基于改进蚁群算法的云计算资源分配策略研究

刘灯明,荆俊峰,刘 凯,房志奇
(华北计算机系统工程研究所,北京 100083)

摘 要:在实际的项目中会发现蚁群算法直接应用于云计算资源分配时经常会出现负载失衡的情况,导致资源利用率不高,同时导致任务完成时间太长,算法迭代次数过大。这种情况不仅会大大地降低云计算系统的效率,还会造成系统不稳定。因此针对蚁群算法进行了一系列改进,具体包括:引入伪随机比例规则,进行全局信息素强化,引入了交叉变异操作,将蚁群算法与遗传算法相融合。然后进行了MATLAB仿真实验,实验结果表明:改进算法的任务完成时间更短,算法迭代次数更少,负载均衡效果更好。由此可以得出结论:对蚁群算法的改进是有效的。

关键词:蚁群算法;改进;云计算;负载均衡
中图分类号:TP39 **文献标识码**:A DOI:10.16157/j.issn.0258-7998.211725

中文引用格式:刘灯明,荆俊峰,刘凯,等. 基于改进蚁群算法的云计算资源分配策略研究[J].电子技术应用,2022,48(5):104-109.
英文引用格式:Liu Dengming,Jing Junfeng,Liu Kai,et al. Research on cloud computing resource allocation strategy based on improved ant colony algorithm[J]. Application of Electronic Technique,2022,48(5):104-109.

Research on cloud computing resource allocation strategy based on improved ant colony algorithm

Liu Dengming,Jing Junfeng,Liu Kai,Fang Zhiqi
(North China Institute of Computer Systems Engineering,Beijing 100083,China)

Abstract:In actual projects,it is found that if the ant colony algorithm is directly applied to cloud computing resource allocation,there will often be load imbalances,resulting in low resource utilization. And at the same time,the task completion time is too long,and the number of algorithm iterations is too large. This situation will not only greatly reduce the efficiency of the cloud computing system,but also cause system instability. Therefore,this article has made a series of improvements to the ant colony algorithm,including:the introduction of pseudo-random proportional rules,global pheromone enhancement,the introduction of cross mutation operations,and integration of ant colony algorithm and genetic algorithm. And then MATLAB simulation experiments are carried out.The experimental results show that the task completion time of the improved algorithm is shorter,the number of algorithm iterations is less,and the load balancing effect is better. From this,it can be concluded that the ant colony algorithm is better. The improvement is effective.

Key words:ant colony algorithm;improvement;cloud computing;load balancing

0 引言

现代社会进入了大数据时代,传统的计算模式存在很多局限性,不能够满足这种大数据的处理需求,因此“云计算”应运而生[1]。云计算中一个十分关键的问题就是负载均衡,负载均衡的含义是把任务平均地分配到云计算系统中的各个资源点上,所以设计出高效合理的资源分配策略非常重要[2]。目前资源分配策略的相关研究已经取得了不错的研究成果,例如:谭一鸣等人提出了一种能够降低系统能耗的资源分配策略,李安南创新性地提出了一种QoS约束简化的资源分配策略[3]。在云计算资源分配策略中采用了各种算法,例如:蚁群算法。蚁群算法有很多优点,因此经常被应用到云计算资源分配问题上[4]。然而在实际的项目中会发现蚁群算法直接应用于云计算资源分配问题时效果不好,常常会出现负载失衡,所以本文针对蚁群算法进行了一系列改进,对蚁群算法进行改进方面的研究是本文的研究重点,改进后进行了实验,实验结果表明:对蚁群算法的改进是有效的。

1 蚁群算法

1.1 蚁群算法的原理

蚁群算法启发于蚂蚁寻找食物的过程,蚂蚁视力很差但却能够轻松地找到食物源到蚁穴的最优路径,也就是最短路径,相关研究发现蚂蚁会在路径上释放信息素

来进行通信[5]，某条路径上的信息素越多，对蚂蚁的吸引力就越强，蚂蚁会选择走该条信息素多的路径。与此同时，会把信息素释放在该条路径上，该路径上的信息素将得到进一步增加，反过来提升了对蚂蚁的吸引力，这里面存在一种正反馈过程，随着这个过程不断进行，最终的结果是会收敛得到某条最优路径，于是便确定了蚁穴到食物源的最短路径[6]。

1.2 蚁群算法的数学模型

根据旅行商问题(Traveling Salesman Problem，TSP)来阐述蚁群算法的数学模型，TSP是假设有一组城市，经过每个城市一次，求最短路径和，各因子含义如下：d_{ij}是i、j城市间的距离大小；n是城市个数，m是蚂蚁数量；$\tau_{ij}(t)$是t时刻在(i,j)路径上的信息素含量；$b_i(t)$表示t时刻位于城市i的蚂蚁数，于是$m=\sum_{i=1}^{n}b_i(t)$；tabu_k是禁忌表，作用是存放蚂蚁目前已经经过的城市。蚂蚁k在t时刻从城市i迁移到城市j的状态转移概率$p_{ij}^{k}(t)$为：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{s\subset \text{allowed}_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}} & j\in \text{allowed}_k\\ 0 & \text{其他}\end{cases} \tag{1}$$

其中，C代表所有城市组成的城市集合，$\text{allowed}_k=\{C-\text{tabu}_k\}$代表蚂蚁$k$能够迁移到的城市范围；$\eta_{ij}(t)$代表启发因子，表示路径长短对蚂蚁的吸引力，并且$\eta_{ij}(t)=1/d_{ij}$，$d_{ij}$越小，则$\eta_{ij}(t)$越大，于是$p_{ij}^{k}(t)$就越大；$\alpha$代表信息素启发性因子，该因子的含义表示路径上累积的信息素的权重大小；β代表期望启发式因子，该因子表示$\eta_{ij}(t)$的权重大小[7]。随着算法的不断进行，路径上积累的信息素会越来越多，会导致启发信息被残留信息淹没，因此路径上的信息素需要持续不断地挥发，所以$t+n$时刻在路径(i,j)上的信息素更新公式如下：

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{2}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{3}$$

其中，ρ表示信息素挥发系数，$\Delta\tau_{ij}(t)$表示信息素的增量大小，$\Delta\tau_{ij}^{k}(t)$是蚂蚁k释放在(i,j)路径上的信息素含量。蚁群算法有3种基本的计算模型，分别是Ant-Cycle、Ant-Quantity、Ant-Density，这3种模型的区别仅仅是采用了不同的信息素更新方式，即$\Delta\tau_{ij}^{k}(t)$的计算方式不同。这3种计算方式分别为：

(1)Ant-Cycle模型

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{第}k\text{只蚂蚁在本次循环中经过路径}(i,j)\\ 0 & \text{其他}\end{cases} \tag{4}$$

(2)Ant-Quantity模型

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{d_{ij}} & \text{第}k\text{只蚂蚁在}t\text{和}t+1\text{之间经过路径}(i,j)\\ 0 & \text{其他}\end{cases} \tag{5}$$

(3)Ant-Density模型

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}Q & \text{第}k\text{只蚂蚁在}t\text{和}t+1\text{之间经过路径}(i,j)\\ 0 & \text{其他}\end{cases} \tag{6}$$

其中，Q是信息素浓度大小，L_k是蚂蚁经过的路径总长。Ant-Cycle模型表示蚂蚁经过全部城市后才更新信息素，是一种信息素全局更新方式；Ant-Quantity模型以及Ant-Density模型表示蚂蚁每经过一个城市就更新信息素，是一种信息素局部更新方式。通常来讲，Ant-Cycle模型求解TSP效果最佳，所以通常都是采用Ant-Cycle模型来解决该类问题[8]。

1.3 蚁群算法流程图

蚁群算法流程图如图1所示。

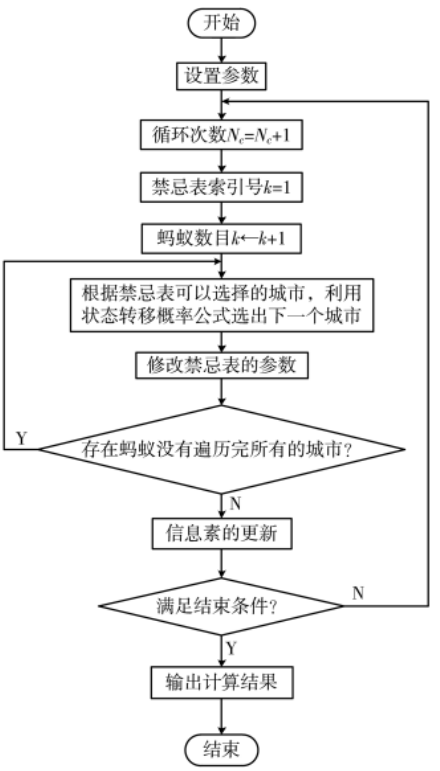

图1 蚁群算法流程图

2 蚁群算法的改进

蚁群算法比较容易陷入到局部最优解，在算法搜索的初始阶段，每条路径上的信息素都比较稀缺，使得算法在刚开始不久进行搜索时速度太慢，严重影响了算法的性能，所以有必要对算法进行一些改进[9]。本文的改进包括两个方面：(1)对蚁群算法自身进行改进，具体包括：引入伪随机比例规则，进行全局信息素强化，引入了交叉变异操作；(2)将遗传算法和蚁群算法进行融合，形

成遗传蚁群算法。

2.1 引入伪随机比例规则

蚁群算法存在一个很严重的问题，就是容易陷入局部最优解。为了解决这个问题，必须扩大蚂蚁的选择范围，使得蚂蚁不仅能选择当前最优路径，还能够以一定的概率选择其余路径。因此引入伪随机比例规则：

$$\begin{cases} \arg\max_{s\in \text{allowed}_k}\{[\tau_{is}(t)]^{\alpha}[\eta_{is}(t)]^{\beta}\} & q\leqslant q_0 \\ \text{根据轮盘赌算法选择} & q>q_0 \end{cases} \tag{7}$$

式中，q_0 代表一个遵循均匀分布的变量。按照信息素含量以及启发信息，可以计算出蚂蚁最优转移方式的可能性大小 q_0，与此同时，蚂蚁能够以 $1-q_0$ 的概率搜寻其他路径。为了使得蚂蚁能够选择究竟是搜寻最优路径，还是搜寻其他地方的路径，能够通过设置 q_0 的值来控制算法对下一条路径的选择状态，把 q 定义为 $1-e^{-\frac{1}{K}}$，其中 $K=1,2,\cdots,NC$，假如 $q\leqslant q_0$，就根据式(7)计算；否则采用轮盘赌方式，轮盘赌方式如下：

(1)设置参数 $r=\text{rand}(0,1)$；

(2)若 $s\geqslant r$，转步骤(4)；

(3)$i=i+1$，$s=s+p_{ij}^{k}(t)$，按式(1)计算 $p_{ij}^{k}(t)$ 的值，接着转步骤(2)；

(4)算法结束[10]。

2.2 全局信息素强化

蚁群算法中蚂蚁选择路径是互不影响、相对独立的，存在一定的盲目性，这导致算法收敛速度慢，收敛准确性也低；另外当问题的规模比较大时，蚁群算法会求出很多无效路径，存储太多无效路径会使得算法效率降低。因此，本文给出了一种优化方法：用一个变量来标记每次迭代过程完成后的一条最优路径，然后对最优路径做一次信息素的加强操作，加强后，最优路径上的信息素比原来的信息素要强，这样会有效地避免求出太多无效路径，大大地加快了算法的寻优速度。因此全局信息素更新式(2)可以改成式(8)。

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t)+e\Delta\tau_{ij}^{\text{bs}}(t) \tag{8}$$

$$\Delta\tau_{ij}^{\text{bs}}(t)=\begin{cases} \dfrac{Q}{L_{\text{bs}}} & \text{蚂蚁 bs 经过}(i,j)\text{路径} \\ 0 & \text{否则} \end{cases} \tag{9}$$

全局信息素更新公式比原来多加了一个 $e\Delta\tau_{ij}^{\text{bs}}(t)$。$e$ 是一个变量，它越大表示信息素加得越多，e 的值要按照实际情况来确定，e 的取值不宜过大，过大会使得算法搜寻的随机性下降，造成算法停滞，陷入局部最优；也不能取得太小，太小了会使得加强的效果不好，本文把 e 取为 0.5。bs 用来标记每次迭代完成后的最优路径，式(9)确保了如果最优路径经过 (i,j) 路径，则 (i,j) 路径上的信息素做一次增强，这样在提高了搜寻速度的同时又增强了搜寻的准确性。

2.3 交叉变异操作

为了克服蚂蚁容易陷入局部最优解的缺点，本文给出了一种优化方法：该方法启发于遗传算法的变异操作，随机地在当次迭代过程中获得的解集中选择一条路径并且设置一个变异率 p_λ，控制 p_λ 为一个较小的值，当满足变异率时对该路径的信息素值作一个随机的修改，这些值要在 $[\tau_{\min},\tau_{\max}]$ 之间，$\tau_{\min}$ 代表的是信息素最小值约束，$\tau_{\max}$ 代表的是信息素最大值约束，通过这种约束条件可以使得信息素含量限制在一定的区间内，有效地避免了算法因为各条路径上的信息素存在较大的差距而使得算法的搜索寻优过程出现停滞，造成陷入局部最优解的情况的发生。

从遗传算法的交叉操作中受到启发，可以设置另外一个交叉率 p_θ，把 p_θ 限制为一个较小的值，随机地在该次迭代所有的解集中选择两条较优解，然后再把这两条较优解进行交叉，交叉后会得到一条新的解，然后再把这条新的解加入到解集中，新的解通常更加接近最优解，交叉操作增加了算法的随机性，避免了算法陷入到局部最优解[11]。

2.4 遗传算法结合蚁群算法

由于遗传算法在初始阶段的时候搜索能力特别强，与之相反，蚁群算法在初始阶段搜索能力特别弱，如果单独用蚁群算法进行资源分配，效率会非常低下。因此，本文将两种算法融合起来，开始用遗传算法进行搜索，然后把遗传算法搜索到的较优解变换为蚁群算法的起始信息素，后期再转入蚁群算法进行搜索。于是便出现了两个问题：第一是两种算法的融合点怎么确定，第二是怎么把遗传算法的较优解变换为蚁群算法的起始信息素。下面先简单地介绍一下遗传算法，然后再具体阐述[12]。

2.4.1 遗传算法原理

遗传算法模拟了自然界中生物的进化过程，首先通过随机的方式产生一定数量的个体，由这些个体构成起始种群，利用适应度函数求出所有染色体的适应度值，对于某个个体而言，它的适应度值越高，表示它的适应力越强，所以从中选择适应度值高的种群个体进行交叉变异，交叉变异后会得到适应力更强的个体。该过程不断迭代，满足结束条件时终止算法，遗传算法通过不断地迭代来优化问题的解[13]。

2.4.2 遗传算法流程图

遗传算法的流程图如图 2 所示。

2.4.3 融合点的确定

为了将两种算法进行融合，就必须要确定两者的融合点，于是绘制了两种算法的速度时间曲线图，如图 3 所示。由图 3 可知：0~t_a 时间段，遗传算法搜寻速度明显快于蚁群算法；在 t_a 时间段以后，情况相反，蚁群算法搜寻速度明显快于遗传算法，t_a 时间点两种算法的搜寻速

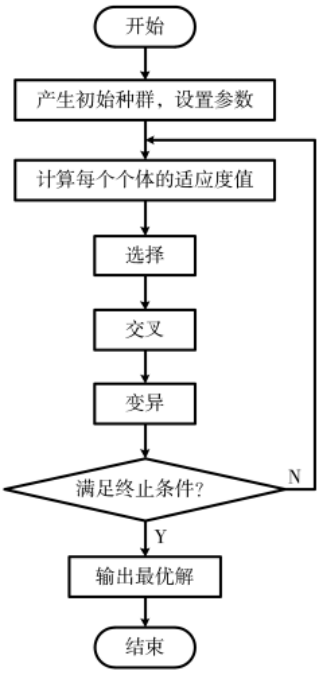

图 2　遗传算法流程图

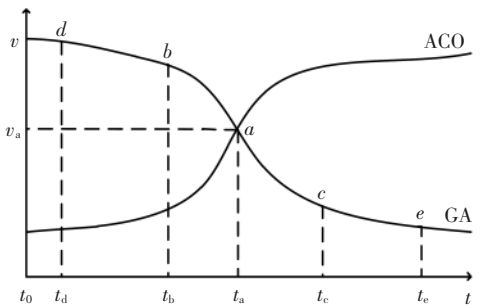

图 3　速度-时间曲线图

度相同，因此 t_a 点就是融合点。确定融合点的具体思路是：设置一个最小进化率 p_w，倘若连续三代的进化率都比 p_w 要小，那么该点就为融合点，于是在该点结束遗传算法，进入蚁群算法搜索阶段[14]。

2.4.4 初始信息素转化方法

本文将遗传算法的解变换为蚁群算法的起始信息素，具体实现过程为：在遗传算法结束后以 15% 的概率选择适应度值最高的个体，每个个体都代表了一个较优解；然后再将这些解变换成节点上的蚂蚁数；接着再使用信息素转化因子，将这些蚂蚁数变换成节点上的信息素。于是 (i,j) 路径上的起始信息素定义为：

$$\tau_{ij}^{s}=\tau_0+\tau_{ij}^{GA} \tag{10}$$

$$\tau_{ij}^{GA}=\theta(x_i+x_j) \tag{11}$$

其中，τ_{ij}^{GA} 是通过遗传算法的较优解转化而来的信息素值，θ 是信息素转化因子，τ_0 是信息素常数值，x_i 和 x_j 分别是资源节点 i 和资源节点 j 上的蚂蚁数[15]。

2.4.5 融合后的算法流程图

融合后的算法流程图如图 4 所示。

3 实验与分析

为了更加直观地验证对蚁群算法进行改进后的效果，本文分别对改进蚁群算法、基本遗传算法和基本蚁群算法进行了 MATLAB 仿真实验，并把它们的实验结果进行对比分析，从任务完成时间、算法迭代次数、负载均衡效果来评价算法的改进效果，同时设置了资源点数为 8 个，任务数量从 20~100 个，算法的各个参数是通过以往的文献来确定的，各参数取值见表 1。

通过实验得到实验数据并进行实验结果的对比分析，改进算法与基本蚁群算法任务完成时间对比如图 5 所示，改进算法与基本遗传算法的任务完成时间对比如图 6 所示。迭代次数对比如图 7 所示，负载均衡对比如图 8 所示。

由图 5 和图 6 可知：改进算法的任务完成时间明显

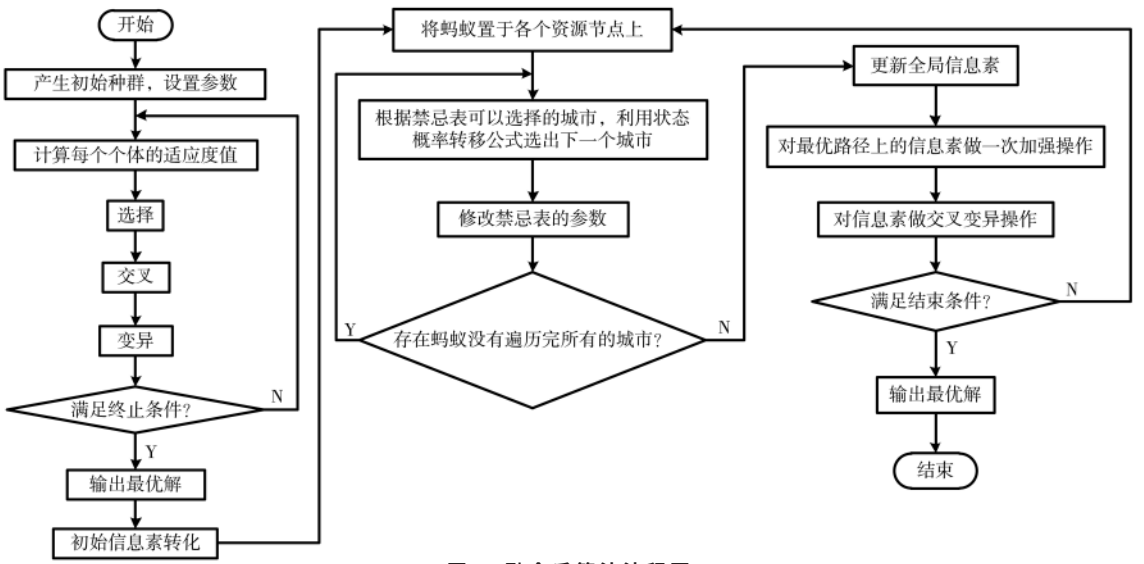

图 4　融合后算法流程图

表 1 各参数取值

参数	数值	参数	数值
种群规模/个	30	挥发系数 ρ	0.4
变异概率	0.05	蚁群迭代次数	500
交叉概率	0.6	信息素转化因子 θ	0.5
遗传迭代次数	500	α	0.3
最小进化率 p_w	3%	β	0.5

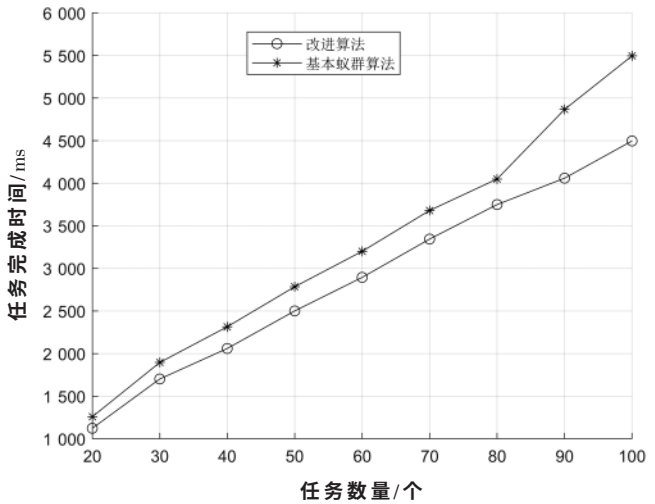

图 5 改进算法与基本蚁群算法任务完成时间对比

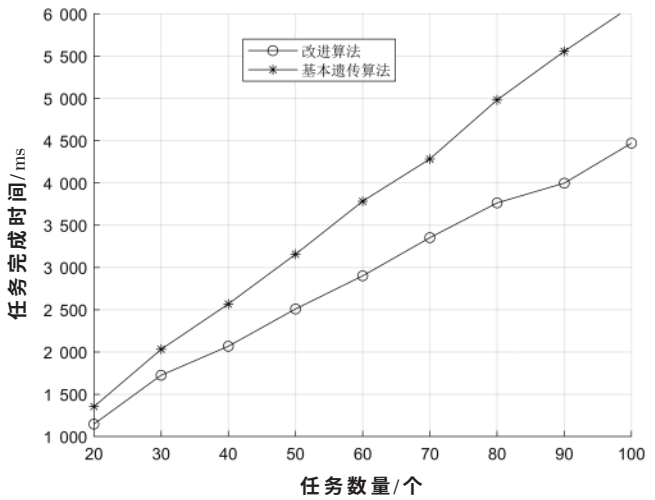

图 6 改进算法与基本遗传算法任务完成时间对比

短于两种未改进的算法，且在任务数量较大时改进效果越明显，因此改进算法更加适合于大规模资源分配问题。由图 7 可知：迭代次数随着任务数量的增加而增加，改进算法的迭代次数比两种未改进算法的迭代次数更少，说明对算法进行改进提升了算法的效率。由图 8 可知：改进算法的负载均衡效果要好于其他两种未改进的算法，说明了对算法进行改进提升了算法负载均衡性能。

4 结论

本文针对蚁群算法进行了一系列改进，并进行了 MATLAB 仿真实验。实验结果表明，改进算法的任务完成

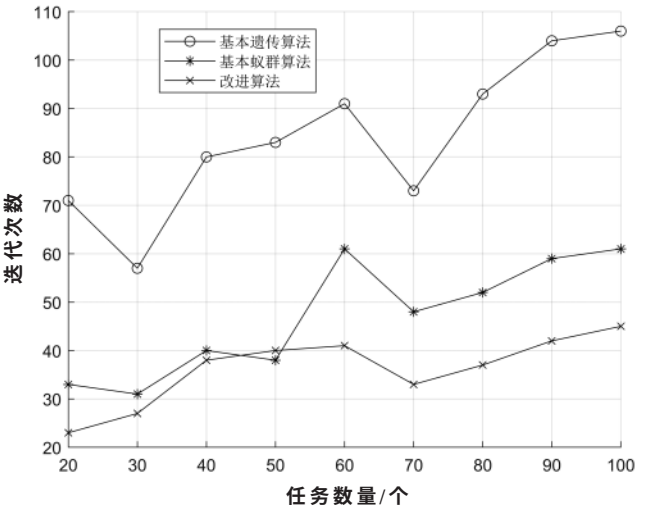

图 7 迭代次数对比

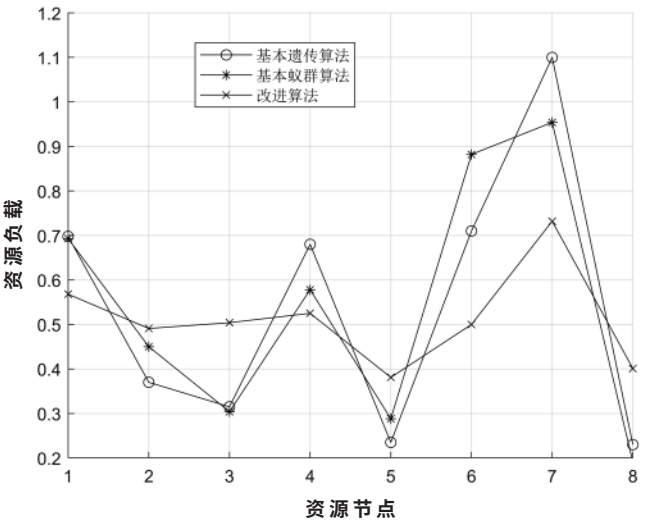

图 8 负载均衡对比

时间更短，且在任务数量较大时效果越明显，说明改进算法更加适合大规模资源分配问题。改进算法的迭代次数更少，说明对算法进行改进提升了算法的效率。此外，改进算法的负载均衡效果也更好。由此可以得出结论：对算法的改进是有效的，能明显提升算法的综合性能。

参考文献

[1] 罗孝蓉.基于改进蚁群算法的云计算任务调度研究[D].北京：华北电力大学，2019.

[2] Zhu Linan，Li Qingshui，He Lingna.Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm[J].IJCSI International Journal of Computer Science Issues，2012，9(5)：54-58.

[3] GAO S，JIANG X Z，TANG K.Hybrid algorithm combining ant colony optimization algorithm with pariticle swarm optimization[C]//Proceedings of the 25th Chinese Control Conference，2006：1428-1432.

[4] Deng Wu，Xu Junjie，Zhao Huimin.An improved ant colony

optimization algorithm based on hybrid strategies for scheduling problem[J].IEEE Access,2019,7:20281-20292.
[5] Yang Hai.Improved ant colony algorithm based on PSO and its application on cloud computing resource scheduling[J].Advanced Materials Research Vols,2014,989-994:2192-2195.
[6] JAIN R.EACO:an enhanced ant colony optimization algorithm for task scheduling in cloud computing[J].Int. J. Secur. Appl.,2020,13(4):91-100.
[7] 王艳平.基于蚁群算法的云计算资源调度研究[D].曲阜:曲阜师范大学,2015.
[8] 王科.基于改进蚁群算法的云计算任务调度策略研究[D].杭州:杭州电子科技大学,2012.
[9] Lu Xin,Gu Zilong.A load-adapative cloud resource scheduling model based on ant colony algorithm[C]//Proceedings of IEEE CCIS2011,2011:296-300.
[10] 罗栋梁.基于改进蚁群算法的云计算任务调度研究[D].哈尔滨:哈尔滨工程大学,2016.
[11] 李柯.云计算下基于蚁群优化算法的资源分配研究[D].杭州:杭州电子科技大学,2016.
[12] 苗培.蚁群优化算法在云计算资源分配上的应用[D].济南:山东师范大学,2015.
[13] 查安民.优化粒子群和蚁群算法的云计算任务调度研究[D].南京:南京航空航天大学,2016.
[14] 秦晓娜.基于改进蚁群算法的云计算资源分配研究[D].广州:华南理工大学,2014.
[15] 张晓东.遗传算法和蚁群算法相融合的云计算任务调度算法研究[D].镇江:江苏大学,2013.

(收稿日期:2021-05-08)

作者简介:
刘灯明(1992-),男,硕士,主要研究方向:计算机软件及信息系统集成。
荆俊峰(1967-),男,硕士,高级工程师,主要研究方向:计算机软件及信息系统集成。
刘凯(1989-),男,硕士,工程师,主要研究方向:云计算信息系统应用。

扫码下载电子文档

“边缘计算在工业互联网的应用”专栏征稿

工业互联网是国家新基建战略中重要的一环,我国工业互联网已步入发展的关键时期。边缘计算技术作为新型的计算形式,在工业互联网的发展中起到了重要作用。为了更好地促进边缘技术推动未来工业互联网技术发展,汇聚行业技术创新力量,展示国内外边缘计算技术相关研究成果以及最新进展,洞悉边缘计算及工业互联网产业的未来趋势,《电子技术应用》》杂志拟于2022年第9期推出“边缘计算在工业互联网的应用”主题专栏。欢迎相关领域研究者大力关注,踊跃投稿!

1.专栏特约主编

林荣恒 北京邮电大学 副教授

2.征稿范围(包括但不限于以下方面)

(1)边缘计算与工业互联网结合的体系架构与技术发展研究
· 典型工业互联网与边缘计算平台架构研究
· 边缘计算硬件以及芯片的生态研究
(2)工业互联网中网络与通信技术发展趋势研究
· 云-边协同算法与智能调度
· 工业互联网相关网络通信标准化研究
· 支持边缘计算的工业互联网新型协议研究
(3)边缘计算结合数字孪生技术发展及应用
· 工业互联网中数字孪生技术的应用
· 数字孪生对边缘计算的影响及技术改进
(4)边缘智能与工业互联网发展及应用
· 边缘智能在工业互联网中的发展趋势
· 边缘计算与人工智能的协同等研究进展
· 基于边缘智能的工业互联网应用
(5)工业互联网的典型行业应用
· 智能制造领域边缘计算的应用
· 面向其他工业领域边缘计算的应用

3.稿件要求:文章需具有创新性且未在其他期刊公开发表过。文中图表需清晰,文字规范。详见《电子技术应用》投稿须知(http://www.chinaaet.com/paper/notice/)。

4.截稿日期:2022年7月20日。

5.投稿方式:请登录《电子技术应用》官网(http://www.ChinaAET.com/),投稿页面中选择“边缘计算在工业互联网的应用”专栏投稿,按要求提交。

版权声明