2025 年 第 25 卷 第 12 期
2025, 25(12): 05045-13

科 学 技 术 与 工 程
Science Technology and Engineering

ISSN 1671—1815
CN 11—4688/T

DOI:10.12404/j.issn.1671-1815.2403824
引用格式:陈攀, 孙鉴, 吴佳伟, 等. 基于改进粒子群的云计算任务调度算法[J]. 科学技术与工程, 2025, 25(12): 5045-5057.
Chen Pan, Sun Jian, Wu Zhuiwei, et al. Cloud computing task scheduling algorithm based on improved particle swarm optimization[J]. Science Technology and Engineering, 2025, 25(12): 5045-5057.

自动化技术、计算机技术

基于改进粒子群的云计算任务调度算法

陈攀[1], 孙鉴[1,2*], 吴佳伟[1], 武涛[1], 杨晓焕[1], 马宝全[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)

摘 要 传统粒子群算法(particle swarm optimization,PSO)在云计算任务调度的性能和效率方面仍然存在局部搜索效率较低、搜索精度有限等不足,导致难以找到全局最优解并容易陷入局部最优解,针对此问题提出一种改进的粒子群任务调度算法(improved particle swarm optimization,IPSO)。首先,通过反向学习策略生成分布更加均匀的初始种群,提高算法的收敛速度。其次,在粒子更新过程中引入正弦余弦算法(sine cosine algorithm,SCA)以此提高粒子的寻优能力,平衡全局搜索和局部开发两个过程。最后,添加了基于平均适应度的搜索行为进一步扩大搜索解空间以找到更好的最优解,防止陷入局部最优。在 CloudSim 仿真平台上进行实验验证。实验结果表明:改进粒子群算法在降低系统任务的成本和最大完工时间上均有着显著的优势。特别是当任务数量达到 500 时,IPSO 在总成本上相较于自适应粒子群算法(adaptive particle swarm optimization,AdPSO)、正弦余弦粒子群算法(sine cosine algorithm-particle swarm optimization,SCA-PSO)、模拟退火粒子群算法(simulated annealing particle swarm optimization,SAPSO)、增强型吞噬遗传算法(enhanced phagocytosis genetic algorithm,EPGA)、竞争交叉机制遗传算法(competitive crossover mechanism genetic algorithm,C2PGA)、反向学习粒子群算法(opposition based learning-particle swarm optimization,OBL-PSO)和 PSO 分别提升了 10%、4.6%、8.6%、9.2%、8.2%、10.4%和 11.3%,在最大完工时间上分别提升了 34.1%、27%、41.7%、28.5%、21.6%、50.3%和 54.8%,验证了 IPSO 在不同任务规模下解决云计算任务调度问题的可行性和有效性。

关键词 云计算; 任务调度; 粒子群算法(PSO); 正弦余弦算法(SCA); CloudSim
中图法分类号 TP391; **文献标志码** A

Cloud Computing Task Scheduling Algorithm Based on Improved Particle Swarm Optimization

CHEN Pan[1], SUN Jian[1,2*], WU Zhui-wei[1], WU Tao[1], YANG Xiao-huan[1], MA Bao-quan[1]
(1. School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China;
2. The Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission, North Minzu University, Yinchuan 750021, China)

[**Abstract**] The traditional particle swarm optimization (PSO) algorithm still has shortcomings in terms of performance and efficiency of cloud computing task scheduling, such as low local search efficiency and limited search accuracy, which often makes it difficult to find the global optimal solution and easily falls into the local optimal solution. To solve this problem, an improved particle swarm optimization task scheduling algorithm(IPSO) was proposed. Firstly, a opposition-based learning strategy was used to create a more homogeneous initial population and the Rate of convergence of this algorithm was enhanced. Secondly, in the particle update process, the sine cosine algorithm(SCA) was introduced to enhance the optimization ability of the particles and balance the two processes of global search and local development. Finally, a search behavior based on average fitness was added to further expand the search solution space to find better optimal solutions and prevent falling into local optima. Experimental verification was conducted on the CloudSim simulation platform. The experimental results show that the improved particle swarm algorithm has significant advantages in reducing the cost and maximum completion time of system tasks. In particular, when the number of tasks reaches 500, IPSO improves the total cost by 10%, 4.6%, 8.6%, 9.2%, 8.2%, 10.4% and 11.3% respectively compared with adaptive particle swarm optimization (AdPSO), sine cosine algorithm-particle swarm optimization (SCA-PSO), simulated annealing particle swarm

收稿日期: 2024-05-23; **修订日期**: 2025-01-30
基金项目: 国家自然科学基金(62062002);宁夏自然科学基金(2024AAC03192);北方民族大学中央高校基本科研业务费专项(FWNX09);北方民族大学校级一般项目(2021XYZJK01)
第一作者: 陈攀(1996—),男,汉族,湖南湘潭人,硕士研究生。研究方向:大数据分析与知识工程。E-mail:934926124@qq.com。
*通信作者: 孙鉴(1982—),男,汉族,山东烟台人,博士,副教授。研究方向:任务调度、大数据存储与管理。E-mail:2014132@nun.edu.cn。

optimization (SAPSO), enhanced phagocytosis genetic algorithm (EPGA), competitive crossover mechanism genetic algorithm (C2PGA), opposition based learning-particle swarm optimization (OBL-PSO) and PSO, and improves the maximum completion time by 34.1%, 27%, 41.7%, 28.5%, 21.6%, 50.3% and 54.8% respectively, which verifies the feasibility and effectiveness of IPSO in solving cloud computing task scheduling problems under different task scales.

[**Keywords**] cloud computing; task scheduling; particle swarm optimization(PSO); sine cosine algorithm(SCA); CloudSim

随着智能设备和5G等无线通信技术在万物互联时代的广泛应用和发展,通信网络已经不再是由单一网络构成,而是将多种不同的网络属性融合到一个通信系统中的异构网络系统[1]。在这个数字时代,每天都会产生海量的信息,数据的承载形式多样,使得数据处理和存储的规模空前。云计算作为一种分布式虚拟化资源,以其强大的计算能力、存储能力和便利性,成为互联网中处理数据的常用解决方案。云计算的核心思想是通过异构技术整合多个闲置的物理计算机资源(如计算资源CPU、存储资源DRAM、网络资源I/O等)并由云进行管理。用户在使用云资源时无需考虑云平台底层细节,也无需购买软硬件等基础设施,只需根据购买需求付费即可[2]。在云计算系统中,任务调度是云环境的主要组成部分,也是需要优化的最具挑战性的问题,被称为NP-hard问题。策略的合理性直接关系到调度的效率和云计算系统的整体性能,是一种组合优化问题[3]。合理的调度策略将减少任务完成时间和成本,因此,系统利用率、可靠性和用户满意度都会提高,用户体验也会得到改善。然而,云系统的复杂性和用户多样性增加了在云环境下任务调度研究的难度[4]。

云计算任务调度方案分为三类:传统调度、启发式调度和元启发式调度[5]。传统的调度算法有:Min-Min、Max-Min和Sufferage算法[6]等。启发式调度算法有:先来先服务算法(first-come, first-served, FCFS)、最短作业优先算法(shortest job first,SJF)和最小松弛度优先算法(least slack time first, LSTF)等。元启发式调度算法涵盖:粒子群优化算法(particle swarm optimization,PSO)、模拟退火算法(simulated annealing, SA)、人工鱼群算法(artificial fish swarm algorithm, AFSA)和遗传算法(genetic algorithm,GA)等。由于调度问题属于云计算中的NP难问题,因此在多项式时间内无法找到最优解。若采用枚举搜索来解决这类问题,则生成调度解决方案所需的时间可能会非常长。因此,采用基于进化和群体智能的元启发式算法可以获得最优的调度解决方案[7]。这些算法能够在复杂空间并行搜索潜在解,提供多种解,并解决多目标优化问题[8],因此被广泛应用于解决云计算中的任务调度问题,如蚁群算法[9]、人工蜂群算法[10]和粒子群算法[11-12]。

遗传算法最早由John[13]于1975年提出,借鉴了自然界生物进化的规律,将达尔文“适者生存”的进化论应用到实践中。该算法通过选择、交叉和变异进行迭代,以产生最佳解决方案[6]。然而遗传算法仍然存在早熟现象,即当种群进化到算法中后期时,种群的多样性已经被破坏,算法的搜索陷入停滞,导致算法陷入局部最优解,从而无法得到全局最优解[14]。模拟退火算法[15]的核心思想是种串行结构的贪心策略,每次以一定的概率跳出当前的局部最优解,并在温度足够低时停止迭代。由于概率和温度难以控制,迭代速度也受到影响,因此模拟退火算法存在着陷入局部最优解的风险,并且寻找全局最优解的能力不足[14]。人工蜂群算法[16](artificial bee colony algorithm,ABC)模拟蜜蜂串行或并行采集蜂蜜的行为。蜜蜂之间分工不同,领导蜂和跟随蜂负责采蜜并加速结果的收敛,而侦查蜂则提供保障,防止算法陷入局部最优。然而,种群大小、迭代次数和侦查蜂能力等参数的选择完全依赖于经验,这可能会影响算法的收敛速度,导致陷入局部最优解或者无法收敛到合理的解[14]。因此,单一的元启发式算法容易出现早熟、收敛速度慢等问题,导致算法只能获得局部最优解,而无法达到全局最优解,这会影响算法的全局收敛性和计算能力。为了获得更令人满意的结果,可以将两种或多种元启发式算法结合起来,构建一种混合优化算法,各算法在不同方面展现出各自的优势,这样的组合使用可以显著提高搜索效率和寻优能力。

Agarwal等[17]提出一种粒子群优化遗传算法(genetic algorithm enabled particle swarm optimization, PSOGA),尽管GA在寻找精确解方面可能会遇到困难,但在探索全局区域方面却具有优势,而PSO的群体交互则有助于搜索最优解。PSO负责搜索解空间,而GA则通过本地搜索优化已有的解。Senthil等[18]介绍了一种混合遗传蚁群优化算法(hybrid genetic-ant colony optimization algorithm,HGA-ACO),GA的主要优势在于它能够有效地解决任务调度问题,并且可以与现有的模拟和模型轻松连接。GA搜索庞大的解空间,并且不要求特定问题的解具有固定长度。由于蚁群算法(ant colony optimization,ACO)具有可扩展性和并发性,因此非常适合用于云计算任务调度。在ACO内存中,存储了遍历路径,因此在

每次迭代结束时,都可以获得蚂蚁的路径。ACO 优化帮助 GA 获得全局最优解,GA 帮助 ACO 提供初始信息素。实验结果表明,以上混合元启发式算法均提高了任务分配的性能,并减少了最大完工时间。因此将混合优化算法应用于解决云计算中的任务调度问题是非常有前途且可靠的。

Agarwal 等[19]结合了反向学习(opposition based learning,OBL)与 PSO,旨在克服 PSO 的过早陷入局部最优和收敛速度缓慢等问题。OBL 技术的主要优势在于将当前解决方案在搜索空间中转换为新的或全局搜索空间,以维持解决方案的多样性。Senthil 等[20]提出一种将 PSO 与灰狼算法(grey wolf optimization,GWO)相融合的算法,GWO 具有较快的收敛速度和更好的寻找全局最优解能力,而 PSO 则具备较强的探索性,两者相结合可提高算法的响应速度和完成时间。汪婷等[21]提出一种多策略融合的 PSO 算法。该算法引入模拟退火算法(simulate anneal arithmetic,SA)动态更新惯性权重,以避免陷入局部最优。通过饥饿游戏搜索(hungary games search,HGS)加快算法的收敛速度并提高结果精度。最后,采用双重变异限制策略(double restrictions)来提高算法的寻优效率。

与其他元启发式算法相比,PSO 具有简单易实现、收敛速度快、需要调整的参数少等优点。此外,收敛速度快、解决单目标问题效率高等优良特性,促使研究者尝试将 PSO 应用到多目标优化领域[22]。基于此,针对混合元启发式算法解决云计算中的任务调度问题,使得任务分配的最大完工时间短、成本低,采用正弦余弦算法(sine cosine algorithm,SCA)[23]中正弦和余弦函数的可变性和周期性作为寻找最优解和实施迭代运算的设计目标。在 PSO 中通过利用正弦和余弦的周期性振荡生成迭代方程,该方程实现两个线程功能:全局搜索和局部开发,通过该方程对解集进行扰动和更新,提出一种改进的粒子群算法(improved particle swarm optimization,IPSO),并在 CloudSim 仿真平台上实现模拟任务调度的实验,验证算法的有效性。算法的主要改进如下:将反向学习策略加入粒子群的种群初始化,提升初始种群的质量,扩大粒子群的搜索解空间;改进了粒子的搜索行为,通过 SCA 中的正余弦函数的周期性和波动性,以更好地协调全局搜索和局部开发两个部分,提高粒子的寻优能力;引入平均适应度值,进一步扩大搜索解空间,以找到更好的最优解,避免算法陷入局部最优。通过优化粒子群算法,可以更精确地搜索到全局最优解,从而有效地提升云计算环境中任务调度的效率,降低资源浪费。

1　云计算任务调度模型

1.1　任务调度描述

在云计算环境下,任务调度涉及有效地分配和管理用户提交的任务,以优化资源利用率、提高系统性能,并满足用户的性能和服务质量要求。任务调度在云计算中扮演着关键的角色,云计算平台通常包含大量网络、计算和存储资源,而任务调度的合理性能直接影响到资源的利用效率和用户体验。云计算任务调度的目标是通过智能地分配和管理资源,实现高效、可靠、弹性的任务执行,满足用户的需求,同时提高整个云计算系统的利用效率[24]。云计算任务调度模型如图 1 所示。

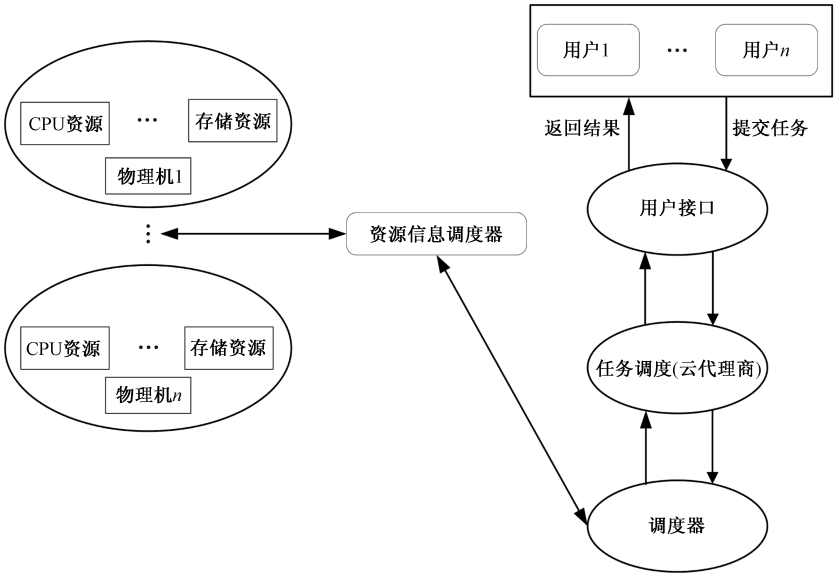


图 1　云计算任务调度模型

Fig. 1　Cloud computing task scheduling model

该模型的任务集合为

$$T = \{t_1, t_2, \cdots, t_n\} \tag{1}$$

式(1)中:n 为任务数;t_i 为第 i 个任务。

云环境下的虚拟机集合为

$$V_{\mathrm{m}} = \{v_{\mathrm{m}_1}, v_{\mathrm{m}_2}, \cdots, v_{\mathrm{m}_m}\} \tag{2}$$

式(2)中:m 为虚拟机的个数;v_{m_j} 为第 j 台虚拟机。

将 n 个任务分配给 m 个虚拟机分配矩阵,可表示为

$$\boldsymbol{E} = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1m} \\ E_{21} & E_{22} & \cdots & E_{2m} \\ \vdots & \vdots & & \vdots \\ E_{n1} & E_{n2} & \cdots & E_{nm} \end{bmatrix} \tag{3}$$

式(3)中:E_{ij} 为是否将第 i 个任务分配给第 j 个虚拟机,当 $E_{ij}=0$ 时,任务 i 没有分配给虚拟机 j;当 $E_{ij}=1$ 时,任务 i 分配给虚拟机 j。

1.2 问题求解

适应度函数是优化算法中的关键组成部分,是在算法过程中用于评估个体(解决方案)的优劣程度,从而影响算法的收敛和最终的优化结果。为了降低在云环境下任务调度的执行时间和成本,在设计适应度函数时主要考虑最大完工时间和总成本,具体设计如下所述。

(1)任务最大完工时间(M_{S})。任务最大完工时间定义为用户通过用户接口提交任务在云环境下执行所耗费的总时间,即最后一个任务执行完成所花费的总时间。任务最大完工时间越小,则该算法的任务调度能力越优。

任务的执行时间可表示为

$$t_{ij} = \frac{T_{\mathrm{len}}(i)}{V_{\mathrm{mm}}(j)} \tag{4}$$

式(4)中:$T_{\mathrm{len}}(i)$ 为虚拟机所分配任务的大小;$V_{\mathrm{mm}}(j)$ 为虚拟机的计算性能。

则每个任务分配到各个虚拟机的执行时间矩阵 $\boldsymbol{E}_{\mathrm{T}}$ 可表示为

$$\boldsymbol{E}_{\mathrm{T}} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{bmatrix} \tag{5}$$

定义 V_{mT_j} 为虚拟机 j 上执行所分配任务的执行时间总和,可表示为

$$V_{\mathrm{mT}_j} = \sum_{i=1}^{n} E_{ij} t_{ij} \tag{6}$$

虚拟机执行任务的最大完工时间 M_{S} 计算公式为

$$M_{\mathrm{S}} = \mathrm{Max}(V_{\mathrm{mT}_1}, V_{\mathrm{mT}_2}, \cdots, V_{\mathrm{mT}_m}) \tag{7}$$

(2)任务执行总成本(C_{ost})。任务执行总成本定义为在云环境下计算完成用户提交任务所耗费的成本开销,是用户使用云服务配置时所要支付的实际费用。任务执行总成本越低,说明所使用的调度算法提供了更优的解决方案。

定义 C_j 为第 j 个虚拟机执行所分配任务的总成本,计算公式为

$$C_j = V_{\mathrm{mT}_j} P_j \tag{8}$$

式(8)中:P_j 为虚拟机 j 在单位时间内执行所分配任务花费的成本价格。

则虚拟机执行完所分配任务所耗费的总成本 C_{ost} 可表示为

$$C_{\mathrm{ost}} = \sum_{j=1}^{n} C_j \tag{9}$$

(3)多目标适应度函数。本文目标在于降低任务最大完工时间和减少执行总成本。在求解过程中,适应度值作为粒子更新位置的评判标准,帮助算法在任务调度过程中找到最优解。

本文多目标适应度函数可表示为

$$F_{\mathrm{it}} = \ln M_{\mathrm{S}} + \ln C_{\mathrm{ost}} \tag{10}$$

当 F_{it} 越小时,表示算法执行任务调度的效果越好。

2 任务调度算法介绍

2.1 传统粒子群优化算法

受到鸟群和鱼群觅食行为研究的启发,Kennedy 等[25]提出了粒子群算法,模拟了鸟群在飞行中寻找食物的过程,通过群体间的协作,鸟群能够找到最优的群体目标。其基本思想是,在解决问题的解空间中,整个种群通过种群中个体间的信息交流,从无序向有序演化,从而获得解决问题的可行方案。设粒子个体的位置为向量 $\boldsymbol{X} = \{x_1, x_2, \cdots, x_n\}$,粒子所处位置的适应度 $Y = f(\boldsymbol{X})$,粒子群优化算法描述如下。

2.1.1 位置更新

每个粒子通过更新其位置和速度来模拟搜索过程。位置的更新是根据当前位置、速度和个体与全局最优位置的差异来计算的。这个过程使得粒子在搜索空间中向着更有希望的方向移动。设粒子 i 当前一代位置为 x_i^k,粒子 i 下一代更新后的位置为 x_i^{k+1},粒子 i 下一代的速度为 v_i^{k+1},则粒子根据式(11)进行位置更新。

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{11}$$

2.1.2 速度更新

速度的更新考虑了粒子的历史速度、个体最优和全局最优的影响。这个过程调整粒子的移动速度,以使其更好地探索搜索空间。设粒子 i 当前一

代速度为 v_i^k,粒子 i 当前一代位置为 x_i^k,则粒子 i 下一代更新后的速度 v_i^{k+1},可分别表示为

$$v_i^{k+1} = F_1 + F_2 + F_3 \tag{12}$$

$$F_1 = \omega v_i^k \tag{13}$$

$$F_2 = c_1 r_1 (p_{b_i} - x_i^k) \tag{14}$$

$$F_3 = c_2 r_2 (g_b - x_i^k) \tag{15}$$

式中:ω 为用于调整解空间中搜索区域的非负惯性权重;c_1、c_2 为调整粒子最大学习步幅的加速度常数;r_1、r_2 为在[0,1]范围内生成的两个随机数,用于增加粒子搜索解空间的随机性;p_{b_i} 为粒子 i 个体的历史最佳位置;g_b 为粒子种群的全局最佳位置;F_1 为单个粒子的运动惯性(简称运动惯性),即粒子在当前迭代中保持上一次速度的一部分,以保持前进方向的连续性;F_2 为粒子自身思维的“自觉”部分(简称自我认知),可以理解为粒子受到自身历史最佳位置的引导,希望朝着个体历史最佳位置移动;F_3 为粒子个体间的信息交流与合作(简称社会经验),表示为粒子受到整个群体历史最佳位置的引导,希望朝着群体历史最佳位置移动。

2.2　正弦余弦算法

正弦余弦算法(sine-cosine algorithm,SCA)是由 Mirjalili[23] 于 2016 年提出的群智能优化算法,其灵感来自数学中正弦和余弦函数的周期性和振荡性,该算法将这些特性作为设计目标,并采用搜索和迭代算子来寻找和迭代最优解。相较于其他群智能优化算法,如 GA、PSO 等,正余弦算法的特点是参数数量更少、结构更简单、实现更容易、收敛速度更快,从而在实际应用中具有更好的性能。正余弦算法将迭代策略归纳为两个线程:全局搜索和局部开发。在全局搜索中,通过施加较大的随机波动,使当前解集中的解受到显著影响,以便探索解空间中的未知区域,这有助于在全局范围内寻找可能得最优解;在局部开发中,对解集进行微弱的随机扰动,以实现对当前解的局部搜索,这样的随机扰动有助于在解的邻域进行更深入的探索,提高算法对局部最优解的发现能力。然后利用目标适应度函数对所求解的适应度进行迭代评估,并根据指定的更新策略对解集进行随机迭代,以获取最优解。

设正余弦算法中的个体位置的向量为 $\boldsymbol{X}_i = \{x_{i1}, x_{i2}, \cdots, x_{in}\}$,种群最优候选解的位置向量为 $\boldsymbol{P}_i = \{p_{i1}, p_{i2}, \cdots, p_{in}\}$,则个体解的更新如式(16)~式(18)所示。

$$\boldsymbol{X}_i^{t+1} = \begin{cases} \boldsymbol{U}_1, & r_4 > 0.5 \\ \boldsymbol{U}_2, & r_4 \leqslant 0.5 \end{cases} \tag{16}$$

$$\boldsymbol{U}_1 = \boldsymbol{X}_i^t + r_1 \sin(r_2) \left| r_3 \boldsymbol{P}_i^t - \boldsymbol{X}_i^t \right| \tag{17}$$

$$\boldsymbol{U}_2 = \boldsymbol{X}_i^t + r_1 \cos(r_2) \left| r_3 \boldsymbol{P}_i^t - \boldsymbol{X}_i^t \right| \tag{18}$$

$$r_1 = a - \frac{a}{T} t \tag{19}$$

式中:$\boldsymbol{X}_i^t$ 和 $\boldsymbol{X}_i^{t+1}$ 分别为第 t 次迭代和第 $t+1$ 次迭代时个体 i 所处的位置;r_1 由式(19)所示的更新函数确定,如果 r_1 较小,算法偏向于局部开发算法;如果 r_1 较大,算法倾向于全局搜索;a 为常数,设置为 2;t 为当前迭代次数;T 为最大迭代次数;r_2 为属于[0,2]之间的随机数;r_3 为属于(0,∞)的随机数;r_4 为属于(0,1)的随机数;$\boldsymbol{P}_i^t$ 为第 t 次迭代的最优候选解的位置。

3　改进的粒子群算法

3.1　反向学习

反向学习(opposition-based learning,OBL)由 Tizhoosh[26] 于 2005 年提出,成功应用于多种问题,并在短时间内证明了自己是提高各种性能的有效概念[27-28]。其核心思想是通过引入相反的元素或策略来提高搜索的多样性和全局性。通常在搜索空间中引入对立元素,是算法更全面地探索潜在解的空间。定义公式如下。

设 $\boldsymbol{P} = (x_1, x_2, \cdots, x_n)$ 为一个 N 维向量,当 x_i 属于 $[a_i, b_i]$,并且 $i = 1, 2, \cdots, n$。则 $\boldsymbol{P}$ 的相反向量定义为:$\boldsymbol{P}' = (x_1', x_2', \cdots, x_n')$。

x_i' 可表示为

$$x_i' = a_i + b_i - x_i \tag{20}$$

式(20)中:a_i 和 b_i 分别为 x_i 取值的上下界。

3.2　基于反向学习的种群初始化策略

在 PSO 中种群初始化至关重要,一个好的初始种群可以显著提升算法的收敛速度、使粒子可以更好地探索潜在的解,并更好地找到全局最优解防止算法陷入局部最优解。为了促使整个种群更快地朝向全局最优解收敛,加速搜索过程,最终提高解的精度。引入反向学习策略,将其应用于种群初始化阶段。基于反向学习的种群初始化流程如算法 1 所示,其中 p_{osx} 为粒子群中粒子的总数量。

算法 1　反向学习初始化种群

输入:种群大小 N,位置向量维度 d_{im};
输出:种群位置向量集 $\boldsymbol{p}_{os}$;
①随机产生 N 个 d_{im} 维位置向量,命名为 $\boldsymbol{p}_{osx}$
②for i in $\boldsymbol{p}_{osx}$
根据式(20)对向量集 $\boldsymbol{p}_{osx}$ 的第 i 个向量 $\boldsymbol{p}_{osxi}$ 生成相应的反向学习向量 $\boldsymbol{p}_{osyi}$,并加入反向学习向量集 $\boldsymbol{p}_{osy}$ 中
③end for
④计算 $\boldsymbol{p}_{osx}$ 和 $\boldsymbol{p}_{osy}$ 的适应度值
⑤从 $\boldsymbol{p}_{osx}$ 和 $\boldsymbol{p}_{osy}$ 中挑选出 N 个适应度较好的向量,并将其加入初始种群向量集 $\boldsymbol{p}_{os}$ 中
⑥return $\boldsymbol{p}_{os}$

3.3 基于平均适应度值的选择策略

在某些情况下,粒子可能过早地收敛到局部最优,从而无法充分的探索全局搜索空间,导致搜索精度不高。为了更有效地协调全局搜索和局部搜索,以提高算法的搜索效能,提出一种基于平均适应度的粒子选择。

首先根据式(21)计算每次迭代后种群的平均适应度值,并与每个粒子个体当前位置的适应度值进行比较,以确定粒子的移动速度和方向。

$$F_{\text{avg}}^{k} = \frac{\sum_{i=1}^{N} f(X_i^k)}{N} \tag{21}$$

式(21)中:X_i^k 为粒子 i 第 k 次迭代时所处位置;$f(X_i^k)$ 为第 k 次迭代时粒子 i 的适应度;N 为粒子群中粒子数量;F_{avg}^{k} 为第 k 次迭代时粒子群的平均适应度。

3.4 改进惯性权重

惯性权重 ω 反映了粒子保持先前速度的能力,用于平衡粒子速度的惯性和对个体或全局最优解的吸引力。一个较小的权重值更利于局部搜索,使得粒子更容易集中在局部最优解附近[29-30]。而一个较大的惯性权重值则有利于全局搜索,能够帮助粒子跳出局部最优解,更好地探索整个解空间。

若粒子个体当前所处位置的适应度小于平均适应度值,则根据式(22)改变惯性权重值;若粒子个体当前所处位置的适应度大于平均适应度值,则根据式(23)改变惯性权重值。

$$\omega = 0.3 + 0.1\frac{t}{T} \tag{22}$$

$$\omega = 1 + 0.1\frac{t}{T} \tag{23}$$

式中:t 为当前迭代次数;T 为最大迭代次数。

3.5 改进粒子速度更新

在 PSO 中,每个粒子都有一个速度,它决定了粒子在搜索空间中的方向和速度。速度根据粒子的历史最优位置和整个群体中的全局最优位置的平均距离进行更新。通过调整粒子的速度和位置来找到最优解,这样粒子就能更好地探索潜在的解空间,改进的粒子速度更新如式(24)所示。

$$V_i^{k+1} = F_1' + F_2' + F_3' \tag{24}$$

$$F_1' = \omega V_i^k \tag{25}$$

$$F_2' = c_1 r_1 \left(\frac{p_{\text{b}_i} + g_{\text{b}}}{2} - X_i^k\right) \tag{26}$$

$$F_3' = c_2 r_2 \left(\frac{p_{\text{b}_i} + g_{\text{b}}}{2} - X_i^k\right) \tag{27}$$

3.6 改进粒子的位置更新

PSO 的位置更新机制通过调整粒子的位置,使得粒子在搜索空间中不断移动。这有助于在解空间中进行全局搜索和局部搜索,使得搜索更加有针对性。为了引导粒子进行更有效的搜索,从而找到最优解。提出一种基于平均适应度值的位置更新策略。

首先将粒子 i 当前所处位置的适应度值与种群平均适应度值进行比较。若粒子 i 的适应度值小于平均适应度值,则用式(22)、式(24)~式(27)和式(11)对粒子 i 进行位置更新。

若粒子 i 的适应度值大于平均适应度值,则要进行另一轮的比较。首先,通过式(23)、式(24)~式(27)和式(11)对粒子 i 的位置进行更新,并计算更新位置后的适应度值;然后通过式(16)~式(19)对更新位置前的粒子 i 进行位置更新,并计算更新后的适应度值;将两次粒子 i 更新后的适应度值再次进行比较;若前者小于后者,则保存前者更新后的位置;若后者小于前者,则保存后者更新后的位置。

设粒子第 k 次迭代所处位置为 X_i^k,计算粒子当前适应度值 $f(X_i^k)$,与平均适应度值 F_{avg}^{k} 进行比较,若 $f(X_i^k) < F_{\text{avg}}^{k}$,则根据式(22)改变 ω 的值,并根据式(24)~式(27)和式(11)对粒子 i 进行位置更新,更新后的位置为 $X_{i,\text{PSO}}^{k+1}$。

若 $f(X_i^k) \geqslant F_{\text{avg}}^{k}$,则根据式(23)修改 ω 的值,并根据式(24)~式(27)和式(11)更新粒子 i 的位置,更新后的位置为 $X_{i,\text{PSO}}^{k+1}$,并计算新位置的适应度值 $f(X_{i,\text{PSO}}^{k+1})$。然后对粒子 i 更新前的位置根据式(16)~式(19)进行位置更新,更新后的位置为 $X_{i,\text{SCA}}^{k+1}$,并计算新位置的适应度值 $f(X_{i,\text{SCA}}^{k+1})$,将两次更新后的适应度值进行比较,若 $f(X_{i,\text{PSO}}^{k+1}) < f(X_{i,\text{SCA}}^{k+1})$,则保留 $X_{i,\text{PSO}}^{k+1}$ 的位置更新,若 $f(X_{i,\text{PSO}}^{k+1}) \geqslant f(X_{i,\text{SCA}}^{k+1})$,则保留 $X_{i,\text{SCA}}^{k+1}$ 的位置更新。其中,$X_{i,\text{PSO}}^{k+1}$ 为使用改进的 PSO 来更新粒子 i 的位置,$X_{i,\text{SCA}}^{k+1}$ 为使用 SCA 来更新粒子 i 的位置。

算法 2 执行流程如下。其中 p 为粒子 i 使用 SCA 进行正弦扰动更新位置或余弦扰动更新位置的概率,取值为(0,1)的随机数。

3.7 基于 IPSO 的云计算任务调度

基于 IPSO 的云计算任务调度过程如图 2 所示。算法 3 流程如下。

算法 2　改进的粒子位置更新

输入:X_i^k;

输出:X_i^{k+1};

① if $f(X_i^k) < F_{avg}^k$

② $\omega = 0.3 + 0.1\frac{t}{T}$

③ 根据式(24)~式(27)和式(11)更新 X_i^k

④ $X_{i,PSO}^{k+1} = X_i^k + V_i^{k+1}$

⑤ renturn $X_{i,PSO}^{k+1}$

⑥ else

⑦ $\omega = 1 + 0.1\frac{t}{T}$

⑧ 根据式(24)~式(27)和式(11)更新 X_i^k

⑨ $X_{i,PSO}^{k+1} = X_i^k + V_i^{k+1}$

⑩ 计算 $f(X_{i,PSO}^{k+1})$

⑪ 根据式(16)~式(19)更新 X_i^k

⑫ $\boldsymbol{X}_{i,SCA}^{t+1} = \begin{cases} \boldsymbol{U}_1, & p > 0.5 \\ \boldsymbol{U}_2, & p \leqslant 0.5 \end{cases}$

⑬ 计算 $f(X_{i,SCA}^{k+1})$

⑭ if $f(X_{i,PSO}^{k+1}) < f(X_{i,SCA}^{k+1})$

⑮ return $X_{i,PSO}^{k+1}$

⑯ else

⑰ return $X_{i,SCA}^{k+1}$

⑱ return X_i^{k+1}

算法 3　改进的粒子群算法

输入:设置粒子群种群大小 N、最大迭代次数 M_{ax}、粒子位置向量维度 d_{im}、粒子速度边界 M_{speed}、粒子位置边界 M_{posi}。

输出:最优调度解 X_{gbest}。

① 对用户提交的任务根据虚拟机数量进行编码

② 根据算法 1 进行粒子群位置向量集初始化

③ 计算当前粒子个体的适应度值并记录历史最佳和全局最佳位置

④ while 不满足停止条件

⑤ 根据式(21)计算 F_{avg}^k

⑥ for i in N

⑦ 比较 $f(X_i^k)$ 和 F_{avg}^k 的大小

⑧ 根据算法 2 执行改进的位置更新

⑨ 更新粒子个体最佳适应度值、历史最佳和种群全局最佳位置

⑩ end for

⑪ 判断是否满足终止条件,若满足则返回最优解 X_{gbest},否则继续从步骤④进行下一次迭代

输入:设置粒子群种群大小 N、最大迭代次数 M_{ax}、粒子位置向量维度 d_{im}、粒子速度边界 M_{speed}、粒子位置边界 M_{posi}。

输出:最优调度解 X_{gbest}。

3.8　IPSO 时间复杂度分析

时间复杂度是一种用来衡量算法执行时间随输入规模增长而增长的量度。它通常用大 O 符号表示。时间复杂度描述了算法运行时间与输入规模之间的关系,以指导了解在不同输入情况下算法的性能表现。IPSO 的时间复杂度分析如表 1 所示。

表 1　IPSO 时间复杂度分析

Table 1　IPSO time complexity analysis

IPSO 算法步骤	时间复杂度	IPSO 算法步骤	时间复杂度
1. 初始化种群	$O(N)$	5. SCA 位置更新	$O(N)$
2. 评估并记录当前粒子的历史最优和全局最优解	$O(1)$	6. 计算适应度	$O(1)$
3. 计算种群的平均适应度值	$O(N)$	7. 更新历史最优和全局最优解	$O(N)$
4. PSO 速度和位置更新	$O(N)$	8. 输出最优解	$O(1)$

由表 1 可知,当种群规模为 N 时,可以求得 PSO 的时间复杂度为 $O[(2T+1)N]$。SCA 的时间复杂度也为 $O[(2T+1)N]$。对于 IPSO,假设最好情况下,当前粒子适应度都小于种群平均适应度,因此只需要对粒子进行改进的 PSO 算法更新,此时的时间复杂度为 $O[(3T+1)N]$;假设最坏情况下,当前粒子适应度都大于种群平均适应度,则每次都需要对改进的 PSO 和 SCA 算法更新后的适应度进行比较,此时的时间复杂度为 $O[(4T+1)N]$;故 IPSO 算法的时间复杂度介于 $O[(3T+1)N] \sim O[(4T+1)N]$。

由此可见,3 种算法的时间复杂度均为 $O(N)$ 级。策略虽然增加了 IPSO 的时间消耗,但并没有增加时间复杂度。

4　仿真实验与结果分析

4.1　实验环境设置

本实验采用澳大利亚墨尔本大学网格实验室开发的 CloudSim[31] 平台进行仿真模拟。实验环境配置为 AMD Ryzen 5 处理器、16 GB 内存、并运行在 Windows10 操作系统。在实验中,通过 CloudSim 模拟一个 IaaS 云提供商,其拥有一个数据中心、包含 5 台不同配置的虚拟机。云环境中规定使用 5 台虚拟机进行实验,每台虚拟机的详细配置信息如表 2 所示,所有对比算法都将在相同虚拟机和任务配置下运行,以确保实验的公平性。考虑资源的处理速度和待处理任务的长度并参考文献[32-33]中任务规模的划分,将实验中的任务数量按规模划分为 3 类,包括 50 个任务的小规模,200 个任务的中规模,500 个任务的大规模。每个任务大小都均在[100,1 000 000]范围内随机生成。由于 IPSO 和对比算法均属于随机搜索算法,每次搜索可能得到不同的解,为确保实验的准确性并减少由于任务大小的不确定性对结果的影响,每种算法均在同一测试环境下运行了 20 次,最终的实验结果取其平均值进行分析。表 3 展示了 6 种算法在性能方面的对比结果。

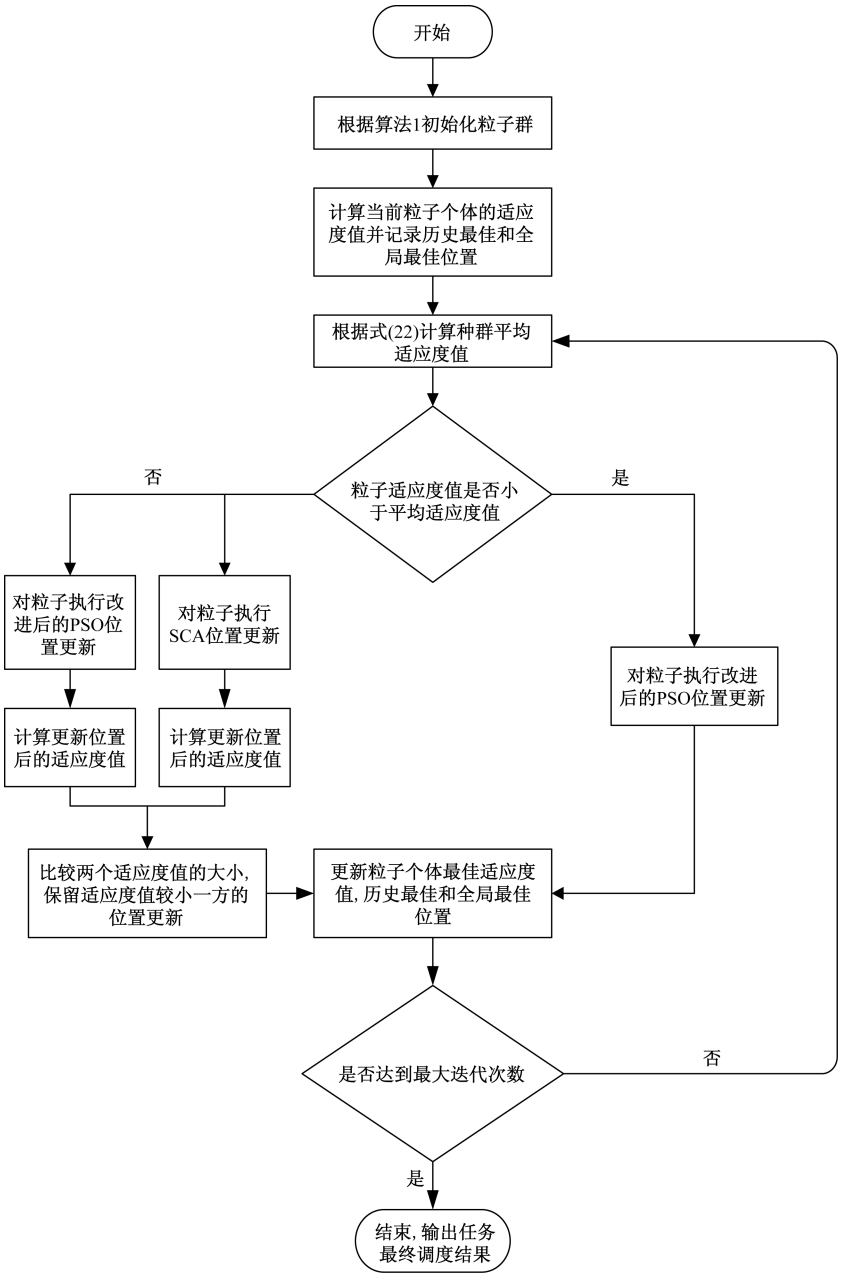


图 2 算法 IPSO 流程图

Fig. 2 Algorithm IPSO flowchart

表 2 虚拟机配置

Table 2 Virtual machine configuration

Id	MIPS/s	Ram/GB	Band Width/(GB·s^{-1})	价格/元
1	100	512	1	0.19
2	200	512	1	0.27
3	300	512	1	0.43
4	400	512	1	0.51
5	500	512	1	0.62

4.2 算法参数设置

为验证改进算法在云计算任务调度中的有效性,将提出的 IPSO 与 AdPSO[34]、SCA-PSO、SAPSO[35]、EPGA[36]、C2PGA[37]、OBL-PSO[38]及 PSO 这 7 种云计算任务调度算法进行对比。所有算法的迭代次数均设置为 200,种群大小为 100。IPSO、AdPSO、SCA-PSO、SAPSO、OBL-PSO 以及 PSO 的主要参数设置如表 4 所示,AdPSO、SAPSO、EPGA、C2PGA 和 OBL-PSO 其他参数分别参照文献[34-38]设置。

4.3 小规模任务

图 3 展示了在小规模任务情况下,8 种算法适应度迭代情况的对比结果。IPSO 的适应度值明显低于其他 7 种对比算法,这表明 IPSO 在云计算任务调度中表现出色。改进后的粒子群算法在迭代初期就取得优秀的初始值,显然优于其他对比算法,

表 3　IPSO 算法与其他算法性能比较

Table 3　Performance comparison of IPSO algorithm and other algorithms

任务规模	算法名称	执行总成本/元	最大完工时间/s	适应度函数值
小规模任务	IPSO	**8.16**	**14 793.78**	**11.70**
	AdPSO	9.41	17 063.51	11.99
	SCA-PSO	9.39	17 257.55	11.99
	SAPSO	9.24	16 632.76	11.94
	EPGA	9.66	17 750.14	12.05
	C2PGA	9.60	17 771.72	12.05
	OBL-PSO	9.50	17 106.84	12.00
	PSO	9.75	21 572.02	12.25
中规模任务	IPSO	**34.80**	**62 203.84**	**14.59**
	AdPSO	39.09	82 224.00	14.98
	SCA-PSO	37.02	70 633.12	14.77
	SAPSO	37.80	81 256.29	14.93
	EPGA	38.29	70 117.08	14.80
	C2PGA	37.89	69 689.03	14.79
	OBL-PSO	37.62	83 562.40	14.96
	PSO	39.28	97 960.72	15.16
大规模任务	IPSO	**89.02**	**158 952.78**	**16.47**
	AdPSO	98.82	241 287.30	16.98
	SCA-PSO	93.26	217 726.08	16.82
	SAPSO	97.40	272 866.40	17.09
	EPGA	98.07	204 233.44	16.81
	C2PGA	97.01	202 735.05	16.79
	OBL-PSO	99.32	319 761.53	17.27
	PSO	100.35	351 290.27	17.38

注:加粗数字表示在小、中、大 3 种规模任务情况下,所提出的 IPSO 算法相较于其他对比算法,在执行总成本、最大完工时间和适应度函数值 3 个指标上,性能均处于最高。

表 4　算法主要参数设置

Table 4　Algorithm main parameter settings

参数	数值
种群大小 N	100
d_{im}	50、200、500
M_{ax}	200
M_{speed}	[-3,3]
M_{posi}	[0,number of vm -1]

注:number of vm 表示 CloudSim 中虚拟机的数量。

这是因为良好的初始种群可以显著提高算法的收敛速度,使得粒子个体能够更好地探索潜在的解空间,并更有效地找到全局最优解,从而避免算法陷入局部最优解。显示出反向学习(OBL)初始化策略和改进粒子学习行为的优化效果,使得 IPSO 能够更快、更精准地找到全局最优解。

图 4 展示了在 50 任务规模情况下,不同算法在云环境下执行任务调度的最大完工时间和执行总成本方面的表现。结果显示,IPSO 在两个指标上均取得了最佳效果,其最大完工时间为 14 793.78 s,执行成本为 8.16 元。在总成本上 IPSO 相较于 AdPSO、SCA-PSO、SAPSO、EPGA、C2PGA、OBL-PSO

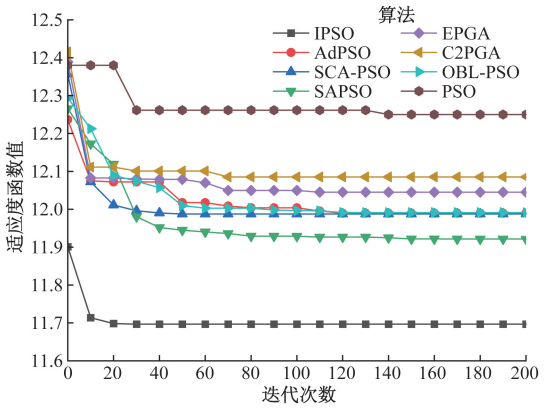
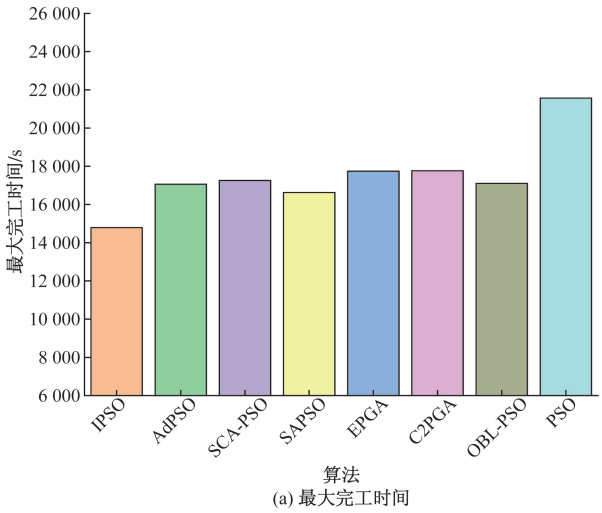
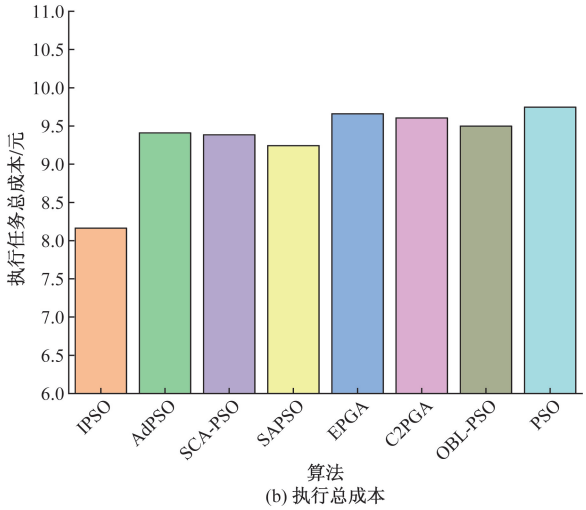


图 3　50 任务规模的收敛情况

Fig. 3　Convergence of 50 tasks



(a) 最大完工时间



(b) 执行总成本

图 4　50 任务规模的指标优化情况

Fig. 4　Indicator optimization of 50 tasks

和 PSO 分别降低了 13.3%、13.1%、11.7%、15.5%、15%、14.1% 和 16.3%,在最大完工时间上分别降低了 13.3%、14.3%、11.1%、16.7%、16.8%、13.5% 和 31.4%。这是因为基于平均适应度值的选择机制考虑了整个种群的平均适应度水

平,以帮助引导粒子像更有潜力的解空间移动。而传统 PSO 在最大完工时间和执行总成本方面表现最差,分别为 21 572.02 s 和 9.75 元。结果表明,IPSO在优化最大完工时间和执行总成本方面表现出显著优势。尽管其他 7 种对比算法在优化上均优于传统的粒子群算法,但是 IPSO 相比,仍存在差距。

4.4 中规模任务

图 5 展示了 200 任务规模时各算法的收敛情况。可以看出,IPSO、AdPSO、SCA-PSO、C2PGA 和 OBL-PSO 均能获得较好的初始值,而 SAPSO、EPGA 和 PSO 的初始值相对较高。在算法迭代过程中,IPSO表现出更快的收敛速度。IPSO 算法由于其改进的惯性权重、速度和位置更新,这种动态调整加强了粒子个体的寻优能力,能更好地平衡局部开发和全局搜索,确保算法在迭代过程中能够逐步收敛到最优解,使得粒子在迭代过程中不仅能更快地找到全局最优解,还能通过局部搜索提高解的精度。结果表明,IPSO 的迭代效果明显优于其他 7 种对比算法。

图 6 为 200 任务规模时各算法执行任务调度的最大完工时间和执行总成本的实验结果。可以看出,随着任务数量的增加,AdPSO、SAPSO、OBL-PSO 和 PSO 的最大完工时间均超过 80 000 s,SCA-PSO、EPGA 和 C2PGA 则在 70 000 s 附近,而 IPSO 的最大完工时间最低,在 62 000 s 附近。IPSO 在总成本上相较于 AdPSO、SCA-PSO、SAPSO、EPGA、C2PGA、OBL-PSO 和 PSO 分别降低了 10.1%、6%、8%、9.1%、8.2%、7.5% 和 11.4%,在最大完工时间上分别降低了 24.3%、12%、23.4%、11.3%、10.7%、25.6% 和 36.5%。这是由于改进的粒子行为选择根据平均适应度值进行位置更新,使得粒子在解空间中的搜索更具有针对性,并有助于发现最优解。由此可见,IPSO 在任务调度的多目标优化中表现出明显的优势。

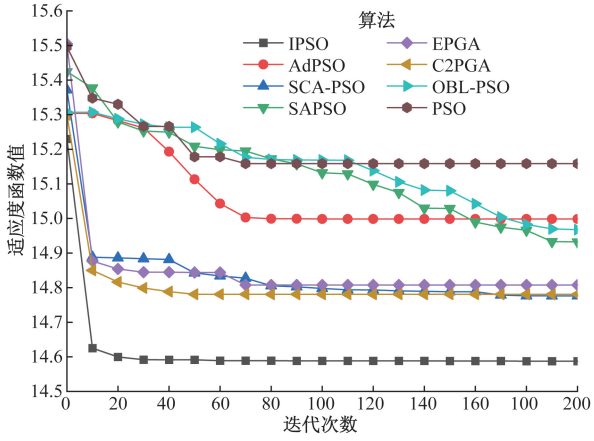


图 5 200 任务规模的收敛情况

Fig. 5 Convergence situation of 200 tasks

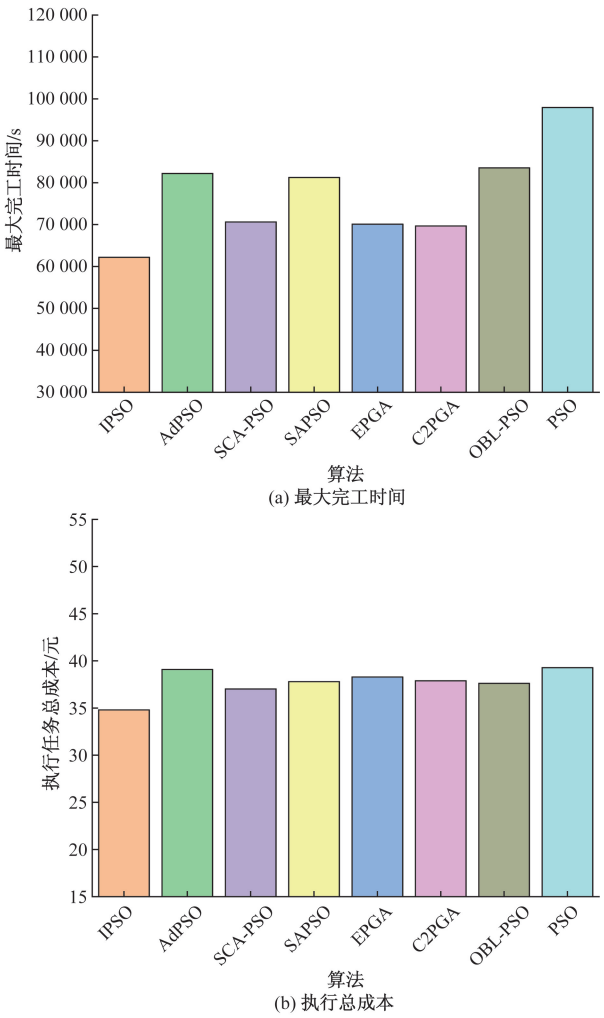


图 6 200 任务规模的指标优化情况

Fig. 6 Indicator optimization of 200 tasks

4.5 大规模任务

图 7 展示了在 500 大规模任务情景下各算法的收敛过程。可以看出,在更具随机性的大规模任务环境下,IPSO 能够在算法迭代初期就取得较低的初始值,并且具有较快的收敛速度。通过反向学习和改进的惯性权重策略,动态地调整惯性权重的大小,从而控制了粒子的运动速度,影响了粒子在解空间中的探索和利用程度,较小的惯性权重有助于粒子在局部最优解周围进行精细搜索,较大的惯性权重则有助于粒子跨越搜索空间中的局部最优解,从而更有可能找到全局最优解。这样,粒子可以更好地探索潜在解空间,并更有效地找到全局最优解,避免陷入局部最优解,从而加快算法的收敛速度。相对于其他 7 种对比算法,IPSO 表现出更快地优化速度和更高的优化精度,其算法迭代结果更为出色。

从图 8 可以明显看出,在大规模任务中,任务执行的总成本存在明显差异。随着任务数量的不断

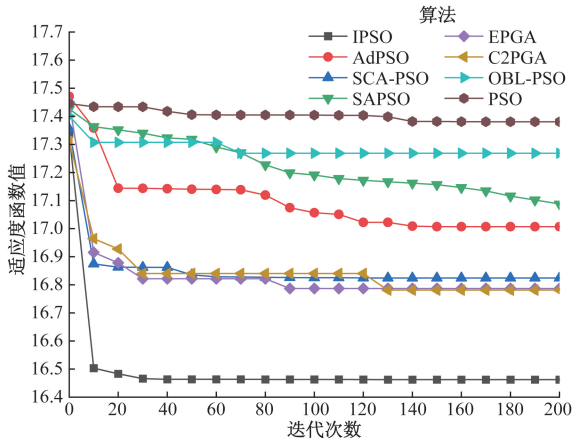


图 7　500 任务规模的收敛情况

Fig. 7　Convergence situation of 500 tasks

增加,所有算法的最大完工时间和执行总成本都呈现出增加的趋势,而 IPSO 仍然保持最低值,SCA-PSO 次之。其中,AdPSO、SAPSO、EPGA 和 C2PGA 的总成本相差不大,OBL-PSO 略优于 PSO。观察最大完工时间指标,IPSO 执行任务所需的时间为 158 952.78 s,明显低于其他 7 种对比算法。IPSO 相较于 AdPSO、SCA-PSO、SAPSO、EPGA、C2PGA、OBL-PSO 和 PSO 在总成本上分别降低了 10%、4.6%、8.6%、9.2%、8.2%、10.4% 和 11.3%,在最大完工时间上分别降低了 34.1%、27%、41.7%、28.5%、21.6%、50.3% 和 54.8%。原因在于改进的惯性权重和粒子行为选择根据与种群平均适应度值的比较,动态地调整,这样粒子可以更有效地搜索解空间,这种动态调整确保了 IPSO 在迭代过程中可以逐步收敛到最优解,并且能够在全局和局部之间取得平衡,以实现更好的搜索效果。实验结果表明,即使在大规模任务的云环境下,IPSO 依然能够实现较优的任务调度方案,使得任务的最大完工时间和总成本花费都低于其他几种对比算法。

4.6　IPSO 算法优势分析

在传统粒子群算法中,由于其惯性部分、自我认知部分和社会经验部分,使得粒子个体能够充分搜索解空间,平衡全局搜索能力和局部搜索能力,这使得该算法在全局寻优能力,收敛速度等方面表现出色,但由于对初始粒子的位置选择较为敏感,不同的初始值可能导致不同的最优解。此外,在寻优过程中容易陷入局部最优解,由于粒子之间信息共享的方式,有时会导致算法过早收敛到局部最优解而无法跳出。同时,算法在迭代早期就可能过早收敛,导致没有充分探索解空间的机会。

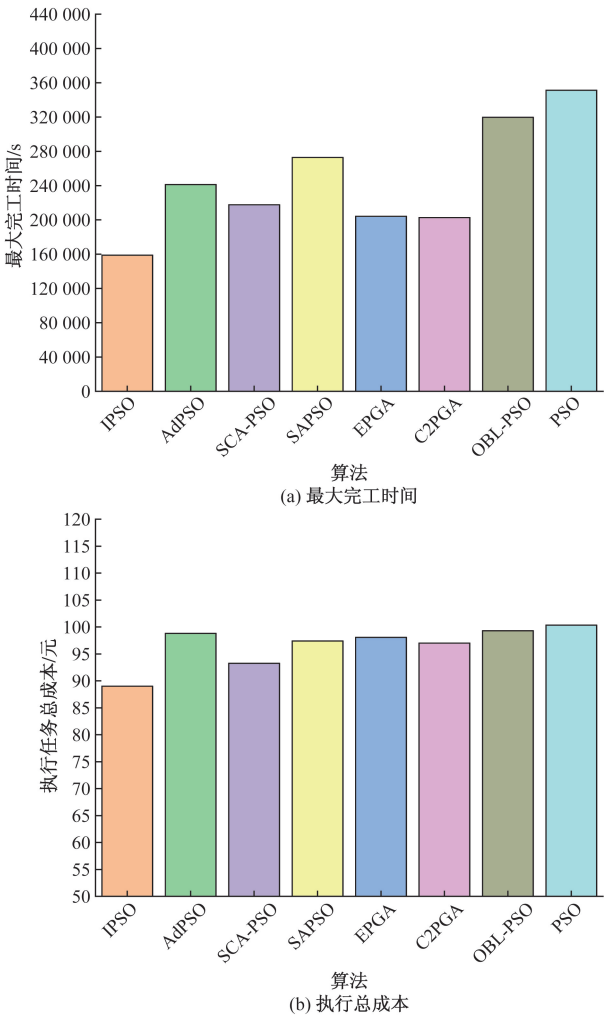


图 8　500 任务规模的指标优化情况

Fig. 8　Indicator optimization of 500 tasks

反向学习策略(OBL)可以丰富初始粒子种群的多样性,使得初始解的空间更加广泛,有助于避免粒子陷入局部最优解,使其更具有全局搜索的能力,从而更好地探索搜索空间。在正余弦算法(SCA)中,通过交替使用正弦和余弦函数,实现了全局搜索和局部开发之间的平衡,这有助于在算法的搜索过程中兼顾全局和局部最优解。将其与粒子群算法结合起来,可以通过更灵活的方式更新粒子个体和全局的最优位置,以适应更广泛的搜索空间,帮助粒子摆脱局部最优解,以获得更好的全局最优解。同时,引入种群平均适应度机制,通过考虑整个粒子群的平均适应度,更好地指导粒子移动,不同适应度值的粒子选择相应的位置更新方式,使得粒子的移动更有效和合理,以提高对全局最优解的搜索能力,增强粒子对全局结构的认知,有助于避免陷入局部最优解。上述通过第四章中 SCA-PSO 和 OBL-PSO 的实验结果可以证实这些改进的有效性。

5 结论

通过采用基于反向学习的种群初始化策略、平均适应度值的选择机制、改进的位置更新策略和正余弦算法的周期性扰动,提出一种改进的粒子群云计算任务调度算法。在 CloudSim 云计算仿真平台上,将提出的 IPSO 与其他 7 种算法进行对比。得出如下结论。

(1)在 500 任务数量时,IPSO 在总成本上相较于 AdPSO、SCA-PSO、SAPSO、EPGA、C2PGA、OBL-PSO 和 PSO 分别提升了 10%、4.6%、8.6%、9.2%、8.2%、10.4%和 11.3%,在最大完工时间上分别提升了 34.1%、27%、41.7%、28.5%、21.6%、50.3%和 54.8%,在收敛性能上分别提升了 3.1%、2.1%、3.8%、2%、1.9%、4.6%和 5.2%。

(2)在其他不同的任务规模下,IPSO 在任务最大完工时间、执行总成本、收敛速度等指标上均取得了显著的优化效果。这是因为反向学习(OBL)策略可以丰富初始粒子种群的多样性,使得初始解的空间更为广泛;同时,种群平均适应度机制和自适应惯性权重策略通过动态地调整从而更好地指导粒子移动,有助于避免陷入局部最优解,加快算法的收敛速度;改进的粒子学习行为引入更多的多样性和探索性,有助于粒子更广泛地搜索解空间,增强算法的全局搜索能力,更容易找到全局最优解。针对大规模和复杂的云计算环境下,IPSO 的性能和效率将会降低。

(3)在未来的研究中,计划考虑加入虚拟机负载、能耗、资源利用率等指标,以进一步完善解决云计算任务调度问题的优化目标。

参 考 文 献

[1] 王昊, 李晖, 宋端正, 等. 面向云-雾计算系统中的遗传算法任务调度研究[J]. 电子测量与仪器学报, 2023, 37(8): 40-51.
Wang Hao, Li Hui, Song Duanzheng, et al. Research on genetic algorithm task scheduling in cloud-fog computing systems[J]. Journal of Electronic Measurement and Instrumentation, 2023, 37(8): 40-51.

[2] Zhang Y. Cloud computing task scheduling algorithms and advances [J]. Highlights in Science, Engineering and Technology, 2022, 7: 368-373.

[3] 陈俊仁, 郭一晶. 基于交换改进粒子群算法的云计算任务调度[J]. 计算机应用与软件, 2023, 40(6): 223-228.
Chen Junren, Guo Yijing. Cloud computing task scheduling based on exchange improved particle swarm algorithm[J]. Computer Applications and Software, 2023, 40(6): 223-228.

[4] Li K, Jia L, Shi X. Research on cloud computing task scheduling based on PSOMC[J]. Journal of Web Engineering, 2022, 21(6): 1749-1766.

[5] Prity F S, Gazi M H, Uddin K M A. A review of task scheduling in cloud computing based on nature-inspired optimization algorithm [J]. Cluster Computing, 2023, 26(5): 3037-3067.

[6] Fang Y Q, Xiao X, Ge J W. Cloud computing task scheduling algorithm based on improved genetic algorithm[C]//2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference(ITNEC). New York: IEEE, 2019: 852-856.

[7] Jaber S S, Ali Y, Ibrahim N. Task scheduling in cloud computing based on the cuckoo search algorithm[J]. Iraqi Journal of Computers, Communications, Control and Systems Engineering, 2022, 22(1): 86-96

[8] Beegom A S A, Rajasree M S. Integer-PSO: a discrete PSO algorithm for task scheduling in cloud computing systems[J]. Evolutionary Intelligence, 2019, 12: 227-239.

[9] 周敬东, 高伟周, 杨文广, 等. 基于改进蚁群算法的移动机器人路径规划[J]. 科学技术与工程, 2022, 22(28): 12484-12490.
Zhou Jingdong, Gao Weizhou, Yang Wenguang, et al. Mobile robot path planning based on improved ant colony algorithm[J]. Science Technology and Engineering, 2022, 22(28): 12484-12490.

[10] 贾嘉, 慕德俊. 基于人工蜂群的云计算负载均衡算法[J]. 科学技术与工程, 2020, 20(16): 6532-6537.
Jia Jia, Mu Dejun. Cloud computing load balancing algorithm based on artificial bee colony[J]. Science Technology and Engineering, 2020, 20(16): 6532-6537.

[11] 张艳华, 黄静梅, 黄景光, 等. 基于改进粒子群算法的梯级水风光短期调峰优化调度[J]. 科学技术与工程, 2022, 22(12): 4993-5000.
Zhang Yanhua, Huang Jingmei, Huang Jingguang, et al. Short-term peak-shaving optimization scheduling of cascade hydropower, wind power and solar power based on improved particle swarm optimization algorithm [J]. Science Technology and Engineering, 2022, 22(12): 4993-5000.

[12] 王飞, 杨清平. 基于改进粒子群算法的城市物流无人机路径规划[J]. 科学技术与工程, 2023, 23(30): 13187-13194.
Wang Fei, Yang Qingping. Path planning for urban logistics drones based on improved particle swarm algorithm[J]. Science Technology and Engineering, 2023, 23(30): 13187-13194.

[13] John H H. Adaptation innatural and artificial systems[M]. Berlin: Springer International Publishing, 1975.

[14] Zhang Y. Cloud computing task scheduling algorithms and advances[J]. Highlights in Science, Engineering and Technology, 2022, 7: 368-373.

[15] Kirkpatrick S, Gelatt J C D, Vecchi M P. Optimization by simulated annealing[J]. Science, 1983, 220(4598): 671-680.

[16] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007, 39: 459-471.

[17] Agarwal M, Srivastava G M S. Genetic algorithm-enabled particle swarm optimization(PSOGA)-based task scheduling in cloud computing environment[J]. International Journal of Information Technology & Decision Making, 2018, 17(4): 1237-1267.

[18] Senthil K A M, Venkatesan M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment[J]. Wireless Personal Communications, 2019, 107: 1835-1848.

[19] 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.

[20] Senthil K A M, Krishnamoorthy P, Soubraylu S, et al. An efficient task scheduling using GWO-PSO algorithm in a cloud computing environment[C]//Proceedings of International Conference on Intelligent Computing, Information and Control Systems. Singapore: Springer, 2021: 751-761.

[21] 汪婷, 邵鹏, 李光泉, 等. 改进的粒子群优化算法在云计算任务调度中的应用[J]. 科学技术与工程, 2023, 23(29): 12594-12603.
Wang Ting, Shao Peng, Li Guangquan, et al. Application of improved particle swarm optimization algorithm in cloud computing task scheduling[J]. Science Technology and Engineering, 2023, 23(29): 12594-12603.

[22] 谢承旺, 邹秀芬, 夏学文, 等. 一种多策略融合的多目标粒子群优化算法[J]. 电子学报, 2015, 43(8): 1538-1544.
Xie Chengwang, Zou Xiufen, Xia Xuewen, et al. A multi-objective particle swarm optimization algorithm with multi-strategy fusion [J]. Journal of Electronics, 2015, 43(8): 1538-1544.

[23] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems[J]. Knowledge-Based Systems, 2016, 96: 120-133.

[24] Al-Olimat H S, Alam M, Green R, et al. Cloudlet scheduling with particle swarm optimization[C]//2015 5th International Conference on Communication Systems and Network Technologies. New York: IEEE, 2015: 991-995.

[25] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proceedings of ICNN'95-International Conference on Neural Networks. New York: IEEE, 1995: 1942-1948.

[26] Tizhoosh H R. Opposition-based learning: a new scheme for machine intelligence[C]//International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on intelligent Agents, Web Technologies and Internet Commerce(CIMCA-IAWTIC'06). New York: IEEE, 2005: 695-701.

[27] Malisia A R, Tizhoosh H R. Applying opposition-based ideas to the ant colony system[C]//2007 IEEE Swarm Intelligence Symposium. New York: IEEE, 2007: 182-189.

[28] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64-79.

[29] Huang X, Li C, Chen H, et al. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies[J]. Cluster Computing, 2020, 23: 1137-1147.

[30] Sriperambuduri V K. Effective workflow scheduling in cloud using constriction factor based inertia weight particle swarm optimization [J]. International Journal on Recent and Innovation Trends in Computing and Communication, 2023, 11: 122-131.

[31] 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.

[32] 王镇道, 张一鸣, 石雪倩. 基于竞争粒子群算法的云计算资源调度策略[J]. 湖南大学学报(自然科学版), 2021, 48(6): 80-87.
Wang Zhendao, Zhang Yiming, Shi Xueqian. Cloud computing resource scheduling strategy based on competitive particle swarm algorithm[J]. Journal of Hunan University(Natural Science Edition), 2021, 48(6): 80-87.

[33] 孙长亚, 王向文. 基于 MGA-PSO 的云计算多目标任务调度[J]. 计算机应用与软件, 2021, 38(6): 212-218.
Sun Changya, Wang Xiangwen. Cloud computing multi-objective task scheduling based on MGA-PSO[J]. Computer Applications and Software, 2021, 38(6): 212-218.

[34] Nabi S, Ahmad M, Ibrahim M, et al. AdPSO: adaptive PSO-based task scheduling approach for cloud computing[J]. Sensors, 2022, 22(3): 920.

[35] Wang X H, Li J J. Hybrid particle swarm optimization with simulated annealing[C]//Proceedings of 2004 International Conference on Machine Learning and Cybernetics. New York: IEEE, 2004: 2402-2405.

[36] 付学良, 孙扬, 王海芳, 等. 改进遗传算法在云任务调度中的应用研究[J]. 内蒙古农业大学学报(自然科学版), 2020, 41(4): 64-69.
Fu Xueliang, Sun Yang, Wang Haifang, et al. Application of improved genetic algorithm in cloud task scheduling[J]. Journal of Inner Mongolia Agricultural University(Natural Science Edition), 2020, 41(4): 64-69.

[37] 孙扬. 基于改进智能优化算法的云任务调度研究[D]. 呼和浩特: 内蒙古农业大学, 2021.
Sun Yang. Research on cloud task scheduling based on improved intelligent optimization algorithm[D]. Hohhot: Inner Mongolia Agricultural University, 2021.

[38] 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.