计算机科学
COMPUTER SCIENCE

基于改进粒子群算法的云数据中心能耗优化任务调度策略

刘陈伟, 孙鉴, 雷冰冰, 徐涛, 吴佳伟

引用本文

刘陈伟, 孙鉴, 雷冰冰, 徐涛, 吴佳伟基于改进粒子群算法的云数据中心能耗优化任务调度策略[J]. 计算机科学, 2023, 50(7): 246-253.

LIU Chenwei, SUN Jian, LEI Bingbing, XU Tao, WU Zhuiwei. Task Scheduling Strategy for Energy Consumption Optimization of Cloud Data Center Based on Improved Particle Swarm Algorithm [J]. Computer Science, 2023, 50(7): 246-253.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900176

基于改进粒子群算法的云数据中心能耗优化任务调度策略

刘陈伟[1]　孙　鉴[1,2]　雷冰冰[1,2]　徐　涛[1]　吴隹伟[1]
1 北方民族大学计算机科学与工程学院　银川 750021
2 北方民族大学图像图形智能处理国家民委重点实验室　银川 750021
(liuchenwei@tiangong.edu.cn)

摘　要　随着云计算的发展,能耗急剧上升,这进一步限制了云数据中心整体性能的提高,因此能耗问题引起了工业界和学术界的重视。同时,传统粒子群算法被广泛应用于数据中心任务调度问题的求解,但其收敛速度慢、精度低,容易忽略集群能耗问题。为此提出了一种基于反向学习的混沌映射自适应粒子群算法(OAPSO)。首先,采用反向学习的方法产生初始种群,使粒子更加均匀地分布于初始解空间,提高了初始种群的质量;其次,在粒子更新方式中引入非线性递减的动态惯性权重策略,以改变粒子的寻优能力,使局部搜索和全局搜索达到平衡,避免算法陷入局部最优;然后,引入混沌映射策略,在最优解位置进行扰动变异产生新解,提高算法从局部最优中跳出的能力。最后,在Cloudsim平台上对所提算法进行实验验证,结果表明,与PSO,OBL_TP_PSO和SAPSO算法相比,OAPSO算法资源利用率更高,节能效果更好。

关键词:云数据中心;任务调度;粒子群算法;混沌映射;能耗优化
中图法分类号　TP393

Task Scheduling Strategy for Energy Consumption Optimization of Cloud Data Center Based on Improved Particle Swarm Algorithm

LIU Chenwei[1],SUN Jian[1,2],LEI Bingbing[1,2],XU Tao[1] and WU Zhuiwei[1]
1 School of Computer Science and Engineering,North Minzu University,Yinchuan 750021,China
2 Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission,North Minzu University,Yinchuan 750021,China

Abstract　With the development of cloud computing,energy consumption has increased dramatically,which further limits the improvement of the overall performance of the cloud data center,and thus the energy consumption issue has attracted the attention of industry and academia. Meanwhile,traditional particle swarm optimization algorithm(PSO) is widely used to solve data center task scheduling problems,but it has the shortcomings of slow convergence and low accuracy,and it is easy to ignore the cluster energy consumption problem. A chaotic mapping adaptive particle swarm optimization algorithm based on opposition-based learning(OAPSO) is proposed. Firstly,the initial population is generated by the method of opposition-based learning,which makes the particles more evenly distributed in the initial solution space and improves the quality of the initial population. Secondly,a nonlinear decreasing dynamic inertia weight stra-tegy is introduced into the particle updating mode to change the particle optimization ability,so as to balance the local search and global search and avoid the algorithm falling into the local optimal. Thirdly,the chaotic mapping strategy is introduced to generate new solutions by perturbation and mutation at the optimal location,which improves the ability of the algorithm to jump out of the local optimal. Finally,the proposed algorithm is verified by experiments on the Cloudsim platform,and the results show that,compared to PSO,OBL_ TP_PSO and SAPSO,OAPSO algorithm has higher resource utilization and better energy-saving effect.

Keywords　Cloud data center,Task scheduling,Particle swarm optimization,Chaotic mapping,Energy consumption optimization

1　引言

云计算作为一种新型的商业服务模式[1],具有可伸缩、虚拟化、按需服务、高可靠性和高性价比等优点[2],越来越多的企业也开始利用云计算平台实现数字化转型。云计算的快速发展使得云服务商需要专门建立数据中心以存放数据。

到稿日期:2022-09-19　返修日期:2023-02-08
基金项目:国家自然科学基金(62062002,62102201);宁夏自然科学基金(2022AAC03289,2022AAC03245,2021AAC03034);北方民族大学中央高校基本科研业务费专项资金(FWNX09);北方民族大学校级一般项目(2021XYZJK01)
This work was supported by the National Natural Science Foundation of China(62062002,62102201),Natural Science Foundation of Ningxia,China(2022AAC03289,2022AAC03245,2021AAC03034),Fundamental Research Funds for the Central Universities of Ministry of Education of China(FWNX09) and Research Project of North Minzu University(2021XYZJK01).
通信作者:孙鉴(2014132@nun.edu.cn)

随着计算资源数量的增加，当系统处于空闲状态时，会产生大量的能源损耗，造成一定的经济损失。能耗过高、不具备可持续发展能力，是云计算面临的一大问题。

据 UNEP DTU Partnership 估计，全球数据中心用电量在全球用电量中的份额在 2025 年将从目前的 3% 上升到 20%[3-4]。从国内来看，2020 年的《中国数据中心可再生能源应用发展报告》中显示，预计到 2030 年，数据中心的能耗总量将在此基础上翻一倍，达到全社会总用电量的 1.5%～2%[5]。数据中心在能量消耗巨大的同时资源利用率却极低，相关数据显示，目前数据中心的资源利用率通常在 5%～25%之间，资源浪费严重[6-7]。这让云计算能效对环境影响的问题更为突出。

与目前的从数据中心每个集群组件着手降低能耗的方法相比，将整个数据中心的池化资源视为一个整体来寻找节能方案更为高效。因此，如何提高资源利用率以降低成本和能源损耗，为云计算设计高能效的任务调度策略已经迫在眉睫。

为了避免能耗方面产生较大的开销，研究者从各个角度采用一些算法来解决任务调度问题。传统的启发式任务调度算法实现简单，但容易陷入局部最优，有一定的局限性，不能满足用户多样化的需求。相比之下，近年来国内外许多学者青睐于用能够较快找到全局最优值的元启发式算法（如遗传算法[8-10]、蚁群算法[11-12]、禁忌搜索算法[13]、模拟退火算法[14-15]等）来求解此类问题。

虽然现有算法在针对各自的问题上都取得了良好的效果，但它们都没有同时关注降低能耗和完成时间。与这些研究相比，OAPSO 算法侧重于在保持算法性能的同时最小化能耗。为了避免粒子群优化算法的早熟收敛，该算法通过优化初始种群、改变惯性权重和引入混沌映射策略自适应粒子群优化来获得全局最优解。

针对云数据中心任务调度问题，本文提出了一种基于反向学习的混沌映射自适应粒子群任务调度策略，通过缩短任务完成时间来平衡能耗和用户成本之间的关系，实现降低能耗的调度目的。本文的主要贡献如下：

（1）在粒子更新方式中引入非线性递减的动态惯性权重策略，通过改变粒子的寻优能力来平衡算法的局部搜索和全局搜索，避免算法陷入局部最优。

（2）引入混沌映射策略，在最优解位置进行扰动变异产生新解，增强算法跳出局部最优的能力。

（3）在 Cloudsim 平台上，将所提 OAPSO 算法与 PSO，OBL_TP_PSO 和 SAPSO 算法进行了大量的对比实验，结果表明，OAPSO 算法在最大完工时间、执行时间和能耗方面均优于对比算法。

2 相关工作

2.1 启发式算法

许多研究者致力于研究云数据中心调度算法以有效利用资源。经典的 Min-Min 算法[16]首先考虑执行小任务来缩短任务完成时间，但该算法按照任务长度进行排序形成任务队列，容易造成系统负载不均衡甚至空载。为了解决负载均衡问题，在 Min-Min 算法的基础上，文献[17]提出了 Max-Min 算法。虽然这两种算法有效地解决了负载均衡问题，但牺牲了任务完成时间，系统能耗过高。

2.2 元启发式算法

传统的启发式任务调度算法实现简单，但容易陷入局部最优，有一定的局限性，不能很好地满足用户的多样化需求。在处理数据中心任务调度问题时，粒子群算法收敛速度慢、精度低，容易忽略集群能耗问题。为此，Eberhart 等[18]提出了传统的粒子群算法来解决云环境下的任务调度问题，该算法相较于传统启发式算法可以满足用户的多样化需求，但是算法收敛速度过快，最终只能得到局部的最优解。Zhou 等[19]提出在传统粒子群算法的基础上引入反向学习和试探感知的方法来解决任务调度问题，提出了 OBL_TP_PSO 算法，旨在满足用户的服务质量（Quality of Service，QoS）需求，使得算法在负载和能耗方面有明显的改善。WANG 等[20]通过整合模拟退火算法（Simulated Annealing，SA）与粒子群算法的优势，提出了一种融合调度算法（Hybrid Particle Swarm Optimization with Simulated Annealing，SAPSO）。该算法通过模拟退火算法的思想来控制粒子群的惯性权重系数，达到优化的目的，但只考虑了优化执行时间。在 QoS 约束的基础上，考虑到任务完成时间、虚拟机能耗、用户成本、可靠性和负载平衡等不同参数，Pandey 等[21]提出了一种基于粒子群优化的任务调度问题算法，以最小化成本和时间。Ritu 等[22]提出了一种多目标列表调度算法，以优化科学工作流约束向量划分的能耗、可靠性、成本和完成时间。然而，该算法对于较少数量的任务是低效的。文献[23-24]使用遗传算法解决任务调度问题，目标是优化完工时间和执行成本。然而，遗传算法对于任务调度问题效率不高。此外，粒子群优化算法的收敛速度比遗传算法快，而且使用的算法参数比遗传算法少，这使得算法对参数调整的依赖性更小。

3 云数据中心任务调度与能耗模型

3.1 云数据中心任务调度

任务调度就是在任务执行时将任务调度到适应性资源上，为任务的执行创建适当的顺序，使任务在一定的约束条件下执行，分配处理器上的负载，使任务完成时间最小化。云计算环境下的任务调度问题是一个关系到整个云计算设施的效率的组合优化问题[25]。调度问题可概括为在 b 台虚拟机上执行 a 个子任务（$a \geqslant b$），调度目标为最小化任务完成时间。

假设用户提交了 a 个相互独立的任务，且数据中心存在 b 个资源，其中 $a \geqslant b$。定义矩阵 $\boldsymbol{Time}(a,b)$ 为在第 b 个资源上执行第 a 个任务所需要的时间，$Task(a)$ 表示任务 a 的大小，$Mips(b)$ 表示资源 b 的计算能力。在资源 b 上执行任务 a 所需要的时间如式(1)所示：

$$T_{ab}=\frac{Task(a)}{Mips(b)} \tag{1}$$

此时任务到各个资源上执行的时间矩阵 $\boldsymbol{Time}(a,b)$ 如式(2)所示：

$$\boldsymbol{Time}(a,b)=\begin{pmatrix} T_{11} & \cdots & T_{1b} \\ \vdots & \ddots & \vdots \\ T_{a1} & \cdots & T_{ab} \end{pmatrix} \tag{2}$$

系统执行完资源 b 上的任务所需要的时间可用 FT_b 表示。当资源 b 上分配了{1,2,3,k}这 4 个任务时，执行完资源 b 上所有的任务所花费的时间如式(3)所示：

$$FT_b = T_{1b} + T_{2b} + T_{3b} + T_{kb} \tag{3}$$

任务最大完成时间 Makespan 指虚拟机上最后一个任务的完成时间，如式(4)所示：

$$Makespan = Max_{\{FT_s | s=1,2,3,\cdots,\alpha\}} \tag{4}$$

系统执行任务的总时间即每台虚拟机执行任务所需的时间之和，如式(5)所示：

$$T_s = \sum_{i=1}^{m} T_i \tag{5}$$

其中，T_s 表示系统执行任务的总时间；m 为系统中虚拟机的台数；T_i 为第 i 台虚拟机执行任务的时间。T_i 的计算式如式(6)所示：

$$T_i = \sum_{j=1}^{n} \left(\frac{task_j \cdot len}{VM_i \cdot Mips_i} \cdot S_{i,j} \right) \tag{6}$$

其中，n 为任务数；$task_j \cdot len$ 表示第 j 个任务的长度；$VM_i \cdot Mips_i$ 表示第 i 台虚拟机的执行速度；$S_{i,j}$ 用于表示任务是否被分配到虚拟机，计算式如式(7)所示：

$$S_{i,j} = \begin{cases} 1, & \text{如果 } task_j \text{ 被分配到 } VM_i \\ 0, & \text{如果 } task_j \text{ 未被分配到 } VM_i \end{cases} \tag{7}$$

负载均衡因子代表系统的负载均衡度，其公式如式(8)所示。为使系统更均衡，应尽量使负载均衡因子 L 较小。

$$L = \sqrt{\frac{1}{m} \sum_{j=1}^{m} (T_j - T_{\text{javg}})^2} \tag{8}$$

其中，$T_{j\,\text{avg}}$ 为所有虚拟机的平均完成时间，其定义如式(9)所示：

$$T_{j\,\text{avg}} = \frac{1}{m} \sum_{j=1}^{m} T_j \tag{9}$$

3.2 云数据中心优化模型

云计算集群的能耗源于多个方面，主要分为调度能耗、动态能耗和静态能耗[26]，动态能耗占云集群总能耗的主要部分。研究表明集群的能耗与集群中每个虚拟机的完成时间基本呈线性关系[27]。

数据中心中由物理机产生的能源消耗主要取决于 CPU、内存、磁盘存储和网络接口等的消耗。而相比其他系统资源，CPU 又占据了其中大部分的能耗[28-30]。物理机的 CPU 利用率与能源消耗存在一定的线性关系[31]，每个虚拟机有满载和空闲两种状态，与满载状态相比，空闲状态下的虚拟机的能耗减少了 60%[32]。因此，关闭数据中心空闲的物理机能够降低能源消耗，服务器能耗模型的定义如式(10)所示：

$$P(u) = k \cdot P_{\max} + (1-k) \cdot P_{\max} \cdot u \tag{10}$$

其中，$P_{\max}$ 是服务器满负载时的最大耗能；k 为服务器空闲时的耗能与服务器满负载时耗能的比例；u 为服务器 CPU 的利用率。

由于 CPU 利用率是随着时间变化的，那么在某段时间内服务器能耗可用 P 在这个时间的积分来表示，如式(11)所示：

$$E = \int_{T_1}^{T_2} P(u)\,\mathrm{d}T \tag{11}$$

综上所述，系统消耗的总能量(经济消耗)[33-35]由式(12)给出。

$$E = P(u) \cdot T_s \tag{12}$$

其中，E 为一台服务器在时间 T 内消耗的总能量；T_s 对应服务器运行的总时间；P 为服务器的功率。

本文提出的任务调度策略同时关注云数据中心能耗和完成时间，因此任务调度的目标函数为：min(E)及 min($Makespan$)。

4 基于反向学习的混沌映射自适应粒子群算法

4.1 标准粒子群算法

粒子群优化算法[36-38](Particle Swarm Optimization，PSO)是由 Eberhart 等于 1995 年提出的基于种群的优化算法，它模拟了动物群的行为，如鸟群和鱼群，群中的每个成员都被称为粒子。粒子群优化算法是一种元启发式算法，它在较短的计算时间内提供近似最优解，在质量和计算时间方面均优于传统算法，可用于求解连续问题和离散问题。粒子群优化算法的目的是通过粒子或分子之间的协作和数据共享来寻找最优解。

PSO 初始化为一组随机粒子，通过迭代求解得到最优解。在每一次迭代中，粒子都会根据粒子本身找到的最优解 P_i 和整个种群当前找到的最优解 P_g 两个极值来更新它们的位置和速度。假设粒子群在 d 维目标搜索空间中包含 N 个粒子，则在 $t+1$ 时刻第 i 个粒子的速度和位置可以根据 t 时刻的速度 $V_i^d(t)$ 和位置 $X_i^d(t)$ 进行更新，如式(13)所示：

$$\begin{cases} V_i^d(t+1) = \omega V_i^d(t) + c_1 r_1 [P_i^d(t) - X_i^d(t)] + \\ \qquad c_2 r_2 [P_g^d(t) - X_i^d(t)] \\ X_i^d(t+1) = X_i^d(t) + \alpha V_i^d(t+1) \end{cases} \tag{13}$$

其中，$V_i^d(t)$ 和 $X_i^d(t)$ 表示粒子的速度和位置；$P_i^d(t)$ 表示第 i 个粒子的历史个体最优，$P_g^d(t)$ 表示整个种群全局最优；r_1 和 r_2 分别为[0,1]内的随机值；α 为约束因子，用来控制速度的权重；ω 为惯性权重；c_1 和 c_2 为学习因子。

4.2 适应度函数设计

适应度函数是对所求解问题的抽象表述，通常用一个数学模型来表示具体问题，它是各种优化算法的优化目标[39]。适应度函数设计是否合理，直接影响到最优解的选取与算法的收敛速度，种群个体的生存能力在一定程度上由适应度值决定。

云服务商致力于以最小的任务执行总时间和任务最大完成时间来满足云用户的需求[40]。因此，在设计适应度函数时要综合考虑优化系统执行任务的最大完成时间和能耗来提高资源利用率。为了将两个相差较大的优化对象整合到一起，我们对每一个对象取对数再取倒数最后相加得到目标函数[41-42]，具体的适应度函数设计过程如式(14)所示。云数据中心任务调度的优化目标转变为求适应度函数的最优值。适应度函数值越大，说明个体越优。

$$fitness = \frac{1}{\log_2(Makespan)} + \frac{1}{\log_2(E)} \tag{14}$$

4.3 反向学习

反向学习(Opposition-based Learning，OBL)最初由 Rahnamayan 等提出[43]，并成功地应用于多智能优化算法，其主要思想是同时评估当前可行解及其对应的反向解，并为下一代迭代计算选择一个更优解[44]。

当用粒子群算法求解优化问题时，得到的解对初始种群的质量非常敏感。找到合适的初始粒子群优化方法是解决这一问题的关键。因此，本文利用反向学习方法生成初始种群来提高初始种群的质量。

设 $\boldsymbol{X}^i=a+b-\boldsymbol{X}_i$，其中 $\boldsymbol{X}_i$ 是源数据，$\boldsymbol{X}^i$ 是目标数据，a 和 b 分别是区间(a,b)的下限和上限，此时称 $\boldsymbol{X}^i$ 是 $\boldsymbol{X}_i$ 的反向向量。由此延伸到多维向量，假定 n 维向量 $\boldsymbol{X}_i=(\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{X}_3,\cdots,\boldsymbol{X}_n)$每个维度的取值区间分别为$(a_k,b_k)$，那么由 $\boldsymbol{X}_i$ 得到的反向向量，如式(15)所示：

$$\boldsymbol{X}^i=(a_1+b_1-\boldsymbol{X}_1,a_2+b_2-\boldsymbol{X}_2,a_3+b_3-\boldsymbol{X}_3,\cdots,a_n+b_n-\boldsymbol{X}_n) \tag{15}$$

此时，假设在空间中有 n 个粒子组成一个称为原始种群的种群。其中 P_i 代表粒子 i，它在空间中具有一定的速度和位置。$\boldsymbol{V}_i$ 表示其速度矢量，$\boldsymbol{S}_i$ 表示其位置矢量，那么第 i 个粒子表示为 $P_i=\{\boldsymbol{V}_i,\boldsymbol{S}_i\}$。反向学习方法表明空间中一定存在一个反向粒子 $P^i=\{\boldsymbol{V}^i,\boldsymbol{S}^i\}$，此时，由原始种群反向学习构成的种群，称为反向种群。

采用粒子群优化算法求解优化问题时，假设种群规模为 n，反向学习生成初始种群的步骤如下：

(1)随机初始化 n 个粒子，形成原始种群。

(2)通过反向学习产生 n 个反向粒子，形成反向种群。

(3)计算初始种群和反向种群中对应的原粒子和反向粒子的适应度值，并从两者中选择最佳粒子，形成粒子群优化算法的初始种群。

4.4 混沌映射

初始化种群的多样性会极大地影响群智能算法的收敛速度和准确性，而 PSO 的随机初始化不能保证种群的多样性，容易出现收敛过早、陷入局部最优等问题。当粒子发生早熟时，必须有一个外部扰动机制，使粒子群从“早熟”状态中跳出。

混沌映射具有随机性、遍历性和规律性。混沌变量优化搜索可以加快算法的搜索速度，避免陷入局部最优。它在智能算法优化中得到了广泛的应用，并取得了良好的效果。常见的混沌映射有 Logistic 映射、Tent 映射和 Cat 映射[45]。为了更好地利用解空间，引入 Logistic 映射来初始化种群，混沌扰动公式如式(16)所示：

$$X_{n+1}=\mu X_n(1-X_n) \tag{16}$$

其中，μ 为混沌参数。

4.5 非线性递减的动态惯性权重

惯性权重 ω 的取值会影响 PSO 的性能，它是维持粒子群算法全局搜索和局部搜索的最重要的控制因子[46]。全局搜索和局部搜索的平衡在寻找最优解方面起着关键作用。ω 的值较大，有利于全局搜索，可以加快求解速度；ω 的值较小，有利于局部搜索，能够提高解的精度。标准 PSO 中惯性权重 ω 的取值固定，不利于在算法运行过程中实现全局搜索和局部搜索的平衡。

在 PSO 运行早期，粒子相对分散，具有较好的多样性，这时应充分利用算法的最大惯性系数，提高粒子群优化算法的全局搜索能力；在算法运行后期，随着迭代次数的增加，粒子越来越聚集，这时充分利用最小惯性系数可以提高局部搜索能力[47]。很多算法将惯性权重设置为随迭代次数的增加线性递减，在一定程度上提高了算法的性能，但是线性递减策略对于动态系统的效果并不理想。因此，提出了一种随迭代次数非线性递减的动态惯性权重策略，如式(17)所示：

$$\begin{cases}\omega_{\max}+\dfrac{\omega_{\min}+\omega_{\max}}{\mathrm{e}^{-\omega_{\max}}+\mathrm{e}^{-1.2+20\frac{t}{T_{\max}}}}, & 0\leqslant\dfrac{t}{T_{\max}}\leqslant 0.1\\ \omega_{\max},\text{otherwise}\end{cases} \tag{17}$$

其中，$\omega_{\max}$ 表示惯性权重最大值，$\omega_{\min}$ 表示惯性权重最小值，t 表示当前迭代步数，$T_{\max}$ 表示最大迭代数。

4.6 算法描述

(1)初始化全局粒子速度惯性和随机产生的 n 个原始粒子，构成种群大小为 n 的原始种群。

(2)通过反向学习生成 n 个原始粒子的反向粒子，形成 n 个粒子大小的反向学习种群。

(3)从原始种群和反向学习种群中选出较优的前 n 个粒子，形成初始粒子群。

(4)计算每个粒子的适应度值，更新个体和群体粒子历史最佳适应度值，自适应更新粒子的全局速度惯性。

(5)采用传统粒子群算法规则更新粒子的速度和位置。

(6)计算每个粒子的适应度与该粒子的个体历史最佳适应度之间的差值。

(7)若差值较小，则开启混沌变异策略，更新粒子的速度和位置，否则不开启混沌变异策略。

(8)确定是否满足循环结束条件：循环达到预定的最大循环数或连续 10 次循环有相同结果。如果是，则退出循环，否则返回步骤(4)。

算法的流程图如图 1 所示。

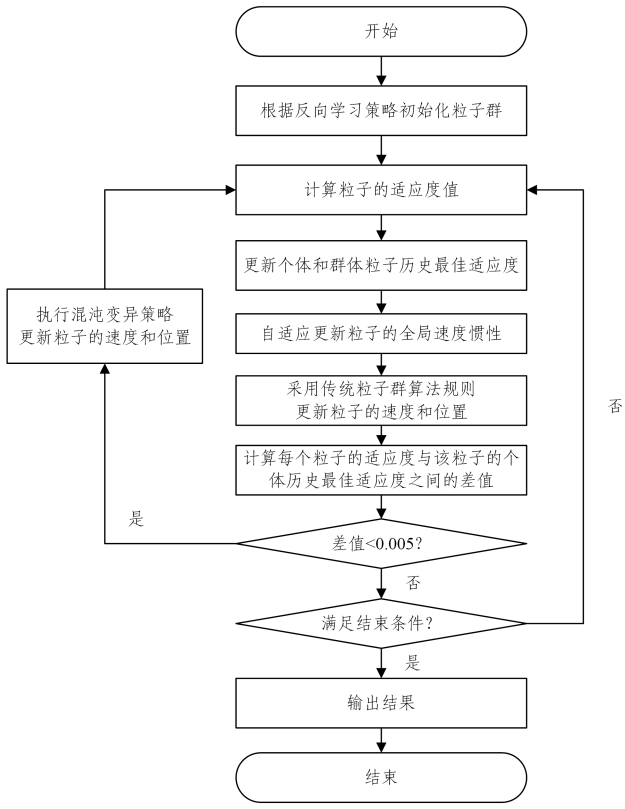

图 1 算法流程图

Fig. 1 Algorithm flow chart

5 仿真实验结果与分析

CloudSim[48]是澳大利亚墨尔本大学网格实验室和 Gridbus 项目联合研发的云计算仿真平台，该平台主要对云环境进行仿真，并对不同服务模型的调度策略进行测试。为了验证和评价本文提出的算法的性能，CloudSim 平台在 Intel i5 处理器、16 GB 内存、Windows 10 操作系统上进行了测试。实验的初始化参数如表 1 所列。将本文提出的 OAPSO 算法进行比较，使用 Makespan、系统执行任务总时间、能源消耗和适应度函数值作为评价指标，验证了该算法的性能较好。

实验选取 5 台虚拟机，虚拟机参数如表 2 所列。根据任务的数量，我们将云数据中心任务规模分为两类：1)小规模实验条件的部署，调度的任务数量为 100～200；2)大规模任务调度场景，任务数量为 1 000～2 000。算法中的任务大小都设定为[100，1 000 000]区间均匀分布的随机数，设定种群规模为粒子数和任务总数保持一致。利用表 3 的算法参数及初始化值测试算法的性能。为了验证所提算法的优劣，选取了 PSO 算法、SAPSO 算法和 OBL_TP_PSO 算法与其在不同任务量下进行实验对比分析。在相同实验环境下每种算法都独立运行 20 次，最后记录并保存实验结果以保证实验的准确性。

表 1 实验初始化参数

Table 1 Experimental initialization parameters

参数	参数含义	初始化值
P_{max}	机器满负载能耗	250
k	服务器空闲时的耗能与服务器满负载时耗能的比例	0.7

表 2 虚拟机参数

Table 2 Virtual machine parameters

VM 编号	VM 性能/MIPS	内存/GB	带宽/(GB/s)
1	100	512	1
2	200	512	1
3	300	512	1
4	400	512	1
5	500	512	1

表 3 算法参数及初始化值

Table 3 Algorithm parameters and initialization values

参数	参数含义	初始化值
ω	惯性权重	0.4
ω_{max}	惯性权重最大值	0.9
ω_{min}	惯性权重最小值	0.4
c_1	学习因子	2.0
c_2	学习因子	2.0
N	粒子种群规模	100，200，1 000，2 000
Iteration	迭代次数	300
α	速度约束因子	1.0
μ	混沌参数	4.0

5.1 任务最大完成时间对比

在实验中，所有的性能指标都是针对特定数量的任务(100 到 2 000)和不同的种群规模进行评估的。表 4 列出了 Makespan 随任务增加的变化。如图 2 所示，Makespan 随任务数的增加而增加。OAPSO 算法的任务最大完成时间明显更少，节省了资源，更适合解决云数据中心任务调度问题。

表 4 任务最大完成时间对比

Table 4 Comparison of makespan of tasks

任务规模/个	PSO	OBL_TP_PSO	SAPSO	OAPSO
100	43 313	35 553	33 489	33 006
200	100 581	81 665	73 267	69 057
1 000	729 646	692 629	543 261	447 664
2 000	1 607 748	1 540 230	1 277 177	924 569

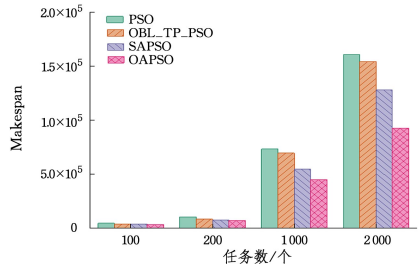

图 2 任务最大完成时间对比

Fig. 2 Comparison of makespan of tasks

5.2 任务执行总时间对比

表 5 列出了 20 次实验中 4 种算法在任务数为 100，200，1 000，2 000 时的任务执行总时间。从结果来看，随着任务数的增加，OAPSO 算法的性能有更明显的提升。无论小规模任务(任务数为 100，200)还是大规模任务(任务数为 1 000，2 000)，OAPSO 算法的任务执行总时间都是最短的。

表 5 任务执行总时间对比

Table 5 Comparison of total task execution time

任务规模/个	PSO	OBL_TP_PSO	SAPSO	OAPSO
100	188 904	173 018	165 895	155 410
200	399 722	364 407	343 670	339 910
1 000	2 253 874	2 205 620	2 040 721	1 881 251
2 000	4 662 323	4 601 617	4 380 047	3 757 212

如图 3 所示，OAPSO 算法的任务执行总时间整体上要少于 PSO 算法，这是因为 OAPSO 算法在迭代过程中能够跳出局部最优，在提高解精度的同时以更好的方式收敛得到解决方案。

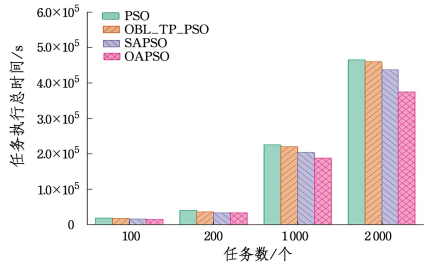

图 3 任务执行总时间对比

Fig. 3 Comparison of total task execution time

5.3 能源消耗对比

从表 6 中可见，在小规模任务中，算法的能源消耗差别较小，此时 OAPSO 算法的能耗最小。随着任务规模的增大，能源消耗增大，此时 OAPSO 的节能效率更为突出，如图 4 所示。可见，OAPSO 算法更适合用于解决大规模任务调度问题。

表 6 能源消耗对比

Table 6 Comparison of energy consumption

任务规模/个	PSO	OBL_TP_PSO	SAPSO	OAPSO
100	1.672	1.660	1.612	1.491
200	3.298	3.270	3.212	3.209
1 000	16.546	16.487	16.482	16.426
2 000	33.105	33.059	32.899	32.583

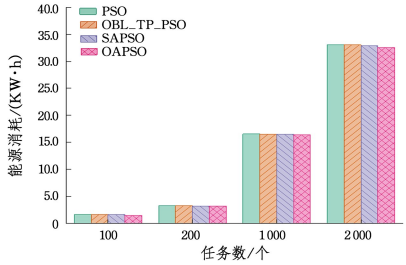

图 4 能源消耗对比

Fig. 4 Comparison of energy consumption

5.4 适应度值对比

表 7 列出了 4 种算法用于云数据中心任务调度时的适应度值，算法的适应度值随着任务数的增加而降低，任务数为 100 时，算法的适应度值最大。从图 5 可以看出，PSO 算法的适应度值最小，用户的成本最高，此时 OAPSO 算法的适应度值相对较大，达到了优化的目的。本文提出的算法最终的适应度最大，寻优效果最好，具有较强的全局搜索能力。

表 7 适应度值对比

Table 7 Comparison of fitness values

任务规模/个	PSO	OBL_TP_PSO	SAPSO	OAPSO
100	1.425	1.447	1.563	1.801
200	0.642	0.648	0.656	0.666
1 000	0.300	0.300	0.302	0.303
2 000	0.248	0.248	0.249	0.249

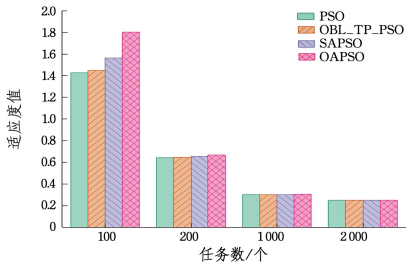

图 5 适应度值对比

Fig. 5 Comparison of fitness values

5.5 负载均衡因子对比

表 8 列出了 4 种算法用于云数据中心任务调度时的负载均衡因子。从图 6 中可以看出，OBL_TP_PSO 和 SAPSO 算法的负载均衡因子变化较大，相比其他几种算法，OAPSO 算法的负载均衡因子在任务量变化时波动较小，达到了优化的目的。

表 8 负载均衡因子对比

Table 8 Comparison of load balancing factors

任务规模/个	PSO	OBL_TP_PSO	SAPSO	OAPSO
100	0.223	0.041	0.010	0.104
200	0.312	0.202	0.106	0.200
1 000	0.347	0.345	0.339	0.263
2 000	0.353	0.357	0.385	0.261

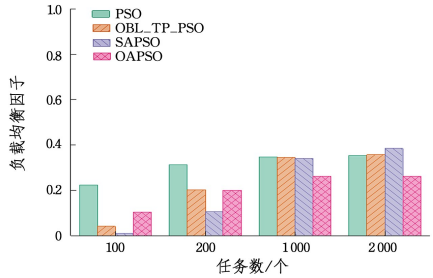

图 6 负载均衡因子对比

Fig. 6 Comparison of load balancing factors

5.6 算法性能对比

算法的收敛速度和精度是评价算法性能的两个重要指标。在所述硬件环境的基础上，对改进后的算法进行测评。云任务数分别设置为 100，200，1 000，2 000 时，不同算法的适应度值随着迭代次数的收敛性能如图 7－图 10 所示。

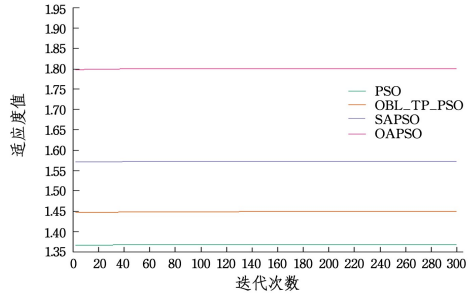

图 7 云任务数为 100 时，算法收敛曲线对比

Fig. 7 Comparison of algorithm convergence curves when the number of cloud tasks is 100

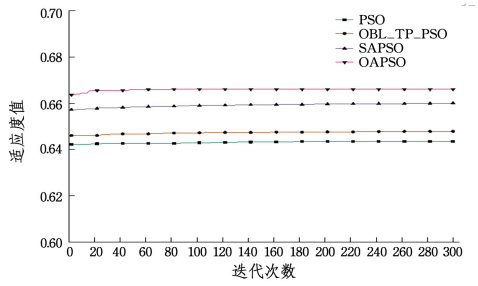

图 8 云任务数为 200 时，算法收敛曲线对比

Fig. 8 Comparison of algorithm convergence curves when the number of cloud tasks is 200

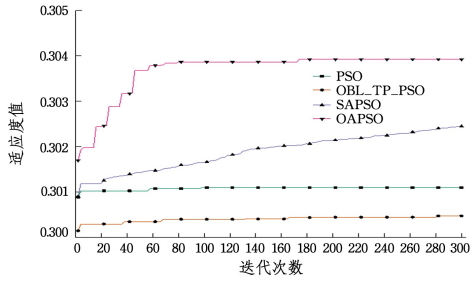

图 9 云任务数为 1 000 时，算法收敛曲线对比

Fig. 9 Comparison of algorithm convergence curves when the number of cloud tasks is 1 000

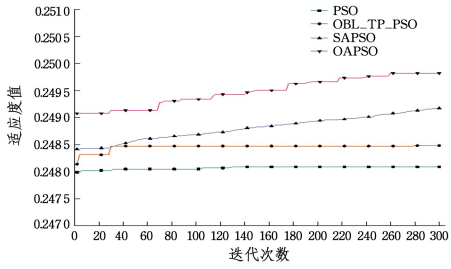

图 10 云任务数为 2000 时，算法收敛曲线对比

Fig. 10 Comparison of algorithm convergence curves when the number of cloud tasks is 2000

从图 7—图 10 可以看出，相比其他几种算法，OAPSO 算法的收敛效果最好。传统 PSO 算法收敛速度快，但容易陷入局部最优且收敛精度较低。SAPSO 在传统 PSO 的基础上可以通过控制惯性权重的大小来跳出局部最优解，提高了解的精度。OBL_TP_PSO 算法提高了初始种群的质量，引入试探感知的策略，降低了陷入局部最优的概率。本文算法在收敛效果方面表现较好的原因主要得益于初始种群质量的提高，且引入了混沌变异策略来自适应惯性权重，平衡了全局搜索与局部搜索的能力。

结束语 由于收敛速度快且易于实现，粒子群任务调度方法被认为是一种更适用于任务负载平衡调度的方法。针对 PSO 算法早熟且收敛精度低的缺陷，提出改进后的粒子群算法 OAPSO。仿真实验结果表明，所提算法比标准 PSO 算法收敛精度更高，收敛速度更快，节能效率更高，调度策略上要优于 SAPSO 算法和 OBL_TP_PSO 算法。因此，所提任务调度策略能够较好地解决云数据中心任务调度分配问题。然而，本算法依然有待改进，在解决任务调度问题时忽略了虚拟机部署的负载均衡问题，这些问题将在以后的工作中进一步研究。

参 考 文 献

[1] MA X S, TAN J, CHEN S Y et al. Research on optimal particle swarm optimization for multi-objective task scheduling in cloud computing[J]. Journal of Electronic Measurement and Instrumentation, 2020, 34(8): 133-143.

[2] WANG G Y, WANG Q S, ZHAO T. The improved strategy of shuffled frog leaping algorithm in the resource scheduling of cloud computing [J]. Science Technology and Engineering, 2018, 18(4): 297-303.

[3] LI J, WANG X, GUO L, et al. Innovative Data-Centre Cooling Technologies in China-Liquid Cooling Solution[R]. China. Copenhagen Centre on Energy Efficiency, 2020.

[4] TAO J. Research on sdn-based dynamic management of energy consumption of cloud data center[C]// 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference(IMCEC). IEEE, 2021: 1266-1270.

[5] ZHANG C. Research on energy consumption optimization scheduling in the cloud data center[D]. Dalian: Dalian University of Technology, 2021.

[6] ZHOU Z, SHOJAFAR M, ABAWAJY J, et al. IADE: An improved differential evolution algorithm to preserve sustainability in a 6G network[J]. IEEE Transactions on Green Communications and Networking, 2021, 5(4): 1747-1760.

[7] DAYARATHNA M, WEN Y, FAN R. Data center energy consumption modeling: A survey[J]. IEEE Communications Surveys & Tutorials, 2015, 18(1): 732-794.

[8] OMARA F A, ARAFA M M. Genetic algorithms for task scheduling problem[M]. Berlin: Springer, 2009: 479-507.

[9] ZHAO C, ZHANG S, LIU Q, et al. Independent tasks scheduling based on genetic algorithm in cloud computing[C]// 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing. IEEE, 2009: 1-4.

[10] CHEN H, ZHU J, ZHU X, et al. Resource-delay-aware scheduling for real-time tasks in clouds[J]. Journal of Computer Research and Development, 2017, 54(2): 446-456.

[11] DORIGO M, CARO G D. Ant colony optimization: A new metaheuristic[C]// Proceedings of 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE, 1999: 1470-1477.

[12] DORIGO M, STÜTZLE T. The ant colony optimization metaheuristic: Algorithms, applications, and advances[M]// Handbook of metaheuristics: International Series in Operations Research & Management Science. Berlin: Springer, 2003: 250-285.

[13] GLOVER F. Tabu search-part I[J]. ORSA Journal on Computing, 1989, 1(3): 190-206.

[14] SHROFF P, WATSON D W, FLANN N S, et al. Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments[C]// 5th Heterogeneous Computing Workshop(HCW'96). 1996: 98-117.

[15] GAN G, HUANG T, GAO S. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment [C]// 2010 International Conference on Intelligent Computing and Integrated Systems. IEEE, 2010: 60-63.

[16] KOKILAVANI T, GEORGE AMALARETHINAM D I. Load balanced Min-Min algorithm for static Meta Task scheduling in grid computing[J]. International Journal of Computer Applications, 2011, 20(2): 42-48.

[17] HUNG T C, HIEU L N, HY P T, et al. MMSIA: improved max-Min scheduling algorithm for load balancing on cloud computing [C]// Proceedings of the 3rd International Conference on Machine Learning and Soft Computing. New York: Association for Computing Machinery, 2019: 60-64.

[18] EBERHART R, KENNEDY J. Particle swarm optimization [C]// Proceedings of the IEEE International Conference on Neural Networks. 1995: 1942-1948.

[19] ZHOU Z, LI F, ABAWAJY J H, et al. Improved PSO algorithm integrated with opposition-based learning and tentative perception in networked data centres[J]. IEEE Access, 2020, 8: 55872-55880.

[20] WANG X H, LI J J. Hybrid particle swarm optimization with simulated annealing [C]// Proceedings of 2004 International Conference on Machine Learning and Cybernetics. IEEE, 2004, 4: 2402-2405.

[21] PANDEY S, WU L, GURU S M, et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments[C]// 2010 24th IEEE Interna-

tional Conference on Advanced Information Networking and Applications. IEEE，2010：400-407.

[22] RITU G，KUMAR S A. Multi-objective workflow grid scheduling using e-fuzzy dominance sort based discrete particle swarm optimization[J]. Journal of Supercomputer，2014，68(2)：709-732.

[23] KAUR S，VERMA A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment[J]. International Journal of Information Technology and Computer Science(IJITCS)，2012，4(10)：74-79.

[24] LI H H，FU Y W，ZHAN Z H，et al. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling[C]// 2015 IEEE Congress on Evolutionary Computation(CEC). IEEE，2015：870-876.

[25] WANG J L，GONG B，LIU H，et al. Green heterogeneous scheduling algorithm through deep integration of hardware and software energy saving principles [J]. Journal of Software，2021，32(12)：3768-3781.

[26] LIANG B，DONG X，WANG Y，et al. A low-power task scheduling algorithm for heterogeneous cloud computing[J]. Journal of Supercomputing，2020，76(9)：7290-7314.

[27] LI H H，FU Y W，ZHAN Z H，et al. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling[C]// 2015 IEEE Congress on Evolutionary Computation(CEC). IEEE，2015：870-876.

[28] DING Z Q，LIU C，XIONG T G. Virtual machine placement based on load balancing and power consumption[J]. Computer & Digital Engineering，2015，43(11)：1962-1967.

[29] ZHANG C Y，FU X，QIAO L. Virtual machine placement based on multi-objective optimization in cloud computing environment [J]. Computer Applications and Software，2021，38(3)：32-38.

[30] YUAN J Z，LIU Y T. Interrelation strategy for virtual machine selection and placement [J]. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition)，2021，33(1)：163-170.

[31] FAN X B，WOLF-DIETRICH W，LUIZ A B. Power provisioning for a warehouse-sized computer[J]. ACM SIGARCH Computer Architecture News，2007，35(2)：13-23.

[32] AGARWAL M，SRIVASTAVA G M S. Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing[J]. Journal of Ambient Intelligence and Humanized Computing，2021，12(10)：9855-9875.

[33] GONG B J，FENG Q Q，ZHAO X F. Virtual machine placement algorithm optimizing energy-efficiency of cloud data center[J]. Computer Engineering and Design，2018，39(2)：527-531.

[34] LIU J. energy-saving algorithm research ofcomputing resources in data center[D]. Chengdu：University of Electronic Science and Technology of China，2013.

[35] SUN M. Research on Energy Consumption Optimization Strategy for Green Cloud Computing[D]. Nanjing：Nanjing University of Posts and Telecommunications，2017.

[36] BRATTON D，KENNEDY J. Defining a standard for particle swarm optimization[C]// 2007 IEEE Swarm Intelligence Symposium. IEEE，2007：120-127.

[37] POLI R，KENNEDY J，BLACKWELL T. Particle swarm optimization[J]. Swarm Intelligence，2007，1(1)：33-57.

[38] KENNEDY J，EBERHART R C. Particle Swarm Optimization [C]// Proceedings of the IEEE International Conference on Neural Networks. Piscataway：IEEE Press，1995：1942-1948.

[39] ZHANG J Q，XU S W，LI X C et al. Cloud computing task scheduling based on orthogonal adaptive whale optimization [J]. Journal of Computer Applications，2022，42(5)：1516-1523.

[40] KUMAR M，SHARMA S C. PSO-COGENT：Cost and energy efficient scheduling in cloud environment with deadline constraint[J]. Sustainable Computing：Informatics and Systems，2018，19：147-164.

[41] XIN F J. Research of Multi-objective Task Scheduling based on Chaos Cat Swarm Optimization in Cloud Computing[D]. Handan：Hebei University of Engineering，2019.

[42] GUO F. Research of Multi-objective Task Scheduling based on Fireworks Algorithm in Cloud Computing[D] Handan：Hebei University of Engineering，2018.

[43] RAHNAMAYAN S，TIZHOOSH H R，SALAMA M M A. Opposition-based Differential Evolution[J]. IEEE Trans. on Evolutionary Computation，2008，12(1)：64-79.

[44] YU W W，XIE C W. Hybrid Particle Swarm Optimization with Multiply Strategies [J]. Computer Science，2018，45(S1)：120-123.

[45] WANG J，QIN J T. Improved seagull optimization algorithm based on chaotic map and t-distributed mutation strategy [J]. Application Research of Computers，2022，39(1)：170-176，182.

[46] SUN C Y，WANG X W. Multi-objective task scheduling of cloud computing based on MGA-PSO [J]. Computer Applications and Software，2021，38(6)：212-218.

[47] BI X J，HU S Y. Firefly algorithm with high precision mixed strategy optimized particle filter[J]. Journal of Shanghai Jiaotong University，2019，53(2)：232-238.

[48] CALHEIROS R N，RANJAN R，BELOGLAZOV A，et al. CloudSim：a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software，Practice and Experience，2011，41(1)：23-50.

LIU Chenwei，born in 1995，postgraduate，is a member of China Computer Federation. His main research interests include cloud computing and task scheduling.

SUN Jian，born in 1982，Ph.D，lecturer，master supervisor，is a member of China Computer Federation. His main research interests include big data storage and big data management.

（责任编辑：何杨）