研究与开发

改进的海洋生物捕食算法在网络成本优化上的应用

高明[1]，沈艺程[1]，刘铭[2]
（1. 浙江工商大学信息与电子工程学院，浙江 杭州 310018；
2. 浙江工业大学计算机科学与技术学院，浙江 杭州 310023）

摘　要：容器技术作为轻量级虚拟化方案，已成为多云网络架构的核心支撑技术，但跨云通信的高昂成本仍是关键挑战。创新点在于将多云网络资源调度问题建模为二次分配问题（QAP），并提出一种改进的离散海洋捕食算法（DMPA）。DMPA通过以下策略显著提升性能：基于均匀分布与伪反向学习的混合种群初始化策略，避免初始解陷入局部最优；引入自适应收敛因子动态调整搜索范围，平衡全局探索与局部开发；采用可变交换间隔的2-exchange突变策略，增强种群多样性；结合禁忌搜索优化精英解，避免重复搜索。实验表明，DMPA在30个QAP实例中22个达到已知最优解，平均偏差率低于3%；在多云网络真实数据集下，优化效果明显，通信成本有所降低。相比MPA、HPSO等算法，DMPA在优化精度与稳定性上均表现突出，为多云网络成本优化提供了高效解决方案。

关键词：多云网络；云原生；二次分配；群智能优化算法
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2025118

Application of improved discrete marine predatiors algorithm in network cost optimization

GAO Ming[1], SHEN Yicheng[1], LIU Ming[2]
1. College of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
2. School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China

Abstract: Container technology, as a lightweight virtualization solution, remains a core supporting technology for multi-cloud network architectures. However, the high cost of cross-cloud communication remains a critical challenge. The key innovation of models of the resource scheduling problem in multi-cloud networks as a quadratic assignment problem (QAP) and an improved discrete marine predators algorithm (DMPA) was proposed. The DMPA was signifi-

收稿日期：2024−12−21；**修回日期：**2025−03−13
通信作者：沈艺程，beardshen@hotmail.com
基金项目：国家自然科学基金资助项目（No.61871468）；浙江省新型网络标准与应用技术重点实验室基金资助项目（No. 2013E10012）；浙江省基础公益研究计划项目（No.LGG20F010015）
Foundation Items: The National Natural Science Foundation of China (No.61871468), Zhejiang Provincial Key Laboratory of New Network Standards and Applications (No. 2013E10012), Zhejiang Provincial Basic Public Welfare Research Program (No.LGG20F010015)

cantly enhanced performance through the following strategies: a hybrid population initialization strategy based on uniform distribution and pseudo-reverse learning, which avoided the initial solution from falling into local optima; an adaptive convergence factor was introduced to dynamically adjust the search range, balancing global exploration and local exploitation; a 2-exchange mutation strategy with variable exchange intervals was employed to enhance population diversity; the integration of tabu search to optimize elite solutions and avoid repetitive searches. Experiments show that DMPA achieves known optimal solutions in 22 out of 30 QAP instances, with an average deviation rate of less than 3%. Under real-world multi-cloud network datasets, the optimization effect is significant, and communication costs are reduced. Compared to algorithms such as MPA and HPSO, DMPA demonstrates outstanding performance in both optimization accuracy and stability, providing an efficient solution for cost optimization in multi-cloud networks.

Key words: multi-cloud network, cloud native, secondary distribution, swarm intelligence optimization algorithm

0　引言

近年来，随着云原生技术的大力发展，企业能更好地在公有云、私有云、混合云等新型动态云环境下构建和运行可弹性扩展的应用，极大提高了企业对于服务器资源的利用率[1]。

然而，仅依赖单一云平台会出现一些问题。首先，单云没有足够的资源为来自不同地区的所有互联网应用程序提供服务。其次，结合近3年国内几大云厂商发布的故障公告，平均几个月就会因为云安全问题链路无法进行数据传输，此时，单云模式因为其单一的架构布局在面对这些突发状况时就会展露其灵活性差的弊端[2]。但由故障公告可以发现，不存在某一特定时刻，让几大云服务厂商同时无法提供服务，因此，出于安全以及可用性的考虑，多云模式逐渐代替单云模式成为当前云服务环境下一大主流的部署方式[3]，但其带来的跨云通信成本也显著增加。当前，多数研究集中于利用群智能优化算法以及强化学习方法解决多云环境下的调度问题[4-5]，但这些方法都没能考虑高昂的网络通信开销，仅对计算资源进行了优化，并且这些方法在处理大规模的二次分配问题时，普遍存在计算复杂度高、稳定性差的问题。

为应对这一挑战，本文提出了一种基于Kubernetes的多云网络成本优化方法，旨在帮助云租户降低多云、多地区、多数据中心之间的网络资源经济成本，其主要由以下几部分组成。

（1）在Kubernetes环境下部署跨云平台和跨地域的服务，通过Kubeshark实时采集各服务之间的网络数据包，并解析包头信息以构建带权网络拓扑。

（2）将该带权网络拓扑转化为二次分配问题（quadratic assignment problem，QAP）。

（3）采用改进的海洋捕食算法（discrete marine predators algorithm，DMPA）计算每个时刻的资源特征，并根据计算结果生成一个最小化全局网络资源成本的资源调度方案。

（4）将该调度方案提交至Kubernetes调度器队列，在下一个时间周期结束时评估调度效果，并通过阶段性反馈优化模型。

1　相关工作

目前，多云网络在数据处理和业务部署中的应用日益广泛，多数商业公司已将业务迁移至多云环境。例如，某大型集装箱企业通过AWS、Azure和VMware的混合基础设施战略提高了系统可用性和存储效率；ChinaUnicom则从Docker容器化与VMware的组合转向使用Kubernetes管理其50个微服务及新开发项目等，显著提升了资源利用率，并降低了IT成本。然而，

该方法对传输高度依赖，存在复杂度高且弹性不足的问题。

就目前网络技术而言，已有研究致力于优化路由算法以减少网络环境中的传输成本。例如，文献[6]引入深度强化学习原理优化软件定义网络中的路由过程；文献[7]采用蚁群算法进行路由优化；文献[8]提出混合虚拟网络环境概念，利用遗传算法部署虚拟网络功能（virtual network function，VNF）以最小化带宽开销并最大化链路利用率；文献[9]通过复制VNF确保负载均衡，并为大型网络设计了遗传算法优化方案；文献[10]则提出了在虚拟环境下基于能源的VNF放置方法，从而最大限度地降低能耗并确保服务质量。这些工作主要关注局域网下的传输成本。相比之下，本文强调多云网络环境基于网络虚拟化技术构建，在不同环境下路由策略各异，使得租户难以更改基础设置。因此，需结合容器虚拟化技术和优化路由算法才能有效降低网络成本。

就容器虚拟化技术而言，许多学者对多云网络中计算资源节点的部署位置问题进行了研究。为降低网络资源成本，文献[11]提出了一种自适应虚拟机（virtual machine，VM）监控和实时迁移策略，用于边缘云架构中的物联网应用。文献[12]考虑了云数据中心环境下工作负载波动和应用程序复杂度，根据工作负载需求动态地为虚拟化系统提供资源以优化能源消耗。文献[13]提出了一种利用自主预测系统，通过动态调整计算资源的位置达到了优化网络资源成本与系统效率的效果。此类文献具备多云环境的应用条件。

本文基于多云环境研究，提出了一种通过调整计算资源位置而不改变网络资源拓扑的方法，以实现计算资源随时间和流量变化的动态调整，从而降低网络资源的经济成本。该方案在物理机器和虚拟机部署阶段的时间成本较高，但借助云原生技术，能够实现毫秒级的跨云跨地区服务调度，从而使该方案的实施成为可能。

2 系统架构

本模型基于Kubernetes云原生编排系统，采用Kubernetes中最小可调度单元（Pod）间的流量特征进行研究，旨在实现实时的云原生网络资源成本优化。通过计算近优解，模型能够动态调度计算资源，以最小化网络资源开销。

网络资源成本优化的核心是网络数据，首先需要对这些数据进行采集和结构化处理。系统使用Kubeshark指标采集器从内核态获取业务服务与网络服务间的网络数据包，并将其输出至Cilium网络功能服务。随后，Fluent Bit数据转发器将处理后的数据写入Kafka消息队列，通过Logstash实现实时数据传输，并将清洗后的关键元数据存储至分布式时间序列检索引擎（elastic-search，ES）。最终，数据通过Kibana进行查询与分析，并在处理完成后存入PostgreSQL数据库。

经过处理的核心网络数据指标包括各服务间的连接关系和连接频率。这些指标可用于优化计算和网络资源的调度。例如，时效性要求较低且数据量较小的服务可以定期运行并在任务完成后销毁，从而有效节省计算资源。对于通信频率较高的服务，通过减少它们之间的物理距离和提高亲和性，可以降低网络资源成本。

模型的数据流图如图1所示。在多个云服务提供商的不同地区和数据中心内，操作系统内核态采集网络数据包，并在数据特征清洗后，使用DMPA进行计算。通过多次迭代，模型能够获得近优解，并据此生成新的计算资源放置方案。每个计算周期（定义为5 min）进行一次计算，以平衡调度频率与流量变化反馈之间的关系。最终，最佳解所确定的计算资源位置会被调度器应

用到Kubernetes的计算资源调度系统中，确保资源的位置接近期望状态。

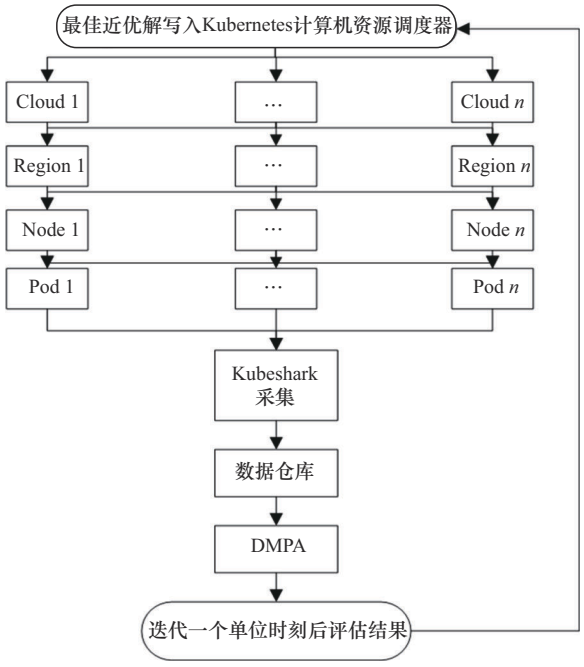


图 1　模型的数据流图

3　模型构建

在云原生环境中，成本主要包括计算资源成本和网络资源成本。本文重点在固定计算资源成本的基础上，针对网络资源成本进行分时优化。优化过程中，将Pod所在计算资源池的子集，计算资源插槽（Slot）之间的网络流量价格作为Pod之间的流量成本权重（流量价格参照2022年云服务提供厂商发布的价格，单位：元/GB），规定每个Slot最大仅可容纳1个Pod，并且每个Slot具备流量成本权重表中的位置属性。流量成本权重差异详见表1。将多云网络分层为云（Cloud）、区（Region），由Slot所在的不同位置而导致的流量成本权重差异可参照表1，两个不同Slot_m与Slot_n若在相同Cloud和Region下，则通信费用为0；若在相同Cloud、不同Region以及其他情况下，成本R_{mn}可由式（1）得到，其中$\boldsymbol{L}$=[Cloud、Region]。

$$R_{mn}=P\times\sum\nolimits_1^2 L_i^m\oplus L_i^n \tag{1}$$

表 1　流量成本权重差异

序号	Cloud	Region	成本/元
1	相同	相同	0
2	相同	不同	R_{mn}
3	不同	不同	R_{mn}
4	不同	相同	R_{mn}

其中，P为源云厂商下某地区到目的云厂商某地区发送一个数据包的单价。令$b_{s_1}^{s_2}(t)$是t时刻从计算插槽$s_1\in S$传输1 GB数据到另一个计算插槽$s_2\in S$的价格，$\lambda_{d_1}^{d_2}(t)$是在t时刻，从Pod_{d_1}传输到Pod_{d_2}的数据量大小。假设当前网络环境下有n个Pod与Slot，可定义一个大小为$n\times n$的矩阵$B=\left(b_{s_i}^{s_j}\right)$，$A=\left(\lambda_{d_i}^{d_j}\right)(1\leqslant i,j\leqslant n)$用来表示不同计算插槽之间的通信单价与不同Pod之间传输的流量大小。因此，则t时刻带宽的总费用见计算式（2）。

$$C_b(t)=\sum_{i\in n}\sum_{j\in n}\lambda_{d_i}^{d_j}(t)b_{s_i}^{s_j}(t) \tag{2}$$

基于以上各费用成本的定义，结合所研究的目标，找到一组满足约束的流量分配方案X，如式（3）所示，X为Slot的分配方案。

$$X:\{1,2,\cdots,n\}\to\{1,2,\cdots,n\},X\in\Pi(n) \tag{3}$$

其中，n为Pod和Slot的数量，为1到n之间所有整数排列组合的集合，即表示将Pod与Slot逐一形成绑定关系。使其在时间集合T内的通信总成本尽可能小，如式（4）所示。

$$\text{Cost}=\min_X\left(C(t)\right) \tag{4}$$

约束条件如下。

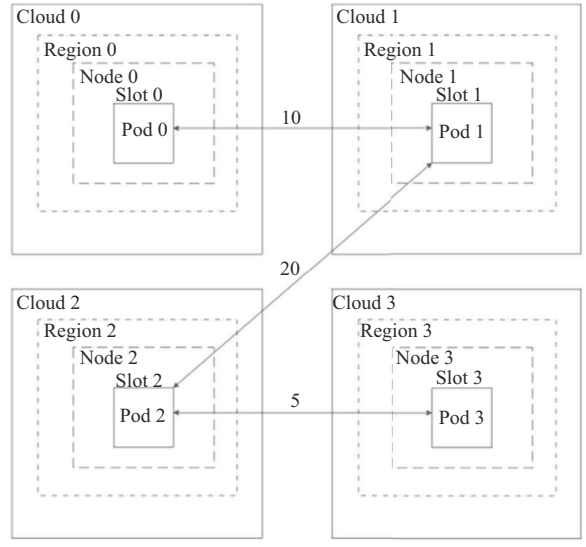
$$\text{s.t.}\forall t\in T,1\leqslant i\leqslant D,1\leqslant j\leqslant S,\lambda_i^t=0,X_{i,j}^t=0 \tag{5}$$

$$\forall t\in T,1\leqslant i\leqslant D,\lambda_i^t>0 \tag{6}$$

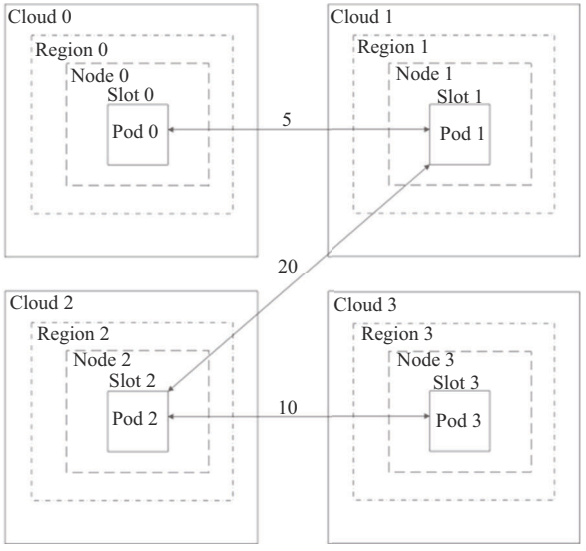
$$\sum\nolimits_{j=1}^{S}X_{ij}^t=1 \tag{7}$$

由于云租户的费用计算是基于实际使用的流量大小，显然，将数据传输量较大的两个Pod安排在流量费用较低的两个Slot中，可以有效降低通信成本。Pod位置交换示意图如图2所示，图2

（a）为初始状态下云租户的分布情况，根据图示，Pod_0 与 Pod_1 之间通信的数据量为 10 GB，Pod_1 与 Pod_2 为 20 GB，Pod_2 与 Pod_3 为 5 GB，Slot 之间的通信费用参考见表 2。将实际流量传输情况与不同云之间的单价抽象为矩阵 $A^{(1)}$、$B^{(1)}$。因此，在初始状态下，由通信带来的开销可通过式（4）计算得出，总费用为 18 元。默认分配方案 X=[0,1,2,3]，经过算法优化后，新的流量分配分案为 X=[3,2,0,1]，调整为如图 2（b）所示，可抽象为矩阵 $\boldsymbol{A}^{(2)}$。



（a）初始Pod位置



（b）调整后Pod位置

图 2　Pod 位置交换示意图

表 2　通信费用参考

对比项	Slot0	Slot1	Slot2	Slot3
Slot0	0	0.2	0.1	0.3
Slot1	0.2	0	0.7	0.2
Slot2	0.1	0.7	0	0.4
Slot3	0.3	0.2	0.4	0

$$A^{(1)}=\begin{bmatrix}0&10&0&0\\10&0&20&0\\0&20&0&5\\0&0&5&0\end{bmatrix}$$

$$B^{(1)}=\begin{bmatrix}0&0.2&0.1&0.3\\0.2&0&0.7&0.2\\0.1&0.7&0&0.4\\0.3&0.2&0.4&0\end{bmatrix}$$

$$A^{(2)}=\begin{bmatrix}0&5&0&20\\5&0&0&0\\0&0&0&10\\20&0&10&0\end{bmatrix}\tag{8}$$

调整矩阵行列计算费用如图 3 所示。费用减少至 11 元，有效地降低了网络成本。

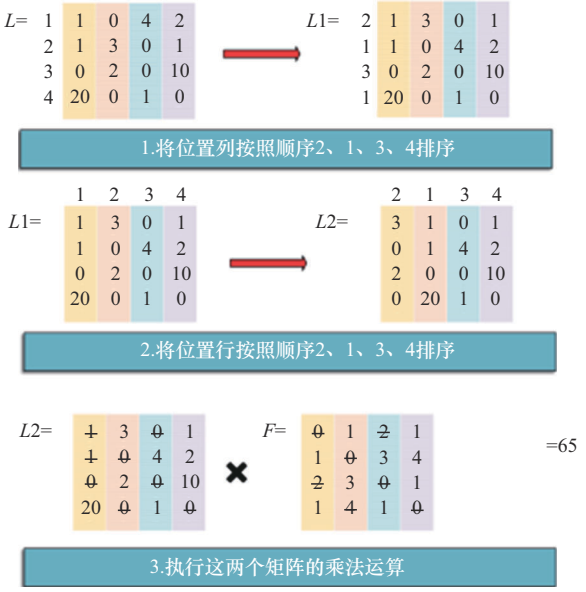


图 3　调整矩阵行列计算费用

4　海洋捕食算法的改进策略

在本研究中，选择海洋捕食算法（marine predators algorithm，MPA）[11]来解决设施布局问

题（QAP），是因为该算法在应对复杂优化问题时展现了优异的性能。MPA是一种基于自然启发式的算法，模拟了海洋捕食者觅食行为，能够高效地探索解空间，避免陷入局部最优解。与其他优化算法相比，MPA具备较强的全局搜索能力和较快的收敛速度，尤其适合解决大规模、高维度的组合优化问题，如QAP等。此外，MPA的灵活性使其能够根据问题需求自适应调整搜索策略，从而提高解的精度和稳定性。在本研究中，捕食者和猎物的位置向量被用来表示调整方案X，进而通过生成近优解来指导Pod的重新部署，从而实现降低网络资源开销的目标。

4.1　方法的表示与解码

许多现有的元启发式算法用于求解QAP还存在不足，其改进的关键在于使Pod在多云网络下的位置与海洋捕食者的位置之间建立合理的映射关系。

本文选择路径表示来编码QAP的解[12]，种群中的每个海洋生物代表一个可能的QAP解，向量$\boldsymbol{X}$表示捕食者的位置，其中每个元素$x_{i,j}$对应需要交换的行或列的下标。该表示法有助于DMPA通过随机游走和局部搜索等方法有效解决QAP问题。在更新海洋生物位置时，可能出现小数或重复值导致的不可行解。因此，本文提出了一种基于顺序的排列方法来解决这一问题。具体来说，首先对更新后的解进行排序，解码后的解通过对应元素的序列号表示。解决方案的解码、编码过程如图4所示。

初始样本	1	2	3	4	5	6
位置调整	1.1	1.8	2.1	2.4	0.8	1.4
标准化	5	3	2	1	6	4

图4　解决方案编码、解码过程

4.2　种群初始化

初始的海洋捕食算法使用随机生成法来初始化种群个体。然而，在求解大规模优化问题时，如果初始种群中的个体集中在某一局部区域或远离最优解的区域，该算法往往容易陷入局部最优解。为了解决这一问题，本文结合均匀分布和伪反向学习策略对种群进行初始化，具体方法如下。

假设测试函数的可行域中每个维度的上界为ub，每个维度的下界为lb，根据均匀分布的初始方法，群体中第i个个体的位置可表示为$X_i=\left(X_i^1,X_i^2,\cdots,X_i^d\right)$，$i$=1,2,⋯,$N$，其中$d$表示问题的维度，$X_i^j$的计算式如下：

$$X_i^j=\left(\mathrm{RNUM}_i-1\right)\times\frac{\left(\mathrm{ub}-\mathrm{lb}\right)}{N}+\mathrm{rand}\left(0,\frac{\mathrm{ub}-\mathrm{lb}}{N}\right)\tag{9}$$

$$\mathrm{RNUM}=\mathrm{randperm}(\mathrm{NUM})\tag{10}$$

其中，RNUM表示数组NUM经过元素顺序随机变化后形成的新数组，$\mathrm{NUM}=\left[\mathrm{NUM}_1,\mathrm{NUM}_2,\cdots,\mathrm{NUM}_N\right]$表示从1到种群中个体总数$N$的整数数组，其中$\mathrm{NUM}_i=i$，randpern(NUM)表示数组NUM中元素顺序的随机变化，对其重新组合排列。再使用伪反向学习策略的计算式对种群中第i个个体$\overline{X_i}=\left(\overline{X_i}^1,\overline{X_i}^2,\cdots,\overline{X_i}^d\right)$，$i$=1,2,⋯,$N$的更新策略如下：

$$\tilde{X}_i^j=\mathrm{ub}+\mathrm{lb}-X_i^j,j=1,2,\cdots,d\tag{11}$$

$$\overline{X}_i=\begin{cases}\mathrm{rand}(M_j,\tilde{X}_i^j),X_i^j\leqslant M_j\\ \mathrm{rand}(\tilde{X}_i^j,M_j),X_i^j>M_j\end{cases}\tag{12}$$

其中，$M_j=0.5(\mathrm{ub}-\mathrm{lb})$，更好地进行边界划分。

4.3　收敛因子改进

猎物的移动在很大程度上依赖于参数P，但固定的参数可能导致猎物无法根据不同阶段及时调整游动速度。在迭代初期，活动范围较大，本次研究希望猎物能探索更多区域，从而增强算法的全局搜索能力。随着迭代进展，捕食者出现后，活动范围逐渐缩小，研究期望猎物能够探索之前未涉足的区域，提高局部搜索能力并加速收敛。因此，本文改进了固定参数P，引入了自适应收敛因子，如计算式（12）所示。

$$P=\left(\cos\left(\Pi\times\left(t/T_{\max}\right)\right)+1\right)\times0.5\tag{13}$$

在迭代初期，乘以0.5可以使收敛因子下降得更为缓慢，从而保持较宽的搜索范围，有利于全局搜索，避免过早陷入局部最优，随着迭代进行，收敛因子逐渐变小，此时乘以0.5有助于加速局部搜索和精细化调整，使算法在后期能够迅速收敛到更优解。该系数经过大量实验验证，能更好地平衡全局搜索与局部开发的需求，从而提升整体优化效果。

4.4 漩涡形成和FAD效应

在MPA中，鱼类聚集效应（fish aggregating devices，FAD）代表了算法可能陷入的局部最优解。FAD在MPA算法中的影响如式（13）所示，在解决离散优化问题上的效果不佳，易陷入局部最优，从而降低收敛速度。为了提高种群多样性并平衡多样化和优化效果，本文采用2-exchange突变替代原算法的更新策略。

$$\overrightarrow{\text{Prey}_i}\begin{cases}\overrightarrow{\text{prey}_i}+\text{CF}\left[\vec{X}_{\min}+\vec{R}\otimes\left(\vec{X}_{\max}-\vec{X}_{\min}\right)\right]\otimes\vec{U}, & r\leqslant\text{FADS}\\ \overrightarrow{\text{prey}_i}+\left[\text{FADS}(1-r)+r\right]\left(\overrightarrow{\text{prey}_{r1}}-\overrightarrow{\text{prey}_{r2}}\right), & r>\text{FADS}\end{cases} \tag{14}$$

为了避免较高的时间复杂度，突变操作并非每次迭代都执行，而是通过加入附加条件来判断是否陷入局部最优。当最佳适应值长时间未更新时，才会执行突变操作。具体实现如下：首先设置判定界I，作为触发突变的第一个条件；然后设置参数count，用于判断最佳适应度值保持不变的次数。使用iter表示当前迭代次数。当iter<I时不进行突变；当iter>I且当前最佳适应度值与第（iter-count）次迭代的最佳适应度值相同，则判定陷入局部最优，执行突变。本次研究中，I的值在[1/5Max_iter, 4/5Max_iter]区间。

2-exchange突变介绍如下。

为了增强种群多样性，在算法中重复执行2-exchange突变，直到达到最大重复次数（突变上界）。除了增加多样性外，突变过程还需要保持良好解的特征，因为这些解的某些区域可能与最优解相符。因此，算法仅从变异后的个体中选择适应度最佳者用于后续迭代更新。交换间隔=3时的2-exchange突变过程如图5所示。

固定交换间隔无法有效平衡不同迭代阶段的“游动速度”，为此，本文提出了可变交换间隔的改进方法。初始时，交换间隔μ设为最小值，然后逐步增加，直到达到预设上界$\mu_{\max}$。此后，μ将自动重置为$\mu_{\min}$，并继续迭代，直到达到突变上界。其中$\mu_{\min}=\lfloor 0.2n\rfloor$，$\mu_{\max}=\lfloor 0.5n\rfloor$（$n$表示待解决问题的维度）。为了平衡全局搜索与局部搜索，当迭代次数达到5/6×Max_iter后，2-exchange突变的范围将缩小至原解决方案的一半，前后两部分各自进行突变操作。

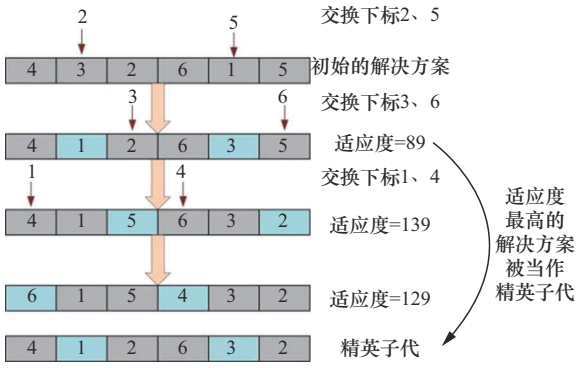


图5 交换间隔=3时的2-exchange突变过程

4.5 禁忌搜索

在DMPA中，禁忌搜索只考虑精英矩阵中的最优解作为候选解。搜索过程从初始解开始，通过随机交换下标值，迭代到具有最佳局部目标函数值的相邻解p_0。为避免重复局部最优解，引入禁忌列表来存储最近m次迭代中的禁忌移动，从而防止无效的重复操作。然而，随着迭代的进行，之前被禁忌的解可能会变得更优，因此，本文在禁忌搜索中加入了开放规则，在特定情况下允许解除禁忌状态。这些条件包括：

- 禁忌移动产生的新解比当前最佳解更优；
- 禁忌“激励”值小于预定义上限。

由于每次交换运算的计算成本较高，本文使用领域搜索代理来计算交换后得到的新解。假设当前排列为Π，经过一次随机交换操作后得到新排列Π^*。由于实验假设矩阵是对称的，则本次排列Π中两个位置(j,k)的移动的差成本计算如下：

$$\Delta_{\text{cost}}(\Pi,i,j)=\sum_{p=1,p\neq i,j}^{n}\left(f_{ip}-f_{jp}\right)\left(d_{\Pi(j)\Pi(p)}-d_{\Pi(i)\Pi(p)}\right) \tag{15}$$

$$\Delta_{\text{cost}}\left(\Pi^*,i,j\right)=\Delta_{\text{cost}}\left(\Pi,i,j\right)+\left(f_{iu}-f_{iv}+f_{jv}-f_{ju}\right)\left(d_{\Pi(i)\Pi(u)}-d_{\Pi(i)\Pi(v)}+d_{\Pi(j)\Pi(v)}-d_{\Pi(j)\Pi(u)}\right) \tag{16}$$

为了更好地搜索解空间，禁忌列表的大小应适中。若列表太小，可能导致搜索循环；若列表过大，则降低了找到最优解的概率。Taillard建议随机调整禁忌列表的大小[13]，据此将禁忌搜索列表大小的范围设置为$\left[\lfloor 0.1n\rfloor,\lfloor 0.9n\rfloor\right]$。详细可见算法1，具体来说，算法1主要时间开销为两层嵌套循环，单次迭代复杂度为$O(n^2)$，总复杂度为$O(\text{Max_Iter}\times n^2)$。通过限制禁忌列表长度和激励阈值，实际运行时间显著降低；空间开销上存储禁忌列表和差值矩阵需$O(n^2)$，与问题规模呈平方关系。

算法1　改进后的禁忌搜索

输入　p=当前解决方案，Max_Iter=最大迭代次数，h_{low}=最小禁忌列表长度，h_{high}=最大禁忌列表长度，Aspiration=激励前的迭代次数

输出　p^*=最佳解决方案

初始化Tabu_List、Delta为0矩阵

$\Pi^*=\Pi$

For i=1 to $n-1$

　For $j=i+1$ to n

　　Delta[i][j]=使用式（14）对当前解决方案的优劣进行计算

　End For

End For

For m=1 to Max_Iter

　h=Random(h_{low},h_{high})

　Delta_min=In f，u=In f，v=In f

　For i=1 to $n-1$

　　For j=1 to n

　　Tabu1=Tabu_List[i]　[p[j]]; tabu2=Tabu_List[j][p[i]]

　　Tabu=(Tabu1≥m) and (tabu2≥m)

　　Aspired= (Tabu1<m−Aspiration and Tabu2<m−Aspiration) or (z(p)+Delta[i][j])<$z(p^*)$ and Tabu=1)

　　If (Delta[i][j]<Delta_min and Tabu ≠ 1) or (aspired =1)

　　$u=i$，$v=j$，Delta_min=Delta[i][j]

　　End If

　End For

End For

If $u\leq n$

　P’=根据u和v交换p中对应下标

　For i =1 to $n-1$

　　For $j=i+1$ to n

　　Delta[i][j]=根据式（15）计算当前解决方案p’的优劣

　　End For

　End For

　Tabu_List[u][p[v]]=$m+h$，Tabu_List[v][p[u]]=$m+h$

　$p=p'$

　If $z(p)<z(p^*)$

　$p^*=p$

　End If

End If

End For

返回p^*

5 DMPA的性能测试和分析

本节旨在展示所提出算法的有效性和可行性。用于进行实验的配置是Windows 11 64位操作系统；8 GB内存；处理器英特尔酷睿i5-8250U CPU，1.60 GHz处理器处理速度。该算法在MATLAB 2018a仿真软件上实现。所提出的算法DMPA在9个不同的实例上运行，这些实例由30个测试问题组成，主要包括Bur、Chr、Lipa、Nug、Rou、Scr、Ste、Tai、Tho。测试问题的规模在12~150，用以模拟不同规模的集群。数据集可在线获取[14]，为了找到参数值的最佳组合，进行了多次实验，以找到优化效果最优的参数组合。DMPA最终的参数设置见表3。DMPA算法在QAP实例上得到的仿真结果见表4，显示了DMPA用于求解QAP时的数值结果，规模大小从12到150。为了获得可靠的数据，每个实例进行了30次独立的运行。表4第二列展示实例的名称以及QAP的规模，如Bur26a，它表示QAP的规模为26×26。列“最优”“最佳”“平均值”分别表示报告中的最佳值、DMPA发现的最佳值和平均值。列“PDB(%)”和“PDA(%)”分别表示30次运行中发现的最佳值与最优解的百分比偏差，30次运行时发现的平均值与最优解的百分比偏差。它们的计算式可参照式（16）~式（17），最后一列“时间”是30次运行的平均时间。表4中粗体表示在当前实例中表现较为突出的数值，可以发现，改进后的DMPA算法在处理QAP方面具有良好的性能。总体而言，DMPA在22/30测试案例中找到了最佳解决方案。在86.67%的实例中，PDB在1%以内。就平均偏差而言，除Chr25a、Tai80b、Tho150外，所有实例的PDA都在3%以内。这表明，随着问题规模的增加，DMPA仍然可以保持解决方案稳定性。在“时间”列中可以发现，所有的执行都在可接受范围内。

表3 DMPA参数设置

参数	值
种群大小	20
迭代次数	2 000
FAD	0.35

5.1 DMPA的优化效果

本节展示了将高斯变异扰动和短步长扰动与MPA算法相结合的效果。3种MPA变体中6个不同实例的性能对比见表5，表5中粗体表示当前实例下各算法获得最优解的数据。MPA的3种变体如下：

（1）标准MPA算法（MPA）；

（2）将MPA算法与重复2-exchange突变相结合（GMPA）；

（3）将MPA算法与多种改进策略相结合（DMPA）。

从表5可以发现，DMPA算法在所有测试问题上都能达到最优值，且在其中两个测试用例上要优于GMPA算法。此外，GMPA在6个测试问题上优于MPA。因此，将高斯变异扰动和短步长扰动相结合，改进了MPA算法的寻优结果。在融入禁忌搜索后，进一步改善了算法的寻优效率与范围。DMPA性能对比箱线图如图6所示。箱线图绘制了20次独立运行下MPA、GMPA和DMPA这3种算法结果的差异。从图6中可以发现，DMPA算法相较于MPA和GMPA拥有更低平均值与中位值，优化效果明显。

$$\mathrm{PDB}(\%)=\frac{(\mathrm{Best}-\mathrm{Optimal})}{\mathrm{BKS}}\times 100\% \quad (17)$$

$$\mathrm{PDA}(\%)=\frac{(\mathrm{Average}-\mathrm{Optimal})}{\mathrm{BKS}}\times 100\% \quad (18)$$

其中，BKS表示该实例的理论最优值。

5.2 先进算法对比

为了证明DMPA算法的可行性和性能表现，本文选择了文献中证实的可行算法以及经过离散化

表 4　DMPA 算法在 QAP 实例上得到的仿真结果

编号	样例	已知最佳值	最佳值	平均值	平均值偏差	最优值偏差	时间/s
1	Bur26a	5 426 670	**5 426 670**	**5 426 670**	0.00	0.00	2.43
2	Bur26b	3 817 852	**3 817 852**	**3 817 852**	0.00	0.00	4.30
3	Bur26c	5 426 795	**5 426 795**	**5 426 795**	0.00	0.00	5.50
4	Chr12a	9 552	**9 552**	**9 552**	0.00	0.00	3.47
5	Chr12b	9 742	**9 742**	**9 742**	0.00	0.00	3.38
6	Chr15a	9 896	**9 896**	9 916	0.00	0.20%	4.07
7	Chr15b	7 990	**7 990**	8 035.8	0.00	0.57%	3.96
8	Chr20a	2 192	**2 192**	2 223	0.00	1.415	5.41
9	Chr22a	6 156	6 198	6 273.83	0.68%	1.91%	12.72
10	Chr22b	6 194	**6 194**	6 331.2	0.00	2.21%	13.58
11	Chr25a	3 796	**3 796**	4 237.33	0.00	11.63%	15.56
12	Lipa20a	3 683	**3 683**	**3 683**	0.00	0.00	8.41
13	Lipa20b	27 076	**27 076**	**27 076**	0.00	0.00	13.46
14	Lipa40a	31 538	**31 538**	**31 538**	0.00	0.00	50.13
15	Lipa50a	62 093	**62 093**	**62 093**	0.00	0.00	110.35
16	Lipa60a	107 218	**107 218**	107 547	0.00	0.31%	164.43
17	Lipa70a	169 755	**169 755**	171 673.7	0.00	1.13%	266.66
18	Lipa80a	253 195	254 782	259 401.8	0.63%	2.45%	500.44
19	Lipa90a	360 630	362 743	362 887.4	0.59%	0.63%	700.53
20	Nug16a	1 610	**1 610**	1 613.3	0.00	0.20%	8.11
21	Nug30a	6 124	6 219	6 258.2	0.08%	2.19%	54.53
22	Rou20a	725 532	**725 532**	**725 532**	0.00	0.00	7.5
23	Scr12a	31 410	**31 410**	**31 410**	0.00	0.00	3.23
24	Ste36b	15 852	**15 852**	16 026	0.00	1.09%	45.07
25	Tai12a	224 416	**224 416**	**224 416**	0.00	0.00	6.20
26	Tai50a	4 938 796	5 026 573	5 030 625.8	1.78%	1.86%	207.61
27	Tai80b	818 415 043	828 464 121	843 284 285.3	1.23%	3.04%	908.30
28	Tai100a	21 044 752	21 556 849	21 613 648	2.43%	2.71%	1209.10
29	Tho30a	149 936	**149 936**	150 124.25	0.00	0.13%	20.03
30	Tho150	8 133 398	8 327 176	8 412 618.66	2.38%	3.43%	990.38

处理后较为新颖的群智能优化算法作为其对比算法。它们包括 MPA 的改进算法离散化海洋捕食算法（nonlinear marine predator algorithm，NMPA）[15]，以及性能表现突出的算法混合粒子群算法（hybrid particle swarm optimization，HPSO）[16]、优化蚁狮算法（improved antlion optimization，IALO）[17]、最后是经过离散化处理后的蜣螂优化算法（discrete dung beetle optimizer，DDBO）[18]。

表 5　3 种 MPA 变体在 6 个基准实例上的性能对比

测试问题	最优值	MPA	GMPA	DMPA
Chr12b	9 742	10 102	**9 742**	**9 742**
Els19	17 212 548	17 294 834	**17 212 548**	**17 212 548**
Esc32a	130	206	140	**130**
Had20	6 922	6 936	**6 922**	**6 922**
Wil100	273 038	285 110	282 472	**273 038**
Tai64c	1 855 928	1 857 646	**1 855 928**	**1 855 928**

为了公平比较，所有算法的公共参数设置相同：独立运行 20 次，最大迭代次数为 500 次。

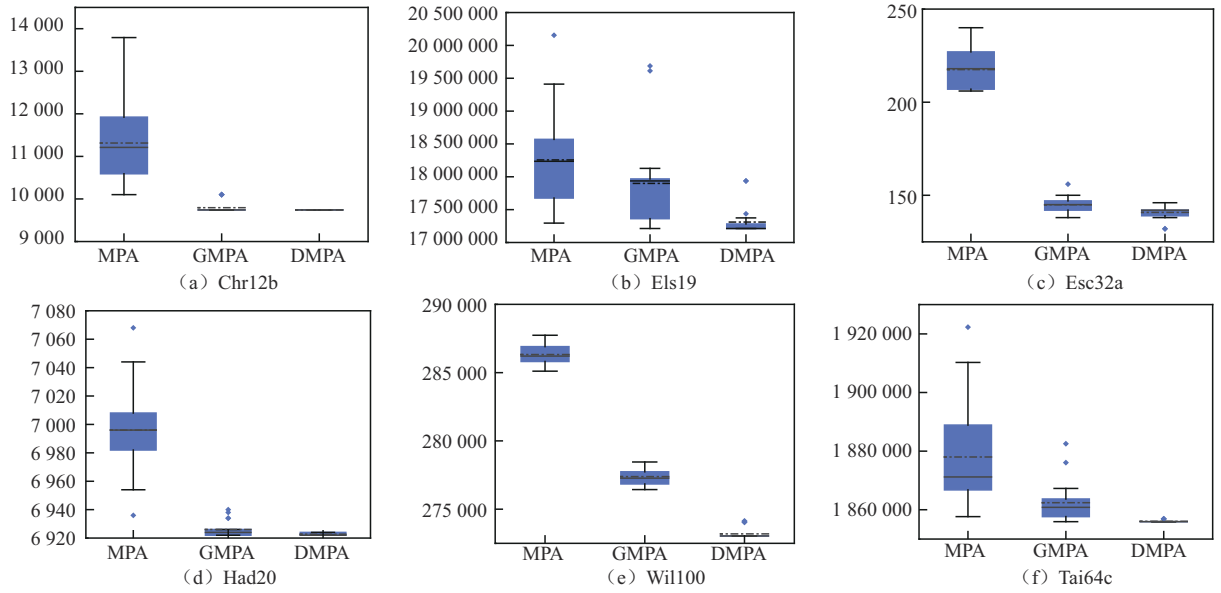


图 6　DMPA 性能对比箱线图

由于不同的参数设置会对算法的表现结果产生影响，因此对比算法的参数设置是根据其对应参考文献所选择的参数进行设置。NMPA、HPSO、IALO、DDBD、DMPA 算法的对比结果见表 6。表 6 结果显示，在多组测试实例中，DMPA 算法无论在最优、平均还是最差解上均表

表 6　NMPA、HPSO、IALO、DDBD、DMPA 算法的对比结果

实例	IALO			HPSO			DDBO		
	Best（最优值）	Average（最优解）	Worst（最劣值）	Best（最优值）	Average（最优解）	Worst（最劣值）	Best（最优值）	Average（最优解）	Worst（最劣值）
Chr12a	11 952	14 423.8	18 782	9 552	10 377	11 838	10 852	12 311.3	14 134
Chr20a	3 690	5 102.3	5 718	2 490	3 002.1	3 406	2 816	4 014.4	5 152
Esc32a	252	279.1	290	148	162.3	182	200	245.7	284
Lipa50a	63 584	63 617.9	63 646	62 984	63 067.9	63 166	63 416	635 871.5	63 654
Lipa90a	366 339	366 420	366 497	364 292	364 600	364 820	366 270	366 410	366 555
Nug30a	7 120	7 246.6	7 332	6 302	6 409.5	6 512	6 512	6 699.8	6 858
Ste36b	38 832	43 358.2	46 482	17 672	19 844.3	22 214	23 840	28 360.2	36 878
Tai80b	1 101 315 992	1 113 841 782	1 124 617 399	874 370 915	9 028 966 689	930 620 026	953 313 822	991 223 617	1 039 135 397

实例	NMPA			DMPA		
	Best（最优值）	Average（平均值）	Worst（最劣值）	Best（最优值）	Average（平均值）	Worst（最劣值）
Chr12a	9 988	11 641.3	13 076	9 552	9 552	9 552
Chr20a	3 170	3 911.4	4 702	2 196	2 201	2 223
Esc32a	194	239.4	274	130	132.7	140
Lipa50a	63 482	63 563.8	63 622	62 093	62 565	62 875.7
Lipa90a	366 086	366 354.5	36 6496	363 654	363 878.3	364 432
Nug30a	6 514	6 767.9	6 982	6 124	6 152	6 170
Ste36b	22 960	28 562.3	33 328	15 852	15 947.7	16 046
Tai80b	970 740 883	102 404 035	106 740 136	840 017 681	853 051 324.65	869 819 256

现出较高的优化精度和稳定性。例如，在Chr12a、Chr20a和Esc32a等实例中，DMPA均取得最低值，而在高维问题如Lipa50a和Lipa90a中，其结果波动也远小于对比算法。这些数据充分验证了改进策略的有效性，表明通过结合自适应收敛因子、可变交换间隔、禁忌搜索机制，DMPA能够在保证全局搜索能力的同时实现快速且稳定的局部收敛，从而显著降低通信成本。

各算法的平均误差比较如图7所示，DMPA算法具有最低的平均误差，值为0.72。AE_{avg}的定义可参考计算式（19）。

$$AE_{avg}=\frac{1}{n}\times\sum_{i=0}^{n}\frac{y-opt}{opt}\times 100 \qquad (19)$$

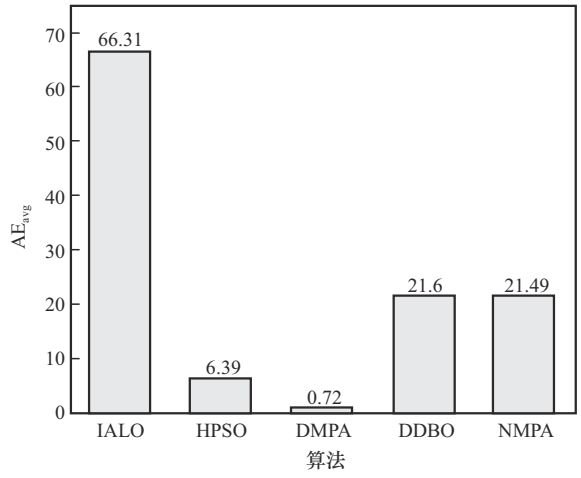


图 7　各算法的平均误差比较

其中，opt表示算法迭代出来的最优解值。

结果表明本文所提出的改进机制可以显著提高MPA算法的性能，能在大多数测试用例上找到最优解。

5.3　在网络成本优化模型中的应用

为验证本文方法在多云网络成本优化中的寻优质量和稳定性，本节利用真实数据集进行评估。样本数据来源于某互联网平台生产环境2022年1月22日全天的数据。为了满足实际需求，采集的时间间隔为5 min。每个子样本牵涉3家云服务提供商、29个地区、3个分区、87个节点，以及174个Pod对象。实验环境硬件设备信息见表7。

表 7　实验环境硬件设备信息

名称	类型	型号	操作系统
S0	华三交换机	S1205	Switch
H0	树莓派主板	PI4B	Ubuntu
H1	树莓派主板	PI4B	Ubuntu
H2	树莓派主板	PI4B	Ubuntu
H3	树莓派主板	PI4B	Ubuntu

基于OpenEuler Linux操作系统部署了Ubuntu22.04系统，并在该编排系统内部署实验环境软件，实验环境软件信息见表8。

表 8　实验环境软件信息

名称	版本	类型
Ubuntu	22.04 LTS	操作系统
Kubernetes	V1.23.1	资源编排系统
Kubershark	V41.6	流量监测应用
Fluent bit	V1.9.0	数据转发器
Elasticsearch	V8.0.0	分布式存储
Kibana	V8.0.0	分布式检索
PostgreSQL	V12.12.0	结构型数据库

实验采样时刻0、60、110、160、210的样本数据，使用DMPA对样本迭代500次，并选取近3年发表论文中的4种算法进行对比，包含1种MPA改进算法和4种非MPA类型的群智能算法。取10运行结果的最优值确定最终结果。DMPA与5种非MPA改进算法蜻蜓优化算法（cuckoo search algorithm，CSA）[20]、IALO、HPSO、DDBO、HACO[19]的迭代次数—成本优化效果进行比较，分时刻成本优化效果对比如图8所示，可以看出，本文提出的DMPA算法在500次迭代过程中，在数据集下随机挑选的6个时刻中，都表现得十分出色，平均优化效果达到了99.34%，以时刻0为例，DMPA的优化效果达到了99.38%，相较于HACO提高了7.72%，HPSO提高了8.74%。就执行时间而言，DDBO的执行

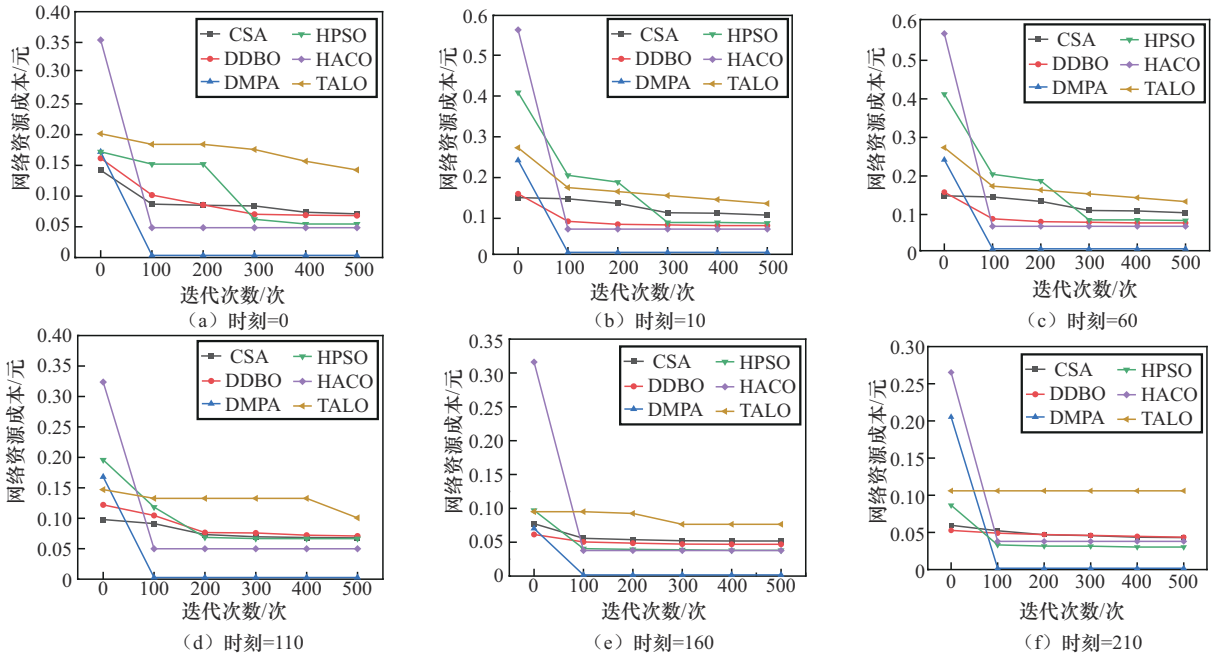


图 8　分时刻成本优化效果对比

时间最短，仅花费 15.8 s，相较于 DMPA 节省了 28.7 s，但优化效果却相差了 8.88%。面对大型多云网络环境时，DMPA 的执行时间在合理的范围内。因此，DMPA 算法在优化方案的质量和执行时间上都取得了不错的效果。

除了拥有较好的寻优能力之外，DMPA 算法还展现出优秀的稳定性。在第 150 次迭代时求得的各项参数指标见表 9，表中给出了迭代次数为 150 时表现较好的前 3 个算法以及初始 MPA 算法在 10 次运行时求得网络成本的最差值、最优值、平均值、标准差。选择第 150 次迭代时的数据做稳定性对比是因为在数据集中，迭代轮次为 150 时，网络成本优化效果对应的折线形态变化均最显著，且与第 500 次迭代所得出的数据相比波动较小。从表 9 中的数据可以看出，与其他 4 种算法相比，DMPA 在数据集上表现出了更出色的优化性能，而其他类比算法则相对较差。此外，DMPA 还展现出更高的优化稳定性，标准差明显低于其他类比算法，增强了其在不同运行中的性能一致性。这对于各种实际应用都具有重要意义。

最后，为验证算法的持续优化效果，在参数固定的情况下，利用数据集进行了实验，各时刻下网络资源成本变化如图 9 所示，显示了各个时刻下系统总花费的变化情况，总花费随着当前时刻网络环境下传输的数据量大小而变化。

为了防止偶然性对于结果的影响，本次研究进行了 2 次实验，取最好优化效果作为实验的结果。经过 DMPA 算法的优化调整后，费用曲线有了显著下降，相比初始成本降低了 97.0%，时间上花费 168 min，平均每个时刻花费 0.58 min，满足多云网络环境下位置拓扑变化的时间要求。原始的 MPA 算法对于初始成本的优化效果为 88.93%，时间上花费 94.5 min，平均每个时刻花费 0.3 min，相比于 DMPA 算法在时间上更具优势，平均每个时刻节省 0.2 min，但是优化效果相差 8.07%。经过比对分析，DMPA 在大型的多云网络拓扑环境下更具优势，能够更好地节省云租户的成本。

表 9　在第 150 次迭代时求得的各项参数指标

算法	评价指标	不同时刻下求得的网络成本/元						平均优化时间/s
		0	10	60	110	160	210	
SCA	最优值	828 069	1 076 422	825 607	427 991	516 979	480 093	32.4
	最差值	1 055 416	1 506 106	986 454	607 134	739 784	596 549	
	平均值	928 280	1 235 600	889 050	511 295	624 050	511 310	
	标准差	0.011 4	0.022 2	0.008 2	0.008 9	0.011 15	0.006 4	
HACO	最优值	486 001	632 933	498 751	368 129	377 387	379 825	80.3
	最差值	497 652	643 819	509 281	389 291	389 291	380 927	
	平均值	506 321	654 292	519 282	409 281	392 818	382 928	
	标准差	0.001 6	0.001 7	0.001 6	0.003 2	0.001 1	0.000 3	
DMPA	最优值	26 822	38 553	24 804	8 679	17 246	11 930	40.3
	最差值	40 620	43 429	40 746	198 70	28 230	18 303	
	平均值	38 176	40 253	35 001	14 874	26 910	17 118	
	标准差	0.000 8	0.000 3	0.000 8	0.000 6	0.000 7	0.000 4	
MPA	最优值	718 927	976 660	666 658	428 247	472 941	426 939	10.1
	最差值	1 302 159	1 869 937	1 017 268	578 997	720 338	596 093	
	平均值	1 061 118.4	1 358 956.4	887 905.3	482 075.3	611 069.1	506 363.3	
	标准差	0.029 6	0.045 1	0.018 1	0.007 8	0.012 5	0.008 5	
HPSO	最优值	355 871	348 759	341 442	387 887	346 219	345 673	13.6
	最差值	534 225	539 056	414 052	465 897	490 567	572 424	
	平均值	428 470	434 420	390 880	422 280	418 050	419 120	
	标准差	0.009 1	0.009 6	0.003 9	0.003 9	0.007 2	0.012	

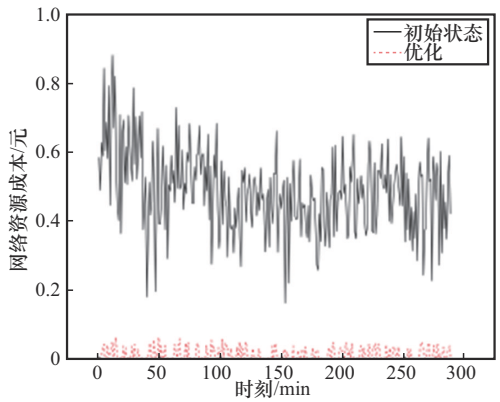


图 9　各时刻下网络资源成本变化

6　结束语

本文聚焦于多云环境下集群网络特性的深度采集与详尽分析，并据此在Kubernetes平台上设计且实现了一个创新模型。此模型具备根据网络资源实时状态动态调整计算资源分配的能力，同时支持在多云及多地域环境下实现负载的高可用性部署。该模型成功地满足了云原生网络环境下对成本优化的迫切需求，为云原生应用系统构建了一个统一的云化承载平台与高效的运维管理体系。通过构建原型系统并进行实验验证，本文证明了该架构的可行性与实用性。这一研究成果表明，云原生技术不仅在网络服务的灵活管理方面展现巨大潜力，而且能够大幅度降低网络资源的使用成本，为采用多云策略的租户带来了可观的经济效益。

参考文献

[1] 中国信息通信研究院 . 云原生发展白皮书（2020 年）[R]. 2020.
China Academy for Information and Communications Technology. Cloud native development white paper (2020) [R]. 2020.

[2] ACHAR S. An overview of environmental scalability and security in hybrid cloud infrastructure designs[J]. Asia Pacific Journal of Energy and Environment, 2021, 8(2): 39-46.

[3] VIEIRA C C A, BITTENCOURT L F, GENEZ T A L, et al. RAaaS: Resource Allocation as a Service in multiple cloud providers[J]. Journal of Network and Computer Applications, 2024, 221: 103790.

[4] MANGALAMPALLI S S, KARRI G R, MOHANTY S N, et al. Multi-objective prioritized task scheduler using improved asynchronous advantage actor critic (a3c) algorithm in multi cloud environment[J]. IEEE Access, 2024, 12: 11354-11377.

[5] SURESH P, KEERTHIKA P, MANJULA DEVI R, et al. Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment[J]. Applied Soft Computing, 2024, 165: 112129.

[6] ABUALIGAH L, DIABAT A, MIRJALILI S, et al. The arithmetic optimization algorithm[J]. Computer Methods in Applied Mechanics and Engineering, 2021, 376: 113609.

[7] MOIN M M, NARAYAN D G, PATIL S. A hybrid bio-inspired algorithm for routing in software defined networks[C]//Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT). Piscataway: IEEE Press, 2021: 1-7.

[8] FULBER-GARCIA V, LUIZELLI M C, DOS SANTOS C R P, et al. Customizable mapping of virtualized network services in multi-datacenter environments based on genetic metaheuristics[J]. Journal of Network and Systems Management, 2023, 31(4): 71.

[9] RANKOTHGE W, MA J F, LE F, et al. Towards making network function virtualization a cloud computing service[C]//Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). Piscataway: IEEE Press, 2015: 89-97.

[10] CAÑETE A, AMOR M, FUENTES L. HADES: an NFV solution for energy-efficient placement and resource allocation in heterogeneous infrastructures[J]. Journal of Network and Computer Applications, 2024, 221: 103764.

[11] MIANO S, RISSO F, BERNAL M V, et al. A framework for eBPF-based network functions in an era of microservices[J]. IEEE Transactions on Network and Service Management, 2021, 18(1): 133-151.

[12] WANG Z, SUN D, XUE G T, et al. Ada-Things: an adaptive virtual machine monitoring and migration strategy for Internet of Things applications[J]. Journal of Parallel and Distributed Computing, 2019, 132: 164-176.

[13] MARQUES G, SENNA C, SARGENTO S, et al. Proactive resource management for cloud of services environments[J]. Future Generation Computer Systems, 2024, 150: 90-102.

[14] FARAMARZI A, HEIDARINEJAD M, MIRJALILI S, et al. Marine predators algorithm: a nature-inspired metaheuristic[J]. Expert Systems with Applications, 2020, 152: 113377.

[15] CIEPLIŃSKI P, GOLAK S. Crossover operator inspired by the selection operator for an evolutionary task sequencing algorithm[J]. Applied Sciences, 2024, 14(24): 11786.

[16] TAILLARD E. Robust taboo search for the quadratic assignment problem[J]. Parallel Computing, 1991, 17(4/5): 443-455.

[17] BURKARD R E, ÇELA E, KARISCH S E, et al. QAPLIB[DB]. 2011.

[18] SADIQ A S, DEHKORDI A A, MIRJALILI S, et al. Nonlinear marine predator algorithm: a cost-effective optimizer for fair power allocation in NOMA-VLC-B5G networks[J]. Expert Systems with Applications, 2022, 203: 117395.

[19] AMIRTEIMOORI A, MAHDAVI I, SOLIMANPUR M, et al. A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation[J]. Computers & Industrial Engineering, 2022, 173: 108672.

[20] KıLıÇ H, YUZGEC U Y. Improved antlion optimization algorithm for quadratic assignment problem[J]. Malaysian Journal of Computer Science, 2021, 34(1): 34-60.

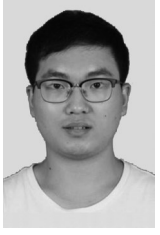
[21] XUE J K, SHEN B. Dung beetle optimizer: a new meta-heuristic algorithm for global optimization[J]. The Journal of Supercomputing, 2023, 79(7): 7305-7336.

[22] 高明, 刘铭, 陈泱婷, 等. 基于Kubernetes的多云网络成本优化模型[J]. 电信科学, 2023, 39(2): 71-82.
GAO M, LIU M, CHEN Y T, et al. Cost optimization model for multi-cloud network based on Kubernetes[J]. Telecommunications Science, 2023, 39(2): 71-82.

[23] SADEGHI M, AGHAYAN I, GHAZNAVI M. A cuckoo search based approach to design sustainable transit network[J]. Transportation Letters, 2021, 13(9): 635-648.

[作者简介]

高明（1979– ），男，博士，浙江工商大学副教授，浙江省新型网络标准及应用技术重点实验室副主任，主要研究方向为新型网络体系架构和云计算。



沈艺程（1998– ），男，浙江工商大学硕士生，主要研究方向为通信网络、云原生、智能优化算法。

刘铭（1997– ），男，浙江工业大学计算机科学与技术学院博士生，主要研究方向为时空数据分析、网络科学。