计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(7):151−156 [doi: 10.15888/j.cnki.csa.007005]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

云计算调度粒子群改进算法①

罗　云, 唐丽晴

(武警海警学院 计算机教研室, 宁波 315801)

摘　要: 云计算资源调度是云计算中一个关键且复杂的调度问题, 需要考虑众多的因素. 为减少任务完成时间, 本文提出了一种云资源调度粒子群改进算法. 首先, 本文在惯性权重线性递减的基础上, 加入了混沌随机数扰动, 使惯性权重有概率的适度增加, 以便于跳出局部搜索, 进行全局搜索; 其次, 针对粒子群算法和蚁群算法都容易陷入局部最优的缺点, 结合粒子群算法和蚁群算法的优化策略, 提出了一种改进的混合优化策略. 其仿真结果及实际算例测试结果表明, 在相同条件下改进算法能够寻到更精确的解.

关键词: 云计算; 资源调度; 粒子群; 惯性权重递减; 混沌随机数扰动

引用格式: 罗云,唐丽晴.云计算调度粒子群改进算法.计算机系统应用,2019,28(7):151–156. http://www.c-s-a.org.cn/1003-3254/7005.html

Improved Particle Swarm Optimization Algorithm for Cloud Computing Scheduling

LUO Yun, TANG Li-Qing

(Department of Computer Application, China Coast Guard Academy, Ningbo 315801, China)

Abstract: Cloud computing resource scheduling is a key and complex scheduling problem in cloud computing, and many factors need to be considered. In order to reduce the time of cloud computing, an Improved Particle Swarm Optimization (IPSO) algorithm is proposed. Based on the linear decreasing inertia weight, the chaotic constant disturbance is added to increase the inertia weight with little probability, so as to get rid of the local search and get the global search. Meanwhile, in order to solve the defect that the two algorithms fall into partial optimization easily, the proposed algorithm combines the optimization strategy of particle swarm optimization and ant colony optimization. The Matlab simulation and the testing of practical examples results show that the improved algorithm can get a more accurate solution under the same condition.

Key words: cloud computing; resource scheduling; Particle Swarm Optimization (PSO); Linear Decreasing inertia Weight (LDW); chaotic constant disturbance

引言

云计算环境中存在着诸多不同类型的计算资源, 各种类型的资源之间存在着一定的差异. 云计算资源调度问题正是将云环境中的各类资源进行合理调度, 以达到缩短计算时间且提高资源利用率等目的的复杂调度问题. 随着“互联网+”和“自媒体”概念的日益升温, 长期以来被应用的调度效率较低的云计算资源调度算法不再能够满足人们的现实需求[1].

云计算资源调度算法研究已成为云计算领域的研究热点之一. 云计算资源调度需要同时兼顾计算效率和负载均衡, 这依赖于具有强大搜索能力的智能优化算法. 目前, 已经有诸多的智能优化算法应用于云计算资源调度中. 针对云计算服务集群资源调度和负载平衡的优化问题, 刘万军[2]等提出了一种适用于云计算资源调度的粒子群改进算法. 针对目前静态的网络资源调度算法不能满足动态的云计算资源调度要求的问题,

① 收稿时间: 2019-01-06; 修改时间: 2019-02-03, 2019-02-21; 采用时间: 2019-03-04; csa 在线出版时间: 2019-07-01

周文俊[3]等提出了一种基于预测及蚁群算法的云计算资源调度策略. 为了提高云计算资源的利用率, 以保持负载平衡, 孟令玺[4]等提出一种适用于云计算资源调度的文化粒子群算法. 为了提高云计算资源的调度效率, 袁浩[5]等提出了一种适合于云计算资源调度的社会力群智能优化算法. 针对传统资源调度算法中存在资源利用率低等缺陷, 卓涛[6]等提出一种基于改进人工蜂群算法的云计算资源调度模型 (Improved Artificial Bee Colony, IABC). 尽管上述的改进算法在一定程度上能够提升云计算资源调度的效率和精度, 但仍有一些不足之处.

云计算调度问题极难求得令人满意的解. 这一方面是因为云计算资源问题具有大规模、非线性、逻辑复杂的特点, 另一方面则是由于现有的智能优化算法及其改进策略都存在着对模型参数敏感、迭代后期容易陷入局部收敛等缺陷. 为有效弥合智能优化算法的缺陷以改善其优化性能, 本文提出了一种云计算调度粒子群改进算法. 基于云资源调度优化模型, 本文将混沌随机数扰动引入现有的惯性权重线性递减 (LDW) 粒子群算法中, 且结合了蚁群算法的寻优策略. 基于 2 种测试函数和 1 个云计算资源调度实际算例对 3 种不一样的优化算法进行比对试验, 其对比试验结果表明, 本文提出的粒子群改进算法的算法性能更佳.

1 云计算资源调度问题的优化模型

云计算资源调度问题的核心是节约其执行任务的时间. 因此, 云计算资源调度问题中最重要的优化目标即是其最大任务的执行时间, 记为$T_{\max}$. 此外, 云计算资源调度的重要优化目标是负载不平衡程度, 记为B_{degree}. 具体的以花费时间和负载平衡作为优化目标的优化模型为[7]:

$$\min \quad T_{\max} \quad B_{\text{degree}}$$
$$\text{s.t.}\begin{cases}\sum_{i=1}^{m} e_{ij}t_{ij} \leqslant T_{\max} & \forall j \in N \\ e_{ij} \in \{0,1\} & \\ \exists \sum_{j=1}^{n} e_{ij} = 1 & \forall i \in M\end{cases} \tag{1}$$

式 (2) 中, $M=\{1,2,3,\cdots,m\}$为任务集合, 含m个不同的任务; $N=\{1,2,3,\cdots,n\}$为节点集合, 含n个不同的节点; t_{ij}为第i个任务在第j个计算节点上的估算执行时间; e_{ij}是决策变量, 为第j个计算节点上执行第i个任务, 若是, 则$e_{ij}=1$, 否则, $e_{ij}=0$.

最大任务的执行时间$T_{\max}$采用下述公式计算得到:

$$T_{\max} = \max\left(\sum_{i=1}^{m} e_{ij}t_{ij}\right) \quad \forall j \in N \tag{2}$$

负载不平衡程度B_{degree}采用下述公式计算得到:

$$B_{\text{degree}} = E(e_{num}) \tag{3}$$

式 (3) 中, $e_{num}=\{e1_{num},e2_{num},\cdots,en_{num},\}$为$n$个不同节点的执行次数的集合, $E(X)$为集合X的方差.

上述优化模型求得最优需使得任务完成时间最短, 也即:

$$cost = \min(T_{\max}) \tag{4}$$

式中, $cost$为最短的任务完成时间.

由上述云计算资源调度问题的优化模型可知, 采用传统的梯度下降算法等常规算法是不易求得满意的优化解的, 因此, 一般会采用粒子群算法等智能优化算法进行求解.

2 改进的粒子群算法

2.1 粒子群算法

美国心理学家 Kennedy 和电气工程师 Eberhart 于 1995 年共同提出了粒子群算法 (PSO)[8]. 假设群体规模为N, 目标空间维度为D, 第i个粒子的坐标位置向为$\vec{X}_i=(x_{i1},x_{i2},\cdots,x_{iD})^{\mathrm{T}}$、速度向量为$\vec{V}_i=(v_{i1},v_{i2},\cdots,v_{iD})^{\mathrm{T}}$, 个体极值和全局极值分别为$\vec{P}_i=(p_{i1},p_{i2},\cdots,p_{iD})^{\mathrm{T}}$、$\vec{P}_g=(p_{g1},p_{g2},\cdots,p_{gD})^{\mathrm{T}}$. 具体的粒子群算法的更新迭代计算公式如式 (5) 所述.

$$\begin{cases} v_{i,t+1}^{d} = w*v_{i,t}^{d} + c_1*rand*(p_{i,t}^{d}-x_{i,t}^{d}) \\ \qquad\qquad + c_2*rand*(p_{g,t}^{d}-x_{i,t}^{d}) \\ x_{i,t+1}^{d} = x_{i,t}^{d} + v_{i,t+1}^{d} \end{cases} \tag{5}$$

式中, $i=1,2,\cdots,N$为粒子序号, t为粒子维度, d为迭代次数, w为权重因子, c_1,c_2为加速随机数, 一般来说, $c_1,c_2\in[0,2]$, $rand\in[0,1]$, 也为随机数.

2.2 惯性权重线性递减的粒子群算法

为改善粒子群算法极易陷入局部收敛的缺陷, 众多学者考虑了一种惯性权重系数ω 线性递减的改进策略. 具体的惯性权重线性递减的粒子群算法更新迭代计算公式如式 (6) 所述.

$$\begin{cases} v_{i,t+1}^d = \omega_{i,t}^d * v_{i,t}^d + c_1 * rand * (p_{i,t}^d - x_{i,t}^d) \\ \qquad + c_2 * rand * (p_{g,t}^d - x_{i,t}^d) \\ x_{i,t+1}^d = x_{i,t}^d + v_{i,t+1}^d \\ \omega_{i,t}^d = \omega_{\max} - \omega_k * t \end{cases} \tag{6}$$

式中, $\omega_{\max}$ 为惯性权重最大值, ω_k 为惯性权重递减斜率.

惯性权重ω是一个粒子群算法的重要参数. 其能够使粒子保持运动惯性, 从而有能力对新区域进行搜索. 惯性权重的增大对全局搜索有帮助, 有助于加快其收敛速度, 但不利于寻到精确解; 惯性权重的减小对局部搜索有帮助, 从而可以获得到更精确的解, 但代价却是降低了算法的收敛速度和增加了算法局部收敛的概率. 因此, 惯性权重的取值需要同时兼顾收敛速度和寻优精度. 由上述分析可知, 对于粒子群算法而言, 其迭代前期, 收敛速度的重要性较大; 随着迭代次数的增加, 寻优精度的重要性越来越大.

惯性权重线性递减 (Linear Decreasing inertia Weight, LDW) 的具体策略如下所述: 在 LDW 粒子群算法的整个过程中, 以进化代数为标准, 惯性权重以ω_k的递减斜率线性递减. 其惯性权重ω和进化代数t的关系曲线如图 1 所述.

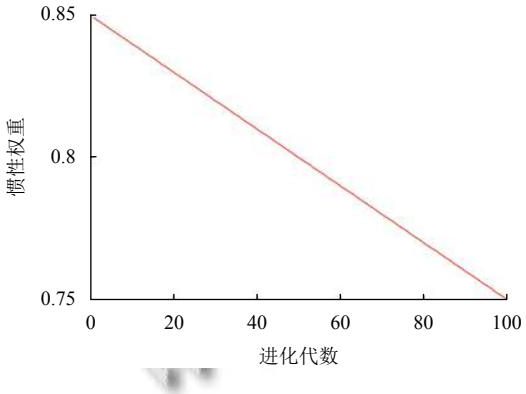

图 1 惯性权重线性递减下的惯性权重ω和进化代数t的关系曲线

2.3 增加混沌随机数扰动的惯性权重线性递减粒子群算法

惯性权重线性递减 (LDW) 粒子群算法较基本粒子群算法而言, 具有更好的寻优性能. 然而, 由于粒子群算法实际搜索过程具有非线性的特点, 于是惯性权重ω线性递减不能体现实际搜索过程的非线性的特点, 因而导致得不到足够理想的优化效果[9]. 因此, 粒子群算法实际计算过程中, 若算法通过长时间的计算都无法获得更理想的全局极值, 即迅速加入合理范围内的混沌随机数扰动, 以使得惯性权重ω突然增大, 从而起到抑制粒子群算法局部收敛的作用[10]. 其增加混沌随机数扰动的惯性权重线性递减的粒子群算法更新迭代计算公式如式 (7) 所述.

$$\begin{cases} v_{i,t+1}^d = \omega_{i,t}^d * v_{i,t}^d + c_1 * rand * (p_{i,t}^d - x_{i,t}^d) \\ \qquad + c_2 * rand * (p_{g,t}^d - x_{i,t}^d) \\ x_{i,t+1}^d = x_{i,t}^d + v_{i,t+1}^d \\ \omega_{i,t}^d = \omega_{\max} - \omega_k * t + A_t \\ A_t \in \{0, 0.1\} \end{cases} \tag{7}$$

式中, A_t为惯性权重的混沌扰动随机数, 在一定的扰动概率下, $A_t \in \{0, 0.1\}$, 其余, $A_t = 0$. 混沌现象是非线性系统中一种普遍的现象, 它的变化过程看似混乱, 实际上其内在具有规律性, 能够在一定范围之内, 按照自身规律不重复地遍历所有状态. 具体的惯性权重的混沌扰动随机数A_t的随机公式如下所示:

$$A_t = 0.1 \cdot rand^2 \cdot \sin(\pi \cdot rand) \tag{8}$$

其增加混沌随机数扰动的惯性权重ω和进化代数t的关系曲线如图 1 所述.

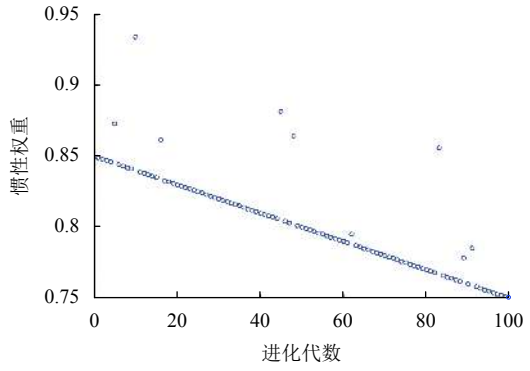

图 2 增加混沌随机数扰动的惯性权重线性递减下的惯性权重ω和进化代数t的关系曲线

2.4 蚁群算法

1992 年, 意大利学者 Dorigo 提出了蚁群算法[11]. 在蚁群算法中, 每只蚂蚁均在进行路径搜索后独立的产生一个可行解, 并产生信息素. 假设当前共有m个节点等待蚂蚁访问及选择, 在t时刻, 第k只蚂蚁正处于节点i处, 即将选择下一节点j进行移动, 其移动规则如下:

$$j = \begin{cases} \arg\max\limits_{u \in J_k(i)} [(\tau_{iu})^{\alpha} \cdot (\eta_{iu})^{\beta}], q \leqslant q_0 \\ J, q > q_0 \end{cases} \tag{9}$$

每只蚂蚁在构建可行解的同时分泌信息素, 局部信息素更新方式如下式所示:

$$\tau_{ij}^{new} \leftarrow (1-\rho)\cdot\tau_{ij}^{old}+\rho\cdot\tau_0 \tag{10}$$

当一次循环中的所有蚂蚁均构建了可行解后, 对该次循环产生的最优解与已知的最优解进行比较, 判断是否产生了新的当前的最优解, 若产生, 则对该解上的弧信息素进行更新, 如下式所示:

$$\begin{cases}\tau_{ij}^{new} \leftarrow (1-\rho)\cdot\tau_{ij}^{old}+\rho\cdot\tau_0\\ \tau_{ij}=\begin{cases}\dfrac{1}{L_{best}},(i,j)\in V_{best}\\ 0,(i,j)\notin V_{best}\end{cases}\end{cases} \tag{11}$$

式 (8)–(10) 中, 蚁群算法主要变量如下: τ_{ij}为弧(i,j)上的蚂蚁信息素密度, 初始值为τ_0; d_{ij}为弧(i,j)上的欧式长度; η_{ij}为弧(i,j)上的启发函数值, 通常$\eta_{ij}=1/d_{ij}$; α为蚂蚁信息素密度的相对重要性, $\alpha\geqslant 0$; β为启发函数的相对重要性, $\beta\geqslant 0$; ρ为蚂蚁信息素的衰减参数, $0<\rho<1$; L_{best}为当前最优解的路径长度; V_{best}为当前最优解.

2.5 混合优化算法改进策略

粒子群算法和蚁群算法的寻优模式相对单一, 总是全部个体向最优个体进行学习, 其最优解对种群的进化方向的影响是很大的. 为了克服采用单一的有相对固定寻优指引方向的寻优模式导致的算法容易陷入局部最优的缺点, 本文提出了一种同时混入蚁群优化策略的改进 LDW 粒子群优化策略的混合优化算法, 记为 Improved PSO. 目的在于在保持原有粒子群算法和蚁群算法搜索性能的同时, 防止算法因为寻优模式单一而导致的陷入局部收敛的缺点. 具体的计算步骤如下所述:

Step1. 初始化混合优化算法的各项参数: 种群规模$m+N$、粒子群规模 (粒子数量)N、迭代次数d、最大惯性权重$\omega_{\max}$、惯性权重系数的递减斜率ω_k、学习因子c_1,c_2、蚁群规模 (蚂蚁数量)m、蚂蚁信息素密度的相对重要性α、启发函数的相对重要性β、蚂蚁信息素的衰减参数ρ等, 其中的粒子群规模与蚁群规模相同$m=N$;

Step2. 随机生成$m+N$种不同的解, 也即生成初始种群, 初始种群包含初始粒子群和初始蚁群;

Step3. 计算N种解的适应度函数值, 并基于此对当前粒子群进行排序, 以得到个体极值和全局极值;

Step4. 采用增加混沌随机数扰动的惯性权重的线性递减 (LDW) 粒子群算法的更新规则对所有粒子进行更新;

Step5. 将这m只“蚂蚁”置于起始结点 0 上, 采用蚁群算法的信息素更新规则对所有蚂蚁进行更新;

Step6. 粒子群和蚁群交换一定数目k的个体, 其交换数目要小于粒子群规模和蚁群规模$k<\min(m,N)$;

Step7. 查看是否达到最大迭代次数, 如果是则转步骤 Step5, 否则转步骤 Step2;

Step8. 输出计算得到的最优解的适应度函数值.

本文中, 粒子群算法和蚁群算法的主要参数如下: 粒子群规模N为 30、迭代次数d为 50、最大惯性权重$\omega_{\max}$为 0.85、惯性权重系数的递减斜率ω_k为 0.001、学习因子c_1,c_2均为 0.5、蚂蚁信息素密度的相对重要性α为 1、启发函数的相对重要性β为 0.9、蚂蚁信息素的衰减参数ρ为 0.2.

3 仿真实验

基于 2 种不同的测试函数, 采用粒子群改进算法 (IPSO)、增加混沌随机数扰动的惯性权重线性递减 (LDW) 的粒子群算法[12] (linear decreasing inertia weight PSO with chaotic constant disturbance, CLDWPSO) 和普通的惯性权重线性递减粒子群算法[13] (Linear Decreasing inertia Weight PSO, LDWPSO) 求极小值. 具体的仿真结果如下所述.

(1) De jong 函数, 最优解为 0.0. 其 De jong 函数的解析式为$F(x_1,x_2)=100*(x_1-x_2)^2+(1-x_1)^2$.

具体的 De jong 函数的仿真测试结果如图 3 和表 1 所述.

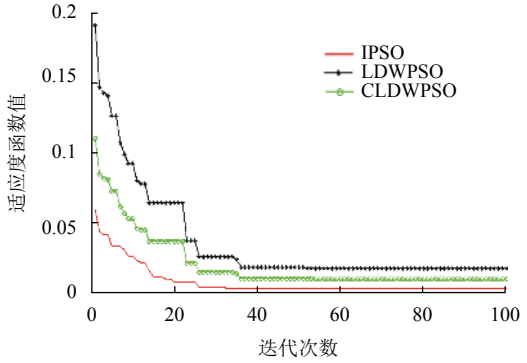

图 3 采用 De jong 函数的仿真测试收敛曲线

由图 3 和表 1 可知, 相比于其它算法, 粒子群改进算法 (IPSO) 最优, 寻优得到的最优解为: $(x_1,x_2)=$

(0.9928,0.9984); 最优解函数值$F(x_1,x_2)=0.0032$. 与此同时, 粒子群改进算法 (IPSO) 的收敛时间最短, 为 20.21 秒.

表 1 采用 De jong 函数的仿真测试结果

算法	(x_1,x_2)	$F(x_1,x_2)$	收敛时间
IPSO	0.9928, 0.9984	0.0032	20.21 秒
CLDWPSO	0.9793, 0.9894	0.0106	34.58 秒
LDWPSO	0.9794, 0.9925	0.0176	36.79 秒

(2) Schaffer 函数, 最优解 0.0. 其 Schaffer 函数的解析式为$F(x_1,x_2)=0.5+\frac{((\sin(x_1^2+x_2^2)^{0.5})-0.5)}{(1+0.001\times(x_1^2+x_2^2))^2}$.

具体的 Schaffer 函数的仿真测试结果如图 4 和表 2 所述.

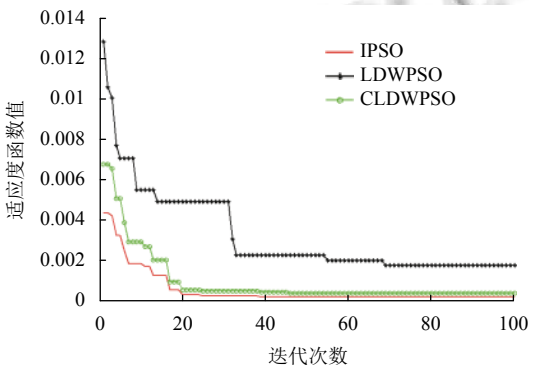

图 4 采用 Schaffer 函数的仿真测试收敛曲线

表 2 采用 Schaffer 函数的仿真测试结果

算法	(x_1,x_2)	$F(x_1,x_2)$	收敛时间
IPSO	0.00028, 0.00022	3.62×10^{-4}	43.86 秒
CLDWPSO	0.00045, 0.00033	5.58×10^{-4}	49.47 秒
LDWPSO	0.00165, 0.00102	1.94×10^{-3}	71.02 秒

由图 4 和表 2 可知, 粒子群改进算法 (IPSO) 同样最优, 寻优得到的最优解为: $(x_1,x_2)=(2.8\times10^{-4}, 2.2\times10^{-4})$; 最优解函数值$F(x_1,x_2)=3.62\times10^{-4}$. 与此同时, 粒子群改进算法 (IPSO) 的收敛时间最短, 为 43.86 秒.

4 实际算例测试

为了验证算法的改进效果, 本文采用粒子群改进算法 (IPSO)、增加混沌随机数扰动的惯性权重线性递减的粒子群算法 (CLDWPSO) 和普通的惯性权重线性递减粒子群算法 (LDWPSO) 求解了一个云资源调度实际算例的最短任务完成时间及其不平衡程度. 实际算例有 10 个计算节点和 100 个任务.具体的云资源调度问题的估算执行时间表如表 3 所述[14].

表 3 云资源调度问题的估算执行时间表

序号	节点 1	节点 2	节点 3	…	节点 10
任务 1	26.297 49	25.311 85	25.760 25	…	12.762 46
任务 2	21.112 87	24.255 67	28.514 73	…	13.292 85
任务 3	12.975 42	18.626 27	27.709 28	…	21.604 28
任务 4	12.609 57	17.848 85	19.399 58	…	27.446 53
任务 5	12.348 75	19.975 46	25.157 84	…	24.447 42
任务 6	10.450 89	17.428 29	26.665 32	…	26.985 13
任务 7	17.480 12	28.198 78	20.366 51	…	11.613 34
任务 8	21.985 28	21.816 52	13.555 69	…	17.768 71
任务 9	29.861 31	22.584 79	22.801 28	…	10.845 48
		…			
任务 100	13.634 57	16.571 43	24.623 54	…	21.801 23

具体的云资源调度问题的实测收敛曲线如图 5 和图 6 所述.

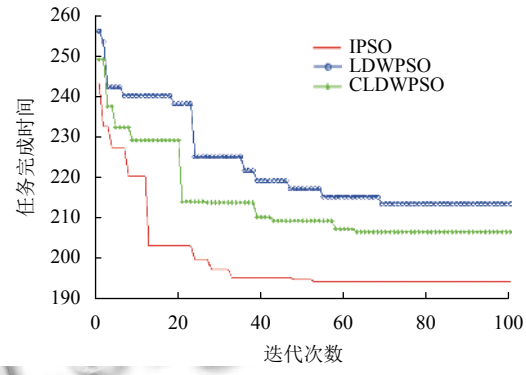

图 5 云资源调度问题任务完成时间的实测收敛曲线

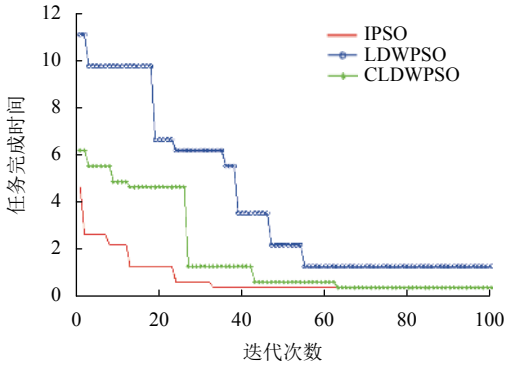

图 6 云资源调度问题负载不平衡程度的实测收敛曲线

由图 5 和图 6 可知, 相比于其它算法, 粒子群改进算法 (IPSO) 最优, 求解得到的任务执行节点序列为

e = [7 4 1 7 5 7 10 …9], 估算的任务完成时间为194.4568. 与此同时, 粒子群改进算法 (IPSO) 寻到的调度方案的负载不平衡程度最小, 为 0.4444, 其调度方案的计算节点的执行次数的集合为e_{num} = [10 10 9 …9].

5 结论

本文基于云资源调度问题的优化模型, 提出了一种粒子群改进算法 (IPSO). 首先, 基于惯性权重线性递减粒子群算法, 引入适当的混沌随机数扰动; 其次, 将蚁群算法寻优策略引入粒子群算法中以改善其全局优化性能. 本文采用了 2 种测试函数 (De jong、Schaffer) 和 1 个云计算资源调度实际算例来验证优化算法的性能. 其验证结果表明, 相比于其他两种优化算法, 本文提出的改进算法 (IPSO) 更佳.

参考文献

1 Fan GS, Yu HQ, Chen LQ. A formal aspect-oriented method for modeling and analyzing adaptive resource scheduling in cloud computing. IEEE Transactions on Network and Service Management, 2016, 13(2): 281–294. [doi: 10.1109/TNSM.2016.2553157]

2 刘万军, 张孟华, 郭文越. 基于 MPSO 算法的云计算资源调度策略. 计算机工程, 2011, 37(11): 43–44, 48. [doi: 10.3778/j.issn.1002-8331.2011.11.013]

3 周文俊, 曹健. 基于预测及蚁群算法的云计算资源调度策略. 计算机仿真, 2012, 29(9): 239–242, 246. [doi: 10.3969/j.issn.1006-9348.2012.09.058]

4 孟令玺, 李洪亮. 基于 CA-PSO 算法的云计算资源调度策略. 计算机仿真, 2013, 30(10): 406–410. [doi: 10.3969/j.issn.1006-9348.2013.10.093]

5 袁浩, 李昌兵. 基于社会力群智能优化算法的云计算资源调度. 计算机科学, 2015, 42(4): 206–208, 243. [doi: 10.11896/j.issn.1002-137X.2015.04.041]

6 卓涛, 詹颖. 改进人工蜂群算法的云计算资源调度模型. 微电子学与计算机, 2014, 31(7): 147–150, 155.

7 袁正午, 李君琪. 基于改进粒子群算法的云资源调度. 计算机工程与设计, 2016, 37(2): 401–404, 412.

8 Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks. Perth, WA, Australia, 1995: 1942–1948.

9 钟臻, 张楷旋, 吴贞龙, 等. 基于改进的 LDW 粒子群算法的风-火电力系统联合优化调度策略. 贵州电力技术, 2017, 20(10): 50–55.

10 陈艳. 物流配送路线优化粒子群改进算法. 浙江水利水电学院学报, 2019, 31(2): 69–74.

11 游晓明, 刘升, 吕金秋. 一种动态搜索策略的蚁群算法及其在机器人路径规划中的应用. 控制与决策, 2017, 32(3): 552–556.

12 张选平, 杜玉平, 秦国强, 等. 一种动态改变惯性权的自适应粒子群算法. 西安交通大学学报, 2005, 39(10): 1039–1042. [doi: 10.3321/j.issn:0253-987X.2005.10.001]

13 Gao YL, An XH, Liu JM. A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. Proceedings of 2008 International Conference on Computational Intelligence and Security. Suzhou, China, 2008: 61–65.

14 刘钙. 物联网平台下基于云计算的智能药盒系统[硕士学位论文]. 镇江: 江苏科技大学. 2018.