doi:10.11887/j.cn.202502003 http://journal.nudt.edu.cn

改进协同演化算法求解超多目标云工作流调度问题

周佳军[1],姬小晖[1],卢 超[1*],高 亮[2]
(1. 中国地质大学(武汉) 计算机学院,湖北 武汉 430078;2. 华中科技大学 机械科学与工程学院,湖北 武汉 430074)

摘 要:目前主流优化方法通常将云工作流调度建模为单目标或者不超过三个目标的多目标优化问题,未能充分考虑实际应用场景需求。为克服传统方法局限性,将云工作流调度问题直接建模为涉及时间、费用、可靠性、资源消耗度、负载均衡等众多指标的超多目标优化问题,并针对该问题提出一种改进协同演化算法,利用双阶段策略和多性能指标协同机制有效地平衡解集的收敛性和多样性,提升算法寻优能力。在七类真实工作流实例上的实验表明,所提方法相比现有算法在大多数情况下可找到更好的调度方案。

关键词:云工作流调度;超多目标优化;协同演化;双阶段策略;性能指标
中图分类号:TP18 **文献标志码**:A **文章编号**:1001-2486(2025)02-035-14

论文拓展

Improved co-evolutionary algorithm for solving many-objective cloud workflow scheduling problem

ZHOU Jiajun[1], *JI Xiaohui*[1], *LU Chao*[1*], *GAO Liang*[2]
(1. School of Computer Science, China University of Geosciences(Wuhan), Wuhan 430078, China;
2. School of Mechanical Science & Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: Most current studies formulate the cloud workflow scheduling as a single-objective or multi-objective optimization problem with at most three objectives, which is unable to fully meet practical scenarios' needs. To address the limitations above, many-objective cloud workflow scheduling model was established, where many indicators such as time, cost, reliability, resource consumption, load balancing, were taken into account. Then, an improved co-evolutionary algorithm was introduced to address this problem, where dual-stage search strategy and multi-indicator cooperation mechanism were employed to effectively balance the convergence and diversity of solution set, so as to enhance the search capability of algorithm. Experiments on seven types of real life workflow instances reveal that our proposal outperforms the existing peers and can find better scheduling schemes in most cases.

Keywords: cloud workflow scheduling; many-objective optimization; co-evolution; dual-stage strategy; performance indicator

随着科学研究和工程实践活动的深入,计算任务的规模和复杂度不断增加,许多实际工程应用涉及大量的计算密集型任务,如天气预报、军事模拟、森林火灾探测等,需要高性能计算系统的支持。传统高性能计算系统配置和管理复杂,维护成本相对高昂,对中小企业和组织并不友好。云计算通过虚拟化技术汇聚海量计算资源,以按需使用的方式为用户提供高性能计算服务,从而提高资源利用率并降低使用成本[1]。工作流模型将大规模计算过程划分为多个相互依赖的计算任务,工作流调度即为这些计算任务分配合适的计算资源,以达到较优的调度效果和目标[2]。如何有效利用云平台提供的计算资源,为工作流任务提供最佳的服务质量,是云计算的核心问题之一。

云平台以虚拟机的形式提供计算资源,但多个服务质量指标之间通常相互冲突,如时间和费用,执行任务耗费时间较短的计算资源通常性能更好,但费用更高,反之,费用相对较低的计算资源执行任务所需时间较长。此外,能耗近来也成为云计算关注的重点[3],以国家超级计算中心天

收稿日期:2024-01-05
基金项目:国家自然科学基金资助项目(51825502,51905198,52175490)
第一作者:周佳军(1989—),男,湖北黄冈人,教授,博士,博士生导师,E-mail:zhoujiajun@cug.edu.cn
***通信作者**:卢超(1986—),男,湖北嘉鱼人,副教授,博士,博士生导师,E-mail:luchao@cug.edu.cn
引用格式:周佳军,姬小晖,卢超,等. 改进协同演化算法求解超多目标云工作流调度问题[J]. 国防科技大学学报,2025,47(2):35-48.
Citation: ZHOU J J, JI X H, LU C, et al. Improved co-evolutionary algorithm for solving many-objective cloud workflow scheduling problem[J]. Journal of National University of Defense Technology, 2025, 47(2): 35-48.

河二号为例,整机平均功率约为 17 808 kW,以 1 元/(kW · h)单价计算,天河二号平均每年的电费约为 1.56 亿元[4-5]。随着云服务计算能力的提升,用户越来越关注计算系统的可靠性,但高可靠性往往伴随着更高的成本和更大的资源开销。可靠性是任务成功执行的关键,硬件故障、资源丢失、超时故障和网络故障等均会导致云计算的可靠性下降[6]。计算资源的利用率是影响云计算效率的关键因素,充分利用计算资源空闲时段以降低工作流执行成本并提高计算资源利用率至关重要[7]。在调度工作流任务时考虑负载均衡指标,可提高虚拟机工作负荷相对均衡度,利于系统稳定[8]。任务执行时间和通信开销反映资源消耗度,异构嵌入式系统中工作流的可靠性和资源消耗对提升其稳定性至关重要[9]。目前针对云工作流调度优化目标的研究主要集中在执行时间和费用两个指标,少部分研究涉及能耗和可靠性。然而,其他重要优化目标如资源利用率、负载均衡及资源消耗等很少被考虑。

云工作流调度为 NP 难问题[10]。目前,大部分研究工作将其建模为带有约束的单目标问题以及多目标问题,仅优化其中的两个或三个目标,采用启发式算法或演化算法等方法进行求解,已经取得了一定的进展。Rodriguez 和 Buyya[11]提出截止日期约束下成本最小化单目标工作流调度模型,旨在实现给定截止日期约束下以最小成本执行目标任务。针对该模型,多种求解算法相继被提出,如协同进化遗传算法[12]、动态分组学习分布式粒子群算法[13]、蒙特卡罗采样法[14]等。在考虑能耗约束的情况下,Shi 等[15]以最小化作业完成时间为优化目标,确定任务放置计划和资源分配计划。Xie 等[16]提出在给定响应时间范围内可靠性最大化的优化方法。此外,部分研究针对响应时间和可靠性约束条件下最小化能耗[17]和资源消耗[18]的工作流调度问题进行了研究。以上工作大多是面向单目标工作流调度优化。

部分研究直接采用多目标模型进行优化。Chen 等[19]将云工作流调度问题建模为双目标(时间和费用)优化问题,设计了一种基于协同进化多目标多种群框架的多目标蚁群算法;Li 等[20]针对该问题模型提出了一种基于评分和动态层次的非支配排序遗传算法。Li 等[21]在该问题模型的基础上增加了截止日期约束,并提出了一种基于多群协同进化的混合智能优化算法。Xue 等[22]构建了针对云环境下完工时间、成本和负载最优化的云工作流调度模型,采用参考向量引导进化算法获得合适的工作流调度策略。上述多目标云工作流调度模型仅考虑两个或三个目标,忽略了其他重要目标,缺乏对问题的全盘考虑,鲜有涉及四个及以上目标的云工作流调度研究工作,存在一定的局限性。将云工作流调度建模为超多目标优化问题,并提出相应的求解方法,可为云计算平台资源分配提供更丰富的决策信息。

超多目标通常指四个及以上目标,超多目标优化存在维度爆炸、支配阻抗、收敛性与多样性难以合理平衡以及非规则前沿面等难点。其中,维度爆炸容易导致寻优种群陷入局部最优;支配阻抗是指种群中非支配解的比例急剧增加,导致基于非支配排序的选择机制难以区分解的质量;多样性与收敛性难以平衡是指由于目标空间维度增大、搜索难度增加,解集的收敛性和多样性难以平衡;非规则前沿面使权重向量设计困难和种群多样性维持机制失效。在这一具有挑战性的背景下,需要一种更加高效、灵活且具备适应性的算法。

目前,多目标演化算法主要分为基于非支配排序、基于分解以及基于性能指标的方法[23],这些算法在处理超多目标问题时表现出显著的性能差异。随着非支配解的比例增加,基于非支配排序的选择机制效率急剧降低。基于分解的算法中设置合适的权向量和参考点较为困难,对复杂问题的适应性不佳。基于性能指标的方法利用解的性能指标贡献度来引导搜索,但单一性能指标难以全面刻画解的贡献度。协同演化利用多个互补的搜索机制对问题进行协作优化[24],被认为是弥补单一性能指标不足的有效手段。目前将协同演化算法应用于云工作流调度,尤其是将其应用于超多目标的云工作流调度问题中的研究比较少见。因此,基于协同演化机制,本文提出一种双阶段改进协同演化算法,旨在解决超多目标云工作流调度问题,该算法首先生成高质量解集,然后通过双档案集和多性能指标协同机制在搜索过程中平衡收敛性和多样性。

1 超多目标云工作流调度问题

将云工作流调度建模为包含 7 个目标的超多目标问题,具体目标为可靠性、执行时间、能耗、执行成本、资源空闲率、负载均衡度和资源消耗度,问题模型所涉及的符号和变量含义见表 1。云工作流采用有向无环图(direct acyclic graph, DAG)表示,其简单示例如图 1 所示。

表 1　符号和定义

Tab. 1　Symbols and definitions

符号	定义
G	云工作流
T_i	第 i 个任务
V_j	第 j 个虚拟机
n	任务数量
$\tilde{m}$	虚拟机数量
P_j	第 j 个虚拟机的计算能力
C_j	第 j 个虚拟机的单位时间租赁价格
DT_i	第 i 个任务数据大小
$T_{\text{pred}}(i)$	第 i 个任务的前驱
$T_{\text{succ}}(i)$	第 i 个任务的后继
$x[i]$	第 i 个任务选择的虚拟机编号
λ_j	虚拟机 V_j 的恒定故障率
β	虚拟机电路相关常数
Γ_{v}	虚拟机静态电路系数
γ	虚拟机增长率
γ^*	虚拟机通信成本增长率
$w_{i,j}$	第 i 个任务在第 j 个虚拟机上的执行时间
$e_{i,j}$	第 i 个任务到第 j 个任务的数据传输时间
$ST(i)$	第 i 个任务的开始时间
$ET(i)$	第 i 个任务的结束时间
$LST(j)$	第 j 个虚拟机的租赁开始时间
$LET(j)$	第 j 个虚拟机的租赁结束时间
$R(T_i,V_j)$	虚拟机 V_j 执行任务 T_i 的可靠性
$E(T_i,V_j)$	虚拟机 V_j 执行任务 T_i 产生的能耗
$u(j)$	虚拟机 V_j 的利用率
$idle(j)$	虚拟机 V_j 的空闲率
$R(G)$	总可靠性
TET	总执行时间
$E(G)$	总能耗
TEC	总执行成本
$IR(G)$	总空闲率
LB	负载均衡度
$RC(G)$	资源消耗度

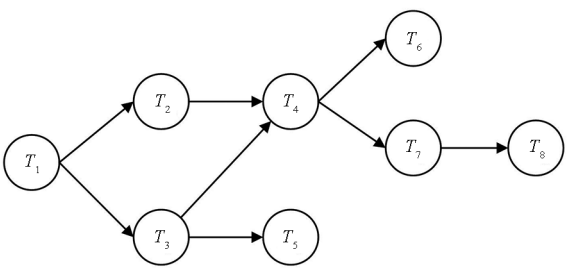


图 1　简单云工作流示例

Fig. 1　Simple cloud workflow example

1)设 $V=\{V_1,\cdots,V_{\tilde{m}}\}$ 是虚拟机集合,工作流 $G=(T,E)$,其中 $T=\{T_1,\cdots,T_n\}$ 代表任务集合,$E_{ij}\in E$ 代表任务之间的依赖关系,若 $E_{ij}=1$,则 T_j 的开始时间必须在 T_i 完成之后。工作流中第 i 个任务的调度方案可表示为 $S_i=[T_i,x[i],ST(i),ET(i)]$,其中 $x[i]$ 代表执行任务 T_i 的虚拟机编号,且每个任务只能分配给一个虚拟机,$ST(i)$ 和 $ET(i)$ 分别表示任务 T_i 的开始时间和结束时间。此外,虚拟机 V_j 的租赁开始时间和租赁结束时间分别为 $LST(j)$ 和 $LET(j)$。任务 T_i 在虚拟机 V_j 上的执行时间根据任务大小和虚拟机的计算能力采用式 $w_{i,j}=DT_i/P_j$ 得到。根据任务之间的依赖关系,T_i 的开始时间、结束时间以及对应虚拟机的租赁开始时间和租赁结束时间可采用递归的方式得到,具体计算方法如下:

$$ST(i)=\begin{cases}LET(x[i]) & T_{\text{pred}}(i)=\varnothing\\ \max\left(\max\limits_{T_k\in T_{\text{pred}}(i)}\{ET(k)\},LET(x[i])\right) & T_{\text{pred}}(i)\neq\varnothing\end{cases}\tag{1}$$

$$Transfer(i)=\sum_{\substack{T_k\in T_{\text{succ}}(i)\\ x[i]\neq x[k]}}e_{i,k}\tag{2}$$

$$ET(i)=ST(i)+w_{i,x[i]}+Transfer(i)\tag{3}$$

$$LST(x[i])=ST(i)\tag{4}$$

$$LET(x[i])=ET(i)\tag{5}$$

其中,$Transfer(i)$ 代表任务 T_i 在分配虚拟机之后传输数据所需时间。如果工作流中有多个输入和输出节点,则在对应位置设虚拟任务节点,初始任务的开始时间 ST 为 0。总执行时间 TET 和总执行成本 TEC 的计算如下:

$$TET=\max\{ET(i)\},1\leqslant i\leqslant|T|\tag{6}$$

$$TEC=\sum_{i=1}^{|T|}C_{x[i]}\times(LET(x[i])-LST(x[i]))\tag{7}$$

2)云工作流调度可靠性定义为虚拟机在给定时间间隔内连续工作不发生故障的概率,即在虚拟机 V_j 上执行任务 T_i 的可靠性为 $R(T_i,V_j)=e^{-\lambda_j\times w_{i,j}}$,其中 λ_j 代表虚拟机 V_j 的恒定故障率。整个云工作流的可靠性为:

$$R(G)=\prod_{T_i\in G}R(T_i,V_{x[i]})\tag{8}$$

3)能耗功率模型可表示为 $\Gamma=\Gamma_{\text{s}}+\Gamma_{\text{d}}=\Gamma_{\text{s}}+\beta\cdot f^p$,其中:$\Gamma_{\text{s}}$ 代表虚拟机待机静态功耗;Γ_{d} 代表虚拟机处理任务时动态功耗,Γ_{d} 取决于虚拟机的电路相关常数 β 和虚拟机频率 f;p 是动

态幂指数,通常取大于2的固定值。为简化起见,默认所有虚拟机运行时均以最大频率运行,频率f设为固定值1。因此,在虚拟机V_j上执行任务T_i的能耗为:

$$E(T_i,V_j)=(\Gamma_v(j)+\beta_j)\cdot w_{i,j} \tag{9}$$

式中,Γ_v代表虚拟机的静态电路系数。整个云工作流的总能耗为:

$$E(G)=\sum_{i=1}^{|T|}E(T_i,V_{x[i]}) \tag{10}$$

4)空闲率反映所分配的虚拟机有效利用的情况,其值越小,代表被租用的虚拟机空置时间越少。与之相对应的是虚拟机利用率,虚拟机空闲率越小,则虚拟机利用率越高。虚拟机V_j的利用率计算方式如下:

$$u(j)=\frac{\sum(ET(i)-ST(i))}{\max\{ET(i)\}-\min\{ST(i)\}},x[i]=j \tag{11}$$

当V_j上没有任何任务时,其利用率为0。虚拟机V_j的空闲率为$idle(j)=1-u(j)$,整个云工作流的空闲率为:

$$IR(G)=\sum_{j=1}^{|V|}idle(j) \tag{12}$$

5)负载均衡度反映虚拟机利用率的均衡程度,具体计算方法为:

$$LB=\frac{\sum_{j=1}^{|V|}(u(j)-\mu)^2}{|V|} \tag{13}$$

式中,μ代表所有虚拟机利用率的平均值。

6)资源消耗度取决于任务的执行时间和通信时间,用RC表示,设γ_j表示虚拟机V_j的增长率,对于在虚拟机V_j上执行的任务T_i,其计算代价为$\gamma_j\cdot w_{i,j}$。同时,假定通信成本随着通信开销而增长,并且对于所有处理器来说增长率是恒定的,γ^*表示通信成本的增长率,则任务T_i在虚拟机V_j上执行产生的资源消耗度为:

$$RC(T_i,V_j)=\gamma_j\cdot w_{i,j}+\sum_{T_k\in T_{\text{pred}}(i)}\gamma^*\cdot e_{k,i} \tag{14}$$

云工作流的资源消耗度为:

$$RC(G)=\sum_{i=1}^{|T|}RC(T_i,V_{x[i]}) \tag{15}$$

云工作流调度问题的目标为:

$$\min F=(TET,TEC,-R(G),E(G),IR(G),LB,RC(G)) \tag{16}$$

为便于理解,根据图1所示工作流,生成一个简单实例。设虚拟机数量为4,表2为随机生成的虚拟机各项参数。

表2 虚拟机参数

Tab. 2 Virtual machine parameters

虚拟机编号	单位时间成本	失败率	γ	γ^*	Γ_v	β
1	0.55	8×10^{-5}	1.17	0.5	0.06	1.1
2	0.29	7×10^{-5}	0.83	0.5	0.05	1.0
3	0.85	9×10^{-5}	1.50	0.5	0.05	1.1
4	0.01	6×10^{-5}	0.50	0.5	0.05	0.8

设任务在虚拟机上的执行时间矩阵$\boldsymbol{w}$和任务之间的数据传输时间矩阵$\boldsymbol{e}$为:

$$\boldsymbol{w}=\begin{array}{c}\\T_1\\T_2\\T_3\\T_4\\T_5\\T_6\\T_7\\T_8\end{array}\begin{array}{c}\begin{array}{cccc}V_1&V_2&V_3&V_4\end{array}\\\begin{bmatrix}8&4&4&3\\7&8&11&12\\4&5&4&4\\5&9&10&10\\4&9&7&10\\10&9&12&11\\5&7&3&3\\7&3&6&6\end{bmatrix}\end{array}$$

$$\boldsymbol{e}=\begin{array}{c}\\T_1\\T_2\\T_3\\T_4\\T_5\\T_6\\T_7\\T_8\end{array}\begin{array}{c}\begin{array}{cccccccc}T_1&T_2&T_3&T_4&T_5&T_6&T_7&T_8\end{array}\\\begin{bmatrix}0&2.87&1.27&0&0&0&0&0\\0&0&0&2.27&0&0&0&0\\0&0&0&2.23&0.29&0&0&0\\0&0&0&0&0&2.39&2.04&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1.52\\0&0&0&0&0&0&0&0\end{bmatrix}\end{array}$$

给定初始解$\boldsymbol{x}=(4,2,4,1,1,2,3,2)$,待虚拟机分配完成之后,任务执行顺序确定,解$\boldsymbol{x}$对应的甘特图如图2所示。图中白色方块代表任务在虚拟机上的执行时间,灰色方块代表任务之间的数据传输时间。

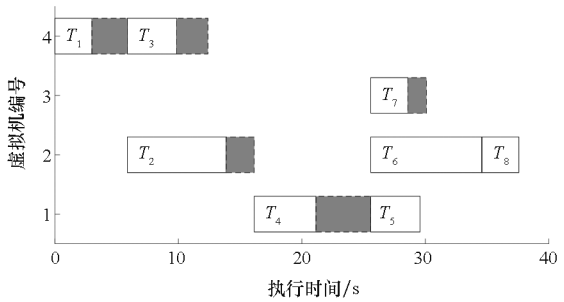


图2 解$\boldsymbol{x}$对应的甘特图

Fig. 2 Gantt chart corresponding to the solution $\boldsymbol{x}$

任务执行顺序确定后,根据前述公式,可得总执行时间 TET = 37.57 s,总执行成本 TEC = 17.85,总可靠性 $R(G)$ = 99.622 4%,总能耗 $E(G)$ =67.7,总空闲率 $IR(G)$ =0.297,负载均衡度 LB =0.017,资源消耗度 $RC(G)$ =41.97。

为对比初始解 $\boldsymbol{x}$ 得到的结果,随机生成解 $\boldsymbol{y}$ = (4,1,4,1,3,1,4,2),其对应的甘特图如图 3 所示。

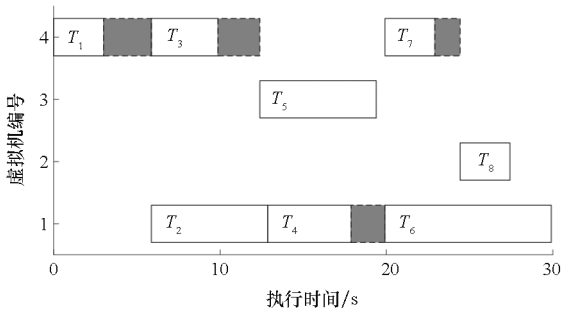


图 3　解 $\boldsymbol{y}$ 对应的甘特图

Fig. 3　Gantt chart corresponding to the solution $\boldsymbol{y}$

可得解 $\boldsymbol{y}$ 对应的总执行时间 TET = 29.91 s,总执行成本 TEC = 20.26,总可靠性 $R(G)$ = 99.622 9%,总能耗 $E(G)$ = 63.53,总空闲率 $IR(G)$ =0.308,负载均衡度 LB =0.018,资源消耗度 $RC(G)$ = 48.14。对比解 $\boldsymbol{x}$ 和解 $\boldsymbol{y}$ 的性能指标,可知解 $\boldsymbol{x}$ 在总执行时间、可靠性和能耗上差于解 $\boldsymbol{y}$,但其余目标上均优于解 $\boldsymbol{y}$,二者在不同目标上有各自优势。

云工作流调度问题的解空间具有组合爆炸特征,图 1 工作流包含 8 个任务和 4 个虚拟机,其搜索空间规模为 4^8 =65 536。随着任务数量和虚拟机数量的增加,搜索空间规模呈指数增长。此外,高维空间中不同目标冲突进一步加剧,对寻优算法提出了更高要求。

2　双阶段改进协同演化算法

由于超多目标优化问题存在维度爆炸和支配阻抗等特征,传统的演化算法在解决此类问题时寻优性能下降,并且高维空间中平衡解集的收敛性和多样性更加困难,为此,提出一种双阶段改进协同演化算法(dual-stage improved co-evolutionary algorithm, DSICEA):

1)引入双阶段策略使算法分阶段探索解空间。第一阶段通过集成指标选择综合性能较好的个体,作为第二阶段的初始解;第二阶段通过双档案集互补指标协同来引导种群搜索,兼顾收敛性和多样性。

2)在第一阶段搜索进程中,提出一种融合解集收敛性与多样性的集成指标,引导种群进化,以期提高解集质量,缓解支配阻抗问题。

3)对第二阶段中父代个体更新方式进行改进,从双档案集中选择父代个体,以提升生成新解的多样性,避免种群陷入局部最优。

2.1　算法框架

DSICEA 流程如图 4 所示,第一阶段采用集成指标获得综合性能较好的解集,第二阶段采用双指标双档案集引导种群进化,以更好地平衡解集收敛性和多样性。DSICEA 伪代码见算法 1。首先初始化种群 P 和第二阶段初始化标志位 $Initialized$(1 ~2 行),算法执行第一阶段搜索的条件为 $FE < t \times \max FEs$,其中 FE 代表当前迭代的评价次数,参数 t 控制第一阶段搜索资源投入占比。在第一阶段中(5 ~7 行),首先采用随机配对选择得到父代集合 $Parent$(5 行),通过遗传算子产生子代种群 Off (6 行),之后执行环境选择算子,通过集成指标选择个体得到更新后的种群 (7 行)。在第二阶段中(9 ~18 行),对 Two_Arch2 算法[25]进行延伸和拓展,首先对收敛性档案集 CA 和多样性档案集 DA 赋值 ,通过随机配对选择生成父代集合 $Parent_1$ 和 $Parent_2$。$Parent_2$ 在 Two_Arch2 中仅从档案集 CA 中选择个体,未考虑档案集 DA 中的个体,难以充分发挥档案集的作用,

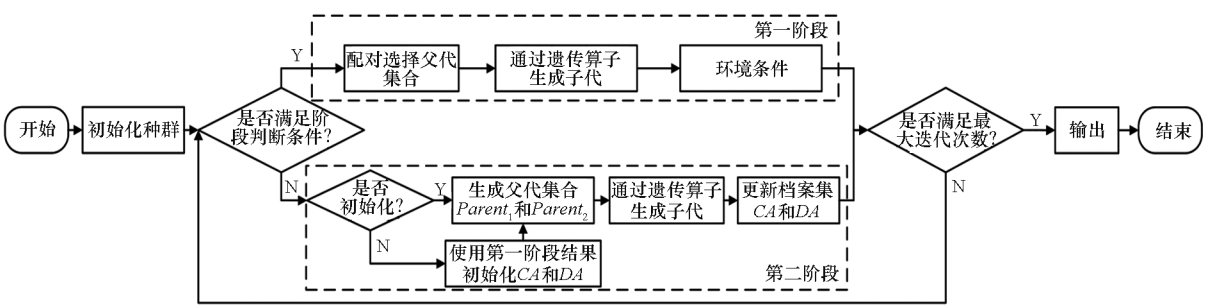


图 4　DSICEA 流程图

Fig. 4　DSICEA flow chart

算法 1 DSICEA 程序

Alg. 1 Procedure of DSICEA

输入:种群大小 N
最大评价次数 maxFEs
输出:最终种群 P

```
1. Initialize P
2. Initialized = 0
3. while FE < maxFEs do
4.   if FE < t × maxFEs then
5.     Parent←MatingSelection(P)
6.     Off←OperatorGA(Parent)
7.     P←EnvironmentalSelection(P∪Off)
8.   else
9.     if Initialized = 0 then
10.      CA = P
11.      DA = P
12.    end if
13.    [Parent1,Parent2]←MatingSelection(CA,DA)
14.    Off←OperatorGA(Parent1,Parent2)
15.    CA←UpdateCA(Off,CA)
16.    DA←UpdateDA(Off,DA)
17.    P = DA
18.    Initialized = 1
19.  end if
20. end while
21. return P
```

DSICEA 对此进行改进。通过遗传算子生成子代后根据指标 $I_{\epsilon+}$ 更新档案集 CA(15 行),根据 L_p 距离指标更新档案集 DA(16 行)。迭代进化完成后将多样性表现更好的档案集 DA 作为最终解集输出。

2.2 集成指标策略

DSICEA 第一阶段搜索过程中,采用集成指标 I 反映解的综合性能,其由两部分组成,其中第一部分 $I_1(x)$ 为:

$$I_1(x) = \sum_{y\in P, y\neq x} -\mathrm{e}^{-I_{\epsilon+}(x,y)/0.05} \tag{17}$$

$$I_{\epsilon+}(x,y) = \min_{\epsilon}(f_i(x) - \epsilon \leqslant f_i(y), i\in\{1,\cdots,m_d\}) \tag{18}$$

式中,m_d 为目标空间维度,$I_{\epsilon+}(x,y)$ 代表个体 x 在目标空间中支配个体 y 所需要的最小距离。$I_{\epsilon+}$ 具有支配保持的特性,若 x 支配 y,则 $I_{\epsilon+}(x,y)<0$。指标 I_1 仅和个体目标值相关,无须设置参考点或者相关参数,避免了参数设置对结果的影响。第二部分 $I_2(x)$ 可以反映个体的多样性,具体计算方式如下:

$$I_2(x) = \min_{y\in P, y \text{ precedes } x}\{I_{\mathrm{SDE}}(x,y)\} \tag{19}$$

$$I_{\mathrm{SDE}}(x,y) = \sqrt{\sum_{1\leqslant i\leqslant m} sd(f_i(x),f_i(y))^2} \tag{20}$$

$$sd(f_i(x),f_i(y)) = \begin{cases} f_i(y) - f_i(x) & f_i(x) < f_i(y) \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

其中:I_{SDE}通过距离反映个体的密度值,较小的距离值对应较大的密度值,在进化过程中更容易被淘汰,可有效维持种群多样性;y precedes x 代表个体 y 在种群 P 中的下标位置先于个体 x[26],经过验证,使用该设置不仅可减少计算开销,而且会使得 DSICEA 的寻优效果进一步提升。为平衡解集的收敛性和多样性,集成指标由 I_1 和 I_2 归一化处理后相加:

$$I(x) = \frac{I_1(x) - \min(I_1)}{\max(I_1) - \min(I_1)} + \frac{I_2(x) - \min(I_2)}{\max(I_2) - \min(I_2)} \tag{22}$$

$I(x)$的值越大,对应个体 x 质量越好。算法 2 给出了基于集成指标 I 的环境选择步骤。

算法 2 环境选择

Alg. 2 Environmental selection

输入:当前种群 P'
子代种群 Off
种群大小 N
输出:下一代种群 P

```
1. P←P′∪Off
2. I1←Compute each individual by (17)
3. I2←Compute each individual by (19)
4. I←Compute each individual by (22)
5. P←Select N individuals with maximum I
6. return P
```

2.3 父代集合更新策略

在 Two_Arch2 中,档案集 CA 维持收敛性,档案集 DA 维持多样性,并且 CA 在多样性方面远差于 DA。Two_Arch2 中父代集合 $Parent_2$ 通过随机方式只从 CA 当中生成,没有考虑 DA 中的个体。为了充分利用档案集中的信息,DSICEA 从 CA 和 DA 的合集中选择表现更好的个体组成 $Parent_2$,具体操作过程见算法 3。

2.4 复杂度分析

DSICEA 的时间复杂度与迭代次数 K、种群规模 N、目标个数 m 和任务数 $|T|$ 有关。DSICEA 包括两个阶段,第一阶段通过集成指标选择优势个

算法 3　配对选择 $Parent_2$

Alg. 3　Mating selection for $Parent_2$

输入:收敛性档案集 CA
多样性档案集 DA
种群大小 N
输出:$Parent_2$

1. P_1←Select $N/2$ individuals randomly from DA
2. P_2←Select $N/2$ individuals randomly from DA
3. **if** P_1 dominate P_2 **then**
4. $Parent_2$←Select individual from P_1
5. **else**
6. $Parent_2$←Select individual from P_2
7. **end if**
8. $Parent_2$←Select $N/2$ individuals randomly from CA
9. return $Parent_2$

体,该阶段时间复杂度为$O(0.3KmN^2|T|^2)$;第二阶段通过双档案集指导种群进化,该阶段时间复杂度为 $\max\{O(0.7KN\lg^{m-2}N|T|^2),O(0.7KmN^2|T|^2)\}$。综上,DSICEA 的总时间复杂度为$O(mKN^2|T|^2+\max\{KN\lg^{m-2}N|T|^2,mKN^2|T|^2\})$。

3　实验结果及分析

本节对 DSICEA 进行实验验证,首先分析参数 t 对算法性能的影响,随后通过消融实验检验改进策略的有效性,最后通过与目前主流方法进行对比实验,检验 DSICEA 的寻优能力。所有算法独立重复运行 30 次,并采用非参检验测试所得结果的显著性差异。

3.1　实验设置

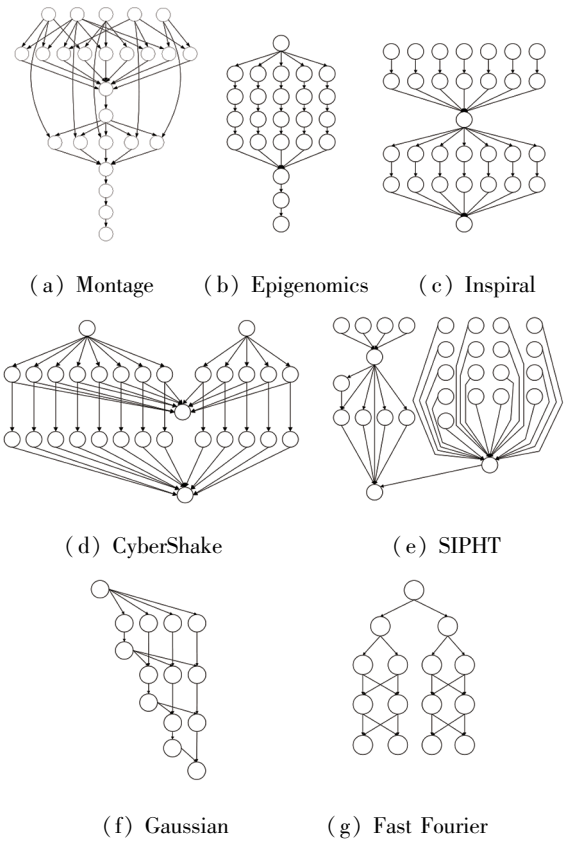
实验数据集为科学工作流 CyberShake、Epigenomics、Inspiral、Montage、SIPHT 以及 Gaussian 工作流和 Fast Fourier 工作流。每一类工作流均设置其对应的小、中、大不同规模的问题实例。

5 类科学工作流的所有实例数据及任务之间的依赖关系均来自南加州大学开发的一套开源工作流仿真软件的真实数据,Gaussian 和 Fast Fourier 工作流的所有实例数据随机生成。云服务中虚拟机的各项参数参考文献[27 – 28],其中虚拟机电路相关常数为 $0.8\leqslant\beta\leqslant1.2$,虚拟机静态电路系数为 $0.03\leqslant\Gamma_v\leqslant0.07$,虚拟机 V_j 的恒定故障率与虚拟机单位时间租赁价格线性相关,单位时间租赁价格越高的虚拟机代表其故障率越低,范围设为$[6\times10^{-5}/\text{ms},9\times10^{-5}/\text{ms}]$。结合实际情况将虚拟机增长率设为 $0.5\leqslant\gamma\leqslant1.2$ kbit/ms,虚拟机通信成本增长率设为 $\gamma^*=0.5$ kbit/ms。表 3 为不同规模工作流对应的任务数量和虚拟机数量,小规模任务场景下工作流的拓扑结构如图 5 所示。所有实验中种群规模 N 设为 100,目标函数最大评价次数 maxFEs 为 50 000。采用逆世代距离(inverted generational distance, IGD)对解集进行评价,IGD 值越小,代表解集的收敛性和分布性越好。

表 3　云工作流实例参数配置

Tab. 3　Parameter configurations of cloud workflow instances

工作流类型	工作流规模	任务数量	虚拟机数量
CyberShake	小/中/大	30/50/100	5/8/10
Epigenomics	小/中/大	24/47/100	5/8/10
Inspiral	小/中/大	30/50/100	5/8/10
Montage	小/中/大	25/50/100	5/8/10
SIPHT	小/中/大	29/58/97	5/8/10
Gaussian	小/中/大	14,35/77/135,209	4/8/16
Fast Fourier	小/中/大	15/39/95,223	4/8/16



(a) Montage　(b) Epigenomics　(c) Inspiral

(d) CyberShake　(e) SIPHT

(f) Gaussian　(g) Fast Fourier

图 5　云工作流拓扑结构

Fig. 5　Cloud workflow topology structures

$$IGD(PF,Q) = \frac{\sum_{v \in PF} dist(v,Q)}{|PF|} \tag{23}$$

其中，PF 代表真实 Pareto 解集，Q 代表算法输出的解集，$dist(v,Q)$ 代表 PF 中的个体 v 与 Q 中的目标向量之间的最小欧氏距离。由于所求问题理论最优前沿解集未知，因此通过多次运行所有参与比较的算法，从获取的解集中筛选出非支配解，以构建参考前沿解集。

3.2　参数分析

DSICEA 中的参数 t 用于控制第一阶段计算资源投入占比，为考察其取值对算法性能的影响，对比参数 t 的 9 个不同取值（从 0.1 至 0.9，间隔为 0.1），在 24 个问题实例上分别运行 30 次。表 4给出了单因素方差分析（analysis of variance，ANOVA）的结果，从表中可以得到 p 值远小于 0.05，表明参数 t 对算法有显著影响。为寻找参数 t 的最佳取值，观察 30 次运行得到的 IGD 指标值具体分布情况。图 6 给出了 IGD 指标归一化之后的具体结果，当 $t=0.3$ 时，IGD 值处于最佳状态，并且在该取值情况下，IGD 中位数的值明显好于其余情况。

表 4　单因素方差分析结果

Tab. 4　One-way analysis of variance results

误差来源	平方和	自由度	均方	F 统计量	p 值(F)
组间误差	6.720 1	8	0.840 02	9.44	2.090 2E－11
组内误差	22.418 7	252	0.088 96		
合计	29.138 8	260			

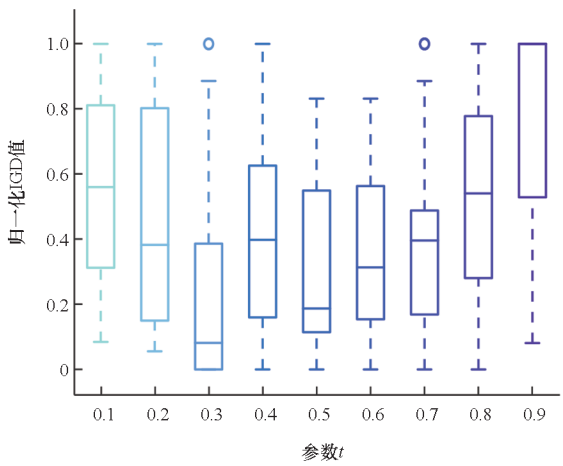


图 6　IGD 指标值箱线图

Fig. 6　Box plot of IGD indicator values

图 7 给出了参数 t 不同取值下在 24 个问题实例上运行 30 次计算得到 IGD 值的 Friedman 排名情况，从图中可以看出，$t=0.3$ 时排名第一。与图 6 对应，以 $t=0.3$ 为分界点，增大或减小 t 值，算法性能均有所降低。t 值过小，算法第一阶段中生成解集的质量欠佳，进而影响后续的进化过程；t 值过大，算法第一阶段消耗计算资源过多，导致第二阶段没有足够的计算资源对解集进行优化，使最终结果变差，当 t 取 0.3 时平衡效果较佳。

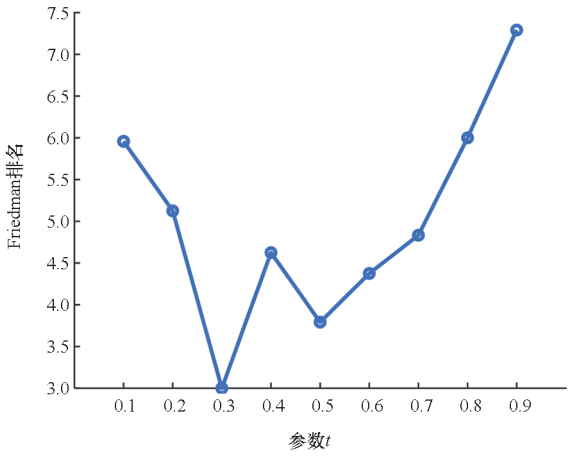


图 7　在不同 t 取值下所得 DSICEA 的 Friedman 排名

Fig. 7　Friedman ranking of DSICEA for different values of t

3.3　消融分析

为考察所提策略对算法性能的影响，设计了四种变体，记作 DSICEA-Ⅰ、DSICEA-Ⅱ、DSICEA-Ⅲ、DSICEA-Ⅳ。DSICEA-Ⅰ 中，I 被替换为 $I_{\epsilon+}$；DSICEA-Ⅱ 中 I 被替换为 I_{SDE}；DSICEA-Ⅲ 中，将第一阶段删除；DSICEA-Ⅳ 中，将第二阶段 $Parent_2$ 的更新方式用随机生成的方式替换。

表 5 给出了 DSICEA 与其他四种变体在 24 个问题实例上运行 30 次得到的 IGD 平均值、标准差及统计分析结果。Friedman 代表算法在所有实例上排名的平均值，括号中的数值代表算法的最终排名；Best 代表算法排名第一的问题数量和问题总数；表格最后给出了 Wilcoxon 秩和检验结果，其中 +/=/－ 分别代表 DSICEA 显著优于、相似于、劣于该算法的问题实例数量。表格中灰色背景数值为该问题上的 IGD 最优值，符号 ●/§/○ 代表 DSICEA 优于、相似于、劣于该算法。从表中数据可以观察到 DSICEA 在 11 个实例上表现效果最优，而剩下四种变体分别只在 3、3、1、6 个实例上表现最好，并且 DSICEA 只在 3 个实例上略差于 DSICEA-Ⅱ，在 1 个实例上略差于 DSICEA-Ⅳ，在剩余实例上 DSICEA 优于或相似于其他变体，且 DSICEA 的 Friedman 排名第一，证明改进策略有效。

表 5　DSICEA 与变体在所有问题实例上的 IGD 指标均值及标准差

Tab. 5　Mean and standard deviation of IGD metric for DSICEA and variants over all instances

问题实例	DSICEA	DSICEA-Ⅰ	DSICEA-Ⅱ	DSICEA-Ⅲ	DSICEA-Ⅳ
CyberShake_30	5.43E+1(2.3E+1)	6.39E+1(1.6E+1) §	5.94E+1(2.6E+1) §	7.62E+1(4.7E+1) ●	6.10E+1(2.0E+1) §
CyberShake_50	3.81E+2(1.5E+2)	3.90E+2(1.5E+2) §	4.13E+2(1.8E+2) §	5.14E+2(1.8E+2) ●	2.76E+2(1.4E+2) ○
CyberShake_100	6.24E+2(2.6E+2)	5.74E+2(1.9E+2) §	5.96E+2(1.9E+2) §	9.24E+2(4.0E+2) ●	5.99E+2(2.8E+2) §
Epigenomics_24	5.52E+1(5.9E+0)	8.46E+1(1.7E+1) ●	5.56E+1(5.1E+0) §	5.71E+1(8.3E+0) §	5.56E+1(5.1E+0) §
Epigenomics_47	2.19E+2(4.0E+1)	2.75E+2(9.7E+1) ●	2.22E+2(4.1E+1) §	2.51E+2(5.0E+1) ●	2.25E+2(4.3E+1) §
Epigenomics_100	4.18E+3(2.0E+2)	4.34E+3(3.2E+2) ●	4.28E+3(2.8E+2) §	4.37E+3(4.7E+2) ●	4.26E+3(2.7E+2) §
Fast Fourier_15	2.25E+1(2.3E+1)	2.96E+1(2.5E+1) ●	1.22E+1(1.5E+1) ○	3.69E+1(3.2E+1) ●	1.49E+1(1.7E+1) §
Fast Fourier_39	1.48E+2(1.1E+2)	1.45E+2(7.4E+1) §	1.47E+2(1.1E+2) §	3.03E+2(1.9E+2) ●	1.43E+2(9.6E+1) §
Fast Fourier_95	3.87E+2(1.8E+2)	4.69E+2(2.6E+2) §	5.03E+2(2.1E+2) ●	1.00E+3(4.3E+2) ●	5.44E+2(2.4E+2) ●
Fast Fourier_223	1.50E+3(6.7E+2)	1.35E+3(7.6E+2) §	1.92E+3(8.9E+2) §	4.36E+3(9.6E+2) ●	1.28E+3(7.0E+2) §
Gaussian_14	2.16E+1(4.4E+1)	1.76E+1(2.5E+1) §	8.84E+0(1.1E+1) ○	6.88E+1(1.0E+2) ●	7.90E+0(1.9E+0) §
Gaussian_35	1.50E+2(6.5E+1)	1.80E+2(6.0E+1) §	1.73E+2(7.6E+1) §	1.69E+2(6.4E+1) §	1.77E+2(3.3E+1) §
Gaussian_77	3.99E+2(1.5E+2)	3.41E+2(1.1E+2) §	4.37E+2(2.1E+2) §	6.47E+2(3.1E+2) ●	3.81E+2(1.2E+2) §
Gaussian_135	8.00E+2(2.3E+2)	7.47E+2(2.1E+2) §	1.07E+3(4.2E+2) ●	1.64E+3(4.7E+2) ●	7.42E+2(1.6E+2) §
Gaussian_209	1.35E+3(5.7E+2)	1.42E+3(4.8E+2) §	1.27E+3(6.1E+2) §	2.33E+3(8.8E+2) ●	1.31E+3(6.2E+2) §
Inspiral_30	2.96E+1(2.8E+0)	3.93E+1(9.8E+0) ●	3.08E+1(7.1E+0) §	3.32E+1(7.9E+0) §	2.97E+1(3.7E+0) §
Inspiral_50	1.11E+2(1.0E+1)	1.36E+2(2.1E+1) ●	1.15E+2(1.2E+1) §	1.17E+2(1.2E+1) §	1.17E+2(1.1E+1) ●
Inspiral_100	2.70E+2(5.1E+1)	2.83E+2(7.1E+1) §	2.49E+2(4.3E+1) §	3.62E+2(7.7E+1) ●	2.76E+2(6.6E+1) §
Montage_25	4.33E+0(2.7E+0)	3.38E+0(1.3E+0) §	3.72E+0(2.5E+0) §	8.54E+0(5.2E+0) ●	3.46E+0(2.5E+0) §
Montage_50	1.08E+1(3.4E+0)	1.28E+1(4.9E+0) §	1.40E+1(6.7E+0) ●	2.56E+1(1.1E+1) ●	1.31E+1(6.6E+0) §
Montage_100	4.22E+1(8.8E+0)	4.57E+1(9.3E+0) §	4.56E+1(1.2E+1) §	9.22E+1(4.1E+1) ●	4.66E+1(9.8E+0) ●
SIPHT_29	4.12E+1(2.6E+0)	4.43E+1(7.3E+0) §	4.20E+1(3.4E+0) §	4.18E+1(1.9E+0) §	4.24E+1(5.8E+0) §
SIPHT_58	1.81E+2(6.0E+1)	2.46E+2(1.2E+2) ●	1.70E+2(8.1E+1) ○	1.65E+2(6.6E+1) §	1.67E+2(6.6E+1) §
SIPHT_97	2.96E+2(3.6E+1)	3.25E+2(5.9E+1) §	2.90E+2(3.3E+1) §	3.63E+2(6.1E+1) ●	2.83E+2(3.5E+1) §
Friedman	2.208 3(1)	3.375(4)	2.625(3)	4.416 7(5)	2.375(2)
Best	11/24	3/24	3/24	1/24	6/24
+/=/−		7/17/0	3/18/3	18/6/0	3/20/1

实验结果表明,第一阶段生成的高质量解集对第二阶段双档案集协同搜索具有积极作用,使得最终解集质量更好。仅使用有偏好的单一性能指标 $I_{\epsilon+}$ 或 I_{SDE} 无法兼顾收敛性和多样性,难以产生综合性能较好的解集。相较于仅从 *CA* 中选择配对个体,从 *CA* 与 *DA* 的合集中选择配对个体可提升配对个体的多样性,一定程度上避免陷入局部最优,进而提高算法寻优能力。

3.4　对比实验

为验证 DSICEA 的寻优能力,将其与目前主流方法 AGE-MOEA[29]、BCE-IBEA[30]、BCE-MOEA/D[30]、NSGA-Ⅲ[31]、SPEA2[32]、Two _ Arch2 和 MSPSO[33]进行对比。表6 给出了8 种算法在24 个实例上运行 30 次得到的 IGD 平均值、标准差及统计分析结果。DSICEA 的 Friedman 排名为 1.166 7,在 8 种算法当中排名第一。DSICEA 在 20 个问题实例上表现最优,且分别在 19、21、17、24、21、19、16 个实例上显著优于其余算法,仅在 CyberShake 规模为 100 和 SIPHT 规模为 58 时表现欠佳。

Friedman 检验结果为:Friedman 值为 109.458 3,大于临界值 $\chi^2 = 14.07$,因此在显著性水平 $\alpha = 0.05$ 的情况下,8 种算法在 24 个实例上运行 30 次得到的平均 IGD 值存在显著差异。结合 Bonferroni-Dunntest 计算得到临界差 $CD = 2.143\ 2$,算法 Friedman 排名差值大于该值即存在显著差异。8 种算法的平均排名如图 8 所示,阈值 $Threshold = 3.309\ 9$ 小于所有对比算法的排名,且 DSICEA 的平均排名与对比算法有明显差距,充分证明了 DSICEA 远远优于其余 7 种算法。

表 6　对比算法在全部实例上的 IGD 指标均值及标准差

Tab. 6　Mean and standard deviation of IGD metric of compared algorithms over all instances

问题实例	DSICEA	AGE-MOEA	BCE-IBEA	BCE-MOEA/D	MSPSO	NSGA-Ⅲ	SPEA2	Two_Arch2
CyberShake_30	5.43E+1 (2.3E+1)	6.17E+1 (2.5E+1) §	6.56E+1 (2.1E+1) ●	7.24E+1 (2.2E+1) ●	5.53E+2 (2.6E+2) ●	6.68E+1 (2.7E+1) §	5.98E+1 (1.7E+1) §	8.44E+1 (5.2E+1) ●
CyberShake_50	3.81E+2 (1.5E+2)	3.89E+2 (2.1E+2) §	4.62E+2 (1.5E+2) ●	4.66E+2 (1.7E+2) §	1.64E+3 (2.5E+2) ●	3.63E+2 (1.8E+2) §	4.03E+2 (1.6E+2) §	4.80E+2 (1.8E+2) ●
CyberShake_100	6.24E+2 (2.6E+2)	6.93E+2 (3.4E+2) §	8.75E+2 (3.5E+2) ●	4.71E+2 (2.2E+2) ○	6.75E+3 (1.3E+3) ●	8.63E+2 (2.3E+2) ●	1.31E+3 (4.4E+2) ●	8.40E+2 (3.0E+2) ●
Epigenomics_24	5.53E+1 (5.9E+0)	9.53E+1 (1.1E+1) ●	8.74E+1 (8.9E+0) ●	1.19E+2 (1.4E+1) ●	5.27E+2 (1.5E+2) ●	1.78E+2 (1.9E+1) ●	9.08E+1 (2.1E+1) ●	5.75E+1 (6.7E+0) §
Epigenomics_47	2.19E+2 (4.0E+1)	3.58E+2 (8.7E+1) ●	3.11E+2 (7.7E+1) ●	4.20E+2 (1.9E+2) ●	6.53E+3 (1.3E+3) ●	1.55E+3 (2.7E+2) ●	3.81E+2 (7.4E+1) ●	2.94E+2 (1.1E+2) ●
Epigenomics_100	4.18E+3 (2.0E+2)	5.22E+3 (5.0E+2) ●	5.17E+3 (7.0E+2) ●	6.45E+3 (1.0E+3) ●	7.84E+4 (1.2E+4) ●	1.67E+4 (6.6E+3) ●	5.29E+3 (3.8E+2) ●	4.34E+3 (5.4E+2) §
Fast Fourier_15	2.25E+1 (2.3E+1)	3.32E+1 (2.4E+1) ●	3.02E+1 (1.9E+1) ●	3.17E+1 (2.4E+1) §	2.04E+2 (1.0E+2) ●	3.35E+1 (1.8E+1) ●	2.89E+1 (2.3E+1) §	3.14E+1 (2.5E+1) ●
Fast Fourier_39	1.48E+2 (1.1E+2)	2.43E+2 (1.6E+2) ●	2.00E+2 (1.2E+2) §	2.10E+2 (1.2E+2) ●	1.07E+3 (3.0E+2) ●	2.51E+2 (1.4E+2) ●	2.06E+2 (1.1E+2) ●	2.87E+2 (2.0E+2) ●
Fast Fourier_95	3.87E+2 (1.8E+2)	8.29E+2 (3.2E+2) ●	6.45E+2 (2.5E+2) ●	5.12E+2 (3.0E+2) §	2.63E+3 (3.8E+2) ●	6.87E+2 (2.8E+2) ●	6.71E+2 (3.2E+2) ●	8.57E+2 (3.2E+2) ●
Fast Fourier_223	1.50E+3 (6.7E+2)	3.04E+3 (9.8E+2) ●	2.62E+3 (9.5E+2) ●	2.52E+3 (9.4E+2) ●	1.47E+4 (1.1E+3) ●	1.93E+3 (8.1E+2) ●	3.15E+3 (7.8E+2) ●	3.56E+3 (1.1E+3) ●
Gaussian_14	2.17E+1 (4.4E+1)	5.64E+1 (5.8E+1) ●	2.67E+1 (5.3E+1) §	6.24E+1 (6.8E+1) ●	2.14E+2 (1.5E+2) ●	4.21E+1 (6.2E+1) §	3.64E+1 (5.6E+1) §	4.30E+1 (7.3E+1) §
Gaussian_35	1.50E+2 (6.5E+1)	1.71E+2 (6.6E+1) §	1.75E+2 (4.9E+1) §	1.83E+2 (5.0E+1) ●	7.66E+2 (2.0E+2) ●	2.33E+2 (6.3E+1) ●	1.88E+2 (7.3E+1) ●	1.77E+2 (7.1E+1) §
Gaussian_77	3.99E+2 (1.5E+2)	5.60E+2 (2.5E+2) ●	5.34E+2 (2.4E+2) ●	4.29E+2 (1.5E+2) §	2.15E+3 (4.7E+2) ●	6.88E+2 (2.7E+2) ●	6.85E+2 (2.2E+2) ●	6.14E+2 (3.1E+2) ●
Gaussian_135	8.00E+2 (2.3E+2)	1.41E+3 (6.1E+2) ●	1.32E+3 (3.9E+2) ●	8.55E+2 (4.1E+2) §	6.39E+3 (1.1E+3) ●	1.56E+3 (5.6E+2) ●	1.35E+3 (5.4E+2) ●	1.73E+3 (5.7E+2) ●
Gaussian_209	1.50E+3 (5.3E+2)	1.98E+3 (6.1E+2) ●	2.00E+3 (7.2E+2) ●	1.92E+3 (6.8E+2) ●	1.30E+4 (1.7E+3) ●	2.19E+3 (5.8E+2) ●	2.86E+3 (9.6E+2) ●	2.17E+3 (6.1E+2) ●
Inspiral_30	2.96E+1 (2.8E+0)	4.41E+1 (1.3E+1) ●	3.71E+1 (7.6E+0) ●	4.81E+1 (6.2E+0) ●	7.50E+2 (2.6E+2) ●	1.36E+2 (4.2E+1) ●	4.48E+1 (5.7E+0) ●	3.08E+1 (5.1E+0) §
Inspiral_50	1.11E+2 (1.0E+1)	1.37E+2 (1.4E+1) ●	1.34E+2 (1.2E+1) ●	1.64E+2 (1.6E+1) ●	9.30E+2 (1.2E+2) ●	3.48E+2 (8.3E+1) ●	1.86E+2 (2.5E+1) ●	1.19E+2 (1.2E+1) ●
Inspiral_100	2.70E+2 (5.1E+1)	4.09E+2 (1.1E+2) ●	3.99E+2 (1.2E+2) ●	4.08E+2 (9.5E+1) ●	4.42E+3 (4.0E+2) ●	1.79E+3 (3.2E+2) ●	5.58E+2 (1.3E+2) ●	3.33E+2 (8.0E+1) ●
Montage_25	4.33E+0 (2.7E+0)	7.05E+0 (4.9E+0) ●	7.63E+0 (7.0E+0) ●	4.49E+0 (2.7E+0) §	1.43E+2 (3.4E+1) ●	9.61E+0 (6.1E+0) ●	4.98E+0 (3.9E+0) §	7.81E+0 (5.2E+0) ●
Montage_50	1.08E+1 (3.4E+0)	2.53E+1 (1.3E+1) ●	1.91E+1 (1.1E+1) ●	1.73E+1 (8.6E+0) ●	4.04E+2 (4.9E+1) ●	3.46E+1 (1.2E+1) ●	2.54E+1 (1.4E+1) ●	2.28E+1 (1.0E+1) ●
Montage_100	4.22E+1 (8.8E+0)	8.89E+1 (2.5E+1) ●	7.14E+1 (2.4E+1) ●	6.36E+1 (2.2E+1) ●	1.01E+3 (7.9E+1) ●	1.01E+2 (2.1E+1) ●	1.03E+2 (2.5E+1) ●	8.80E+1 (2.6E+1) ●
SIPHT_29	4.13E+1 (2.6E+0)	6.21E+1 (7.6E+0) ●	5.60E+1 (6.5E+0) ●	7.31E+1 (8.0E+0) ●	1.00E+3 (4.0E+2) ●	1.72E+2 (6.0E+1) ●	6.21E+1 (3.8E+0) ●	4.09E+1 (2.1E+0) §
SIPHT_58	1.81E+2 (6.0E+1)	1.94E+2 (7.0E+1) §	2.47E+2 (1.6E+2) ●	2.51E+2 (7.5E+1) ●	1.27E+3 (1.7E+2) ●	5.14E+2 (2.3E+2) ●	2.62E+2 (3.6E+1) ●	1.56E+2 (3.4E+1) ○
SIPHT_97	3.00E+2 (3.6E+1)	3.82E+2 (5.1E+1) ●	3.26E+2 (4.2E+1) ●	4.37E+2 (9.2E+1) ●	5.42E+3 (6.0E+2) ●	7.27E+2 (1.2E+2) ●	4.80E+2 (4.2E+1) ●	3.11E+2 (5.5E+1) §
Friedman	1.166 7 (1)	4.25 (5)	3.375 (2)	4.166 7 (4)	8 (8)	6 (7)	4.916 7 (6)	4.125 (3)
Best	20/24	0/24	0/24	1/24	0/24	1/24	0/24	2/24
+/=/−		19/5/0	21/3/0	17/6/1	24/0/0	21/3/0	19/5/0	16/7/1

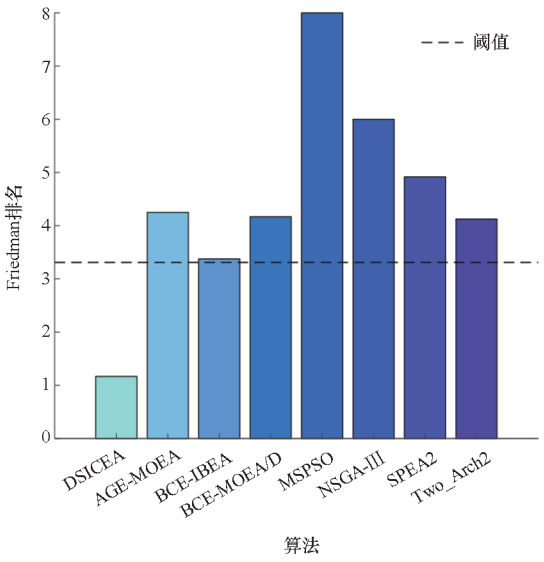


图 8　对比算法的 Friedman 排名

Fig. 8　Friedman ranking of compared algorithms

图 9 和图 10 给出了 8 种算法在大规模 Gaussian 工作流和 Montage 科学工作流上 IGD 度量值中位数对应的归一化解集的对比情况,子图中每条不同颜色的折线代表一个可行解。AGE-MOEA 和 NSGA-Ⅲ解集收敛性较好,但在多样性方面表现较差;BCE-IBEA、BCE-MOEA/D 和 Two_Arch2 在收敛性方面表现较差;BCE-MOEA/D 和 SPEA2 在多样性方面表现较优,但解集整体质量效果欠佳。MSPSO 易陷入局部最优且易出现只关注某一目标而忽略其余目标的情况。结果表明 7 种对比算法无法同时兼顾收敛性和多样性,而 DSICEA 拥有良好的收敛性和多样性,且得到的解集质量更优。

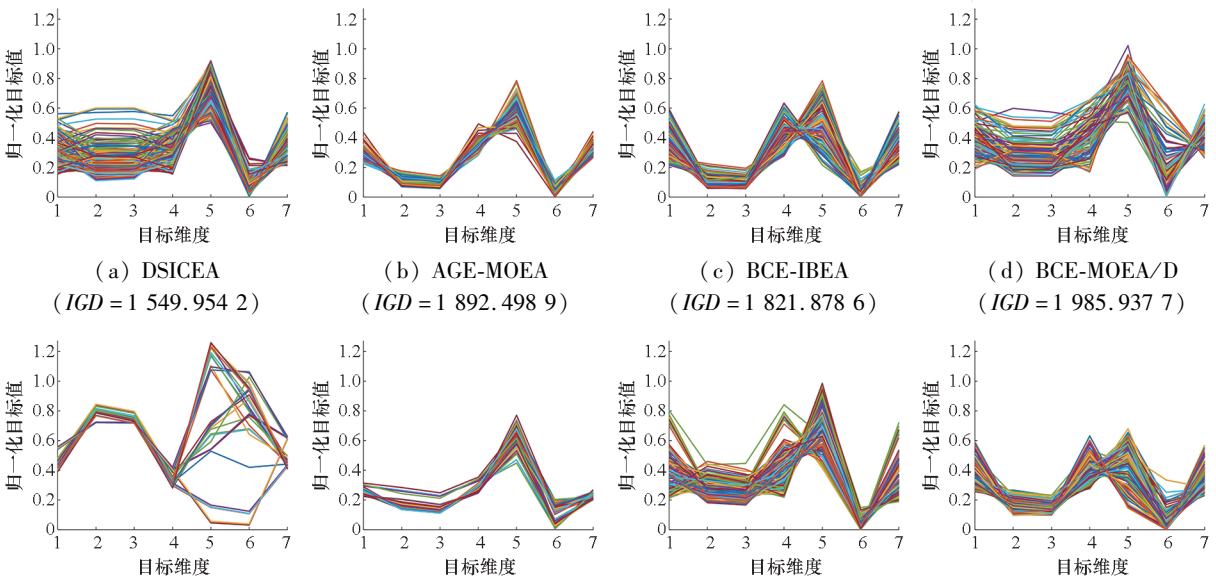
造成上述结果的可能原因如下:

1) AGE-MOEA 对前沿解集的几何形状进行预估,通过多样性和收敛度指标来选择优势个体,然而该策略难以应对高维多目标前沿面,易陷入局部最优。DSICEA 在进化过程中不依赖前沿面的形状,通过集成指标和双档案集协同引导种群搜索,有效提升算法的鲁棒性。

2) BCE-IBEA 和 BCE-MOEA/D 中支配准则和 SPEA2 的支配强度准则无法解决超多目标问题中的支配阻抗问题,收敛压力不足,致使最终解集质量变差。DSICEA 使用多阶段策略及集成指标和多指标协同机制引导种群进化,有效缓解了支配阻抗问题,进而提升算法寻优能力。

3) 在高维目标空间中,BCE-MOEA/D 算法和 NSGA-Ⅲ算法面临维度爆炸和种群数量限制的挑战,导致其基于分解的策略和基于参考点的策略无法发挥作用。具体而言,预设的权重向量和参考点可能难以适应实际问题,导致寻优种群无法充分探索目标空间,容易陷入局部最优,影响解集的质量。同时,种群数量的限制也是一个制约因素,限制了算法在高维目标空间中的搜索能力,无法寻找到高质量解集。DSICEA 不受预设参考向量或参考点限制,对非规则前沿面适应性较好。

4) MSPSO 通过适应度函数将多目标问题转化为单目标问题,在迭代过程中易出现仅优化其

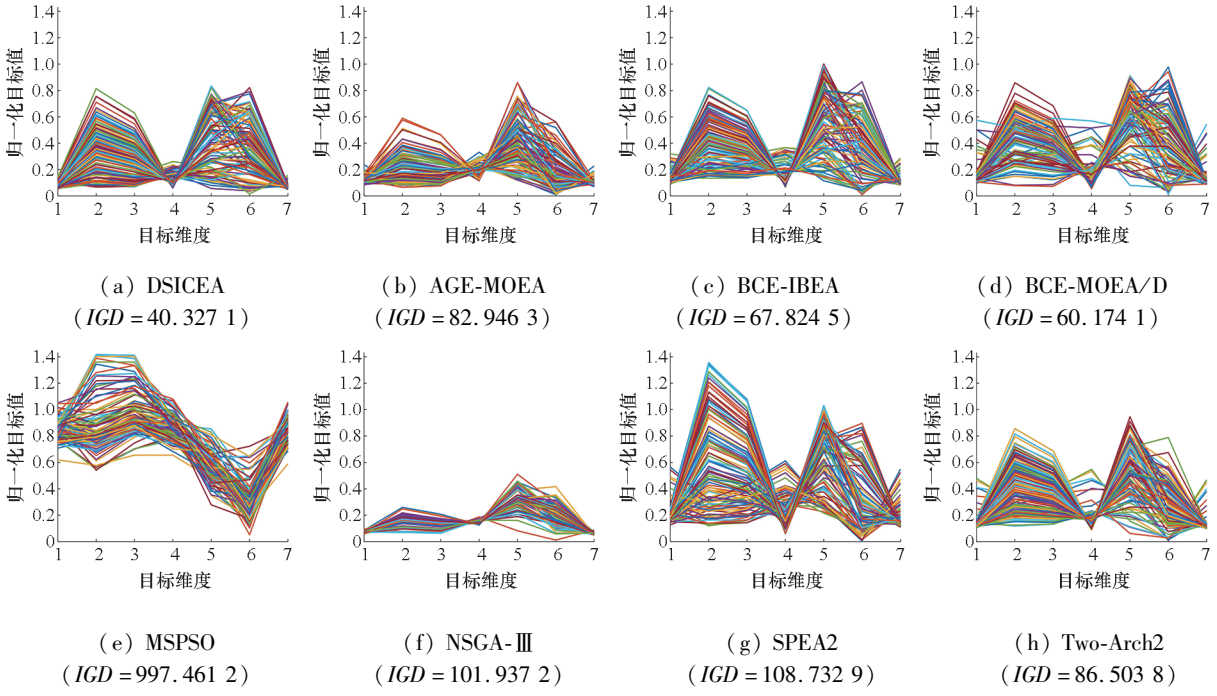


(a) DSICEA (*IGD* = 1 549.954 2)　(b) AGE-MOEA (*IGD* = 1 892.498 9)　(c) BCE-IBEA (*IGD* = 1 821.878 6)　(d) BCE-MOEA/D (*IGD* = 1 985.937 7)

(e) MSPSO (*IGD* = 12 908.085 1)　(f) NSGA Ⅱ (*IGD* = 2 016.755 1)　(g) SPEA2 (*IGD* = 2 816.900 4)　(h) Two_Arch2 (*IGD* = 2 164.637 7)

图 9　对比算法在 7 目标大规模 Gaussian 工作流上所得解集的目标值的平行坐标

Fig. 9　The final solution sets of compared algorithms on the 7-objective large-scale Gaussian workflow, shown by parallel coordinates



（a）DSICEA（*IGD* = 40.327 1）　（b）AGE-MOEA（*IGD* = 82.946 3）　（c）BCE-IBEA（*IGD* = 67.824 5）　（d）BCE-MOEA/D（*IGD* = 60.174 1）

（e）MSPSO（*IGD* = 997.461 2）　（f）NSGA-Ⅲ（*IGD* = 101.937 2）　（g）SPEA2（*IGD* = 108.732 9）　（h）Two-Arch2（*IGD* = 86.503 8）

图 10　对比算法在 7 目标大规模 Montage 科学工作流上所得解集的目标值的平行坐标

Fig. 10　The final solution sets of compared algorithms on the 7-objective large-scale Montage workflow, shown by parallel coordinates

中某一目标而忽略了其余目标的情况，且 MSPSO 虽然使用不同种类的粒子组成的多个子群来提高算法的探索能力，但在高维空间易陷入局部最优，难以充分探索目标空间。DSICEA 使用集成指标在第一阶段选择优秀个体，在第二阶段通过双档案集指导种群进化可以有效避免种群陷入局部最优，并提升解集质量。

5）Two_Arch2 通过多样性档案集和收敛性档案集协同指导种群的进化，但是随机初始化档案集质量难以保证，导致双档案集协同搜索效果欠佳。DSICEA 通过第一阶段生成高质量解集作为第二阶段的初始解，可以有效发挥双档案集的指导作用，加快种群进化速度，进而提升算法性能。

3.5　运行时间

所有实验在一台 Core i5 2.3 GHz、16 GB RAM、Windows 10 的笔记本电脑上使用 PlatEMO 平台[34]运行，表 7 记录了 8 种算法在 24 个问题实例上运行 30 次的平均运行时间。DSICEA 使用多性能指标协同机制平衡解集的收敛性和多样性，增加了一部分额外的计算开销，因此在运行时间方面 DSICEA 无法取得较大的优势。但随着问题规模的增加，DSICEA 与对比算法之间的差距逐渐缩小，证明 DSICEA 有着良好的可扩展性和适应性。MSPSO 算法通过适应度函数对比个体好坏，因此运行时间最短，但其对问题的优化效果最差。Two_Arch2 算法因维护双档案集造成了较大的运行时间开销。BCE-MOEA/D 算法的分解策略和支配准则无法缓解维度爆炸和支配阻抗等问题，导致了运行时间增加。虽然 DSICEA 在运行时间方面无法超越所有对比算法，但 DSICEA 中的双阶段策略和多性能指标协同机制可以有效平衡解集的收敛性与多样性，充分探索目标空间，提升解集质量，其效果显著优于对比算法。

4　结论

研究将云工作流调度建模为超多目标优化问题，提出改进协同演化算法，通过双阶段策略使算法分阶段探索解空间，有效平衡收敛性和多样性，避免陷入局部最优。第一阶段中使用集成性能指标引导种群搜索，缓解支配阻抗，通过改进第二阶段中父代更新方式以提升算法的寻优能力。在 7 类工作流实例上的测试结果表明，DSICEA 在收敛性、多样性和解集质量等方面均有更好的表现，算法的寻优能力远优于目前主流演化算法。

今后可尝试将不同特征的指标结合运用在算法的第一阶段生成更高质量的解集，还可考虑将不同的算法融入第二阶段当中引导算法在保持多样性的同时快速收敛，对于第一阶段搜索资源投入占比的控制参数 t 的自适应研究同样也是一个具有前景的方向。

表 7　对比算法在全部实例上的平均运行时间

Tab. 7　Average runtime of compared algorithms on all instances

单位:s

问题实例	DSICEA	AGE-MOEA	BCE-IBEA	BCE-MOEA/D	MSPSO	NSGA-Ⅲ	SPEA2	Two_Arch2
CyberShake_30	51.38	29.82	35.50	59.22	19.17	21.51	37.31	51.24
CyberShake_50	58.15	33.67	38.48	61.63	27.72	30.51	41.98	52.94
CyberShake_100	73.99	56.97	59.47	81.86	54.23	50.96	61.67	72.04
Epigenomics_24	48.33	27.92	34.53	54.36	16.07	18.20	44.34	53.94
Epigenomics_47	51.34	32.85	38.97	67.42	27.82	27.75	41.23	52.13
Epigenomics_100	77.63	58.14	64.51	91.99	55.81	49.63	79.83	82.91
Fast Fourier_15	46.12	22.21	28.01	51.91	12.37	16.26	36.23	48.45
Fast Fourier_39	57.00	30.83	39.03	62.93	20.07	25.59	41.67	56.88
Fast Fourier_95	81.06	56.28	62.75	90.06	45.01	51.55	65.39	84.24
Fast Fourier_223	155.24	130.15	137.85	173.73	112.33	125.54	136.22	149.48
Gaussian_14	50.98	22.06	31.24	55.79	9.62	14.59	49.95	52.07
Gaussian_35	52.54	30.39	35.86	60.20	19.11	25.22	37.79	51.94
Gaussian_77	69.32	45.83	53.77	73.75	36.12	40.85	52.04	68.63
Gaussian_135	96.44	75.72	78.58	101.66	61.90	68.89	80.51	96.69
Gaussian_209	137.27	117.28	122.74	151.21	113.01	115.32	127.27	143.16
Inspiral_30	48.63	28.57	32.86	56.17	17.80	19.82	40.66	51.20
Inspiral_50	55.84	35.44	41.39	67.25	26.44	28.15	54.73	59.67
Inspiral_100	76.97	54.41	60.29	91.63	48.64	48.04	82.53	80.33
Montage_25	48.18	25.12	30.89	53.13	16.88	19.49	32.06	52.11
Montage_50	55.75	33.74	43.31	64.45	26.93	27.41	36.70	50.57
Montage_100	71.18	50.50	56.02	80.97	48.29	47.94	56.41	69.01
SIPHT_29	46.10	27.28	31.08	51.48	13.56	16.61	48.78	51.82
SIPHT_58	52.99	33.19	38.33	62.62	24.46	25.59	56.20	58.69
SIPHT_97	66.35	45.79	52.13	83.62	36.87	40.40	72.18	73.29

参考文献(References)

[1] MADHUSUDHAN H S, SATISH K T, GUPTA P, et al. A Harris Hawk optimisation system for energy and resource efficient virtual machine placement in cloud data centers[J]. PLoS One, 2023, 18(8): e0289156.

[2] XIE Y, GUI F X, WANG W J, et al. A two-stage multi-population genetic algorithm with heuristics for workflow scheduling in heterogeneous distributed computing environments[J]. IEEE Transactions on Cloud Computing, 2023, 11(2): 1446 – 1460.

[3] LI H F, XU G H, WANG D J, et al. Chaotic-nondominated-sorting owl search algorithm for energy-aware multi-workflow scheduling in hybrid clouds[J]. IEEE Transactions on Sustainable Computing, 2022, 7(3): 595 – 608.

[4] ZHANG L X, LI K L, LI C Y, et al. Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems[J]. Information Sciences, 2017, 379: 241 – 256.

[5] HOSSEINZADEH M, GHAFOUR M Y, HAMA H K, et al. Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review[J]. Journal of Grid Computing, 2020, 18: 327 – 356.

[6] YE L J, XIA Y Q, TAO S Y, et al. Reliability-aware and energy-efficient workflow scheduling in IaaS clouds[J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(3): 2156 – 2169.

[7] WANG Y, ZUO X Q. An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules[J]. IEEE/CAA Journal of Automatica Sinica, 2021, 8(5): 1079 – 1094.

[8] GUPTA I, GUPTA S, CHOUDHARY A, et al. A hybrid meta-heuristic approach for load balanced workflow scheduling in IaaS cloud[C]//Proceedings of the 15th International Conference on Distributed Computing and Internet Technology, 2019.

[9] XIE G Q, CHEN Y K, LIU Y, et al. Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems[J]. IEEE Transactions on Industrial Informatics, 2017, 13(4): 1629 – 1640.

[10] XIE Y, SHENG Y H, QIU M Q, et al. An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling[J]. Engineering Applications of Artificial Intelligence, 2022, 112: 104879.

[11] RODRIGUEZ M A, BUYYA R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds[J]. IEEE Transactions on Cloud Computing,

2014, 2(2): 222 – 235.

[12] LIU L, ZHANG M, BUYYA R, et al. Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing [J]. Concurrency and Computation: Practice and Experience, 2017, 29(5): e3942.

[13] WANG Z J, ZHAN Z H, YU W J, et al. Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling[J]. IEEE Transactions on Cybernetics, 2020, 50(6): 2715 – 2729.

[14] WU Q W, FANG J Z, ZENG J, et al. Monte Carlo simulation-based robust workflow scheduling for spot instances in cloud environments [J]. Tsinghua Science and Technology, 2024, 29(1): 112 – 126.

[15] SHI L, ZHANG Z M, ROBERTAZZI T. Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(6): 1607 – 1620.

[16] XIE G Q, ZENG G, LIU Y, et al. Fast functional safety verification for distributed automotive applications during early design phase [J]. IEEE Transactions on Industrial Electronics, 2018, 65(5): 4378 – 4391.

[17] HU B, CAO Z C, ZHOU M C. Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems [J]. IEEE Transactions on Services Computing, 2022, 15(5): 2766 – 2779.

[18] LACHEHEUB M N, HAMEURLAIN N, MAAMRI R. Resources consumption analysis of business process services in cloud computing using Petri Net[J]. Journal of King Saud University: Computer and Information Sciences, 2020, 32(4): 408 – 418.

[19] CHEN Z G, ZHAN Z H, LIN Y, et al. Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach [J]. IEEE Transactions on Cybernetics, 2019, 49(8): 2912 – 2926.

[20] LI H F, WANG B Y, YUAN Y, et al. Scoring and dynamic hierarchy-based NSGA-Ⅱ for multiobjective workflow scheduling in the cloud [J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(2): 982 – 993.

[21] LI H F, WANG D J, ZHOU M C, et al. Multi-swarm co-evolution based hybrid intelligent optimization for bi-objective multi-workflow scheduling in the cloud [J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(9): 2183 – 2197.

[22] XUE F, HAI Q R, GONG Y L, et al. RVEA-based multi-objective workflow scheduling in cloud environments [J]. International Journal of Bio-Inspired Computation, 2022, 20(1): 49 – 57.

[23] LIU Z Z, WANG Y, HUANG P Q. AnD: a many-objective evolutionary algorithm with angle-based selection and shift-based density estimation [J]. Information Sciences, 2020, 509: 400 – 419.

[24] ZHOU X B, CAI X, ZHANG H, et al. Multi-strategy competitive-cooperative co-evolutionary algorithm and its application [J]. Information Sciences, 2023, 635: 328 – 344.

[25] WANG H D, JIAO L C, YAO X. Two_Arch2: an improved two-archive algorithm for many-objective optimization [J]. IEEE Transactions on Evolutionary Computation, 2015, 19(4): 524 – 541.

[26] LI B D, TANG K, LI J L, et al. Stochastic ranking algorithm for many-objective optimization based on multiple indicators[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(6): 924 – 938.

[27] XIE G Q, ZENG G, LI R F, et al. Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing [J]. IEEE Transactions on Sustainable Computing, 2017, 2(2): 62 – 75.

[28] CONVOLBO M W, CHOU J. Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources[J]. The Journal of Supercomputing, 2016, 72: 985 – 1012.

[29] PANICHELLA A. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization[C]// Proceedings of the Genetic and Evolutionary Computation Conference, 2019: 595 – 603.

[30] LI M Q, YANG S X, LIU X H. Pareto or non-Pareto: bi-criterion evolution in multiobjective optimization [J]. IEEE Transactions on Evolutionary Computation, 2016, 20(5): 645 – 665.

[31] DEB K, JAIN H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints [J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 577 – 601.

[32] ZITZLER E, LAUMANNS M, THIELE L. SPEA2: improving the strength Pareto evolutionary algorithm[R/OL]. [2023 – 12 – 25]. https://doi.org/10.3929/ethz-a-004284029.

[33] SUBRAMONEY D, NYIRENDA C N. Multi-swarm PSO algorithm for static workflow scheduling in cloud-fog environments[J]. IEEE Access, 2022, 10: 117199 – 117214.

[34] TIAN Y, ZHU W J, ZHANG X Y, et al. A practical tutorial on solving optimization problems via PlatEMO [J]. Neurocomputing, 2023, 518: 190 – 205.