

混合灰狼鲸鱼优化算法的云计算任务调度

曾浩文¹, 谢晓兰^{1,2}, 郭强¹

(1. 桂林理工大学 计算机科学与工程学院; 2. 广西嵌入式技术与智能系统重点实验室, 广西 桂林 514000)

摘要: 针对传统优化算法在面对复杂问题时的局限性, 以及容易陷入局部最优解、收敛速度较慢与搜索能力不足等问题, 创新性地提出一种融合灰狼优化算法(GWO)与鲸鱼优化算法(WOA)的混合优化策略。该策略综合利用GWO强大的全局探索能力与WOA快速收敛的特性, 显著提升了算法的全局搜索效率。在CEC2017基准函数上的实验结果表明, 该混合算法相比传统优化算法性能显著提升。在云计算中, 可在提升任务调度效率的同时有效降低成本, 提升虚拟机资源的利用率。

关键词: 云计算; 任务调度; 灰狼优化算法; 鲸鱼优化算法; 混沌映射; 柯西变异

DOI: 10.11907/rjdk.241450

扫描二维码阅读全文:



中图分类号: TP18; TP393.9

文献标识码: A

文章编号: 1672-7800(2025)010-0117-10

Cloud Computing Task Scheduling with Hybrid Grey Wolf Whale Optimization Algorithm

ZENG Haowen¹, XIE Xiaolan^{1,2}, GUO Qiang¹

(1. School of Information Science and Engineering, Guilin University of Technology;

2. Guangxi Key Laboratory of Embedded Technology and Intelligent Systems, Guilin 514000, China)

Abstract: In response to the limitations of traditional optimization algorithms in addressing complex problems, such as tendency to fall into local optima, slow convergence speed, and insufficient search capability, this paper innovatively proposes a hybrid optimization strategy that integrates the Grey Wolf Optimizer (GWO) and the Whale Optimization Algorithm (WOA). By leveraging GWO's strong global exploration capability and WOA's fast convergence behavior, the proposed strategy significantly enhances the overall search efficiency. Experimental results on the CEC2017 benchmark functions show that the hybrid algorithm outperforms traditional optimization algorithms considerably. In cloud computing applications, the method can improve task scheduling efficiency while reducing costs and enhancing the utilization of virtual machine resources.

Key Words: cloud computing; task scheduling; grey wolf optimization algorithm; whale optimization algorithm; chaotic mapping; Cauchy variation

0 引言

云计算作为分布式计算的重要组成部分, 广泛应用于数据存储、数据分析和物联网等领域。其服务模式包括基础设施即服务(Infrastructure as a Service, IaaS)、平台即服务(Platform as a Service, PaaS)和软件即服务(Software as a Service, SaaS)^[1-2]。根据云计算白皮书的数据, 中国的云计算市场在2024年达到5 864亿元, 较2023年增长19.4%,

反映了该行业的快速发展。然而, 如何高效地调度和分配资源一直是云计算的核心挑战, 吸引了众多学者的关注。

任务调度在云计算中扮演着关键角色, 其主要目标是提高虚拟机的利用率, 降低数据中心的运营成本, 从而提升服务质量(Quality of Service, QoS)和整体性能。高效的任务调度策略需要在满足QoS要求的同时, 优化时间、成本、能耗等^[3]。独立调度和从属调度是两种主要的任务调度方法。本文重点关注独立任务调度。已有研究表明, 元启发式算法相较于传统算法, 具有更强的全局搜索能力和

收稿日期: 2024-05-12

基金项目: 广西重点研发计划项目(2022AB05005, 2022AB05024)

作者简介: 曾浩文(1999-), 男, 桂林理工大学计算机科学与工程学院硕士研究生, 研究方向为云计算、任务调度、智能优化算法; 谢晓兰(1974-), 女, 博士, 桂林理工大学计算机科学与工程学院教授、博士生导师, 研究方向为云计算、大数据; 郭强(1977-), 男, 桂林理工大学计算机科学与工程学院讲师, 研究方向为大数据、智能计算。本文通讯作者: 谢晓兰。

鲁棒性,能够更有效地解决任务调度中的多种难题^[4-5]。

然而,单一算法往往存在局限性,尤其在求解速度与精度方面表现不足。为此,研究者逐渐开始探索混合元启发式算法,通过整合两种或多种算法的优势,以提升算法的整体性能。此类混合策略不仅有助于维持种群多样性、避免早熟收敛,而且能有效提升全局搜索效率与解的准确性,从而更高效地应对复杂的任务调度问题。

杂交灰狼鲸鱼优化算法(Grey Wolf Optimizer Whale Optimization Algorithm, GWOWOA)是Obadina等^[6]在2022年提出的一种新型混合算法,其结合了灰狼优化算法(Grey Wolf Optimizer, GWO)^[7]和鲸鱼优化算法(Grey Wolf Optimizer, WOA)^[8]的优点,已应用于任务调度等领域。然而,GWOWOA在面对复杂多模态问题时收敛速度较慢,易陷入局部最优,且对初始参数设置敏感,在动态环境下缺乏自适应性。

为解决以上问题,本文提出一种改进的混合算法TC-GWOWOA(Tent Cauchy GWOWOA)。通过引入Tent映射增强种群初始化的多样性,改进非线性收敛因子以提高收敛速度,并采用柯西变异增强全局搜索能力。这些改进旨在全面提升算法性能,从而更有效地解决复杂问题。实验结果表明,TCGWOWOA在基准测试函数和云计算任务调度中的表现优于GWOWOA算法,显著提高了任务调度效率和系统性能。

1 云计算任务调度

1.1 研究背景

随着信息技术的迅速发展,云计算已成为现代计算的重要组成部分。通过互联网提供计算资源和服务,云计算显著提高了计算效率和灵活性。其具备强大的计算和存储能力,可以高效完成很多复杂任务。例如,在科学计算中,研究人员可利用云计算平台进行气候模拟和基因组分析等;在企业应用中,云计算可帮助企业进行订单处理和财务分析等。

1.2 研究目标

本文旨在解决云计算任务调度中的关键问题,即最小化任务的最大完成时间(MakeSpan)。为实现这一目标,本文将比较不同算法在以下3个方面的性能:①执行时间:系统完成所有任务所需的总时间;②最大完成时间:所有任务被完全调度并执行完毕所需的总时间,本文以最大完成时间来衡量任务调度效率;③成本:系统在完成任务时所产生的总成本,包括计算资源的使用费用等。

1.3 问题描述与符号定义

1.3.1 问题描述

任务调度是云计算实际应用中的关键问题,其目标是将任务高效分配到虚拟机上,以优化性能指标,最小化最大完成时间(MakeSpan)、降低执行成本并提高资源利

用率。

1.3.2 任务与虚拟机合集

假设有 n 个任务的集合 Task 和云资源中 m 个虚拟机的集合 Vm:

$$\begin{aligned} \text{Task} &= \{\text{task}_1, \text{task}_2, \dots, \text{task}_n\} \\ \text{Vm} &= \{\text{vm}_1, \text{vm}_2, \dots, \text{vm}_m\} \end{aligned} \quad (1)$$

1.3.3 任务与虚拟机之间映射矩阵

定义一个任务与虚拟机之间的映射矩阵 A,其中每个任务只能在一个虚拟机上运行,但一个虚拟机可以处理多个任务。所有任务都是相互独立的。矩阵 A 表示如下:

$$A_{nm} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \quad (2)$$

1.3.4 任务执行时间

任务执行时间可通过以下公式进行计算:

$$ETC_{ij} = \frac{\text{length}(i)}{\text{mips}(j)} \quad (3)$$

其中, ETC_{ij} 表示第 i 个任务在第 j 个虚拟机上的执行时间, $\text{length}(i)$ 表示第 i 个任务的长度, $\text{mips}(j)$ 表示第 j 个虚拟机的计算能力。

定义虚拟机上执行所分配任务的时间总和,计算公式如下:

$$\text{vmTime}_j = \sum_{i=1}^n A_{ij} \cdot ETC_{ij} \quad (4)$$

1.3.5 任务最大完成时间

系统中任务的最大完成时间(MakeSpan)计算公式如下:

$$\text{MakeSpan} = \max_{j \in \{1, \dots, m\}} (\text{vmTime}_j) \quad (5)$$

1.3.6 任务成本

任务成本是指任务执行的时间成本、通信时间成本和等待时间成本。具体计算公式如下:

$$\text{Cost} = \text{ETC} \cdot \text{ETCCost} + \text{commTime} \cdot \text{commCost} + \text{waitTime} \cdot \text{waitCost} \quad (6)$$

其中,ETC表示预计执行时间,ETCCost表示每单位执行时间的成本,commTime表示通信时间,commCost表示每单位通信时间的成本,waitTime表示等待时间,waitCost表示每单位等待时间的成本。

2 相关工作

任务调度是云计算领域的研究热点,近年来已开展了大量研究工作。现有任务调度算法主要分为启发式方法、元启发式方法和混合方法。以下讨论几类典型的调度策略及其局限性,并与本文工作进行比较。

2.1 启发式方法

启发式算法包括 Min-Min、Max-Min 和最短任务优先

(Shortest Remaining Time First, SJF) 算法等, 因其算法简单、计算效率高而得到了广泛应用。这些方法虽然能快速提供调度方案, 但往往只考虑了局部信息, 容易陷入局部最优, 且在动态云计算环境中适应性差, 难以灵活应对资源和任务的变化。

2.2 元启发式方法

元启发式方法如遗传算法 (Genetic Algorithm, GA)、粒子群优化 (Particle Swarm Optimization, PSO) 算法等, 具有较强的全局搜索能力, 适用于复杂搜索空间。文献[9]改进了人工蜂群算法优化云任务调度, 但仅考虑了任务完成时间。文献[10]通过基于交换改进的 PSO 优化任务调度, 但未考虑计算成本、带宽、能耗等因素。文献[11]尽管改进了多目标任务调度, 但未讨论任务调度成本。

2.3 混合启发式方法

混合方法结合了启发式和元启发式方法的优势, 可提升算法性能和问题解决的质量。文献[12]提出将萤火虫算法与遗传算法融合, 但其仅简单结合两者的操作步骤, 未深入探讨如何充分利用相关算法的优点并克服缺点。文献[13]通过融合 PSO 和模拟退火算法提升优化效果, 但其评价标准单一, 仅考虑了子任务时间, 未全面衡量算法性能。

本文提出一种改进的混合优化方法, 旨在弥补现有研究中的不足。通过综合考虑多个优化标准, 深入分析不同算法的优势和缺陷, 以构建更高效的任务调度算法, 并提升算法整体性能。

3 基础算法描述

3.1 灰狼优化算法

灰狼优化算法 (Grey Wolf Optimizer, GWO) 的灵感源于灰狼群体的社会行为, 主要模拟其猎食过程。该算法主要分为以下几个阶段:

(1) 包围猎物。灰狼试图包围猎物, 其数学模型如下:

$$D = |C \cdot X_p(t) - X(t)| \quad (7)$$

$$X(t+1) = X_p(t) - A \cdot D \quad (8)$$

其中, t 是当前迭代次数, A 、 C 是系数向量, X_p 、 X 分别是猎物和灰狼的位置向量。 A 和 C 的计算公式如下:

$$A = 2a \cdot r_1 - a \quad (9)$$

$$C = 2 \cdot r_2 \quad (10)$$

$$a = 2 - \frac{2t}{t_{\max}} \quad (11)$$

其中, a 是收敛因子, 随着迭代次数线性从 2 下降到 0, r_1 和 r_2 的模取 $[0, 1]$ 之间的随机数。

(2) 狩猎猎物。在狩猎过程中, 灰狼按照 α 、 β 和 δ 3 种角色对猎物进行定位, 更新位置如下:

$$\begin{cases} D_\alpha = |C_1 \cdot X_\alpha - X| \\ D_\beta = |C_2 \cdot X_\beta - X| \\ D_\delta = |C_3 \cdot X_\delta - X| \end{cases} \quad (12)$$

$$\begin{cases} X_1 = X_\alpha - A_1 \cdot (D_\alpha) \\ X_2 = X_\beta - A_2 \cdot (D_\beta) \\ X_3 = X_\delta - A_3 \cdot (D_\delta) \end{cases} \quad (13)$$

$$X(t+1) = \frac{(X_1 + X_2 + X_3)}{3} \quad (14)$$

(3) 攻击猎物。在算法的最后阶段, 灰狼对猎物进行攻击, 通过微调位置, 旨在跳出局部最优陷阱, 搜寻全局最优解。

3.2 鲸鱼优化算法

鲸鱼优化算法 (Whale Optimization Algorithm, WOA) 是一种基于生物行为的全局优化算法, 该算法的灵感来源于座头鲸群体捕食的策略, 特别是其独特的泡泡网捕食行为。该算法主要分为以下几个阶段:

(1) 包围猎物。鲸鱼调整位置以包围目标, 减小到目标的距离, 形成包围猎物的策略。

$$D = |C \cdot X^*(t) - X(t)| \quad (15)$$

$$X(t+1) = X^*(t) - A \cdot D \quad (16)$$

$$C_1 = 2r_2 \quad (17)$$

$$A_1 = 2a \cdot r_1 - a \quad (18)$$

其中, t 为当前迭代次数, X^* 为当前最佳解, A 和 C 为系数, r 为 $[0, 1]$ 中的随机向量。

(2) 螺旋更新位置。鲸鱼通过螺旋运动产生气泡, 逐渐接近并捕获目标, 实现更精准的搜索。

$$X(t+1) = D' \cdot e^{bl} \cos(2\pi l) + X^*(t) \quad (19)$$

$$D' = |X^*(t) - X(t)| \quad (20)$$

$$X(t+1) = \begin{cases} X^*(t) - A \cdot D_1 & p < 0.5 \\ D_2 \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t) & p \geq 0.5 \end{cases} \quad (21)$$

其中, D' 为当前搜索个体与当前最优解的距离, b 为螺旋形状参数, l 是一个在区间 $[-1, 1]$ 内均匀生成的随机数。 p 为捕食机制概率, 值域为 $[0, 1]$ 的随机数。

(3) 搜索猎物。为保证所有鲸鱼能在解空间中充分搜索, WOA 根据鲸鱼彼此之间的距离来更新位置, 以达到随机搜索的目的。

$$D = |C \cdot X_{\text{rand}}(t) - X(t)| \quad (22)$$

$$X(t+1) = X_{\text{rand}} - A \cdot D \quad (23)$$

4 改进的混合算法 TCGWOWOA

4.1 杂交灰狼鲸鱼优化算法

杂交灰狼鲸鱼优化算法的创新性主要体现在两个方面。一方面, 在改进 α 的位置更新上进行优化。在灰狼优化算法中, 头狼 α 的位置对最优解至关重要。本文将头狼

α 的位置更新与鲸鱼优化算法的气泡网攻击阶段相结合, 以实现更精确、高效的优化, 提升算法在不同问题上的适应性。

$$d_{\alpha} = \begin{cases} r \left| \vec{c}_1 x_{\alpha} - \vec{x} \right| & \text{if } p_1 < 0.5 \\ \rho e^{bl} \cos(2\pi l) \left| \vec{c}_1 x_{\alpha} - \vec{x} \right| & \text{if } p_1 \geq 0.5 \end{cases} \quad (24)$$

其中, p_1 、 r 和 ρ 是 $[0, 1]$ 范围内的随机数, l 是 $[-1, 1]$ 范围内的随机数, b 是常数。

另一方面的改进侧重于灰狼开发阶段的 $\vec{a}$, 具体公式如下:

$$\vec{a} = 2 \left(1 - \frac{t}{2\text{Max}_{\text{iter}}} \right) \quad (25)$$

该改进旨在优化灰狼个体的开发阶段, 通过调整 $\vec{a}$ 来提升算法的整体性能, 从而增强其求解复杂优化问题的能力。然而, 以上针对 $\vec{a}$ 调整的效果不如预期。因此, 本文进一步引入非线性收敛因子, 以提升算法性能。所引入的非线性动态机制旨在提升算法的适应性, 以应对复杂多变的优化环境, 从而提升其全局探索能力。

4.2 改进的 Tent 混沌映射初始化种群

种群初始化在元启发智能算法中至关重要。随机初

始化会导致个体聚集、多样性不足、分布不均匀等问题, 影响优化结果。为克服以上问题, 本文引入混沌映射, 以提升搜索的多样性和均匀性^[14]。通过优化种群初始化策略, 算法能够更全面地覆盖问题空间, 探索潜在的解决方案。

Tent 映射是常用的混沌映射, 但其最小周期和不动点模型可能导致局部循环, 影响效率。因此, 本文改进了 Tent 混沌映射, 以提高其遍历性和搜索效率。这一改进可更好地支持种群初始化, 为优化提供更可靠的起点。

标准的 Tent 映射公式如下:

$$X_{i+1} = \begin{cases} 2 * X_i, & 0 \leq X_i < 1/2 \\ 2 * (1 - X_i), & 1/2 \leq X_i < 1 \end{cases} \quad (26)$$

改进的 Tent 映射公式如下:

$$X_{i+1} = \begin{cases} 1 - X_i * \text{rand}(0.1) & X_i = 0 \\ 2 * X_i, & 0 \leq X_i < 1/2 \\ 2 * (1 - X_i), & 1/2 \leq X_i < 1 \end{cases} \quad (27)$$

原始 Tent 映射初始化种群与改进后 Tent 映射初始化种群对比如图 1 所示。由图 1 可见, 原始 Tent 映射呈现出明显的短周期状态, 而改进后的 Tent 混沌映射分布更加均匀。这表明本文算法具备更佳的种群多样性, 可获得更好的优化效果。

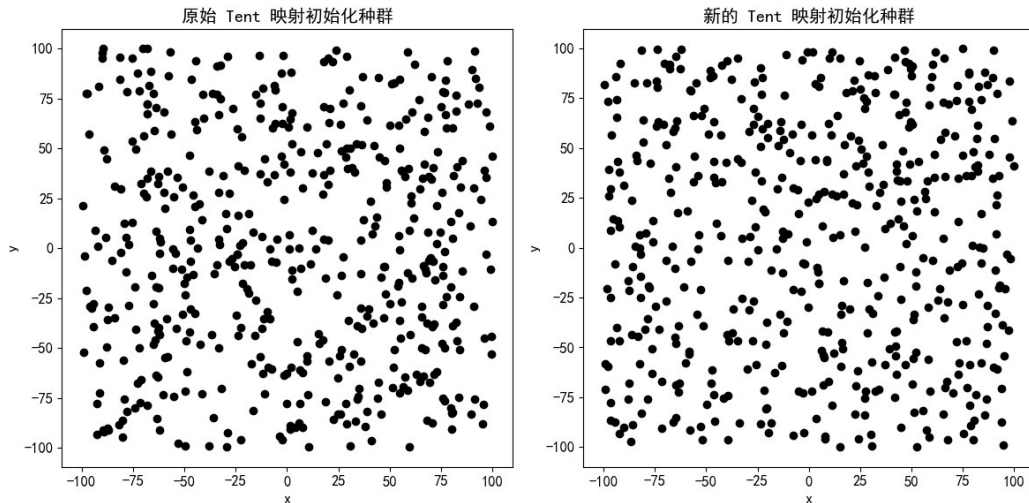


Fig. 1 Comparison between the original Tent mapping initialization population and the improved Tent mapping initialization population

图 1 原始 Tent 映射初始化种群与改进后 Tent 映射初始化种群对比

4.3 非线性收敛因子

灰狼优化算法和鲸鱼优化算法的探索与开发能力主要依赖其内在的搜索机制。参数 A 对算法的局部开发和全局勘探至关重要, 且由收敛因子 a 调节。随着迭代次数增加, 线性递减至 0。

在迭代初期, 较大的 a 值可提升全局搜索能力, 促进算法快速探索解空间, 避免过早收敛。在迭代后期, 较小的 a 值可增加局部搜索权重, 使算法更细致地深入搜索, 提高局部搜索效率, 加速算法的收敛过程^[15]。这种动态调整机制平衡了全局探索与局部优化, 使算法更适应不同阶段的问题特性。具体公式如下:

$$a = \begin{cases} \text{start} + 0.3 \cdot \sin\left(\frac{2\pi \cdot t}{t_{\max}}\right) \cdot \exp\left(-0.1 \cdot \frac{t}{t_{\max}}\right), & \text{if } t \leq \frac{t_{\max}}{2} \\ \text{start} - (\text{start} - \text{end}) \cdot \frac{t - \frac{t_{\max}}{2}}{0.5 \cdot t_{\max}}, & t > \frac{t_{\max}}{2} \end{cases} \quad (28)$$

$$\text{start} = 2 - \frac{2 \cdot t}{(t_{\max} - 1)} \quad (29)$$

$$\text{end_point} = 2 - \frac{2 \cdot t_{\max}}{t_{\max} - 1} \quad (30)$$

非线性收敛因子与线性函数对比如图 2 所示。

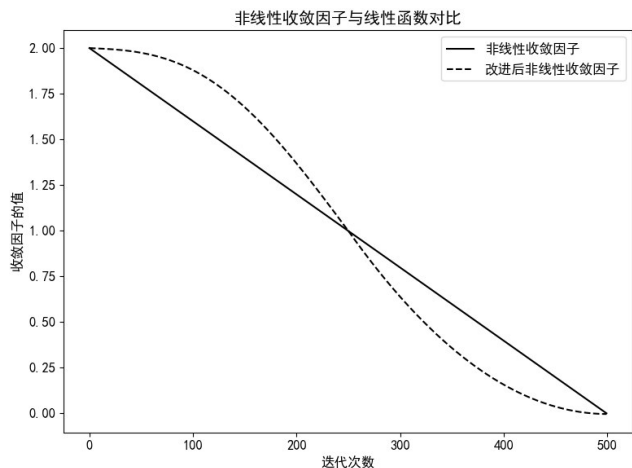


Fig. 2 Comparison of nonlinear and linear convergence factors

图2 非线性收敛因子与线性函数对比

4.4 柯西变异

为克服 GWOWOA 易陷入局部最优的问题, 本文引入柯西变异, 通过柯西分布对最优个体进行扰动。柯西分布在原点处峰值较小, 两端分布较长, 有助于算法跳出局部最优, 同时加快求解速度。

柯西变异对最优个体的更新公式如下:

$$f(x; x_0, \gamma) = \frac{1}{\pi\gamma \left[1 + \left(\frac{x - x_0}{\gamma} \right)^2 \right]} \quad (31)$$

其中, x_0 是位置参数, γ 是尺度参数。

柯西分布概率密度函数图如图 3 所示。

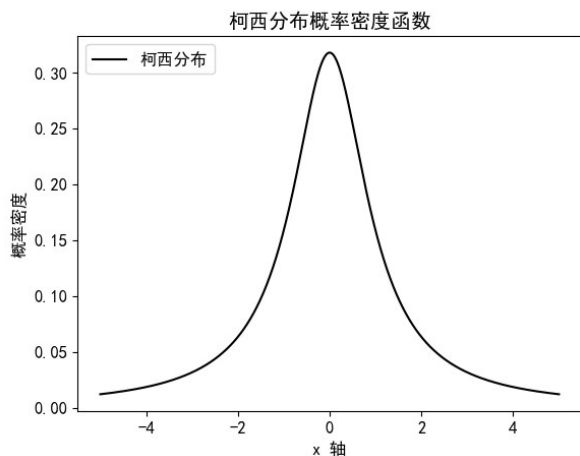


Fig. 3 Probability density function graph of the Cauchy distribution

图3 柯西分布概率密度函数图

4.5 混合灰狼鲸鱼优化算法

混合灰狼鲸鱼优化算法如下:

算法 TCQLGWO_WOA

输入: 最大迭代次数 max_epoch, 种群大小 pop_size, 问题边界 problem_bounds, 目标函数 obj_fun

输出: 最优解 best_solution

1. Tent 初始化种群, 大小为 pop_size, 生成问题边界内的解。

2. 使用 obj_fun 评估每个解的适应度。
3. 根据适应度排序种群。
4. 将初始种群中的最佳解设为 best_solution。
5. FOR epoch 从 0 到 max_epoch-1
6. 使用收敛函数 convergence_function1(epoch) 计算自适应参数 a
7. 选择前 3 个解作为 alpha、beta 和 delta 狼
8. 对于种群中的每个解:
9. 生成随机系数 A1、A2、A3、C1、C2、C3
10. 计算 alpha 狼的位置更新 X1: $X1 = \alpha - A1 * C1 * \alpha$ - 当前解
11. 计算 beta 狼的位置更新 X2: $X2 = \beta - A2 * C2 * \beta$ - 当前解
12. 计算 delta 狼的位置更新 X3: $X3 = \delta - A3 * C3 * \delta$ - 当前解
13. 使用 alpha、beta 和 delta 狼及系数计算新位置: 新位置 = $(X1 + X2 + X3) / 3$
14. 使用 WOA 计算新位置: 根据选择的捕食方式(包围、螺旋更新位置或随机搜索)更新位置
15. 如果随机数 < 0.5 且 $A < 1$, 则使用最佳解决方案更新位置(包围猎物)
- 新位置 = $\alpha - A * \alpha$ - 当前解
16. 如果随机数 < 0 且 $A \geq 1$, 则随机选择一个种群中的解决方案更新位置(随机搜索)
- 新位置 = 随机解 - $A * C * \text{随机解}$ - 当前解
17. 如果随机数 ≥ 0.5 , 则使用螺旋方程更新位置(螺旋更新位置)
- $D = \alpha$ - 当前解
- 新位置 = $D * e^{(b * l)} * \cos(2 * \pi * l) + \alpha$
18. 计算新位置作为新的候选解
19. 如果候选解超出边界, 则进行修正
20. 使用柯西变异对候选解进行变异
21. 评估新候选解的适应度
22. 如果新候选解的适应度更高, 则替换旧解
23. 如果种群中的任何解适应度优于 best_solution, 则更新 best_solution
24. 返回 best_solution

4.6 改进后混合算法流程

改进后的杂交灰狼鲸鱼优化算法 TCGWOWOA 流程如图 4 所示。

5 CEC2017 仿真实验

5.1 实验配置与参数设置

在仿真环境中, 采用 Windows 10 操作系统、Intel Core i7-7700HQ 处理器和 16 GB 内存。实验使用 CEC2017 基准函数进行性能评估, 涵盖单峰、多峰、可分离和不可分离函数等。实验设置种群规模为 50, 最大迭代次数为 500。每个算法独立运行 30 次, 并采用 mealpy 库作为自动化框架, 为求解优化问题提供了高效的解决方案, 确保实验公正、可靠, 避免因不同条件引起的偏差^[16]。

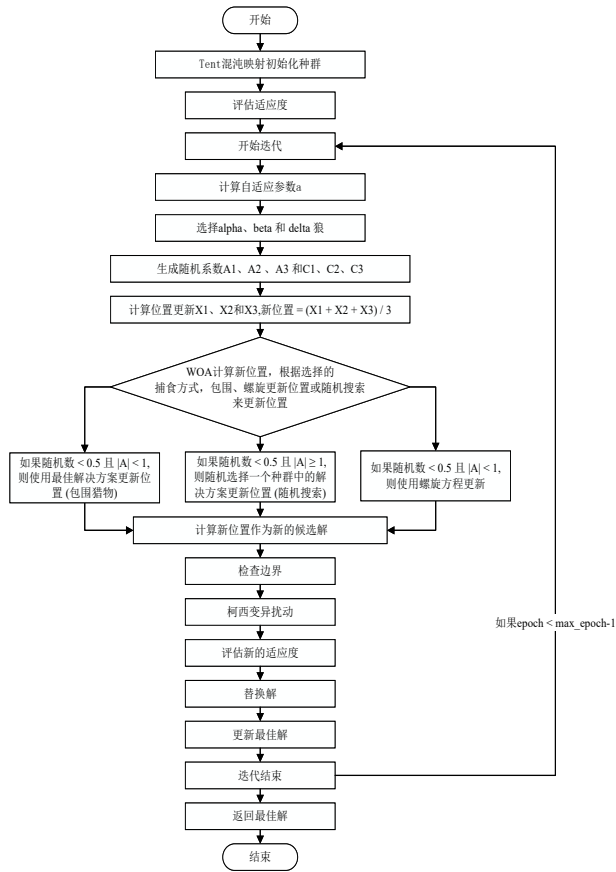


Fig. 4 Improved hybrid algorithm flow

图4 改进混合算法流程

5.2 算法测试

为了测试算法的改进效果,本文选取无约束的 CEC2017 基准函数进行测试。测试数据如表 1、表 2 所示。

5.3 不同算法对比

本文通过多方面对比实验验证 TCGWOWOA 算法的

Table 1 Unimodal functions

表1 单模态函数

标准基准函数测试	搜索范围	维度	极值
$f_1(X) = \sum_{i=1}^D X_i^2$	$[-100, 100]$	30	0
$f_2(x) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	$[-100, 100]$	30	0
$f_3(x) = \sum_{i=1}^D \left(\sum_{j=1}^i x_j \right)^2$	$[-100, 100]$	30	0
$f_4(x) = \max_i \{ x_i , 1 \leq i \leq D \}$	$[-30, 30]$	30	0
$f_5(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2) + (x_i - 1)^2 \right]$	$[-100, 100]$	30	0
$f_6(X) = \sum_{i=1}^D (X_i + 0.5)^2$	$[-100, 100]$	30	0
$f_7(X) = \sum_{i=1}^D iX_i^4 + \text{rand}[0, 1]$	$[-1.28, 1.28]$	30	0

Table 2 Multimodal functions

表2 多模态函数

标准基准函数测试	搜索范围	维度	极值
$f_8(X) = \sum_{i=1}^D -X_i \sin(\sqrt{ i })$	$[-500, 500]$	30	-418.982 9
$f_9(X) = \sum_{i=1}^D [X_i^2 - 10\cos(2\pi X_i) + 10]$	$[-5.12, 5.12]$	30	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30	0
$f_{11}(X) = \frac{1}{4000} \left(\sum_{i=1}^D X_k^2 - \prod_{i=1}^D \cos\left(\frac{X_i}{\sqrt{i}}\right) + 1 \right)$	$[-600, 600]$	30	0
$f_{12}(x) = \frac{\pi}{D} \left\{ 10\sin(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \right\} + \sum_{i=1}^D u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]$	30	1
$f_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)] \right\} + \sum_{i=1}^D u(x_i, 5, 100, 4)$	$[-50, 50]$	30	0.000 3

优越性。首先,对比经典算法 GWO^[11]、WOA^[12]、ACO^[17];其次,对比近年来提出的多种高性能的改进算法 GWOWOA^[10]、IGWO^[18]、HI-WOA^[19]。其中,GWOWOA 混合算法融合了灰狼优化与鲸鱼优化算法的优势,有效增强了算法的局部开发与全局探索能力,产生更优的候选解;IGWO 通过随机选择领导者更新跟随者位置,引入自适应参数,增强多样性和全局搜索能力,提高算法的灵活性;HI-WOA 引入杂交修正机制和非线性收敛因子,以解决 WOA 在复杂环境下的局部最优问题,提高收敛速度和精度。

最后,与近 5 年元启发式算法进行对比,包括 SOA^[20] (Seagull Optimization Algorithm, SOA)、HHO (Harris Hawks Optimization, HHO)^[21]、SCSO^[22] (Sand Cat Swarm Optimization, SCSO) 和 SSA (Sparrow Search Algorithm, SSA)^[23] 算法。SOA 是一种模拟海鸥迁徙和捕食行为的优化算法。海鸥擅长长距离飞行寻找食物,SOA 通过模拟其导航和飞行模式来寻找全局最优解。HHO 算法受到哈里斯鹰群体狩猎行为的启发,哈里斯鹰通过合作追逐猎物以将其围困,HHO 模拟该策略,通过追逐、伏击和攻击行为来探索

搜索空间,寻找最优解。SCSO 算法模仿沙漠猫在极端环境中寻找食物和水源的策略,沙漠猫能在严苛条件下生存,SCSO 借鉴其捕猎和社交行为来优化问题解。SSA 算法源于麻雀的觅食和反掠行为,麻雀在觅食时会表现出警戒

和随机搜索行为。SSA 通过模拟这些行为,将其觅食策略转化为优化算法。

算法对比结果如表 3—表 5 所示,最优结果用粗体标出。这些对比实验全面评估了 TCGWOWOA 算法的性能。

Table 3 Comparison between TCGWOWOA and classical algorithms
表 3 TCGWOWOA 与经典算法比较

基准 函数	TCGWOWOA		GWO		WOA		ACO	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
F1	9.63E+07	1.22E+07	2.2E+08	1.8E+08	2.4E+09	2.04+09	7.3E+09	1.2E+08
F2	2.03E+02	3.2E+01	2.1E+03	2.3E+03	2.28E+04	1.53E+04	7.17E+04	4.89E+03
F3	3.77E+01	1.9E+01	1.66E+02	5.9E+01	5.77E+02	2.32E+02	6.92E+02	1.08E+0
F4	4.03+02	3.8E+01	7.57E+02	1.66E+02	1.16E+04	4.99E+03	5.7E+03	8.80E+02
F5	6.1E-03	8.2E-03	1.05E-03	1.54E-03	1.57E-03	5.06E-03	2.52E-02	3.697E-03
F6	3E+03	1.1E+03	3.2E+04	3.58E+05	2.6E+05	4.78E+03	1.85E+05	1.98E+05
F7	3.67E+02	9.90E+02	3.53E+02	9.5E+02	7.2E-02	2.83E-01	5.62E+02	5.4E+01
F8	7.2E-01	7.1E-01	3.14E+00	1.68E+00	1.7E+01	6.42E+00	1.4E+00	1.9E+00
F9	2.8E+03	3.72E+03	3.24E+03	7.3E+02	-1.32E+02	7.04E+02	3.64E+03	5.51E+02
F10	9.09E+04	4.59E+04	1.39E+05	2.3E+04	8.85E+05	3.41E+04	2.04E+06	1.4E+06
F11	1.13E+07	4.2E+06	3.3E+07	2.7E+07	1.12E+06	7.39E+06	4.7E+08	1.11E+08
F12	5.1E+02	2.2E+02	1.42E+02	3.1E+02	3.54E+02	1.22E+07	5.52E+02	1.37E+02
F13	1.9E+05	5.17E+05	6.07E+05	3.49E+06	3.76E+05	6.91E+05	6.67E+06	3.3E+06

Table 4 Comparison between TCGWOWOA and existing improved algorithms
表 4 TCGWOWOA 与现有改进算法比较

基准 函数	TCGWOWOA		GWOWOA		IGWO		HI-WOA	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
F1	9.63E+07	1.22E+07	2.5E+09	1.66E+09	4.2E+08	4.71E+08	9.2E+09	4.8E+09
F2	2.03E+02	3.2E+01	7.55E+04	2.1E+03	2.1E+03	1.8E+03	4.06E+04	1.12E+04
F3	3.77E+01	1.9E+01	2.35E+02	6.2E+01	1.32E+02	7.3E+01	9.7E+02	5.74E+02
F4	4.03+02	3.8E+01	4.63E+03	3.11E+02	9.92E+02	3.4E+02	2.21E+04	6.94E+03
F5	6.1E-03	8.2E-03	4.59E-03	3.35E-03	1.75E-03	3.68E-03	6.98E-03	5.3E-03
F6	3E+03	1.1E+03	1.1E+05	1.01E+05	8.48E+03	6.9E+03	4.6E+05	3.59E+05
F7	3.67E+02	9.90E+02	3.84E-01	4.9E+01	3.1E+02	1.46E+02	7.57E-02	3.4E+01
F8	7.2E-01	7.1E-01	3.5E+00	6.6E-01	9.3E-01	6.6E-01	2.6E+01	9.78E+00
F9	2.8E+03	3.72E+03	2.6E+03	5.24E+02	1.4E+03	2.2E+03	-1.3E+03	7.86E+02
F10	9.09E+04	4.59E+04	1.79E+05	3.89E+04	1.14E+05	2.5E+04	1.19E+05	3.8E+05
F11	1.13E+07	4.2E+06	1.3E+08	2.70E+07	1.75E+07	1.0E+07	2.7E+07	2.0E+07
F12	5.1E+02	2.2E+02	2.3E+02	7.1E+01	1.25E+02	1.08E+02	8.32E+02	5.9E+02
F13	1.9E+05	5.17E+05	7.4E+05	4.7E+05	6.13E+05	4.2E+05	2.21E+06	2.1E+06

由表 3 可以看出,TCGWOWOA 算法在多个测试函数上相较于传统的 GWO、WOA 和 ACO 算法表现更为优异。具体而言,在差距最大的 F6 函数上,TCGWOWOA 相较于 GWO 和 WOA 提升了一个量级,而在差距最小的 F9 函数上提升了 14%。相较于 ACO,其尽管在 F7 函数上提升了 31%,但在其他函数上的提升更加显著,达到多个量级,表明 TCGWOWOA 在全局搜索和局部勘探方面具有更好的平衡性能,可有效避免陷入局部最优解。

由表 4 可以看出,相较于 GWOWOA,TCGWOWOA 在 F1 和 F4 函数上提升了一个量级,在 F6 函数上提升了两个量级。相较于 HI-WOA,TCGWOWOA 在收敛精度、稳定性和收敛速度等多项性能指标上均显著优于对比算法,特别是在 F10 函数上提升了 26%。虽然在 F7、F9 和 F12 函数上

TCGWOWOA 的平均值略低于 IGWO,但实际效果也相当接近。除个别函数外,TCGWOWOA 的表现都十分优异。

由表 5 可以看出,TCGWOWOA 相较于近 5 年元启发式算法性能也有显著提升,证明其在优化问题上的竞争力,且具有较强的鲁棒性。混合算法的设计使其在全局搜索和局部优化方面更加平衡,在解决复杂问题时更加高效。

此外,TCGWOWOA 通过引入参数自适应调整机制,显著增强了对复杂函数及多样化问题场景的适应能力,展现出优异的泛化性能。实验结果表明,该算法在全局探索与局部开发能力上均优于对比算法,尤其适用于复杂的优化问题。

综上所述,TCGWOWOA 融合了混合改进的设计理念,不仅在当前主流算法中具备明显的竞争力,而且为求解动

Table 5 Comparison between TCGWOWOA and meta-heuristic algorithms in the past five years

表5 TCGWOWOA 与近5年元启发式算法比较

基准函数	TCGWOWOA		SOA		HHO		SCSO		SSA	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
F1	9.63E+07	1.22E+07	6.4E+10	3.34E+09	1.7E+09	1.51E+09	4.9E+09	4.59E+09	1.20E+08	8.2E+07
F2	2.03E+02	3.2E+01	1.9E+05	7.74E+04	1.86E+03	4.06E+03	1.9E+05	5.7E+03	2.8E+05	5.06E+05
F3	3.77E+01	1.9E+01	1.81E+04	2.8E+03	3.15E+02	1.13E+02	5.98E+02	4.60E+02	6.35E+02	8.29E+02
F4	4.03+02	3.8E+01	8.2E+05	8.6E+03	4.8E+03	2.71E+03	3.60E+03	2.48E+03	1.16E+04	3.9E+03
F5	6.1E-03	8.2E-03	5.3E-02	1.06E-02	7.2E-03	5.8E-03	2.98E-03	1.44E-03	1.91E-03	1.39E-03
F6	3E+03	1.1E+03	3.04E+06	3.34E+05	9.2E+05	4.28E+05	2.648E+05	2.17E+05	2.78E+05	8.23E+04
F7	3.67E-02	9.90E+02	9.84E+01	5.0E+00	5.18E+02	8.1E+02	4.64E+02	5.08E+01	1.25E+01	2.74E+01
F8	7.2E-01	7.1E-01	4.7E+01	1.21E+01	1.29E+01	2.75E+00	1.5E+01	2.82E+00	1.37E+01	4.08E+00
F9	2.8E+03	3.72E+03	3.56E+03	1.46E+03	2.61E+02	1.28E+03	-9.67E+02	7.92E+02	-2/3E+03	1.12E+03
F10	9.09E+04	4.59E+04	2.6E+08	2E+08	1.77E+05	4.88E+05	1.48E+05	1.50E+04	1.8E+05	1.93E+05
F11	1.13E+07	4.2E+06	1.3E+10	4.31E+09	7.2E+07	5.50E+07	5.5E+07	3.2E+07	2.3E+08	2.88E+08
F12	5.1E+02	2.2E+02	1.67E+06	4.20E+03	6.48E+02	3.09E+02	5.51E+02	3.2E+02	4.79E+03	5.4E+03
F13	1.9E+05	5.17E+05	3.2E+07	2.4E+07	3.82E+05	1.68E+05	1.11E+06	1.8E+06	3.66E+06	2.7E+06

态复杂的优化问题提供了创新且高效的解决方案。

Table 9 Task parameters

表9 任务参数信息

名称	配置属性	大小/范围	单位
CloudLet	Mi	500~1 500	KB
	Number	50~300	个
	InputFileSize	20~200	KB
	OutFileSize	20~200	KB

6 云计算实验与分析

6.1 云计算实验配置

本文使用澳大利亚墨尔本大学网格实验室开发的 CloudSim 模拟工具进行实验。CloudSim 是一种广泛应用于云计算研究领域的模拟平台,能够有效模拟云计算环境中的资源管理、任务调度和虚拟机管理等场景^[24]。相关实验配置如表6—表9所示。

Table 6 Cloud computing experiment configuration environment

表6 云计算实验配置环境

实验环境	配置属性
CPU	Intel(R)Core(TM)i7-7700HQ
内存	16G
硬盘	1TB
开发工具	IDEAJ
JDK	JDK1.8
操作系统	Window 10 64 位
仿真平台版本	Cloudsim4

Table 7 Virtual machine parameters

表7 虚拟机参数信息

名称	配置属性	大小/范围	单位
VM	Number	10	个
	Ram	512	Ms
	Bandwidth	50	GB
	Mips	50~1 000	bps

Table 8 Data center parameters

表8 数据中心参数信息

名称	配置属性	大小/范围	单位
DataCenter	Number	1	个
	Ram	16 384	GB
	Storage	1 000 000	GB
	Mips	100 000	Ms
	Bandwidth	10 000	bps

6.2 实验结果与分析

本文通过仿真实验测试云平台在不同任务数量与虚拟机规模下的性能表现。实验设置云任务数量分别为50、100、150、200、250和300,虚拟机数量分别为4、8和12。仿真实验结果取相同条件下的10次平均值,并与基准算法进行比较。结果表明,随着云任务数量的增加和虚拟机数量的变化,TCGWOWOA算法的表现均优于基准算法。

不同算法的任务执行时间比较如表10所示。

Table 10 Task execution time comparison of different algorithms

表10 不同算法任务执行时间比较 (ms)

虚拟机个数	算法	云任务个数					
		50	100	150	200	250	300
4	GWO	28.28	53.34	85.78	115.82	148.87	172.14
	IGWO	23.02	46.36	70.72	92.75	117.38	141.79
	WOA	20.80	41.84	62.14	82.56	104.40	126.27
	GWOWOA	19.68	38.07	59.16	79.35	101.27	122.28
	TCGWOWOA	17.16	30.05	51.06	71.89	92.47	113.27
	GWO	48.52	104.58	162.64	222.96	283.44	346.65
8	IGWO	33.43	85.39	138.24	189.74	244.86	299.48
	WOA	33.58	50.12	76.03	107.43	128.92	208.88
	GWOWOA	31.06	49.96	72.39	96.85	122.48	195.55
	TCGWOWOA	28.55	42.57	63.44	85.08	108.86	168.38
12	GWO	53.24	131.94	217.60	301.65	392.47	478.07
	IGWO	56.68	122.16	196.44	264.94	339.98	413.83
	WOA	58.40	137.09	223.62	329.54	399.96	483.49
	GWOWOA	57.18	136.39	216.03	318.44	394.79	481.83
	TCGWOWOA	48.04	118.50	200.23	281.09	358.35	433.91

由表 10 可知, 本文算法在任务执行中消耗的时间最短。具体来说, 当虚拟机数量为 4 时, 相较于 GWO、IGWO、WOA 和 GWOWOA 算法, 本文算法的平均执行时间分别降低了 37.78%、23.60%、14.18% 和 10.45%。当虚拟机数量增至 8 时, 降低比例进一步提升至 57.48%、15.19%、17.86% 和 12.56%。当虚拟机数量为 12 时, 平均执行时间分别降低了 8.56%、-3.33%、11.73% 和 10.37%, 虽不如 IGWO 的表现, 但结果相对接近。

不同算法的任务最大完成时间比较如表 11 所示。由

Table 11 Maximum completion time of task comparison of different algorithms

表 11 不同算法任务最大完成时间比较 (ms)		云任务个数					
虚拟机 个数	算法	50	100	150	200	250	300
		50	100	150	200	250	300
4	GWO	34.31	47.89	52.03	56.05	61.04	68.83
	IGWO	30.12	35.68	47.1	47.01	49.42	62.29
	WOA	33.37	48.83	58.48	62.04	71.85	71.41
	GWOWOA	30.39	39.17	56.23	59.27	70.09	65.28
	TCGWOWOA	27.56	32.16	46.85	50.93	49.21	58.72
8	GWO	33.93	36.11	38.99	46.66	49.81	54.74
	IGWO	29.83	33.05	36.85	35.98	39.15	43.67
	WOA	37.55	36.52	39.17	41.28	46.16	48.79
	GWOWOA	27.79	31.96	35.24	38.43	43.0	46.15
	TCGWOWOA	23.73	27.23	33.07	34.52	38.49	41.55
12	GWO	20.98	21.58	23.89	27.15	34.15	38.34
	IGWO	18.35	18.77	20.18	25.77	32.81	34.21
	WOA	19.52	20.44	21.73	26.37	32.7	36.23
	GWOWOA	18.11	18.67	20.78	24.36	30.72	35.42
	TCGWOWOA	17.67	16.48	18.59	22.57	26.82	29.33

表 11 可知, 本文算法在最大完成时间上表现最优。当虚拟机数量为 4 时, TCGWOWOA 相较于 GWO、IGWO、WOA 和 GWOWOA, 最大完成时间平均降低了 17.10%、2.27%、23.48% 和 17.16%。当虚拟机数量为 8 时, 平均降低 23.68%、6.99%、20.39% 和 10.77%。当虚拟机数量为 12 时, 平均降低 20.85%、11.82%、16.26% 和 11.80%。

不同算法的任务成本比较如表 12 所示。由表 12 可知, 本文算法在任务成本上表现最优。当虚拟机数量为 4 时, TCGWOWOA 相较于 GWO、IGWO、WOA 和 GWOWOA, 成本平均降低了 6.05%、2.34%、12.18% 和 10.06%。当虚拟机数量为 8 时, 平均降低了 7.1%、2.36%、10.86% 和 9.90%。当虚拟机数量为 12 时, 平均降低了 6.14%、3.21%、6.55% 和 3.79%。

综上, TCGWOWOA 算法在任务调度效率、成本和虚拟机资源利用率方面表现优异。通过结合 GWO 的全局搜索能力和 WOA 的局部优化能力, TCGWOWOA 实现了全局搜索与局部优化的平衡, 显著提升了寻优和收敛性能。具体而言, TCGWOWOA 结合了 GWO 的全局搜索和 WOA 的局部搜索, 避免陷入局部最优。引入的 Tent 初始化方法可生成均匀分布的初始种群, 增强了全局搜索能力, 并通过非

线性收敛因子和柯西变异操作提高了收敛速度和精度。

Table 12 Task costs comparison of different algorithms

表 12 不同算法任务成本比较 (元)

虚拟机 个数	算法	云任务个数					
		50	100	150	200	250	300
4	GWO	110.16	200.41	338.24	463.88	589.62	693.19
	IGWO	96.01	193.16	329.63	448.44	563.39	674.79
	WOA	133.67	263.76	383.15	494.31	598.92	688.93
	GWOWOA	121.63	220.88	353.2	484.24	590.44	681.84
	TCGWOWOA	94.39	184.90	317.63	452.69	547.50	653.31
8	GWO	118.97	243.76	407.62	557.41	704.51	832.66
	IGWO	89.02	227.23	372.68	533.28	687.93	815.37
	WOA	142.89	276.16	428.59	583.17	702.03	852.63
	GWOWOA	136.29	265.12	421.72	569.0	690.82	842.84
	TCGWOWOA	75.98	219.70	349.84	528.86	681.4	805.24
12	GWO	101.43	222.95	364.65	516.02	644.53	771.36
	IGWO	89.75	217.54	363.29	490.02	630.65	750.44
	WOA	104.72	232.2	363.55	514.05	649.08	769.04
	GWOWOA	93.23	220.57	355.61	499.97	631.83	755.94
	TCGWOWOA	81.41	202.71	336.44	482.35	616.2	740.88

7 结语

为解决 GWOWOA 算法在处理复杂问题时收敛速度慢、易陷入局部最优的问题, 本文提出 TCGWOWOA 混合优化算法。TCGWOWOA 通过引入 Tent 映射增强初始化种群的多样性, 改进非线性收敛因子以平衡全局与局部搜索能力, 并利用柯西分布扰动防止陷入局部最优。实验结果表明, TCGWOWOA 在 CEC2017 基准函数上表现出更高的求解精度和更强的鲁棒性, 优于 GWOWOA 等其他算法。在云计算中, TCGWOWOA 显著提升了任务调度效率, 降低了成本, 提高了虚拟机资源利用率。

未来计划将 TCGWOWOA 与深度学习方法相结合, 以解决高维大规模数据和复杂模型参数优化问题, 提高算法在深度学习任务上的性能。同时, 考虑将算法扩展应用于多模态问题, 深化理论研究并拓展其应用领域, 使其成为解决实际优化问题高效、可靠的途径。

参考文献:

[1] LI Q, ZHENG X. Overview of cloud computing research status[J]. Computer Science, 2011, 38(4): 32-37.
李乔, 郑啸. 云计算研究现状综述[J]. 计算机科学, 2011, 38(4): 32-37.

[2] JUN Z, JIN J H, SONG A B, et al. Cloud computing: architecture and key technologies[J]. Journal of Communications, 2011, 32(7): 3-21.
军舟, 金嘉晖, 宋爱波, 等. 云计算: 体系架构与关键技术[J]. 通信学报, 2011, 32(7): 3-21.

[3] LAI H, LIAO H, ŠAPARAUSKAS J, et al. Sustainable cloud service provider development by a Z-number-based DNMA method with Gini-coeffi-

- cient-based weight determination [J]. Sustainability, 2020, 12 (8): 3410.
- [4] GARG D, KUMAR P. A survey on metaheuristic approaches and its evaluation for load balancing in cloud computing [C]//Proceedings of Advanced Informatics for Computing Research, 2019: 585–599.
- [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] OBADINA O O, THAHA M A, ALTHOEFER K, et al. Dynamic characterization of a master-slave robotic manipulator using a hybrid grey wolf-whale optimization algorithm [J]. Journal of Vibration and Control, 2022, 28: 1992–2003.
- [7] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69: 46–61.
- [8] MIRJALILI S, LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51–67.
- [9] REN J X, DU Z Z, WANG X K. Cloud task scheduling based on improved artificial bee colony algorithm [J]. Journal of Henan University of Science and Technology (Natural Science), 2022, 43(4): 55–60.
任金霞, 杜增正, 王兴康. 改进人工蜂群算法的云任务调度 [J]. 河南科技大学学报(自然科学版), 2022, 43(4): 55–60.
- [10] CHEN J R, GUO Y J. Cloud computing task scheduling based on commutative improved particle swarm optimization [J]. Computer Applications & Software, 2023, 40(6): 223–228.
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.
- [11] CHEN X R, ZENG B Q. Cloud computing multi-objective task scheduling with improved squirrel search algorithm [J]. Computer Engineering and Design, 2022, 43(7): 1990–1997.
陈孝如, 曾碧卿. 改进松鼠搜索算法的云计算多目标任务调度 [J]. 计算机工程与设计, 2022, 43(7): 1990–1997.
- [12] MENG Q Y, WANG J J. Task scheduling optimization in cloud computing based on hybrid firefly genetic algorithm [J]. Microcomputer Applications, 2021, 37(5): 158–160, 164.
孟庆岩, 王晶晶. 基于混合萤火虫遗传算法的云计算中的任务调度优化 [J]. 微型电脑应用, 2021, 37(5): 158–160, 164.
- [13] YANG X G, ZHANG Q S, ZHANG Y M, et al. Research on cloud computing task scheduling problem based on improved hybrid particle swarm algorithm [J]. Journal of Inner Mongolia University (Natural Science Edition), 2019, 50(6): 674–678.
杨晓光, 张奇松, 张益民, 等. 基于改进混合粒子群算法的云计算任务调度问题研究 [J]. 内蒙古大学学报(自然科学版), 2019, 50(6): 674–678.
- [14] ZHANG Y, CHEN F. An improved whale optimization algorithm [J]. Computer Engineering, 2018, 44(3): 208–213.
张永, 陈锋. 一种改进的鲸鱼优化算法 [J]. 计算机工程, 2018, 44(3): 208–213.
- [15] WANG L D, WANG X C, LIU G. An improved algorithm for convergence factor nonlinear decreasing multi-objective whale optimization based on angular penalty distance [J]. Computer Application Research, 2022, 39(5): 1395–1401.
王龙达, 王兴成, 刘罡. 基于角度惩罚距离的收敛因子非线性递减多目标鲸鱼优化改进算法 [J]. 计算机应用研究, 2022, 39(5): 1395–1401.
- [16] VAN T N, MIRJALILI S. MEALPY: an open-source library for latest meta-heuristic algorithms in Python [J]. Journal of Systems Architecture, 2023, 139: 102871.
- [17] SOCHA K, DORIGO M. Ant colony optimization for continuous domains [J]. European Journal of Operational Research, 2008, 185(3): 1155–1173.
- [18] KAVEH A, ZAKIAN P. Improved GWO algorithm for optimal design of truss structures [J]. Engineering with Computers, 2018, 34: 685–707.
- [19] TANG C, SUN W, WU W, et al. A hybrid improved whale optimization algorithm [C]//Proceedings of 2019 IEEE 15th International Conference on Control and Automation, 2019: 362–367.
- [20] DHIMAN G, KUMAR V. Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems [J]. Knowledge-based Systems, 2019, 165: 169–196.
- [21] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: algorithm and applications [J]. Future Generation Computer Systems, 2019, 97: 849–872.
- [22] SEYYEDABBASI A, KIANI F. Sand cat swarm optimization: a nature-inspired algorithm to solve global optimization problems [J]. Engineering with Computers, 2023, 39(4): 2627–2651.
- [23] XUE J, SHEN B. A novel swarm intelligence optimization approach: sparrow search algorithm [J]. Systems Science & Control Engineering, 2020, 8(1): 22–34.
- [24] BUYYA R, RANJAN R, CALHEIROS R N. Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities [C]//Proceedings of 2009 International Conference on High Performance Computing & Simulation, 2009: 1–11.

(责任编辑:黄健)