2023 届硕士专业学位研究生学位论文

分类号：_____ 学校代码：10269

密　级：_____ 学　　号：71194501140

華東師範大學

East China Normal University

硕士专业学位论文

Master's Degree Thesis(Professional)

论文题目：基于群体智能算法的云计算任务调度策略研究

院　　系：	软件工程学院
专业学位类别：	工程硕士
专业学位领域：	软件工程
学位申请人：	罗婷婷
指 导 教 师：	郭建 副教授

2023 年 11 月 10 日

Thesis for Master's Degree (Professional) in 2023

University code:10269

Student ID:71194501140

East China Normal University

Study on Cloud Computing Task Scheduling Strategy Based on Swarm Intelligence Algorithm

Department/School：	**Software Engineering Institute**
Category：	**Master of Engineering**
Field：	**Software Engineering**
Candidate：	**Luo Tingting**
Supervisor：	**Associate Professor Jian Guo**

November ,2023

罗婷婷硕士学位论文答辩委员会成员名单

姓名	职称	单位	备注
张苗苗	研究员	同济大学	主席
杜德慧	教授	华东师范大学	
赵涌鑫	副教授	华东师范大学	

摘要

云计算现在有很大的应用市场，但受限于云计算任务调度策略的研究不够深入，不能匹配日益增长的用户数和数据量的需求。为了满足这一迫切需要，深入研究云计算任务调度策略的特性和算法将具有十分重要的意义。

现有云计算任务调度策略按搜索特性可分为传统调度算法和启发式智能调度算法两类。传统调度算法具有搜索速度快的优势，但不能满足日益复杂的调度需求。启发式智能调度算法兼具搜索速度和调度效率，但性能不够稳定。本文在现有的启发式调度算法基础上，着重研究群体智能搜索算法，实现在提升稳定性的同时保证算法的调度效率。

本文主要研究内容如下：

（1）在云计算任务调度中，研究、实现并适当调整蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法这四种群体智能算法。考虑到对传统的搜索任务而言，数据的预排序与否将很大程度上影响算法性能，但云计算任务调度中，是否需要对任务和虚拟机进行预排序尚未有明确结论。因此本文先设计任务与虚拟机的排序算法，在完成任务与虚拟机预排序后基于 Cloudsim 平台调用这四种算法，对比四种算法在不同虚拟机数目和不同任务数目下的完工时间、系统负载平衡、完工费用的综合表现，最终得出结论：不对任务与虚拟机进行预排序，引入更多的合理随机性和适度扩张搜索范围，将更有利于提高群体智能算法的任务调度性能。

（2）考虑到蚁群算法容易导致系统负载不均和早期搜索效率低，本文提出一种新的免疫-蚁群优化算法 IMACO（Immune-Ant Colony Optimization Algorithm），引入负载监测启发因子、动态挥发和双重奖惩信息素更新策略，针对性改进启发因子和信息素更新规则；利用免疫算法全局搜索能力强、解多样性高、不易陷入局部最优的优点获得蚁群的初始信息素分布，可有效规避蚁群算法初始求解慢的缺陷，并基于 Cloudsim 平台完成与蚁群算法的对比，免疫-蚁群优化算法在完工时间、系统负载平衡、完工费用的综合表现更优。

（3）本文提出一种新的改进白鲸优化算法 IBWO（Improved Beluga Whale Optimization），一方面引入模拟退火算法，给予一定概率接受次优解，协助算法跳出局部最优，同时在鲸落步骤后加入准反向学习策略，避免单向搜索，扩大解范围。基于 Cloudsim 平台完成与蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法及本文提出的 IMACO 的对比，IBWO 在完工时间、系统负载平衡、完工费用的综合表现更优。

关键词：云计算，任务调度，群体智能算法，蚁群算法，白鲸优化算法

Abstract

Cloud computing now has a large application market, but due to insufficient research on cloud task scheduling strategies, it cannot match the growing demand for users and data volume. In order to meet this urgent need, in-depth research on the characteristics and algorithms of cloud task scheduling strategies will have significant meaning.

The existing cloud task scheduling strategies can be divided into two categories based on search characteristics: traditional scheduling algorithms and heuristic intelligent scheduling algorithms. Traditional scheduling algorithms have the advantage of fast search speed, but they cannot meet increasingly complex scheduling needs. The heuristic intelligent scheduling algorithm combines search speed and scheduling efficiency, but its performance is not stable enough. On the basis of existing heuristic scheduling algorithms, this thesis focuses on researching swarm intelligence search algorithms to improve stability while ensuring the scheduling efficiency.

The main research content of this thesis is as follows:

(1) In cloud task scheduling, research, implement, and appropriately adjust four swarm intelligence algorithms: Ant Colony Optimization, Grey Wolf Algorithm, Whale Optimization Algorithm, and Beluga Whale Optimization. Considering that for traditional search tasks, the pre-sorting of data will greatly affect algorithm performance, but there is no clear conclusion on whether pre-sorting of tasks and virtual machines is necessary in cloud task scheduling. Therefore, this thesis first designs a sorting algorithm for tasks and virtual machines. After pre-sorting the tasks and virtual machines, these four algorithms are called based on the Cloudsim platform. The comprehensive performance of the four algorithms in terms of completion time, system load balancing, and completion cost under different virtual machine numbers and task numbers is compared. The final conclusion is that not pre-sorting tasks and virtual

machines, introducing more reasonable randomness and moderately expanding the search range, will be more conducive to improving the task scheduling performance of swarm intelligence algorithms.

(2) Considering that Ant Colony Optimization can easily lead to uneven system load and low early search efficiency, this thesis proposes a new Immune-Ant Colony Optimization IMACO (Immune-Ant Colony Optimization Algorithm), which introduces load monitoring heuristic factor, dynamic volatilization, and dual reward and punishment pheromone update strategy, and improves heuristic factor and pheromone update rules accordingly. By utilizing the advantages of strong global search ability, high solution diversity, and difficulty in falling into local optima of Immune Algorithm, the initial pheromone distribution of ant colony can be obtained, which can effectively avoid the defect of slow initial solution of Ant Colony Optimization. Based on the Cloudsim platform, the comparison with Ant Colony Optimization is completed. The Immune-Ant Colony Optimization performs better in the comprehensive performance of completion time, system load balance, and completion cost.

(3) This thesis proposes a new Improved Beluga Whale Optimization (IBWO) algorithm, which introduces Simulated Annealing algorithm to give a certain probability of accepting suboptimal solutions and assist the algorithm in jumping out of local optima. At the same time, a Quasi-Opposition-Based-Learning strategy is added after the whale fall step to avoid unidirectional search and expand the solution range. Based on the Cloudsim platform, IBWO performs better comprehensively in terms of completion time, system load balancing, and completion cost compared to Ant Colony Optimization, Grey Wolf Algorithm, Whale Optimization Algorithm, Beluga Whale Optimization and IMACO proposed in this thesis.

Keywords: cloud computing, resource scheduling, swarm intelligence algorithm, Ant Colony Optimization , Beluga Whale Optimization

目 录

插图

表格

第 1 章 绪论

1.1 研究背景和意义

云计算是一种 2007 年诞生的商业化的计算模式，它将不同的计算任务按一定策略分配在众多计算机组成的资源池上，使用户按需获取资源，包括算力、存储空间和其他服务，具有大规模、虚拟化、高可靠、低价等优势[1]。

随着用户增长和 5G 普及，如何高效完成任务调度、改善云资源利用率引起广泛关注。一方面，云用户对费用、完工时间、可靠性等服务质量（Quality of Service, QoS）敏感；另一方面，云服务提供者要控制成本。因此需要高效合理的云计算任务调度策略，不仅能改善服务质量，同时可以提高云资源利用率和系统负载平衡，降低服务成本[2]。

根据没有免费午餐定理[3]，没有一种通用算法可以在各种任务中都具备优秀表现，因而近年针对云计算任务调度策略的研究很活跃，主要有四种形式，一种是将其他领域的现有算法移植到云计算任务调度领域，第二种是优化云计算任务调度领域已有算法，第三种是基于现有算法的优点混合两种或多种算法，第四种是针对云计算任务调度需求提出新的算法[4]。

本文计划在云计算任务调度算法领域进行科学调查与实验，分析与总结现有算法的同时，在现有算法基础上提出改进，获得更有效的云计算任务调度算法，从而达到提升现有云计算平台性能的目的，因此在实际应用与理论研究层面都具有一定意义。

1.2 国内外研究现状

1.2.1 云环境下任务与虚拟机的排序算法

云计算任务调度算法的作用是按调度策略将任务分配到合适的虚拟机上，使得完工时间、系统负载平衡等性能参数表现优异。

邢焕来等人[5]用有向循环图表示任务序列，提出关键路径节点优先的任务优先级计算方式，完成任务优先级排序后再实现虚拟机分配，有效降低了系统能

耗。武小年等人[6]提出两阶段任务调度算法，预先计算各任务完成时间、费用和损失度，并完成任务预排序，在调度时为任务匹配满足时间-费用偏好的最小资源。

云环境下任务和虚拟机的现有排序算法一般只针对任务本身展开排序，而对任务和虚拟机两者之间排序的研究非常少。关于两者排序与否能否提升云环境下任务调度算法的性能，尚没有明确结论，因此本文将进行任务与虚拟机排序相关的研究。

1.2.2 传统云计算任务调度算法

（1）最小调度算法（Minimum-minimum completion time,简称 Min-Min）

郭平等人[7]提出一种考虑负载平衡的 LL-Min-Min 算法，可满足云计算任务不确定性及多样性需求，实现改善数据中心负载平衡和集群吞吐率的目的。潘钰[8]针对 Min-Min 算法做改进，提出一种以云环境中完成每个任务的最小能耗为目标的任务调度算法 Min-Energy，新算法在节能上改善显著。

（2）最大最小调度算法（Maximum-minimum completion time,简称 Max-Min）

王政[9]将云环境下的任务按长度划分为简单和复杂两类，当任务序列中简单任务占比多采用 max-min 算法，当复杂任务占比多采用 min-min 算法，当两者占比接近时则采用他提出的两阶段调度算法 GTSA-TW，缩短任务完工总时间的同时提高了系统负载平衡。

传统调度算法虽然实现过程简单，但目前已无法满足云环境下大规模任务处理时的 QoS 需求。

1.2.3 启发式云计算任务调度算法

（1）遗传算法（Genetic Algorithm，简称 GA）

夏学文等人[10]提取精英个体的最长公共子序列，将其应用至遗传算法的交叉和突变操作，降低优秀解被破坏的概率从而实现更有效的任务调度。

（2）粒子群优化算法（Particle Swarm Optimization，简称 PSO）

王镇道等人[11]提出竞争粒子群优化算法，构建了完成时间、系统能耗和负

载均衡的多目标优化模型，引入混沌优化方法初始化粒子种群，并用自适应的高斯变异更新粒子位置，算法的全局搜索能力得到有效提高。

（3）模拟退火算法（Simulated Annealing Algorithm，简称 SA）

SA 算法一般会和其他启发算法结合以提高性能，如何庆等人[12]提出一种改进遗传模拟退火优化算法，改进遗传算法的适应度函数和交叉变异算子，使算法有效跳出局部最优，增强算法寻优能力。

（4）蚁群算法（Ant Colony Optimization，简称 ACO）

Gao, R.[13]提出一种通过蚁群优化实现的动态负载平衡算法，引入最大最小触发规则用于快速找到系统欠载和过载节点。

1.2.4 云计算任务调度算法优化目标

合理高效的云计算任务调度策略可以大幅提升云平台的工作效率和有效降低成本，目前主要优化目标有以下四种：以缩短完工时间为目标、以提高服务质量QoS为目标、以稳定系统负载平衡为目标、以降低能耗和成本为目标。

以上四个目标在实际调度过程中彼此存在一定关联性，并非互相独立。目前国内外研究者一般会针对其中一个或选取多个目标进行组合优化，本文就现阶段研究较多的方向进行了深入调研学习。

（1）以缩短时间跨度为目标的调度策略

一般把云系统处理用户提交的全部任务所耗费的时间称为时间跨度，是调度性能优化中最直观的衡量因子。Ali Belgacem等人[14]提出一种空间多目标蚁狮算法，显著减小任务时间跨度。张金泉等人[15]提出一种正交自适应的鲸鱼优化算法，应用正交试验设计初始化种群，运用双向搜索策略扩大解空间，获得了优秀的时间跨度。

（2）以提高服务质量QoS为目标

常用QoS参数有费用、完成时间、可靠性、丢包率、延迟、带宽等。聂清彬等人[16]针对用户开销、完成时间和结果是否有效这三个QoS需求，引入随机选择机制提出一种改进蚁群算法，提高了算法QoS表现。

（3）以稳定系统负载平衡为目标

负载平衡指的是在云计算分布式环境中，跨多个节点均匀地动态分配工作负载，最终目标是确保没有节点过载或欠载。

目前以稳定系统负载平衡为单一优化目标的研究较少，如Nakrani T.等人[17]在算法中加入适应度函数，提出了一种基于遗传算法的负载平衡任务调度方法。

（4）以降低成本和能耗为目标

云计算作为一项商业性质的服务，成本关乎利润，是提供商最重视的问题；同时数据中心不断攀升的能耗已成为限制其发展的主要因素之一。郑瑛[18]提出一种基于能耗感知的虚拟机调度算法，使虚拟机向负载率高的物理机迁移，减小数据中心耗能。

（5）以缩短时间跨度和提高服务质量QoS为目标

任金霞等人[19]创建了时间、性能、费用三维QoS需求模型，引入多维QoS约束时间贪心策略获得初始解，提高了用户满意度。姜力争等人[20]提出一种具备自适应的信息素调节方式的蚁群优化算法，通过虚拟机状态变化来调整启发因子和信息素浓度，最优跨度和系统稳定性表现优秀。

（6）以缩短时间跨度和稳定系统负载平衡为目标

Gabi D.等人[21]提出了基于田口方法的动态任务调度算法，把正交田口方法运用至优化猫群算法的局部搜索中提高收敛速度，并以最短执行时间在虚拟机上进行任务调度。刘愉等人[22]提出了一种基于适应度函数、染色体编码的改进遗传算法，先将负载平衡、最优跨度函数纳入目标函数，设计高效合理的染色体编码方案，该算法可有效缩短任务完工时间、提高系统负载平衡。

（7）以稳定系统负载平衡和降低能耗、成本为目标

Lu等人[23]提出了一种高效的全局优化资源感知算法，旨在组合优化负载平衡和能耗问题；马小晋等人[24]提出了一种改进模拟退火优化算法，提高了虚拟机在物理机之上的调度分配效率，有效压缩供应商成本投入。王艳红等人[25]提出了基于狮群算法的云计算任务调度策略，通过引入混沌映射策略、模拟退火，克服了传统狮群算法易早熟的不足。

综合考虑用户与服务供应商的需求,本文选取任务完工时间、系统负载均衡

和任务完工费用的综合表现评价云计算任务调度算法的性能。

1.3 云计算中群体智能算法研究现状

云计算领域众多算法中，群体智能算法因其参数简单、易实现、收敛快等优点被广泛研究与应用。其中捕食类群体智能算法均通过模拟食物或猎物获取的过程实现搜索和寻优，具有一定的相似性，本文选取以下 4 种捕食类群体智能算法做研究。

（1）蚁群算法

蚁群算法是由 Colorni 等人[26]提出的一种参考蚂蚁觅食行为的正反馈启发式智能算法，因其参数较少和分布式并行计算的特性被广泛用于大规模云系统的任务调度[27]。

赵辉等人[28]提出了一种基于虚拟机性能感知的虚拟机迁移策略，利用蚁群算法进行虚拟机调度，实现最大化虚拟机性能和最小化实体数量的目的。华夏渝等人[29]创建了一个用于预测下一任务执行速度的模型，并将该模型运用至蚁群算法下一跳选择中，使算法的响应时间减少同时运行质量提升。

（2）灰狼算法

灰狼算法由 Seyedali Mirjalili 等人[30]于 2014 年首次提出，根据狼群严格的等级制度和捕猎过程实现寻优，在搜寻、包围、攻击猎物的过程中，每次迭代会保留当前结果最好的 3 只灰狼，根据它们的位置更新剩余狼的位置[31]。

Bilal H.等人[32]提出一种分布式灰狼优化算法，通过引入最大价值方法将算法产生的连续候选解转化为离散候选解，从而更快地将任务分配到虚拟机并缩短计算时间，有效避免了灰狼算法无法直接求解离散域的不足。叶峰阳等人[33]提出一种增强型多目标灰狼优化算法，利用反向学习策略提高初始探索空间的能力，对控制参数采取非线性调整策略，提高全局搜索能力，该算法在服务质量和能耗的表现均更优。

（3）鲸鱼优化算法

鲸鱼优化算法由 Mirjalili 等人[34]2016 年首次提出，通过模拟座头鲸搜索、包围、攻击猎物的捕食过程，实现寻优。

张永等人[35]针对鲸鱼优化算法收敛速度和精度的不足，利用分段混沌映射初始化鲸鱼位置，引入非线性自适应权重策略，提高种群多样性、平衡全局和局部搜索能力。Mangalampalli 等人[36]初始化种群后计算任务优先值、虚拟机优先值和适应度函数值，值的大小对应不同的位置更新策略，以此获得较高的客户满意度。

（4）白鲸优化算法

白鲸优化算法由钟昌廷等人[37]于 2022 年提出，是一种模拟白鲸群体觅食过程的智能优化算法，由探索、开发、鲸落 3 个阶段组成。决定由探索转入开发的平衡因子和鲸落概率是自适应的，且在开发阶段引入莱维飞行增强全局收敛，因此相较于其他启发式算法，白鲸优化算法在大规模寻优问题上更有潜力。

蚁群算法实现简单、应用领域广泛，但存在显著不足，如容易导致系统负载失衡、早期求解速度慢，有很大的优化空间。白鲸优化算法大规模寻优潜力可观，因此本文计划对蚁群算法和白鲸优化算法做基于云平台的改进，提高其在任务完工时间、系统负载平衡、任务完工费用三方面的综合性能，也为后续改进其他群体智能算法打好基础。

1.4 本文主要工作

基于文献调研，由于关于任务与虚拟机排序与否能否提升云环境下任务调度算法的性能，尚没有明确结论，本文先研究任务与虚拟机的预排序对群体智能算法在云任务调度中的性能影响，选取任务完工时间、系统负载平衡、任务完工费用这三个性能指标综合评价算法表现；得出结论后，为进一步提升云平台性能，提出一种新的免疫-蚁群优化算法 IMACO 和改进白鲸优化算法 IBWO。本文主要工作包括以下 3 方面：

（1）基于云任务调度环境，实现蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法，设计了虚拟机和任务的排序优先级算法并完成虚拟机和任务的预排序。基于 Cloudsim 平台调用以上四种算法完成任务调度，虚拟机数目分别设置为 6 和 8，任务数量依次为 200、400、800、1600，对比任务与虚拟机排序和不排序情况下算法完工时间、系统负载平衡、完工费用这三个性能指标的综合

表现，得出结论：本文云环境中，不进行虚拟机与任务预排序，算法性能更优；

（2）基于第（1）项工作得出的结论，考虑到蚁群算法应用领域广泛但存在容易负载不均、早期搜索速度慢的缺点，调研并改进蚁群算法，创建负载监测启发因子、动态挥发和双重奖惩信息素更新策略，针对性改进启发因子和信息素更新规则，使得任务对虚拟机的选择概率与系统负载平衡相匹配；引入全局搜索能力强的免疫算法用以获得蚁群的初始信息素分布，有效规避蚁群算法初始求解慢的问题，并基于 Cloudsim 完成和经典蚁群算法的性能对比试验，虚拟机数目设置为 6，任务数目依次为 200、400、800、1600，分析数据后得出结论：IMACO 在完成时间、负载平衡、完工费用的综合表现优于 ACO；

（3）基于第（1）项工作得出的结论，考虑到白鲸优化算法优秀的大规模寻优潜力，但仍存在易陷入局部最优的不足，调研并改进白鲸优化算法，引入模拟退火和准反向学习策略，使得白鲸优化算法在现有基础上具备更强的空间搜索能力和随机性，同时有更强的跳出局部最优的能力。基于 Cloudsim 完成算法性能对比试验，虚拟机数目设置为 6，任务数目依次为 200、400、800、1600，IBWO 在完成时间、负载平衡、完工费用的综合表现优于 ACO、GWO、WOA、BWO 及本文提出的 IMACO。

1.5 本文组织结构

本文由 6 个章节组成。

第 1 章对云计算任务和虚拟机排序算法、任务调度算法、优化目标做了文献调研和总结，重点研究了群体智能算法。

第 2 章先对云计算的概念、特点和体系架构做概述，再阐述云计算任务调度模型、传统云计算任务调度算法和常见启发式调度算法的原理，重点介绍四种群体智能算法的数学模型和流程，包括：蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法。

第 3 章实现蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法，设计了虚拟机和任务的排序优先级算法并完成虚拟机和任务的预排序。基于 Cloudsim 平台完成算法性能对比实验并得出结论：在本文云环境下，不进行虚拟机与任务

预排序，更有利于提高群体智能算法在云计算任务调度中的性能。

第 4 章提出免疫-蚁群优化算法 IMACO 和改进白鲸优化算法 IBWO，分别阐述两个新算法的设计要点，详细介绍算法的实现步骤和过程，并完成算法复杂度分析。

第 5 章基于第 3 章结论，不对任务和虚拟机做预排序，在 Cloudsim 平台上先完成本文改进免疫-蚁群优化算法和经典蚁群算法、本文改进白鲸优化算法和白鲸优化算法的性能对比，再完成改进白鲸优化算法、免疫-蚁群优化算法和蚁群算法、灰狼算法、鲸鱼算法、白鲸优化算法的综合性能对比，通过分析实验数据与结果得出结论。

第 6 章总结本文的研究内容及成果，为下一步工作做准备与展望。

第2章 云计算概述与群体智能算法研究

本章介绍本文研究主题的相关知识，先对云计算的概念、特点和体系架构做概述，再阐述云计算任务调度模型、传统云计算任务调度算法和常用启发式调度算法的原理，重点介绍四种群体智能算法的数学模型和流程，包括：蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法。

2.1 云计算概述

2.1.1 云计算概念与特点

NIST（National Institute of Standards and Technology）即美国国家标准与技术研究所给云计算下了一个定义：云计算是一种可按需访问一个可配置的计算资源（如网络、存储、服务器等）的公共集的模型，这些资源能够被快速提供并发布，且服务供应商的干涉和管理成本也被大幅降低[38]。

与其他计算范式相比，云计算有其显著优点[39]：超大规模、虚拟化、高可靠性、弹性服务、按需付费、廉价。

2.1.2 云计算体系架构

云计算体系架构如图 2.1，分为三部分：核心服务、服务管理和用户访问接口[40]。

核心服务层的功能是把硬件基础设施、应用程序及其运行环境抽象为服务；服务管理层的作用是给予核心服务层相应支持，它的存在为核心服务层的服务质量、安全和可靠性提供了保障；用户访问接口层满足端到云的访问。

核心服务层提供三类服务模式：基础设施即服务(Infrastructure as a Service，IaaS)、平台即服务(Platform as a Service，PaaS)、软件即服务(Software as a Service，SaaS)。

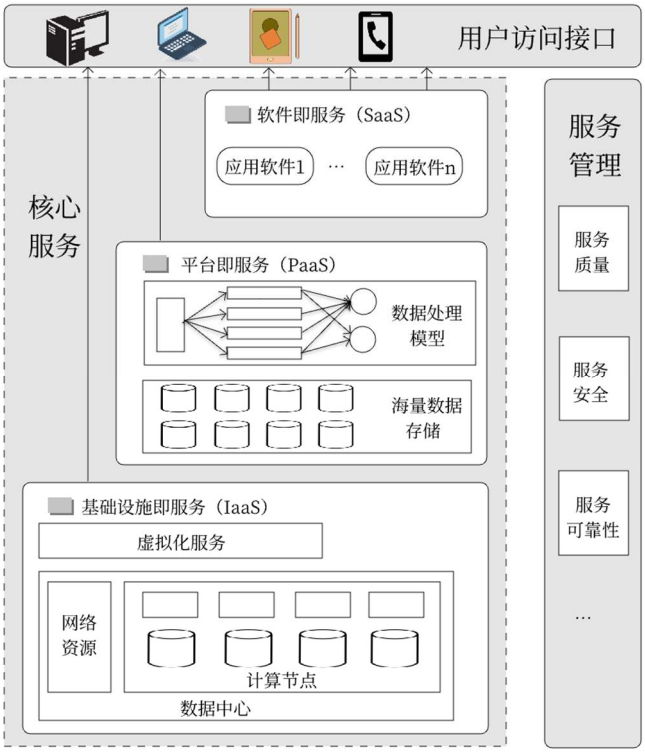

图 2.1　云计算体系架构

2.2 云计算任务调度研究

云计算任务调度框架如图 2.2，首先用户经由互联网发送任务请求，并在用户层组织为任务序列；接着服务层根据各请求的需求，申请资源层对应资源；最后资源层结合当下系统实际情况利用资源调度策略给用户任务分配相应资源。

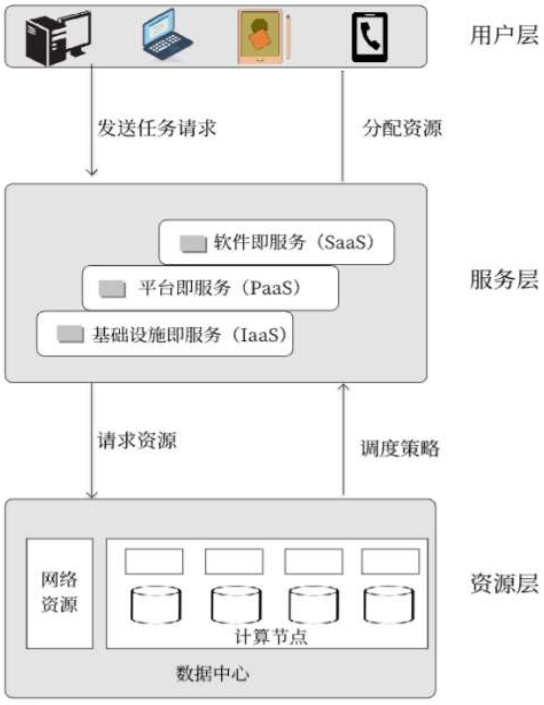

图 2.3　云计算任务调度框架

虚拟化技术实现了数据中心的海量物理资源得以被虚拟化为大量逻辑资源即虚拟机，使云资源具备两级调度的特点，第一级为虚拟机和主机间的调度，要求虚拟机能够被合理地放置、迁移到适合的主机上，形成映射关系；第二级是用户提交的任务和虚拟机间的调度，通过任务调度算法实现。

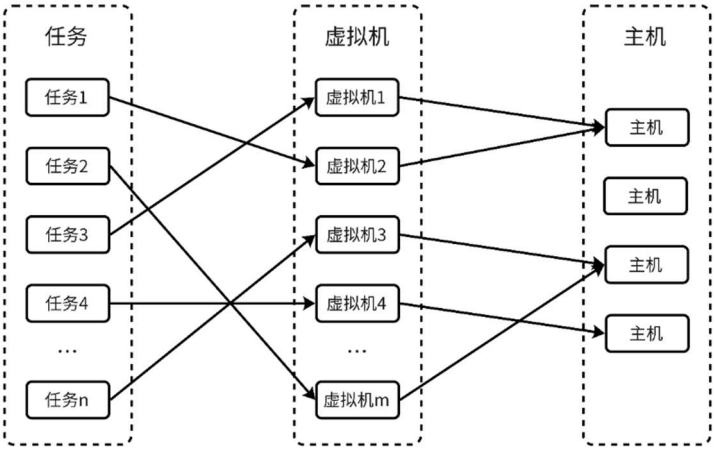

图 2.5　云计算二级任务调度模型

本文的研究对象为任务与虚拟机间的调度算法，它是实现计算资源合理分配、提高系统性能的重要手段。云计算任务调度策略按搜索特性分为两大类：传统经典和启发式智能的调度策略。

2.2.1 传统经典任务调度算法

最小调度算法（Minimum-minimum completion time,简称 min-min）简单有效，执行过程有三步：一是计算所有待完成任务在各虚拟机上的完成时长，二是将完成时长最短的任务分配至对应虚拟机执行，并将该任务从待完成任务列表删除，三是重复一、二直至所有任务完成，该算法以总的完工时间为唯一指标[41]。

显然，该算法需要等所有待处理任务到齐后，再进行各任务完成时长的计算，以完成时长为唯一考量会引起诸多不良后果，包括高性能的资源长时间不间断运转，相应而来的其他资源闲置，由此引发系统负载不均，以及完成时长较长的任务即使紧急也不能被及时处理。

与 min-min 算法类似的有最大最小调度算法（Minimum-maximum completion time，简称为 max-min 算法），二者主要的不同在于 max-min 算法首先调度的是

完成时长最长的任务。因此，max-min 算法可以在 min-min 算法的基础上有效提高云资源利用率、缩短全部任务的总完成时间和改善系统负载平衡，但其也无法满足大规模寻优的需求。

轮询调度算法（Round-robin scheduling）原理为任务按照到来顺序依次被安排至云资源上处理，对云资源的调用是以循环往复的方式进行。该算法在云平台各节点资源的软硬件配置接近且各个任务的申请服务的时间间隔变化不大时，有良好表现。但实际云环境中，各节点的运算能力不尽相同，因此在轮询调度算法的基础上，提出了加权轮询调度算法（Weighted Round-robin scheduling），根据各节点的处理能力为其分配权值，使其处理的任务数能够和权值大小对应，权值越大的节点被分配到的任务数越多，该算法有效避免负载不均。

以上传统调度算法虽然实现过程简单，但都未充分站在云用户的立场考虑，仅以单一性能作为目标，无法适应大规模寻优。

2.2.2 启发式任务调度算法

（1）遗传算法

遗传算法是由 Holland 教授率先提出的一种仿生智能启发式算法[42]，通过模拟生物界进化寻优，包含选择、交叉、变异三个关键操作算子。

初始种群随机生成，个体优劣由预先设定的适应度函数评估。选择过程实现个体被复制去下一代的可能性和其适应度值呈正相关，遵循生物进化论的“适者生存”原则。交叉过程要完成在以特定概率选择的两条染色体之间交换部分染色体片段，遵循进化过程“信息交换”原则。变异过程会以特定概率和变换方法改变个体的某些基因，从而拓展新的解空间，避免早熟[43]。

综上，遗传算法具有优良的全局搜索和并行搜索能力，且进化过程由适应度评价函数驱动、实现简单，但它在应用中也有一个显著不足，局部搜索能力欠佳，随着任务和计算规模的增长，收敛速度会越来越慢。

（2）粒子群优化算法

粒子群优化算法是美国的 Kennedy 和 Eberhart 于 1995 年提出的仿生种群智能优化算法[44]，通过种群内个体协作和信息共享寻找最优解。

粒子具备2个基本属性：速度和位置，速度决定移动快慢，位置决定移动方向。每个粒子在搜索空间中独立搜寻最优解，找到后记作当前个体极值，并将此值与其他粒子共享，把全体中最优个体极值设为当前全局最优解。引入自身认知项、社会认知项系数分别控制粒子对自身最优解、全局最优解的依赖比重，粒子根据自己的当前个体极值与当前全局最优解来调整自身速度和位置[45]。

通过粒子寻优过程可以看到 PSO 算法具有参数少易实现、个体局部信息和群体全局信息协同搜索的优点，但也存在局部搜索能力弱、易早熟等不足。

2.3 群体智能算法

群体智能算法参数简单、搜索性能优异，近年被广泛研究与应用，接下来构造蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法的数学模型。

2.3.1 蚁群算法

ACO 是一种基于群体的随机搜索算法，主要有 2 个阶段：适应和协作。适应阶段，初始随机解根据进化过程中信息变化而变化，协作阶段，解之间进行信息共享从而搜索到最优解。

（1）适应阶段：

对每只蚂蚁，R_k依照访问顺序记录该蚂蚁已访问过的节点序号。定义下一跳选择概率函数为：

$$P_k(i,j)=\frac{[\tau(i,j)]^{\alpha}[\varphi(i,j)]^{\beta}}{\sum_{\mu\in J_k(i)}[\tau(i,\mu)]^{\alpha}[\varphi(i,\mu)]^{\beta}} \tag{2.1}$$

式中:

$P_k(i,j)$表示蚂蚁k在节点i时选择节点j作为下一跳的概率；

$J_k(i)$表示从i出发可直接到达且不在该蚂蚁已访问序列R_k中的节点集合；

$\tau(i,j)$为节点i到节点j路线上的信息素值；

$\varphi(i,j)$是一个表征路径质量的信息，一般由节点i和j间距离的倒数决定，即两节点距离愈近，该路径质量愈高，$\varphi(i,j)$值也越大；

μ为$J_k(i)$中的节点。

可见，路径长度越短、信息素浓度越高的路线被选中的概率越大。α和β为预设参数，用以调控信息素浓度和路线长度在选择下一跳概率中的权重。为避免算法早熟，一般会引入轮盘赌法为蚂蚁选择下一个节点加入随机性。

（2）协作阶段：信息素更新

算法初始化时所有路径的信息素均为τ_0。每迭代一次，所有路径的信息素以一定蒸发率挥发，可通过乘以一个小于 1 的常数实现。蚂蚁依据自身构建路径的长度在本轮行经路径上释放信息素，遵循路径越短蚂蚁释放信息素越多的原则，重复此过程直至算法满足退出条件。

信息素更新分两种：一种是蚂蚁路过一个节点后立即更新该路径的信息素，称为局部更新；另一种是蚂蚁遍历所有节点后，对本轮迭代产生的优秀路径更新其信息素，称为全局更新。

局部更新的信息素更新公式为：

$$\tau^{t+1}(\mathrm{i,j}) = (1-\rho)\cdot\tau^{t}(\mathrm{i,j}) + \Delta\tau_k^t(i,j) \tag{2.2}$$

全局更新的信息素更新公式为：

$$\tau^{t+1}(\mathrm{i,j}) = (1-\rho)\cdot\tau^{t}(\mathrm{i,j}) + \Delta\tau^t(i,j) \tag{2.3}$$

两式第一部分均表示i和j间路径残余信息素，ρ为挥发因子，需满足$\rho \in (0,1)$；其中$\Delta\tau^t(i,j)$表示全体蚂蚁在从i到j转移后在路径上留下信息素总和：

$$\Delta\tau^t(i,j) = \sum_{k=1}^{n}\Delta\tau_k^t(i,j) \tag{2.4}$$

式中$\Delta\tau_k^t(i,j)$表示蚂蚁k从i到j后的信息素增量：

$$\Delta\tau_k^t(i,j) = \begin{cases} \dfrac{Q}{C_k}, & if\ (i,j)\epsilon T^k(t) \\ 0, & otherwise \end{cases} \tag{2.5}$$

式中：Q表示信息素浓度常量；C_k表示蚂蚁k行经路线长度；$T^k(t)$表示蚂蚁k在本轮循环后的分配方案。信息素增量只与搜索路径长度有关，与具体的(i,j)路径无关，且蚂蚁k完成一次搜索后，更新线路上所有路径信息素。

2.3.2 灰狼算法

狼群按严格等级从上至下分为：α狼、β狼、δ狼、ω狼。α狼是狼群领导者，即优化算法中的最优解；β狼协助α狼决策，即次优解；δ狼听从α、β狼的命令，一般负责侦察、护理工作，适应度降低的α、β狼会转变为δ狼；ω狼需要服从前三层的狼。

灰狼优化算法（Grey Wolf Optimization，简称 GWO）分为狼群等级分层、包围、狩猎、进攻、探索五步。

（1）初始化种群完成等级分层，计算所有个体适应度值，选取适应度值前三的狼依次标记为α狼、β狼、δ狼，GWO 通过每代种群中的最优的 3 个解进行位置更新；

（2）包围猎物：

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_p}(t) - \vec{X}(t)\right| \tag{2.6}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A} \cdot \vec{D} \tag{2.7}$$

式中：t是当前迭代次数，$\vec{A}$ 和 $\vec{C}$ 是系数向量，$\overrightarrow{X_p}(t)$表示迭代次数为t时的猎物的位置向量，$\vec{X}(t)$是当前狼本次迭代的位置向量，$\vec{X}(t+1)$是当前狼下一次迭代的位置向量。

$\vec{A}$ 和 $\vec{C}$ 由以下公式计算：

$$\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} - \vec{a} \tag{2.8}$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \tag{2.9}$$

式中：$\vec{a}$ 随迭代次数增加从 2 线性减小到 0，$\overrightarrow{r_1}$ 和 $\overrightarrow{r_2}$ 是[0,1]里的随机向量。

（3）狩猎：用α、β、δ保存系统迄今为止最优的 3 个解，使ω狼按α、β、δ的位置去更新位置。

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \vec{X}\right| \tag{2.10}$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \vec{X}\right| \tag{2.11}$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X}\right| \tag{2.12}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot \overrightarrow{D_\alpha} \tag{2.13}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \overrightarrow{D_\beta} \tag{2.14}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot \overrightarrow{D_\delta} \tag{2.15}$$

$$\vec{X}(t+1)=\frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \tag{2.16}$$

式中：$\overrightarrow{X_\alpha}$、$\overrightarrow{X_\beta}$、$\overrightarrow{X_\delta}$分别表示当前狼群中$\alpha$、$\beta$、$\delta$狼的位置向量，$\vec{X}$表示当前$\omega$狼的位置向量，$\overrightarrow{D_\alpha}$、$\overrightarrow{D_\beta}$、$\overrightarrow{D_\delta}$分别表示当前$\omega$狼与最优 3 条狼的距离。

更新后狼的位置落在由α、β、δ狼的位置决定的一个圆内，即 α、β、δ狼估计猎物的位置，其他狼在猎物周围随机更新位置。

（4）进攻：$\vec{A}$ 是一个$[-a,a]$的随机向量，当$\left|\vec{A}\right|>1$，远离猎物扩大搜索范围，当$\left|\vec{A}\right|<1$，靠近猎物并攻击。

（5）探索：$\vec{C}$ 是[0,2]里的随机值构成的向量，当$\left|\vec{C}\right|>1$，增强猎物位置对候选狼更新位置的权重，$\left|\vec{C}\right|<1$，弱化猎物位置对候选狼更新位置的权重，且$\overrightarrow{r_2}$ 是[0,1]里的随机向量，有利于算法跳出局部最优。

2.3.3 鲸鱼优化算法

鲸鱼优化算法（Whale Optimization Algorithm，简称 WOA）通过模拟座头鲸群体的捕食方式：围捕、气泡网攻击、搜索猎物实现寻优。每只座头鲸的位置即对应一个潜在解，算法通过不断更新鲸鱼位置获得全局最优解。鲸鱼优化算法实现较为简单，参数少且易跳出局部最优。其围捕、气泡网收缩进攻、搜索猎物的数学模型和相关公式如下：

（1）围捕：

$$\vec{D}=\left|\vec{C}\cdot\overrightarrow{X^*}(t)-\vec{X}(t)\right| \tag{2.17}$$

$$\vec{X}(t+1)=\overrightarrow{X^*}(t)-\vec{A}\cdot\vec{D} \tag{2.18}$$

式中：t是当前迭代次数，$\vec{A}$ 和 $\vec{C}$ 是系数向量，$\overrightarrow{X^*}(t)$为迭代次数t时最优解的位置向量，$\vec{X}(t)$是候选解此时的位置向量，$\vec{X}(t+1)$是下一次迭代时的位置向量。

$\vec{A}$ 和 $\vec{C}$ 由以下公式计算：

$$\vec{A}=2\vec{a}\cdot\overrightarrow{r_1}-\vec{a} \tag{2.19}$$

$$\vec{C}=2\cdot\overrightarrow{r_2} \tag{2.20}$$

式中：$\vec{a}$ 随迭代次数增加从 2 线性减小到 0，$\overrightarrow{r_1}$ 和 $\overrightarrow{r_2}$ 是[0,1]里的随机向量。

（2）气泡网攻击：

机制一：收缩包围

在$[-1,1]$中为$\vec{A}$设置随机值，可在鲸鱼的原始位置和当前最佳位置之间的任何位置定义新位置。

机制二：螺旋更新

模拟座头鲸螺旋式运动创建以下公式：

$$\vec{X}(t+1)=\overrightarrow{D'}\cdot e^{bl}\cdot\cos(2\pi l)+\overrightarrow{X^*}(t) \tag{2.21}$$

式中：$\overrightarrow{D'}=\left|\overrightarrow{X^*}(t)-\vec{X}(t)\right|$，表示当前鲸鱼和猎物之间的距离，$b$是一个定义对数螺旋形状的常数，$l$是$[-1,1]$的随机数。

座头鲸一边收缩包围猎物，一边按螺旋形状喷出气泡，通过设置概率p决定座头鲸采用何种方式更新位置，公式如下：

$$\vec{X}(t+1)=\begin{cases}\overrightarrow{X^*}(t)-\vec{A}\cdot\vec{D} & if\ p<0.5\\ \overrightarrow{D'}\cdot e^{bl}\cdot\cos(2\pi l)+\overrightarrow{X^*}(t) & if\ p\geq 0.5\end{cases} \tag{2.22}$$

式中：p是$[0,1]$里的随机数。

（3）搜索：WOA 的搜索阶段和开发阶段不同，候选鲸鱼位置更新不依据当前最优解的位置而依据随机选取的鲸鱼位置，结合$\left|\vec{A}\right|>1$，可以扩大搜索范围，提高全局搜索性能。搜索阶段公式如下：

$$\vec{D}=\left|\vec{C}\cdot\overrightarrow{X_{rand}}-\vec{X}\right| \tag{2.23}$$

$$\vec{X}(t+1)=\overrightarrow{X_{rand}}-\vec{A}\cdot\vec{D} \tag{2.24}$$

式中：$\overrightarrow{X_{rand}}$为系统内随机选取的鲸鱼位置向量。

显然，$\vec{A}$的自适应变化使得$\left|\vec{A}\right|>1$算法进入搜索阶段，$\left|\vec{A}\right|<1$算法转入开发阶段。

2.3.4 白鲸优化算法

白鲸优化算法（Beluga Whale Optimization，简称 BWO）包含 3 个阶段：探索、开发和鲸落。每条白鲸都是一个候选解，白鲸的位置矩阵为：

$$X=\begin{bmatrix}x_{1,1} & x_{1,2} & \cdots & x_{1,d}\\ x_{2,1} & x_{2,2} & \cdots & x_{2,d}\\ \vdots & \vdots & \ddots & \vdots\\ x_{n,1} & x_{n,2} & \cdots & x_{n,d}\end{bmatrix} \tag{2.25}$$

其中：n表示白鲸数量，d表示变量维度。

每只白鲸的适应度函数值被存储在如下矩阵中：

$$F_X = \begin{bmatrix} f(x_{1,1}, x_{1,2}, \dots, x_{1,d}) \\ f(x_{2,1}, x_{2,2}, \dots, x_{2,d}) \\ \vdots \\ f(x_{n,1}, x_{n,2}, \dots, x_{n,d}) \end{bmatrix} \tag{2.26}$$

BWO 引入平衡因子B_f来调控算法在探索和开发之间的转换：

$$B_f = B_0(1 - T/2T_{max}) \tag{2.27}$$

式中：T是当前迭代次数，T_{max}是预设最大迭代次数，B_0在每次迭代中随机出自$(0,1)$，当$B_f > 0.5$，算法进入探索阶段；当$B_f \le 0.5$，算法转入开发阶段。显然，随着迭代次数T增加，B_f的变化范围由$(0,1)$缩小为$(0,0.5)$，即算法在开发阶段的概率随迭代次数增加而增大。

（1）探索：自然界中白鲸可以同步或镜像的方式游泳，搜索代理位置由一对白鲸的游泳决定，白鲸的位置更新公式如下：

$$\begin{cases} X_{i,j}^{T+1} = X_{i,p_j}^{T} + \left(X_{r,p_1}^{T} - X_{i,j}^{T}\right)(1 + r_1)\sin(2\pi r_2), & j = even \\ X_{i,j}^{T+1} = X_{i,p_j}^{T} + \left(X_{r,p_1}^{T} - X_{i,j}^{T}\right)(1 + r_1)\cos(2\pi r_2), & j = odd \end{cases} \tag{2.28}$$

式中：T是当前迭代次数，$X_{i,j}^{T+1}$代表第i只白鲸在第j维的更新后的位置，$p_j(j = 1,2,\dots,d)$是从d维中随机选择的，X_{i,p_j}^{T}表示当前第i条白鲸在第p_j维的位置，X_{r,p_1}^{T}表示第r条白鲸在第一维的当前位置，其中r是随机选的，r_1和r_2是$(0,1)$上的随机数，用于加强探索阶段，$\sin(2\pi r_2)$和$\cos(2\pi r_2)$表示镜像白鲸的鳍是朝向水面的。根据维度是偶数还是奇数，按不同策略更新位置，可反映白鲸在游泳或潜水中的同步或镜像行为。

（2）开发：BWO 引入莱维飞行策略加强全局搜索，公式如下：

$$X_i^{T+1} = r_3 \cdot X_{best}^{T} - r_4 \cdot X_i^{T} + C_1 \cdot L_f \cdot (X_r^{T} - X_i^{T}) \tag{2.29}$$

式中：T表示当前迭代次数，X_r^{T}、X_i^{T}是随机的第r、i条白鲸的当前位置，X_i^{T+1}是第i只白鲸的新位置，X_{best}^{T}是当前最优解，r_3、r_4是$(0,1)$上的随机数，$C_1 = 2r_4(1 - T/T_{max})$是一个衡量莱维飞行强度的随机跳跃因子，$L_f$是莱维飞行方程：

$$L_f = 0.05 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{2.30}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{(1+\beta)}{2}\right) \times \beta \times 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}} \tag{2.31}$$

式中：u、v是正态分布的随机数，β是常数 1.5。

（3）鲸落：迁徙、觅食过程中，少数白鲸未幸存而坠入海底，此现象即鲸落，以鲸鱼坠落概率模拟群体变化。假设白鲸要么转移至新位置，要么坠入深海，为保证白鲸总数不变，利用白鲸的位置和鲸落的步长建立位置更新公式：

$$X_i^{T+1} = r_5 X_i^T - r_6 X_r^T + r_7 X_{step} \tag{2.32}$$

式中：r_5、r_6、r_7是$(0,1)$上的随机数，X_{step}是鲸落步长，计算公式为：

$$X_{step} = (u_b - l_b) \exp\left(-\frac{C_2 T}{T_{max}}\right) \tag{2.33}$$

式中：$C_2 = 2W_f \times n$，W_f是鲸落概率，n是种群数量，u_b、l_b分别是变量的上下界，W_f是一个线性函数：

$$W_f = 0.1 - \frac{0.05T}{T_{max}} \tag{2.34}$$

显然，随着迭代次数增加，W_f从 0.1 线性减小到 0.05，即当白鲸越接近食物，鲸落概率越小。

2.4 本章小结

本章先介绍云计算的概念、特点、体系架构及任务调度模型，再介绍传统云任务调度算法和启发式调度算法的原理。通过对各类云计算任务调度算法的研究，发现群体智能算法参数简单、搜索性能好。本章重点阐述蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法这四种群体智能算法的数学模型和实现流程，为下文研究云计算任务调度中预排序对群体智能算法效率的影响做好准备。

第3章 云环境下预排序对群体智能算法性能影响的研究

本章研究云环境下任务与虚拟机的预排序对群体智能算法性能的影响。先建立云计算任务调度模型，完成任务与虚拟机的预排序；选取任务完工时间、系统负载平衡度、任务完工费用这三个指标构建适应度函数，同时评价算法性能表现。接着实现蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法，基于 Cloudsim 平台设计对照实验，比较四种算法在任务与虚拟机进行预排序和不进行预排序两种情况下的性能表现。

3.1 实验设计

建立云计算任务调度的数学模型，选取完工时间、负载平衡度、完工费用构造适应度函数，再实现蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法。

基于 Cloudsim 平台设计对照试验，设置虚拟机数目依次为 6、8，云任务数目依次为 200、400、800、1600，对应的虚拟机-任务组合共有 8 种。在每种虚拟机-任务组合下，分别实现任务与虚拟机预排序、任务与虚拟机不排序，依次调用蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法完成任务调度，输出每个算法基于任务与虚拟机排序和不排序两种场景下的完工时间、负载平衡度、完工费用函数值和综合因子大小。

通过对比并分析算法在任务与虚拟机预排序前后，完成不同虚拟机-任务组合调度的性能表现，得出结论。

表 3.1 预排序对群体智能算法性能影响的实验设置

任务与虚拟机排序与否	虚拟机数目	云任务数目	待测试算法	算法性能指标
进行预排序	6	200	蚁群算法 ACO	完工时间
		400	灰狼算法 GWO	负载平衡度
不进行预排序	8	800	鲸鱼优化算法 WOA	完工费用
		1600	白鲸优化算法 BWO	综合因子

3.2 建立云计算任务调度模型

首先定义云环境中的任务调度模型及相关公式表达。用户提交的任务和虚拟机间的映射可描述为：用户任务集是$T=\{t_1,t_2,\cdots,t_n\}$，虚拟机集合是$V=\{v_1,v_2,\cdots,v_m\}$，并且$n \gg m$，规定一个任务只允许在一台虚拟机上运行，则任务和虚拟机的关系可由矩阵TV_{map}表示：

$$TV_{map}=\begin{bmatrix} T_1V_1 & T_2V_1 & \cdots & T_nV_1 \\ T_1V_2 & T_2V_2 & \cdots & T_nV_2 \\ \vdots & \vdots & \ddots & \vdots \\ T_1V_m & T_2V_m & \cdots & T_nV_m \end{bmatrix} \tag{3.1}$$

T_iV_j表示任务t_i被分到虚拟机v_j上运行，当满足t_i被分到v_j上时，T_iV_j等于 1，否则等于 0。

本章研究云环境下任务与虚拟机的预排序对群体智能算法效率的影响，先完成用户任务、虚拟机的预排序。计算任务和虚拟机优先级的公式如下：

$$t_i = taskLength_i + taskSize_i + taskOutput_i \tag{3.2}$$

$$v_j = cp_j * 0.1 + bw_j * 0.1 + size_j * 0.05 + ram_j * 0.1 \tag{3.3}$$

式中：$taskLength_i$表示用户任务t_i的计算量，$taskSize_i$表示用户任务t_i的数据大小，$taskOutput_i$表示用户任务结果输出大小；cp_j表示虚拟机v_j的算力，bw_j为虚拟机v_j的 CPU 带宽，$size_j$表示虚拟机外存大小；ram_j表示虚拟机内存大小。

其中cp_j可由：

$$cp_j = numCPU_j \times mipsCPU_j \tag{3.4}$$

计算得到。mips 用以描述计算机运算速度，$mipsCPU_j$是虚拟机v_j单个处理器的运算速度，$numCPU_j$则是虚拟机v_j的处理器数目，二者乘积用以衡量该虚拟机算力。

将用户任务和虚拟机按以上规则完成预排序，形成一个新的任务-虚拟机映射关系，满足轻量任务被分配到低性能虚拟机上、大负荷任务被分配到高性能虚拟机上。

$Time$矩阵表示任务集在虚拟机上的预测运行时间：

$$Time = \begin{bmatrix} Time_{11} & Time_{21} & \cdots & Time_{n1} \\ Time_{12} & Time_{22} & \cdots & Time_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ Time_{1m} & Time_{2m} & \cdots & Time_{nm} \end{bmatrix} \tag{3.5}$$

该矩阵中，$Time_{ij}$表示任务t_i在虚拟机v_j的预测运行时间，运行时间由数据传输时间、计算时间两部分组成，定义为：

$$Time_{ij} = \frac{taskLength_i}{cp_j} + \frac{taskSize_i}{bw_j} \tag{3.6}$$

3.3 构建适应度函数

3.3.1 创建任务完工时间函数

虚拟机并行、满负荷不间歇处理任务，完工总时间为全体虚拟机中预测执行时间的最大值，由$Time$矩阵可计算任务完工总时间TCT(Total Completion Time):

$$TCT = max_{j=1}^{m}\left(Time_{v_j}\right) \tag{3.7}$$

$$Time_{v_j} = \sum_{i=1}^{Task_j} Time_{ij} \tag{3.8}$$

式中：$Time_{v_j}$表示分配在虚拟机v_j上的全部任务的预测运行时间之和，取所有虚拟机中该值的最大值即得到任务完工总时间。

3.3.2 创建资源负载平衡度函数

任务调度期间，系统负载平衡度对系统性能影响显著，紧密关系到 QoS 和资源利用率。针对该问题，本文引入资源负载平衡度函数，并定义为：

$$LB = \sqrt{\frac{\sum_{j=1}^{m}\left(Tasknum_{v_j} - \overline{Tasknum}\right)^2}{m}} \tag{3.9}$$

式中：m表示系统虚拟机总数目，$Tasknum_{v_j}$表示分配在虚拟机v_j上的任务总数，$\overline{Tasknum}$表示各个虚拟机上分配的任务的平均数。该式用以衡量虚拟机v_j

在处理数目为$Tasknum_{v_j}$任务过程中的整体性能稳定程度。

3.3.3 创建任务费用函数

系统完成任务的总成本TCC(Total Completion Cost)可由虚拟机运行时间与其对应单位时间费用的乘积获得：

$$TCC = \sum_{j=1}^{m} \left(Time_{v_j} \times Cost_j \right) \tag{3.10}$$

式中：$Cost_j$为虚拟机v_j的单位时间费用，计算公式为：

$$Cost_j = mipsCPU_j * 3 + bw_j * 0.1 + size_j * 0.05 + ram_j * 0.1 \tag{3.11}$$

3.3.4 构建适应度函数

本文选取的算法目标是达成用户任务完成总时间、系统负载平衡度、任务完成总费用的最小化，是一个多目标的优化问题。

多目标优化需同时满足若干目标函数，一般目标函数之间存在相互作用，一个目标函数的优化会牺牲其他目标函数的利益。本文中的云计算任务调度策略从用户任务总时间跨度TCT(Total Completion Time)、系统负载平衡度LB（Load Balance）和任务完工总费用TCC(Total Completion Cost)三方面综合评价，其数学模型为：

$$min(TCT), min(LB), min\ (TCC)$$

$min\ (TCT)$代表最小化任务完工总时间的目标函数，$min(LB)$代表最小化系统负载平衡度的目标函数,$min(TCC)$代表最小化任务完工总费用的目标函数。

本文采取解决多目标的问题优化时常用的几何平均法，转化多目标为单一目标优化问题。优化目标函数(Optimized Objective Function)可表示为：

$$Obj = \sqrt[3]{O(TCT) \cdot O(LB) \cdot O(TCC)} \tag{3.12}$$

其中：

$$O(TCT) = \frac{TCT_i - TCT_{min}}{TCT_{max} - TCT_{min}} \tag{3.13}$$

$$O(LB) = \frac{LB_i - LB_{min}}{LB_{max} - LB_{min}} \tag{3.14}$$

$$O(TCC) = \frac{TCC_i - TCC_{min}}{TCC_{max} - TCC_{min}} \tag{3.15}$$

3.4 实验过程

3.4.1 Cloudsim 简介

Cloudsim是目前运用最广泛的云计算仿真开源平台，由墨尔本大学网格实验室和Gridbus项目研发并在 2009 年推出，可提供数据中心虚拟化技术、建模和仿真虚拟化云等功能，基于该平台可完成各种调度策略可量化的测量与评估[46]。

Cloudsim体系结构如图 3.1。Cloudsim核心模拟引擎提供新的事件管理框架，支持上层软件组织；Cloudsim层实现对数据中心的建模与仿真，提供虚拟机分配至主机的调度、管理各应用程序运行与监测系统环境的功能；用户代码层是最上层，提供基本实体如主机（数量、特性）、虚拟机、用户数目、任务数目与任务类型、调度策略等，在该层通过扩展基本实体可实现基于云的场景建立。

Cloudsim中主要有以下 5 种核心类：Cloudlet、DataCenter、DataCenterBroker、Host、VirtualMachine，共同完成云环境配置、虚拟机与任务的设置及管理、实现自定义任务调度算法。

其中Cloudlet类是任务类，包含任务属性；DataCenter是数据中心，提供虚拟化技术，完成虚拟机查询与对资源的分配；DataCenterBroker完成虚拟机的新建、销毁和任务提交，透明化虚拟机管理；Host类扩展了主机参数设置范围，如CPU、内存、带宽等；VirtualMachine类实现虚拟机仿真，基于Host类运行，租用并管理相关云资源，包括CPU、内存和虚拟机内部调度策略。

3.4.2 环境配置与算法参数设置

在进行仿真实验前需完成相关实验环境配置，本实验基于JDK1.8 版本，本机硬件配置如表 3.2 所示。

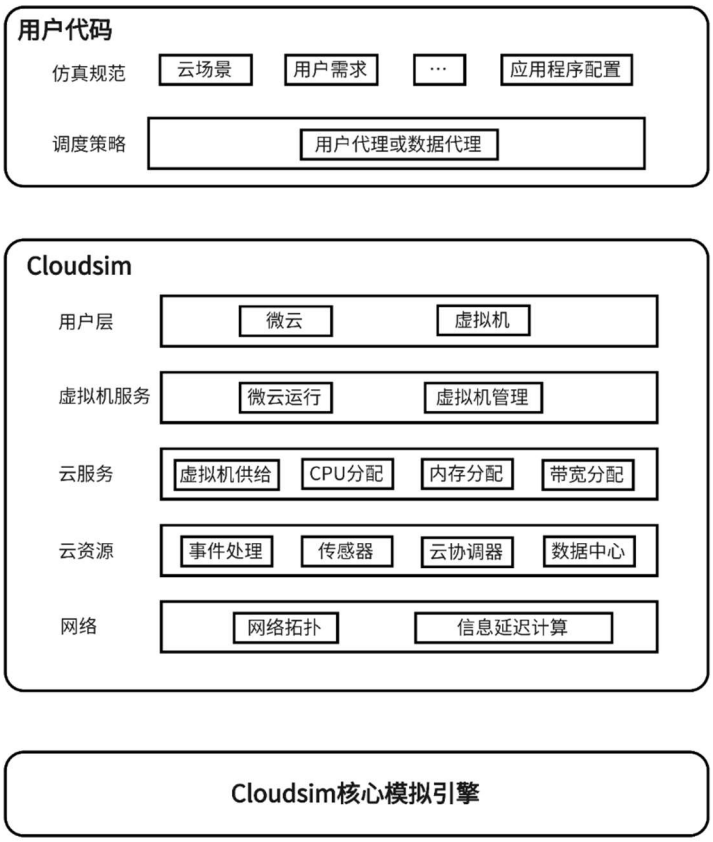

图 3.1　Cloudsim 体系结构图

基于 Cloudsim 平台，可根据具体需求设计实体参数。本实验先完成数据中心、主机、虚拟机、用户云任务这 4 个实体的设置：创建 2 个数据中心，主机与虚拟机参数设置如表 3.3，用户任务的参数设置如表 3.4。

表 3.2　本机硬件配置

参数	值
操作系统	Windows10
CPU	AMD Ryzen 7 PRO 4750U with Radeon Graphics　1.70 GHz
内存	16.0 GB

表 3.3　主机、虚拟机参数设置

参数	单位	主机	虚拟机
数量	个	2	6-8
RAM	MB	2048	256-512
带宽	bit/s	10000	500-1000
CPU核数	个	2-4	2-4
计算能力	Mips	1000	100-200
外存空间	MB	1000000	5000-10000

表 3.4　用户任务参数设置

参数	单位	用户任务
任务数据	MB	150-300
输出大小	MB	150-300
任务计算量	Mflops	500-1000

结合第二章，在实验前完成蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法相关参数设置。

(1) 蚁群算法参数设置：

ACO初始信息素设置按如下公式：

$$\tau(i,j)_0 = \omega_1 * \frac{mipsCPU_j}{\sum_{j=1}^{m} mipsCPU_j} + \omega_2 * \frac{bw_j}{\sum_{j=1}^{m} bw_j} + \omega_3 * \left(1 - \frac{Cost_j}{\sum_{j=1}^{m} Cost_j}\right) \quad (3.16)$$

式中：ω_1、ω_2、ω_3是虚拟机的处理速度、带宽、处理成本在信息素初始时的权重系数。

合理的初始信息素浓度有利于算法收敛，公式 3.16 在将任务i分配到虚拟机j上时考虑了虚拟机j的处理速度、带宽、处理成本，兼顾了虚拟机的 3 个核心性能参数。蚁群算法参数选取见表 3.5。

表 3.5　ACO 实验参数设置

参数	初始值	含义
α	1	表明信息素浓度对任务分配至虚拟机概率的影响权重
β	3	表明处理时长对任务分配至虚拟机概率的影响权重
ρ	0.5	信息素挥发因子
Q	100	信息素浓度常量
ω_1	0.3	虚拟机的处理速度在信息素初始时的权重系数
ω_2	0.3	虚拟机的带宽在信息素初始时的权重系数
ω_3	0.4	虚拟机的处理成本在信息素初始时的权重系数
$numAnt$	30	蚁群蚂蚁数目
MAX_ITER	200	预设最大迭代次数

(2) 灰狼算法参数设置见表 3.6。

表 3.6　GWO 实验参数设置

参数	值	含义
MAX_ITER	200	预设最大迭代次数
$POPULATION\_SIZE$	100	灰狼种群数目

(3) 鲸鱼优化算法参数设置见表 3.7。

表 3.7　WOA 实验参数设置

参数	值	含义
MAX_ITER	200	预设最大迭代次数
$POPULATION\_SIZE$	100	鲸鱼种群数目

(4) 白鲸优化算法参数设置见表 3.8。

表 3.8　BWO 实验参数设置

参数	值	含义
MAX_ITER	200	预设最大迭代次数
$POPULATION\_SIZE$	100	白鲸种群数目

3.5 实验结果对比

本文选取任务完工时间TCT、系统负载平衡度LB、任务完工费用TCC这三个指标评价算法性能，这三项指标的计算见公式 3.7、3.9、3.10，定义综合因子C如下：

$$C = \sqrt[3]{TCT \cdot LB \cdot TCC} \tag{3.17}$$

在主机、虚拟机、任务参数一致的前提下，基于 Cloudsim 分别调用蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法，并通过改变虚拟机数目、任务数目观测各算法性能差异，评估得出任务与虚拟机排序与否对算法的性能影响。

当虚拟机数目设置为 6 时，使任务数依次为 200、400、800、1600，基于 Cloudsim 分别调用蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法，得到算法性能数据如下。

（1）虚拟机数目为 6，任务数为 200，ACO、GWO、WOA、BWO 四种算

法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出以下结论：

① 相较于任务与虚拟机排序时的情况，任务与虚拟机不排序时 ACO、GWO、BWO 的任务完成时间均缩短、完工费用均降低。ACO 尤为显著，完成时间缩短 56.79%，完工费用降低 3.05%；

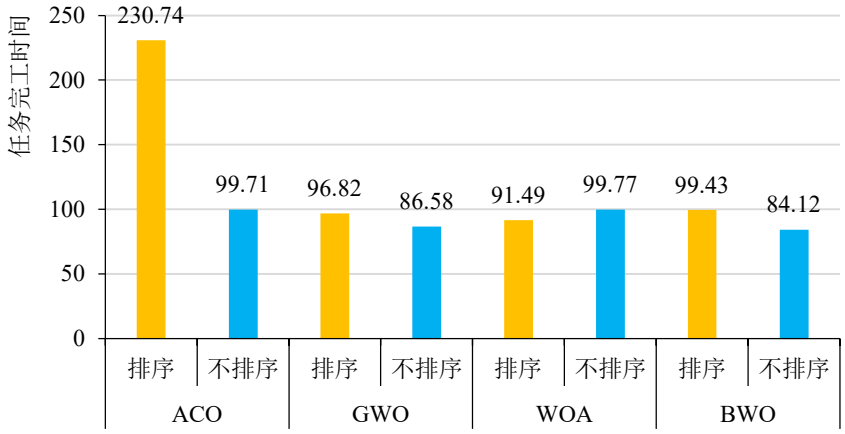

图 3.2　虚拟机数目为 6 任务数为 200 时任务与虚拟机排序与否算法完工时间对比

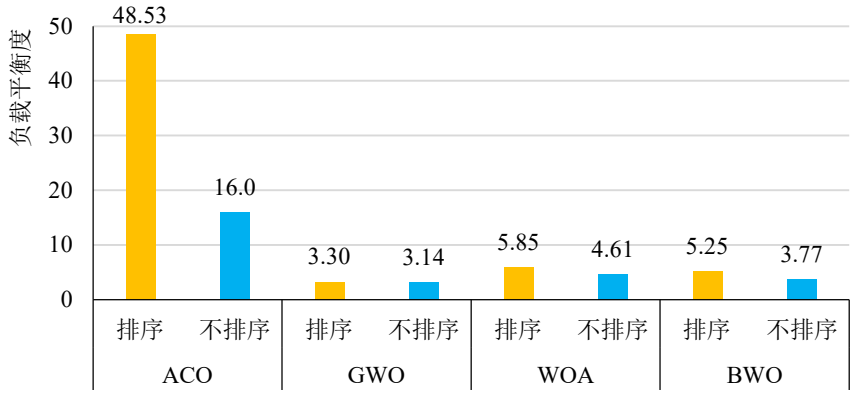

图 3.3　虚拟机数目为 6 任务数为 200 时任务与虚拟机排序与否算法负载平衡度对比

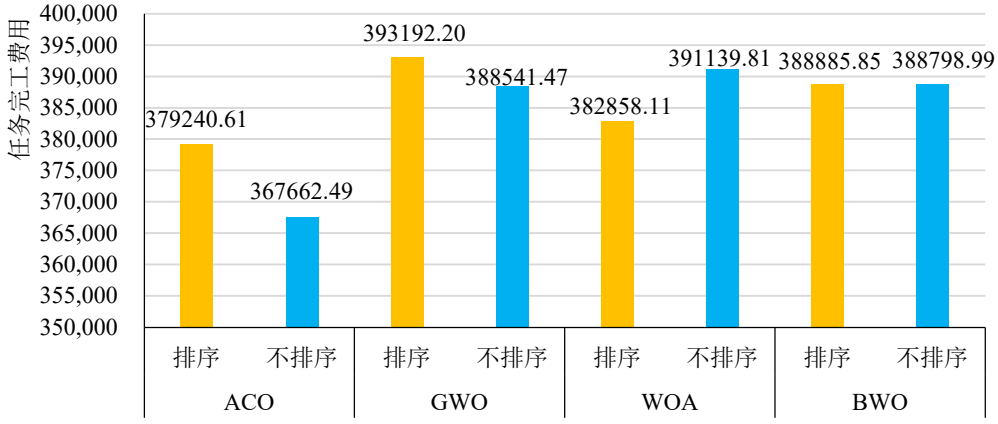

图 3.4　虚拟机数目为 6 任务数为 200 时任务与虚拟机排序与否算法完工费用对比

② 任务与虚拟机不排序时，ACO、GWO、WOA、BWO 四种算法的负载平衡度均优于排序时，说明此环境下任务与虚拟机不排序更利于云计算任务调度时的系统负载平衡；

③ 任务与虚拟机不排序时，ACO、GWO、WOA、BWO 四种算法的综合因子均低于排序时，说明任务与虚拟机不排序，算法在完工时间、费用、负载平衡三方面的综合表现更优；

④ 任务与虚拟机不排序时，虽然 WOA 的完工时间、费用略有增加，但负载平衡显著降低 21.20%，因此综合因子仍比任务与虚拟机排序时低。

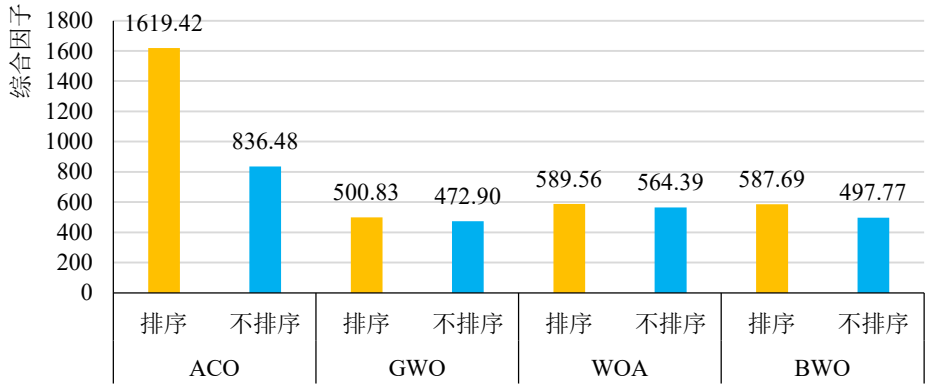

图 3.5　虚拟机数目为 6 任务数为 400 时任务与虚拟机排序与否算法综合因子对比

（2）虚拟机数目为 6，任务数为 400 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出以下结论：

① 相较于任务与虚拟机排序，任务与虚拟机不排序时 ACO、GWO、WOA、BWO 的任务完工时间均明显缩短，其中 WOA、BWO 在任务与虚拟机不排序时完工时间分别缩短 38.40%、44.32%；

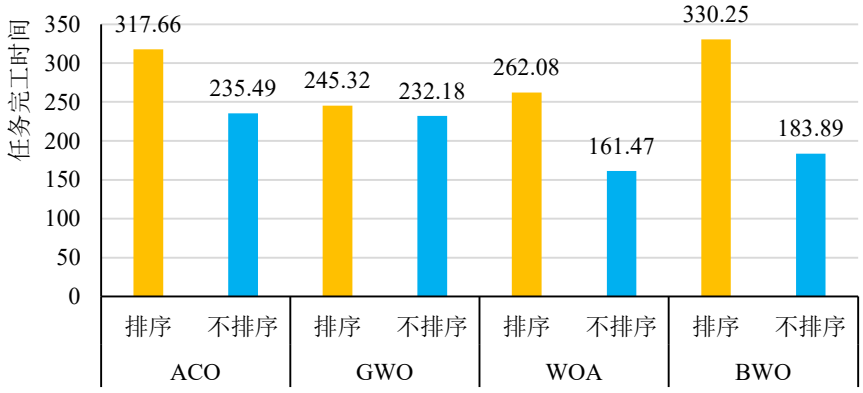

图 3.6　虚拟机数目为 6 任务数为 400 时任务与虚拟机排序与否算法完工时间对比

② 任务与虚拟机不排序时，ACO、GWO、WOA、BWO 四种算法的综合因子均低于排序时，说明任务与虚拟机不排序，算法在完工时间、费用、负载平衡三方面的综合表现更优；

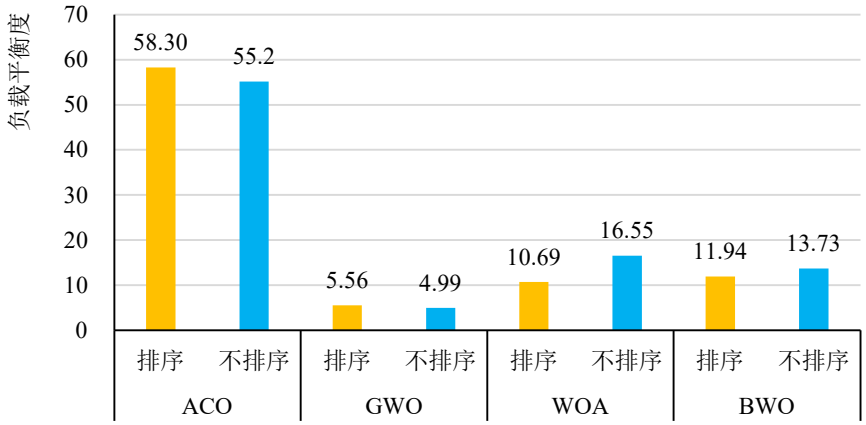

图 3.7　虚拟机数目为 6 任务数为 400 时任务与虚拟机排序与否算法负载平衡度对比

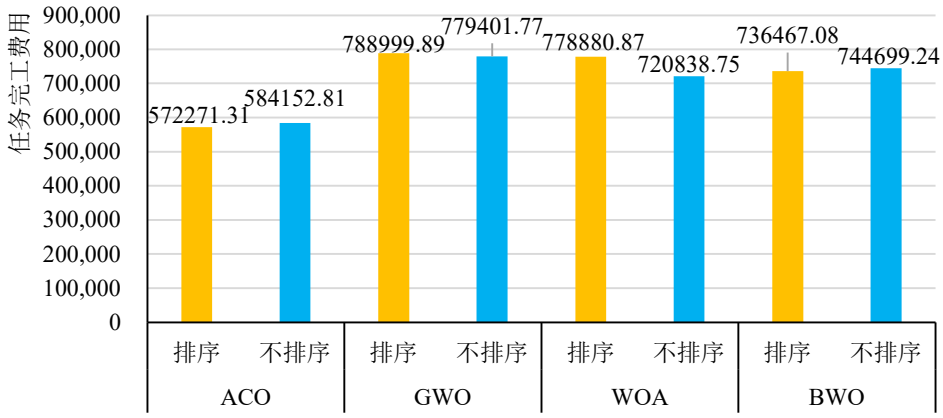

图 3.8　虚拟机数目为 6 任务数为 400 时任务与虚拟机排序与否算法完工费用对比

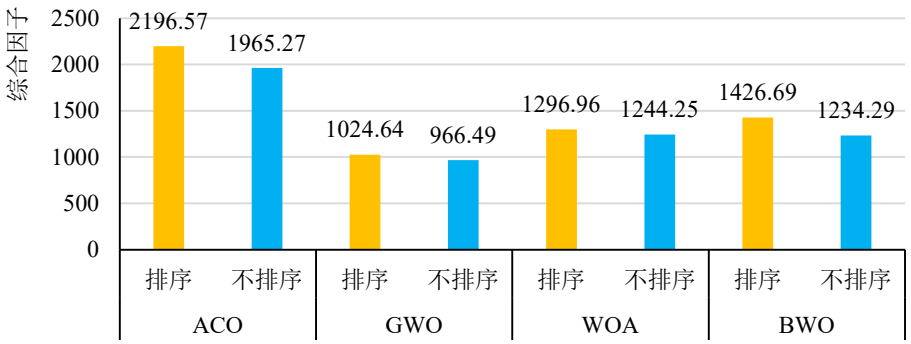

图 3.9　虚拟机数目为 6 任务数为 400 时任务与虚拟机排序与否算法综合因子对比

③ 在此环境下，当任务与虚拟机不排序时，ACO、BWO 的完工费用稍有增

加，分别增加了 2.08%、1.11%；但 GWO、WOA 的完工费用降低，分别降低了 1.22%、7.45% 。可见任务与虚拟机不排序可以缩短完工费用，且不会造成完工费用这一性能显著下降。

④ 在此环境下，当任务与虚拟机不排序时，ACO、GWO 的负载平衡度分别下降 5.35%、10.25%，WOA、GWO 的负载平衡度分别上升 54.82%、15.00%，可见任务与虚拟机不排序时，WOA、GWO 实现完工时间显著缩短的同时牺牲了负载平衡。

（3）虚拟机数目为 6，任务数为 800 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出以下结论：

① 相较于任务与虚拟机排序，任务与虚拟机不排序时 ACO、GWO、WOA、BWO 的负载平衡度均显著降低，分别降低了 15.54%、15.98%、39.82%、10.68%；

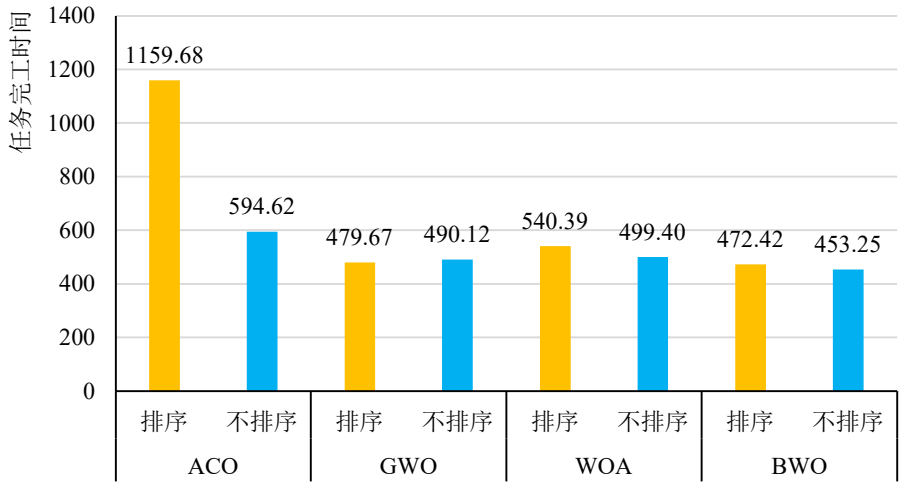

图 3.10　虚拟机数目为 6 任务数为 800 时任务与虚拟机排序与否算法完工时间对比

② 任务与虚拟机不排序时，ACO、GWO、WOA、BWO 四种算法的综合因子均低于排序时，说明此环境下任务与虚拟机不排序，算法在完工时间、费用、负载平衡三方面的综合表现更优；

③ 相较于任务与虚拟机排序，任务与虚拟机不排序时 ACO、WOA、BWO 的任务完工时间均缩短，其中 ACO 完工时间缩短 48.73%，GWO 的完工时间略微增加，增长了 2.18%；GWO、WOA、BWO 的完工费用均下降，分别下降 1.14%、1.60%、3.52%，ACO 的完工费用略有上升，增加了 1.76%；可见任务

与虚拟机不排序对缩短完工时间、降低完工成本更有效果；

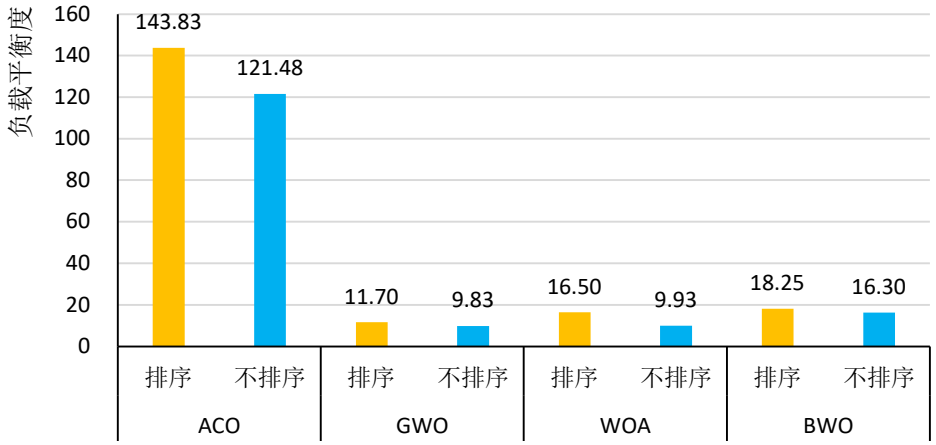

图 3.11　虚拟机数目为 6 任务数为 800 时任务与虚拟机排序与否算法负载均衡度对比

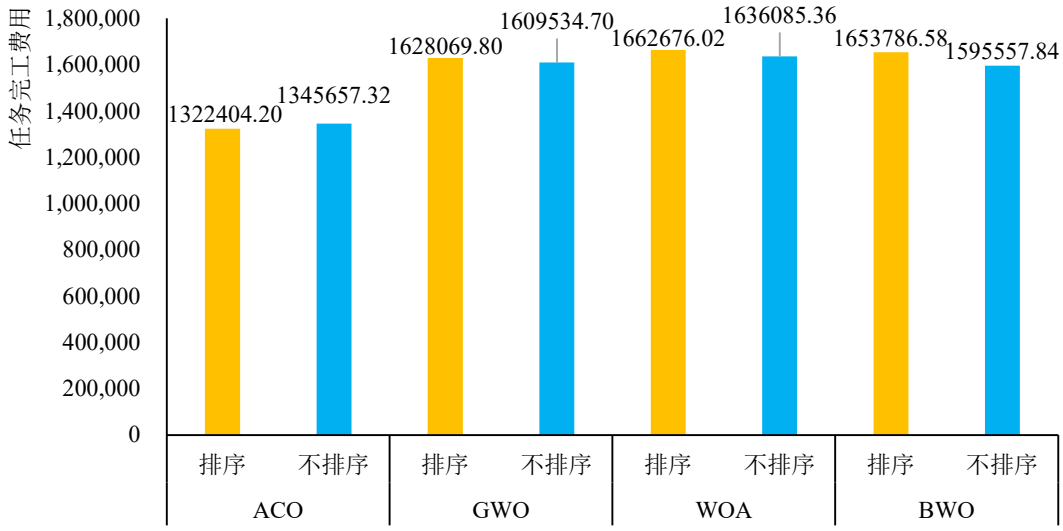

图 3.12　虚拟机数目为 6 任务数为 800 时任务与虚拟机排序与否算法完工费用对比

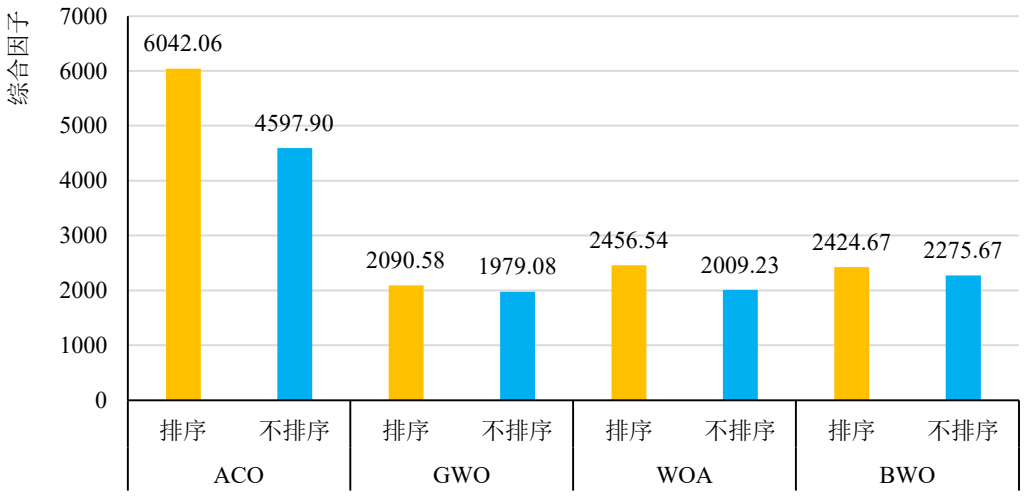

图 3.13　虚拟机数目为 6 任务数为 800 时任务与虚拟机排序与否算法综合因子对比

（4）虚拟机数目为 6，任务数为 1600 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出结论：

① 相较于任务与虚拟机预排序，不做任务与虚拟机预排序时，ACO、WOA 的完工时间显著缩短，分别缩短了 28.98%、44.44%；GWO、BWO 的完工时间略有增长，分别增加了 2.73%、21.12%；

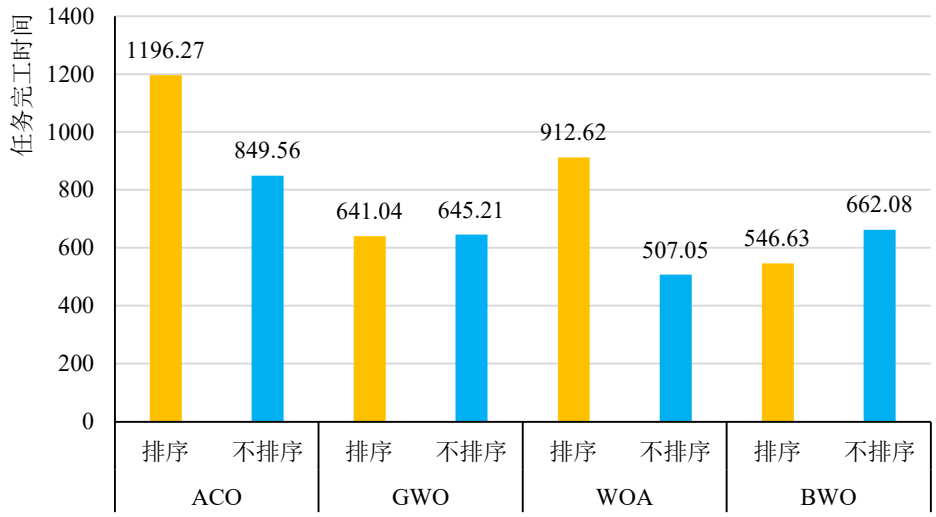

图 3.14　虚拟机数目为 6 任务数为 1600 时任务与虚拟机排序与否算法完工时间对比

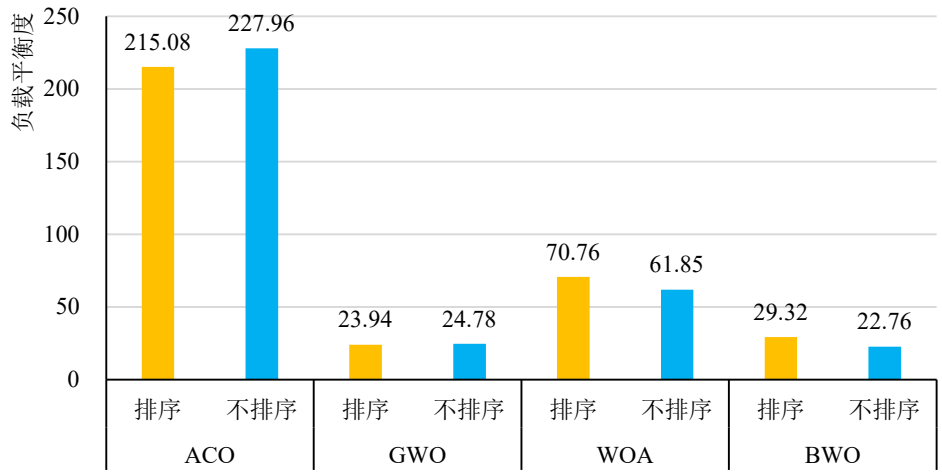

图 3.15　虚拟机数目为 6 任务数为 1600 时任务与虚拟机排序与否算法负载平衡度对比

② 相较于任务与虚拟机预排序，不做任务与虚拟机预排序时，WOA、BWO 的负载平衡值有较大改善，分别改善了 12.59%、22.36%；ACO、GWO 的负载平衡略有上升，分别上升了 5.99%、3.50%；

③ 完工费用：相较于任务与虚拟机预排序，不做任务与虚拟机预排序时，

ACO、GWO、WOA、BWO 的完工费用均下降，分别下降了 0.45%、6.46%、5.36%、1.68%。

④ 综合因子：相较于任务与虚拟机预排序，不做任务与虚拟机预排序时，ACO、GWO、WOA、BWO 的综合因子均得到改善，分别减小了 9.17%、0.86%、22.83%、2.58%。

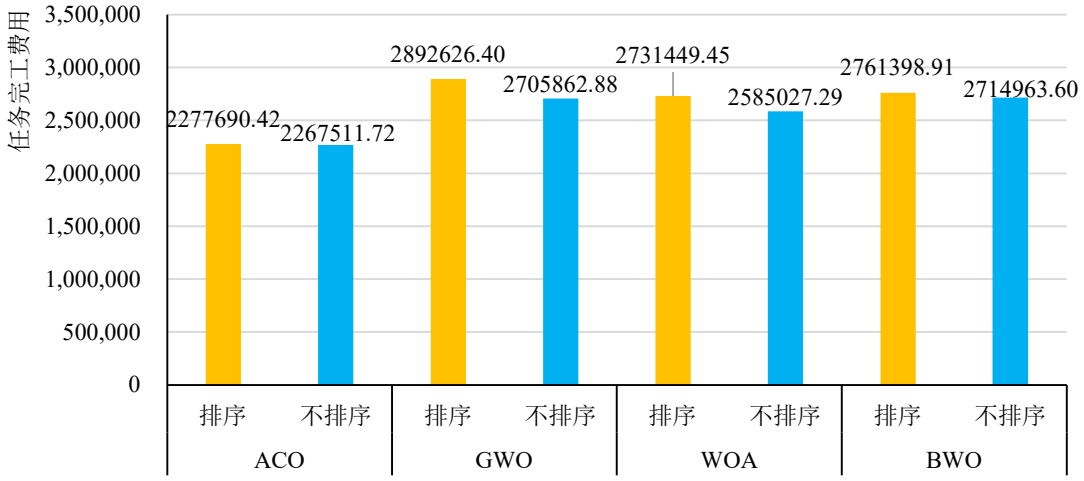

图 3.16　虚拟机数目为 6 任务数为 1600 时任务与虚拟机排序与否算法完工费用对比

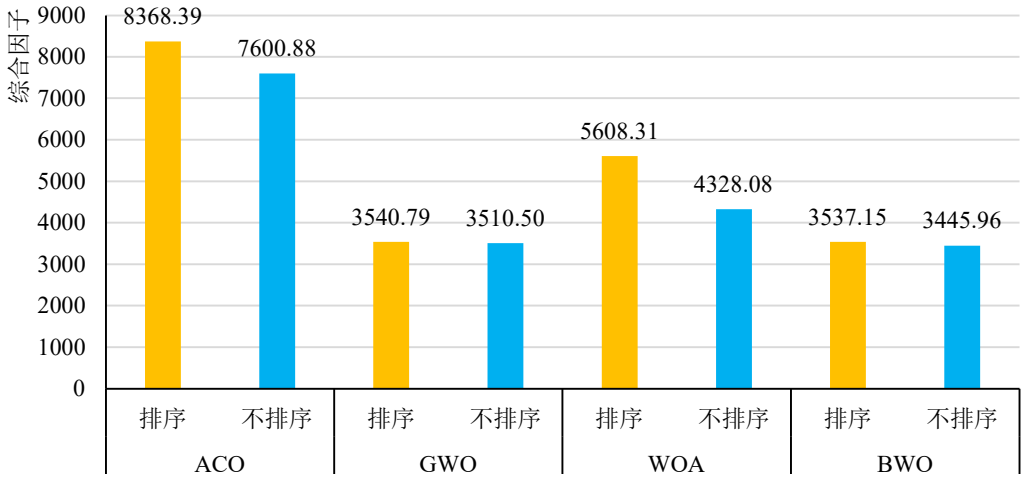

图 3.17　虚拟机数目为 6 任务数为 1600 时任务与虚拟机排序与否算法综合因子对比

当虚拟机数目设置为 8 时，使任务数依次为 200、400、800、1600，基于 Cloudsim 分别调用蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法，得到算法性能数据如下。

（1）当虚拟机数目为 8，任务数为 200 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出结论：

① 相较于任务与虚拟机预排序，不排序时四种算法的综合因子表现更好，值分别下降了 48.82%、21.12%、1.11%、5.48%；

② 相较于任务与虚拟机预排序，不排序时 ACO、GWO、WOA 的完工时间均显著缩短，分别缩短了 68.88%、24.78%、13.95%，GWO 的完工时间略有上升，增长了 2.61%；

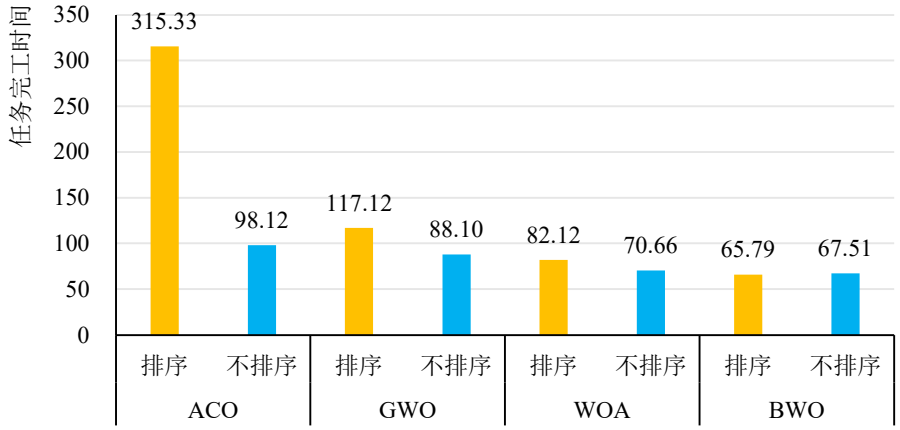

图 3.18　虚拟机数目为 8 任务数为 200 时任务与虚拟机排序与否算法完工时间对比

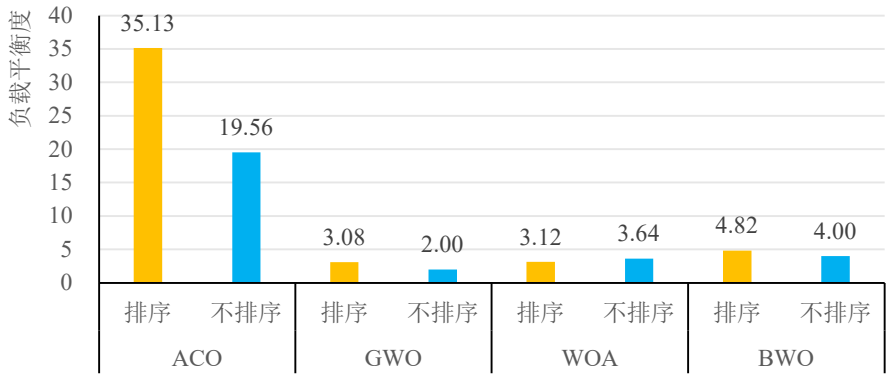

图 3.19　虚拟机数目为 8 任务数为 200 时任务与虚拟机排序与否算法负载平衡度对比

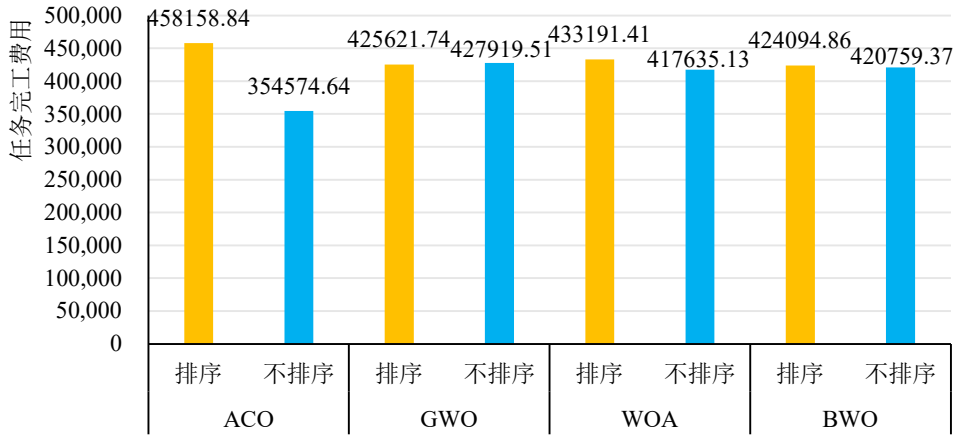

图 3.20　虚拟机数目为 8 任务数为 200 时任务与虚拟机排序与否算法完工费用对比

③ 相较于任务与虚拟机预排序，不排序时除了 ACO 的完工成本显著降低 22.61%，其他三个算法的完工成本基本不变；

④ 相较于任务与虚拟机预排序，不排序时 ACO、GWO、BWO 的负载均衡得到明显改善，值分别降低了 44.33%、35.11%、17.04%，但 WOA 的负载均衡变差，分析可知 WOA 的完工时间和费用都降低，牺牲了部分负载均衡性能；

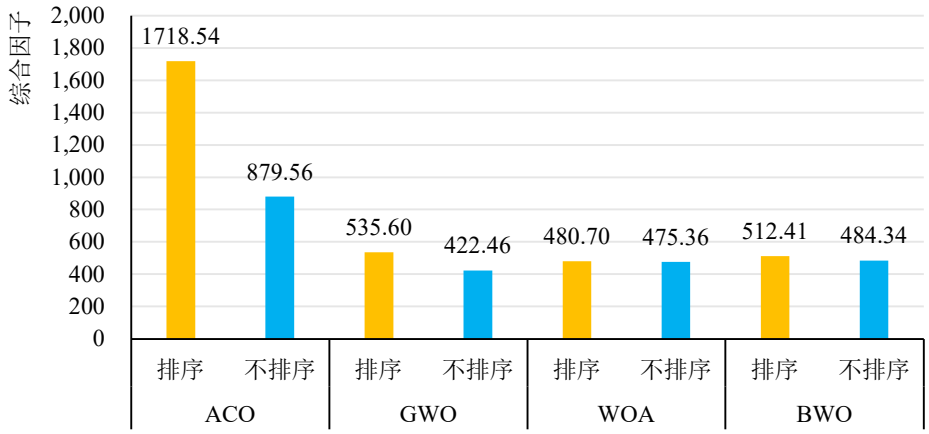

图 3.21　虚拟机数目为 8 任务数为 200 时任务与虚拟机排序与否算法综合因子对比

（2）当虚拟机数目为 8，任务数为 400 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出如结论：

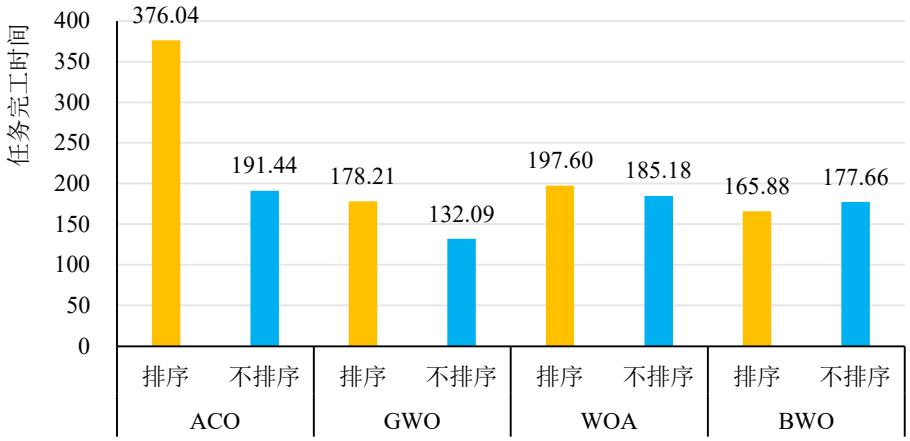

图 3.22　虚拟机数目为 8 任务数为 400 时任务与虚拟机排序与否算法完工时间对比

① 相较于任务与虚拟机预排序，不排序时四种算法的综合因子均更优；

② 相较于任务与虚拟机预排序，不排序时 ACO、GWO、WOA 的完工时间均缩短，分别缩短了 49.09%、25.88%、6.29%，BWO 的完工时间略有上升；

③ 相较于任务与虚拟机预排序，不排序时 ACO、WOA、BWO 的负载平衡度均更好，分别下降了 53.58%、15.90%、13.69%，GWO 的负载均衡度上升 27.34%，这是该算法在缩短完工时间和降低费用时做的牺牲；

④ 相较于任务与虚拟机预排序，不排序时四种算法的完工费用变化不大。

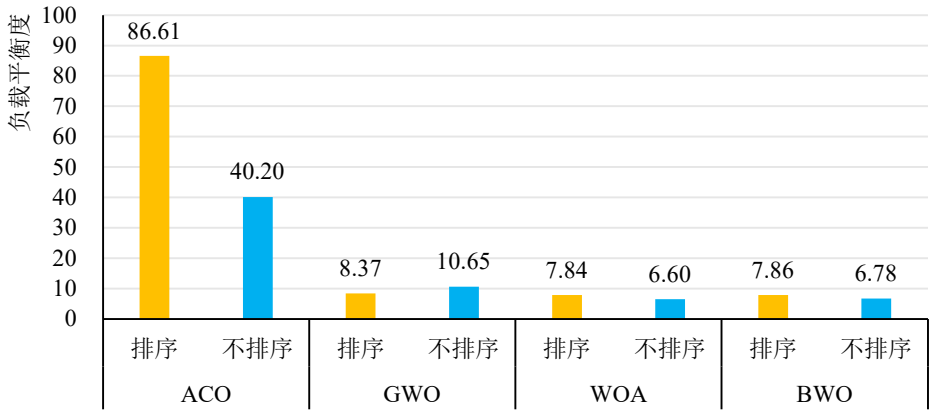

图 3.23　虚拟机数目为 8 任务数为 400 时任务与虚拟机排序与否算法负载平衡度对比

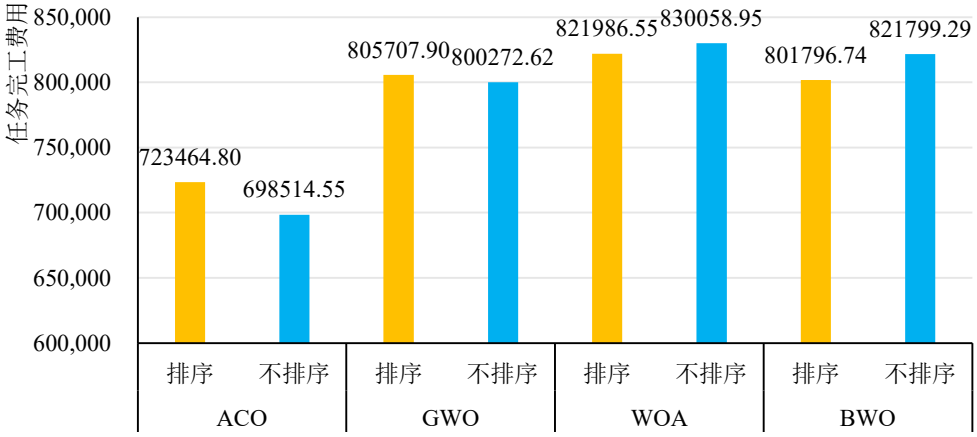

图 3.24　虚拟机数目为 8 任务数为 400 时任务与虚拟机排序与否算法完工费用对比

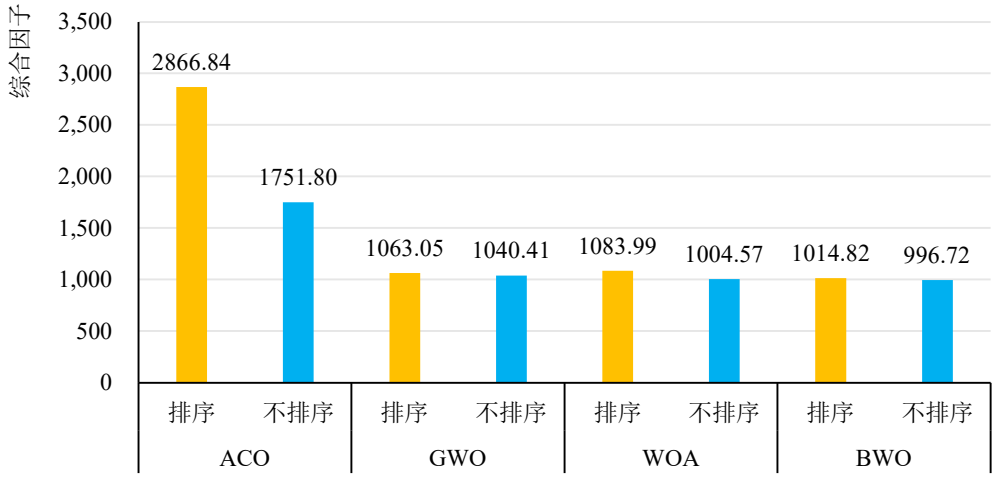

图 3.25　虚拟机数目为 8 任务数为 400 时任务与虚拟机排序与否算法综合因子对比

（3）当虚拟机数目为 8，任务数为 800 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出结论：

① 相较于任务与虚拟机预排序，不排序时四种算法的综合因子、完工时间均更优，综合因子分别下降了 13.93%、4.78%、1.39%、13.56%，完工时间分别缩短了 37.45%、5.36%、24.20%、10.81%；

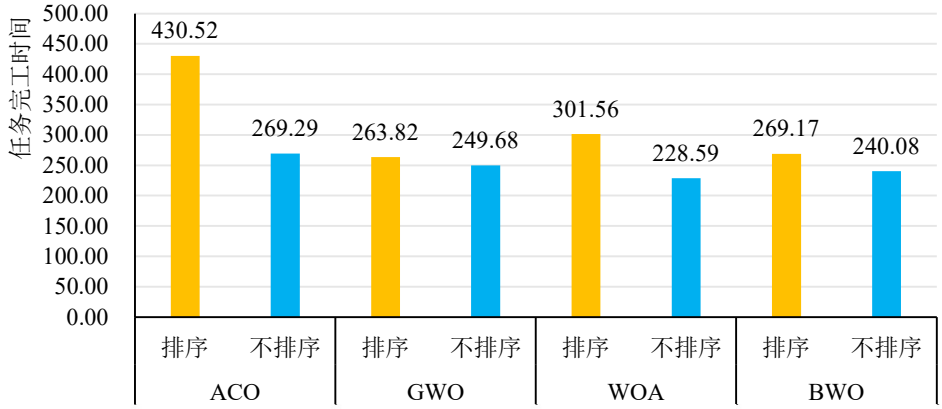

图 3.26　虚拟机数目为 8 任务数为 800 时任务与虚拟机排序与否算法完工时间对比

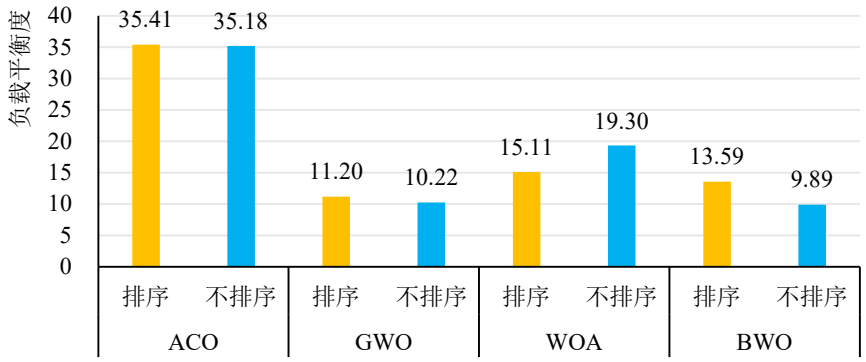

图 3.27　虚拟机数目为 8 任务数为 800 时任务与虚拟机排序与否算法负载平衡度对比

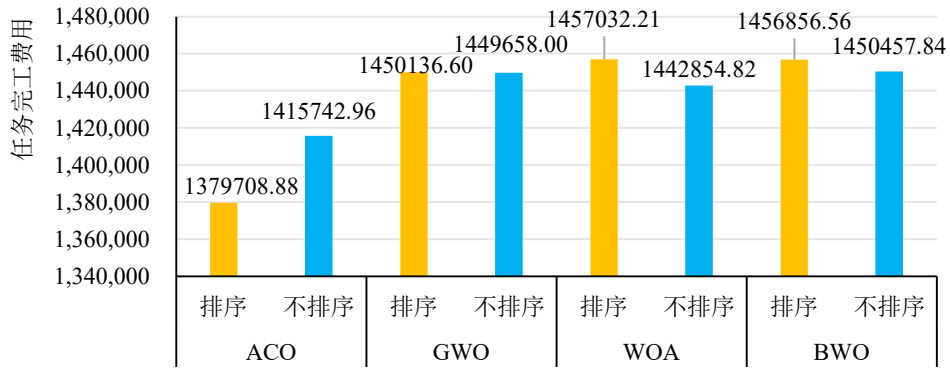

图 3.28　虚拟机数目为 8 任务数为 800 时任务与虚拟机排序与否算法完工费用对比

② 相较于任务与虚拟机预排序，不排序时 ACO 的负载平衡基本无变化，GWO 和 BWO 的负载平衡显著改善，负载平衡度值分别降低了 8.75%、27.26%，WOA 的负载平衡度值上升了 27.75%；

③ 在此环境下，任务与虚拟机排序与否对完工成本基本没有影响。

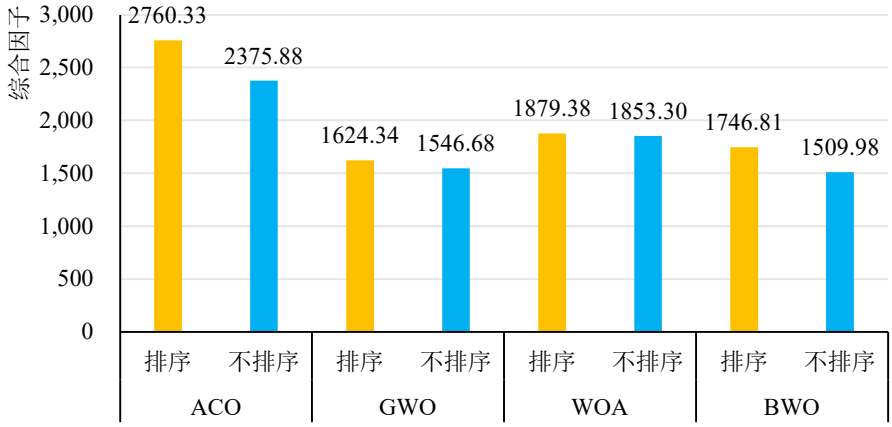

图 3.29　虚拟机数目为 8 任务数为 800 时任务与虚拟机排序与否算法综合因子对比

（4）当虚拟机数目为 8，任务数为 1600 时，ACO、GWO、WOA、BWO 四种算法的完工时间、费用、负载平衡度、综合因子对比图如下，可得出结论：

① 相较于任务与虚拟机预排序，不排序时四种算法的综合因子、完工时间、费用均更优，综合因子分别下降了 17.35%、9.99%、2.14%、13.76%，完工时间分别缩短了 14.70%、8.54%、23.30%、8.46%，费用分别降低了 9.15%、1.03%、0.97%、0.02%；

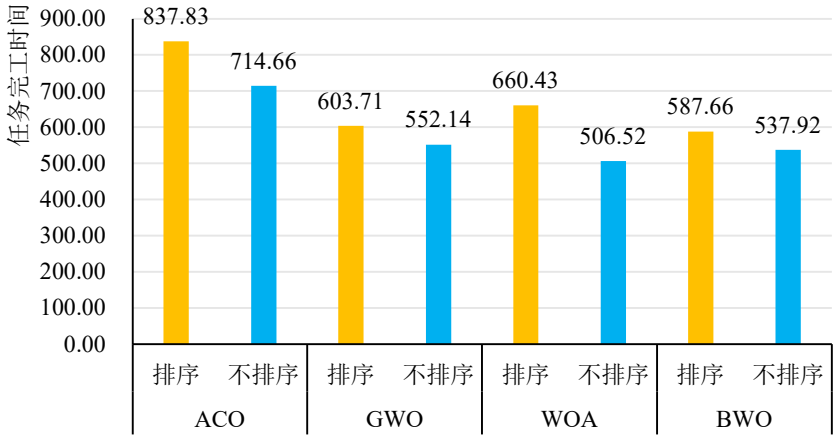

图 3.30　虚拟机数目为 8 任务数为 1600 时任务与虚拟机排序与否算法完工时间对比

② 相较于任务与虚拟机预排序，不排序时 ACO、GWO、BWO 的负载平衡度均显著更好，其值分别降低了 27.14%、19.42%、29.92%，WOA 的负载平衡

度值升高了 23.39%，由于 WOA 的完工时间显著缩短，因此综合因子仍更优。

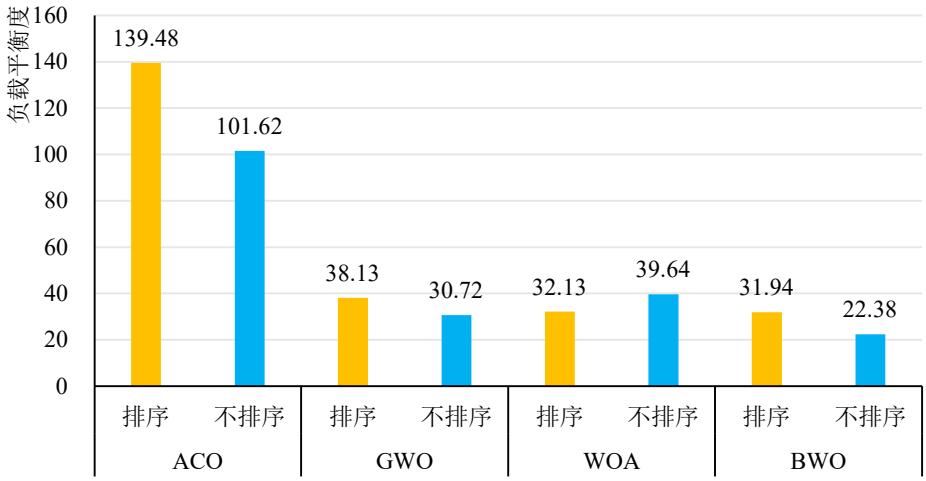

图 3.31 虚拟机数目为 8 任务数为 1600 时任务与虚拟机排序与否算法负载平衡度对比

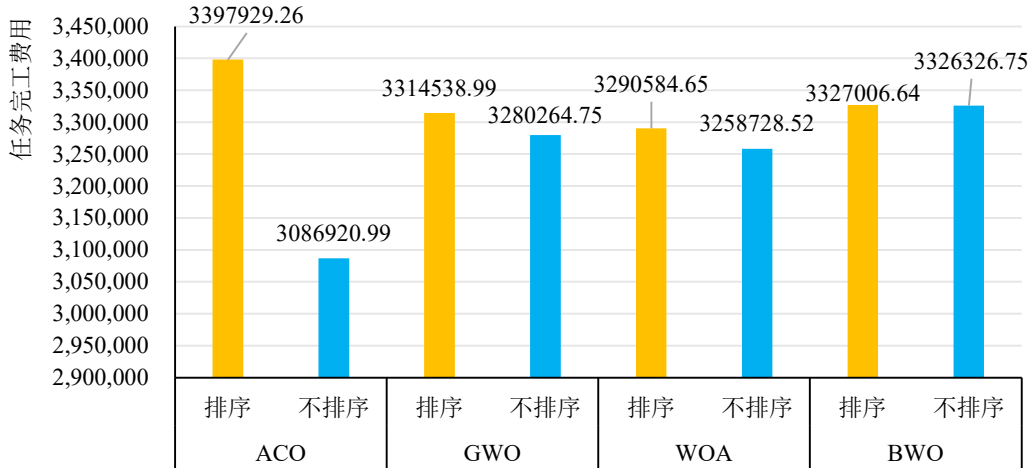

图 3.32· 虚拟机数目为 8 任务数为 1600 时任务与虚拟机排序与否算法完工费用对比

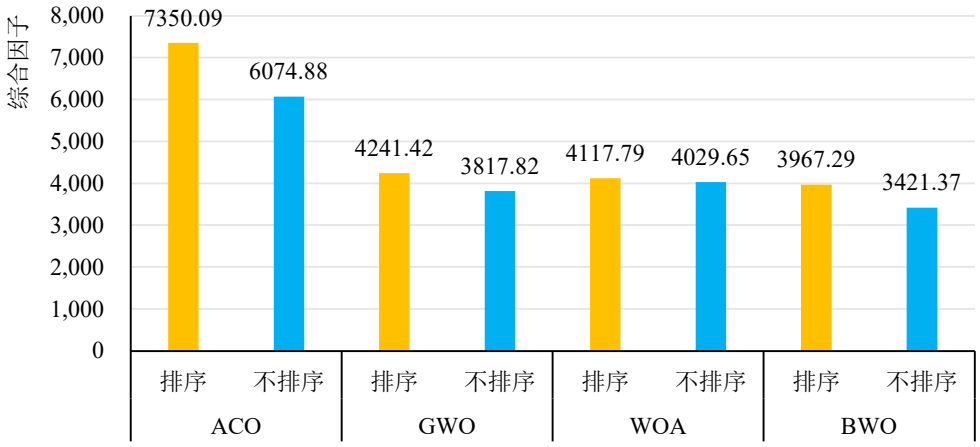

图 3.33　虚拟机数目为 8 任务数为 1600 时任务与虚拟机排序与否算法综合因子对比

3.6 实验结果分析

表 3.9 统计对比蚁群算法 ACO、灰狼算法 GWO、鲸鱼优化算法 WOA、白鲸优化算法BWO在不同虚拟机数和不同任务数组合下的完工时间，可以得出结论如下：

（1）当虚拟机数目分别设置为 6 和 8，对应任务数目分别为 200、400、800、1600 时，任务与虚拟机预排序与否对算法的完工时间影响很大，在不进行任务与虚拟机预排序时完工时间的值最多可比进行预排序时的值降低 68.88%；

（2）任务与虚拟机预排序与否，蚁群算法的完工时间都是四个算法中最长的，且波动大不稳定，有较大改进空间。

表 3.9　在不同虚拟机数和任务数下基于任务与虚拟机排序与否四种算法的完工时间对比

虚拟机数	算法	云任务数目							
		200		400		800		1600	
		排序	不排序	排序	不排序	排序	不排序	排序	不排序
6	ACO	230.74	99.71	317.66	235.49	1159.68	594.62	1196.27	849.56
	GWO	96.82	86.58	245.32	232.18	479.67	490.12	641.04	645.21
	WOA	91.49	99.77	262.08	161.47	540.39	499.40	912.62	507.05
	BWO	99.43	84.12	330.25	183.89	472.42	453.25	546.63	662.08
8	ACO	315.33	98.12	376.04	191.44	430.52	269.29	837.83	714.66
	GWO	117.12	88.10	178.21	132.09	263.82	249.68	603.71	552.14
	WOA	82.12	70.66	197.60	185.18	301.56	228.59	660.43	506.52
	BWO	65.79	67.51	165.88	177.66	269.17	240.08	587.66	537.92

表 3.10 统计对比蚁群算法 ACO、灰狼算法 GWO、鲸鱼优化算法 WOA、白鲸优化算法BWO在不同虚拟机数和不同任务数组合下的系统负载平衡度，可以得出结论如下：

（1）当虚拟机数目分别设置为 6 和 8，对应任务数目分别为 200、400、800、1600 时，任务与虚拟机预排序与否对算法的负载平衡影响很大，在不进行任务与虚拟机预排序时负载平衡度的值最多可比进行预排序时的值降低 67.03%；

（2）任务与虚拟机预排序与否，蚁群算法在任务调度时系统的负载平衡度都是四个算法中表现最差的，且蚁群算法的负载平衡波动是四种算法中最显著

的。针对蚁群算法的负载平衡性能有较大的改进和优化空间，值得进一步研究。

表 3.10　在不同虚拟机数和任务数下基于任务与虚拟机排序与否四种算法负载平衡度对比

虚拟机数	算法	云任务数目							
		200		400		800		1600	
		排序	不排序	排序	不排序	排序	不排序	排序	不排序
6	ACO	48.53	16.00	58.30	55.2	143.83	121.48	215.08	227.96
	GWO	3.30	3.14	5.56	4.99	11.70	9.83	23.94	24.78
	WOA	5.85	4.61	10.69	16.55	16.50	9.93	70.76	61.85
	BWO	5.25	3.77	11.94	13.73	18.25	16.30	29.32	22.76
8	ACO	35.13	19.56	86.61	40.20	35.41	35.18	139.48	101.62
	GWO	3.08	2.00	8.37	10.65	11.20	10.22	38.13	30.72
	WOA	3.12	3.64	7.84	6.60	15.11	19.30	32.13	39.64
	BWO	4.82	4.00	7.86	6.78	13.59	9.89	31.94	22.38

表 3.11 统计对比蚁群算法 ACO、灰狼算法 GWO、鲸鱼优化算法 WOA、白鲸优化算法BWO在不同虚拟机数和不同任务数组合下的完工费用，可以得出结论：当虚拟机数目分别设置为 6 和 8，对应任务数目分别为 200、400、800、1600 时，任务与虚拟机预排序与否对算法的完工费用这一性能影响不大，波动最大的是当虚拟机数为 8、任务数为 200 时，ACO 算法的完工成本下降了22.61%，其余算法在不同虚拟机和任务数目下的完工费用波动都在 7.5%以内。

表 3.11　在不同虚拟机数和任务数下基于任务与虚拟机排序与否四种算法完工费用对比

虚拟机数	算法	云任务数目							
		200		400		800		1600	
		排序	不排序	排序	不排序	排序	不排序	排序	不排序
6	ACO	379241	367662	572271	584153	1322404	1345657	2277690	2267512
	GWO	393192	388541	789000	779402	1628070	1609535	2892626	2705863
	WOA	382858	391140	778881	720839	1662676	1636085	2731449	2585027
	BWO	388886	388799	736467	744699	1653787	1595558	2761399	2714964
8	ACO	458159	354575	723465	698515	1379709	1415743	3397929	3086921
	GWO	425622	427920	805708	800273	1450137	1449658	3314539	3280265
	WOA	433191	417635	821987	830059	1457032	1442855	3290585	3258729
	BWO	424095	420759	801797	821799	1456857	1450458	3327007	3326326

表 3.12 统计对比蚁群算法 ACO、灰狼算法 GWO、鲸鱼优化算法 WOA、白鲸优化算法BWO在不同虚拟机数和不同任务数组合下的综合因子，可以得出结论：

（1）在不进行虚拟机与任务预排序时，ACO、GWO、WOA 和 BWO 的综合因子的值均呈现一定程度的下降，反映出群体智能算法在云任务调度中的综合性能更优；

（2）蚁群算法的综合因子明显高于其他算法，它在任务完工时间和负载均衡上存在很大改进空间，因此本文将在经典蚁群算法基础上进行优化，弥补蚁群算法在云任务调度中的不足从而提高它的整体性能；

（3）白鲸优化算法在不同虚拟机和任务数目组合下的完工时间、系统负载均衡、完工费用的综合性能表现稳定且优秀，在云计算任务调度中具有较大优势，受此启发本文将在现有白鲸优化算法基础上做改进，进一步提升它在云任务调度中的性能。

表 3.12　在不同虚拟机数和任务数下基于任务与虚拟机排序与否四种算法综合因子对比

虚拟机数	算法	云任务数目							
		200		400		800		1600	
		排序	不排序	排序	不排序	排序	不排序	排序	不排序
6	ACO	1619.42	836.48	2196.57	1965.27	6042.06	4597.90	8368.39	7600.88
	GWO	500.83	472.90	1024.64	966.49	2090.58	1979.08	3540.79	3510.50
	WOA	589.56	564.39	1296.96	1244.25	2456.54	2009.23	5608.31	4328.08
	BWO	587.69	497.77	1426.69	1234.29	2424.67	2275.67	3537.15	3445.96
8	ACO	1718.54	879.56	2866.84	1751.80	2760.33	2375.88	7350.09	6074.88
	GWO	535.60	422.46	1063.05	1040.41	1624.34	1546.68	4241.42	3817.82
	WOA	480.70	475.36	1083.99	1004.57	1879.38	1853.30	4117.79	4029.65
	BWO	512.41	484.34	1014.82	996.72	1746.81	1509.98	3967.29	3421.37

综上，基于 Cloudsim 平台完成四种群体智能算法在不同虚拟机数目和任务数目下的任务调度并对比完工时间、系统负载平衡、完工费用的综合表现，最终得出结论：在本文云环境中，不对任务与虚拟机进行预排序更利于提高群体智能算法的云计算任务调度性能。

从时间性能分析，任务与虚拟机的预排序是借助 Collections.sort()实现的，它是 Java 的 Collections 类提供的一种静态方法，用于对一个实现了 List 接口的集合完成升序排序。当集合长度小于 32，使用二分插入排序算法（用于虚拟机排序），平均时间复杂度为$O(n^2)$；否则使用优化的归并排序算法（用于任务排序），平均时间复杂度为$O(n\,log\,n)$，可见预排序额外消耗了时间，一定程度导致群体智能算法完成任务分配的整体运行时间延长。

从搜索随机性分析，对任务和虚拟机按排序算法完成预排序，可以实现任务调度时将计算量小的任务分配至算力较弱的虚拟机，计算量大的任务分配至算力较强的虚拟机，从而确保任务和虚拟机的适配，但同时也放弃了搜索随机性和更大的搜索范围，会导致任务完工时间、系统负载均衡或完工费用中的一项或多项性能下降，最终导致调度算法整体性能下降。

可见不进行任务和虚拟机的预排序，引入了更多的合理随机性和适度扩张了搜索范围，更有利于提升调度算法的综合性能，使其在完工时间、系统负载均衡、完工费用上的综合表现更优。

3.7 本章小节

本章先在云任务调度环境下实现了蚁群算法、灰狼算法、鲸鱼优化算法和白鲸优化算法，设计了任务与虚拟机排序的算法并完成排序。虚拟机数分别设置为 6 和 8，对应任务数依次设置为 200、400、800、1600，在不同虚拟机和任务数组合下基于 Cloudsim 平台调用以上四个算法，对比算法在任务完工时间、系统负载平衡、任务完工费用的综合性能，得出结论：对群体智能算法，不进行任务与虚拟机预排序处理，引入更多的合理随机性和适度扩张搜索范围更利于算法在云计算任务调度中发挥更优秀的性能。

第4章 基于云平台的群体智能算法改进

本章基于云平台先完成免疫-蚁群优化算法的设计，通过创建负载监测启发因子和动态挥发、双重奖惩的信息素更新策略，改进信息素更新方式，引入免疫算法获取蚁群算法初始信息素。接着完成改进白鲸优化算法的设计，引入模拟退火算法，利用 Metropolis 准则以一定概率接受次优解，帮助算法跳出局部最优，并在鲸落步骤后加入准反向学习策略，以扩大解空间，提高算法的全局搜索能力。

4.1 蚁群算法优化设计

基于文献调研和第三章实验结果，可以发现蚁群算法在云计算任务调度中系统负载容易失衡，并且早期搜索效率低导致任务完工时间长，但蚁群算法实现简单、应用广泛且有很大的改进空间，因此本节对蚁群算法做改进，提出一种新的免疫-蚁群优化算法 IMACO（Immune-Ant Colony Optimization），IMACO 算法的主要特点如下：

- **动态更新信息素** 引入负载监测启发因子和动态挥发、双重奖惩信息素更新策略，使得信息素的更新同时考虑完工时间和系统负载，引导任务分配至处理时间短、负载均衡好的虚拟机，从而实现算法加速收敛和负载平衡；
- **免疫操作初始化信息素** 在充分研究免疫算法的基础上，融合免疫算法和蚁群算法，利用免疫算法获得高效合理的初始信息素，改善了蚁群算法初始信息素分布不合理、搜索效率低下的问题，使得 IMACO 的收敛速度显著提高。

4.1.1 创建负载监测启发因子

创建负载监测启发因子σ如下：

$$\sigma = LB_j \cdot Time_{ij} \tag{4.1}$$

任务i分配到虚拟机j的评价指标更新规则如下：

$$\varphi'(i,j)=\frac{1}{\sigma} \tag{4.2}$$

式中：LB_j为虚拟机j的负载平衡度，$Time_{ij}$为任务i在虚拟机j上的预测运行时间，$\varphi'(i,j)$为任务i分配到虚拟机j的评价指标。

显然，当虚拟机j的负载平衡越小、处理任务i的完工时间越短，则任务i被分配给虚拟机j的概率会越大。

将负载监测启发因子σ加入转移概率函数中，使得下一跳概率不仅考虑信息素浓度、任务完工时间，也顾及系统负载均衡，定义新转移概率如下，

$$P_k'(\mathrm{i,j})=\frac{[\tau(i,j)]^{\alpha}[\varphi'(i,j)]^{\beta}}{\sum_{\mu\in J_k(i)}[\tau(i,\mu)]^{\alpha}[\varphi'(i,\mu)]^{\beta}} \tag{4.3}$$

4.1.2 创建动态挥发和双重奖惩信息素更新策略

蚂蚁进行路径搜索时会受其上残留信息素浓度引导，且信息素浓度是蚁群算法正反馈的基础，决定寻优成败。

自然界中信息素浓度会随时间流逝而不断蒸发，在算法中以信息素挥发因子ρ控制其挥发程度，ρ值过大会使搜索效率降低，ρ值过小会造成搜索能力减弱，因此挥发因子ρ的配置对算法性能至关重要。

本文引入信息素动态挥发因子，更高效地进行信息素更新，定义为：

$$\rho'=\begin{cases}\rho_{min} & \rho_k<\rho_{min}\\ \rho_k & \rho_{max}\le\rho_k<\rho_{min}\\ \rho_{max} & \rho_k\ge\rho_{max}\end{cases} \tag{4.4}$$

其中：

$$\rho_k=1-\ln\frac{Time_{v_j}+C}{Time_{v_j}} \tag{4.5}$$

式中：ρ_{min}和ρ_{max}为预设参数，$Time_{v_j}$为虚拟机j的任务完工总时间，C为常数。

ρ_k单调递增，虚拟机上的执行时间越短，ρ_k越小，此时信息素挥发因子小，达成了对优质虚拟机的信息素奖励目标；而且ρ可以保持在$[\rho_{min},\rho_{max}]$之间，平衡搜索能力和搜索效率，规避算法早熟同时拓展了解空间。

本文提出双重奖惩信息素如公式 4.6，使信息素的改变兼顾完工时间、用户花费、系统负载均衡，只有三者综合表现优秀的虚拟机才会得到信息素奖励。

$$\Delta\tau'^{t}_{k}(i,j)=\begin{cases}(1+\omega)\times\dfrac{Q}{Obj(i,j)} & Obj(i,j)<Obj_{min}\\ (1-\omega^{2})\times\dfrac{Q}{Obj(i,j)} & Obj_{min}\le Obj(i,j)<Obj_{max}\\ (1-\omega)\times\dfrac{Q}{Obj(i,j)} & Obj(i,j)\ge Obj_{max}\end{cases}\tag{4.6}$$

式中：ω为奖惩系数，Q表示信息素浓度常量，$Obj(i,j)$是算法的优化目标函数，Obj_{min}和Obj_{max}分别为算法目标函数的最小、最大值。本文目标函数综合评价任务完工时间、系统负载平衡度和任务完工费用。

全体蚂蚁完成一轮寻优后，信息素需完成全局更新，采用信息素动态挥发和双重奖惩策略更新，如公式 4.7：

$$\tau^{t+1}(i,j)=(1-\rho')\cdot\tau^{t}(i,j)+\tau'^{t}(i,j)\tag{4.7}$$

4.1.3 免疫算法初始化信息素

免疫算法（Immune Algorithm，IA）是在遗传算法基础上发展的，具有维持解的多样性的优点，克服了遗传算法未成熟就收敛、易陷入局部最优的不足[47]。本文引入免疫算法，旨在利用其全局搜索能力强、解的多样性高、不易陷入局部最优等优点用以获得蚁群的初始信息素分布，从而克服蚁群算法初始信息素匮乏、求解速度慢的不足。

免疫算法是基于模拟生物免疫系统的处理机制实现的，如：组成生物的细胞分布在身体各部分，无中央控制，可视为一个分布式自治的系统；免疫系统的多样性抗体遗传机制提供了优秀的搜索能力。

实际应用常把多目标优化问题映射至生物免疫系统，基本思想为待优化问题的目标函数视作抗原，待优化问题的全体解视作抗体[48]，通过免疫识别机制计算抗体与抗原的亲和度，亲和度最佳的抗体就是待求最优解。

经典免疫算法通过 4 个步骤实现：识别抗原、初始化抗体、评价抗体和免疫操作。本文对免疫算法进行改进以适应云计算环境特点，其流程如下。

（1）第一步，识别抗原，本文将算法的目标函数作为抗原，目标函数即公式：

$$Obj(x)=\sqrt[3]{O(TCT)\cdot O(LB)\cdot O(TCC)}\tag{4.8}$$

（2）第二步，随机初始化抗体，并完成抗体编码。

采用虚拟机-任务编码方式，随机生成若干编码种群，种群里每个抗体个体代表一个可行解，抗体长度为n，n表示待分配的任务总数，每一位的取值范围需满足$[0, m-1]$，m表示当下的虚拟机数目。

（3）第三步，亲和度、抗体浓度和激励值计算。

① 将目标函数作为亲和度函数：

$$aff(x) = \sqrt[3]{O(TCT) \cdot O(LB) \cdot O(TCC)} \tag{4.9}$$

② 抗体浓度计算方式如下：

$$Density(x_i) = \frac{1}{n} \cdot \sum_{j=1}^{n} S(x_i, x_j) \tag{4.10}$$

$$S(x_i, x_j) = \begin{cases} 0 & |aff(x_i) - aff(x_j)| < S \\ 1 & |aff(x_i) - aff(x_j)| \geq S \end{cases} \tag{4.11}$$

式中：$S(x_i, x_j)$是抗体的相似度函数，由两者目标函数值之差的绝对值大小衡量，S是相似度阈值。

③ 激励值取决于亲和度值和抗体浓度，计算方式如下:

$$Incentive(x_i) = p \cdot aff(x_i) - q \cdot Density(x_i) \tag{4.12}$$

式中：p和q是亲和度和抗体浓度的权重参数，可见激励值可以保证促进高亲和度抗体遗传，抑制高浓度抗体遗传，从而实现解的多样性。

（4）第四步，完成免疫处理，包括免疫选择、克隆、变异。

选取激励值前 30%的抗体做免疫处理，使其活化并克隆得到副本，对克隆副本进行变异操作。变异方案为：在激励值优秀的抗体群中，随机抽取抗体并随机选择抗体序列上的某位置，互相交换选定位置的值，产生新抗体。

（5）第五步，完成种群刷新。在新产生的子抗体群中，按亲和度值大小对抗体排序，选取前 20%的抗体，按下式替换概率进行随机替换：

$$P_C = \frac{aff_{child}}{aff_{parent}} \tag{4.13}$$

此时生成一个随机数，当P_C小于该随机数时，用父抗体替换子抗体，否则不替换，最终形成新一代抗体。

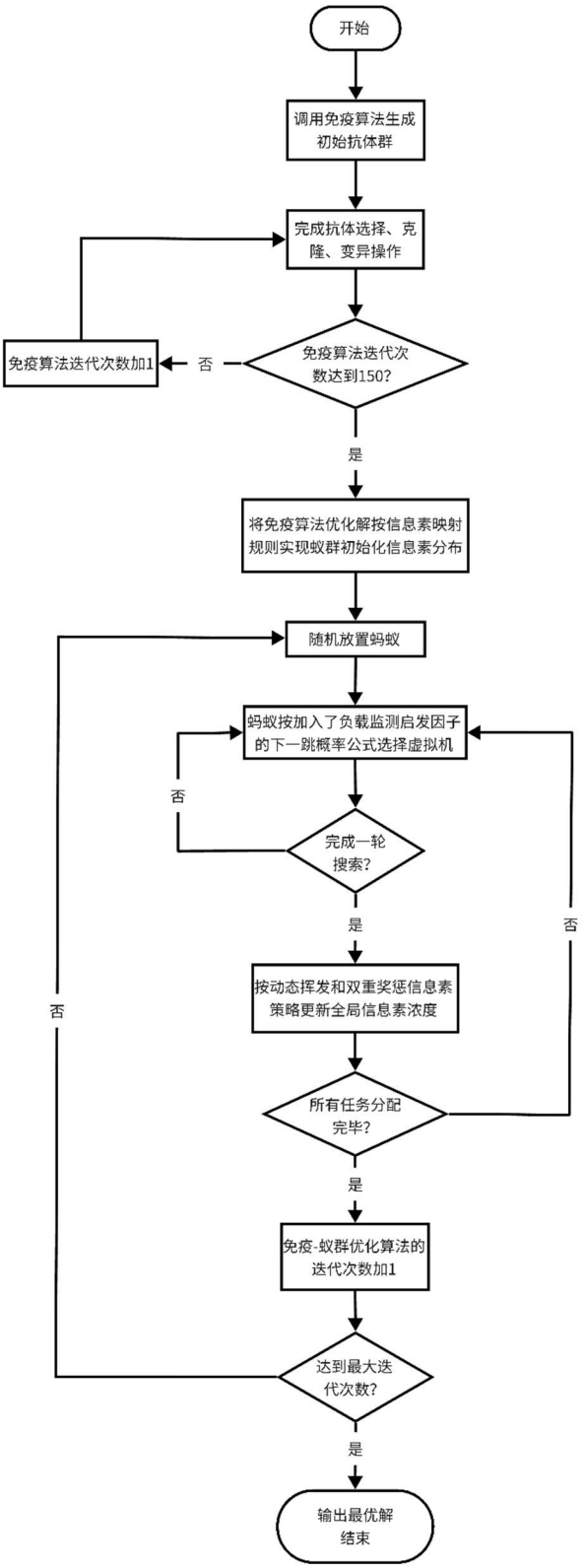

图 4.1　免疫-蚁群优化算法流程图

不同于在第二章中详细介绍过的蚁群算法，免疫-蚁群优化算法的流程图如图 4.1，流程如下：先运行免疫算法生成初始抗体群，再完成免疫操作，具体为克隆抗体群中激励值前 30%的抗体并完成变异；接着按抗体亲和度值大小排序，选取前 20%的抗体，按规定概率随机进行子代与父代的替换，得到下一代抗体群；当免疫算法迭代次数达到预设值，完成算法切换，将免疫算法获取的解按照信息素映射规则产生蚁群算法初始信息素分布，蚂蚁携带的任务根据初始信息素浓度进行虚拟机选择；完成一轮寻优后，蚂蚁根据负载监测因子和动态挥发、双重奖惩信息素更新策略改进后的全局信息素更新公式 4.7 更新信息素浓度；直至迭代次数到达免疫-蚁群算法的预设上限，退出循环、结束算法并输出最优解。

4.2 免疫-蚁群优化算法实现

本节介绍免疫-蚁群优化算法的实现，包括免疫-蚁群优化算法框架概述、免疫算法获取初始信息素、免疫算法和蚁群算法的切换条件、免疫算法得到解并映射至蚁群算法、引入负载监测因子和动态挥发双重奖惩信息素策略后的蚁群信息素更新。

4.2.1 免疫-蚁群优化算法框架概述

免疫-蚁群优化算法伪代码如算法 1 所示。首先根据公式 3.16 生成初始信息素，使得初始信息素兼顾虚拟机的处理速度、带宽和处理成本。

再调用免疫算法完成免疫-蚁群优化算法的初始信息素分布，如算法 1 中的步骤 4-10，首先生成初始抗体群，选取抗体群的优秀抗体（激励值前 30%）进行克隆操作，即保留激励值优秀的任务-虚拟机序列；接着调用算法 2 对它们完成变异操作，算法 2 在 4.2.2 介绍；再调用算法 3 完成种群刷新，操作步骤为选取亲和度值前 20%的优秀抗体按照公式 4.13 计算的概率进行父抗体与子抗体的替换，当父抗体与子抗体的差异越大，替换的概率越大，扩大了解范围，算法 3 在 4.2.2 介绍。

当免疫算法达到预设迭代次数后，需切换入蚁群算法，此时利用公式 4.14

完成免疫算法获得初始信息素到蚁群算法的映射，开始蚁群算法寻优，步骤 10 实现了利用免疫算法解的多样性高的优点使得蚁群初始信息素浓度合理。

步骤 11-20 切换入蚁群算法寻优，在一轮任务-虚拟机分配完成后，调用算法 4 更新全局信息素浓度，算法 4 在 4.2.5 介绍。当免疫-蚁群优化算法达到预设迭代次数，算法退出循环并输出当前最优解。

表 4.1　免疫-蚁群算法伪代码

算法 1 免疫-蚁群优化算法

输入：任务和虚拟机数目、免疫算法和蚁群算法相关参数

输出：任务与虚拟机分配的最优方案

1：　初始化参数：任务数为 n，虚拟机数为 m，蚂蚁数为 k，蚁群算法最大迭代次数为

2：　T_{max}，信息素挥发因子ρ的上下界分别为ρ_{min}和ρ_{max}

3：　按公式 3.16 生成初始信息素

4：　调用免疫算法，完成任务-虚拟机随机分配

5：　**while** $iter_{IA} < 150$ **then**

6：　　调用算法 2，完成变异操作，获得变异后的子代抗体

7：　　调用算法 3，完成种群刷新，获得新一代抗体

8：　　$iter_{IA} = iter_{IA} + 1$

9：　**end while**

10：按公式 4.14 将免疫算法得到的优化解按信息素映射规则实现蚁群初始化信息素分布

11：随机放置 k 只蚂蚁；

12：**while** $iter_{IMACO} < T_{max}$ **then**

13：　**for** $i = 1,2,\cdots,k$ **do**

14：　　蚂蚁按公式 4.3：$P_k'(\mathrm{i,j}) = \frac{[\tau(i,j)]^{\alpha}[\varphi'(i,j)]^{\beta}}{\sum_{\mu \in J_k(i)}[\tau(i,\mu)]^{\alpha}[\varphi'(i,\mu)]^{\beta}}$计算下一跳概率并选择虚拟机

15：　**end for**

16：　计算全体分配方案的目标函数值，记录Obj_{max}和Obj_{min}

17：　调用算法 4，完成动态信息素更新

18：　**while** k只蚂蚁完成 n 个任务的分配 **then**

19：　　$iter_{IMACO} = iter_{IMACO} + 1$

20：　**end while**

21：**end while**

22：**return** 任务与虚拟机分配的最优方案

4.2.2 免疫算法获取初始信息素

免疫算法获取蚁群初始信息素需实现两个核心步骤：变异和种群刷新。变异操作伪代码如算法 2，该操作充分利用免疫算法的优势，通过交叉变异丰富了子代抗体多样性并扩大了解空间，利于跳出局部最优，为后续切换入蚁群算法寻优准备了合理高效的初始信息素。

变异操作方法为从激励值前 30%的优秀抗体中随机抽取任务-虚拟机序列，并互换任务分配的虚拟机，假如某任务-虚拟机序列片段为任务 1 被分配至虚拟机 2，任务 2 被分配至虚拟机 1，则互换后变为任务 1 被分配至虚拟机 1，任务 2 被分配至虚拟机 2，重复该过程直至激励值前 30%的抗体均完成变异。

表 4.2 变异操作伪代码

算法 2 变异操作算法

输入：父代抗体

输出：变异后的子代抗体

1： 父代抗体数为 N，按公式 4.12 计算父代抗体激励值并完成升序排序，选取前 30%的

2： 优秀抗体，数目为 $0.3N$

3： **for** $i = 1, 2, \dots, 0.3N$ **do**

4： $p = randChoose(0.3N)$ // 随机选取父代抗体p

5： $p_i = randChoose(p)$ // 随机选取父代抗体p交换的位置i

6： $p_j = randChoose(p)$ // 随机选取父代抗体p交换的位置j

7： $swap(father[p_i], father[p_j])$

8： **end for**

9： **return** 变异后的子代抗体

种群刷新操作伪代码如算法 3，该步骤可有效避免代表局部最优的子代抗体大量被遗传至下一代抗体中。步骤 3 选取目前抗体群中亲和度值优秀的抗体，它们很可能是局部最优解，步骤 5-9 规定了子抗体待交换的片段，步骤 11-15 实现了父抗体和子抗体的片段替换，抑制了具有优秀亲和度值的抗体直接传至下一代的概率。

表 4.3 种群刷新操作算法伪代码

算法 3 种群刷新操作算法

输入：变异后的抗体群

算法 3 种群刷新操作算法（续）

输出：新一代抗体

按公式 4.9 计算变异后子代抗体亲和度值并完成升序排序，选取前 20%的优秀抗体
for $i = 1, 2, \dots, 0.2N$ **do**
    $p = randChoose(0.2N)$ // 从亲和度值前 20%的抗体中随机选取子抗体p
    $q = randChoose(N)$ // 随机选取父抗体q
    按公式 4.13 计算：$P_C = \frac{aff_p}{aff_q}$
    $p_i = randChoose(p)$ // 随机选取子抗体p待交换的起始位置
    $p_j = randChoose(p)$ //随机选取子抗体p待交换的终止位置
    if $p_i > p_j$ **then**
        swap(p_i, p_j)
    end if
    生成随机数 r
    if $P_C < r$ **then**
        for $n = i, i+1, i+2, \dots, j-1, j$ **do**
            $swap[p_n, q_n]$
        end for
    end if
end for
return 新一代抗体

4.2.3 免疫算法和蚁群算法的切换

处理混合启发算法的一般做法是为第一种算法设置固定迭代次数，当其迭代次数达到预设值后，切换入另一算法。该策略的难点在于如何设置合理迭代次数，若该值较小，则免疫算法结束过早，尚未得到较优解就转入蚁群算法，达不到信息素早期合理分布的目的；若该值过大，免疫算法将冗余迭代，导致算法效率降低。

为充分利用免疫算法和蚁群算法各自优点，需保证在免疫算法接近收敛时才切换入蚁群算法。基于此，本文采用单一变量控制法进行多轮对比试验，选出了一个适合于本文提出的免疫-蚁群优化算法的迭代次数值。

验证步骤如下：根据前期对比实验结果，免疫算法表现优良的迭代次数处于（50，200）之间，梯度设置免疫算法迭代次数为 50，100，150，200，并在此4个迭代次数下基于Cloudsim调用免疫算法分别完成200、400、800、1600的任务，得出结论：任务数分别为 200、400、800、1600 时，免疫算法迭代次数为 150 时的任务完工时间最短、完工费用最低、综合因子最佳，利于高效获得蚁群初始信息素。因此，本文设置免疫算法和蚁群算法的切换条件为免疫算法迭代次数达到150。

4.2.4 免疫算法解映射至蚁群算法

当免疫算法迭代次数达到预设值，需切换到蚁群算法，并将免疫算法获取的较优解映射为蚁群的初始信息素分布。

免疫算法编码已经实现了免疫算法获取的每一个解均与蚁群算法中蚂蚁的一条路径对应。τ_I是由免疫算法得到的解映射至蚁群算法的信息素浓度，计算方法与蚁群算法里信息素更新规则类似，映射到蚁群算法的初始信息素τ_0公式如下：

$$\tau_0 = \frac{4}{3}\tau_I \tag{4.14}$$

4.2.5 动态信息素更新

切换到蚁群算法后的寻优核心步骤是信息素的更新，本文为改进蚁群算法搜索时导致的系统负载严重不均引入动态信息素更新策略。挥发因子ρ被限制在ρ_{min}和ρ_{max}之间，且虚拟机执行时间越短，ρ_k越小，抑制了优质虚拟机的信息素挥发。

当任务i到虚拟机j分配方案的目标函数值低于上一轮最小目标函数值Obj_{min}，信息素增量得到奖励；当任务i到虚拟机j分配方案的目标函数值大于上一轮最大目标函数值Obj_{ma}，信息素增量被削减；当任务i到虚拟机j分配方案的目标函数值介于Obj_{min}和Obj_{max}之间，小幅度削减信息素增量，调控因子ω一般取0.5，从而实现双重奖惩信息素更新。

表 4.4 动态信息素更新算法伪代码

算法 4 动态信息素更新算法

输入：由免疫算法获得的解映射到蚁群的初始信息素浓度、算法 1 步骤 16 计算得到的Obj_{max}和Obj_{min}

输出：更新后的全局信息素

1： 计算信息素动态挥发因子ρ

2： **if** $\rho < \rho_{min}$ **then**

3： $\rho = \rho_{min}$

4： **else if** $\rho > \rho_{max}$ **then**

5： $\rho = \rho_{max}$

6： **else** $\rho = 1 - \ln\frac{Time_{v_j}+C}{Time_{v_j}}$

7： **end if**

8： **if** $Obj(i,j) < Obj_{min}$ **then**

9： $\Delta\tau'^{t}_{k}(i,j) = (1+\omega) \times \frac{Q}{Obj(i,j)}$ // 计算信息素增量，奖励优秀分配方案

10： **else if** $Obj(i,j) > Obj_{max}$ **then**

11： $\Delta\tau'^{t}_{k}(i,j) = (1-\omega) \times \frac{Q}{Obj(i,j)}$ // 计算信息素增量，抑制差的分配方案

12： **else** $\Delta\tau'^{t}_{k}(i,j) = (1-\omega^2) \times \frac{Q}{Obj(i,j)}$

13： **end if**

14： 利用公式 4.7 完成信息素更新$\tau^{t+1}(i,j) = (1-\rho') \cdot \tau^{t}(i,j) + \tau'^{t}(i,j)$

15： **return** 更新后的全局信息素

4.2.6 算法复杂度分析

在 IMACO 算法中，时间复杂度主要取决于两方面。一是用免疫算法获取蚁群初始信息素分布，其时间复杂度为$O(iter_{IA} * N * n^2)$，其中$iter_{IA}$表示免疫算法的迭代次数，如 4.2.3 介绍，设置为 150，N表示免疫算法抗体种群数，n表示待处理任务数。

二是切换到蚁群算法生成最终任务分配方案，其时间复杂度为$O(T_{max} * n * m)$，T_{max}为蚁群算法最大迭代次数、n为待处理任务数、m为虚拟机数目。综上，IMACO 算法的时间复杂度为$O(iter_{IA} * N * n^2 + T_{max} * n * m)$。

IMACO 的空间复杂度主要用于存储免疫算法初始分配方案、父代和子代抗体对应分配方案、以及蚁群算法各任务-虚拟机最终分配方案，因此其空间复杂

度为$O(iter_{IA} * N * n + n * m)$。

4.3 白鲸优化算法的优化设计

根据第三章实验结果，可以看到白鲸优化算法在云任务调度中的完工时间、负载均衡、完工费用性能表现优异而且稳定，但也存在易陷入局部最优的问题，因为白鲸优化算法在更新白鲸位置时，只有当新位置的适应度值优于原位置，该新位置才会被接受。因此本节引入模拟退火算法，利用 Metropolis 准则计算概率并以此概率接受次优解，使算法跳出局部最优，同时在白鲸优化算法的鲸落步骤后加入准反向学习策略，避免单向搜索，扩大解范围，提高算法的全局搜索能力。

4.3.1 模拟退火算法

模拟退火（Simulated Annealing，SA）算法是一种通过模拟金属降温的热力学过程解决局部最优的启发式算法，广泛用于组合优化问题。

SA 算法求解过程如下：

（1）初始化温度，产生随机初始解并计算其目标函数值。一般初始温度设置得较高，以便算法在解空间进行大规模搜索，初始解满足公式 4.15：

$$\exp\left(-\frac{\Delta f}{T_0}\right) \approx 1 \tag{4.15}$$

式中：Δf表示目标函数差值，T_0表示初始温度，T_0足够大时可以使算法跳出局部最优并搜索到全局最优解，但T_0过大会使算法迭代次数太多导致收敛速度慢。

（2）在该温度下引入扰动，产生新解并计算其目标函数值，当新解目标函数值小于初始解目标函数值，以概率 1 接受新解，反之根据 Metropolis 准则计算的概率接受新解，重复本步骤直到达到该温度的平衡态。

Metropolis 准则的原理为，设状态 i 为当前状态，该状态下系统能量为 E_i，对该状态做随机扰动，获得新状态 j，新状态的系统能量为 E_j，如果 $E_j < E_i$，那么 j 就是重要状态，如果 $E_j > E_i$，则需根据计算的概率判断 j 是否是重要状态，概率计算公式如 4.16：

$$r = \exp(\frac{E_i - E_j}{k_b T}) \quad r < 1 \tag{4.16}$$

式中：k_b表示波兹曼常数，T表示绝对温度。

将 r 值与[0,1]之间的随机数比较，若 r 值较大，则接受 j 为重要状态，并取代 i 成为当前状态，重复以上新状态的生成过程直到寻得最优解。

可见 Metropolis 准则可以以一定概率放弃局部最优解，协助算法跳出局部最优，利用这一特性可以对白鲸优化算法易陷入局部最优的不足做改进。

（3）达到终止条件时输出最优解，当满足初始温度足够高和温度降低地足够慢，SA 算法可收敛到全局最优。它以 Metropolis 准则计算的概率接受较差点，具备了跳出局部最优的显著优点。

改进白鲸优化算法的优化目标为任务完工时间、系统负载平衡、任务完工费用的综合最佳，即向下优化。引入模拟退火算法，设计白鲸更新位置时的规则为：

（1）当潜在新位置的优化目标函数值低于当前位置，则让当前白鲸下一步到潜在新位置，完成位置更新；

（2）当潜在新位置的优化目标函数值高于当前位置，引入 Metropolis 准则提出较差位置接受概率计算公式 p，以指导当前白鲸的下一步走向：

$$p = \exp\left[\frac{Obj_T - Obj_{T+1}}{(1-C)^T * Temperature}\right] \tag{4.17}$$

式中：Obj_T为当前白鲸所在位置的目标函数值，Obj_{T+1}为新位置的目标函数值，T为当前迭代次数，C为模拟退火参数冷却率，$Temperature$为模拟退火的初始温度。

随着迭代次数增加，较差位置接受概率 p 会逐渐减小，这一机制有助于改进白鲸优化算法在寻优早期跳出局部最优。

4.3.2 准反向学习策略

Tizhoosh[49]首先提出基于反向的学习（Opposition Based Learning, 简称为 OBL），反向学习是在明确变量上下限基础上，按规则求其对应反向解，具体过

程为：假设白鲸种群数为N，搜索空间为d维，第i只白鲸在第d维空间的位置表示为：

$$X_i = \left(x_i^1, x_i^2, \cdots, x_i^d\right) \tag{4.18}$$

式中：

$i = 1,2, \ldots, N$

$x_i^j \epsilon \left[a_i^j, b_i^j\right]$，$j = 1,2, \ldots, d$；$a_i^j, b_i^j$分别表示$x_i^j$的下界和上界。

定义x_i^j对应反向解为：

$$\hat{x}_i^j = a_i^j + b_i^j - x_i^j \tag{4.19}$$

为进一步扩大搜索范围，Rahnamayan 等人[50]在反向学习的基础上提出了准反向学习（Quasi-Opposition-Based Learning, 简称为 QOLB），在寻找全局最优解时，准反向学习策略比反向学习更有效，基于本文需求设计准反向解的计算方式如下：

$$\widehat{x'}_i^j = \frac{a_i^j + b_i^j}{2} + \left(\hat{x}_i^j - \frac{a_i^j + b_i^j}{2}\right) \times rand(0,1) \tag{4.20}$$

式中：$rand(0,1)$为$(0,1)$之间的随机数。

本文在白鲸优化算法的鲸落步骤后引入准反向学习策略，有利于改进白鲸优化算法扩大搜索范围，加速寻得全局最优解。

不同于第二章详细介绍过的白鲸优化算法，改进白鲸优化算法的流程图如图 4.2，大体流程为：首先初始化白鲸种群，计算全体白鲸的目标函数值，选取最小值作为当前最优位置。再计算平衡因子B_f，B_f计算公式见 2.27，当$B_f > 0.5$，算法进入探索阶段，并按公式 2.28 计算白鲸下一步位置；当$B_f \leq 0.5$，算法进入开发阶段，并按公式 2.29 计算白鲸下一步位置。

当白鲸下一步位置的目标函数值小于当前白鲸位置的目标函数值，则接受该新位置并将白鲸转移至新位置；当白鲸下一步位置的目标函数值大于当前白鲸位置的目标函数值，现有白鲸优化算法会直接放弃该位置从而使算法无法跳出局部最优；而改进的白鲸优化算法引入 Metropolis 准则，按该准则计算接受较差位置的概率p，并生成$(0,1)$间的随机数，比较 p 与该随机数的大小，若随

机数大于概率 p 则放弃该新位置，若随机数小于概率 p，则接受该新位置并将白鲸转移至新位置；接着计算全体白鲸的目标函数值并选取最小值作为当前最优位置，完成当前最优位置的更新。

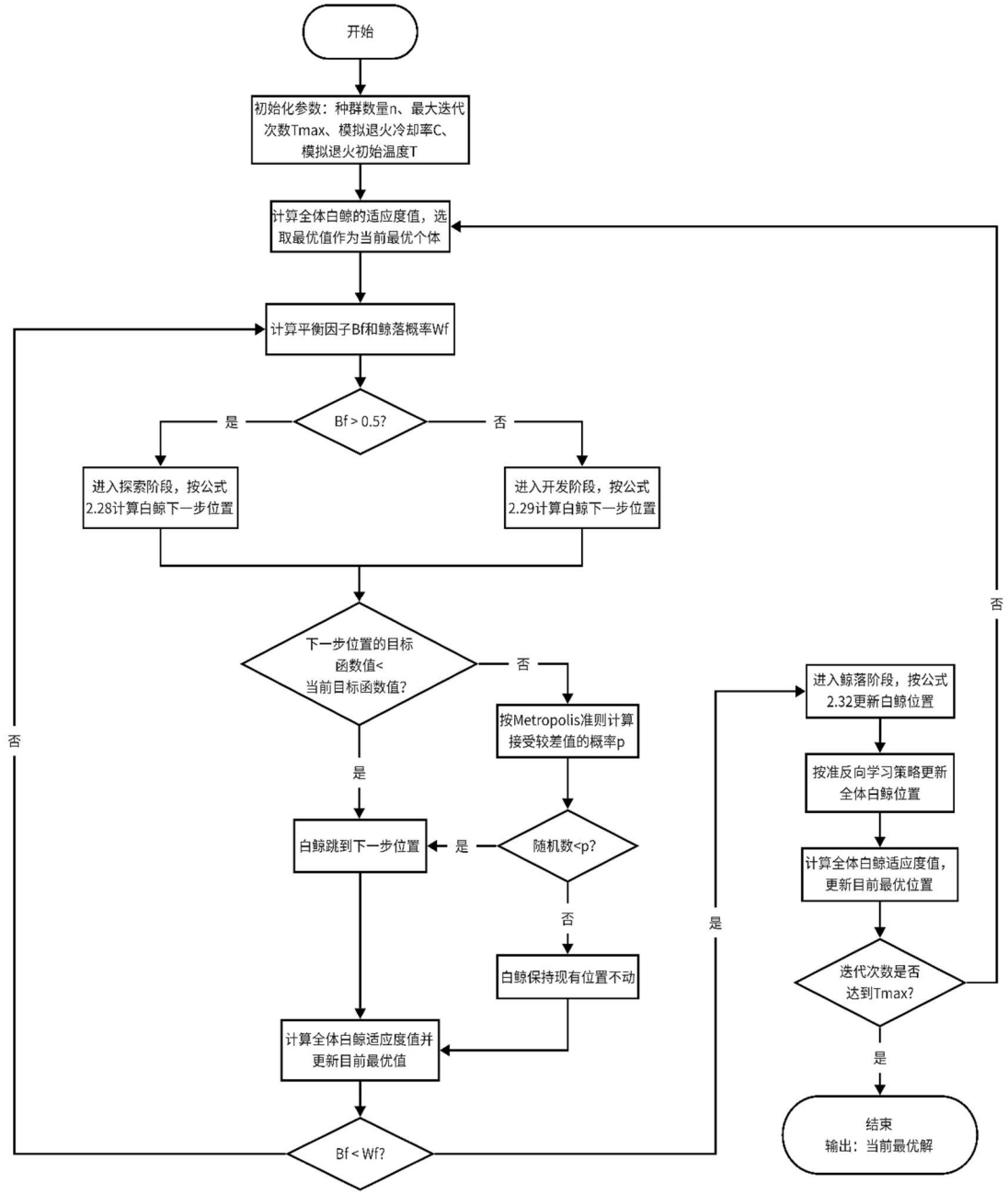

图 4.2　改进白鲸优化算法流程图

计算鲸落概率W_f，计算公式为 2.34，比较B_f和W_f的大小，当$B_f < W_f$，算法进入鲸落阶段，现有白鲸优化算法在鲸落阶段仅按公式 2.33 更新全体白鲸位置，该操作导致算法单向搜索，搜索空间有限，改进后的白鲸优化算法在鲸落阶段引入准反向学习策略更新所有白鲸位置，避免了单向搜索，扩大解范围。

直至算法迭代次数到达预设上限，结束算法并输出最优解。

4.4 改进白鲸优化算法实现

本节介绍改进白鲸优化算法的实现过程，包括改进白鲸优化算法框架概述、算法探索阶段的实现、算法开发阶段的实现、Metropolis 准则的应用、算法鲸落阶段的实现和准反向学习操作。

4.4.1 改进白鲸优化算法框架概述

改进白鲸优化算法的实现分五个步骤：探索、开发、Metropolis 准则接受次优解、鲸落、准反向学习，伪代码见算法 5。

表 4.5 改进白鲸优化算法伪代码

算法 5 改进白鲸优化算法

输入：任务与虚拟机数目、白鲸优化算法和模拟退火算法相关参数

输出：任务与虚拟机分配的最优方案

初始化参数：种群m，最大迭代次数T_{max}，模拟退火冷却率C和初始温度

$Temperature$

计算全体白鲸目标函数值，选取最小值作为当前最优个体

while $T < T_{max}$ **then**

  for $i = 1,2,\cdots,m$ **do**

    调用算法 6，完成白鲸的探索和开发阶段，决定白鲸下一步位置

    if $B_f < W_f$ **then**

      进入鲸落阶段，按公式 2.32 更新下一步位置

      调用算法 8，利用准反向学习策略更新全体白鲸位置

      更新最优白鲸位置

    end if

  end for

  $T = T + 1$

end while

return 任务与虚拟机分配的最优解

参数诸如白鲸种群数m、算法预设最大迭代次数T_{max}、模拟退火冷却率C和模拟退火的初始温度$Temperature$由算法输入的初始值决定。当算法迭代次数

未达到T_{max}时，进入循环，如步骤 4-14；在循环中，依次计算每只白鲸的平衡因子B_f和鲸落概率W_f，通过比较B_f与 0.5 的大小关系决定算法是进入探索阶段还是开发阶段，并按公式完成下一步位置的目标函数值的计算，算法探索阶段和开发阶段的具体操作，在 4.4.2 的算法 6 中介绍，在算法 6 中，当新位置的目标函数值大于当前位置的目标函数值，调用算法 7 实现 Metropolis 准则接受次优解，算法 7 在 4.4.3 介绍；当$B_f < W_f$算法切入鲸落，并调用算法 8 实现鲸落后准反向学习操作，完成白鲸种群刷新，算法 8 在 4.4.4 介绍；重复以上步骤直至算法迭代次数达到T_{max}，跳出循环并输出当前任务与虚拟机的近似最优分配。

4.4.2 改进白鲸优化算法的探索和开发阶段

调控算法在探索和开发阶段之间切换的是平衡因子B_f，计算公式见 2.27。算法寻优早期，希望更多地进行解空间的探索，而随着迭代次数增加，算法需要更多地进行解空间开发，因而B_f的值随着迭代次数增加而不断下降。当B_f大于 0.5，改进白鲸优化算法将进入探索阶段，当B_f小于 0.5，算法则转入开发阶段，伪代码如算法 6。

表 4.6 白鲸探索和开发阶段实现算法

算法 6 白鲸探索和开发阶段实现算法

输入： 任务与虚拟机、白鲸种群

输出： 白鲸下一步位置、更新全体白鲸最优值

1： 记录目标函数值最优的白鲸个体，其目标函数值为Obj_{best}

2： 按公式 2.27 计算平衡因子$B_f = B_0(1 - T/2T_{max})$ // B_0为(0,1)间随机数

3： 按公式 2.34 计算鲸落概率$W_f = 0.1 - \frac{0.05T}{T_{max}}$

4： **if** $B_f > 0.5$ **then**

5： $\begin{cases} X_{i,j}^{T+1} = X_{i,p_j}^T + \left(X_{r,p_1}^T - X_{i,j}^T\right)(1 + r_1)\sin(2\pi r_2),\ j = even \\ X_{i,j}^{T+1} = X_{i,p_j}^T + \left(X_{r,p_1}^T - X_{i,j}^T\right)(1 + r_1)\cos(2\pi r_2),\ j = odd \end{cases}$ //探索期位置更新公

6： 式 2.28

7： 计算Obj_{T+1} //计算下一步位置的目标函数值

8： **else**

9： $X_i^{T+1} = r_3 \cdot X_{best}^T - r_4 \cdot X_i^T + C_1 \cdot L_f \cdot \left(X_r^T - X_i^T\right)$ //开发期位置更新公式 2.29

10： 计算Obj_{T+1} //计算下一步位置的目标函数值

算法 6 白鲸探索和开发阶段实现算法（续）

11：**end if**

12：**if** $Obj_{T+1} < Obj_T$ **then**

13：　　白鲸 i 游向新位置$X_{i,j}^{T+1}$

14：**else**

15：　　调用算法 7，以 Metropolis 准则接受次优解，决定白鲸下一步位置

16：**end if**

17：计算全体白鲸目标函数值并更新Obj_{best}

18：**return** 全体白鲸下一步位置，更新当前最优值

算法在探索期的位置更新反映了白鲸个体在游泳、跳水时的同步或镜像行为，实现了在解空间内大范围搜索；算法在开发期的位置更新模仿了自然界中白鲸捕食时会根据附近同伴的位置进行移动这一特性，同时考虑最佳位置X_{best}和其他潜在位置X_r，并引入莱维飞行策略加强收敛性。

算法在经过探索期或开发期后均会得到白鲸下一步位置对应的目标函数值，作为算法 7 的输入。

4.4.3 Metropolis 准则接受次优解

为协助改进白鲸优化算法跳出局部最优，引入模拟退火算法，当下一步位置的目标函数值劣于当前位置的目标函数值，不直接放弃下一步位置，而是利用 Metropolis 准则以特定概率接受，实现算法如算法 7。

算法 7 的步骤 2，由 Metropolis 准则接受次优解的概率计算公式 4.17 可见，当$Obj_{T+1} > Obj_T$，概率$p \in (0,1)$，且随迭代次数 T 逐渐增大，概率 p 随之减小，符合算法随着迭代次数增加趋于收敛的特性，即当迭代次数越多时，次优解被接受的概率越低。

表 4.7 Metropolis 准则接受次优解算法伪代码

算法 7 Metropolis 准则接受次优解算法

输入：白鲸下一步位置对应的目标函数值Obj_{T+1}

输出：白鲸下一步位置

1：　按公式 4.17 计算 Metropolis 准则接受次优解的概率$p = \exp\left[\frac{Obj_T - Obj_{T+1}}{(1-C)^T * Temperature}\right]$

2：　生成随机数R

3：　**if** $R < p$ **then**

算法 7 Metropolis 准则接受次优解算法（续）

4： 白鲸 i 接受次优解，游向新位置$X_{i,j}^{T+1}$

5： **else**

6： 白鲸i 放弃次优解，保持在位置$X_{i,j}^{T}$不动

7： **end if**

8： **return** 白鲸下一步位置

4.4.4 鲸落后准反向学习

当平衡因子B_f小于鲸落概率W_f时，算法进入鲸落阶段。不同于白鲸优化算法，改进后的算法在鲸落阶段后加入准反向学习策略，避免解空间的单项搜索，实现算法的伪代码如算法 8。步骤 6-10 实现了只有当准反向学习策略获得的新位置的目标函数值小于原位置的目标函数值，新位置才会被接受。

算法 8 鲸落后准反向学习算法

输入：鲸落阶段更新后的白鲸位置

输出：全体白鲸经准反向学习处理后的位置、更新最优位置

1： 初始化参数：x_i^j的下界为a_i^j，上界为b_i^j

2： 计算中心位置：$\frac{a_i^j+b_i^j}{2}$

3： 计算镜像位置：$a_i^j+b_i^j-X_{i,j}^T$

4： 利用准反向学习策略计算白鲸 i 的新位置：

5： $X_{i,j}^{T+1}=\frac{a_i^j+b_i^j}{2}+\left((a_i^j+b_i^j-X_{i,j}^T)-\frac{a_i^j+b_i^j}{2}\right)\times rand(0,1)$

6： **if** $Obj_{T+1}<Obj_T$ **then**

7： 接受新位置$X_{i,j}^{T+1}$

8： **else**

9： 保持原位置$X_{i,j}^{T}$

10： **end if**

11： **return** 全体白鲸经准反向学习处理后的新位置和当前最优位置

4.4.5 算法复杂度分析

IBWO 算法的核心函数接受搜索空间维度（任务数）、位置上下界（虚拟机数量）、任务列表和虚拟机列表作为输入，并返回一个最优分配方案。该函数首先创建白鲸种群，初始化每只白鲸个体的位置，并计算其目标函数值。接下来，

函数选择种群中目标函数值最好的个体作为当前最优解。

在每次迭代中，函数使用 IBWO 算法更新种群中除最优个体外的所有个体的位置。更新位置的方式根据随机数B_f的值来决定，如果B_f小于 0.5，则使用莱维飞行策略更新位置；否则，根据随机数r_1和r_2的值进行位置更新。当更新后位置的目标函数值小于当前位置的目标函数值则接受该新位置，否则运用 Metropolis 准则决定是否接受该新位置。然后，根据随机数B_f和W_f的值来决定是否进行鲸落操作，鲸落操作根据随机数r_5、r_6和r_7的值更新个体的位置。

下一步，函数对种群中每个个体进行准反向学习操作，并生成新的位置，计算新的目标函数值。在每次迭代结束后，函数更新最优解，并继续下一次迭代，直到达到最大迭代次数。最后，函数返回最优解的位置数组。

综上，IBWO 算法的时间复杂度为$O(T_{max} * m * tasknum)$，其中T_{max}表示算法最大迭代次数、m表示白鲸种群数、$tasknum$表示待分配任务数。其空间复杂度为$O(m * tasknum)$。

4.5 本章小结

本章先针对蚁群算法调度任务时系统负载不均、初始信息素分布不合理导致任务完工时间长的不足，创建负载监测启发因子和动态挥发、双重奖惩信息素更新策略；并引入免疫算法改善蚁群算法初始信息素分布，提出了一种新的免疫-蚁群优化算法 IMACO。

结合文献调研和第三章实验结果，白鲸优化算法因其能够自适应在探索、开发和鲸落阶段切换，具有高效解决大规模寻优问题的潜力，并且在云计算任务调度中白鲸优化算法的完工时间、负载均衡、完工费用的综合性能表现稳定优秀，但是不可忽视它不具备搜索反向解的能力，容易陷入局部最优，为使其具有更强的跳出局部最优的能力和更大的解空间，引入模拟退火算法利用 Metropolis 准则以一定概率接受次优解，协助其跳出局部最优，并在鲸落步骤后加入准反向学习策略扩大解空间，提出一种改进白鲸优化算法 IBWO。

第5章 基于 Cloudsim 的改进群体智能算法性能比较

5.1 实验环境和参数设置

关于实验平台 Cloudsim 在第三章已做详细介绍，本机硬件配置见表 3.1、主机和虚拟机参数设置见表 3.2、任务参数设置见表 3.3。蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法的相关参数分别见表 3.4、3.5、3.6、3.7。

接下来完成第四章提出的免疫-蚁群优化算法 IMACO 和改进白鲸优化算法 IBWO 的参数设置，分别见表 5.1、表 5.2。

表 5.1 免疫-蚁群优化算法 IMACO 参数设置

参数	初始值	含义
α	1	表明信息素浓度对任务分配至虚拟机概率的影响权重
β	3	表明处理时长对任务分配至虚拟机概率的影响权重
ρ	0.5	信息素挥发因子
C	100	动态挥发因子常数
ω	0.5	信息素奖惩系数
Q	100	信息素浓度常量
ω_1	0.3	虚拟机的处理速度在信息素初始时的权重系数
ω_2	0.3	虚拟机的带宽在信息素初始时的权重系数
ω_3	0.4	虚拟机的处理成本在信息素初始时的权重系数
$numAnt$	30	蚁群蚂蚁数目
$numAntibody$	10	免疫算法抗体数量
S	0.5	免疫算法相似度阈值
p	0.5	亲和度的权重参数
q	0.5	抗体浓度的权重参数
MAX_ITER_IA	150	免疫算法预设最大迭代次数
MAX_ITER_ACO	200	蚁群算法预设最大迭代次数

表 5.2 改进白鲸优化算法 IBWO 参数设置

参数	值	含义
MAX_ITER	200	预设最大迭代次数
C	0.1	模拟退火冷却率
$Temperature$	10	模拟退火初始温度
$Population\_Size$	100	白鲸种群数目

5.2 性能比较指标

本文构造了综合评价任务完工时间、系统负载平衡和完工费用的优化目标函数，即适应度函数，本章采用这一模型综合评价算法性能。任务完工时间、系统负载平衡和完工费用的公式见 3.7、3.9、3.10，优化目标函数见 3.12。

5.3 实验结果与分析

（1）IMACO 和 ACO 性能对比：

① 完工时间：如图 5.1，在虚拟机数为 6，任务数 200、400、800、1600 下，IMACO 的任务完工时间均比 ACO 短，分别缩短了 42.57%、41.42%、38.17%、29.44%；

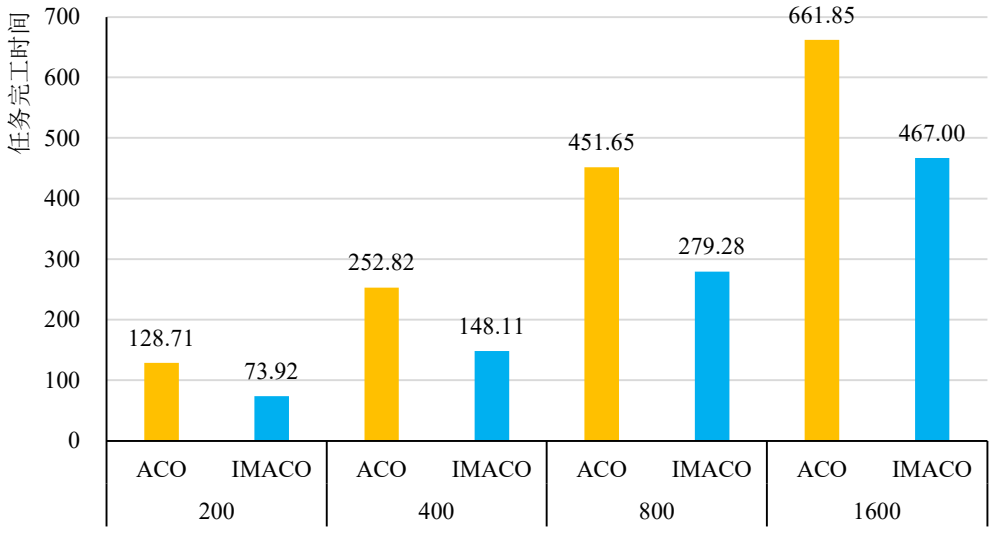

图 5.1　任务数 200、400、800、1600 时 ACO 与 IMACO 完工时间对比

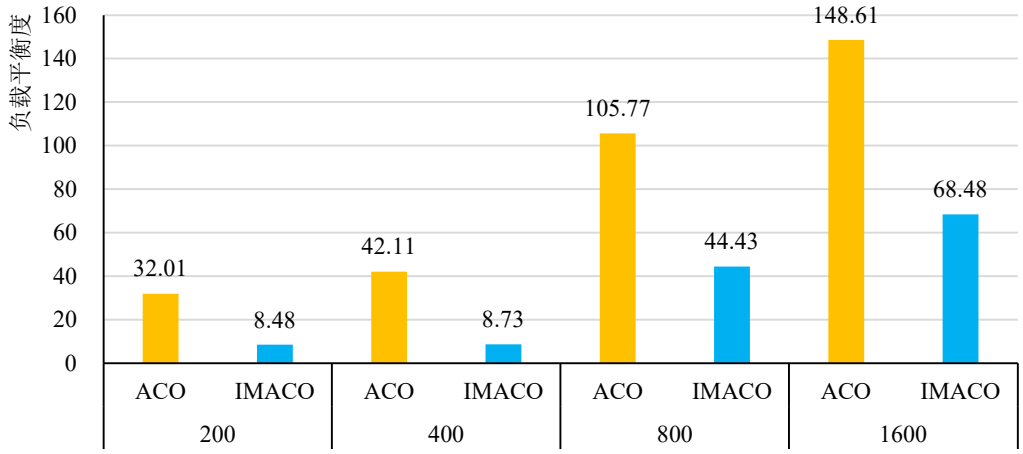

图 5.2　任务数 200、400、800、1600 时 ACO 与 IMACO 负载平衡度对比

② 负载平衡：如图 5.2，在虚拟机数为 6，任务数 200、400、800、1600 下，IMACO 的系统负载平衡度均比 ACO 低，分别降低了 73.51%、79.27%、57.99%、53.92%；

③ 完工费用：如图 5.3，在任务数 200、400、800、1600 下，IMACO 的任务完工费用均比 ACO 大，分别增加了 25.82%、6.56%、11.09%、6.01%；

④ 综合因子：如图 5.4，在任务数 200、400、800、1600 下，IMACO 的综合因子均比 ACO 优秀，分别降低了 42.37%、49.42%、33.92%、29.89%。

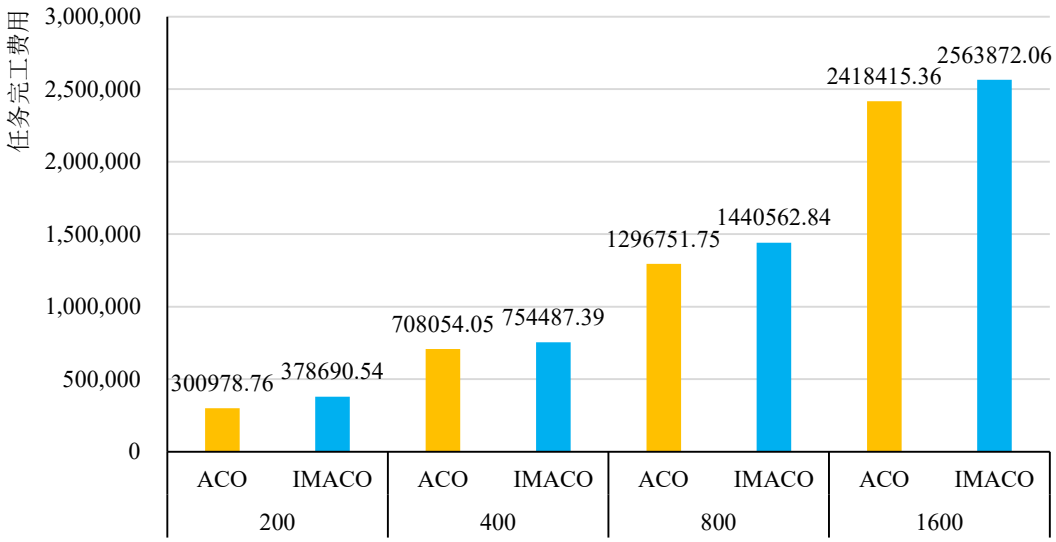

图 5.3　任务数 200、400、800、1600 时 ACO 与 IMACO 完工费用对比

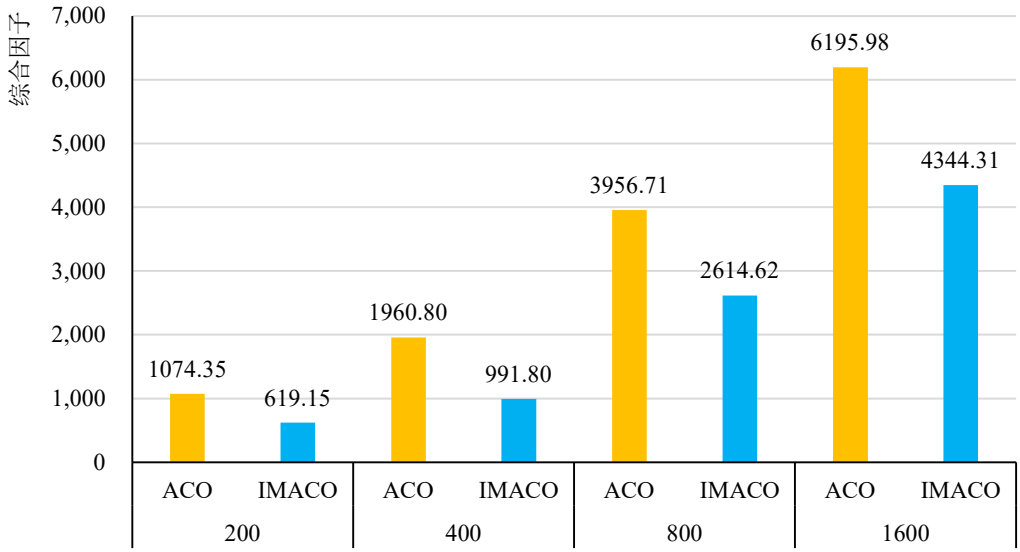

图 5.4　任务数 200、400、800、1600 时 ACO 与 IMACO 综合因子对比

通过表 5.3 可知，IMACO 在大幅缩短任务完工时间、优化系统负载平衡的同时，牺牲了部分完工成本，但综合因子相较 ACO 仍得到显著改进，总体性能表现更优。

表 5.3　不同云任务数目下 ACO 与 IMACO 各项性能对比

算法	性能指标	云任务数			
		200	**400**	**800**	**1600**
蚁群算法 ACO	完工时间	128.71	252.82	451.65	661.85
	负载平衡度	32.0	42.11	105.77	148.61
	完工费用	300978.76	708054.05	1296751.75	2418415.36
	综合因子	1074.35	1960.80	3956.71	6195.98
免疫-蚁群优化算法 IMACO	完工时间	73.92	148.11	279.28	467.00
	负载平衡度	8.48	8.73	44.43	68.48
	完工费用	378690.54	754487.39	1440562.84	2563872.06
	综合因子	619.15	991.80	2614.62	4344.31

（2）IBWO 和 BWO 性能对比：

① 完工时间：如图 5.5，在虚拟机数为 6，任务数 200、400、800、1600 下，IBWO 的任务完工时间均比 BWO 短，分别缩短了 35.13%、31.67%、1.68%、24.61%；

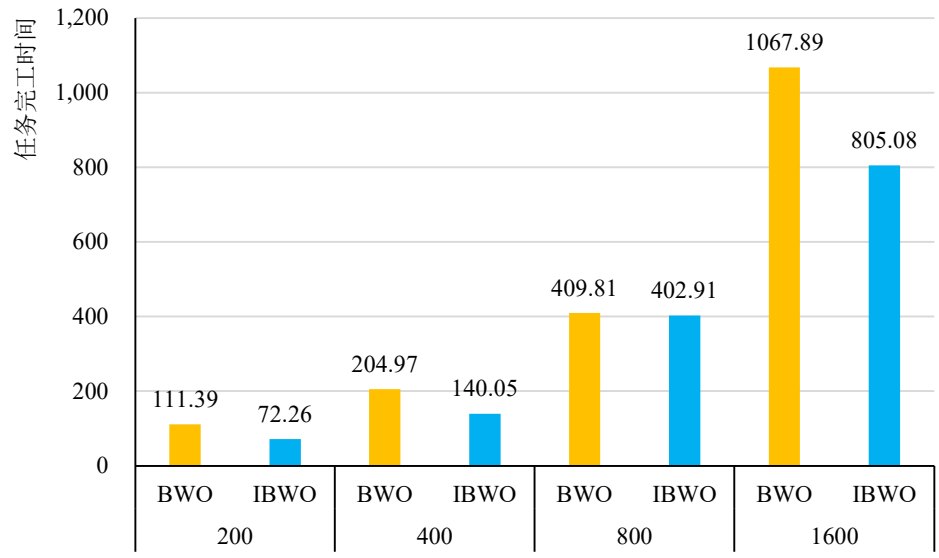

图 5.5　任务数 200、400、800、1600 时 BWO 与 IBWO 完工时间对比

② 负载平衡：如图 5.6，在任务数 200、400、800、1600 下，IBWO 的负载平衡度均比 BWO 低，分别降低了 3.66%、6.47%、28.21%、19.55%；

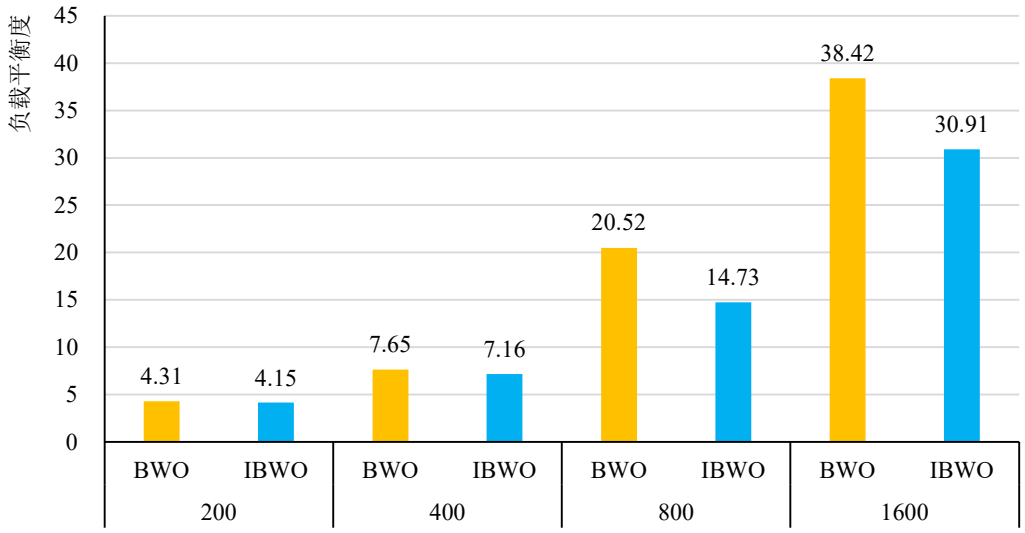

图 5.6　任务数 200、400、800、1600 时 BWO 与 IBWO 负载平衡度对比

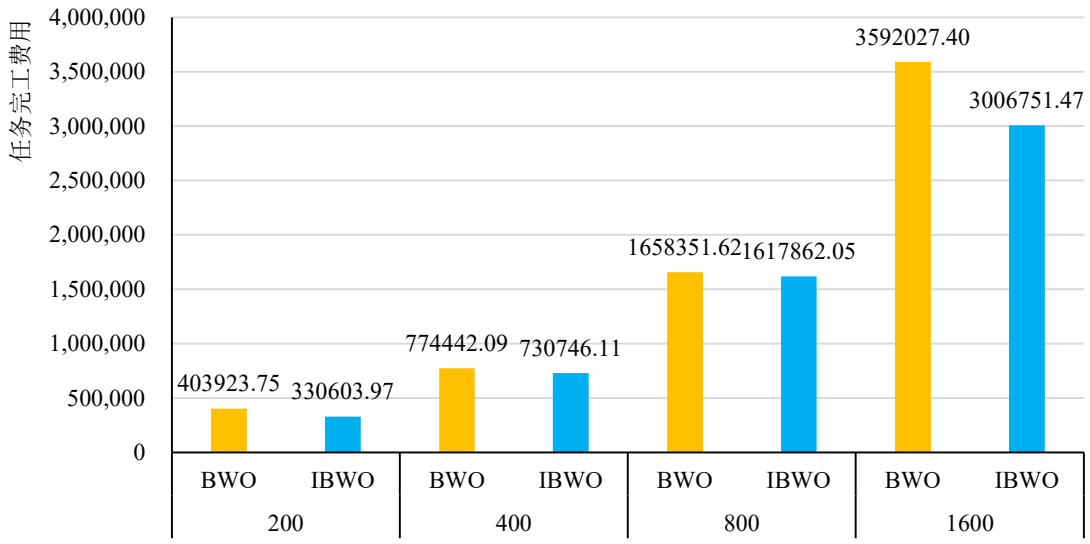

图 5.7　任务数 200、400、800、1600 时 BWO 与 IBWO 完工费用对比

③ 完工费用：如图 5.7，在任务数 200、400、800、1600 下，IBWO 的完工费用均比 BWO 少，分别减少了 18.15%、5.64%、2.44%、16.29%；

④ 综合因子：如图 5.8，在任务数 200、400、800、1600 下，IBWO 的综合因子均比 BWO 表现好，分别改善了 20.03%、15.52%、11.70%、10.19%。

由表 5.4 可知，在不同任务数目下，改进白鲸优化算法 IBWO 在完工时间、

负载平衡、完工费用三者上的综合表现均优于 BWO，算法改进效果显著。

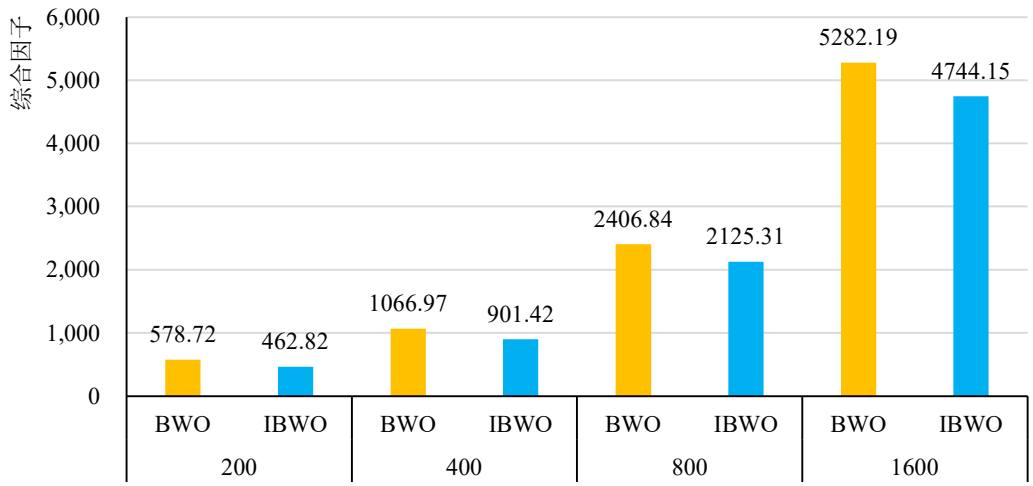

图 5.8　任务数 200、400、800、1600 时 BWO 与 IBWO 综合因子对比

表 5.4　不同云任务数目下 BWO 和 IBWO 各项性能对比

算法	性能指标	云任务数			
		200	**400**	**800**	**1600**
白鲸优化算法 BWO	完工时间	111.39	204.97	409.81	1067.89
	负载平衡度	4.31	7.65	20.52	38.42
	完工费用	403923.75	774442.09	1658351.62	3592027.40
	综合因子	578.72	1066.97	2406.84	5282.19
改进白鲸优化算法 IBWO	完工时间	72.26	140.05	402.91	805.08
	负载平衡度	4.15	7.16	14.73	30.91
	完工费用	330603.97	730746.11	1617862.05	3006751.47
	综合因子	462.82	901.42	2125.31	4744.15

（3）IMACO、IBWO、ACO、GWO、WOA、BWO 综合对比：

当虚拟机数目设置为 6 时，任务数分别为 200、400、800、1600，以上六种算法的各性能数据统计如下，通过对比可得出以下结论。

① 随着任务规模扩大，各算法的完工时间均延长，IMACO 在完工时间上更有优势；

② 随着任务规模扩大，ACO 的负载平衡度值急剧增长，可见蚁群算法对负载平衡这一性能非常敏感，通过引入负载监测启发因子后IMACO的负载平衡得到显著改善。和其他算法对比，IBWO的负载平衡度表现最优秀；

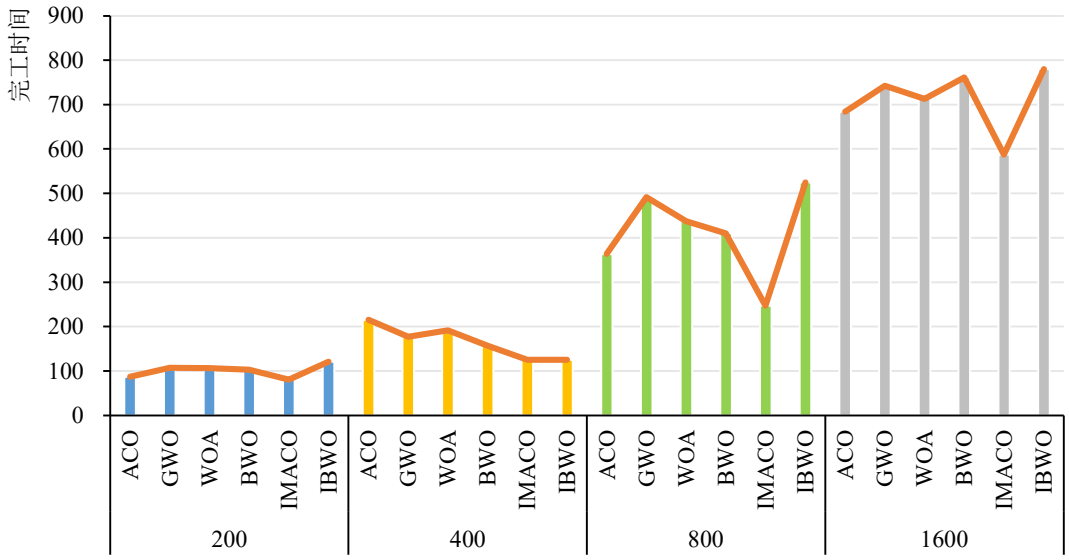

图 5.9　虚拟机为 6 任务数为 200、400、800、1600 时算法完工时间对比

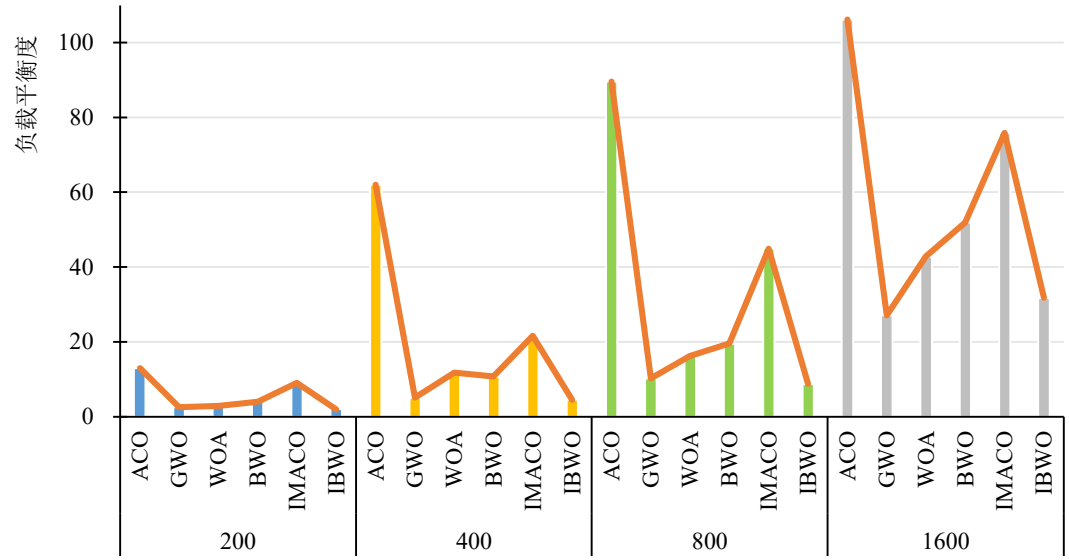

图 5.10　虚拟机为 6 任务数为 200、400、800、1600 时算法负载平衡度对比

③ 随着任务规模扩大，算法的完工费用均增长，当任务数为200、400、800时，各算法的完工费用差异不显著，当任务数扩大到 1600，算法之间的完工费用差异较大，其中 IBWO 完工费用最低，比 ACO、GWO、WOA、BWO、

IMACO 分别少 6.01%、22.26%、9.23%、10.61%、7.78%；

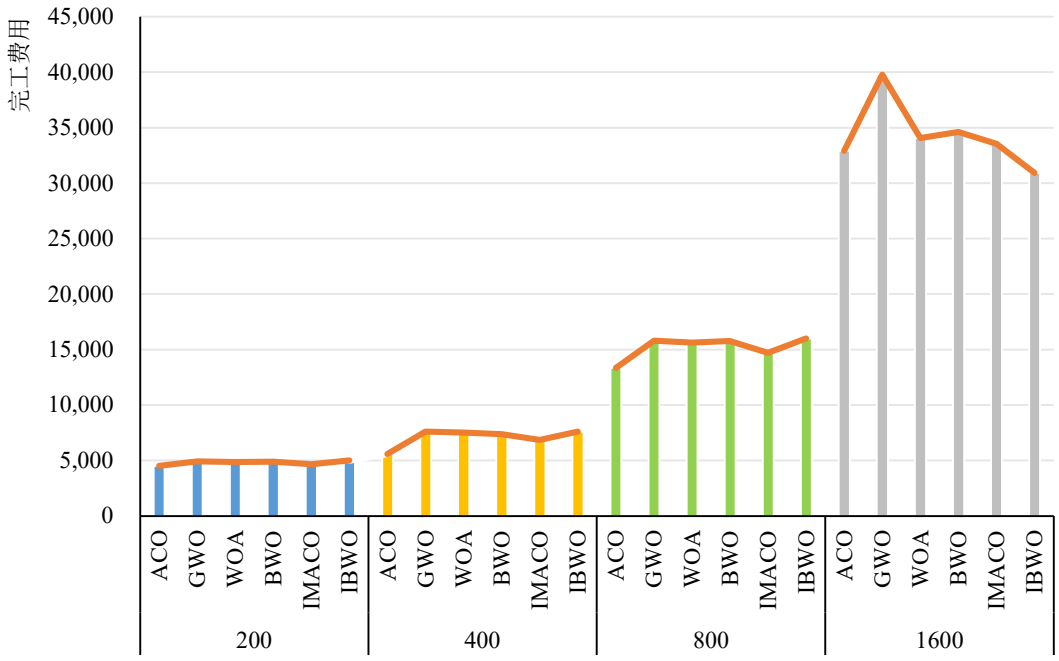

图 5.11　虚拟机为 6 任务数为 200、400、800、1600 时算法完工费用对比

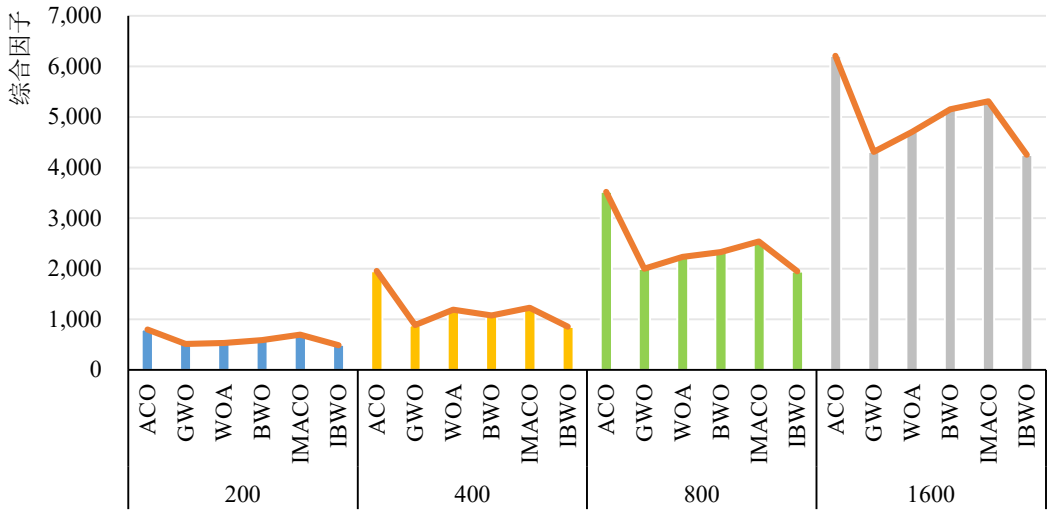

图 5.12　虚拟机为 6 任务数为 200、400、800、1600 时算法综合因子对比

④ 随着任务规模扩大，ACO 的综合因子数值始终最大，即 ACO 的综合性能表现最差。IBWO 的综合因子在四种任务数目下均为最优，它最显著的优势是进行任务调度时系统负载平衡性能稳定优秀，该算法满足了云计算大规模任务数目调度时系统负载的 QoS 需求。

以上六种算法在不同任务数目下的各项性能指标统计如表 5.5，可以看到基于与蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法和本文提出的免疫-蚁

群优化算法的比较，本文提出的改进白鲸优化算法 IBWO 在完工时间、负载平衡、完工费用上的综合表现最优，提高了云计算任务调度时的算法综合性能。

表 5.5　IBWO 和其他五种算法在不同任务数目下的性能指标对比

算法	性能指标	云任务数			
		200	**400**	**800**	**1600**
ACO	完工时间	87.39	215.25	363.75	683.94
	负载平衡度	12.93	62.03	89.60	106.23
	完工费用	451657.64	558365.52	1335378.61	3290755.85
	综合因子	799.17	1953.51	3517.50	6206.63
GWO	完工时间	107.22	177.06	491.37	742.08
	负载平衡度	2.49	5.12	10.26	27.12
	完工费用	491087.10	759840.77	1578358.24	3978964.08
	综合因子	508.31	883.20	1996.28	4310.26
WOA	完工时间	106.28	191.50	437.58	712.78
	负载平衡度	2.87	11.73	16.25	42.81
	完工费用	486709.08	751654.21	1562465.41	3407706.44
	综合因子	529.35	1190.72	2231.61	4702.50
BWO	完工时间	102.76	157.33	410.20	761.03
	负载平衡度	3.99	10.72	19.54	51.93
	完工费用	487815.51	737590	1576726.10	3460119.90
	综合因子	584.62	1075.45	2329.38	5151.89
IMACO	完工时间	81.01	125.62	247.46	587.59
	负载平衡度	9.01	21.60	44.89	75.91
	完工费用	465098.91	685168.21	1470961.57	3354068.39
	综合因子	697.64	1229.60	2537.59	5308.54
IBWO	完工时间	120.70	125.62	524.82	779.90
	负载平衡度	1.97	4.53	8.77	31.77
	完工费用	499310.96	760914.35	1600568.08	3093117.74
	综合因子	491.66	851.60	1945.68	4249.00

5.4　本章小结

基于 Cloudsim 在梯度设置的任务数下调用第四章提出的免疫-蚁群优化算法 IMACO 和蚁群算法 ACO 分别完成云计算任务调度，得出结论：在任务完工时

间、负载均衡、完工费用三方面的综合性能IMACO优于ACO，当任务数为400时，评价指标综合因子改善了49.42%。

接着基于 Cloudsim 在梯度设置的任务数下调用第四章提出的改进白鲸优化算法 IBWO 和白鲸优化算法 BWO 分别完成云计算任务调度，得出结论：在任务完工时间、负载均衡、完工费用三方面的综合性能，IBWO 均优于 BWO，算法改善效果显著，任务数 200、400、800、1600 下综合因子分别提升了 20.03%、15.52%、11.70%、10.19%。

最后，综合对比本文研究的群体智能算法：蚁群算法ACO、灰狼算法GWO、鲸鱼优化算法 WOA、白鲸优化算法 BWO 和本文提出的免疫-蚁群优化算法IMACO、改进白鲸优化算法 IBWO 在处理不同任务数目下的完工时间、负载平衡、完工费用的综合性能表现，得出结论：本文提出的改进白鲸优化算法IBWO的综合性能表现最优，该算法在大规模云任务调度中有很大寻优潜力。

第6章 总结与展望

6.1 总结

当下云用户规模与云服务体量持续增长，在按需付费服务模式中，用户希望获得廉价快速且流畅的云服务。高效、合理的任务调度策略是改善用户服务质量、平衡系统负载、降低成本的有效举措，因此本文先重点研究群体智能算法的原理和不足。

在针对性改进任务调度算法前，先解决任务与虚拟机是否需要做预排序的问题，目的在于提高云平台的任务调度性能表现。基于 Cloudsim 设计实验验证并得出任务与虚拟机不做预排序对云计算任务调度算法的性能更有利的结论之后，提出一种免疫-蚁群优化算法 IMACO 和改进白鲸优化算法 IBWO，具体研究工作如下：

（1）调研云环境下现有任务调度算法，重点研究群体智能算法，结合云计算任务调度特点与实际使用场景，提出综合考虑任务完工时间、系统负载平衡和任务完工费用的评价函数。基于云任务调度实现群体智能算法：蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法，并设计任务与虚拟机的预排序算法，完成二者排序后基于 Cloudsim 调用蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法，对比不同虚拟机数目和任务数目组合下各算法的完工时间、负载平衡、完工费用综合性能，得出任务与虚拟机不排序更利于云计算任务调度算法性能的结论，分析可知不排序增大了搜索随机性和搜索范围，对算法性能产生积极影响；

（2）在（1）的结论基础上，考虑到蚁群算法实现简单、应用广泛的优点和云任务调度中负载不均、初期信息素匮乏收敛慢导致完工时间长的缺点，创建负载监测因子和动态挥发、双重奖惩信息素更新方法，使得信息素更新考虑负载均衡，引入免疫算法，利用其全局搜索能力强、解的多样性高等优点获得蚁群的初始信息素分布，提出了一种融合免疫算法和蚁群算法的免疫-蚁群优化算法 IMACO；

（3）在（1）的结论基础上，考虑到白鲸优化算法在云任务调度中的完工时间、负载均衡、完工费用性能表现优异稳定，但存在易陷入局部最优的缺陷，为进一步扩大白鲸优化算法的解范围和避免其陷入局部最优，引入模拟退火算法，利用 Metropolis 准则以一定概率接受次优解协助算法跳出局部最优，引入准反向学习策略，在鲸落步骤后更新全体白鲸位置时增加反向解，扩大搜索空间，提出了一种新的改进白鲸优化算法 IBWO。在 Cloudsim 平台上完成仿真实验，从任务完工时间、负载平衡度、完工费用三方面综合对比本文提出的新算法与蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法的表现，结果表明，免疫-蚁群优化算法 IMACO 的综合表现显著优于蚁群算法 ACO，改进白鲸优化算法 IBWO 综合表现也优于白鲸优化算法 BWO。

在改进白鲸优化算法 IBWO、免疫-蚁群优化算法 IMACO、蚁群算法、灰狼算法、鲸鱼优化算法、白鲸优化算法这六种算法中，改进白鲸优化算法 IBWO 的综合性能最优。

6.2 展望

本文基于用户需求，提出综合考虑完工时间、负载平衡和完工费用的评价函数，并在研究群体智能算法和任务与虚拟机排序对云计算任务调度算法性能影响的基础上，对蚁群算法和白鲸优化算法做进一步改进。虽然在完工时间、系统负载平衡度和费用方面，算法性能有所改善，但仍存在诸多不足，在今后研究工作中，还需从以下两方面进行深入探究：

（1）本文仅采用完工时间、负载平衡和完工费用来评价算法，未来可以加入更多优化目标，如降低系统能耗，设计高效的云计算任务调度算法以实现更多目标的综合优化，提升云平台性能。本文在文献调研和第三章实验结果的基础上，考虑各群体智能算法的优点和不足，仅选取了蚁群算法和白鲸优化算法做改进，但诸如灰狼算法、鲸鱼优化算法也有较大的改进空间和潜力，后续针对其他群体智能算法可以深入研究，做更多性能优化，以提高云平台综合表现；

（2）随着数据量扩张和业务需求日益复杂，将多个云集成提供服务的多云系统成为云计算行业的新发展趋势，在异构多云环境下的任务调度更具挑战性，

除了要考虑完工时间、负载均衡等性能，还涉及多云之间调度的能耗问题，在后续云计算领域多云任务调度算法的优化值得深入研究。本文仿真实验中，将复杂环境做了简单化处理，未来工作可以考虑更多的现实因素，使仿真更加真实可靠，更接近现实环境。

参考文献

[1] 刘鹏. 云计算（第二版）[M]. 北京：电子工业出版社，2011.

[2] 储雅, 马廷淮, 赵立成. 云计算任务调度:策略与算法[J]. 计算机科学, 2013, 40(11): 8-13.

[3] David Wolpert, William Macready. No Free Lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997, 1 (1): 67–82.

[4] Mahmood, Ibrahim & M.Sadeeq, et al. Task Scheduling Algorithms in Cloud Computing: A Review[J]. Turkish Journal of Computer and Mathematics Education(TURCOMAT), 2021, 12:1041-1053.

[5] Huanlai Xing, Jing Zhu, Rong Qu, et al. An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing, Swarm and Evolutionary Computation, Volume 68, 2022, 101012.

[6] 武小年, 郑鑫, 孟川, et al. 云计算中面向多目标的两阶段任务调度算法[J]. 计算机工程与设计, 2017, 38(06): 1551-1555.

[7] 郭平,李涛,李琪. 一种云计算环境下的负载调度算法[J]. 系统工程理论与实践, 2014, 34(S1): 269-275.

[8] 潘钰. 云计算平台中的能耗管理方法[D]. 南京邮电大学, 2013.

[9] 王政. 基于 Min-Min 和 Max-Min 算法改进的云任务调度策略研究[D]. 哈尔滨师范大 学, 2021.

[10] Xuewen Xia, Huixian Qiu, Xing Xu, et al. Multi-objective workflow scheduling based on genetic algorithm in cloud environment, Information Sciences, Volume 606, 2022: 38-59.

[11] 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(06): 80-87.

[12] 何庆,吴意乐,徐同伟. 改进遗传模拟退火算法在 TSP 优化中的应用[J]. 控制与决策, 2018, 33(02): 219-225.

[13] Gao, R., Wu, et al. Dynamic load balancing strategy for cloud computing with ant colony

optimization[J]. Future Internet, 2015, 7 (4): 465-483.

[14] Ali Belgacem, Kadda Beghdad-Bey, Hassina Nacer, et al. Efficient dynamic resource allocation method for cloud computing environment[J]. Cluster Computing, 2020, 10(2): 123-135.

[15] 张金泉, 徐寿伟, 李信诚, et al. 基于正交自适应鲸鱼优化的云计算任务调度[J]. 计算机应用, 2022, 42(05):1516-1523.

[16] 聂清彬,陈飞旭,秦美峰, et al. 基于 QoS 云计算任务调度优化[J]. 重庆大学学报, 2021, 44(09): 109-116.

[17] Nakrani, T., Hiran, D., Sindhi, C., et al. Genetic Algorithm based task scheduling for load balancing in cloud. lecture notes on data engineering and communications technologies, 2021, 52: 283-293.

[18] 郑瑛. 云计算数据中心节能调度算法改进研究[J]. 西南大学学报(自然科学版), 2019,41(12):135-142.

[19] 任金霞,钟小康. 多维 QoS 约束云任务调度研究[J]. 微电子学与计算机, 2018, 35(07): 97-100+105.

[20] 姜力争,裴云曼,赵建涛. 基于蚁群优化算法的云任务分配策略研究[J]. 计算机应用与软件, 2021, 38(06): 284-287+305.

[21] Gabi, D., Ismail, A.S., Zainal, A. et al. Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Comput & Applic, 2018, 30: 1845–1863.

[22] 刘愉, 赵志文, 李小兰, et al. 云计算环境中优化遗传算法的任务调度策略[J].北京师范大学学报(自然科学版), 2012, 48(04): 378-384.

[23] Yong Lu, Na Sun. An effective task scheduling algorithm based on dynamic energy management and efficient resource utilization in green cloud computing environment[J]. Cluster Computing, 2019, 22(1): 513-520.

[24] 马小晋, 许华虎, 卞敏捷, et al. 基于改进模拟退火算法的虚拟机调度优化方法[J]. 通信学报, 2018, 39(S1): 278-287.

[25] 王艳红, 张革文. 基于改进狮群算法的云计算任务调度策略[J]. 计算机应用与软

件,2021,38(11):269-275.

[26] Colorni A, Dorigo M, Maniezzo V, et al. Distributed optimization by ant colonies[A]. Proc of European Conf on Artificial Life. Paris, 1991: 134-142.

[27] 段海滨, 王道波, 朱家强, et al. 蚁群算法理论及应用研究的进展[J]. 控制与决策, 2004(12):1321-1326+1340.

[28] Hui Zhao, Nanzhi Feng, Jianhua Li, et al. VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment[J]. Journal of Parallel and Distributed Computing, 2023(176): 17-27.

[29] 华夏渝,郑骏,胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报(自然科学版), 2010(01):127-134.

[30] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis. Grey Wolf Optimizer[J]. Advances in Engineering Software, 2014(69): 46-61.

[31] 李雅丽, 王淑琴, 陈倩茹, et al. 若干新型群智能优化算法的对比研究[J]. 计算机工程与应用, 2020, 56(22):1-12.

[32] Bilal H., Abed-alguni, Noor Aldeen Alawad. Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments[J]. Applied Soft Computing, 2021(102): 107113.

[33] Yefeng Yang, Bo Yang, Shilong Wang, et al. An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing[J]. Applied Soft Computing, 2020(87): 106003.

[34] Seyedali Mirjalili, Andrew Lewis. The whale optimization algorithm[J]. Advances in Engineering Software, 2016(95): 51-67.

[35] 张永,陈锋. 一种改进的鲸鱼优化算法[J]. 计算机工程, 2018, 44(03): 208-213+219.

[36] Sudheer Mangalampalli, Ganesh Reddy Karri, Utku Kose. Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization[J]. Journal of King Saud University - Computer and Information Sciences ,2023, 35(2): 791-809.

[37] Changting Zhong, Gang Li, Zeng Meng. Beluga whale optimization: A novel nature-inspired

metaheuristic algorithm[J]. Knowledge-Based Systems, 2022(251): 109215.

[38] 胡云. 对云计算技术及应用的研究[J]. 电脑开发与应用, 2011, 24(03):12-14.

[39] 陈全, 邓倩妮. 云计算及其关键技术[J]. 计算机应用, 2009, 29(09): 2562-2567.

[40] 罗军舟, 金嘉晖, 宋爱波, et al. 云计算:体系架构与关键技术[J]. 通信学报, 2011, 32(07): 3-21.

[41] 杜玉霞, 刘方爱, 郭磊. Min-Min 调度算法的研究与改进[J]. 计算机工程与应用, 2010, 46(24): 107-109.

[42] HOLLAND J H. Adaptation in natural and artificial systems[M]. Cambridge:MIT Press, 1975.

[43] 边霞, 米良. 遗传算法理论及其应用研究进展[J]. 计算机应用研究, 2010, 27(07):2425-2429+2434.

[44] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory[C]. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995: 39-43.

[45] 尹亚日. 基于改进粒子群和蚁群的云计算任务调度研究[D]. 南京邮电大学, 2019.

[46] 王霞俊. CloudSim 云计算仿真工具研究及应用[J]. 微型电脑应用, 2013, 29(08): 59-61.

[47] 王磊, 潘进, 焦李成.免疫算法[J].电子学报, 2000(07): 74-78.

[48] 李琳.基于多目标免疫优化的个性化推荐算法[J].电子技术与软件工程, 2020(18):174-175.

[49] H.R. Tizhoosh. 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), Vienna, Austria, 2005: 695-701.

[50]S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama. Quasi-oppositional Differential Evolution[C] 2007 IEEE Congress on Evolutionary Computation, Singapore, 2007: 2229-2236.