文章编号:1671-4598(2026)10-0208-08 DOI:10.16526/j.cnki.11-4762/tp.2026.10.027 中图分类号:TP18 文献标识码:A

多机器人任务调度的簇引导自适应麻雀算法

张 强[1], 李金铭[1], 杨小槐[2]
(1. 东北石油大学 计算机与信息技术学院, 黑龙江 大庆 163318;
2. 大庆油田勘探开发研究院 应用软件研究室, 黑龙江 大庆 163513)

摘要: 云计算环境下的多机器人任务需在就绪时间、截止时间、执行时长、优先级与CPU异构等多重约束下实现高效调度; 为提升任务分配的可行性与资源利用效率, 构建了一种簇引导的自适应麻雀搜索算法, 通过引入任务特征聚类、选择性反向学习与分级混合变异机制实现全局探索与局部精化的协同优化; 在多规模数据集上的实验表明, 该算法在收敛速度、任务完成率、按时完成率与完工时间等指标上均优于多种经典优化方法, 并在大规模场景中表现出稳定性与可扩展性; 结果验证了所设计方法能够有效降低跨簇任务分散导致的冲突代价, 适用于复杂、异构与大规模的多机器人任务调度应用。

关键词: 机器人调度; 麻雀搜索算法; Logistics映射; t分布扰动; 智能群体; 多机器人系统; 人工智能

Cluster-Guided Adaptive Sparrow Algorithm for Multi-Robot Task Scheduling

ZHANG Qiang[1], LI Jinming[1], YANG Xiaohuai[2]
(1. School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China;
2. Software Research Laboratory, Research Institute of Exploration and Development of Daqing Oilfield Co., Ltd., Daqing 163513, China)

Abstract: In cloud computing environments, the multi-robot task scheduling requires an efficient collocation under multiple constraints such as ready time, deadline, execution duration, priority, and CPU heterogeneity. To improve task allocation feasibility and resource utilization, a cluster-guided adaptive sparrow search algorithm (CG-SSA) is developed, which integrates task feature clustering, selective opposition-based learning, and hierarchical hybrid mutation mechanisms, achieving the collaborative optimization of global search and local refinement. Experiments on muti-scale datasets show that the proposed method outperforms several classical optimization algorithms in the convergence speed, task completion rate, on-time completion rate, and makespan, with a strong stability and scalability in large-scale scenarios. The results demonstrate that, the proposed approach effectively reduces the conflict caused by cross-cluster task dispersion, and is suitable for complex, heterogeneous, and large-scale multi-root task scheduling applications.

Keywords: multi-robot scheduling; SSA; logistics mapping; t-distribution perturbation; intelligent swarm; multi-robot system; artificial intelligence

0 引言

多机器人任务调度[1]作为自动化流程控制、智能仿真、计算机模拟等领域[2-3]的重要优化问题, 其目标是在有限的计算资源和时限约束下, 合理调度虚拟机器人完成预定义操作流程, 实现任务按时、高效、低冲突地

收稿日期: 2025-10-20; **修回日期:** 2025-11-25。
基金项目: 国家自然科学基金项目(42002138); 黑龙江省优秀青年教师基础研究支持计划项目(YQJH2023073); 黑龙江省自然科学基金项目(LH2024F005); 黑龙江省研究生课程思政建设项目计算智能及石油大数据分析导学思政团队(YJSKCSZ_202303)。
作者简介: 张 强(1982-), 男, 博士, 教授。
通讯作者: 李金铭(1998-), 女, 硕士研究生。
引用格式: 张 强, 李金铭, 杨小槐, 等. 多机器人任务调度的簇引导自适应麻雀算法[J]. 计算机测量与控制, 2026, 34(10): 208-215, 250.

完成。随着系统复杂度提升，任务调度问题呈现出高维度、多约束、任务依赖复杂、动态变化等特点，属于典型的 NP 难组合优化问题[4-5]，解空间规模会随着任务和资源数量呈指数级增长。

使用枚举或完全搜索在任务量稍大时就会导致计算时间和存储开销急剧增大，难以在可接受的时间范围内产生调度方案。传统集中式调度方法[6-8]尽管具有全局控制与集中优化的理论优势，但在实际调度环境中暴露出诸多不足。尤其在任务具有动态依赖关系、有向无环图结构、资源负载波动等复杂情形下，集中式方法往往依赖静态优先级或预设规则，缺乏响应调度异常与资源失衡的动态调节能力。难以应对高维度、多约束及动态依赖的调度环境；因此研究者们常采用如遗传算法[9]、粒子群算法[10]蚁群算法[11]、鲸鱼优化算法[12]等，以较低的计算代价寻找到高质量甚至近似最优的调度解。除群体智能算法外，近年来深度学习与多智能体强化学习（MARL，multi-agent reinforcement learning）等方法也逐渐应用于复杂任务调度场景。例如，深度强化学习模型通过策略网络直接学习任务分配或资源调度策略，能够根据系统状态实时做出决策；图神经网络结构则可用于建模具有 DAG 特性的任务依赖关系，学习任务拓扑的潜在结构特征；MARL 方法则通过多个智能体协作完成任务分配，在动态环境中具备一定的自适应优势。然而，此类方法通常面临模型训练成本高、样本需求量大、泛化能力易受任务分布变化影响等限制，且在多机器人大规模异构调度场景下缺乏可解释性和轻量化优势。因此，在需要保证低计算开销、可复制性与稳定性的调度任务中，基于群智能算法的优化方法仍然具备重要研究价值。文献［13］引入反向学习到混沌映射 PSO，提升了初代种群多样性；文献［14］将 DDNQ 与改进 WOA 结合，在运行时实时感知资源负载，显著提高了计算资源的利用率和系统吞吐性能；文献［15］提出了一种两阶段在线调度框架，可以在动态任务环境下自适应调整资源分配和执行顺序；文献［16］针对静态有向无环图任务调度场景，对 WOA 算法进行了改进，引入了任务依赖惩罚和资源负载均衡项，提高了任务完成率和资源利用率。

麻雀搜索算法[17]是一种新兴的智能群体优化算法，以麻雀群体觅食与警戒行为为启发，具有结构简洁、参数较少、收敛速度快、全局搜索能力强的优势，特别适用于高维度、多约束的复杂优化问题。SSA 通过发现者、加入者和警戒者的协作机制，平衡了全局探索与局部开发能力，在优化调度效率和系统稳定性方面具有较大潜力。近年来有很多人就此展开研究，文献［18］采用 sobel 机制和正余弦扰动等策略改进麻雀搜索算法，提高了收敛效果和搜索能力；文献［19］通过结合天鹰优化算法搜索策略与柯西变异策略完成了对麻雀搜索算法的改进，提升了算法的局部搜索能力；文献［20］加入非线性变化权重因子以避免算法进入局部收敛，对比分析结果表明所提改进麻雀搜索算法具有寻优精度高的优点，可有效降低机场微电网运行成本。为克服传统麻雀搜索算法在高维度、多约束、多依赖任务调度场景中的局限性，本文在 SSA 原有框架基础上设计了具有结构化调度感知能力的簇引导自适应搜索机制。与标准 SSA 以及其他改进策略相比，CG-SSA（Cluster-Guided Adaptive Sparrow Search Algorithm）的核心创新主要体现在：引入选择性反向学习策略，通过对原解及其对立解的适应度差进行概率筛选，显著提升初始种群质量并增强搜索起点的多样性；通过 K-Means 对任务特征进行分簇，并在目标函数中加入簇分散度惩罚，使算法能够学习任务在簇内集中调度的内在结构规律，从而降低跨簇调度带来的资源冲突风险；依据不同任务簇的重要性采用差异化变异策略，对关键簇施加 Logistic 混沌跳跃以加强全局搜索，对非关键簇结合 t 分布扰动和最优片段复制实现局部精化与随机跳脱平衡。以上机制共同提升了算法的全局探索能力与局部收敛效率，为提高调度方案的可行性与整体性能提供了结构化支持。在此基础上，本文提出一种簇引导混合搜索框架。在种群初始化时，利用选择性反向学习策略，对每个随机解同时生成反向解，动态概率选择优质对立解，从而提高初始种群质量和多样性。迭代更新时，对发现者仅在簇 0 任务维度上施加 Logistic 混沌映射，实现大范围全局跳跃；对加入者在簇 1 和簇 2 维度上施加 t 分布扰动，结合最优解片段的随机复制，在局部精化的同时偶发大步跳跃，从而提高跳出局部最优的能力。成本函数中加入簇分散惩罚项，使得相同簇的任务尽量集中调度，降低资源冲突，仿真实验结果表明，CG-SSA 算法的收敛速度及调度效果相较于传统的 SSA 均有所提高，能为大规模异构任务调度提供更优的优化解决方案。

1　多机器人任务调度模型

1.1　问题描述

多机器人任务调度问题属于 NP 难的组合优化问题，可以描述为：在给定的异构机器人集群与任务集下，每个任务具有已知的就绪时间、截止时间、执行时长、优先级、依赖关系和资源需求如 CPU 占用率等，机器人之间在计算能力和可用时间上存在差异。在本文的场景中，给每个任务设定了就绪时间和截止时间即任务最早开始时间和任务的最晚结束时间，执行任务所需时间和 CPU 占用率等资源需要指标。在模型中，任务调度需要根据任务的优先级权重进行动态决策，以确保高优先级任务的优先完成；任何时刻，分配给同一机器

人的任务 CPU 占有量不得超过机器人的最大容量，这样才能满足资源约束要求。由于任务与任务之间还存在着有向无环的先行依赖关系，后继任务必须在所有前驱任务完成之后才能进行工作，这要求调度方案严格遵循依赖拓扑，并按时间先后顺序依次安排任务执行。

在多机器人协同作业场景中，系统通常需要在有限资源条件下完成一组具有先后依赖关系、资源消耗特征和优先级属性的复杂任务。典型任务包括数据采集与处理、巡检任务、装配流程中多步骤加工任务或智能仿真平台中的虚拟任务执行，与实际生产流程中的流水线调度、批处理任务调度等具有高度一致性。不同任务在执行时间、CPU 负载、优先级、依赖拓扑结构等方面存在显著差异，且任务之间通常具备有向无环图形式的前后约束，违反依赖关系会直接影响调度的可行性。

1.2 数学建模

1.2.1 符号说明

文中的符号说明如表 1 所示。

表 1 符号及其含义

符号	描述
$T=\{T_1,T_2,\cdots,T_n\}$	任务集合
$R=\{R_1,R_2,\cdots,R_m\}$	机器人集合
r_i	就绪时间(任务最早开始时间)
d_i	截止时间(任务最晚结束时间)
p_i	任务执行所需要时间
c_i	任务的 CPU 占用率
w_i	任务优先级权重
$x_{i,j}\in\{0,1\}$	若任务 i 分配给机器人 j，则 $x\{i,j\}=1$，否则 $x\{i,j\}=0$
s_i	开始时间，满足
f_i	完成时间满足 $f_i=s_i+p_i$
$C=\{C_1,C_2,\cdots,C_p\}$	任务簇划分
pred_i	任务 T_i 的前驱任务集
$\omega_c=\frac{1}{\lvert C_c\rvert}$	簇归一化权重

1.2.2 目标函数

目标函数用于衡量整个调度方案的优劣，其值由完工时间、任务完成率和未完成任务数等多项指标综合而成：

$$cost = \mathrm{Makespan} - \alpha \cdot CR + \gamma \cdot U + \lambda \sum_{c=0}^{K-1} \omega_c (\lvert R_c \rvert - 1) \tag{1}$$

其中：U 在任务调度方案确定后，没有被分配给机器人的任务，或由于其他原因没有完成的任务数量；$\lvert R_c \rvert$ 为簇 c 被分配的机器人台数；ω_c 为簇 c 的规模归一化权重；α、γ、λ 为调节权重和惩罚系数。

CR 是任务完成率，定义为完成并按时完成的任务数占总任务的比例：

$$CR = \frac{1}{n} \sum_{i=1}^{n} 1_{f_i \leqslant d_i} \tag{2}$$

OTR 是指按时完成率，按时完成任务数与已完成任务数之比，量化了执行效率与时间管理的有效性：

$$OTR = \frac{\sum_i 1_{f_i \leqslant d_i}}{\sum_i 1_{f_i < \infty}} \tag{3}$$

Makespan 是所有任务完成的最晚时间：

$$\mathrm{Makespan} = \max_{1 \leqslant i \leqslant n} f_i \tag{4}$$

ND 是归一化延迟指标，仅靠按时完成率无法区分“刚好超期一点”和“大幅超期”两种情况，所以引入延迟风险指标，其中 $\mathcal{L} = \{i \mid f_i > d_i\}$ 为延迟任务集：

$$ND = \frac{1}{1 + \frac{1}{\lvert \mathcal{L} \rvert} \sum_{i \in \mathcal{L}} \frac{f_i - d_i}{p_i}} \tag{5}$$

1.2.3 约束条件

在建构的多机器人任务调度模型中，核心限制条件主要包含以下几点：

1）先行约束：任务之间按照有向无环图（DAG）定义先行关系。如果任务 T_i 存在依赖 pred_i，那么在该任务的所有前驱任务完成之前，不能开始进行该任务。若任务 T_k 是任务 T_i 的前驱任务（$T_k \in \mathrm{pred}_i$），则其完成的时间不能晚于 T_i 的开始时间。表达式为：

$$f_k \leqslant s_i, \forall\, T_i \in T, \forall\, T_k \in \mathrm{pred}_i \tag{6}$$

2）CPU 资源约束：对于任意机器人 j，在任意时刻其所执行任务的 CPU 占有总量不得超过最大容量 $C_j^{\max}$。表达式为：

$$\sum_{i=1}^{n} x_{i,j} c_i \leqslant C_j^{\max}, \forall\, j \in R \tag{7}$$

3）时间连续性和就绪时间约束：每个任务 T_i 只有在自身就绪时间 r_i 之后才能开始执行任务；机器人一旦开始执行某项任务，必须在这个任务结束后才能继续执行其他任务；若有多个任务可以被调度，需要按照任务的优先级权重进行分配。表达式为：

$$s_i \geqslant r_i, f_i = s_i + p_i, f_i \leqslant d_i, \forall\, T_i \in T \tag{8}$$

4）截止时间与延迟惩罚约束：任务具有时效性，任务可以在 d_i 之后完成，但是会被视为延迟完成，并按延迟惩罚计入目标函数，会在后续的成功率计算及延迟惩罚中体现。

2 麻雀搜索算法

麻雀搜索算法[21]是一种模拟麻雀觅食和警戒行为的群体智能优化方法[22]，由 Xue 和 Shen 于 2020 年提出，近年来在智能优化领域获得广泛关注。该算法受麻雀种群的生存机制即觅食行为和反捕行为启发，文中的种群构成包括发现者、加入者和警戒者，该算法在全局优化能力、架构简洁性、收敛效率及参数灵活性等方面

均表现优异。

2.1　发现者

在麻雀搜索算法中存在着 3 种角色，分别是发现者、加入者和警戒者。在群体觅食的时候，发现者的适应能力比较强，会率先发现食物资源。发现者肩负着两方面的职责，一方面是为整个种群寻找食物来源，另一方面又要为加入者给予提示方向。由此可以得知，发现者在空间探索范围上具有显著的优势。根据文献［17］中的规则（1）和规则（2），在每次迭代的过程中，发现者的位置更新描述如下：

$$X_{i,j}^{t+1}=\begin{cases}X_{i,j}^{t}\cdot\exp\left(-\dfrac{i}{\alpha\cdot iter_{\max}}\right) & \text{if}\quad R_2<ST\\ X_{i,j}^{t}+Q\cdot\boldsymbol{L} & \text{if}\quad R_2\geqslant ST\end{cases}\tag{9}$$

式中，t 是算法当前运行次数，$j=1,2,3,\cdots,d$。常数代表的是最大的执行次数。是第 i 个个体在 j 维空间中的坐标参数。α 是属于 0～1 的随机数。R_2 是环境风险阈值，ST 是安全临界值，服从正态分布的扰动因子 Q 是一个随机数。$\boldsymbol{L}$ 是每个元素为 1 的矩阵。

2.2　加入者

群体中的加入者在资源探索的过程中，有一部分的加入者会保持监测机制，监测发现者。当加入者坐标区域出现更高能效资源点时，监测到优质资源的个体集群同步执行位置清空操作，并争夺发现者的资源。如果在争夺的过程中获得胜利，便会获得发现者的食物，失败就会继续在这个发现者的周围进行觅食。加入者的位置更新公式如（10）所示：

$$X_{i,j}^{t+1}=\begin{cases}Q\cdot\exp\left(\dfrac{X_{\text{worst}}-X_{i,j}^{t}}{i^2}\right) & \text{if}\quad i>n/2\\ X_{p}^{t+1}+\left|X_{i,j}^{t}-X_{P}^{t+1}\right|\cdot\boldsymbol{A}^{+}\cdot\boldsymbol{L} & \text{otherwise}\end{cases}\tag{10}$$

其中：X_p 是当前迭代周期内主导个体的最优空间定位，X_{worst} 则表示对应全局能效最低坐标点。$\boldsymbol{A}$ 表示 $1\times d$ 的矩阵，其中每个元素随机赋值为 1 或－1，并且 $\boldsymbol{A}^{+}=\boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{-1}$。当 $i>n/2$ 时，这表明，当适应性较弱的个体 i 未能获取资源，陷入食物匮乏的状态时，需要找其他地方进行寻找资源，来获得更多的能量。

2.3　警戒者

对于警戒者，警戒者主要是对觅食区域的安全进行监控。如果警戒者检测到潜在的危险或是有捕食者靠近，会发出报警信号，并且和发现者一起迁移到新的搜索范围，这样可以避免很多风险，可以继续进行觅食搜索。警戒者位置更新描述如下：

$$X_{i,j}^{t+1}=\begin{cases}\boldsymbol{X}_{\text{best}}^{t}+\beta\cdot\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{\text{best}}^{t}\right| & \text{if}\quad f_i>f_g\\ \boldsymbol{X}_{i,j}^{t}+K\cdot\left(\dfrac{\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{\text{worst}}^{t}\right|}{(f_i-f_w)+\varepsilon}\right) & \text{if}\quad f_i=f_g\end{cases}\tag{11}$$

其中：$\boldsymbol{X}_{\text{best}}$ 是周期内所寻获的最佳解向量。β 是为了精确控制步长，服从均值为 0，方差为 1 的标准正态分布。K 是从－1～＋1 均匀分布中随机选取的元素，f_i 代表了特定个体在群体中的适应度指标。f_g 和 f_w 分别为全局最优和全局最劣解的性能指标。ε 是最极端的常数值，用于防止除零错误。

3　簇引导自适应麻雀搜索算法（CG-SSA）

3.1　选择性反向学习策略

为提升初代种群质量和多样性，本文提出一种选择性反向学习策略（SOBL，selective opposition-based learning），在种群初始化阶段，对于每个随机生成的解向量 x，同步构造其对立解 x'，并通过两者在目标函数上的适应度差异计算保留对立解的概率。不仅能够动态平衡全局探索与局部开发，减少对劣质解的无效评估，还能显著提升初代种群的多样性和整体质量。传统的反向学习策略[23-24]对于任意一个随机生成的解向量 x，同步构造其在解空间中的对称映射点 x'，映射规律为：

$$x'_i=(M-1)-x_i,i=1,2,\cdots,n\tag{12}$$

其中：M 表示可以分配的机器人数量，n 表示任务总数。这样每产生一个原始解，就能在相反方向上补充一个“镜像”候选，从而快速扩大初始种群的搜索覆盖面。在此基础上，进一步融合了选择性反向学习策略机制，对每一对（x，x'）分别计算其在目标函数上的代价值 $f(x)$ 与 $f(x')$，并通过式（13）衡量方向解的优势程度。随后通过式（14）将这一定量映射为选用反向解的概率，当反向解远优于原始解的时候，$P_{\text{select}}\to1$；当反向解不优时，$P_{\text{select}}\to0$，则保留原解。这种自适应概率调控，不仅保证了优质反向解高概率入选，也避免了劣解对种群质量的拖累。优化后的麻雀搜索算法初始种群既具备了反向学习策略的广域探索能力，又借助适应度驱动的自适应选择机制显著提升了初始解的平均质量。

$$\Delta=\max[0,f(x)-f(x')]\tag{13}$$

$$P_{\text{select}}=\frac{\Delta}{f(x)+f(x')+\varepsilon}\tag{14}$$

图 1 对任务维度 0 和 1 的种群初始化结果进行了比较，图 1 中的散点对应原始麻雀搜索算法的随机初始化样本，图 2 中的散点对应引入选择性反向学习策略后的初始化样本。与原始方法相比，引入选择性反向学习策略初始化的种群在二维投影空间中呈现更广泛的覆盖范围和更均匀的分布，充分体现了该策略在提高初代种群质量与多样性方面的作用，验证了选择性反向学习策略在提升初代种群质量和多样性方面的有效性。

3.2　簇引导混合搜索策略

3.2.1　任务特征聚类

K-Means 聚类是一种经典的无监督学习算法，该

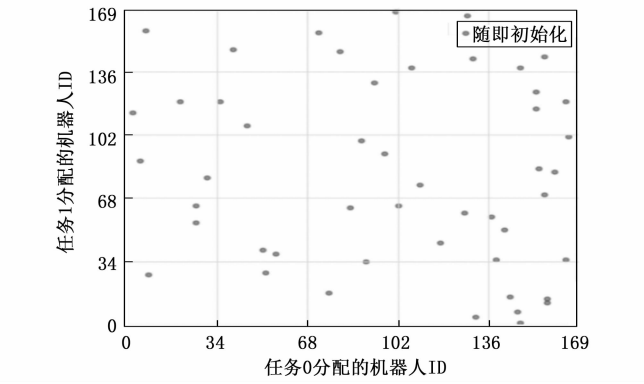

图 1 初始化种群散点图

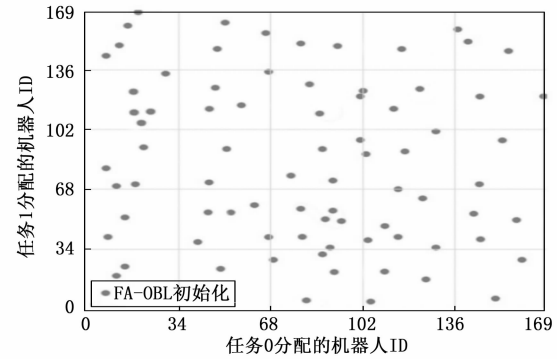

图 2 SOBL 初始化散点图

算法通过迭代优化实现数据划分，首先是根据预设的 K 值随机或者优化选择初始聚类中心，然后将每个数据点归类至最近邻聚类中心并重新计算簇内的均值，当聚类中心位置的变化低于收敛阈值或达到最大迭代次数时终止计算。

在选择聚类特征时，基于多机器人任务调度的目标与约束特性，对任务属性进行了分析。执行时间反映任务在机器人上占用的时间资源，是影响系统完工时间和调度瓶颈的重要因素；CPU 占用率直接决定机器人资源负载，能够体现任务在异构机器人上的资源竞争关系；优先级权重则用于表示任务的重要程度，影响任务是否需要被优先调度和集中执行。因此，三项特征共同刻画了任务的执行消耗、资源需求与重要性，能够较为全面地反映任务之间的调度相似性。此外，对包括截止时间、就绪时间、任务深度、前驱数量等其他候选特征进行了初步测试。结果表明，这些属性更多与时间约束或依赖拓扑相关，而非调度资源分配本身，将其加入聚类反而会减弱簇内部任务相似性对调度搜索方向的引导效果。因此最终保留执行时间、CPU 占用率与优先级作为聚类特征。

这样就会通过较低的时间复杂度处理更大规模的数据，聚类中心也有更明确的物理意义，K-Means 主要有以下两种策略：

1）特征 Z-score 标准化用于 K-Means 聚类：在 K-Means 聚类的时候，为了避免执行时间、CPU 占用率和优先级这些特征因为数值差距过大而产生影响，先对这些数据进行 Z-score 标准化。标准化后，每个特征都会在同一尺度上参与计算，不会让某一个特征过度主导聚类，也可抑制时间特征对最终分组的过度影响。在确定簇数的时候，预设区间 $K\in[2,10]$，进行 10 次独立运行，观察轮廓系数方差，并为每次划分计算 Silhouette 系数。结果表明，根据不同 K 值对应的 Silhouette 系数变化趋势，最终选取 $K=4$ 时的聚类结构，$K=4$ 时对应的轮廓系数为 0.72，聚类效果最优，能够在保证簇内部紧密性、簇间可分性和调度性能提升之间取得较好平衡，为后续簇引导变异策略的有效性提供结构支撑。

2）Silhouette 系数：在 K-Means 中，引入了计算轮廓系数（Silhouette Score）的方法，这个方法可以做到自动确定聚类的最佳簇数。Silhouette 系数通过计算同一簇内样本的紧密程度、不同簇间样本的分离程度来评估聚类质量，其取值范围为－1～1，数值越接近 1 表明聚类效果越佳。这个方法的使用可以避免手工试错固定簇数，算法通过数据本身的分布自适应的给出合理的簇数，对后边算法的运行更有帮助。

3.2.2 簇分散度惩罚

在惩罚逻辑上，设定了簇离散惩罚，并结合变异策略即可形成完整的簇引导搜索框架。分配任务时，该惩罚机制用于避免将同一簇的任务分散到多台机器人。在麻雀搜索算法调度器中，这一惩罚因子鼓励任务尽可能按照簇进行集中调度，减少跨簇任务混合带来的资源冲突。成本函数中，通过计算每个簇实际使用的机器人数量并按簇权重加权累加，将簇分散度转化为惩罚值，最终加入总成本中，引导算法优先选择簇内集中的调度方案。数学形式如式（15）所示：

$$Penalty_{\text{cluster}}=\lambda\sum_{c=0}^{K-1}\omega_c(|R_c|-1) \tag{15}$$

3.2.3 簇混合变异策略

为提升麻雀搜索算法的全局探索能力并防止种群陷入局部最优，本研究提出一种混合分簇变异策略。该策略通过融合任务分簇的结构特征与两种自适应变异机制——基于混沌映射[25]的扰动变异和 t－分布动态变异[26]，从而优化任务调度过程。

在任务调度过程中，使用 K－Means 聚类算法将任务分为多个簇，并且根据执行时间、CPU 占用率和优先级等特征进行聚类。基于任务簇信息，对任务集中的前 20％任务运用 Logistic 映射混沌，该方法可以帮助任务调度在搜索空间中进行更广泛的搜索，减少了过早收敛到局部最优解的风险。将 Logistic 映射如式（16）所

示，生成的混沌值 x 映射到机器人编号空间如式（17）所示：

$$x_{k+1}=4x_k(1-x_k),x_0\sim U(0,1) \tag{16}$$

$$\Delta c=[(x-0.5)R],R\text{ 为机器人总数} \tag{17}$$

发现者更新位置：

$$r_i^{t+1}=(r_i^t+\Delta_c)\bmod R \tag{18}$$

对于任务簇 1 和簇 2，采用的是 t 分布变异策略，随机复制最优解片段，兼顾局部开发与多样性。t 分布的总尾部特性帮助算法在解空间进行较大的跳跃，增强了全局搜索能力。该方法的使用可以有效避免局部最优解，提高任务调度效果。

加入者位置更新公式：

$$r_i^{t+1}=(r_i^t+\delta_i)\bmod R,\delta_i\sim t_{\nu=3} \tag{19}$$

δ_i 是由 t 分布在自由度 $\upsilon=3$ 采样得到的扰动，用于改变任务分配。

3.3　CG-SSA 伪代码

算法 1：CG-SSA

输入：任务集 $T=\{t_1,\cdots,t_N\}$、机器人集 $R=\{r_1,\cdots,r_M\}$、种群规模 P、迭代次数 G、反向学习强度 α_obl、混沌率 ρ_head、变异率 ρ_t、危险概率 ρ_d、簇分散惩罚 λ_disp、从最优解复制基因的比例 p_copy、t 分布自由度 ν

输出：best_solution（任务→机器人映射）

对任务特征做 Z-score；K* ← argmax Silhouette(K)；执行 K-Means(K*) 得到 cluster(t)；$w_c=1/|G_c|$；选关键簇 c*

编码 sol∈{0,1,…,M−1}^N

初始化：随机生成 sol_i；构造反向解 opp_i；评估 cost(sol_i)、cost(opp_i)；按 α_obl 与适应度差概率选择{sol_i,opp_i}

for gen=1..G do

评估并按 cost 升序排序；P_d=ceil(0.2 · P)；best ←当代最优

for i=1..P_d do

if rand()<ρ_d then 混沌变异于簇 c* 的基因位 else 以 p_copy 复制 best 的基因；if rand()<ρ_t then t_ν 扰动 end if

执行 cluster_aware_repair；若改进则替换

end for

for i=P_d+1..P do

if rand()<ρ_t then t_ν 扰动 else 小步随机跳(±1..±2, mod M) end if；

以 ρ_d 概率复制 best 的基因

执行 cluster_aware_repair；若改进则替换

end for

精英保留：当代最优与历史最优比较后保留更优

end for

returnbest_solution

设任务数量为 N、机器人数量为 M、种群规模为 P、迭代次数为 G、K-Means 内迭代次数为 I_{km}、簇数为 K。簇引导自适应麻雀算法的时间复杂度由预处理与主循环两部分构成。预处理的代价为 $O(I_{\text{km}}\cdot N\cdot K)$，仅执行一次。主循环的主导成本来自对候选解的评估，若单次评估复杂度记为 C_{eval}，则主循环复杂度为 $O(G\cdot P\cdot C_{\text{eval}})$。在本文中，朴素实现下 cost 函数需要对就绪任务集合与依赖关系进行逐次扫描，同时为每个任务查找目标机器人，导致最坏情况下复杂度达到 $C_{\text{eval}}=O(N\cdot M+N^2)$，其中 N^2 是依赖约束的遍历检测。若采用更高效的区间表与入度队列，可以将单次评价的复杂度降低至 $O(N\log S)$ 或 $O(N+E+N\log S)$，其中 S 为 interval 数，E 为依赖边数。综上整体时间复杂度可以表示为：$O(I_{\text{km}}NK)+O(GPC_{\text{eval}})$。空间复杂度主要由保存 P 个长度为 N 的候选解向量和每台机器人运行数据构成，为 $O(PN+MS)$。

4　仿真实验与结果分析

为了验证簇引导自适应麻雀算法在解决任务调度问题的优越性，分别用天鹰优化算法[27]、蜣螂优化算法[28]、沙猫群优化算法[29]、鲸鱼优化算法[30]以及未经过优化的麻雀搜索算法进行对比。各个算法的实验环境均一致，在操作系统为 Windows11 的 64 bit 系统上运行，RAM 为 16 GB，处理器为 13th Gen Intel（R）= Core（TM）i5−13500H，在 Python3.13 上进行仿真实验。各算法均采用相同的种群规模和参数设定，并在统一的环境上运行，可以确保结果具有可比性和可靠性。种群规模 pop _ size=50，迭代次数 max _ iter=1 000，在迭代次数的选择中，分别对迭代次数取值：500；1 000；1 500；2 000。当迭代次数 max _ iter=1 000时，在不显著影响任务完成率的情况下，可以明显降低延迟任务占比和调度时间。

仿真实验中，在不同的任务数量中进行实验，可以评估算法在不同的任务数量下的性能，具体的实验数据如表 2 所示。

表 2　测试样例数据

序号	任务数量	机器人数量
(1)	100	20
(2)	300	50
(3)	1 000	170
(4)	2 000	330

调度方案的综合性能既包含系统的鲁棒性，也涉及任务的时效性，度量指标不仅要反映已完成任务的比例，还要考虑未完成任务的数量及延迟执行的严重程度。实验中着重考查以下几个关键指标：任务完成率、任务按时完成率、归一化延迟指标，在完成任务格式相同的情况下总完成的时间，各项结果汇总如表 3 所示。

由表 3 以及图 3～6 可以看出，随着任务规模由 100、300、1 000 到 2 000 逐步增大，各算法在任务完成率和按时完成率方面的差异显著扩大。在任务数量为 100 时，未改进的麻雀搜索算法和簇引导自适应麻雀算法任务的完成率和准时完成率都为 1，在这时进行任务

表 3 实验结果数据

任务数量	算法名称	任务完成率/%	按时完成率/%	归一化延迟指标	完成总时间/s	未完成任务个数
100	AO	100	0.96	0.48	68.67	0
	DBO	99	0.96	0.35	82.90	1
	SCSO	96	0.95	0.28	117.90	4
	WOA	94	0.82	0.02	465.86	6
	SSA	100	1.00	1.00	43.89	0
	CG-SSA	100	100	1.00	40.45	0
300	AO	94	100	1.00	58.83	18
	DBO	90	99	0.52	72.89	30
	SCSO	91	97	0.19	97.55	27
	WOA	82	95	0.02	608.40	53
	SSA	99	100	1.00	56.87	2
	CG-SSA	100	100	1.00	50.99	1
1 000	AO	87	98	0.26	87.34	134
	DBO	86	96	0.23	130.50	137
	SCSO	90	96	0.18	124.41	102
	WOA	86	86	0.01	3 327.18	137
	SSA	91	99	0.34	74.86	95
	CG-SSA	99	99	0.33	73.94	10
2 000	AO	82	97	0.29	105.77	355
	DBO	88	97	0.24	99.96	399
	SCSO	87	98	0.25	103.11	263
	WOA	81	88	0.01	3 313.60	371
	SSA	84	99	0.32	80.85	315
	CG-SSA	98	97	0.21	84.74	44

完成时间的比较，簇引导自适应麻雀算法的完工总时间缩短了约 8%。这是因为选择性反向学习策略在初始化阶段通过适应度差异筛选，有效剔除劣质对立解，使首代种群集中于高质量可行域，进而缩短了初期搜索迭代所需时间。随着任务数增多，簇 0 任务采用 Logistic 混沌映射进行全局跳跃，能够快速打破局部收敛；簇 1/2 任务则通过 t 分布扰动结合最优片段复制，实现对优秀子结构的保留与偶发远距探索，有效降低高优先级任务延迟风险。

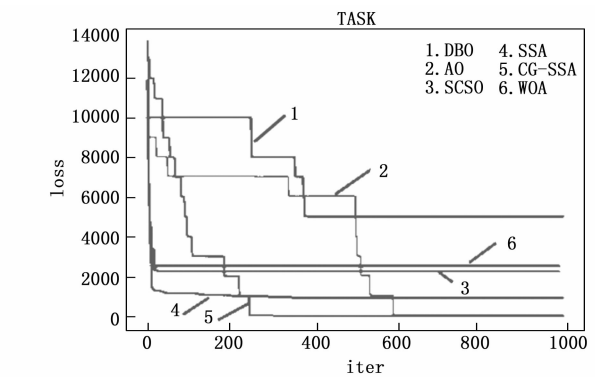

图 3 任务数 100 对比实验迭代图

从实验数据可以看出，CG-SSA 在不同规模任务下

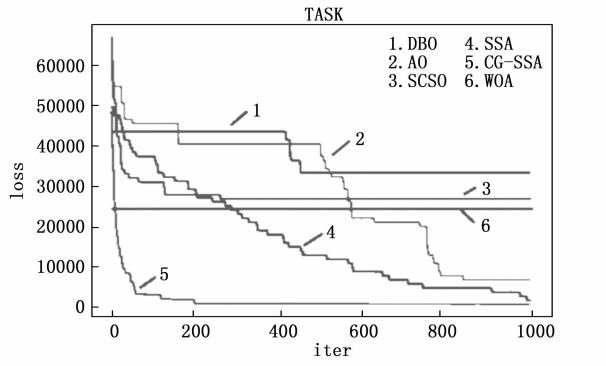

图 4 任务数 300 对比实验迭代图

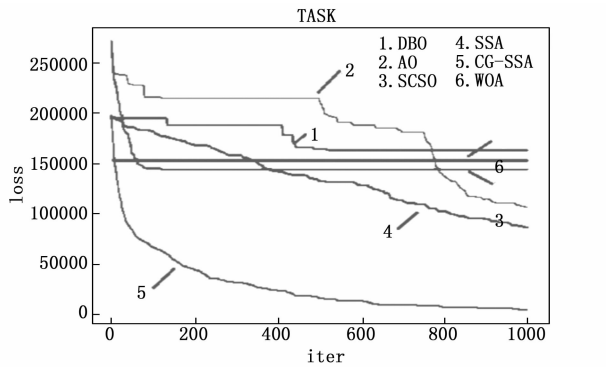

图 5 任务数 1 000 对比实验迭代图

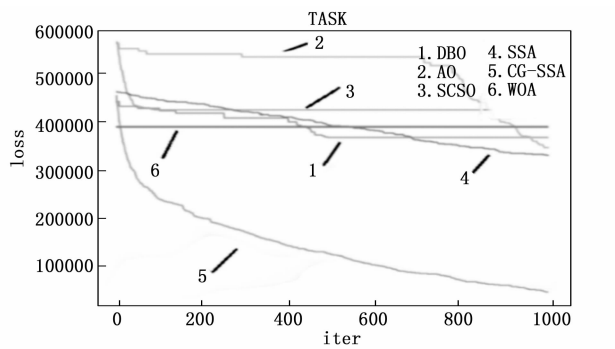

图 6 任务数 2 000 对比实验迭代图

均显著优于未改进的 SSA 及其他对比算法，这一性能提升与算法设计中的多项改进策略密切相关。选择性反向学习策略有效改善了初始种群的质量。传统 SSA 的随机初始化具有明显的偶然性，不同初代会导致较大性能波动；而 CG-SSA 通过生成原解—对立解对并基于适应度差异进行概率选择，使初始解既具备较高多样性，又能在初期偏向可行域与优质区域，从而减少算法前期的无效搜索，提升整体收敛速度。特别是在任务规模达到 1 000 或 2 000 时，初代质量对最终搜索效率影响更为显著，因此 CG-SSA 在大规模任务条件下能保持更快收敛和更高完成率。

簇引导机制通过 K-Means 聚类学习任务在执行时间、资源占用、优先级等方面的隐含结构，使同簇任务

尽可能集中到少量机器人上执行。该机制在目标函数中通过簇分散度惩罚项实现显式优化，能够有效减少跨簇任务混合导致的资源冲突与上下文切换，使任务执行更加平滑紧凑。实验中 CG-SSA 在按时完成率和归一化延迟指标上的显著优势，正是由于簇集中调度使高优先级与高 CPU 占用任务能够得到更稳定的执行环境，从而降低调度延迟风险。

Logistic 混沌跳跃和 t 分布扰动构成的簇分级混合变异策略，是 CG-SSA 在各规模任务中获得更优收敛性的关键因素。Logistic 映射具备序列遍历性与混沌跳跃特性，施加于关键簇任务的决策变量后能够实现大范围搜索跳跃，使算法有效避免在高维空间过早陷入局部最优；而普通簇采用的 t 分布变异具有重尾分布特性，使解在局部细化过程中仍有概率进行远距离探索，增强了解的多样性。同时，结合最优片段复制可以保持高质量解的稳定结构，加速局部收敛。在任务规模逐步增大时，传统 SSA 和 WOA 等算法容易出现收敛早但偏离最优解的现象，而 CG-SSA 在簇分级变异结构的引导下仍能保持较强的全局探索能力，因此在多规模测试中均表现出更高的完成率和更短的完工时间。

5　结束语

本文构建了一个多机器人任务调度模型，模型包含了任务的就绪时间、任务截止时间、任务执行时长、任务优先级权重以及机器人异构计算资源约束，并提出簇引导自适应麻雀搜索算法以求解该模型。簇引导自适应麻雀算法在经典麻雀搜索算法基础上做了两个重要改进：一是种群初始化阶段采用选择性反向学习策略，以概率方式保留优质对立解，从而显著提升初代种群质量和多样性；二是设计簇引导混合变异策略，先用 K-Means 对任务进行聚类，在目标函数中加入簇分散度惩罚，然后针对不同簇的任务分别施加 Logistic 混沌跳跃和 t 分布扰动结合最优片段复制，以兼顾全局探索和局部精化。仿真实验表明簇引导自适应麻雀算法在任务完成率、按时完成率和整体调度效果上都优于传统的麻雀搜索算法和其他主流的群体智能算法。未来的工作将研究依赖任务调度，研究任务拆分与重组、依赖图建模以及面向依赖结构的资源分配策略，从而使得算法可以更好地应用于大规模、多依赖、多机器人场景。综合来看，CG-SSA 在不同任务规模下性能稳定提升的原因不仅来自单一策略改进，而是源于初始化、结构引导和混合变异三部分的协同作用：初始化阶段提升搜索起点，聚类结构提供调度规律引导，分级变异策略保证全球探索与局部开发的动态平衡。三者共同确保了算法在高维、多依赖、多资源约束条件下仍能保持高质量搜索性能，从而在实验结果中获得对传统算法显著的性能优势。

参考文献：

[1] 田倬璟，黄震春，张益农. 云计算环境任务调度方法研究综述 [J]. 计算机工程与应用，2021，57 (2)：1 - 11.

[2] 李建强. 虚拟化环境下安全资源调度方法研究 [J]. 信息安全研究，2025，11 (7)：652 - 660.

[3] 许富景，杜少成，张　燕，等. 应用混沌反向灰狼算法的多终端复杂任务调度策略 [J]. 中国测试，2024，50 (10)：73 - 80.

[4] IBRAHIM I M，RADIE A H，JACKSI K，et al. Task scheduling algorithms in cloud computing：a review [J]. Turkish Journal of Computer and Mathematics Education，2021，12 (4)：1041 - 1053.

[5] ARUNARANI A R，MANJULA D，SUGUMARAN V. Task scheduling techniques in cloud computing：a literature survey [J]. Future Generation Computer Systems，2019，91：407 - 415.

[6] MORADI M. A centralized reinforcement learning method for multi-agent job scheduling in grid [C] //2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)，IEEE，2016：171 - 176.

[7] KOKASH N. An introduction to heuristic algorithms [J]. Department of Informatics and Telecommunications，2005，1：1 - 7.

[8] BEHESHTI Z，SHAMSUDDIN S M H. A review of population-based meta-heuristic algorithms [J]. Int. J. Adv. Soft Comput. Appl.，2013，5 (1)：1 - 35.

[9] COELLO C A. An updated survey of GA-based multiobjective optimization techniques [J]. ACM Computing Surveys (CSUR)，2000，32 (2)：109 - 143.

[10] JAIN M，SAIHJPAL V，SINGH N，et al. An overview of variants and advancements of PSO algorithm [J]. Applied Sciences，2022，12 (17)：8392.

[11] 魏　勇，赵开新，张松青，等. 基于改进蚁群算法的云计算任务调度研究 [J]. 火力与指挥控制，2017，42 (5)：130 - 133.

[12] NASIRI J，KHIYABANI F M. A whale optimization algorithm (WOA) approach for clustering [J]. Cogent Mathematics&Statistics，2018，5 (1)：1483565.

[13] 刘陈伟，孙　鉴，雷冰冰，等. 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.

[14] 沈建国. 基于多目标优化算法的云计算调度任务分配及优化 [J]. 延边大学学报 (自然科学版)，2025，51 (1)：19 - 24.

(下转第 250 页)