

# 融合粒子群与蚁群的云计算任务调度算法

查安民<sup>1</sup>, 谭文安<sup>1,2</sup>

(1. 南京航空航天大学 计算机科学与技术学院, 江苏 南京 210016;

2. 上海第二工业大学 计算机与信息学院, 上海 201209)

**摘要:**在云计算环境中用户数量众多,用户提交的任务总量非常庞大,如何调度这些海量任务使其高效合理地完成成为云计算研究的关键。针对云计算环境的特点,对粒子群和蚁群算法进行改进,提出一种融合二者的任务调度算法。该算法采用粒子群算法进行前期迭代,迭代完成后选取一定数量的优良粒子生成蚁群算法的初始信息素,蚁群算法利用已生成的初始信息素进行后期迭代,并求得最终的任务调度结果。仿真结果表明,该算法优于粒子群算法和蚁群算法,任务的总完成时间明显减少,是一种高效的调度算法。

**关键词:**云计算;任务调度;粒子群优化;蚁群优化

**中图分类号:**TP301.6

**文献标识码:**A

**文章编号:**1673-629X(2016)08-0024-06

doi:10.3969/j.issn.1673-629X.2016.08.005

## A Task Scheduling Algorithm of Cloud Computing Merging Particle Swarm Optimization and Ant Colony Optimization

ZHA An-min<sup>1</sup>, TAN Wen-an<sup>1,2</sup>

(1. College of Computer Science and Technology, Nanjing University of Aeronautics and

Astronautics, Nanjing 210016, China;

2. School of Computer and Information, Shanghai Second Polytechnic University, Shanghai 201209, China)

**Abstract:** In cloud computing environment, there are a large number of users and tasks to be submitted by users. In order to make these tasks to be completed efficiently, how to schedule the tasks becomes the key of cloud computing. According to characteristics of cloud computing environment, improving Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), a task scheduling algorithm combining PSO with ACO. It uses PSO to carry out the previous iteration, and selects a certain number of fine particles to generate the initial pheromone of ACO which carries out the post iteration by it, and then the final task scheduling result is obtained. The simulation shows that the algorithm is better than PSO and ACO, and decreases the total task completion time. It is an effective task scheduling algorithm.

**Key words:** cloud computing; task scheduling; PSO; ACO

## 0 引言

自从云计算概念提出之后,云计算技术就获得了飞速发展。IBM、Google、Amazon 等大公司纷纷投入巨资打造自己的云计算产品。云计算作为一种技术和商业模式,实现了对共享、可配置计算资源的方便、按需访问。云计算可为用户提供3类服务:

(1) 基础设置即服务 (Infrastructure as a Service, IaaS), 如 IBM Blue Cloud、Amazon S3。

(2) 平台即服务 (Platform as a Service, PaaS), 如 Google App Engine、Apache Hadoop。

(3) 软件即服务 (Software as a Service, SaaS), 如 Google Apps、Salesforce CRM。

云计算技术的发展总是伴随着大规模任务调度的研究。因此,如何对云计算环境中的大规模任务进行高效调度,成为了一个亟待解决且具有挑战性的问题。

云计算环境中的任务调度是一个难点<sup>[1]</sup>。传统调

收稿日期:2015-11-27

修回日期:2016-03-08

网络出版时间:2016-08-01

基金项目:国家自然科学基金资助项目(6127036);上海第二工业大学重点学科(XXXZD1301)

作者简介:查安民(1987-),男,硕士研究生,研究方向为云计算、 workflow调度与服务计算;谭文安,博士,教授,CCF 高级会员,通讯作者,研究方向为软件构架技术、协同计算、服务计算与知识管理、信息化工程及其支持环境研究等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160801.0904.032.html>

度算法,如先进先出、短作业优先等,存在诸多问题<sup>[2]</sup>。智能优化算法成为求解此类问题的新工具。文献[3]运用遗传算法调度云计算任务,取得了较好的实验效果。文献[4]分析并改进粒子群算法,在一定程度上减少了任务完成时间。文献[5]也以减少任务完成时间为目的,对蚁群算法进行改进,同样取得了较好的实验效果。智能优化算法有其优点,但是也有不足之处。如何取长补短,将两种或多种智能算法进行融合,成为研究智能算法的一个重要方面。一些专家和学者不但给出智能算法融合的实际应用,而且给出了它们融合的理论基础,这些都成为运用智能算法调度云任务的坚实基础。

众所周知,粒子群算法具有前期收敛速度快但后期收敛精度低的特点,而蚁群算法具有全局寻优能力强但需要花费较长迭代时间的特点。基于此,文中提出一种融合二者优点的任务调度算法,大大减少了任务的完成时间。

## 1 云计算任务调度的描述

云计算任务的执行是建立在“分解”与“调度”基础上的<sup>[6]</sup>。即将一个较大的任务分解为若干较小的子任务,然后为这些子任务分配一定数量的资源节点使其并行执行,最后将运行结果返回给用户<sup>[7]</sup>。假设任务之间相互独立运行。任务调度所要追求的就是对各个任务合理分配资源,使得总的任务完成时间最小。

### 1.1 粒子编码

为了运用粒子群和蚁群算法,首先需要对粒子进行编码。为简单起见,可选择直接法。

设有  $a$  个任务,  $r$  个资源,并且  $a > r$ 。当  $a = 10, r = 3$  时,编码序列(3,2,2,1,3,2,3,1,1,2)即为一个粒子。其中,每个任务对应一个资源。例如,任务3对应资源1。

接着对粒子进行解码,获取每个资源运行的所有任务。例如,资源1上运行的所有任务为{4,8,9}。

使用矩阵  $\mathbf{ETC}_{ij}$  保存任务  $i$  在资源  $j$  上的期望执行时间:

$$\mathbf{ETC}_{ij} = \begin{pmatrix} \text{ETC}_{11} & \cdots & \text{ETC}_{1r} \\ \vdots & \ddots & \vdots \\ \text{ETC}_{a1} & \cdots & \text{ETC}_{ar} \end{pmatrix}$$

使用矩阵  $\mathbf{RTC}_{ij}$  保存任务  $i$  在资源  $j$  上的实际执行时间:

$$\mathbf{RTC}_{ij} = \begin{pmatrix} \text{RTC}_{11} & \cdots & \text{RTC}_{1r} \\ \vdots & \ddots & \vdots \\ \text{RTC}_{a1} & \cdots & \text{RTC}_{ar} \end{pmatrix}$$

$$\text{resourceTime}(j) = \sum_{i=1}^a \text{RTC}_{ij} (j = 1, 2, \dots, r) \quad (1)$$

$$\text{taskTime} = \max(\text{resourceTime}(j)) (j = 1, 2, \dots, r) \quad (2)$$

式(1)表示资源  $j$  上完成所有任务的总时间。其中,  $i$  表示资源  $j$  上执行的第  $i$  个任务;  $a$  表示资源  $j$  上执行的任务总数。

式(2)表示所有任务完成后的总时间。

### 1.2 适应度函数

每个粒子都有一个用于衡量其优劣的适应度。适应度函数的选取并不是固定不变的,需要根据待求问题、优化目标、计算成本等要求合理选择。一般来说,适应度函数应尽量满足单调、非负、最大化等特点。

文中需要优化任务总的完成时间,因此将适应度函数定义为:

$$\text{fitness}(i) = \frac{1}{\text{taskTime}(i)} (i = 1, 2, \dots, s) \quad (3)$$

式中,  $i$  表示第  $i$  个粒子;  $s$  表示种群规模。

## 2 云计算任务调度的设计

粒子群算法<sup>[8]</sup>的基本思想是,每个粒子在寻找最优解过程中会充分考虑两个极值。第一个是粒子自身所能找到的位置极值,也叫个体最优解;第二个则是整个种群所能找到的位置极值,也叫全局最优解。即每个粒子在一定范围的寻优过程中,不但考虑了自身的局部认知,而且考虑了社会的全局认知。

蚁群算法<sup>[9]</sup>的基本思想是,让每只蚂蚁在一定范围内独立寻找最优解,当蚂蚁遇到一个带有多条路径的路口时,如果以前从来没有其他蚂蚁走过,就任意选择一条作为当前路径继续进行搜索,并且释放一定数量的信息素。随着迭代次数的增多,代表最优解的路径上信息素会越来越多,其他路径上则会越来越少,最终蚁群算法趋向于稳定。

粒子群算法的最大特点是前期收敛速度快但后期收敛精度低;而蚁群算法的最大特点是全局寻优能力强但系统开销大。基于两种算法的特点,文中提出一种融合二者优点的任务调度算法(PSACO2)。该算法的基本思想是,将迭代过程分为两个部分,前期使用粒子群算法,后期使用蚁群算法。这样既可实现云计算任务调度的快速收敛,又可提高云计算任务调度对资源的寻优能力。

### 2.1 粒子群算法

#### 2.1.1 粒子群初始化

设种群规模为  $s$ , 任务总数为  $a$ , 资源总数为  $r$ , 则初始化描述为:系统随机产生  $s$  个粒子,向量  $\mathbf{x}_i$  表示粒子  $i$  的位置,定义  $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{ia}\} (1 \leq i \leq s)$ 。其中,  $x_{ij}$  表示任务  $i$  分配到资源  $j$  上执行 ( $1 \leq x_{ij} \leq r$ )。向量  $\mathbf{v}_i$  表示粒子  $i$  的速度,定义  $\mathbf{v}_i = \{v_{i1}, v_{i2},$

$\dots, v_{in}\} (1 \leq n \leq a, 1 \leq i \leq s)$ 。其中  $-r \leq v_{ij} \leq r$ 。粒子初始位置为  $[1, r]$  之间的随机整数, 粒子速度随机取  $[-r, r]$  内的整数。

### 2.1.2 粒子位置与速度更新

根据式(4)、(5)更新粒子的位置和速度。

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t)) \quad (4)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (5)$$

其中,  $\omega$  为惯性权重;  $c_1$  和  $c_2$  为加速系数;  $r_1$  和  $r_2$  为随机因子。

为了提高 PSO 算法的智能性, 提出了很多关于惯性权重  $\omega$  的改进方法。例如, 文献[10-13]分别提出了线性递减权重、线性微分递减权重、带阈值的非线性递减权重、带控制因子的非线性递减权重等思想。此处将  $\omega$  定义为<sup>[14]</sup>:

$$\omega(t) = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \cdot \exp(-k(t/t_{\max}))^2 \quad (6)$$

其中,  $k$  为控制因子;  $\omega_{\min}$  与  $\omega_{\max}$  分别为最小与最大惯性权重;  $t$  为迭代次数;  $t_{\max}$  为最大迭代次数。

针对学习因子  $c_1$ 、 $c_2$  的改进主要集中在两个方面: 一个是线性调整策略, 另一个是非线性调整策略。此处将  $c_1$ 、 $c_2$  分别定义为<sup>[15]</sup>:

$$c_1 = c_{1s} + (c_{1e} - c_{1s}) \times (t/t_{\max}) \quad (7)$$

$$c_2 = c_{2s} + (c_{2e} - c_{2s}) \times (t/t_{\max}) \quad (8)$$

其中,  $c_{1s}$  为  $c_1$  的初始迭代值;  $c_{1e}$  为  $c_1$  的最终迭代值;  $c_{2s}$  为  $c_2$  的初始迭代值;  $c_{2e}$  为  $c_2$  的最终迭代值。

为了进一步提高初始信息素的生成质量和后期蚁群算法的收敛速度, 可对前期粒子群算法进行部分杂交优化<sup>[16]</sup>。具体做法是, 根据适应值大小给每个粒子一定大小的选择概率, 将选择出来的粒子进行杂交变异。经杂交操作产生的子代位置为:

$$x_{\text{child},1}(t) = p x_{\text{parent},1}(t) + (1.0 - p) x_{\text{parent},2}(t) \quad (9)$$

$$x_{\text{child},2}(t) = p x_{\text{parent},2}(t) + (1.0 - p) x_{\text{parent},1}(t) \quad (10)$$

其中,  $x_{\text{child}}$  为子代粒子的位置;  $x_{\text{parent}}$  为父代粒子的位置;  $p$  为  $[0, 1]$  之间的随机数。

子代粒子的速度为:

$$v_{\text{child},1}(t) = \frac{v_{\text{parent},1}(t) + v_{\text{parent},2}(t)}{|v_{\text{parent},1}(t) + v_{\text{parent},2}(t)|} v_{\text{parent},1}(t) \quad (11)$$

$$v_{\text{child},2}(t) = \frac{v_{\text{parent},1}(t) + v_{\text{parent},2}(t)}{|v_{\text{parent},1}(t) + v_{\text{parent},2}(t)|} v_{\text{parent},2}(t) \quad (12)$$

其中,  $v_{\text{child}}$  为子代粒子的速度;  $v_{\text{parent}}$  为父代粒子的速度。

变异操作则采用逆转变异方法。

## 2.2 蚁群算法

为了使用蚁群算法调度云计算任务, 需要将所有的资源节点抽象成一个有向完全图。假设云计算环境

中有 4 个资源, 6 个任务, 则由资源抽象而成的有向完全图见图 1。

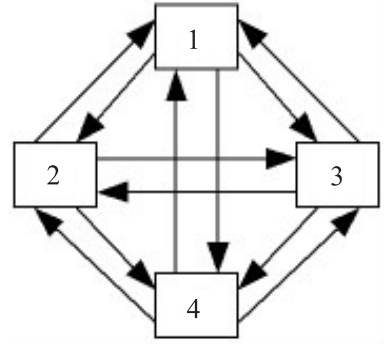


图 1 资源节点组成的有向完全图

图中, 每个节点都有入度与出度和入度边与出度边, 每条边上的权值表示与入度节点相关的信息素值。调度开始时, 将所有任务随机放置在资源节点上。

### 2.2.1 信息素初始化

信息素的初值设为:

$$\tau_s = \tau_c + \tau_g \quad (13)$$

其中,  $\tau_c$  是一个给定的信息素常数;  $\tau_g$  是一个根据前期粒子群算法的调度结果转换而来的信息素值。

具体做法是, 选取粒子群算法终止时种群中适应值最好的前 10% 个体作为优化解集合。开始时, 设置  $\tau_g$  为 0, 对于优化解集合中的每个解,  $\tau_g$  自加常数  $k$ 。

### 2.2.2 路径选择

第  $k$  只蚂蚁在  $t$  时刻从资源节点  $i$  转移到资源节点  $j$  的概率设为:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{s \in \text{allowed}_i} [\tau_{is}(t)]^\alpha [\eta_{is}(t)]^\beta}, & j \in \text{allowed}_k \\ 0, & \text{else} \end{cases} \quad (14)$$

其中,  $\text{allowed}_k = \{C - \text{tabu}_k\}$  表示在  $t+1$  时刻蚂蚁  $k$  允许选择的资源节点,  $\text{tabu}_k$  表示禁忌表, 保存蚂蚁  $k$  曾经搜索过的资源;  $\eta_{ij}$  表示启发信息;  $\tau_{ij}$  表示残留的信息量;  $\alpha$  和  $\beta$  分别表示残留信息和启发信息对蚁群寻优过程的相对重要程度。

$\eta_{ij}$  的取值为:

$$\eta_{ij}(t) = \frac{1}{\text{ETC}_{ij}^k(t)} = \frac{1}{\text{ETC}_{kj}} \quad (15)$$

### 2.2.3 更新信息素

当蚂蚁  $k$  从一个节点  $i$  转移到另一个节点  $j$  时, 节点的信息素会发生变化。及时更新信息素有利于提高蚁群算法的收敛精度。式(16)和式(17)给出了局部更新信息素的方法:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \sum_{k=1}^a \Delta \tau_{ij}^k(t) \quad (16)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{ETC_{ij}^k(t)} = \frac{Q}{ETC_{kj}}, & \text{第 } k \text{ 只蚂蚁在 } t \text{ 和 } t+1 \text{ 之间选择 } x_{ij} \\ 0, & \text{else} \end{cases} \quad (17)$$

其中,  $\rho$  为  $[0,1]$  之间的随机数, 表示信息素挥发系数;  $1-\rho$  为信息素的残留系数;  $\Delta\tau_{ij}^k$  表示蚂蚁  $k$  在当前路径转移中留在路径  $(i,j)$  上的信息素数量;  $Q$  为常量。

当所有蚂蚁完成一次循环, 需对信息素进行全局更新, 公式如下:

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^a \Delta\tau_{ij}^k(t) \quad (18)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{\text{resourceTime}(j)}, & \text{第 } k \text{ 只蚂蚁本次循环中选择 } x_{ij} \\ 0, & \text{else} \end{cases} \quad (19)$$

### 2.3 算法流程

下面给出 PSACO2 算法的详细描述。

输入: 随机产生的初始种群;

输出: 全局最优解。

- (1) 给出算法中相关参数的值;
- (2) 根据要求对粒子进行编码并初始化种群;
- (3) 将种群随机分成两个等量的子群, 并分别编号为 1 和 2;
- (4) 针对当前子群是 1 号还是 2 号分别进行不同处理;
- (5) 如果是 1 号子群, 使用粒子群算法更新粒子的位置和速度;
- (6) 如果是 2 号子群, 使用遗传算法更新粒子的位置和速度;
- (7) 将两个子群重新合并成一个种群;
- (8) 判断前期迭代是否满足停止条件或达到迭代次数;
- (9) 若否, 则返回步骤 (3); 若是, 则执行步骤 (10);
- (10) 从步骤 (7) 中选取适应值最好的前 10% 个体, 并根据式 (13) 生成蚁群算法的初始信息素;
- (11) 建立蚁群算法任务调度模型, 并初始化算法参数;
- (12) 每只蚂蚁根据式 (14) 选择转移节点, 根据式 (16) 和 (17) 进行局部信息素更新, 并将所选节点加入禁忌表中;
- (13) 当所有蚂蚁完成一次循环后, 根据式 (18) 和 (19) 进行全局信息素更新;

- (14) 判断后期迭代是否满足停止条件或达到迭代次数;
- (15) 若否, 则返回步骤 (12); 若是, 则输出全局最优解。

PSACO2 算法的流程图如图 2 所示。

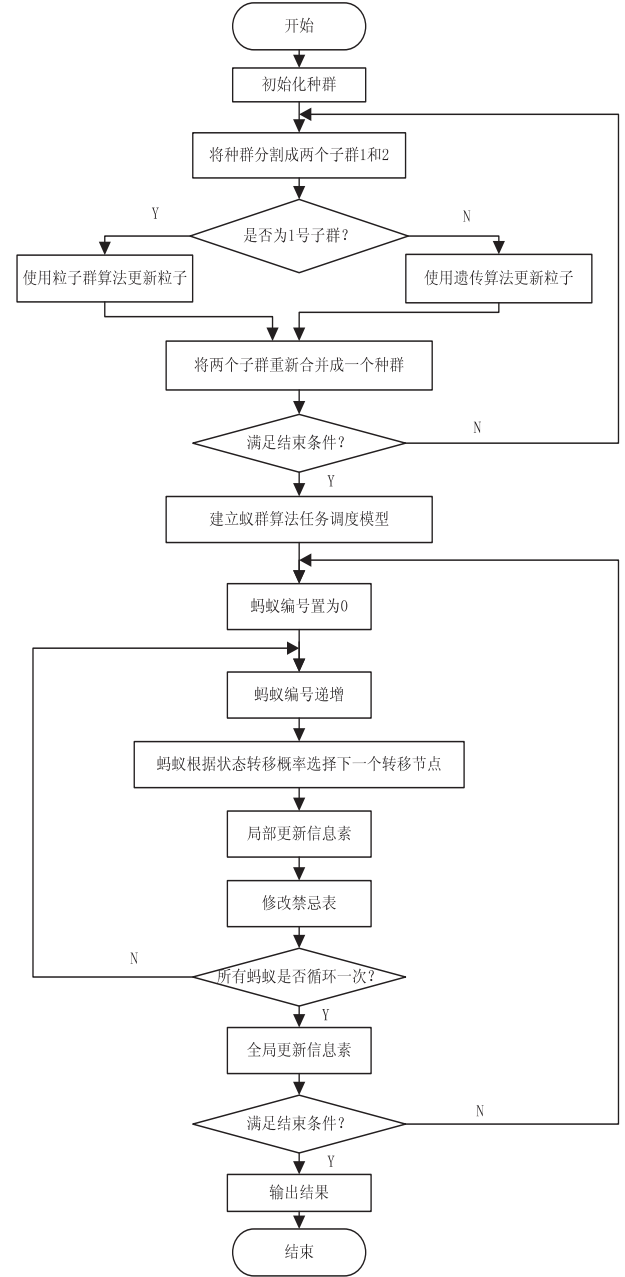


图2 PSACO2 算法流程图

### 2.4 PSACO2 算法的复杂度分析

PSACO2 算法分为两个阶段, 第一阶段使用带遗传算子的粒子群算法, 第二阶段使用蚁群算法。设  $m$  为任务数量,  $n$  为资源数量,  $s$  为粒子数量,  $N_1$  为第一个阶段的迭代次数, 第一个阶段的最大迭代次数为  $N_{1\max}$ ,  $N_2$  为第二个阶段的迭代次数, 第二个阶段的最大迭代次数为  $N_{2\max}$ 。为了分析 PSACO2 算法的时间复杂度, 首先分析粒子群算法的时间复杂度, 见表 1。



遗传算法的时间复杂度的分析过程同粒子群算法,下面分析蚁群算法的时间复杂度,见表 2。

表 1 粒子群算法时间复杂度

Step	内容	$O(n)$
1	初始化	
	set $N_1 = 0$ ;	$O(m \times s)$
	随机产生 $s$ 个粒子	
2	计算粒子的适应值	
	$N_1 = N_1 + 1$ ;	$O(m \times s)$
	计算适应值	
3	更新个体最优解	$O(s)$
4	更新全局最优解	$O(s)$
5	更新粒子的位置和速度	$O(m \times s)$
6	判断是否满足终止条件	
	如果 $N_1 \leq N_{1\max}$	
	返回 Step2	$O(1)$
	否则输出结果	

表 2 蚁群算法时间复杂度

Step	内容	$O(n)$
1	初始化	
	set $t = 0$ ; // $t$ 为时间	
	set $N_2 = 0$ ;	$O(m \times s + n^2)$
	设置初始信息素	
2	初始化禁忌表	$O(m)$
	对于每只蚂蚁	
	选择转移节点	
	局部更新信息素	$O(n^2 m)$
3	修改禁忌表	
	全局更新信息素	
	判断是否满足终止条件	$O(n^2)$
	如果 $N_2 \leq N_{2\max}$	
4	清空禁忌表	
	返回 Step2	$O(nm)$
	否则输出结果	

由于在云计算环境中,任务的数量要大于节点的数量,所以 PSACO2 算法的时间复杂度约为  $O(n) = O(N_1 \times m \times s) + O(N_2 \times n^2 m)$ 。

3 实验与分析

实验以 Cloudsim<sup>[17]</sup> 作为仿真平台、Eclipse 作为开发工具以及 Java 作为开发语言,将提出的 PSACO2 算法与标准 ACO 算法、标准 PSO 算法进行对比分析。实验需要设置的参数如表 3 所示。

表 3 PSACO2 算法参数

参数名称	参数符号	参数取值
种群规模	$s$	100
资源数量	$r$	5
任务数量	$a$	50,500
控制因子	$k$	3.0

续表 3

参数名称	参数符号	参数取值
最大惯性权重	$\omega_{\max}$	0.9
最小惯性权重	$\omega_{\min}$	0.4
$c_1$ 的初始迭代值	$c_{1s}$	2.0
$c_1$ 的最终迭代值	$c_{1e}$	0.5
$c_2$ 的初始迭代值	$c_{2s}$	1.5
$c_2$ 的最终迭代值	$c_{2e}$	3.0
交叉概率	$P_c$	0.9
变异概率	$P_m$	0.02
第一个阶段的最大迭代次数	$t_{1\max}$	50
第二个阶段的最大迭代次数	$t_{2\max}$	150
启发式因子	$\alpha$	1.0
	$\beta$	1.0
挥发系数	$\rho$	0.2

蚁群算法与粒子群算法所需的参数来源表 3。每种算法的迭代次数为 200。实验结果见图 3 和图 4。

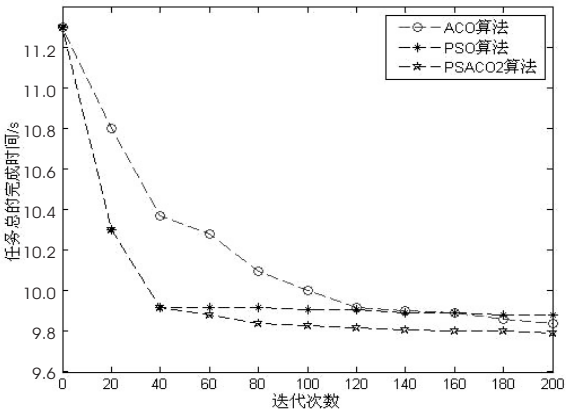


图 3  $a = 50$  时的任务总完成时间

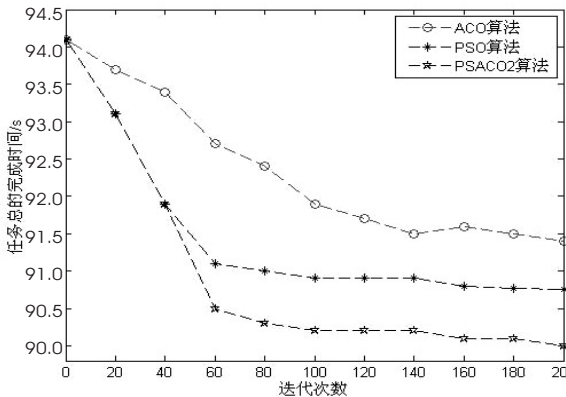


图 4  $a = 500$  时的任务总完成时间

从实验结果可知,PSACO2 调度算法充分吸收了 PSO 算法和 ACO 算法的优点,达到了预期的实验效果。在迭代之初,PSACO2 算法的调度效果优于 ACO 算法,完成一定数量任务所需的执行时间小于 ACO 算法;在迭代后期,PSACO2 算法的调度效果同时优于 PSO 和 ACO 算法,完成一定数量任务所需的执行时间最少。同时还可得出,相比 PSO 算法和 ACO 算法,PSACO2 算法能在相对较小的迭代次数内找到最优解。

4 结束语

文中通过分析 PSO 算法和 ACO 算法的优缺点,提出一种融合二者优点的 PSACO2 算法。仿真结果表明,该算法优于其中每种算法在单独使用时的效果,是一种高效可行的任务调度算法。

PSACO2 算法只是优化了任务完成时间,而在真实环境中还需考虑其他因素,比如平均完成时间、经济效益、负载均衡等。因此有必要进一步研究和改进 PSACO2 算法,以便拓宽算法的应用场景。

参考文献:

[1] Arfeen M A, Pawlikowski K, Willig A. A framework for resource allocation strategies in cloud computing environment [C]//Proc of IEEE 35th annual computer software and applications conference. [s. l. ]:IEEE,2011:261-266.

[2] 李千目,张晟骁,陆路,等.一种 Hadoop 平台下的调度算法及混合调度策略[J]. 计算机研究与发展,2013,50(S1):361-368.

[3] Zhao Chenhong, Zhang Shanshan, Liu Qingfeng, et al. Independent tasks scheduling based on genetic algorithm in cloud computing[C]//Proc of IEEE 5th international conference on wireless communications, networking and mobile computing. Beijing:IEEE,2009:1-4.

[4] Guo Lizheng, Zhao Shuguang, Shen Shigen, et al. Task scheduling optimization in cloud computing based on heuristic algorithm[J]. Journal of Networks,2012,7(3):547-553.

[5] Li Jianfeng, Peng Jian, Cao Xiaoyang, et al. A task scheduling algorithm based on improved ant colony optimization in cloud computing environment[J]. Energy Procedia,2011(13):6833-6840.

[6] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]//Proceedings of the 6th symposium on operating system design and implementation. New York:ACM,2004:137-150.

[7] 周墨颂,朱正东,董小社,等.采用资源划分的云环境下 Hadoop 资源许可调度方法[J]. 西安交通大学学报,2015,49(8):69-74.

[8] Kennedy J, Eberhart R C. Particle swarm optimization[C]//Proc of IEEE international conference on neural networks. Piscataway, NJ:IEEE Service Center,1995:1942-1948.

[9] Dorigo M, Maniezzo V, Colomi A. The ant system: optimization by a colony of cooperating agent[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B,1996,26(1):29-41.

[10] Shi Y, Eberhart R C. A modified particle swarm optimizer [C]//Proc of the 1998 IEEE ECP. Piscataway, NJ:IEEE,1998:67-93.

[11] 胡建秀,曾建潮.微粒群算法中惯性权重的调整策略[J]. 计算机工程,2007,33(11):193-195.

[12] 王丽,王晓凯.一种非线性改变惯性权重的粒子群算法[J]. 计算机工程与应用,2007,43(4):47-48.

[13] 李会荣,高岳林,李济民.一种非线性递减惯性权重策略的粒子群算法优化算法[J]. 商洛学院学报,2007,21(4):16-20.

[14] 李丽,牛奔.粒子群优化算法[M]. 北京:冶金工业出版社,2009.

[15] Ratnawecra A, Halgamuge S. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. Evolutionary Computation,2004,8(3):240-255.

[16] 纪震,廖惠连.粒子群算法及应用[M]. 北京:科学出版社,2009.

[17] Calheiros R N, Ranjan R, Rose C A F D, et al. CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services[EB/OL]. [2009-09-03]. <http://arxiv.org/abs/0903.2525>.