文章编号:1000-5013(2015)06-0650-05 doi:10.11830/ISSN.1000-5013.2015.06.0650

高速收敛混沌粒子群算法的云计算任务调度

王秉

(河南交通职业技术学院 航运海事系,河南 郑州 450000)

摘要: 针对传统粒子群算法在处理云计算任务调度问题时,存在求解精度不高、容易陷入早熟收敛等缺陷,提出一种改进的高速收敛混沌粒子群算法.首先,采用混沌序列对初始化过程进行优化;其次,利用适应度方差对早熟现象进行有效诊断,并对算法在负梯度方向进行修正,使其跳出局部最优,实现高速收敛.仿真实验表明:改进后的粒子群算法能有效地避免早熟,收敛速度及求解精度都明显提高,非常适合云计算任务调度.

关键词: 云计算;任务调度;粒子群算法;混沌

中图分类号: TP 393 **文献标志码:** A

云计算资源服务模式有效地实现了资源配置和按需访问.由于云计算具有分布式计算的特性,每天都要处理海量信息,且处理过程要有很高的实时性.因此,在云计算环境下,进行高效合理的任务调度,实现系统全局最优,成为时下研究的热点问题之一[1-2].与网格计算类似,云环境下的任务调度也属于一个NP难解问题[3].用于网格计算的传统任务调度算法,如Min-Min/Max[4],Sufferage[5]等也被用于解决云环境下的任务调度,但效果不佳.而一些具有启发式性质的智能算法,在处理该问题时显示出较好的效果,如遗传算法[6]、蚁群算法[7]、粒子群算法等[8].刘万军等[9]研究云计算服务集群资源调度和负载平衡优化问题,在粒子群算法中,引入了动态多群体协作和变异粒子逆向飞行思想;苏淑霞[10]对传统粒子群算法进行改进,采用间接方式进行初始化编码,扩展了求解的空间,但是解的精度不高;封良良等[11]采取间接编码形式,同时对适应度函数进行了优化;王波等[12]将粒子群算法与遗传算法结合,引入变异、交叉等机制,提高了算法的全局搜索能力,但求解过程较复杂.尽管取得了一定的成果,但在解决云环境下的任务调度问题时,必须考虑信息量的庞大.因此,粒子群算法应该具有足够的解空间,高效的求解速度及有效克服早熟收敛的能力等,而这些问题在现有文献中的研究还不够深入.基于此,本文研究将粒子群算法用于解决云环境下的任务调度问题.

1 问题的描述

目前,应用最广泛的是谷歌公司提出的Map/Reduce编程模型.该模型将整个计算过程分为2个阶段:映射(Map)和化简(Reduce).一个大任务可以被分解成多个相互独立的子任务,在不同资源节点上,通过并行的方式完成,汇总后给出执行结果.因此,云环境下任务调度问题是将子任务合理的与资源进行匹配,从而使完成任务的总代价最小.

云计算环境下,假设总任务为N,将其拆分为n个相互独立的子任务,而这些子任务需要在m个虚拟资源节点上执行($m<n$).任务集可以表示为$T=\{t_i \mid i=1,2,\cdots,n\}$,资源节点集可以表示为$\mathrm{VM}=\{vm_j \mid j=1,2,\cdots,m\}$,其中,规定每个子任务只能在一个虚拟资源上执行.利用矩阵形式,T与VM之间的关系表示为

收稿日期: 2015-10-08

通信作者: 王秉(1965-),男,副教授,主要从事计算机图形图像的研究.E-mail:wbjtxy@163.com.

基金项目: 国家自然科学基金资助项目(201411326136);河南省科技厅项目(2013132300410337)

$$\boldsymbol{X}=\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}. \tag{1}$$

式(1)中：$x_{i,j}\in\{0,1\}$为子任务 t_i 与虚拟资源 vm_j 之间的分配关系，且满足条件 $\sum_{i=1}^{m}x_{i,j}=1$，$x_{i,j}=1$ 表示 t_i 分配给 vm_j 执行. $\mathrm{ETC}_{i,j}$表示 t_i 在 vm_j 上的期望执行时间，其矩阵形式为

$$\mathbf{ETC}=\begin{bmatrix} \mathrm{ETC}_{1,1} & \mathrm{ETC}_{1,2} & \cdots & \mathrm{ETC}_{1,n} \\ \mathrm{ETC}_{2,1} & \mathrm{ETC}_{2,2} & \cdots & \mathrm{ETC}_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{ETCC}_{m,1} & \mathrm{ETC}_{m,2} & \cdots & \mathrm{ETC}_{m,n} \end{bmatrix}. \tag{2}$$

那么，任务完成的总时间为

$$\mathrm{SFT}=\max_{r=1}^{m}\sum_{i=1}^{n}\mathrm{RT}(r,i). \tag{3}$$

式(3)中：$\mathrm{RT}(r,i)$为第 i 个子任务在第 r 个资源上的完成时间. 任务调度的优化目标即为使 SFT 最小.

2　算法的设计

2.1　标准粒子群算法

首先，在解空间初始化生成一群粒子，每个粒子都代表优化问题的一个潜在最优解，并通过位置、速度和适应度值表征每个粒子的特性. 粒子在解空间中运动，根据个体极值 P_{best} 和群体极值 G_{best} 的变化对位置和适应度值进行更新. 同时，通过对比新粒子的适应度值和 P_{best}，G_{best} 的适应度值，再次更新 P_{best} 和 G_{best}. 速度和位置的更新公式分别为

$$v_{i,d}^{k+1}=\omega v_{i,d}^{k}+c_1r_1(p_{i,d}^{k}-x_{i,d}^{k})+c_2r_2(p_{g,d}^{k}-x_{i,d}^{k}), \tag{4}$$

$$z_{i,d}^{k+1}=z_{i,d}^{k}+v_{i,d}^{k+1}. \tag{5}$$

式(4)～(5)中：ω 为惯性权重；$d=1,2,\cdots,D$；k 为当前迭代次数；$v_{i,d}$为粒子的速度；参数 c_1 和 c_2 为 0 或正常数，代表速度的加权学习因子；r_1 和 r_2 是随机数，取值范围为[0,1].

为了避免粒子在解空间中盲目进行目标搜寻，位置一般取值范围为$[-z_{\max},z_{\max}]$，速度一般取值范围为$[-v_{\max},v_{\max}]$.

2.2　混沌初始化

标准粒子群及许多改进算法在初始化过程中，基本都采用随机方式. 这种方式虽然实现简单，但是初始化粒子的均衡度较差，多样性不高. 考虑到混沌序列具有良好的随机性和遍历性，可以在一定程度上按照自身规律不重复遍历全部状态. 因此，文中研究在粒子群算法的初始化过程中，采用混沌序列，从而有效提高粒子的分布均衡度和多样性.

在混沌理论中，包括多种混沌映射，使用的为立方映射数学模型为

$$y(n+1)=4y(n)^3-3y(n),\qquad -1\leqslant y(n)\leqslant 1,\quad n=0,1,2,\cdots. \tag{6}$$

由式(6)可知：立方映射生成序列的取值范围为(−1,1)，在使用时，还需将其映射到粒子的搜索空间，映射公式为

$$x_{i,d}=\min_d+(1+y_i(d))\frac{(\max_d-\min_d)}{2},\qquad i=1,2,\cdots,N,\quad d=1,2,\cdots,D. \tag{7}$$

式(7)中：$y_i(d)$为第 i 个粒子的第 d 维；$x_{i,d}$为其坐标；$\max_d$ 和 $\min_d$ 为其上下限.

此外，为了进一步提高立方映射的分布多样性，在生成粒子的过程中嵌入位置信息，即要满足

$$\cos(\boldsymbol{z}_i,\boldsymbol{z}_j)=\frac{\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j}{\|\boldsymbol{z}_i\|_2\times\|\boldsymbol{z}_j\|_2}\leqslant\varepsilon,\qquad i,j=1,2,\cdots,N,\quad i\neq j. \tag{8}$$

式(8)中：$\boldsymbol{z}_i$ 为某一个利用立方映射生成的粒子向量；ε 为事先选定的阈值；$\cos(\boldsymbol{z}_i,\boldsymbol{z}_j)$为粒子间的位置信息，$\cos(\boldsymbol{z}_i,\boldsymbol{z}_j)$越大表示粒子越靠近.

2.3 早熟诊断和高速收敛策略

尽管混沌序列初始化加强了粒子的多样性,但还是无法避免粒子群算法出现早熟现象.早熟是算法错误地将一个局部次优解当作真正的全局最优解,而使算法停止搜索.因此,早熟的有效诊断十分重要.在发现早熟现象以后,如何使粒子跳出当前搜索空间,重新进行有效搜索值得研究.

借鉴吕振肃等[13]提出的适应度方差策略诊断算法的早熟现象,定义种群规模为 N,f_i 为第 i 个粒子的当前适应度值,f_{ave} 为平均适应度值,那么,可以得到一个归一化定标因子,即

$$f=\begin{cases}\max(|f_i-f_{\text{ave}}|), & \max(|f_i-f_{\text{ave}}|)>1,\\ 1, & \text{其他}.\end{cases} \tag{9}$$

在此基础上,适应度方差的计算公式为

$$\sigma^2=\sum_{i=1}^{N}\left(\frac{f_i-f_{\text{ave}}}{f}\right)^2. \tag{10}$$

从适应度方差的定义可知:粒子在收敛过程中,σ^2 会逐渐变小直至为0,达到完全收敛的状态,表征的是种群中粒子的收敛程度.因此,可以根据经验事先设定一个阈值,当 σ^2 小于这个阈值时,即判断为算法早熟.尽管如此,当算法搜索到真正的全局最优解时,也会出现该现象,为了避免冲突,算法在运行过程中,还要验证当前的最优适应度值 f_{best} 是否大于理论最优适应度值 f_d,若条件成立,则为早熟.

当判断算法出现早熟现象以后,需给算法添加一个扰动,使粒子跳出当前的搜索空间.采用负梯度方向调整策略,使粒子快速进入新的搜索空间,并且整个算法以较高速率收敛.负梯度调整公式为

$$-\nabla F(f_{\text{best}})=\frac{F(f_{\text{best}}^{\text{old}})-F(f_{\text{best}})}{f_{\text{best}}^{\text{old}}-f_{\text{best}}}. \tag{11}$$

式(11)中:f_{best} 当前种群的最优适应度值;$f_{\text{best}}^{\text{old}}$ 为 f_{best} 最邻近的一次未更新之前的值.

2.4 算法的实现步骤

步骤1 初始化种群规模 N,粒子搜索空间维度 D,惯性权重 ω_0,加速因子 c_1 和 c_2,迭代次数 $t=0$,最大迭代次数为 T.

步骤2 随机生成一组粒子位置的初始向量 $\boldsymbol{z}_1=(z_{1,1},z_{1,2},\cdots,z_{1,D})$,取值范围(-1,1),剩余 $N-1$ 组粒子位置的初始向量按照式(6)产生.要注意满足式(8)条件,并按照式(7)进行映射.

步骤3 定义适应度函数 $F(i)=\dfrac{1}{\text{SFT}_i}$,$1\leqslant i\leqslant N$,并计算适应度值.

步骤4 更新粒子速度、位置和惯性权重.

步骤5 按照式(10)计算适应度方差 σ^2,如果 $\sigma^2<\gamma$ 且 $f_{\text{best}}>f_d$,按照式(11)进行调整.

步骤6 $t=t+1$,如果 $t<T$,转至步骤4;否则,算法结束.

3 仿真实验

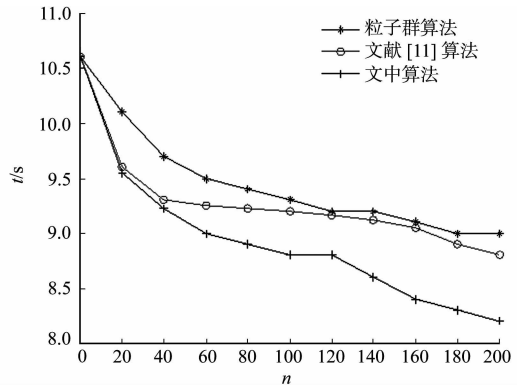
为了验证文中所提算法的性能,在Cloudsim云平台实现算法的仿真分析,并与标准粒子群算法和文献[11]提出的改进粒子群算法进行比较.

设置算法的参数为:种群规模100,惯性因子 $\omega=0.9$,加速因子 $c_1=c_2=2$,最大迭代次数200.完成一组固定任务和资源数量的调度对比试验,共包括2种情况:1) 50个任务,5个资源;2) 500个任务,5个资源.2种情况下的任务性质和资源能力完全相同,所有任务的长度范围为[400 MI,1 000 MI],5个资源的处理能力分别为{400 MIPs,600 MIPs,800 MIPs,1 000 MIPs,1 200 MIPs}.每种算法在相同条件下各运行10次,取平均值作为最终的结果,如图1所示.图1中:n 为迭代次数,t 为总任务完成时间.

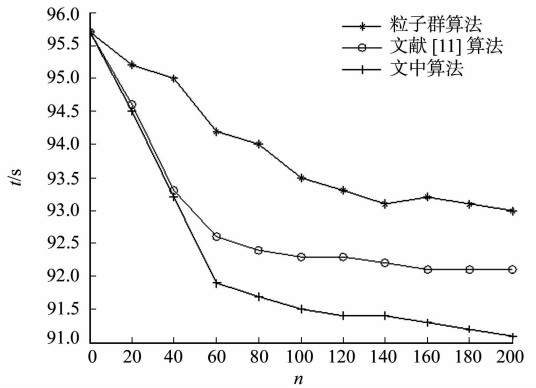
由图1可知:由于文中算法与文献[11]算法在标准粒子群算法的初始化阶段都进行了优化,因此,算法迭代前期,任务调度的时间基本相同,但都优于标准粒子群算法;随着算法迭代的深入,标准粒子群算法和文献[11]算法没有早熟的诊断和跳出机制.因此,找到的都不是最优解,任务调度的时间也要明显多于文中算法,尤其是在情况1时,文献[11]算法的后期结果与标准粒子群算法接近,可见早熟对于粒子群算法影响较大,而文中算法却始终保持一个较好的进化过程,性能明显更好.

考察在资源数量一致且任务量不同的情况下,3种算法的性能.设置资源个数为5,任务量从0依次

递增到 500(每隔 50 递增).算法参数与上述相同,结果如图 2 所示.图 2 中:t 表示任务完成时间;N_r 表示任务数.



(a) 情况 1



(b) 情况 2

图 1　2 种情况下的调度结果

Fig. 1　Scheduling results for two cases

由图 2 可知:当资源数量固定时,随着任务数量的增加,调度时间也逐渐增加;但是标准粒子群算法与文献[11]算法的调度用时明显高于文中算法,尤其是任务数量达到 150 以后,差距更为显著;随着任务数量的进一步递增,文中算法的优势更加明显.

比较任务量固定、资源数量不同时,3 种算法的性能,主要考察资源负载均衡度,结果如图 3 所示.图 3 中:η 为资源负载均衡度;N_d 为资源节点数量.根据以往的研究[14],资源负载均衡度越接近 1,说明资源的利用率越高,算法的调度越理想.

由图 3 可知:文中算法具有更好的系统负载均衡度,平均值为 0.847,比较接近 1,说明该算法对资源的利用率较高.而文献[11]算法和标准粒子群算法的资源负载均衡度分别只有 0.6 和 0.5 左右,相对较低,反映出资源负载具有较高的不平衡度.

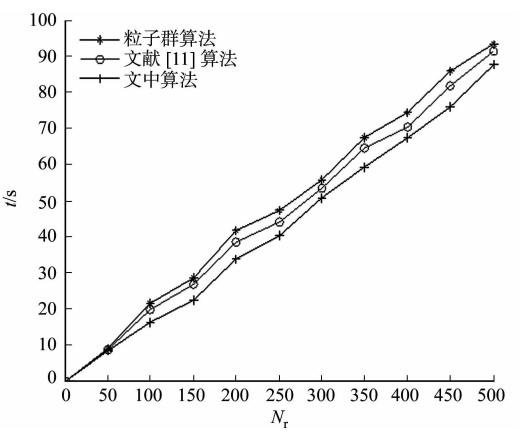


图 2　不同任务量情况下的调度结果

Fig. 2　Scheduling results with various tasks

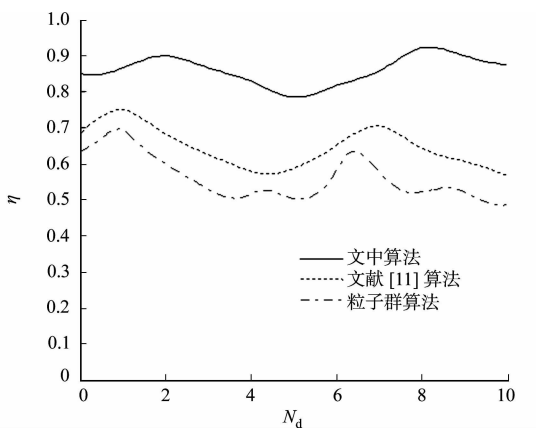


图 3　3 种算法的资源负载均衡度比较

Fig. 3　Comparison of the resource load balancing degree for 3 kinds algorithms

综合上述实验结果,提出的高速收敛混沌粒子群算法的性能更优,更适合解决云计算环境下的资源调度问题.

4　结束语

对云计算环境下的任务调度问题进行深入研究,提出一种基于高速收敛混沌粒子群的任务调度算法.简要分析了云环境下的计算模型,并针对标准粒子群算法的缺陷提出了改进措施.采用混沌映射优化初始化过程,同时给出了算法早熟的诊断方法和高速收敛策略.仿真实验的结果表明:所提算法具有较好的性能,收敛速度快,求解的精度更高,非常适合解决云计算环境下的资源调度问题.在下一步的工

作中,还将研究同时考虑计算成本、节能等因素的云计算调度问题.

参考文献:

[1] 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.
[2] 王燕琼,李国刚. 下行多小区 MIMO 系统协作多点传输联合调度机制[J]. 华侨大学学报(自然科学版),2012,33(3):260-264.
[3] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.
[4] 王观玉. 网格计算中任务调度算法的研究和改进[J]. 计算机工程与科学,2011,33(10):186-190.
[5] 胡志刚,陈俊. 网格工作流中一种扩展的 QD-Sufferage 调度算法[J]. 计算机应用研究,2008,25(5):1504-1506.
[6] 朱宗斌,杜中军. 基于改进 GA 的云计算任务调度算法[J]. 计算机工程与应用,2013,49(5):77-80.
[7] 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, 6(13): 6833-6840.
[8] 郭力争,耿永军,姜长源,等. 云计算环境下基于粒子群算法的多目标优化[J]. 计算机工程与设计,2013,34(7):2358-2362.
[9] 刘万军,张孟华,郭文越. 基于 MPSO 算法的云计算资源调度策略[J]. 计算机工程,2011,37(11):43-44.
[10] 苏淑霞. 粒子群算法在云计算任务调度中的应用[J]. 南京师大学报(自然科学版),2014,37(4):145-149.
[11] 封良良,张陶,贾振红,等. 云计算环境下基于粒子群的任务调度算法研究[J]. 计算机工程,2013,39(5):183-186.
[12] 王波,张晓磊. 基于粒子群遗传算法的云计算任务调度研究[J]. 计算机工程与应用,2015,51(6):84-88.
[13] 吕振肃,侯志荣. 自适应变异的粒子群优化算法[J]. 电子学报,2004,32(3):416-420.
[14] lSARD M, PRABHAKARAN V, CURRCY J, et al. Quincy: Fair scheduling for distributed computing clusters[C]// Proceedings of on Operating Systems Principles. New York: ACM, 2009: 261-276.

Cloud Computing Task Scheduling of High-Speed Convergence of Chaotic Particle Swarm Optimization

WANG Bing

(Department of Maritime, Henan Vocational and Technical College of Communications, Zhengzhou 450005, China)

Abstract: In this paper, we proposed an advanced high speed of convergence chaotic particle swarm algorithm to adjust the common problems of traditional particle swarm algorithm such as low accuracy and easily trapped in premature convergence during the cloud computing task scheduling. Firstly, the initial process was optimzed by chaotic sequence. Then, the effective diagnosis of premature phenomenon was determined by fitness variance. The algorithm correction was performed by negative gradient direction, which could jump out the local optimum and achieve high speed of convergence. Simulation experiments show that the improved particle swarm algorithm can effectively avoid premature, enhance convergence speed and solution accuracy, which is suitable for cloud computing task scheduling.

Keywords: cloud computing; task scheduling; particle swarm optimization algorithm; chaotic

(**责任编辑**: 黄晓楠 **英文审校**: 吴逢铁)