研究与开发

基于 Kubernetes 的多云网络成本优化模型

高明，刘铭，陈泱婷，王伟明
（浙江工商大学信息与电子工程学院，浙江 杭州 310018）

摘　要：以 Kubernetes 为代表的云原生编排系统在多云环境中被云租户广泛使用，随之而来的网络观测性问题愈发突出，跨云跨地区的网络流量成本尤为突出。在 Kubernetes 中引入扩展的伯克利数据包过滤器（extended Berkeley packet filter，eBPF）技术采集操作系统内核态的网络数据特征解决网络观测问题，随后将网络数据特征建模为二次分配问题（quadratic assignment problem，QAP），使用启发式搜索与随机搜索组合的方法在实时计算的场景下求得最佳近优解。此模型在网络资源成本优化中优于 Kubernetes 原生调度器中仅基于计算资源的调度策略，在可控范围内增加了调度链路的复杂度，有效降低了多云多地区部署环境中的网络资源成本。

关键词：Kubernetes；eBPF；多云网络；二次分配问题
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000–0801.2023028

Cost optimization model for multi-cloud network based on Kubernetes

GAO Ming, LIU Ming, CHEN Yangting, WANG Weiming
School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: The cloud-native scheduling system, represented by Kubernetes, is widely used by cloud tenants in a multi-cloud environment. The problem of network observation becomes more and more serious, especially the cost of network traffic across cloud and region. In Kubernetes, the eBPF technology was introduced to collect the network data features of kernel state of operating system to solve the network observation problem, and then the network data features were modeled as QAP, a combination of heuristic and stochastic optimization was used to obtain the best near optimal solution in a real-time computing scenario. This model is superior to the Kubernetes native scheduler in the cost optimization of network resources, which is based on the scheduling strategy of computing resources only, and increases the complexity of scheduling links in a controllable range, effectively reduces the cost of network resources in a multi-cloud area deployment environment.

Key words: Kubernetes, eBPF, multi-cloud network, quadratic assignment problem

收稿日期：2022–11–02；**修回日期**：2023–02–09
基金项目：国家自然科学基金资助项目（No.61871468）；浙江省基础公益研究计划项目（No.LGG20F010015）；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）
Foundation Items：The National Natural Science Foundation of China (No.61871468), The Basic Public Welfare Research Program of Zhejiang Province (No.LGG20F010015), The Key Laboratory of Network Standards and Applied Technology Foundation of Zhejiang Province (No.2013E10012)

0 引言

云原生模式是面向云进行软件工程设计的一种思想理念，是一种充分发挥云效能的最佳实践，用于帮助企业构建弹性可靠、松耦合、易管理的软件系统，从而提升软件的交付效率，降低软件的运维复杂度[1]。Gartner 行业数据显示：到 2025 年，将有超过 95%的新数字项目以云原生平台为基础设施，远高于 2021 年 40%的比例[2]。

以 Kubernetes 为代表的云原生编排系统已被广泛应用在云中，各大云厂商均推出了云原生编排系统产品，如 Google 的 GKE（Google Kubernetes 引擎）、Microsoft 的 AKS（Azure Kubernetes 服务）、Amazon 的 EKS（Elastic Kubernetes 服务）、阿里云的 ACK（阿里云容器服务 Kubernetes 版）以及华为云的 CEE（云容器引擎）。随之而来的网络观测性问题愈发突出，云租户难以感知云原生网络的实际带宽及流量画像。扩展的伯克利数据包过滤器（extended Berkeley packet filter，eBPF）技术的面世推动了云原生可观测性的发展，使得网络观测性问题的解决成为可能。2014 年 eBPF 代码合入 Linux3.18 内核的主分支，新的 Linux 内核加进去的内核观测功能，从系统内核层读取网络数据包，从而观测到云原生编排系统集群内部服务之间的相互调用关系与调用频率，有效解决了云原生网络的观测问题[3]。

此外，没有任何一家云厂商在云的可靠性方面可保证 100%的服务等级协定（service level agreement，SLA），但通过云厂商历年的公开事故报告统计可发现，几乎不存在所有云在同一时刻均不可用的情况。所以，云租户为确保服务的实时可用，已逐渐采用多云部署的方案。

云原生编排系统通常基于计算资源进行服务调度，而非基于网络资源进行服务调度，因此在多云部署的具体实践中，跨云跨地区的网络流量成本居高不下，网络资源成本优化问题亟待解决[4]。针对上述问题，本文提出了一种基于 Kubernetes 的多云网络成本优化模型，目标是为云租户降低使用多云、多地区、多数据中心之间云网络资源的经济成本，该模型的工作步骤如下。

步骤 1 基于 Kubernetes 进行多云多地区服务部署，在操作系统内核态通过 eBPF 指标采集器实时采集不同服务之间的网络数据包，并解析数据包头部的元数据，生成含网络通信数据包数量的权重网络拓扑。

步骤 2 将权重网络拓扑的数据特征数学建模为二次分配问题（quadratic assignment problem，QAP）。

步骤 3 使用蚁群优化（ant colony optimization，ACO）算法、模拟退火（simulated annealing，SA）算法、遗传算法（genetic algorithm，GA）、免疫算法（immune algorithm，IA）4 种启发式搜索算法与随机优化（stochastic optimization，SO）并行的方式计算每一个时刻的数据特征，计算持续的时间为一个单位时刻，最终采用最佳算法的近优解，并生成使全局网络资源经济成本最小的计算资源调度方案。

步骤 4 计算资源调度方案进入 Kubernetes 调度器队列，在下一时刻结束时对此调度方案进行评估，阶段性反馈成本优化率用于优化模型。

实验数据表明，此模型在网络资源成本优化中优于基于计算资源的调度方法，有效降低了多云多地区部署环境中的网络资源成本，虽然增加了调度链路的复杂性，但在当前样本数据规模下，其调度链路复杂度带来的计算资源用量水位的上升是固定可调控的。

1 相关工作

目前有大量文献对网络资源成本优化问题做出了深入的研究。文献[5]通过将深度强化学习原理引入软件定义网络的路由过程优化网络资源成

本。文献[6-8]通过蚁群算法、遗传算法、粒子群算法等启发式算法进行路由优化降低网络资源成本。文献[9]提出了混合虚拟网络环境的概念，使用遗传算法部署虚拟网络功能（virtual network function，VNF），并设计了 4 种遗传算法用于最小化带宽开销和最大化链路利用率优化降低网络资源成本。文献[10]利用复制的方式部署 VNF 保证网络的负载均衡，并为大型网络设计了遗传算法优化降低网络资源成本。文献[11]通过遗传算法在云数据中心动态部署 VNF，最大限度地利用网络资源优化降低网络资源成本。

上述文献主要通过优化路由算法的方式降低网络资源的传输成本，但在多云网络环境的应用中存在阻碍，首先云租户通常无法更改云厂商设定的网络路由策略等基础设施，其次云厂商网络通常基于覆盖网络（Overlay）实现弹性网络，即在现有物理网络的技术上创建的虚拟或逻辑网络，从而导致云租户无法感知物理网络的真实带宽水位。文献[12]提出了一种通过编排 VNF 资源并将其合理部署在物理网络中的方式，通过调整计算资源的位置达到了优化网络资源成本的效果，此类研究具备多云环境的应用条件。

本文基于多云环境研究，在网络成本的优化方案中通过不改变网络资源拓扑仅改变计算资源位置的方法，进一步实现计算资源的位置随时间及流量的变化进行动态调整，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实现的时间成本较高，而基于云原生技术部署服务可实现毫秒级跨云跨地区调度服务，使此方案的实现成为可能。

2　系统架构

模型采用 Kubernetes 云原生编排系统作为基础设施，以 Kubernetes 中最小可调度的计算单元（Pod）之间的流量特征为研究对象，实现云原生网络资源成本优化工作的实时计算，并根据最佳近优解分时刻调度计算资源。模型的系统架构如图 1 所示，自

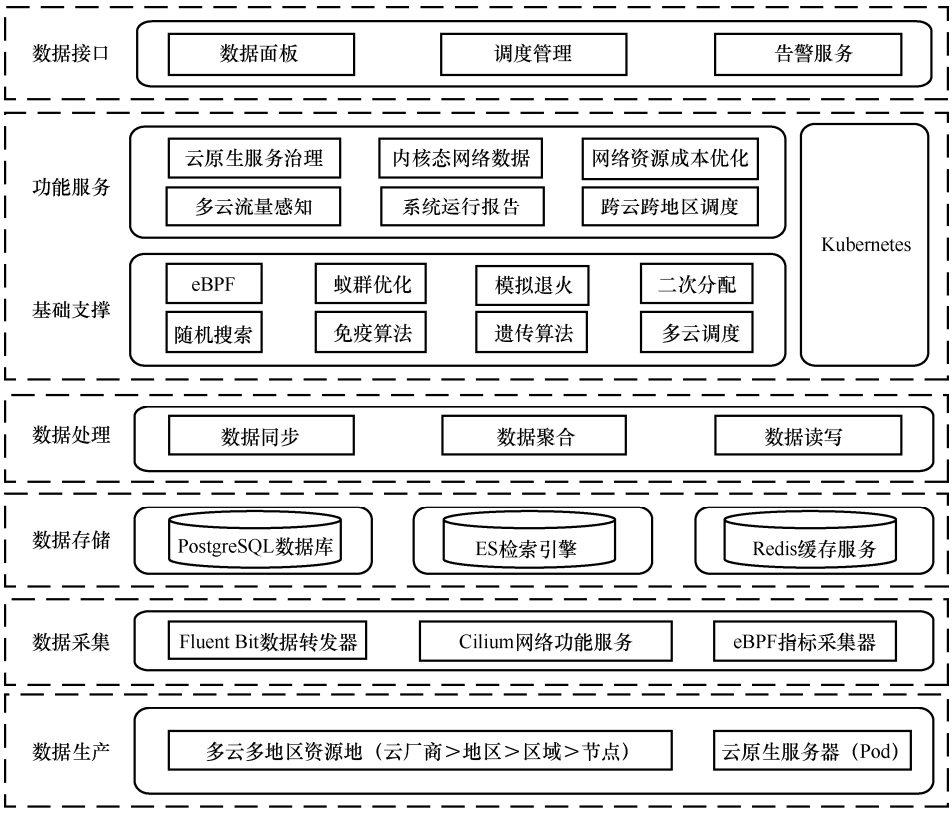

图 1　系统架构

下而上由数据生产、数据采集、数据存储、数据处理、基础支撑、功能服务与数据接口组成。

网络资源成本优化的基础是网络资源的数据信息，首要任务就是采集网络资源的数据并进行结构化用于数据分析。采集云原生网络数据包的核心技术是 eBPF，在经典的网络数据包采集工具链中，通常在用户态采集网络数据包，而通过 eBPF 在更靠近硬件网卡的内核态位置采集网络数据比用户态采集数据具有更高的性能优势[13-14]。

图 1 中的数据生产和数据采集模块属于云原生网络特征观测子系统，是模型的数据来源。云原生网络特征观测子系统在多云多地区的计算资源池内通过 eBPF 指标采集器从内核态采集业务服务与网络功能服务之间的网络数据包，并将其输出到 Cilium 网络功能服务中，由 Fluent Bit 数据转发器将 Cilium 内的网络数据包数据写入 Kafka 内存型消息队列，并使用 Logstash 实时数据传输管道工具消费消息队列内的数据，将元数据的关键字段清洗后转储到分布式时间序列检索引擎（Elastic search engine，ES）集群中，Kibana 数据检索工具可以对 ES 集群中的全量数据进行查询。在数据处理的过程中读取 ES 集群中的数据，处理完将数据落库到 PostgreSQL 结构化数据库中。

最终处理完成的网络数据核心指标包含每个服务被哪些服务连接、连接了哪些服务以及服务之间的连接频率。根据这些指标可以通过调整服务运行的时间和空间优化计算资源成本和网络资源成本。调整服务运行的时间，例如，时效性要求较低且数据量较少的服务可定时运行服务，处理完数据后销毁服务，从而有效节省弹性计算资源成本。调整服务运行的空间，例如，通信频率较高的两个服务的物理间隔位置较远，可增加服务之间的亲和性使其运行在同一台宿主机，降低网络资源成本。

模型数据流图如图 2 所示，在多家云厂商（Cloud）的多个地区（Region）的多个数据中心（Zone）的多个 Kubernetes 工作平面节点（Node）中的操作系统内核态采集到网络数据包后，将数据特征清洗并适配蚁群优化算法、模拟退火算法、遗传算法和免疫算法 4 种启发式搜索算子以及随机搜索算子进行计算，并行在一单位时刻时间内迭代多次取得近优解，获取最佳最优解后生成计算资源新的放置位置，每个时刻循环往复。根据业务需求一单位时刻定义为 5 min，时间太短将导致调度频繁，时间太长将导致模型无法及时对流量的变化产生反馈。最佳算子的产出结果是下一个时刻的目标计算资源的位置状态，调度器以受控速率改变计算资源的实际位置状态，使其达到期望位置状态，本文的调度器复用 Kubernetes 原生计算资源调度器。

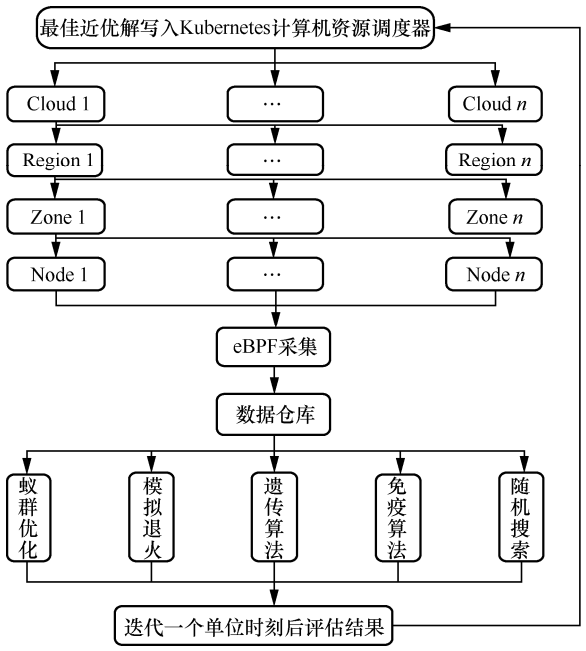

图 2 模型数据流图

3 数学建模

在云原生环境中的成本主要分为计算资源成本和网络资源成本两个部分，本文重点在固定计算资源成本的前提下分时刻优化网络资源成本。在优化网络资源成本时将 Pod 所在的计算资源池的子集（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，每个 Slot 最大仅可容纳 1 个 Pod，且 Slot 具备流量成本权重表中的位置属性，流量成本权重见表 1。

表 1　流量成本权重

序号	Slot	Node	Zone	Region	Cloud	成本/元
1	相同	相同	相同	相同	相同	0
2	不同	相同	相同	相同	相同	0
3	不同	不同	相同	相同	相同	0
4	不同	不同	不同	相同	相同	0
5	不同	不同	不同	不同	相同	C_{mn}
6	不同	不同	不同	相同	不同	C_{mn}
7	不同	不同	不同	不同	不同	C_{mn}

定义 Pod m 与 Pod n 之间的流量成本权重为 C_{mn}，定义位置属性列表 $L=\{\text{Slot},\text{Node},\cdots,\text{Cloud}\}$，例如，Pod 0 当前运行在 Slot x 上，位于 A 云的 B 地区的 C 数据中心的 D 机器上，则此时 $L=\{x,\text{D},\text{C},\text{B},\text{A}\}$。遍历 Pod m 与 Pod n 的当前位置属性列表，若 Region 或 Cloud 属性不一致，则二者的流量成本权重 C_{mn} 根据云厂商不同地区的流量价格计算求得，如式（1）所示。

$$C_{mn}=P\times\sum_{\text{len}(L)-3}^{\text{len}(L)-1}L_i^m\oplus L_i^n \tag{1}$$

其中，P 为源云厂商源地区向目的云厂商目的地区发送一个数据包的单价，单位为元；$\oplus$ 为异或逻辑运算，源 Pod 与目的 Pod 的位置属性列表一致时权重数值为 0，不一致时根据位置属性列表按位异或运算并相加得到权重数值，此处定义按位异或运算时仅包含 Region 和 Cloud 属性。由于样本数据中采集的网络数据包均为传输层的传输控制协议（transmission control protocol，TCP）数据包，需要封装传输层和网络层包头后进入数据链路层计费，在 RFC 894 中定义了 Ethernet 网络类型在数据链路层的最大传输单元（maximum transmission unit，MTU）为 1 500 byte，因此样本数据中单个数据包大小以 1 500 byte 计算成本。云厂商流量通常按每 GB 流量进行单价制定，而 $1\text{ GB}=2^{30}$ byte，因此需要将云厂商 1 GB 流量的单价作为约 72 万个数据包的总价。

本文通过计算 Pod 之间的通信成本和流量的方式调整 Pod 位置，以此优化网络资源成本，因此该方法可建模为组合优化中的 QAP。组合优化是计算资源调度策略实施过程中重要的步骤，其过程是结合不同的调度目标及成本约束给出最优的组合权重[15]。由于 QAP 已被证明为 NP 难问题，无法在多项式时间内计算得到准确结果[16]，而网络流量的数据特征具备时效性，计算资源的调度方案要求实时性，所以最终方案的实施需要兼顾时间成本和优化效率。

基于 Kubernetes 的多云网络成本优化模型需要根据上一时刻网络流量的数据特征规划出网络成本最优的计算资源调度策略，并用于下一个时刻的计算资源调度。给定 n 个 Pod 和 n 个 Slot（$n\in\mathbf{N}^+$），定义矩阵 $\boldsymbol{A}=\left(a_{ij}\right)$ 和矩阵 $\boldsymbol{B}=(b_{ij})(1\leqslant i,j\leqslant n)$，其中，$\boldsymbol{A}$ 表示 Pod 间的流量，a_{ij} 是 Pod i 和 Pod j 之间的流量；B 表示 Pod 间的通信成本，b_{ij} 是 Slot i 和 Slot j 之间的通信成本。每个 Pod 需要被分配一个 Slot，如式（2）所示。

$$P:\{1,2,\cdots,n\}\to\{1,2,\cdots,n\},P\in\prod(n) \tag{2}$$

其中，P 为 Slot 的分配方案；n 为 Pod 和 Slot 的数量；$\prod(n)$ 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。

定义当执行分配方案 P 时，所有 Pod 之间的网络通信成本为 C，如式（3）所示。

$$C(P)=\sum_{i=1}^{n}\sum_{j=1}^{n}\left\{a_{ij}b_{p_ip_j}\right\} \tag{3}$$

其中，a_{ij} 表示 Pod i 和 Pod j 之间的交互的数据包数量，p_i 和 p_j 分别表示 Pod i 和 Pod j 被分配的 Slot 编号，$b_{p_ip_j}$ 为两个 Pod 目前分配到的 Slot 之前进行数据包交互的权重，分配方案的优化目标是使所有 Pod 之间的网络通信成本最低，即 $\min\left(C(P)\right)$。

4 模型建立

模型的核心是在限定时间内求解不定规模的QAP，在无法得知最优解的情况下获取最佳近优解。

云上跨 Cloud、Region、Node、Zone、Pod 之间的流量均属于东西向流量，模型的本质是降低云原生网络东西向流量的成本。模型需要在每个时刻决策把每一个 Pod 分配到指定的 Slot，每个 Slot 具备不同的 Node、Zone、Region 和 Cloud 位置，并且每个位置属性均有固定配额，存在资源门限，一旦达到限制，则无法调度。因此，合理分配所有时刻的所有 Pod 并调度到指定 Slot，降低总网络资源成本是模型的主要工作。

网络资源成本最低的极限方案是把所有计算资源都放在同一个位置，然而单位置有计算资源上限瓶颈，无法满足大规模计算场景，而且单地区或者单云的方案存在限额和灾备问题，模型需要满足以上约束条件。

以式(4)和表 2 为例对模型进行具体化描述，式（4）中的矩阵 $\boldsymbol{A}^{\text{Native}}$ 表示所有 Pod 在时刻 0 时所在 Slot 相互之间的流量数据，以 1 000 万个数据包为单位表示，即统计数据的第一个 5 min Pod 之间交互的数据包数量，示例 Pod 位置属性见表 2，展示了每一个 Pod 的位置属性，即当前时刻 Pod 运行所在的 Node、Zone、Region 和 Cloud。在模型的实际应用中，矩阵 $\boldsymbol{A}^{\text{Native}}$ 为 eBPF 工具在内核态采集的数据，表 2 为 Kubernetes 计算资源的状态，可通过接口实时获取。

$$\boldsymbol{A}^{\text{Native}}=\begin{pmatrix}0&2&0&0&10&0&25&0\\2&0&0&30&0&0&0&0\\0&0&0&0&0&0&15&50\\0&30&0&0&0&0&0&0\\10&0&0&0&0&0&0&0\\0&0&0&0&0&0&20&0\\25&0&15&0&0&20&0&0\\0&0&50&0&0&0&0&0\end{pmatrix} \tag{4}$$

表 2 示例 Pod 位置属性

Pod	Node	Zone	Region	Cloud
Pod 0	ECS 0	BJ A	北京	云厂商 I
Pod 1	ECS 1	BJ A	北京	云厂商 I
Pod 2	ECS 2	HZ A	杭州	云厂商 II
Pod 3	ECS 3	HZ A	杭州	云厂商 II
Pod 4	ECS 4	HZ B	乌兰察布	云厂商 I
Pod 5	ECS 5	HZ B	乌兰察布	云厂商 I
Pod 6	ECS 6	SZ A	青岛	云厂商 II
Pod 7	ECS 7	SZ A	青岛	云厂商 II

表 2 中示例 Pod 当前的位置属性通过式（1）可生成 Slot 之间的流量成本权重，结果为式（5）中的矩阵 $\boldsymbol{B}$，表示在当前 Pod 位置属性前提下 Pod 之间传输 1 000 万个数据包所需要的经济成本，以元为单位表示。

$$\boldsymbol{B}=\begin{pmatrix}0&0&10&10&10&10&9&9\\0&0&10&10&10&10&9&9\\10&10&0&0&11&11&11&11\\10&10&0&0&11&11&11&11\\10&10&11&11&0&0&11&11\\10&10&11&11&0&0&11&11\\9&9&11&11&11&11&0&0\\9&9&11&11&11&11&0&0\end{pmatrix} \tag{5}$$

同时，通过表 2 可生成 Pod 与 Slot 的关联关系，如式（6）中向量 $\boldsymbol{P}^{\text{Native}}$ 所示，$\boldsymbol{P}^{\text{Native}}$ 为一个 8 维向量，每个维度代表一个 Slot，维度上分量的数值表示 Pod 的编号，当 Pod 分布情况为向量 $\boldsymbol{P}^{\text{Native}}$ 时，总网络资源成本可由式（3）根据矩阵 $\boldsymbol{A}^{\text{Native}}$ 和 $\boldsymbol{B}$ 计算所得，为 3 120 元。

$$\boldsymbol{P}^{\text{Native}}=\begin{bmatrix}0&1&2&3&4&5&6&7\end{bmatrix} \tag{6}$$

通过 ACO 算子对 Pod 的当前分布情况限定时间 1 个时刻内进行启发式搜索，得到近优解式(7)中的向量 $\boldsymbol{P}^{\text{ACO}}$。

$$\boldsymbol{P}^{\text{ACO}}=\begin{bmatrix}6&5&3&1&0&4&7&2\end{bmatrix} \tag{7}$$

在 ACO 算子的初始迭代中，采用随机搜索的方式生成实例向量。从样本向量内随机选取一个

维度获得分量，当前示例数据规模为 $n=8$，则第一个分量的随机概率为 1/8，第二个分量的随机概率为 1/7，以此类推。ACO 算子的特点在于信息素概念的引入，初始迭代中每个维度的分量信息素均为 0，随机选择两个维度进行分量置换，并进行成本计算验证，对成本降低情况的两个分量给予正向的信息素奖励，反之给予负向的信息素奖励。在后续的迭代中，信息素的大小与置换的选择概率成正比，为规避算子的快速收敛可能，设定阈值决定信息素选择和随机选择的概率，最终求得最佳近优解。

根据花式索引算法通过向量 $\boldsymbol{P}^{\mathrm{ACO}}$ 对矩阵 $\boldsymbol{A}$ 重新采样，生成矩阵 $\boldsymbol{P}^{\mathrm{ACO}}$，花式索引算法即使用向量 $\boldsymbol{P}^{\mathrm{ACO}}$ 中每个维度的分量作为索引对矩阵 $\boldsymbol{A}$ 的行和列先后进行再采样，生成的新矩阵 $\boldsymbol{P}^{\mathrm{ACO}}$ 如式（8）所示。将矩阵 $\boldsymbol{P}^{\mathrm{ACO}}$ 和矩阵 $\boldsymbol{B}$ 通过式（3）计算得到总网络资源成本为 $C(\boldsymbol{P}^{\mathrm{ACO}})=814$ 元。

$$\boldsymbol{A}^{\mathrm{ACO}}=\begin{pmatrix}0&20&0&0&25&0&0&15\\20&0&0&0&0&0&0&0\\0&0&0&30&0&0&0&0\\0&0&30&0&2&0&0&0\\25&0&0&2&0&10&0&0\\0&0&0&0&10&0&0&0\\0&0&0&0&0&0&0&50\\15&0&0&0&0&0&50&0\end{pmatrix} \tag{8}$$

同理，使用 GA 算子求得 $\boldsymbol{P}^{\mathrm{GA}}$、$\boldsymbol{A}^{\mathrm{GA}}$ 和 $C(\boldsymbol{P}^{\mathrm{GA}})=794$ 元，如式（9）和式（10）所示。

$$\boldsymbol{P}^{\mathrm{GA}}=\begin{bmatrix}5&6&3&1&7&2&0&4\end{bmatrix} \tag{9}$$

$$\boldsymbol{A}^{\mathrm{GA}}=\begin{pmatrix}0&20&0&0&0&0&0&0\\20&0&0&0&0&15&25&0\\0&0&0&30&0&0&0&0\\0&0&30&0&0&0&2&0\\0&0&0&0&0&50&0&0\\0&15&0&0&50&0&0&0\\0&25&0&2&0&0&0&10\\0&0&0&0&0&0&10&0\end{pmatrix} \tag{10}$$

同理，使用 IA 算子求得 $\boldsymbol{P}^{\mathrm{IA}}$、$\boldsymbol{A}^{\mathrm{IA}}$ 和 $C(\boldsymbol{P}^{\mathrm{IA}})=820$ 元，如式（11）和式（12）所示。

$$\boldsymbol{P}^{\mathrm{IA}}=\begin{bmatrix}0&4&1&3&7&2&5&6\end{bmatrix} \tag{11}$$

$$\boldsymbol{A}^{\mathrm{IA}}=\begin{pmatrix}0&10&2&0&0&0&0&25\\10&0&0&0&0&0&0&0\\2&0&0&30&0&0&0&0\\0&0&30&0&0&0&0&0\\0&0&0&0&0&50&0&0\\0&0&0&0&50&0&0&15\\0&0&0&0&0&0&0&20\\25&0&0&0&0&15&20&0\end{pmatrix} \tag{12}$$

同理，使用 SA 算子求得 $\boldsymbol{P}^{\mathrm{SA}}$、$\boldsymbol{A}^{\mathrm{SA}}$ 和 $C(\boldsymbol{P}^{\mathrm{SA}})=794$ 元，如式（13）和式（14）所示。

$$\boldsymbol{P}^{\mathrm{SA}}=\begin{bmatrix}5&6&3&1&7&2&0&4\end{bmatrix} \tag{13}$$

$$\boldsymbol{A}^{\mathrm{SA}}=\begin{pmatrix}0&20&0&0&0&0&0&0\\20&0&0&0&0&15&25&0\\0&0&0&30&0&0&0&0\\0&0&30&0&0&0&2&0\\0&0&0&0&0&50&0&0\\0&15&0&0&50&0&0&0\\0&25&0&2&0&0&0&10\\0&0&0&0&0&0&10&0\end{pmatrix} \tag{14}$$

同理，使用 SO 算子求得 $\boldsymbol{P}^{\mathrm{SO}}$、$\boldsymbol{A}^{\mathrm{SO}}$ 和 $C(\boldsymbol{P}^{\mathrm{SO}})=974$ 元，如式（15）和式（16）所示。

$$\boldsymbol{P}^{\mathrm{SO}}=\begin{bmatrix}5&4&1&3&0&6&7&2\end{bmatrix} \tag{15}$$

$$\boldsymbol{A}^{\mathrm{SO}}=\begin{pmatrix}0&0&0&0&0&20&0&0\\0&0&0&0&10&0&0&0\\0&0&0&30&2&0&0&0\\0&0&30&0&0&0&0&0\\0&10&2&0&0&25&0&0\\20&0&0&0&25&0&0&15\\0&0&0&0&0&0&0&50\\0&0&0&0&0&15&50&0\end{pmatrix} \tag{16}$$

综合对比上述 5 种搜索算子和仅基于计算资源的算子，即原生（Native）算子，启发式搜索和随机搜索的结果均存在随机性，仅基于计算资源的算

子存在大量网络资源成本浪费情况。示例样本网络成本优化算子对比如图 3 所示，由于示例数据采用了较小规模的样本，5 种搜索算子相对于 Native 算子均取得了较好的优化效果，其中 GA 算子和 SA 算子优化效果最佳，都将此样本初始的网络资源成本 3 120 元优化到了 794 元，优化率约为 74.55%。

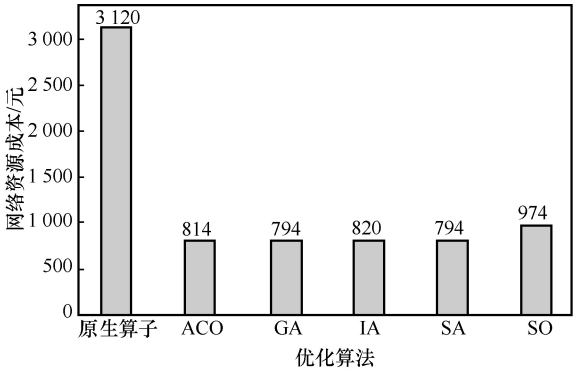

图 3　示例样本网络成本优化算子对比

当不同的网络成本优化算子出现相同的近优解时，随机取 1 个算子作为当前样本的最佳算子，此处选择 SA 算子，将其结果设置为激活状态。示例样本出现多算子相同解情况的原因是样本数据规模较小（$n=8$），而在实际情况中样本数据规模较大（$n>100$），在穷举的情况下需要计算 $n!$ 次，在限定 1 个时刻时间段内出现相同近优解的概率较低，因此对此类情况做随机处理，并不影响模型的准确性，反而会增加结果的多样性，促使模型进一步优化。

当 SA 算子的结果处于激活状态时，模型将 $\boldsymbol{P}^{\mathrm{SA}}$ 作为 Pod 的目标状态调度方案下发指令到计算资源调度器，示例样本计算资源调度方案对比如图 4 所示，表示 Pod 的位置从调度方案 $\boldsymbol{P}^{\mathrm{Native}}$ 到调度方案 $\boldsymbol{P}^{\mathrm{SA}}$ 的变化对比，每个 Pod 占用一个 Slot，Pod 与 Pod 之间的连接线上标记的数字表示当前时刻两个 Pod 之间交换数据包的数量。直观显示，经过模型优化后的计算资源调度方案降低了跨多个位置属性的网络连接，从而达到了网络资源成本优化的目的。

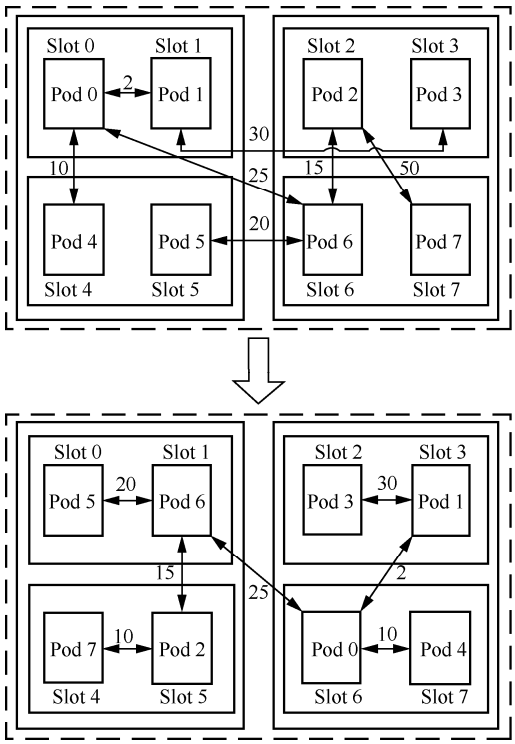

图 4　示例样本计算资源调度方案对比

此模型在 n 时刻采集的网络数据，通过 $n+1$ 时刻的计算能力，应用于 $n+2$ 时刻。如示例样本数据经过第 1 个时刻的内核态数据采集，第 2 个时刻的调度方案搜索与调度，第 3 个时刻的计算资源分配方案来源于第 1 个时刻的网络数据和第 2 个时刻的计算能力。最终在第 3 个时刻结束时，将采集到的第 3 个时刻的流量数据和 Pod 位置数据进行网络资源成本计算，与预期数据对比评估并用于模型的持续优化。

5　模型验证

模型验证实验的样本数据采集自某互联网平台生产环境 2022 年 1 月 22 日全天的数据（已做脱敏处理，仅保留传输层数据包头特征）。初始样本每 5 min 聚合 1 份数据特征，全天共有 288 个 5 min，即 $T(1\leqslant|T|\leqslant 288)$ 为所有时刻的集合，将初始样本数据拆分为 288 个子样本。在每个子样本数据中共有 $Y(1\leqslant Y\leqslant 174)$ 个 Pod、$X(1\leqslant X\leqslant 87)$ 个 Node、$L(1\leqslant L\leqslant 3)$ 个 Zone、$M(1\leqslant M\leqslant 29)$ 个

Region、$N(1 \leqslant N \leqslant 3)$ 个 Cloud，其中 Slot、Node、Zone、Region 和 Cloud 组成计算资源池。

第 i 个 Pod 在时刻 t 的数据包连接请求需求是 $\boldsymbol{D}_i^t(1 \leqslant i \leqslant Y, t \in T)$。在时刻 t，第 i 个 Pod 到第 j 个 Pod 的网络数据包请求量为 $\boldsymbol{O}_{ij}^t(1 \leqslant i \leqslant Y, 1 \leqslant j \leqslant Y, t \in T)$。用 $\boldsymbol{E}$ 表示一组计算资源调度方案，即时刻 t 第 i 个 Pod 分配到第 x 个 Node、第 l 个 Zone、第 n 个 Cloud、第 m 个 Region 的状态 $\boldsymbol{E}_{xlnm}^t(1 \leqslant x \leqslant X, 1 \leqslant l \leqslant L, 1 \leqslant n \leqslant N, 1 \leqslant m \leqslant M, t \in T)$。第 i 个 Pod 在时刻 t 的总逻辑成为 $\boldsymbol{C}_i^t = \sum_Y^{i=1} \boldsymbol{O}_{ij}^t \mid \boldsymbol{E}_{xlnm}^t \mid (1 \leqslant i, j \leqslant Y, t \in T)$，其中，$\forall t \in T, 1 \leqslant i, j \leqslant Y, 1 \leqslant x \leqslant X, 1 \leqslant l \leqslant L, 1 \leqslant n \leqslant N, 1 \leqslant m \leqslant M$，$\boldsymbol{E}_{xlnm}^t \in \boldsymbol{Z}_0^+$，优化目标如式（17）所示。

$$\min_E \sum_{i=1}^{Y} S_i \tag{17}$$

即通过搜索得到一组满足约束条件的计算资源调度方案 $\boldsymbol{E}$，使其在时间集合 T 内的总网络通信成本最低。

模型验证模拟实验环境使用 4 台树莓派模拟 3 云 29 地区数据中心共计 174 个 Slot，使用 1 台二层交换机模拟多云多地区之间的专线网络，实验环境硬件设备信息见表 3。

表 3　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	S1205V	Switch
H0	树莓派主板	PI4B	openEuler
H1	树莓派主板	PI4B	openEuler
H2	树莓派主板	PI4B	openEuler
H3	树莓派主板	PI4B	openEuler

本实验中树莓派主板操作系统的版本采用了 2022 年 3 月发布的 openEuler 22.03 LTS，此版本中 Linux 内核采用了 5.10 版本，Linux 内核自 4.16 版本起支持 eBPF 功能。基于 openEuler Linux 操作系统部署了 Kubernetes 云原生编排系统，并在编排系统内部署实验环境软件，实验环境软件版本信息见表 4。

表 4　实验环境软件版本信息

名称	版本	类型
openEuler	22.03 LTS	操作系统
Linux Kernel	5.10	系统内核
Kubernetes	v1.24.3	资源编排系统
Cilium	v1.12.0	eBPF 网络功能
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

模拟实验首先从优化算子、迭代时间、迭代次数 3 个维度组合的方式对样本进行求解，形成多样化的对比实验任务。采样时刻 0、5、10、60、150 和 280 的样本数据，分别使用 5 种搜索算子对样本迭代 300 次，对比实验结果，实验任务见表 5。

表 5　实验任务

编号	算子	迭代次数	样本/时刻
0	蚁群优化	1～300	0/5/10/60/150/280
1	遗传算法	1～300	0/5/10/60/150/280
2	免疫算法	1～300	0/5/10/60/150/280
3	模拟退火	1～300	0/5/10/60/150/280
4	随机优化	1～300	0/5/10/60/150/280

每个实验任务均采用单核 CPU 处理器进行计算，实验任务结束后统计不同算子迭代 300 次所需时间，不同算子计算单个样本平均使用时间如图 5 所示，其中，GA 算子优化单个样本平均使用 16.93 min，远超单个时刻定义的时间 5 min，应做裁剪后使用结果，仅使用优化到 5 min 时的结果，如式（18）所示，$\text{Epoch}_{\text{target}}$ 表示目标迭代次数，floor 表示对结果向下取整，即 GA 算子采用第 88 次优化结果。同理，IA 算子采用第 219 次优化结果。ACO 算子、SA 算子和 SO 算子优化单个样本平均使用的时间在 5 min 内，可采用最终结果。

$$\text{Epoch}_{\text{target}} = \text{floor}\left(\frac{300 \times 5}{16.93}\right) \tag{18}$$

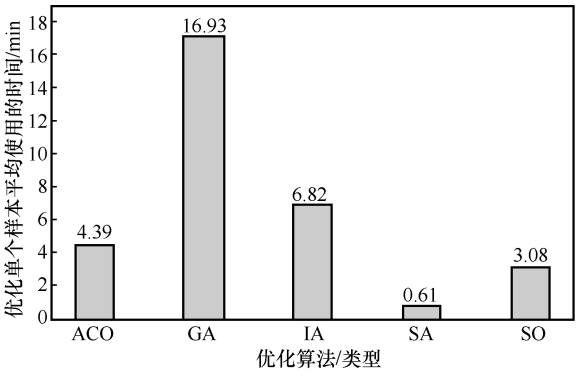

图 5　不同算子计算单个样本平均使用时间

表 5 实验任务中的样本时刻算子成本优化效果如图 6 所示，在 Pod 规模为 174 的场景中，对其中 6 份样本数据迭代运算 300 次，并进行效果对比。图 6 中，GA 算子第 88 次计算结果和 IA 算子在第 219 次计算结果的算子排名与第 300 次计算结果的算子排名无较大差异，因此，同时取 300 次迭代数据进行算子评估。其中，ACO 算子在试验结果中有 5/6 的样本取得了最佳近优解，GA 算子在 1/6 的样本中取得了最佳近优解，IA 算子、SA 算子和 SO 算子在小迭代次数时存在最佳近优解的情况，特别是 50 次以下的迭代，适用于求解时间要求较高的场景。当 Pod 资源规模较大，无法在限定时间内取得优化效

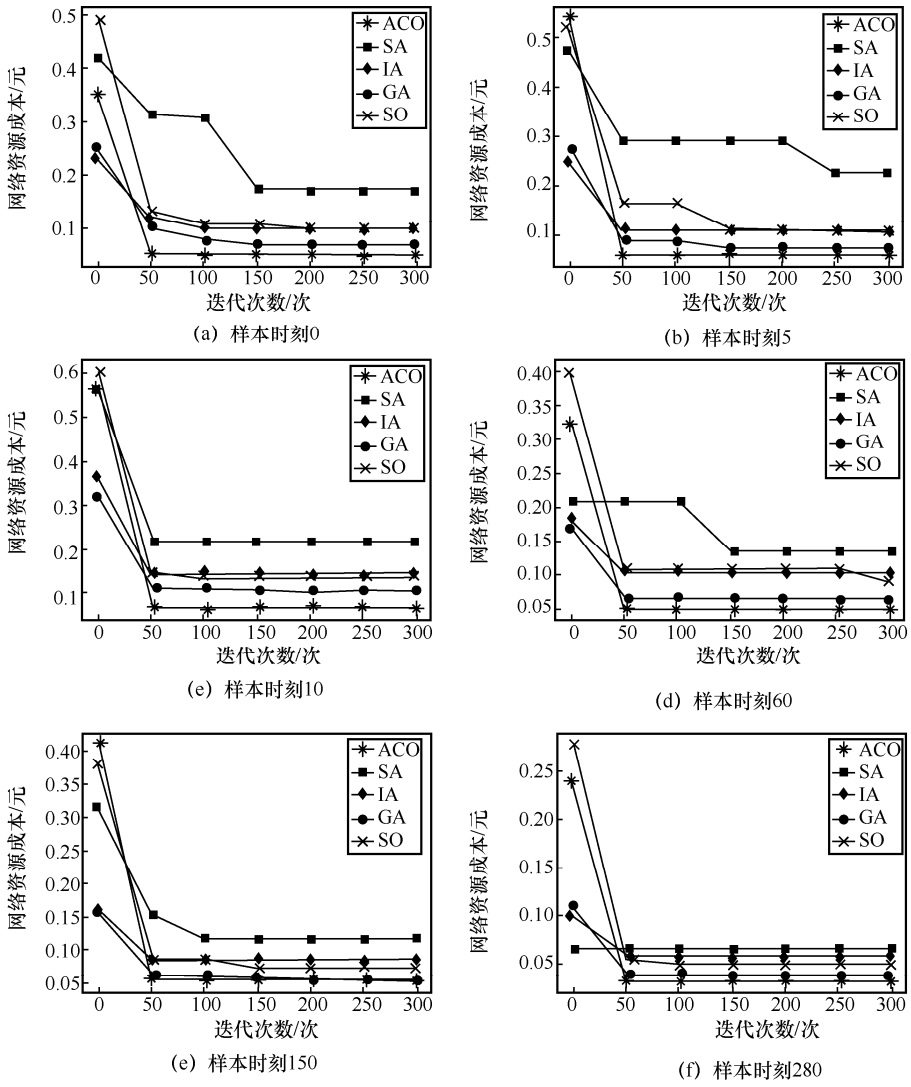

图 6　样本时刻算子成本优化效果

果时，需要对样本数据进行拆分处理，取多个样本数据子集的局部最优解作为全局最优解。

使用 ACO 算子和 GA 算子对全量样本数据进行计算，并保存每次的迭代的结果，参照原生调度方案对比每个时刻搜索算子的优化效果，每个时刻算子优化对比如图 7 所示，两种算子最优解均优于原生调度方案，因此，混合算子最优解更优于原生调度方案。

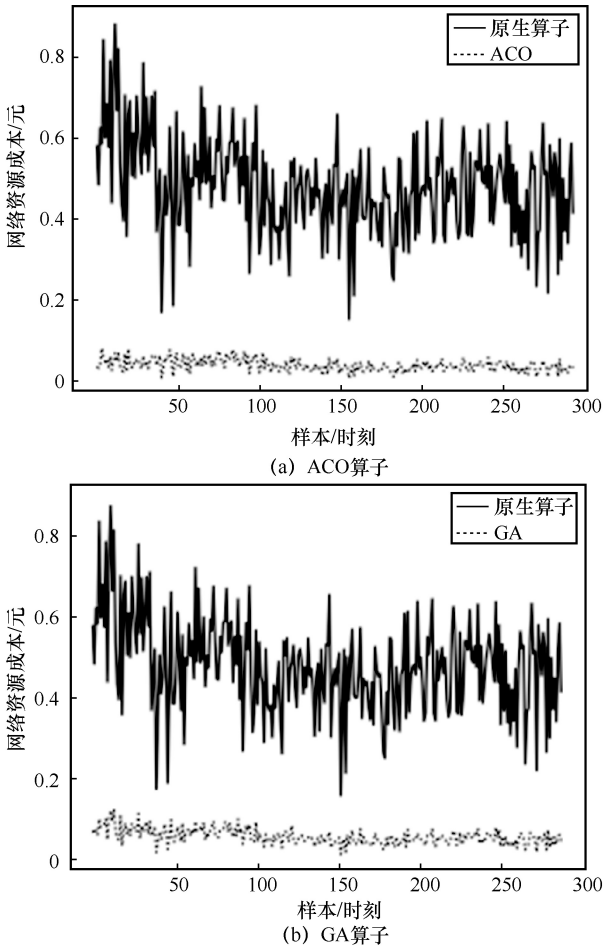

图 7　每个时刻算子优化对比

综合全部样本数据，可得 24 h 成本优化情况，全量样本成本优化对比如图 8 所示，ACO 算子、GA 算子以及 ACO 算子和 GA 算子混合的算子优化率分别是 89.18%、86.85% 和 89.23%，采用 ACO 算子和 GA 算子混合为最佳组合算子。

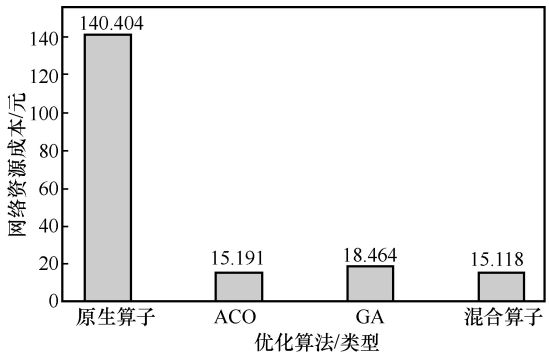

图 8　全量样本成本优化对比

由于本文定义一个时刻单位为 5 min，在限定单位时间内，取 ACO 算子和 GA 算子比例为 5:1 的数量进行模型部署，可取得最佳结果。使用 Kubernetes 编排系统内计算资源的空余水位部署算子，算子的副本数量按照比例根据计算资源空余动态生成。

6　结束语

本文通过对云原生网络特征的采集与分析，基于 Kubernetes 和 eBPF 设计与研发了一套具备根据网络资源的状态调度计算资源、多云多地区负载高可用部署的模型，满足云原生网络成本优化的需求，使得模型初步具备了对以云原生网络为例的云原生应用系统的统一云化承载与运维的系统原型能力。此模型的原型实现证明了该结构的可实施性，云原生技术既能对网络服务灵活管理，也可降低网络资源成本，对多云租户具有较好的经济价值。

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] RIOS J, JHA S, SHWARTZ L. Localizing and explaining faults in micro services using distributed tracing[C]//Proceedings of 2022 IEEE 15th International Conference on Cloud Computing (CLOUD). Piscataway: IEEE Press, 2022: 489-499.

[3] eBFP: extended berkeley packet filter[EB]. 2022.

[4] BE A, MG B, ZZ A. Evaluating and reducing cloud waste and cost—a data-driven case study from Azure workloads[J]. Sustainable Computing: Informatics and Systems, 2022, 35: 100708.

[5] LI W, LI G J, YU X F. A fast traffic classification method based on SDN network[M]. Electronics, Communications and Networks IV.U.S.A: CRC Press, 2015: 223-229.

[6] WANG F, LIU B, ZHANG L J, et al. Dynamic routing and spectrum assignment based on multilayer virtual topology and ant colony optimization in elastic software-defined optical networks[J]. Optical Engineering, 2017, 56(7): 076111.

[7] PARSAEI M R, MOHAMMADI R, JAVIDAN R. A new adaptive traffic engineering method for telesurgery using ACO algorithm over Software Defined Networks[J]. LaRechercheEuropéenneEn Télémédecine, 2017, 6(3/4): 173-180.

[8] WANG J C, DE LAAT C, ZHAO Z M. QoS-aware virtual SDN network planning[C]//Proceedings of 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). Piscataway: IEEE Press, 2017: 644-647.

[9] CAO J Y, ZHANG Y, AN W, et al. VNF placement in hybrid NFV environment: modeling and genetic algorithms[C]//Proceedings of 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). Piscataway: IEEE Press, 2017: 769-777.

[10] CARPIO F, DHAHRI S, JUKAN A. VNF placement with replication for Loac balancing in NFV networks[C]//Proceedings of 2017 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2017: 1-6.

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

[12] YI B, WANG X, LI K, et al. A comprehensive survey of network function virtualization[J]. Computer Networks, 2018(133): 212-262.

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

[14] MAYER A, LORETI P, BRACCIALE L, et al. Performance Monitoring with H^2: hybrid Kernel/eBPF data plane for SRv6 based Hybrid SDN[J]. Computer Networks, 2021(185): 107705.

[15] DRORI I, KHARKAR A, SICKINGER W R, et al. Learning to solve combinatorial optimization problems on real-world graphs in linear time[C]//Proceedings of 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). Piscataway: IEEE Press, 2021: 19-24.

[16] VESSELINOVA N, STEINERT R, PEREZ-RAMIREZ D F, et al. Learning combinatorial optimization on graphs: a survey with applications to networking[J]. IEEE Access, 2020(8): 120388-120416.

[作者简介]

高明（1979– ），男，博士，浙江工商大学信息与电子工程学院副教授、网络系主任，主要研究方向为新型网络体系架构和工业互联网。

刘铭（1997– ），男，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新型网络体系架构和云原生网络。

陈泱婷（1998– ），女，浙江工商大学信息与电子工程学院硕士生，主要研究方向为软件定义网络。

王伟明（1964– ），男，博士，浙江工商大学信息与电子工程学院教授，主要研究方向为新一代网络体系结构和开放可编程网络。