算法获得了广泛的应用,因为它能在适当的时间提供可接受的解,可以解决许多领域中的困难问题。许多来自启发式算法的新算法取决于群体智能(Swarm Intelligence,SI)。应用 SI 启发技术的有蜂群、蚁群、鱼群和鸟群等。近些年,研究人员提出了基于 ACO,GA 和 ABC 的算法,用于解决云计算任务调度的问题[6-11]。

1.1　蚁群优化算法(ACO)

ACO 的主要思想是模拟蚁群的自然行为,图 1 给出了 ACO 的伪代码,该算法主要包含两个迭代步骤:解构建和信息素更新。

1)解构建:根据依赖于信息素轨迹和启发式信息的概率转移规则来完成解构建。

2)信息素更新:使用生成的解来执行信息素的更新。信息素更新规则分为两个阶段,即信息素轨迹自动降低的蒸发阶段和添加正值的增强阶段。

```
1. 初始化信息素路径
2. 重复
   For 每只蚂蚁 Do
      使用信息素轨迹构建解
      更新信息素路径
      蒸发
      加强
      直到停止准则
3. 输出:找到最佳解或解集合
```

图 1　基本 ACO 的伪代码

1.2　遗传算法 (GA)

GA 算法是模拟自然界遗传机制和生物进化理论而形成的一种并行随机搜索最优化方法。图 2 给出了 GA 算法的伪代码。

```
1. 初始化遗传参数,选择适应度函数
2. 随机产生第一代种群并计算种群中每一个个体的适应度
3. 重复
   基于遗传算法的操作
      复制
      交叉
      变异
   对个体进行筛选
   保留适配值高的个体
   直到停止准则
4. 输出:找到最佳解或解集合
```

图 2　基本 GA 的伪代码

1.3　人工蜂群算法(ABC)

基于蜜蜂群巧妙觅食行为的 ABC 算法是一种优化算法。图 3 给出了 ABC 算法的伪代码。该算法从侦察蜂开始随机扫描搜索空间。然后评估这些蜜蜂访问场所的质量。之后,选择具有最高适应度的场所进行邻域搜索。然后,该算法继续搜索选定的场所,分配更多活跃的蜜蜂在这些场所进行邻域搜索。蚁群将在迭代产生的新种群中,为选定的场所招募侦查蜜蜂,分配剩余的蜜蜂随机搜索并评估其适应性。重复这个过程,直到满足给定的停止标准[12]。

```
1. 随机初始化整个蜜蜂群
2. 评估蜜蜂群体的适应性
3. 重复
      选择邻域搜索的场所
      确定补丁大小
      为选定的场所招募蜜蜂并评估其适应性
      从每个补丁中选择代表性的蜜蜂
      分配剩余的蜜蜂随机搜索并评估其适应性
      直到停止准则
4. 输出:找到最佳解或解集合
```

图 3　基本 ABC 的伪代码

2　基于所提混合算法的云调度

提出的混合算法的伪代码如图 4 所示。*Number_of_BeesAntsIndividuals* 变量表示所有成员,即蜜蜂、个体和蚂蚁的总数。每个蜜蜂和每个个体产生一个随机解,解表示为标注 VM 顺序的 ID 数组。第一个任务将分配给虚拟机 ID 矩阵中的第一个 ID,第二个任务将分配给第二个 ID。然后每个 CommonBoard 生成一个随机解。混合算法使用 4 个 CommonBoard。

```
输入:Cloudlet(任务)列表和 VM 列表
输出:VM 上任务分配的最佳解
步骤:
1. 初始化
   设定参数值:Number_of_BeesAntsIndividuals,Number_of_Bees,Number_
   of_Ants,Number_of_Individuals,Number_of_CommonBoard,Max_Num-
   ber_of_Stagnation,t_max
   Set V_Max
   Set 对于任务和 VM 之间的每个路径初始值 τ_ij(t)=c
   Set t=1
   Set BSolution 为空
2. 生成每个蜜蜂随机解
3. 生成每个个体随机解
4. 生成每个 CommonBoard 随机解
5. 检查更新 BSolution
6. For Number_of_BeesAntsIndividuals k:=1
   IF k 是一只蜜蜂
      执行 Bees()
   Else IF k 是一只蚂蚁
      执行 Ants()
   Else
      执行 Individuals()
   End IF
7. 应用全局信息素更新
8. t 加 1
9. If (t<t_max)
      转到步骤 4
   Else
      打印 BSolution
   End If
10. Stop
```

图 4　提出的混合算法的伪代码

第 1 个 CommonBoard 注册并与任何蜜蜂以及其他成员(如蚂蚁和群体中的个体)共享最佳解。第 2 个 CommonBoard 注册并与其他成员(如蜜蜂和群体中的个体)分享任何蚂蚁的最佳解。第 3 个 CommonBoard 注册并与其他成员(如

蜜蜂和蚂蚁)分享最佳解。第 4 个 CommonBoard 注册并与所有成员(如蜜蜂、群体中的个体和蚂蚁)共享随机生成的解。这 4 个 CommonBoard 通过分享来自任何成员的所有可能最优解来装配混合算法。迭代阶段模拟混合程序的技术;使用 for 循环迭代成员,每个成员类型由 Bees(),Ants()和 Individuals()模块的合适模块处理;迭代地检查 *BSolution* 变量以保持整体最优解。

2.1 Bee()模块

Bees()模块的伪代码如图 5 所示。该模块中,蜜蜂首先获得与其当前解成比例的邻居解。逻辑上,每个解都有邻居,相对于当前解的自然邻居解是当前解的替换,其中两个、三个或一些相邻 VM 的 ID 已交换。如果邻居解比目前的解更好,那么这个蜜蜂会接受更好的邻居解。之后,将检查 *BSolution* 并调用 MentionAdvert()模块。*Max_Number_of_Stagnation* 是一个阈值,用于防止蜜蜂停滞解。此时,当前蜜蜂通过随机选择第 2、第 3 或第 4 个 CommonBoard 的一个解来选择另一搜索区域。

```
Bees()
1. 生成邻居解
2. If (邻居解质量>当前蜜蜂解质量)蜜蜂接受更好的邻居解
   重置 Number_of_Stagnation 为 0
   检查更新 BSolution
   执行 MentionAdvert ()
     Else
       Number_of_Stagnation 加 1
   End If
3. If (Number_of_Stagnation>Max_Number_of_Stagnation) 随机选择第 2、第 3
   或第 4 个 CommonBoard 的解
   重置 Number_of_Stagnation 为 0
   End If
4. 返回
```

图 5 Bees()的伪代码

2.2 MentionAdvert()

MentionAdvert()模块是本文提出的一种共享机制,其伪代码如图 6 所示。共享机制模块用来使混合算法成员之间共享解,以此实现快速收敛。将上述成员的解与其 CommonBoard 的解进行比较,如果所提到的解是优选的,那么 CommonBoard 将通过接受上述解来修改其解。

```
MentionAdvert()
1. If (所提到的项是蜜蜂)
     If (提到解的质量>第 1 个 CommonBoard 解的质量)
       第 1 个 CommonBoard 接受上述项的解
     End if
   Else If (所提到的项是蚂蚁)
   If (提到解的质量>第 2 个 CommonBoard 解的质量)
       第 2 个 CommonBoard 接受上述项的解
     End if
   Else
     If (提到解的质量>第 3 个 CommonBoard 解的质量)
       第 3 个 CommonBoard 接受上述项的解
     End if
   End If
2. 为第 4 个 CommonBoard 生成随机解
返回
```

图 6 MentionAdvert()的伪代码

2.3 Ants()

Ants()模块的伪代码如图 7 所示,蚂蚁随机选择第一个任务的起始 VM。

```
Ants()
1. 将该蚂蚁随机放在起始 VM 上
2. Do ant_trip,当侦查蚁并没有结束其旅程
   蚂蚁根据式(3)选择下一个任务的 VM
   End Do
3. 检查更新 BSolution
4. 执行 MentionAdvert ()
5. 应用局部信息素更新
返回
```

图 7 Ants()的伪代码

之后,它通过从一个 VM 移动到另一个 VM 直到完成解来构建解。蚂蚁通过式(3)计算的概率转换规则选择下一任务 i 的 VM_j。

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\eta_{ij}]^{\beta}}{\sum\limits_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\times[\eta_{is}]^{\beta}}, & \text{if } j\in allowed_k\\ 0, & \text{otherwise}\end{cases} \tag{3}$$

其中,$\tau_{ij}(t)$表示路径 $task_i$ 和 VM_j 之间的信息素浓度。符号 $allowed_k$ 表示 ant_k 的允许虚拟机。η_{ij} 表示可见性或启发式信息,表示 VM_j 上任务 i 的预期执行时间。最后,两个参数 α 和 β 分别控制信息素的权重和可见度信息。蚂蚁完成旅程之后将检查 *BSolution*,并调用 MentionAdvert()模块。蚂蚁还在蚂蚁经过的边缘(i,j)上放置少量的信息素。这个过程称为局部信息素更新。

2.4 Individuals()

Individuals()模块的伪代码如图 8 所示,模块演示了遗传算法解决问题的方式。

```
Individuals()
1. 对参数进行编码,并选择合适的适应度函数
2. 根据式(4)计算某一个体被选中的概率,选出适应度较高的个体
3. 对于被选中用于繁殖的个体,随机选择两个个体的相同位置,按交叉概率
   Pc 在选中的位置实行交换
4. 基于生物在遗传中基因变异的原理,按概率 Pm 对某些个体的某些位执行
   变异操作,从而产生新的个体
5. If (获得解适应度>给定阈值)接受所得解
       检查更新 BSolution
       执行 MentionAdvert()
   Else
       Number_of_Stagnation 加 1
   End If
6. If(Number_of_Stagnation>Max_Number_of_Stagnation)
     随机选择第 1、第 3 或第 4 个 CommonBoard 的解,重置 Number_of_
     Stagnation 为 0
     End If
返回
```

图 8 Individuals()的伪代码

GA 中每个个体被选中的概率根据式(4)计算:

$$Pi=\frac{Fi}{\sum\limits_{i=1}^{M}Fi} \tag{4}$$

操作思路是:某一个体被选中的概率与其适应度大小成正比例关系。设群体大小为 M,个体 i 的适应度为 Fi。

从式(4)可以看出:适应度越高的个体保留到下一代参与

繁殖的机会越大；相反，适应度较低的个体获得的机会较小，还有可能会被淘汰。这样即可选择出与最终所求最优解接近的中间解。

3　仿真实验及分析

任务调度是云计算中最重要的问题。本文提出了一种云虚拟机上用于动态任务调度的混合算法，提出了将任务分配给可用的 VM，实现了最小化完成时间并提高了 VM 利用率，使用户更满意。下面介绍评估和实验结果。

3.1　云环境的参数设置

使用 CloudSim 软件构建仿真云环境进行实验，因为 CloudSim 可用于仿真大型云平台的数据中心、主机、服务代理、调度和分配策略。云仿真器的参数设置如表 1 所列。

表 1　CloudSim 中云环境的参数设置

实体类型	参数	值
任务（cloudlets）	任务长度	1000～50000
	任务总数	100～1000
虚拟机	虚拟机总数	25～50
	MIPS	500～2000
	VM 内存(RAM)	128～2048
	带宽	500～1000
	PE 需求量	1～4
数据中心	数据中心数	5
	主机数	10

3.2　混合算法的参数设置

混合算法的控制参数（$Number\_of\_BeesAntsIndividuals$，$Number\_of\_Bees$，$Number\_of\_Ants$，$Number\_of\_Individuals$，$Number\_of\_CommonBoard$，$Max\_Number\_of\_Stagnation$，$V\_MAX$，$\alpha$，$\beta$，$t_{\max}$）敏感，必须进行微调。

测试每个参数的几个值，而所有其他值保持不变。表 2 列出了实验确定的适合所提混合算法的参数的值。所提混合算法的参数设置确定为：$Number\_of\_BeesAntsIndividuals=50$，$Number\_of\_Bees=25$，$Number\_of\_Individuals=20$，$Number\_of\_Ants=5$，$Number\_of\_CommonBoard=4$，$Max\_Number\_of\_Stagnation=10$，$V\_MAX=5$，$\alpha=0.2$，$\beta=0.8$，$t_{\max}=100$。

表 2　混合算法的参数设置

参数	值
$Number\_of\_BeesAntsIndividuals$	50
$Number\_of\_Bees$	25
$Number\_of\_Ants$	5
$Number\_of\_Individuals$	20
$Number\_of\_CommonBoard$	4
$Max\_Number\_of\_Stagnation$	10
V_MAX	5
α	0.2
β	0.8
$t_{\max}$	100

3.3　混合算法、ABC 算法、GA 算法和 ACO 算法的实验结果

本节实验中要比较的云任务调度算法包括 ABC 算法、GA 算法、ACO 算法和提出的混合算法。ABC 算法的参数设置如下：$Number\_of\_Bees=100$，$Number\_of\_active=75$，$Number\_of\_Scout=15$，$Number\_of\_Inactive=10$，$Max\_number\_of\_Vists=70$，$Prob\_Mistake=0.1$，$Prob_Presuasion=0.90$和 $t_{\max}=100$。GA 算法的参数设置如下：$Number\_of\_Individuals=50$，$P_c=0.6$，$P_m=0.01$，如文献[8]所示。ACO 算法的参数设置如下：m(蚂蚁数)$=10$，$t_{\max}=100$，$\alpha=0.3$，$\beta=1$，Q(自适应参数)$=100$，$\rho=0.4$，如文献[10]所示。实验中计算了具有不同任务集的每个算法的完成时间。图 9 给出了混合算法、ABC、GA 和 ACO 完成时间的平均值。从图 9 可以看出，当任务数增加时，混合算法比 ABC，GA 和 ACO 算法花费更少的时间，这表明所提出的算法所花费的时间比其他方法要少。

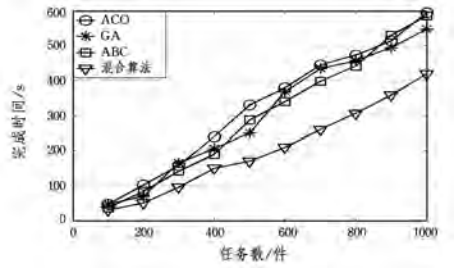

图 9　ACO，GA，ABC 和混合算法的平均完成时间

混合算法建模蜜蜂、种群进化和蚁群的行为，以解决云任务调度问题。它跟踪由任何成员创建的与完成时间长度相关联的整体最佳解。同时，它保存了由相似组成员发现的最佳解，并在外部 CommonBoards 中提供信息与其他两组共享，同时不断更新信息素。相似成员的组具有不同的活动：蜜蜂组模拟蜂群的觅食行为；群体中的个体组模拟自然界适者生存的规则进行迭代，并指导它们从环境中朝向正确的区域运动；蚂蚁组模拟蚂蚁有效觅食行为，试图寻找丰富的食物源。蜜蜂从特定区域持续聚合邻居解，直到该场所被消耗殆尽。之后，它们检查 CommonBoards，选择另一个丰富的区域。个体进行基于生物进化的迭代寻求最优解。蚂蚁利用一种特殊的化学信息素来相互通信并与蜜蜂和个体接触，它们通过在允许的路径上感应信息素来构建解。任何蚂蚁完成旅程之后，放置一些本地信息素更新信息素。全局信息素更新加强了任何成员(蚂蚁、蜜蜂或个体)属于最佳解的边缘信息素，这是混合算法优于其他算法的原因。不平衡程度(DI)衡量了虚拟机之间的不平衡。DI 值较小表明系统的负载更加平衡、有效。可以用 3 种不同的方法测量 DI。第一种方法如式(5)所示，测量定义的最大和最小 VM 的完成时间的差值。

$$DI_1=AT_{\max}-AT_{\min} \tag{5}$$

其中，$AT_{\max}$ 和 $AT_{\min}$ 表示 VM 的最大和最小完成时间。混合算法、ABC、GA 和 ACO 的 DI_1 如图 10 所示。

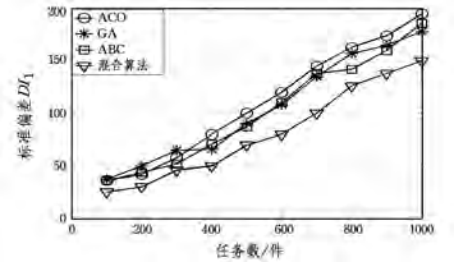

图 10　ACO，GA，ABC 和混合算法的 DI_1

第二种方法衡量不平衡程度，如式(6)所示：

$$DI_2=\frac{T_{\max}-T_{\min}}{T_{avg}} \tag{6}$$

其中，$T_{\max}$，$T_{\min}$ 和 T_{avg} 分别是最大、最小和平均 VM 的完成时间。混合算法、ABC、GA 和 ACO 的 DI_2 如图 11 所示。

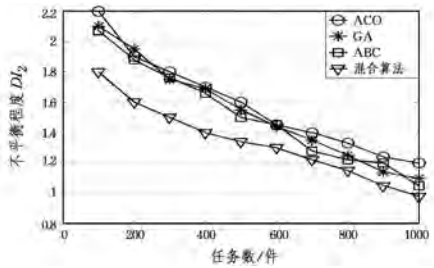

图 11 ACO,GA,ABC 和混合算法的 DI_2

第三种方法如式(7)所示,使用标准差来衡量不平衡度。

$$DI_3=\sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j-\bar{x})^2}\text{ ,for all } j\in VM \text{ list} \tag{7}$$

其中,DI_3 是标准差,N 是 VM 数,x_j 是 VM_j 的完成时间,$\bar{x}$ 是所有 VM 的平均完成时间。如果 DI_3 值很小,则意味着负载的差异很小,VM 上的负载更加平衡。

混合算法、ABC、GA 和 ACO 的 DI_3 如图 12 所示。

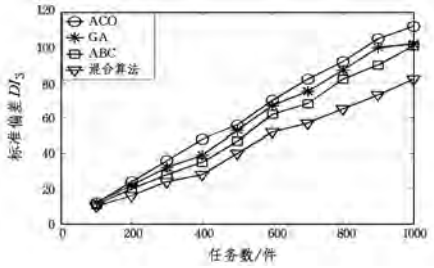

图 12 ACO,GA,ABC 和混合算法的 DI_3

从图 10—图 12 可以看出,混合算法可以实现比 ACO、GA 和 ABC 算法更好的负载平衡。数据中心的虚拟机具有不同的处理能力。提出了混合算法搜索一个解,首先将任务分配给最强大的虚拟机,然后再分配给最低平衡虚拟机负载的虚拟机。

结束语 本文提出了一种用于处理云任务调度的混合算法。所提出的混合算法利用了蚁群优化算法、遗传算法和人工蜂群的优点,建模蜜蜂、种群中个体和蚁群的行为,提出了共享机制模块来使混合算法成员之间共享解,实现快速收敛。最后对提出的混合算法、人工蜂群、蚁群优化、遗传算法进行仿真评估。仿真结果证明,提出的混合算法更好,实现了高资源利用率,并且在完成时间和不平衡程度上明显优于其他算法。

参 考 文 献

[1] 熊聪聪,郝璐萌,王丹,等.一种基于差分策略的群搜索优化算法[J].计算机科学,2017,44(2):250-256.

[2] TAWFEEK M A,ELSISI A,KESHK A E,et al. Cloud Task Scheduling Based on Ant Colony Optimization[J]. International Arab Journal of Information Technology (IAJIT),2015,12(2):129-137.

[3] TAWFEEK M A,ELSISI A,et al. An Ant Algorithm for Cloud Task Scheduling[C]// International Workshop on Cloud Computing and Information Security CCIS. 2013:169-172.

[4] NISHANT K,SHARMA P,KRISHNA V,et al. Load Balancing of Nodes in Cloud Using Ant Colony Optimization[C]// Uksim, International Conference on Modelling and Simulation. IEEE Computer Society,2012:3-8.

[5] 张伟哲,张宏莉,张迪,等.云计算平台中多虚拟机内存协同优化策略研究[J].计算机学报,2011,34(12):2265-2277.

[6] 卓涛,詹颖.改进人工蜂群算法的云计算资源调度模型[J].微电子学与计算机,2014,31(7):147-155.

[7] KAMBLE S V,MANE S U,UMBARKAR A J. Hybrid M ulti-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem [J]. International Journal of Intelligent Systems Technologies&Applications(IJISA),2015,7(4):54-61.

[8] 熊聪聪,冯龙,陈丽仙,等.云计算中基于遗传算法的任务调度算法研究[J].华中科技大学学报(自然科学版),2012,40(1):1-4.

[9] 姚婧,何聚厚.基于自适应蜂群算法的云计算负载平衡机制[J].计算机应用,2012,32(9):2448-2450.

[10] 查英华,杨静丽.改进蚁群算法在云计算任务调度中的应用[J].计算机工程与设计,2013,34(5):1716-1719.

[11] 李建锋,彭舰.云计算环境下基于改进遗传算法的任务调度算法[J].计算机应用,2011,31(1):184-186.

[12] BABU L D D,VENKATA KRISHNA P. Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments[J]. Applied Soft Computing,2013,13(5):2292-2303.

(上接第 269 页)

[8] ZENG Y H,LIANG Y C. Spectrum sensing algorithms for cognitive radio based on statistical covariances [J]. IEEE Transactions on Vehicular Technology,2009,58(4):1804-1815.

[9] FARHANG-BOROUJENY B. Filter bank spectrum sensing for cognitive radios [J]. IEEE Transactions on Signal Processing, 2008,56(5):1801-1811.

[10] RUGINI L,BANELLI P,LEUS G. Small sample size performance of the energy detector [J]. IEEE Communications Letters,2013,17(9):1814-1817.

[11] SHARMA B V,TELLAMBURA C,JIANG H. Approximations for performance of energy detector and p-norm detector [J]. IEEE Communications Letters,2015,19(10):1678-1681.

[12] REISI N,GAZOR S,AHMADIAN M. Distributed cooperative spectrum sensing in mixture of large and small scale fading channels [J]. IEEE Transactions on Wireless Communications, 2013,12(11):5406-5412.

[13] QUAN Z,CUI S G,SAYED A,et al. Optimal multiband joint detection for spectrum sensing in cognitive radio networks [J]. IEEE Transactions on signal processing, 2009, 57 (3): 1128-1140.

[14] ATAPATTU S,TELLAMBURA C,JIANG H. Energy detection for spectrum sensing in cognitive radio [M]. New York: Springer,2014.

[15] SIMON M,ALOUINI M. Digital communication over fading channels [M]. New York,USA:Wiley,2005.

[16] KAY S. Fundamentals of statistical signal processing:detection theory [M]. NJ U. S. A. :Prentice-Hall,1998.

[17] ADAMCHIK V S,MARICHEV O I. The algorithm for calculating integrals of hypergeometric type functions and its realization in reduce system [C]// International Symposium on Symbolic Algebraic Computing. 1990:212-224.

[18] GRADSHTEYN I S,RYZHIK I M. Table of Integrals,Series, and Products(8th ed)[M]. Amsterdam,the Netherlands:Elsevier,2015.