考虑网络时延约束的多场景边缘带宽分配算法研究

重庆大学硕士学位论文

（学术学位）

学生姓名：胡庆弘

指导教师：尚家兴 副教授

学科门类：工　学

学科名称：计算机科学与技术

研究方向：边缘计算

答辩委员会主席：唐亮贵 教　授

授位时间：2023 年 6 月

Research on Multi-Scenario Edge Bandwidth Allocation Algorithms Considering Network Latency Constraint

A thesis submitted to Chongqing University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Qinghong Hu

Supervisor: Associate Professor Jiaxing Shang

June, 2023

摘 要

边缘计算作为一种前沿计算范式，其目的在于向网络边缘的终端用户提供高响应、低延迟的计算、存储和带宽等一系列资源。然而，基于边缘的新型应用程序（例如，现场直播、边缘云游戏、实时 AR/VR 渲染等）通常都是带宽消耗大户，其产生的带宽费用在边缘应用程序提供商的运营成本中占据了很大的比重。同时，边缘基础设施提供商的带宽定价模式也很复杂。因此，如何将边缘用户的带宽需求分配到合适的边缘服务器，以最小化边缘应用程序提供商的货币成本成为了一个重要的问题。

目前，国内外对于边缘带宽分配问题的研究要么基于按需定价的计费模式，要么没有考虑网络时延约束、边缘服务器的成本定价差异等问题。针对上述问题，本文根据带宽需求的分配粒度，对多个场景下的边缘带宽分配算法进行了研究，具体工作如下：

（1）对于带宽需求可分割假设下的边缘带宽分配问题，本文构建了考虑 95 计费模式和网络时延约束的问题模型，提出了 DEBA 算法用于解决该 NP 难问题。该算法主要由两个阶段组成。在第一个阶段，采用基于网络流模型的启发式方法，充分挖掘 95 计费模式的特点，快速地找到一个质量不错的可行初始解。在第二个阶段，基于自适应大邻域搜索算法框架，结合爬山算法和模拟退火算法，对当前解进行持续迭代优化。此外，为了有效地产生邻域解以及验证其可行性，设计了对应的 OptDinic 算法。最后，通过真实边缘带宽需求数据集上的实验验证了 DEBA 算法的有效性。

（2）对于带宽需求不可分割假设下的边缘带宽分配问题，针对高需求和低需求场景，本文提出了相应的 IEBA 算法和 IEBA-L 算法予以解决。首先，本文引入流的概念，将流作为带宽分配调度的最小单位，通过流的迁移交换设计了相应的初始解生成算法和解的优化算法。其次，本文引入边缘服务器成本计算公式，将边缘服务器的开机成本和服务器之间的带宽租用成本差异性考虑在内，并设计了对应的服务器关停算法。最后，在多个数据集上的实验结果表明了 IEBA 算法和 IEBA-L 算法的有效性。

关键词：边缘计算；95 计费模型；边缘带宽分配；邻域搜索；网络时延约束

Abstract

As a promising computing paradigm, edge computing aims at delivering high-response and low-latency computing, storage, and bandwidth resources to end-users at the edge of the network. However, edge-based novel applications (e.g., live broadcast, edge cloud game, real-time AR/VR rendering, etc.) are usually bandwidth-consuming, which has made a considerable contribution to the operating costs of edge application providers. Meanwhile, the bandwidth pricing modes of edge infrastructure providers are also complicated. Therefore, how to allocate the bandwidth demands of edge users to suitable edge servers to minimize the monetary cost of edge application providers becomes an important issue.

Presently, domestic and international researches on edge bandwidth allocation are either based on on-demand pricing, or don't consider the network latency constraint, cost pricing differences of edge servers, and other issues. This thesis has studied edge bandwidth allocation algorithms in multiple scenarios based on the allocation granularity of bandwidth demands in response to the above problems. The main research contents are summarized as follows:

(1) For the edge bandwidth allocation problem assuming that bandwidth demands are divisible, this thesis constructs a problem model that considers the 95th-percentile billing model and network latency constraint, and proposes the DEBA algorithm to solve the NP-hard problem. The algorithm mainly consists of two stages. In the first stage, a heuristic method based on the network flow model is used to fully exploit the characteristics of the 95th-percentile billing model and quickly find a feasible initial solution with good quality. In the second stage, based on the adaptive large neighborhood search algorithm framework, combined with hill climbing algorithm and simulated annealing algorithm, the current solution is continuously and iteratively optimized. In addition, to effectively generate neighborhood solutions and verify their feasibility, a corresponding OptDinic algorithm is designed. Finally, experiments on real edge bandwidth demands datasets demonstrate the effectiveness of the DEBA algorithm.

(2) For the edge bandwidth allocation problem under the assumption of indivisibility of bandwidth demands, this thesis proposes corresponding IEBA and IEBA-L algorithms to solve the high-demand and low-demand scenarios. Firstly, this

thesis introduces the concept of flow, regarded as the minimum bandwidth allocation and scheduling unit. Through the migration and exchange of flows, corresponding initial solution generation algorithm and optimization algorithms are designed. Secondly, this thesis introduces a cost calculation formula for edge servers, considering the power-up cost of edge servers and bandwidth rental cost differences between servers, and designs a corresponding server shutdown algorithm. Finally, experimental results on multiple datasets demonstrate the effectiveness of IEBA and IEBA-L algorithms.

Key words: Edge Computing; 95th-Percentile Billing Mode; Edge Bandwidth Allocation; Neighborhood Search; Network Latency Constraint

目　录

图表目录

1 绪 论

1.1 研究背景与意义

随着现代通信技术和新型物联网（Internet of Things，IoT）技术的飞速发展，如今人们被各种各样的智能移动设备所包围，它们不分昼夜地产生大量的数据，从而带来了利用本地计算资源处理本地数据的需求[1]。为此，多接入边缘计算（Multi-Access Edge Computing，MEC）范式被提出，并已被广泛地用于支持那些计算密集型、延迟敏感[2]和注重隐私保护的应用程序。MEC 的关键思想是将计算、存储和带宽等资源下沉到网络的边缘，也就是更靠近终端用户的位置[3-4]。受益于边缘计算卸载技术，MEC 范式已经成功地支持了各种新型应用程序的实现和繁荣，如极低延迟的现场直播、在线视频游戏的渲染和元宇宙的沉浸式体验等应用。

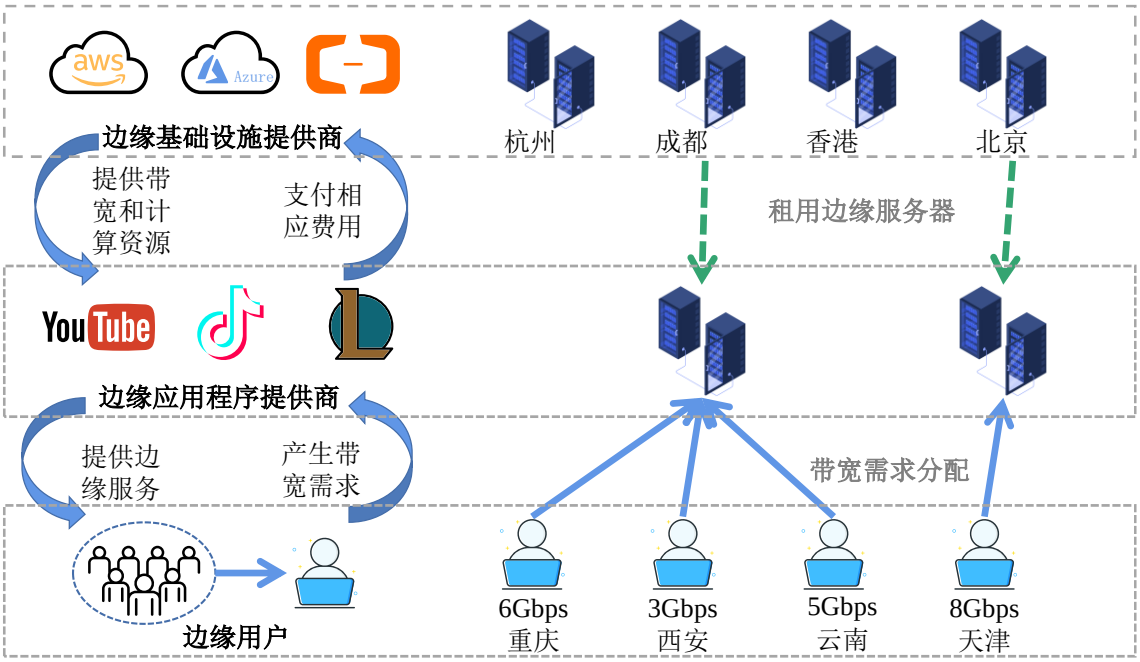


图 1.1 边缘带宽分配问题

Fig.1.1 Edge bandwidth allocation problem

然而，与传统的云计算模式不同，边缘计算通常受到有限资源[5]和高价的限制，特别是带宽资源。根据 Gartner[6]的预测，到 2025 年，带宽开销成本将成为新型边缘计算部署的主要考虑因素，特别是那些内容交付应用程序（如直播、云游戏等）。对于这些边缘应用程序提供商而言，带宽费用在其运营成本中占据了很大的比重。因此，研究如何在满足服务质量（Quality of Service，QoS）约束[7-8]的同时，将边

缘用户的带宽需求分配给合适的边缘服务器来最终达到降低成本的目的，对于这些边缘应用程序提供商而言，具有非常重要的现实意义。

如图 1.1 所示，其中的边缘用户是指位于某一区域内的终端用户群体（例如重庆市内所有使用抖音 App 的用户），其带宽需求为该区域内的所有终端用户在某一时间点上的带宽需求总和。值得注意的是，不同的边缘用户会有不同的带宽需求，而且这些需求会随着时间呈现出明显的周期性波动。边缘应用程序提供商需要选择合适的边缘服务器，并适当地分配边缘用户的带宽需求，从而实现运营成本的最小化和竞争力的有效提升。本文在研究中考虑了边缘云基础设施提供商广泛采用的 95 百分位计费模式，简称 95 计费模式[9-12]。如图 1.2 所示，95 计费模式的计费周期通常为一个月，带宽提供商每 5 分钟对边缘服务器上的实时带宽总和进行一次采样，这些采样点上的值在时间线上会形成一个带宽序列，将这个带宽序列按照数值大小进行升序排序后，取其 95%位置（向上取整）上的带宽值作为该边缘服务器在计费周期内的计费值。在这种模式下，由于忽略了最高的 5%带宽使用量，如果能够适当地分配边缘用户的带宽需求，就可以节省大量的开销成本，这就是边缘带宽分配问题，如例 1.1 所示。

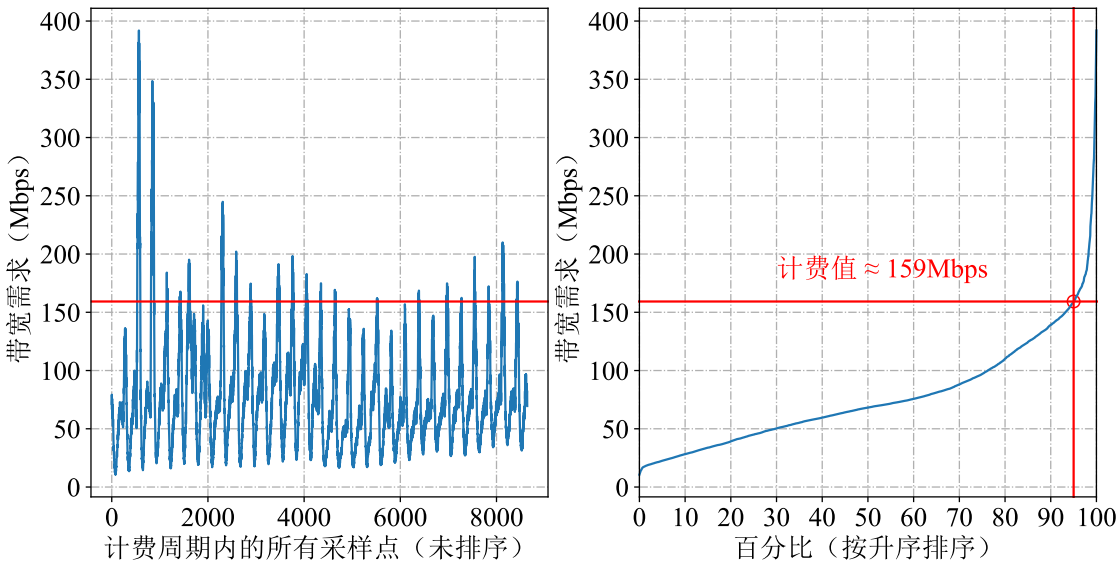


图 1.2 95 计费模式

Fig.1.2 The 95th-percentile billing mode

例 1.1 考虑有 2 个边缘服务器和 1 个边缘用户的场景，边缘用户与边缘服务器之间均连通。假设计费周期为 100 分钟，边缘用户只在第 5 分钟和第 10 分钟有带宽需求，不妨均设为常数 c。此时，若将边缘用户在两个时间段上的带宽需求分配至同一个边缘服务器，则最终的总带宽开销成本为 c。而若通过分配算法，将边

缘用户在两个时间段上的带宽需求分配至不同的边缘服务器，在不考虑服务器开机成本的前提下，最终的总带宽开销成本将降低为 0。

1.2 国内外研究现状分析

对于边缘应用程序提供商而言，他们的目标是最小化支付给边缘基础设施提供商的总带宽开销成本，同时需要保证满足所有边缘用户的带宽需求等一系列约束条件。到目前为止，人们已经对这一问题进行了广泛的研究，其具有重要的现实意义。

1.2.1 计费模式相关研究

对于同一个带宽序列，不同的计费模式最后产生的开销成本可能会有很大的差别。因此，很多学者针对不同的计费模式也做了相关的研究。

Odlyzko[13]提出了固定计费的方法，即按照某一固定周期比如小时、月等单位进行收费，而与用户的使用量无关。但是这种计费模式由于不限制用户的使用量，容易造成资源的浪费，这对基础设施提供商而言并不是一个合理的选择。Altmann 等人[14]提出可以按优先级定价，即带宽的价格与优先级呈正相关，用户可以根据自己的需求进行选择。巴黎地铁定价[15]（Paris Metro Pricing，PMP）也是类似的思路，将网络划分为不同的通道，并在价格和服务质量上予以区分。用户支付的价格越高，他们预期得到的服务质量也相对越好。

但是目前基础设施提供商广泛采用的还是按照使用量计费和按照峰值计费这两种计费模式。Hosanagar 就曾在文献[16]中提到，根据用户在计费周期内的总流量可以按照阶梯计费的模式进行计算。对于按照峰值计费而言，根据计费周期的不同还可以分为按日计费和按月计费等计费模式。而按月计费往往面向的是带宽需求较大、使用频繁的用户，并且可以根据计费点的不同，继续细分为月平均计费、月最大值计费、月 95 峰值计费等。针对平均值计费和最大值计费，也存在很多研究。比如 Adler 等人[17]首次对跨多个网络服务提供商的带宽成本最小化算法进行了研究，并推导出在平均值计费和最大值计费这两种计费模式下的最优离线算法。该离线版本产生的带宽成本下界为实际场景中的分析成本结构提供了重要的依据。接着，他们还在文献[18]中通过将平均值计费和最大值计费这两种计费模式下的在线带宽成本优化问题与滑雪租赁问题相关联，设计了一种确定性算法和一种随机在线算法，可以达到$e/(e+1)$的近似比。

然而，平均值计费无法反映带宽序列的峰值信息，为了应对这些峰值需求，基础设施提供商往往需要随时提供足够大的边缘服务器以供使用，这对他们而言是一笔开销成本；最大值计费又对应用程序提供商不太友好，导致其支出的成本远远大于其使用量。所以作为平均值计费和最大值计费这两种计费模式的折中，

95 计费无论在工业界还是学术界，都被更广泛地使用和研究。Dimitropoulos 等人[19]通过跟踪大量的流量数据，研究观测到 95 计费与采样窗口大小之间存在着依赖性递减的现象。他们基于自相似的网络流量模型为这种依赖现象提供了数学解释证明，并针对窗口大小对 95 计费的影响进行了量化。在 95 计费这种非线性计费模式下，Stanojevic 等人[11]从合作博弈论中引入 Shapley 相关概念，提出一种基于 Shapley 来量化每个用户的成本贡献的方法，并使用蒙特卡洛模拟来对该方法进行逼近。本文的主要研究内容就是基于 95 计费模式，并且本文的主要挑战也是来自于 95 计费这种非线性计费模式。

1.2.2 带宽分配问题相关研究

对于延迟容忍批量（Delay Tolerant Bulk, DTB）数据，例如某科学实验室需要将最新的实验数据推送到世界各地的研究中心进行同步，又例如某数据中心需要进行跨大洲的复制备份，这些数据通常可以接受几小时甚至几天的延迟。由于商业运营商们通常采用的是百分位计费模式，即那些低于计费值的时间段内的数据传输是不计成本的，所以他们更希望能够在非高峰的时间段对这些数据进行传输。然而，由于全球地理位置的差异性，数据传输的源点和汇点的非高峰时间段往往是不同的，这时候就需要对这些数据进行延迟传输。Jain 等人[20]针对时延容忍网络设计了一个路由算法评估框架，提出并评估了几种智能路由算法。这些算法由于考虑了拥塞要素，能够在资源有限的场景中更显现出优势。Wang[21]通过在建立模型时综合考虑 ISP 收取的开销成本和服务延迟惩罚等因素，在不牺牲太多服务质量的前提下，通过选择性地延迟某些流量需求最终达到节省开销成本的目的。他们在离线场景下给出了解决方案，并以此为借鉴给出在线场景下的动态策略。针对在线场景中流量需求与既定模式不匹配的情况，Wang 等人[22]提出了一种实时流量调度算法予以解决。Laoutaris 等人[23]比较了简单的源调度策略 E2E-Sched 和存储转发策略 SnF 在不同的时区差异中各自的性能与优势。结果显示：在时区差异小的情况下，源调度策略 E2E-Sched 因为无需占用额外的存储空间更具优势；在时区差异大的情况下，存储转发策略 SnF 显现出更大的经济优势。Golubchik 等人[24]在允许存在数据传输的短暂延迟下，针对百分位计费模式，提供了离线版本的最优解决方案。同时，他们证明了该问题的在线版本不存在确定性算法，于是提出了一种简单的启发式方法用于解决与之相近的最小化最大值问题。Khare 等人[25]提出了网络成本感知的请求路由算法 NetReq，通过在请求路由中引入端到端的延迟，能够在有效地最小化开销成本的同时，为用户提供良好的网络性能。然而，这些延迟可容忍的方法并不适用于实时场景，因为在这些场景中延迟发送用户数据是不现实、不可接受的。一旦用户因此产生不耐烦的情绪时，该应用程序的产品竞争力和市场地位也会随之下降。

对于不产生服务延迟的带宽分配问题，根据带宽需求的分配粒度，又可分为带宽需求可分割相关问题和带宽需求不可分割相关问题。

对于带宽需求可分割相关问题，可利用最小费用多商品流算法进行求解。比如 Goldenberg 等人[26]从多归属的角度，设计了一系列智能路由算法，在常见的百分位计费模式下结合多种定价函数实现了成本的显著降低。针对离线场景下的可分割问题，他们证明该问题是 NP 难问题，并以最小化总延迟为目标利用最小费用多商品流对问题进行了求解。又比如 Feng 等人[27]提出了 Jetway，在基于百分比的计费模式下，通过解决经典的最小费用多商品流和最大并发流问题，以实时的方式在多个多跳路径上分割和路由视频流，最终实现云服务提供商在数据中心间视频传输的运营成本最小化。

对于跨数据中心的带宽分配问题，除了 Jetway 还存在很多研究。Hong 等人[28]提出了 SWAN，通过软件驱动的方式集中控制业务发送的流量，在不产生拥塞或者中断的同时，提升网络的利用率和公平性。Lin 等人[29]考虑到不同视频流的传输截止日期不同，提出了一种时序驱动的视频流调度系统 EcoFlow，在保证每个视频流满足其传输截止日期的前提下，以不增加链路开销成本为目标对视频流进行了调度，最终降低了跨数据中心视频传输的带宽开销成本。Jalaparti 等人[30]提出了一种结合流量工程和动态定价的广域网网络传输系统 Pretium。他们通过动态定价的方式，使得在需求高的时段优先服务于那些价值更高的请求，而将价值较低的请求转移到利用率低的时段。Li 等人[31]通过遵循“在百分比计费模式中的免费时段中调度更多的峰值流量，在剩余时段中保持较小的流量差异性”这一原则，在无法预知未来流量到达情况的前提下，对跨数据中心的流量传输成本进行了优化。

除此以外，还有很多学者综合考虑了其他因素。Wendell 等人[32]设计了一个分布式系统 DONAR，综合考虑客户端位置、服务器容量、策略偏好等要素，以可拓展、高效的方式为客户提供副本选择策略。Zhang 等人[33]提出了 Entact 方法，与以往只优化成本或者性能中某一指标的研究不同，将成本和性能指标进行联合优化，计算出在线服务提供商的最优成本和性能曲线。Clegg 等人[34]提出了 TARDIS 算法，在时间和空间维度进行了统一，给出了一个关于流量重新分配的解决方案，降低了 ISP 输出成本。Zhan 等人[35]考虑了工作需求的不确定性，建立了一个新的带宽分配问题，即在计费周期前规划用户的带宽使用以实现最大化用户剩余，并通过混合整数规划进行求解。Singh 等人[36]开发了一个高效的在线流量工程系统，CASCARA。他们利用非线性定价机制优化域间带宽分配，在不显著增加客户端延迟的情况下最小化成本，最终达到成本和性能之间的权衡。陈寰[37]在流量分配的过程中考虑分配粒度、分配偏差等复杂要素，在离线场景下提出了 PeCo-N 和 PeCo-Huffman 方法以及在在线场景下提出 OnTPC 方法。Yang 等人[38]针对云计算

中的许多场景，包括内容分发网络、视频直播网络、实时通信网络和云广域网，提出了各自的网络带宽分配问题，并将它们形式化为整数规划问题，为下一步的科研提供了基础。

对于带宽需求不可分割相关问题，Goldenberg 等人[26]设计了在线整数分配算法 GIA 用于解决在线场景中的不可分割问题。You 等人[39]针对 q 百分位计费模式下的跨域任务调度问题，将其转化为了考虑截止时间的资源分配与装箱问题，采用进化算法实现总成本的降低。Xu 等人[40]通过随机抽样的方式对松弛后的连续凸优化问题进行了求解。此外，对于需求不可分割相关问题，通过放松相关约束可以转化为广义指派问题。如 Chu 等人[41]通过将遗传算法应用于广义指派问题的求解中，巧妙地设计出用数组来满足物品只能放置于一个背包这一约束条件。Yagiura 等人[42-43]分别利用变邻域搜索和禁忌搜索两种方式对解进行了优化。

虽然带宽分配相关问题已经被广泛的研究，但是现有的工作在对问题进行建模的时候，并没有将网络时延约束考虑在内，即他们假设所有的边缘用户与边缘服务器之间都是连通可用的。但是在现实场景中，由于地理分布的差异性，用户与服务器之间存在着不同的网络时延，应用程序提供商为了保证实时性和服务质量，将位于广西的某个用户的带宽需求分配给位于北京的服务器这是不现实的。

1.3 论文主要研究内容

本文针对现有带宽分配方法存在的不足开展系统研究，具体工作包括以下两方面：

首先，对于带宽需求可分割假设下的边缘带宽分配问题，本文建立了考虑 95 计费模式和网络时延约束的问题模型，提出了 DEBA（an algorithm for Divisible Edge Bandwidth Allocation problem）算法用于解决该 NP 难问题。该算法主要包括两个阶段。在第一个阶段，采用基于网络流模型的启发式方法，充分挖掘 95 计费模式的特点，快速地找到一个质量不错的可行初始解。在第二个阶段，基于自适应大邻域搜索算法框架，结合爬山算法和模拟退火算法，对当前解进行持续迭代优化。此外，为了有效地产生邻域解以及验证其可行性，本文设计了对应的 OptDinic 算法。通过真实边缘带宽需求数据集上的实验验证了 DEBA 算法的有效性。

其次，对于带宽需求不可分割假设下的边缘带宽分配问题，本文同样建立了对应的模型，同时引入更加复杂的边缘服务器成本计算公式来贴近真实的应用场景，并提出了相应的 IEBA（an algorithm for Indivisible Edge Bandwidth Allocation problem）算法对该 NP 难问题进行了解决。针对带宽需求不可分割这一特点，本文引入了流的概念，通过流的迁移交换，得到一个可行初始解。接着，分别设计了适合该场景的爬山算法、模拟退火算法和自适应大邻域搜索算法，对解进行持

续迭代优化。同时由于边缘服务器存在开机成本，本文还提出 IEBA-L（an algorithm for Indivisible Edge Bandwidth Allocation problem with Low-demand）算法，通过在 IEBA 算法中结合服务器关停算法，在带宽需求较低的场景中实现总成本的进一步优化。本文基于原始真实边缘带宽需求数据集生成了多个数据集，在多个数据集上的实验结果表明了 IEBA 算法和 IEBA-L 算法在各自场景下的有效性。

本文的主要贡献为：

（1）与传统按需定价的计费模式不同，本文针对非线性 95 计费模式下的边缘带宽分配问题开展研究，该问题更具复杂性，也更加贴近现实。

（2）在本文的模型中，通过引入网络时延约束条件，体现边缘用户与边缘服务器之间因通信距离不同所产生的网络时延差异。此外，通过引入更加复杂的边缘服务器成本计算公式，将边缘服务器的开机成本考虑了进来，并且体现出不同边缘服务器之间的带宽租用成本差异性，更加贴近真实应用场景。

（3）根据带宽需求能否分割的特点，本文设计了相应的边缘带宽分配算法分别予以解决。在算法的优化步骤，通过引入爬山算法、模拟退火算法、自适应大邻域搜索算法，达到了加快优化收敛速度、避免陷入局部最优和拓宽搜索空间的目的。

（4）在真实边缘带宽需求数据集上进行了一系列实验，结果表明本文提出的三种边缘带宽分配算法 DEBA、IEBA、IEBA-L 能够在各自的场景中为边缘应用程序提供商节省很大一笔带宽开销成本。

1.4 论文组织结构

本文共分为五个章节，各章节的内容如下：

第一章：绪论。本章首先介绍了边缘带宽分配问题的研究背景与意义，阐明了该研究问题的现实重要性。然后总结了该问题的国内外研究现状以及现有工作存在的不足之处。最后简述了本文的主要研究内容及组织结构。

第二章：相关理论及技术。本章详细介绍了本文所涉及到的理论基础和算法模型，包括网络流模型及其算法，几种常见的邻域搜索优化算法比如爬山算法、模拟退火算法、自适应大邻域搜索算法。

第三章：基于带宽需求可分割假设下的边缘带宽分配问题。本章首先对考虑 95 计费模式和网络时延约束的边缘带宽分配问题给出形式化定义，并提出了 DEBA 算法。然后本章针对 DEBA 算法流程中的各个环节进行了详细介绍。最后本章在真实边缘带宽需求数据集上进行了实验，验证了该算法的有效性。

第四章：基于带宽需求不可分割假设下的边缘带宽分配问题。本章首先建立了新型的边缘带宽分配问题的数学模型，在第三章的模型基础上引入边缘服务器

成本计算公式，更加贴近现实应用场景。针对带宽需求不可分割这一新场景，本章提出对应的 IEBA 算法予以解决，并对其算法细节进行了阐述。针对服务器存在开机成本的情况，本章设计了服务器关停算法，将其与 IEBA 算法结合，提出了适合低带宽需求场景的 IEBA-L 算法。基于原始真实边缘带宽需求数据集，本章生成了多个数据集，在这多个数据集上的实验结果表明了 IEBA 算法和 IEBA-L 算法在各自场景下的有效性。

第五章：总结与展望。本章对本文现有的工作内容进行了总结，并对后续的研究方向进行了展望。

2 相关理论及技术

2.1 网络流模型及其算法

线性规划问题作为一种最常见的最优化问题，在现实生活中得到广泛的应用，其特点是目标函数和约束条件都是线性的[44]。其中，某些线性规划问题可以转化为网络流模型，通过网络流相关算法进行高效地求解。

网络流包括两个重要的组成部分，网络和流。

网络，也叫做有向图。给定一个有向图$G=(V,E)$，若$(u,v)\in E$，则$(v,u)\notin E$，即图中不存在反方向的边，并且图中不允许自环的存在。图中的每一条边都有一个非负的权重$c(u,v)\geq 0$，代表容量值。如果图中不存在某一条边，则该条边上的容量值定义为0。在网络的所有节点中，存在两类特殊的节点，定义为源节点s和汇节点t。顾名思义，源节点的入度为 0，汇节点的出度为 0。如果该网络是连通的，则对于每个节点$v\in V$，其都在从源节点到汇节点的某条路径上。

流，类比于水流。假如把网络想象成一个城市的自来水管道布局，那么流就是某一根水管中流动的水。现给出流的形式化定义：给定一个网络$G=(V,E)$，其容量函数为c， s是网络的源节点，t是网络的汇节点，则G 中的流是一个实值函数$f:V\times V\to R$，其满足如下两条性质[45]：

（1）容量限制：$\forall u,v\in V,0\leq f(u,v)\leq c(u,v)$。

（2）流量守恒：$\forall u\in\{V-s-t\},\sum_{v\in V}f(v,u)=\sum_{v\in V}f(u,v)$。当图中不存在某一条边时，定义该边上的流为 0。所以该条性质可以非形式化地称为“流入等于流出”。

网络流模型主要分为三种：最大流、最小割和费用流。由于本文中只用到了最大流，所以本章暂时只介绍最大流模型。

最大流是指在给定的网络G 中，找到一个值最大的流。而流的值指的是从源节点流出的总流量，等价于向汇节点流入的总流量。

解决最大流问题最常见的方法是 Ford-Fulkerson 方法，其包含几种时间效率不同的算法，比如 FORD-FULKERSON 算法[46]、Edmonds-Karp 算法[47]和 Dinic 算法[48]等。但是它们的思想都是在图G 关联的残存网络G_f 中不断地寻找增广路径，直到找不到为止。下面给出残存网络和增广路径的定义：

残存网络：给定一个网络G 和一个流f，由f 诱导产生的残存网络记为$G_f=(V,E_f)$，其中$E_f=\{(u,v)\in V\times V:c_f(u,v)\geq 0\}$。

上式中的$c_f(u,v)$代表边上的残存容量，其定义如下：

$$c_f(u,v)=\begin{cases} c(u,v)-f(u,v), & (u,v)\in E \\ f(v,u), & (v,u)\in E \\ 0, & 其他 \end{cases} \tag{2.1}$$

公式（2.1）表明，对于原始网络G中的一条边，其原始边上的残存容量等于该条边上的容量减去该条边上的流量，并增加一条反向边，使得反向边上的残存容量等于原始边上的流量大小。

增广路径：给定一个残存网络G_f，从源节点s到汇节点t的一条路径p。对于增广路径上的每一条边，其流量增加的最大幅度即为该条边上的残存容量大小。因此，定义增广路径的残存容量为能够为该路径上所有边增加的最大流量，形式化表述如下：

$$c_f(p)=min\{c_f(u,v):(u,v)\in p\} \tag{2.2}$$

对于 FORD-FULKERSON 算法，其利用深度优先搜索不断地在残存网络中寻找一条增广路径。因为每次找到一条增广路径后，其流量至少增加 1 个单位，假设f^*代表网络中的一个最大流，那么最多循环$|f^*|$次搜索。对于每次循环，利用深度优先搜索找一条增广路径的时间复杂度为$O(V+E)=O(E)$，所以 FORD-FULKERSON 算法的时间复杂度是$O(E|f^*|)$。

对于 Edmonds-Karp 算法，其每次通过广度优先搜索来寻找一条增广路径。因为深度优先搜索存在绕远路的问题，而广度优先搜索能够保证每次找到的是最短的一条路径，其时间复杂度已被证明为$O(VE^2)$。

目前使用最多的当属 Dinic 算法，首先利用广度优先搜索得到残存网络的分层图，然后利用深度优先搜索寻找增广路径。由于其使用了多路增广和当前弧优化这两种优化手段，提升了运行的效率，其时间复杂度是$O(V^2E)$。多路增广，也就是在某一个节点找到一条增广路径时，如果该节点还有流量剩余，则继续寻找从该节点出发的剩余的增广路径，直到该节点剩余的流量为 0 或者已经遍历完所有的增广路径。通过这一操作可以节省从源节点到该节点之间很多不必要的搜索过程。当前弧优化，这里的弧即是边。原本对于每个节点都有一个从该节点出发的边的集合，每次遍历到某个节点时，都需要从头遍历整个集合。但是无论是成功增广还是失败，对于访问过的边，以后都无需再次访问。所以针对每个节点，通过保存已经访问过的集合下标，下次再次遍历到该节点时就无需从头开始遍历了。通过这一操作可以节省从该节点到汇节点之间很多不必要的搜索过程。因此，本文在第三章所使用的最大流算法也是基于 Dinic 算法来实现的。

2.2 邻域搜索算法

邻域搜索算法，又称为局部搜索算法，通过每次迭代地对当前解进行一个操作后得到一个新解，从而期望对解的质量进行改善。邻域搜索算法的种类有很多，因为本文只用到了爬山算法、模拟退火算法和自适应大邻域搜索算法，因此本章只对这三种邻域搜索算法进行介绍。

2.2.1 爬山算法

爬山算法作为一种简单的邻域搜索算法，其主要思想是：针对一个当前解，每次在其邻域解空间中寻找一个最优解，将这个最优解跟当前解进行比较，如果比当前解要优，则把该最优解赋值给当前解并继续循环迭代，否则停止搜索的过程。

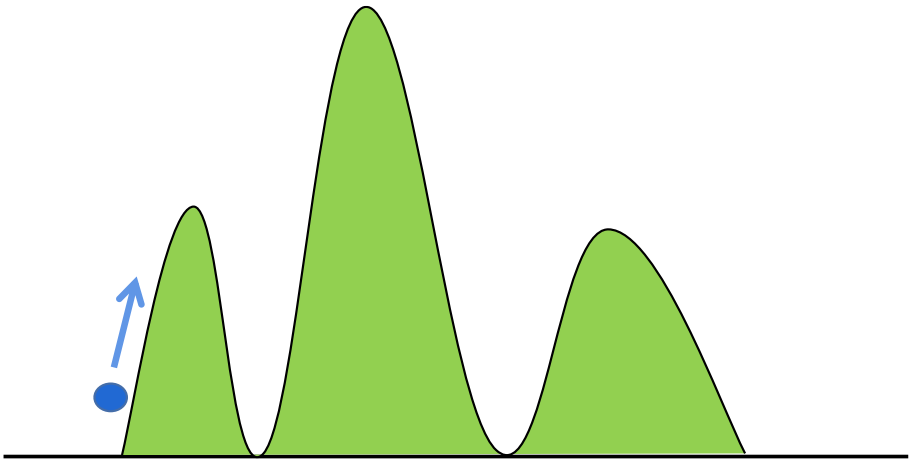


图 2.1 爬山算法

Fig.2.1 Hill climbing algorithm

如图 2.1 所示，爬山算法对初始解的要求极高，因为一旦选择不当就很容易陷入局部最优的情况，导致不能搜索到全局最优解，这也是爬山算法的主要缺点。

2.2.2 模拟退火算法

模拟退火算法是模拟固体退火的过程，将固体升温到一个较高的温度，粒子在高温情况下能量较高，无序性增大，可以进行自由地运动和排列组合。而当温度下降时，粒子的能量、运动的幅度也会随之降低，并逐渐趋于有序，最后在常温下达到基态[49]。其算法的主要流程如下：

（1）从一个较高的初始温度开始，生成一个随机的初始解 S 作为当前解。

（2）在每个温度下，进行多轮搜索。

（3）对于每一轮搜索，对当前解添加随机扰动后得到一个新的邻域解 S'。

（4）根据目标函数 f 计算新解的目标函数值和旧解的目标函数值。如果新解的目标函数值优于旧解，则接受新解。否则，根据 Metropolis 准则以一定的概率接受新解。

（5）继续（3）-（4）步，直到达到搜索的轮数。

（6）判断是否达到终止条件，如果满足则终止算法流程。否则，缓慢降低当前温度，跳转到第（2）步继续进行。

如图 2.2 所示，当前解是一个局部最优解，如果能够接受一个更差的新解，那么之后就可能找到一个全局最优解。这也是模拟退火算法的一个主要优点，能够跳出局部最优。但是模拟退火算法也存在缺点，比如为了找到全局最优解，那么搜索的范围就要足够大，这就要求初始温度设置得足够高，在每个温度下搜索的轮数足够多，温度衰减得足够慢等等，这些都将导致算法的搜索时间大幅延长。

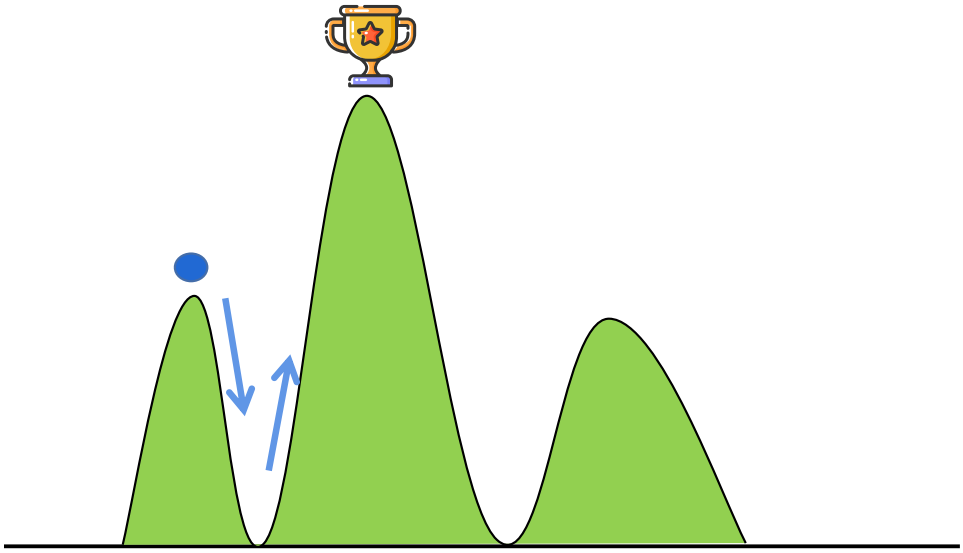


图 2.2 模拟退火算法

Fig.2.2 Simulated annealing algorithm

2.2.3 自适应大邻域搜索算法

自适应大邻域搜索算法是由 Ropke 等人[50]在 2006 年提出的一种优化算法。根据其字面意思，它主要包含两个特点：一是自适应，二是大邻域。自适应，是由于其内部在生成邻域解时是通过多个破坏和修复算子实现的，而破坏和修复算子的选择又是根据算子的历史表现自动进行的。这里的算子可以理解为一种策略或者一种规则。大邻域，指的是邻域搜索的空间范围很大，由于其采用了多种搜索策略，也就是多个算子，根据乘法规则，其邻域搜索空间自然很大。

自适应大邻域搜索算法的主要流程如下所示：

（1）输入一个初始解，将初始解赋值给当前解，并将最优解设置为当前解。输入多个候选破坏算子、修复算子，对于每一种算子，其初始权重均设置为相同值。

（2）根据算子的权重值，随机挑选出一个破坏算子和一个修复算子。

（3）根据这个破坏算子和修复算子，分别对当前解进行破坏操作和修复操作，从而产生一个新的邻域解。

（4）计算当前解和邻域解的目标函数值。

（5）更新当前解：如果这个邻域解满足接受规则，就接受该邻域解作为当前解。

（6）更新最优解：将当前解和最优解进行比较，如果当前解优于最优解，则将最优解更新为当前解。

（7）更新算子的得分和算子的权重。

（8）如果达到终止条件则终止算法流程返回最优解，否则跳转到第（2）步继续进行。终止条件可根据实际情况自行定义，比如：当前的温度小于设置的最低温度、超时退出、最优解没有更新达到一定的轮次等。

在算法流程的第（2）步，每个算子被选中的概率可以通过轮盘赌法的方式来进行计算。以修复算子为例，假设 Ω^+ 代表修复算子的集合，ρ^+ 代表修复算子的权重，那么对于某个修复算子 Ω_i^+，其被选中的概率 w_i^+ 的计算公式如下：

$$w_i^+ = \frac{\rho_i^+}{\sum_{j=1}^{|\Omega^+|}\rho_j^+} \tag{2.3}$$

由公式（2.3）可以看出，一个算子被选中的概率与其权重成正比。

在算法流程的第（5）步，对于新的邻域解的接受规则可以自行定义。比如只有当邻域解比当前解还要优时才接受；或者类似于模拟退火的思想可以以一定的概率去接受比当前解要差的邻域解，接受的概率与当前的温度、目标函数之间的差值有关。

在算法流程的第（7）步，首先计算算子当前轮次的得分 Ψ。若该轮次算子未被选中时，其得分记为 ω_4；若该轮次算子被选中时，其更新规则如下：

$$\Psi = max\begin{cases}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{cases} \tag{2.4}$$

其中 ω_1、ω_2、ω_3、ω_4 都是自定义的非负参数，并且满足 $\omega_1 \geq \omega_2 \geq \omega_3 \geq \omega_4$。当邻域解比最优解还要优时，将算子的得分赋值为 ω_1；当邻域解比当前解要优时，

将算子得分赋值为ω_2；当邻域解被接受时，即此时当前解被更新为邻域解，将算子的得分赋值为ω_3；当邻域解被拒绝时，将算子的得分赋值为ω_4。

假设算子的权重衰减系数为$\sigma \in [0,1]$，则最终算子权重的更新公式如下所示：

$$\rho = \sigma \times \rho + (1-\sigma) \times \Psi \tag{2.5}$$

2.3 本章小结

本章对本文所涉及到的相关理论及其技术进行了介绍。首先，介绍了网络流模型及其算法，这将在本文的第三章得到使用。当带宽需求可分割时，本文通过最大流算法生成可行初始解、邻域解以及验证邻域解的可行性，这将在后续的章节详细地介绍。最后，介绍了三种邻域搜索算法的流程步骤，包括爬山算法、模拟退火算法以及自适应大邻域搜索算法。这些邻域搜索算法将会在本文的第三章和第四章中解的优化部分得到使用。

3 基于带宽需求可分割假设下的边缘带宽分配问题

3.1 概述

在满足边缘用户的需求的前提下，通过对带宽的合理分配调度，最小化网络带宽的使用成本，这对于边缘应用程序提供商而言，是其提升竞争力的关键，具有重要的现实意义。为了保证服务质量，通常不允许出现过高的网络时延。然而现有的研究并未考虑因地理位置、通信距离等对网络时延造成的影响，这就会导致将一个边缘用户的需求分配到一个距离很远的边缘服务器上进行处理，这种分配调度方案显然不能够应用于现实应用场景。为了解决上述问题，本章构建了考虑 95 计费模式和网络时延约束的问题模型，在带宽需求可分割假设下，提出了 DEBA 算法用于解决该 NP 难问题。该算法主要由两个阶段组成。在第一个阶段，基于网络流模型的启发式方法，充分挖掘 95 计费模式的特点，快速地找到一个质量不错的可行初始解。在第二个阶段，基于自适应大邻域搜索算法框架，结合爬山算法和模拟退火算法，对当前解进行持续迭代优化。此外，为了有效地产生邻域解以及验证其可行性，设计了对应的 OptDinic 算法。最后，在真实边缘带宽需求数据集上进行了实验，结果表明了 DEBA 算法的有效性。

3.2 系统模型和问题定义

本节首先建立了考虑 95 计费模式和网络时延约束的问题模型，然后给出了其相应的形式化定义。表 3.1 列出了本章在研究中常用到的一些符号，并给出其描述。

表 3.1 符号列表

Table 3.1 List of notations

符号	描述
t_k	T 中第 k 个时间段，也是第 k 个采样点。
T	采样点的集合，即整个计费周期。
s_i	S 中第 i 个边缘服务器。
S	边缘服务器的集合。
m	边缘服务器的数量。
b_i	s_i 的带宽容量上限。
B	所有边缘服务器的带宽容量上限的集合。
u_j	U 中第 j 个边缘用户。
U	边缘用户的集合。

表 3.1 符号列表（续）

Table 3.1 List of notations(continued)

符号	描述
n	边缘用户的数量。
y_j^k	u_j 在 t_k 处的带宽需求。
Y	所有边缘用户在整个计费周期 T 内的带宽需求矩阵。
l_{ij}	s_i 和 u_j 之间的网络时延。
L	所有边缘服务器和所有边缘用户之间的网络时延矩阵。
τ	边缘服务器向边缘用户提供服务时需要满足的网络时延上限。
x_{ij}^k	在 t_k 处， s_i 向 u_j 分配的带宽数量。
X	整个计费周期内的带宽分配矩阵，也就是最后的解决方案。
w_i^k	在 t_k 处， s_i 从所有边缘用户接收的带宽需求总和。
W_i	s_i 在整个计费周期内的带宽分配序列。
$O_{95}(A)$	将序列 A 按升序进行排列后，取其 95%位置（向上取整）上的值。
q_i	s_i 在整个计费周期内的计费值。
p	边缘服务器提供的带宽的单位成本（元/Mbps）。
C	整个计费周期内所有边缘服务器的总带宽开销成本。
Θ	模拟退火算法的初始温度。
Γ	模拟退火算法中的外层循环迭代轮数。
Φ	模拟退火算法中的内层循环迭代轮数。
α	模拟退火算法中的温度衰减系数。
λ	邻域搜索过程中执行模拟退火算法的频率。
ω_1	当邻域解优于最优解时，算子的得分。
ω_2	当邻域解优于当前解时，算子的得分。
ω_3	当邻域解被接受时，算子的得分。
ω_4	当邻域解被拒绝时，算子的得分。
σ	算子的权重衰减系数。

3.2.1 系统模型

在 95 计费模式下，对于每个使用中的边缘服务器，计费周期 T 通常为一个月，带宽使用量每 5 分钟采样一次，如图 1.2 所示。因此，在一个为期 30 天的计费周期 T 内，共有 8640 个采样点。假设一共有 m 个边缘服务器可供边缘应用程序提供商租用。本章使用 $S=\{s_1,s_2,\ldots,s_m\}$ 来表示边缘服务器的集合，b_i 是第 i 个服务器在每个采样的时间段中可以提供的带宽容量上限，即：在每个时间段，每个服务器

接收到的带宽需求总和不能超过其带宽容量上限。由于 S 中的边缘服务器可以分布在不同的地理位置上，因此它们为同一边缘用户提供服务时可能会存在不同的网络通信延迟。

本章使用 $U=\{u_1,u_2,\ldots,u_n\}$ 来表示边缘用户的集合，其中 u_j 是第 j 个边缘用户，指的是位于某一区域内的终端用户群体。本章使用 l_{ij} 来表示边缘服务器 s_i 和边缘用户 u_j 之间的网络时延。在第 k 个采样点 t_k，使用 y_j^k 来表示边缘用户 u_j 的带宽需求。为了保证服务质量，边缘用户 u_j 只和那些网络通信延迟小于 τ 的边缘服务器保持连通。本章用 x_{ij}^k 来表示在第 k 个采样点 t_k，s_i 分配给 u_j 的带宽数量。因此，s_i 在 t_k 处接收到的带宽需求总和可以通过如下公式进行计算：

$$w_i^k=\sum_{j=1}^{n}x_{ij}^k \tag{3.1}$$

由此形成的有关 s_i 的带宽序列可表示为 $W_i=\left\{w_i^1,w_i^2,\ldots,w_i^{|T|}\right\}$。此时，该计费周期内 s_i 的计费值可以表示为：

$$q_i=O_{95}\left(W_i\right) \tag{3.2}$$

所以，在该计费周期内，边缘应用程序提供商为了满足所有边缘用户的带宽需求，租用边缘服务器所花费的总带宽开销成本的计算公式如下：

$$C=\sum_{i=1}^{m}q_i\times p \tag{3.3}$$

3.2.2 问题定义

基于上一节提到的相关公式，本节所提出的边缘带宽分配问题可以形式化地定义如下：

$$\begin{aligned}&min\ C\\&s.t.\ \ \sum_{i=1}^{m}x_{ij}^k=y_j^k\\&\quad\sum_{j=1}^{n}x_{ij}^k\le b_i\\&\quad l_{ij}<\tau,\quad \forall x_{ij}^k>0\\&\quad x_{ij}^k\in N,\ i\in[1,m],\ j\in[1,n],\ k\in[1,|T|]\end{aligned} \tag{3.4}$$

其中，公式（3.4）中的第一行表明这个问题的优化目标是最小化边缘应用程序提供商的总带宽开销成本；第二行表明需要满足每个边缘用户在每个时段上的带宽需求；第三行表明每个边缘服务器在每个时段接收到的总带宽需求不能大于其带宽容量上限；第四行是网络时延约束，即对于每个边缘用户只有那些网络时延小于预设上限值的边缘服务器才能对其提供服务。

对于 95 计费下的优化问题，Jalaparti 等人[30]已经证明了这是 NP 难问题。类似的，对于本节所提出的问题，如果将 95 计费模式松弛到百分之一百计费模式（取带宽序列的最大值，也就是最大值计费模式），此时讨论的问题就退化成了整数线

性规划问题，这是一个众所周知的 NP 难问题[51]。因此，本节所提出的边缘带宽分配问题也是一个 NP 难问题。

3.3 DEBA 算法

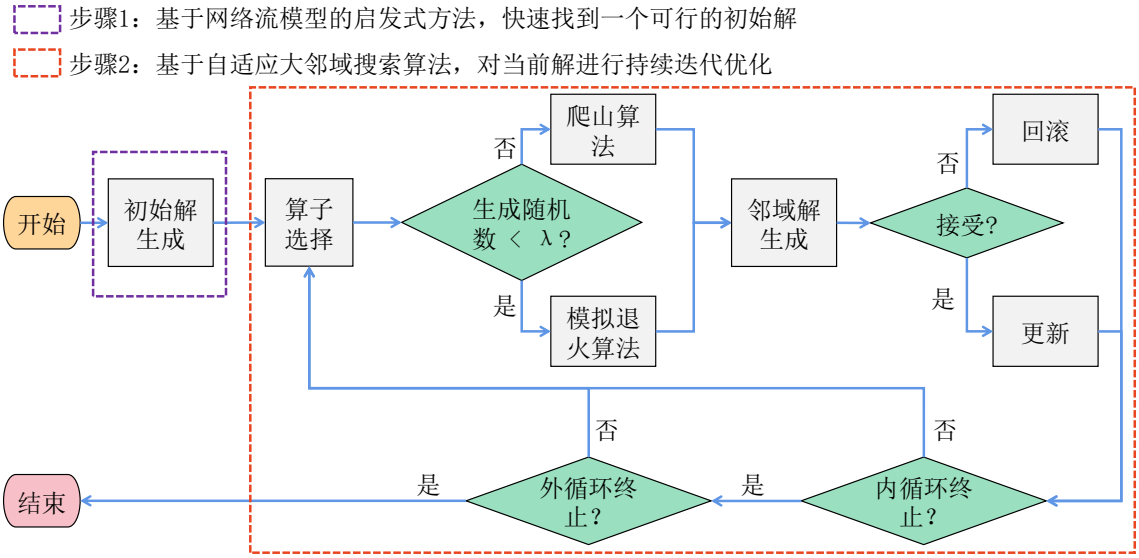


图 3.1 DEBA 算法的执行过程

Fig.3.1 The process of the proposed DEBA algorithm

为了解决上述问题，本节提出了 DEBA 算法，其流程步骤如图 3.1 所示。在第一个阶段，采用基于网络流模型的启发式方法，充分挖掘 95 计费模式的特点，快速地找到一个质量不错的可行初始解。在第二个阶段，基于自适应大邻域搜索算法，结合爬山算法和模拟退火算法，对当前解进行持续迭代优化。

3.3.1 基于网络流的可行初始解生成算法

对于 DEBA 算法的第一阶段，本节设计了一种基于网络流的可行初始解生成算法，简称为 FISG（Feasible Initial Solution Generation algorithm），如算法 3.1 所示。该算法主要包括两个步骤：（1）针对每个边缘服务器的带宽序列中的免费时段，为其尽可能多地分配带宽需求。这里设计了三种排序策略，分别是边缘服务器和边缘用户按照其连通性升序排序，而带宽序列按照剩余带宽需求总和降序排序。因为连通性低的边缘服务器或者边缘用户在后续的分配过程中更不容易满足条件，所以需要提前进行分配。并且出于节省带宽开销成本的考虑，那些带宽需求高的时段更适合作为带宽序列中的免费时段（如第 2-11 行所示）；（2）对于每个采样点上剩余的用户带宽需求，通过利用历史已分配信息获取每个边缘服务器在不增加成本的情况下在当前采样点上能够分配的最大带宽容量，并利用 Dinic 算法

来尝试解决，这是一种高效的网络流算法用于寻找给定图上的最大流。如果 Dinic 算法找到的最大流不等于当前采样点上所有用户的带宽需求总和，说明此时分配的带宽容量无法满足所有用户的带宽需求，则在再次尝试之前将所有边缘服务器的带宽容量提升至其容量上限（如果已成功，此步骤没有影响，如第 12-20 行所示）。该算法的时间复杂度为$O\left(|T|nm\left(n+m\right)^2\right)$。

算法 3.1：基于网络流的可行初始解生成算法 *FISG*

输入：计费周期T，边缘服务器集合S，边缘用户集合U，带宽容量上限集合B，带宽需求矩阵Y，网络时延矩阵L，网络时延上限τ，带宽单位成本p

输出：带宽需求分配矩阵X，总开销成本C

1 $\hat{Y} \leftarrow Y$；$f \leftarrow |T| - \lceil |T| \times 0.95 \rceil$；

2 按连通性将S进行升序排序；

3 **for** $i \leftarrow 1$ *to* m **do**

4 　按剩余带宽需求总和将带宽序列进行降序排序；

5 　**for** $k \leftarrow 1$ *to* f **do**

6 　　$h \leftarrow b_i$；

7 　　按连通性将U进行升序排序；

8 　　**for** $j \leftarrow 1$ *to* n **do**

9 　　　**if** $h = 0$ **then** break；

10 　　　**else if** $l_{ij} < \tau$ **then**

11 　　　　$x_{ij}^k \leftarrow min\left(y_{ij}^k, h\right)$；$y_{ij}^k \leftarrow y_{ij}^k - x_{ij}^k$；$h \leftarrow h - x_{ij}^k$；

12 **for** $k \leftarrow 1$ *to* $|T|$ **do**

13 　$H \leftarrow \varnothing$；

14 　**for** $i \leftarrow 1$ *to* m **do**

15 　　**if** t_k 是 s_i 的一个免费时段 **then** $h_i \leftarrow b_i$；

16 　　**else** $h_i \leftarrow$ 通过公式（3.2）计算其当前的计费值；

17 　　$H \leftarrow H \cup \{h_i\}$；

18 　$\hat{X} \leftarrow Dinic\left(\hat{Y}, H, X, k\right)$；

19 　$H \leftarrow B$；

20 　$X \leftarrow Dinic\left(\hat{Y}, H, \hat{X}, k\right)$；

21 $C \leftarrow$ 通过公式（3.3）计算其总开销成本；

22 **return** X, C；

3.3.2 基于自适应大邻域搜索的持续迭代优化算法

显然，由上一节获得的初始解可以满足公式（3.4）中的约束条件。但是，通过负载的整合，可以进一步缩减总带宽开销成本。因此，在 DEBA 算法的第二阶段，进一步设计了一种基于自适应大邻域搜索的算法来对当前解进行持续迭代优化。自适应大邻域搜索是一种启发式搜索方法，它将算子（破坏算子和修复算子）的自适应选择这一特征引入到了传统的局部搜索中，扩展了搜索空间。同时，为了加快优化的收敛速度和避免陷入局部最优解，本节还引入了基于爬山的优化算法和基于模拟退火的优化算法。

算法 3.2：基于自适应大邻域搜索的持续迭代优化算法

输入：初始解 X，初始开销成本 C，初始温度 Θ，外层循环迭代轮数 Γ，内层循环迭代轮数 Φ，温度衰减系数 α，执行模拟退火算法的频率 λ

输出：最优解 $\bar{X}$，最优开销成本 $\bar{C}$

1 $\hat{X},\hat{C} \leftarrow X,C$；//当前解
2 $\bar{X},\bar{C} \leftarrow \hat{X},\hat{C}$；//最优解
3 $\rho^- \leftarrow (1,1,\ldots,1)$；$\rho^+ \leftarrow (1,1,\ldots,1)$；
4 **for** $e \leftarrow 1$ *to* Γ **do**
5 　**for** $g \leftarrow 1$ *to* Φ **do**
6 　　根据公式（2.3）分别选出一个破坏算子 $d \in \Omega^-$ 和一个修复算子 $r \in \Omega^+$；
7 　　$\delta \leftarrow random(0,1)$；
8 　　**if** $\delta \geq \lambda$ **then** $\hat{X},\hat{C} \leftarrow HCO(\hat{X},\hat{C},d,r,\Theta)$；//基于爬山的优化算法
9 　　**else** $\hat{X},\hat{C} \leftarrow SAO(\hat{X},\hat{C},d,r,\Theta)$；//基于模拟退火的优化算法
10 　　**if** $\hat{C} < \bar{C}$ **then** $\bar{X},\bar{C} \leftarrow \hat{X},\hat{C}$；
11 　　根据公式（2.4）获取算子得分，并根据公式（2.5）更新 ρ^- 和 ρ^+；
12 　$\Theta \leftarrow \Theta \times \alpha$；
13 **return** $\bar{X},\bar{C}$；

如算法 3.2 所示，基于自适应大邻域搜索的持续迭代优化算法主要包含如下步骤：

（1）初始化。将初始解复制给当前解和最优解，将所有候选破坏算子和修复算子的初始权重设置为 1（如第 1-3 行所示）。

（2）在每个温度下，进行多轮搜索。对于每一轮搜索，以轮盘赌法的方式从候选破坏算子集合和修复算子集合中分别选出一种破坏算子和一种修复算子。本文使用一个超参数 λ 来决定两种邻域搜索算法（即模拟退火算法和爬山算法）的

运行比例。当生成的随机数小于λ时，选择基于模拟退火的优化算法产生一个新的邻域解，否则选择基于爬山的优化算法产生一个新的邻域解。根据设计的接受规则，如果当前的邻域解满足接受规则，则更新当前解为邻域解。如果当前解比最优解的成本还要低，则将当前解保存至最优解。破坏算子和修复算子的权重将根据它们在搜索过程中对解的贡献来进行动态调整（如第 5-11 行所示）。

（3）判断当前是否达到终止条件。是，则终止算法；否，则缓慢降低温度，跳转到第（2）步继续进行（如第 4、12 行所示）。

破坏算子和修复算子是自适应大邻域搜索算法的关键组成部分。在这里，分别设计了三种破坏算子和三种修复算子，期望对当前解进一步改善。这里的破坏算子就是选择哪个边缘服务器来降低其成本，即最终的计费值。本节采用的三种破坏算子如下所示：

（1）选择一个随机的边缘服务器。

（2）选择最近一次成功更新当前解的边缘服务器。

（3）选择当前成本（即计费值）最高的边缘服务器。

当通过破坏算子选择了一个边缘服务器s_i之后，s_i的一些采样点上的带宽需求就需要重新分配给其他的边缘服务器。由于本节是基于 OptDinic 算法来产生邻域解，而该邻域解与其构图时的建边顺序存在很大的关系。因此，本节通过在 OptDinic 算法中修改建边的顺序，来决定哪些边缘服务器先承担原先分配给s_i的带宽需求。本节所采用的三种修复算子如下所示：

（1）边缘服务器之间的优先级通过随机产生。

（2）边缘服务器之间的优先级按照连通性进行排序。

（3）边缘服务器之间的优先级按照输入文件中的顺序进行排序。

给定一个经破坏算子挑选出来的边缘服务器s_i，基于爬山的优化算法的目标是：在不增加其他边缘服务器的成本（即计费值）的前提下，尝试尽可能地降低s_i的计费值。如算法 3.3 所示，基于爬山的优化算法主要包括三个步骤：（1）初始化，包括根据破坏算子挑选出待优化的边缘服务器s_i并计算其当前计费值，根据修复算子获取边缘服务器的优先级，根据当前温度随机设置初始下探步长，以及备份当前解，便于算法尝试失败的时候直接回滚（如第 1-3 行所示）。（2）尝试以当前步长去优化s_i，即让s_i的计费值下探至少当前步长大小。在关于s_i的带宽序列中，找到满足采样值小于等于旧计费值且大于新的目标计费值的所有采样点，如果该采样点不属于s_i的免费时段，则s_i需要下探的采样点计数加一（如第 5-9 行所示）。（3）每次尝试下探某个采样点之前，获取其他边缘服务器在不增加成本的情况下最多能够分配的带宽容量，具体而言：若当前采样点是某个边缘服务器的免费时段，则其可以分配的带宽容量为其容量上限，否则设置为当前计费值。

利用提出的 OptDinic 算法去产生新的邻域解，并验证其可行性。如果存在可行的邻域解，则根据残存网络上的反向边更新当前解。如果 s_i 上可优化的采样点数量达到设定值，说明 s_i 优化成功，则此时接受该邻域解，计算总带宽开销成本并直接返回。如果失败，从备份中回滚后，将当前下探步长减半，然后再次进行尝试（如第 10-24 行所示）。该算法的时间复杂度为 $O\left(|T|nm(n+m)^2 log\Theta\right)$。

算法 3.3：基于爬山的优化算法 $HCO\left(\hat{X},\hat{C},d,r,\Theta\right)$

输入：当前解 $\hat{X}$，当前开销成本 $\hat{C}$，破坏算子 d，修复算子 r，当前温度 Θ

输出：当前解 $\hat{X}$，当前开销成本 $\hat{C}$

1 $s_i \leftarrow d$； $order \leftarrow r$； $step \leftarrow random(1,\Theta)$；

2 $X,C \leftarrow \hat{X},\hat{C}$； //备份

3 $q_i \leftarrow$ 通过公式（3.2）计算挑选出来的边缘服务器 s_i 目前的计费值；

4 **while** $step > 0$ **do**

5 　$cnt \leftarrow 0$； $f \leftarrow 0$； $G \leftarrow \varnothing$；

6 　**for** $k \leftarrow 1$ *to* $|T|$ **do**

7 　　**if** $\sum_{j=1}^{n} x_{ij}^{k} \le q_i$ **and** $\sum_{j=1}^{n} x_{ij}^{k} > q_i - step$ **then**

8 　　　$G \leftarrow G \cup \{t_k\}$；

9 　　　**if** t_k 属于 s_i 的计费时段 **then** $f \leftarrow f+1$；

10 　**foreach** $t_k \in G$ **do**

11 　　$H \leftarrow \varnothing$；

12 　　**for** $e \leftarrow 1$ *to* m **do**

13 　　　**if** t_k 属于 s_e 的免费时段 **then** $h_e \leftarrow b_e$；

14 　　　**else** $h_e \leftarrow$ 通过公式（3.2）计算 s_e 目前的计费值；

15 　　　$H \leftarrow H \cup \{h_e\}$；

16 　　**if** $OptDinic\left(H,\hat{X},s_i,order,step,k\right)$ **then**

17 　　　$cnt \leftarrow cnt+1$；

18 　　　根据残存网络上的反向边更新 $\hat{X}$；

19 　　　**if** $cnt = f$ **then**

20 　　　　$\hat{C} \leftarrow$ 通过公式（3.3）计算其总开销成本；

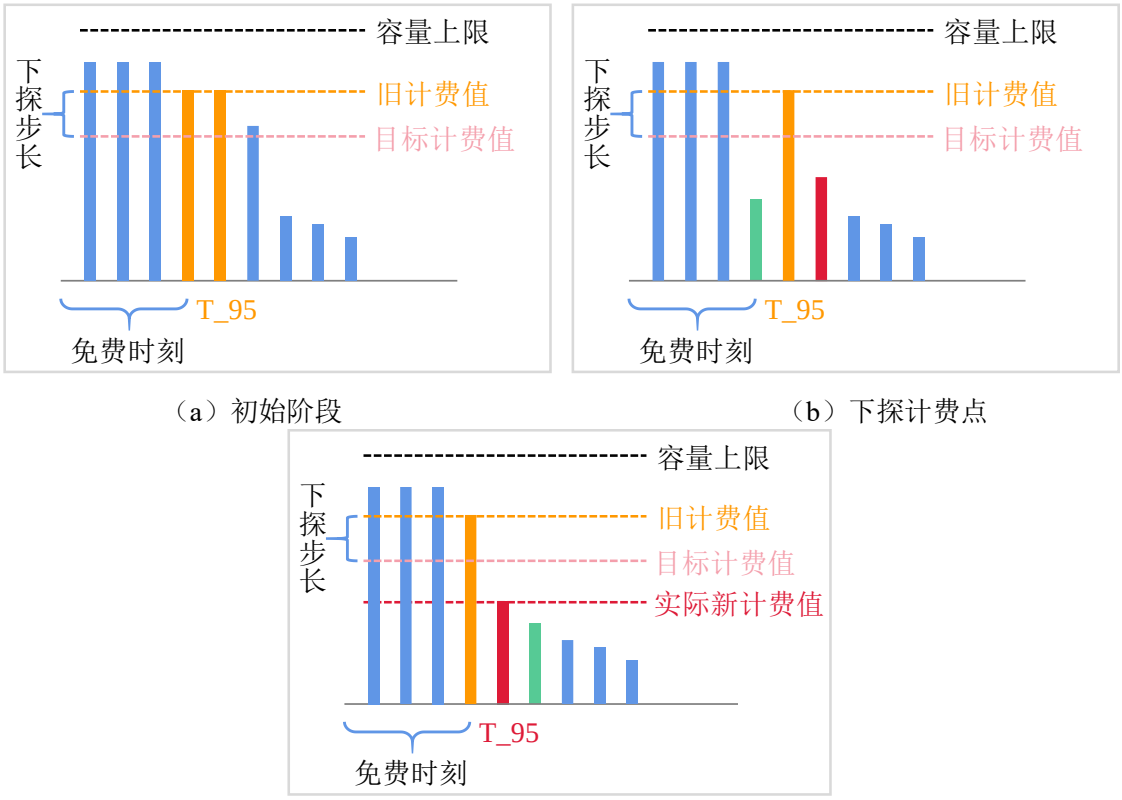
21 　　　　**return** $\hat{X},\hat{C}$；

22 　$step \leftarrow step / 2$；

23 　$\hat{X},\hat{C} \leftarrow X,C$； //从备份中回滚

24 **return** X,C；

图 3.2 给出了一个待优化的目标边缘服务器在算法 3.3 中的优化过程。其中，各种颜色的柱形表示的是该服务器在某时段上的负载，T_95 表示的是计费值所对应的采样时段。图 3.2（a）获取所有当前负载在目标计费值和旧计费值之间的所有采样点，并且得到此时该边缘服务器需要下探的采样点数量为 2。图 3.2（b）选择若干个采样点进行了负载的下探。图 3.2（c）给出该边缘服务器成功优化后的结果，可以看出该边缘服务器新的计费值降到了目标计费值以下。



（a）初始阶段　　（b）下探计费点

（c）将带宽序列重新排序

图 3.2 目标边缘服务器在爬山算法中的优化过程

Fig.3.2 Optimization process of target edge server in hill climbing algorithm

为了验证当前优化方案的可行性，本节基于原始的 Dinic 算法提出了 OptDinic。如图 3.3 所示，共有 6 种节点和 2 种边。其中，绿色的边表示当前已经分配的带宽需求，即在某个特定的采样点上的当前解。相比之下，红色的边表示还未分配的带宽容量。为了实现在某个特定的采样点上降低待优化服务器上的实际负载，通过添加虚节点并设置虚节点的带宽需求恰好等于当前需下探的步长大小。值得注意的是，从源节点到待优化服务器之间没有还未分配的带宽容量，并且虚节点只与待优化服务器连通而与其他边缘服务器之间并不连通。因此，当该残存网络的流的值增加时，必然是待优化服务器将一部分带宽容量分配给了虚节点，而其他

的边缘服务器则用来承担原本被分配给待优化服务器的那些边缘用户的带宽需求。所以，可以根据该残存网络的最大流是否等于当前需下探的步长大小，来判断当前优化方案的可行性。至于哪个边缘服务器首先尝试增加其负载，可以通过传入的边缘服务器的优先级来控制构图时的建边顺序。最后，通过残存网络中反向边上的流就可以获取可行的邻域解。

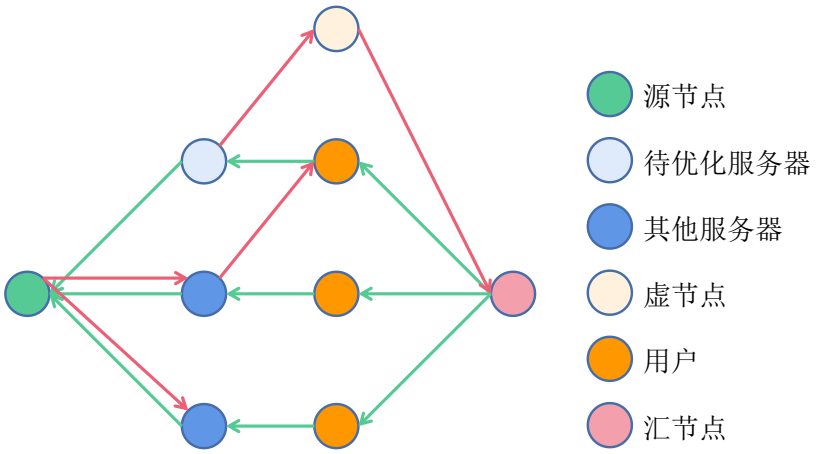


图 3.3 OptDinic 算法中的残存网络

Fig.3.3 Residual network in OptDinic algorithm

给定一个经破坏算子挑选出来的边缘服务器 s_i，基于模拟退火的优化算法的目标是：在尽量不增加其他边缘服务器的成本（即计费值）的前提下，尝试将某一个免费时段上的负载降低至当前计费值以下。对于边缘服务器而言，免费时段上的负载是在初始解生成的步骤决定的，而基于爬山的优化算法并不会降低该大小。若边缘服务器上的计费点下探不下去时，只能通过将该计费点与某个免费时段进行交换后，才能再次尝试下探。因此基于模拟退火的优化算法能够实现跳出局部最优。

如算法 3.4 所示，基于模拟退火的优化算法主要包括三个步骤：（1）初始化，包括根据破坏算子挑选出待优化的边缘服务器 s_i 并计算其当前计费值，根据修复算子获取边缘服务器的优先级排序，以及备份当前解，便于算法尝试失败的时候直接回滚（如第 1-3 行所示）。（2）尝试在不增加总带宽开销成本的情况下跳出局部最优。因为不能增加总成本，所以对于其他的边缘服务器而言，若该采样点是其免费时段，则其可提供的带宽容量是容量上限；若该采样点是其计费时段，其可提供的带宽容量是当前的计费值。然后利用提出的 OptDinic 算法去产生新的邻域解，并验证其可行性。如果邻域解可行，则根据残存网络上的反向边获取新的分配方案并更新当前解，计算总带宽开销成本后直接返回（如第 4-14 行所示）。（3）如果在第（2）步优化失败，则尝试在增加总带宽开销成本的情况下跳出局

部最优。此时，所有边缘服务器可提供的带宽容量均为其自身的容量上限。若新的邻域解可行，且邻域解的总成本根据 Metropolis 准则判断可以接受，则返回该邻域解。如果尝试失败，从原始备份中回滚（如第 15-23 行所示）。该算法的时间复杂度为 $O\left(|T|nm(n+m)^2\right)$。因此，所提出的 DEBA 算法的总时间复杂度为 $O\left(\Gamma\Phi|T|nm(n+m)^2 log\Theta\right)$。

算法 3.4：基于模拟退火的优化算法 $SAO\left(\hat{X},\hat{C},d,r,\Theta\right)$

输入：当前解 $\hat{X}$，当前开销成本 $\hat{C}$，破坏算子 d，修复算子 r，当前温度 Θ

输出：当前解 $\hat{X}$，当前开销成本 $\hat{C}$

1 $s_i \leftarrow d$； $order \leftarrow r$； $f \leftarrow 0$；

2 $q_i \leftarrow$ 通过公式（3.2）计算挑选出来的边缘服务器 s_i 目前的计费值；

3 $X,C \leftarrow \hat{X},\hat{C}$； //备份

4 /*尝试在不增加成本的情况下跳出局部最优*/

5 **foreach** t_k 属于 s_i 的免费时段 **do**

6 　$step \leftarrow \sum_{j=1}^{n} x_{ij}^k - max(q_i - 1,0)$； $H \leftarrow \varnothing$；

7 　**for** $e \leftarrow 1$ *to* m **do**

8 　　**if** t_k 属于 s_e 的免费时段 **then** $h_e \leftarrow b_e$；

9 　　**else** $h_e \leftarrow$ 通过公式（3.2）计算 s_e 目前的计费值；

10 　　$H \leftarrow H \cup \{h_e\}$；

11 　**if** $OptDinic\left(H,\hat{X},s_i,order,step,k\right)$ **then**

12 　　根据残存网络上的反向边更新 $\hat{X}$；

13 　　$\hat{C} \leftarrow$ 通过公式（3.3）计算其总开销成本；

14 　　**return** $\hat{X},\hat{C}$；

15 /*尝试在增加成本的情况下跳出局部最优*/

16 **foreach** t_k 属于 s_i 的免费时段 **do**

17 　$step \leftarrow \sum_{j=1}^{n} x_{ij}^k - max(q_i - 1,0)$； $H \leftarrow B$； $\delta \leftarrow random(0,1)$；

18 　**if** $OptDinic\left(H,\hat{X},s_i,order,step,k\right)$ **then**

19 　　根据残存网络上的反向边更新 $\hat{X}$；

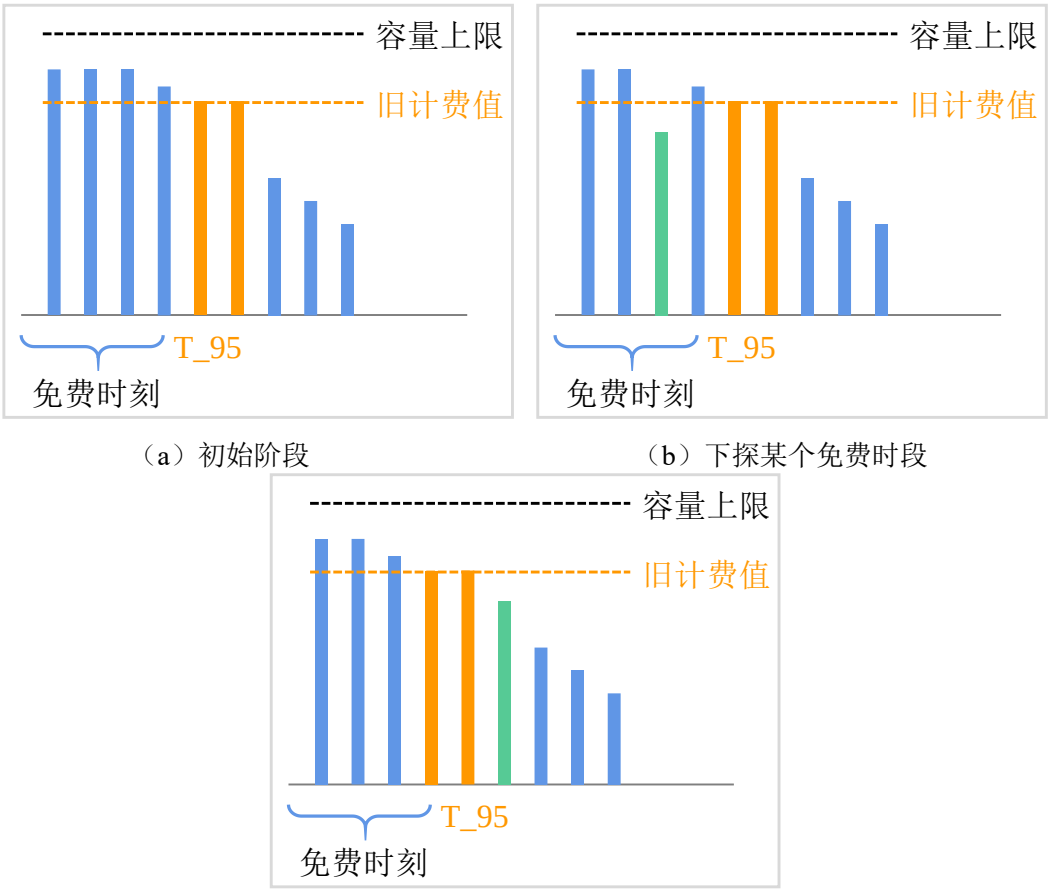
20 　　$\hat{C} \leftarrow$ 通过公式（3.3）计算其总开销成本；

21 　　**if** $\delta < exp\left(\left(C-\hat{C}\right)/\Theta\right)$ **then return** $\hat{X},\hat{C}$；

22 　　**else** $\hat{X},\hat{C} \leftarrow X,C$； //从备份中回滚

23 **return** X,C；

图 3.4 给出了一个待优化的目标边缘服务器在算法 3.4 中的优化过程。



（a）初始阶段　　（b）下探某个免费时段

（c）将带宽序列重新排序

图 3.4 目标边缘服务器在模拟退火算法中的优化过程

Fig.3.4 Optimization process of target edge server in simulated annealing algorithm

3.4 实验结果与分析

在本节，基于一个真实边缘带宽需求数据集进行了一系列的实验，验证了本章所提出的 DEBA 算法的有效性。

3.4.1 实验数据

本文采用的私有数据集来自一个真实的短视频和直播应用程序，包括一个月内的流量记录。以 30 天为计费周期，采样时间间隔设置为 5 分钟，获得该计费周期内的总带宽需求数据集。图 3.5 显示了总带宽需求随时间的波动情况。可以看出，带宽需求具有明显的周期性变化特征，在真实生活中则体现为：凌晨时段应用程序的使用量大大降低，带宽需求下降到谷底，而在夜间时段应用程序的使用量和带宽需求逐渐达到峰值。在该计费周期内，整个网络带宽需求的峰值最高约为 4Tbps。在本节的实验中，设置边缘用户的数量 n 为 25，按照各省的人口数量将每

个时段的带宽总需求分配至各个边缘用户。考虑 110 个分布在中国不同省份的边缘服务器，它们的带宽容量上限 B 假设服从期望为 512Gbps 的正态分布。将边缘服务器与边缘用户之间的网络时延矩阵 L 设置为服从 140ms 到 540ms 之间的均匀分布，并将网络时延上限 τ 设置为 400ms。带宽租用的单位成本 p 从最新的阿里云 ECS 价格表[52]中获得。

因此，本章的实验数据主要包括四个文件：带宽需求数据集（所有边缘用户在每个采样时段上的带宽需求）、网络时延数据集（所有边缘用户与所有边缘服务器之间的网络时延）、带宽容量上限数据集（所有边缘服务器的带宽容量上限）、配置文件（网络时延上限配置和带宽租用单位成本）。

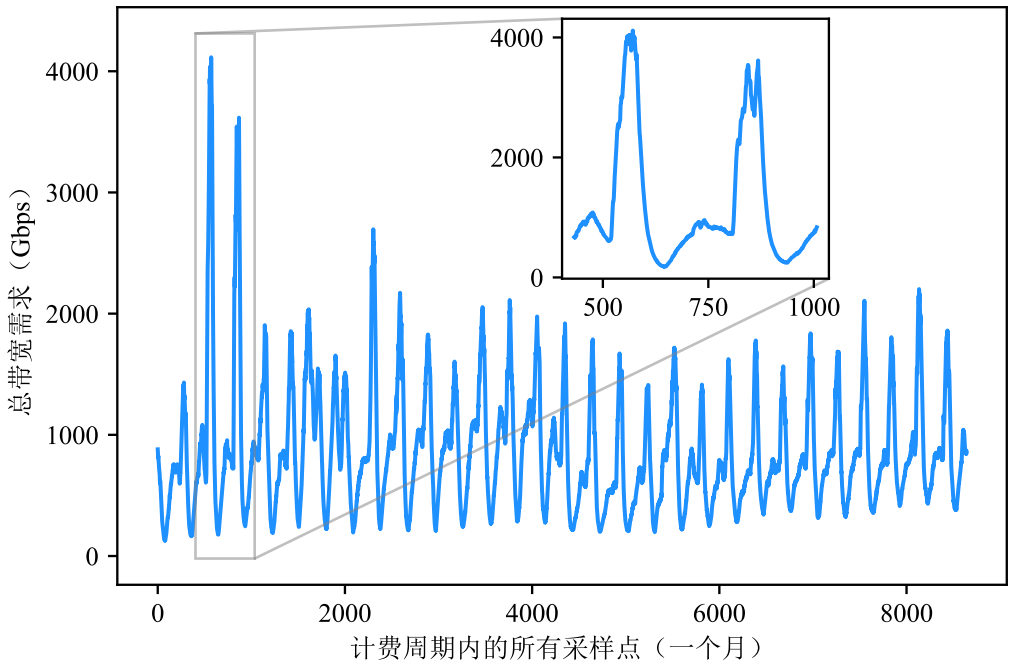


图 3.5 整个计费周期内的带宽总需求

Fig.3.5 Total bandwidth demands over the entire billing period

3.4.2 模型参数设置

本节给出一些重要的超参数的设置结果，如表 3.2 所示。

表 3.2 超参数的设置

Table 3.2 Setting of hyper-parameters

名称	数值
Θ	512
α	0.95

表 3.2 超参数的设置（续）

Table 3.2 Setting of hyper-parameters(continued)

名称	数值
σ	0.4
λ	0.14
Γ	50
Φ	100
ω_1	1.0
ω_2	0.75
ω_3	0.5
ω_4	0.25

3.4.3 对比算法与评价指标

（1）对比算法

由于现有的研究并没有考虑网络时延约束条件，因此在这些文献中提出的算法并不能作为对比算法应用于本章所提出的问题中。比如最近提出的CASCARA[36]、PeCo-Huffman[37]等算法，在这些文献中构建的模型只有每个时段的总带宽需求，并没有用户的概念。而较早的文献中基于最小费用多商品流的一些算法比如 GFA-offline[26]，虽然在模型的构建中涉及到了多种带宽需求类似于本章中的用户概念，但是并没有对用户与服务器之间的连通性进行区分，并且GFA-offline 整体的思想与本章所提出的 FISG 有类似之处，所以本节直接使用 FISG 算法进行替代作为对比算法之一。此外，本节还将常见的分配算法比如 Load Balance[53]、Round Robin[54]和 Equal Split[55]结合本章问题场景稍作修改后作为对比算法。最后，本节对 DEBA 算法中的各个模块分别去除后进行了消融实验。下面将对这些对比算法进行详细介绍：

Load Balance（LB）：针对计费周期内每个边缘用户的带宽需求，LB 将会启发式地随机选择 d 个连通的候选边缘服务器。值得注意的是，考虑到 95 计费的特点，当前采样点是其免费时段的边缘服务器会被优先选择。然后，边缘用户的带宽需求将会被分配到负载最低的候选边缘服务器。如果选定的服务器没有足够的容量，那么剩余的带宽需求将会分配给负载率第二低的边缘服务器，以此类推。

Round Robin（RR）：针对计费周期内每个边缘用户的带宽需求，RR 将会以轮询的方式选择其要分配给的目标边缘服务器。如果选定的服务器没有足够的容量，那么剩余的带宽需求将会以相同的轮询策略分配给其他的边缘服务器。

Equal Split（ES）：针对计费周期内每个边缘用户的带宽需求，ES 会将所有边缘服务器按其容量大小进行排序。按照这个顺序进行分配时，每个边缘服务器分到的带宽数量大小等于其剩余带宽容量和平均剩余带宽需求中的较小值。

FISG：本章 3.3.1 节提出的一种基于网络流的可行初始解生成算法，能够充分利用 95 计费模式的特点，降低总带宽开销成本。

Full-SA：“Full”代表本章提出的 DEBA 算法，“-SA”代表在解的优化过程中去除 3.3.2 节提出的基于模拟退火的优化算法模块。

Full-ALNS：本章提出的 DEBA 算法去除自适应大邻域搜索算法模块，即在算法 3.2 中的每一轮搜索的过程中，破坏算子和修复算子始终选择随机策略。

（2）评价指标

本节选择计费周期内所有边缘服务器的总带宽开销成本作为最终的评价指标。

3.4.4 性能对比分析

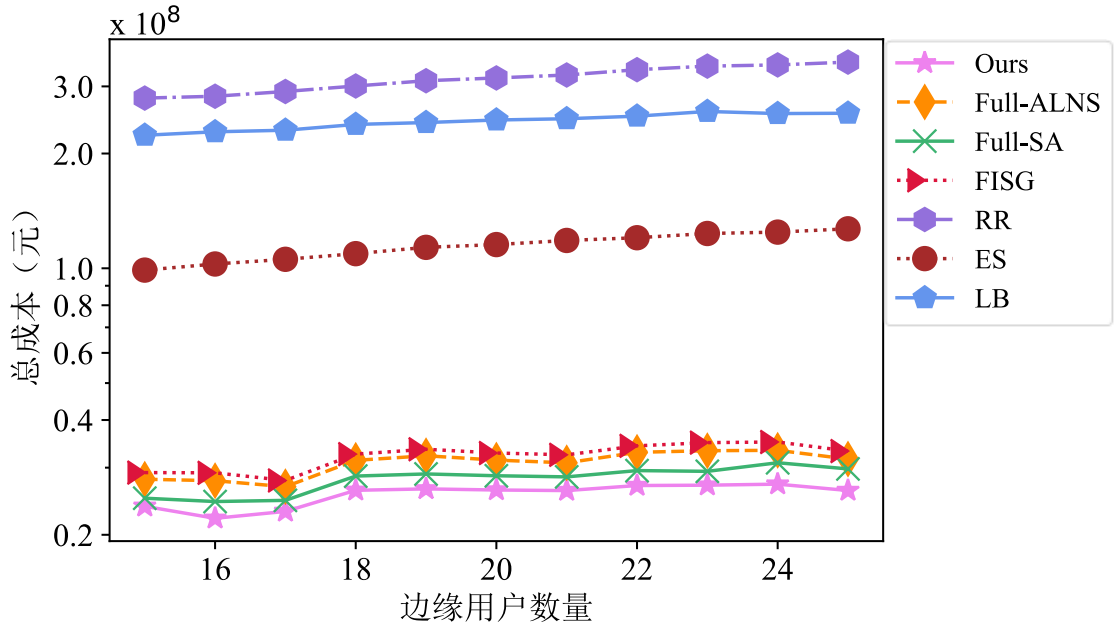


图 3.6 不同用户数量下的总成本比较

Fig.3.6 Comparison of total cost under different number of users

不同用户带宽需求下的比较结果：如图 3.6 和图 3.7 所示，本节根据总带宽开销成本比较了所提出的 DEBA 算法和其他对比算法的性能。可以看到，无论是增加用户数量还是增大每个用户的边缘带宽需求，总成本都呈现出上升的趋势，其中改变用户数量对总成本的影响较小，而缩放每个用户的带宽需求对总成本的影响较明显。其次，RR、LB、ES 算法得到的分配方案的总成本始终排在最高的前三位，而本文在第 3.3.1 节提出的基于网络流的可行初始解生成算法 FISG，明显要

优于这三种算法。在图 3.6 中，FISG 获得的解其总成本大致为 RR 算法的 10.25%，为 LB 算法的 13.26%，为 ES 算法的 28.20%。在图 3.7 中，FISG 获得的解其总成本大致为 RR 算法的 9.84%，为 LB 算法的 13.59%，为 ES 算法的 27.33%。此外，本文所提出的 DEBA 算法能够在 FISG 的基础上，对初始解进行优化，使得成本能够分别平均降低 20.68%和 17.33%。最后，从图中可以看出，无论是去除 ALNS 模块还是 SA 模块，在几乎绝大多数情况下都将导致总成本的上升。

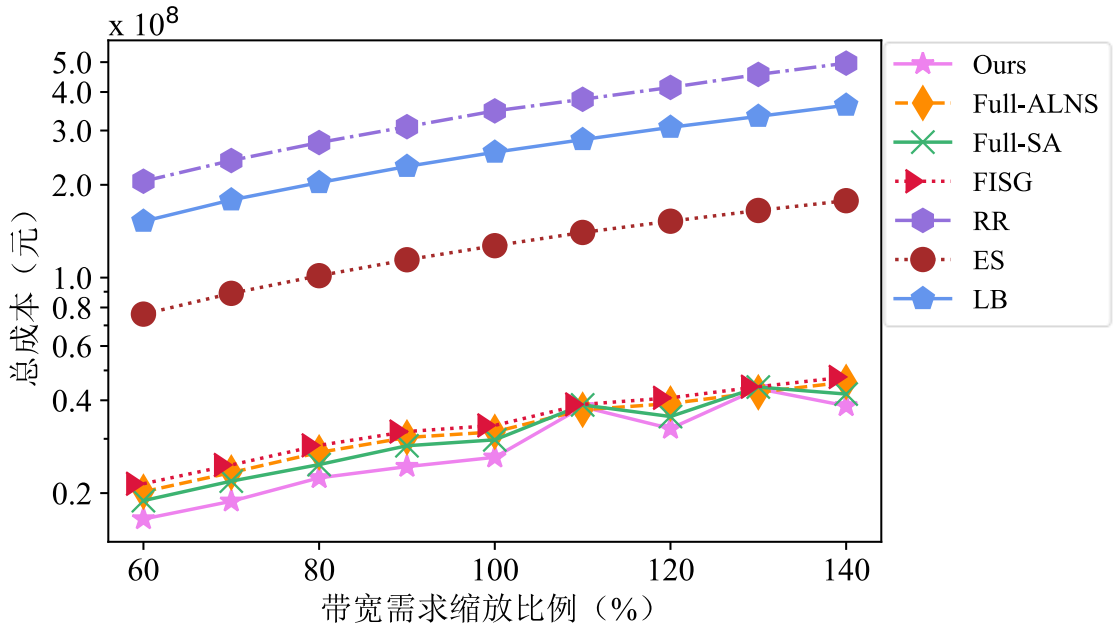


图 3.7 不同带宽需求下的总成本比较

Fig.3.7 Comparison of total cost under different bandwidth demands

分析：因为若干个用户数量的增多对带宽总需求的增加并不明显，并且可以通过服务器的调度分配掉这些增加的带宽需求，所以对总成本的影响不如直接缩放用户带宽需求来得明显。由于 RR 算法通过轮询的方式选择边缘服务器，导致服务器没有被充分的利用，所以其计费值最高。LB 算法每次都随机地选择服务器，ES 算法则根据带宽容量以一个固定的顺序选择服务器，所以 LB 的总成本会高于 ES 算法。本文提出的 FISG 算法由于在第一阶段考虑了三种直接的排序策略，并在第二阶段逐步更新服务器的计费值，使得每个边缘服务器在免费时段和计费时段上都得到充分利用。因此相比于 RR、LB、ES 这三种对比算法，FISG 有着明显的提升。与简单的局部搜索策略相比，基于 ALNS 的方法可以拓宽邻域的搜索空间，基于 SA 的方法可以跳出局部最优，这也是本文提出的 DEBA 算法在几乎所有的实验结果中都优于其他对比算法的原因。

不同网络时延约束下的比较结果：如图 3.8 所示，本节比较了在不同网络时延约束下各个算法的总成本。可以看出，RR、LB、ES 仍然是总成本最高的三种算法，而本文提出的 DEBA 算法实现了最低的总带宽开销成本。此外，随着网络时延约束的放宽，除了 RR 算法和 LB 算法之外，其余的几种算法都呈现出总成本下降的趋势。特别地，当网络时延约束上限放宽到 440ms 时，FISG、Full-ALNS、Full-SA、Ours 这几种算法的总成本都已经降低到 0，为方便显示，因此对后面的实验数据进行了省略。最后，随着网络时延约束的放宽，可以看出本文提出的这几种算法相对于其他对比算法的优势变得更加明显。当网络时延约束上限设置为 350ms 时，FISG 算法的总成本大致为 RR 算法的 61.15%，且本文提出的 DEBA 算法相比于 FISG 算法平均降低了 0.07%。然而，当网络时延约束上限放宽到 430ms 时，FISG 算法的总成本大致为 RR 算法的 3.77%，且本文提出的 DEBA 算法相比于 FISG 算法平均降低了 17.71%。

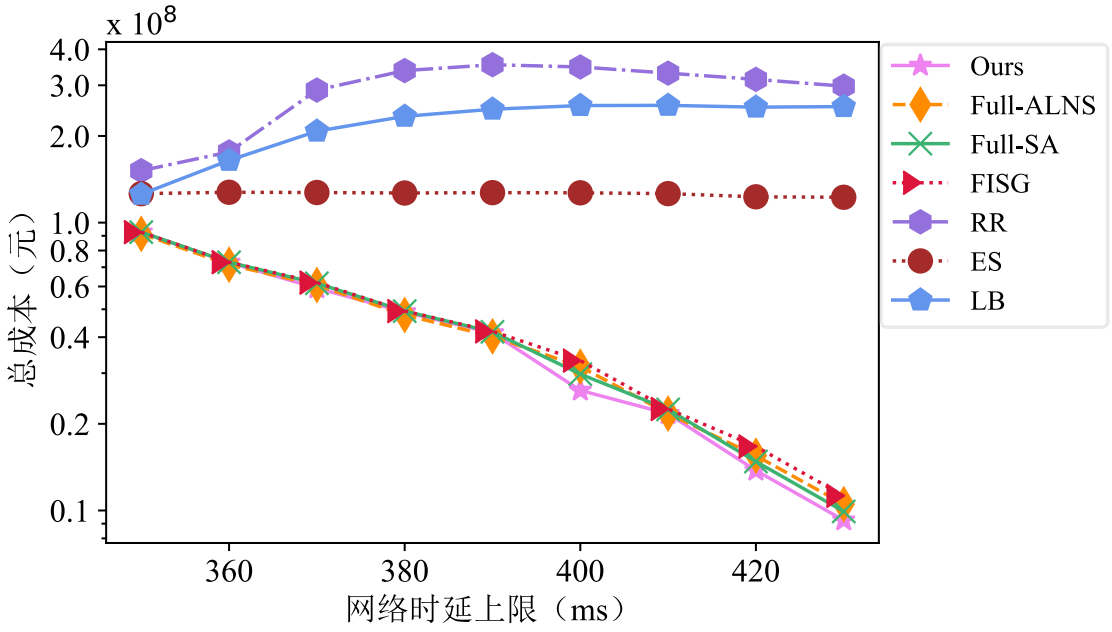


图 3.8 不同网络时延约束下的总成本比较

Fig.3.8 Comparison of total cost under different network latency constraints

分析：随着网络时延约束的放宽，每个边缘用户连通的服务器数量也将增多。由于 RR 算法采用的是轮询策略，而 LB 算法采用的是随机选择的策略，这两种算法的候选服务器的范围也将增大，这就导致更多的服务器未被充分利用，所以总成本反而上升。而剩余的其他算法由于或多或少采用了负载均衡的思想，所以导致总成本的下降。本文提出的 FISG 算法由于其充分利用了 95 百分位计费的特点，当可用的边缘服务器数量增多，这一优势将被放大。此外，可用的边缘服务器数

量增多也意味着更多的邻域解能够满足条件，从而获得更高的优化成功率，提高了解的质量。因此，随着网络时延约束的放宽，本文提出的 DEBA 算法的总成本能够比其他的对比算法下降得更多。

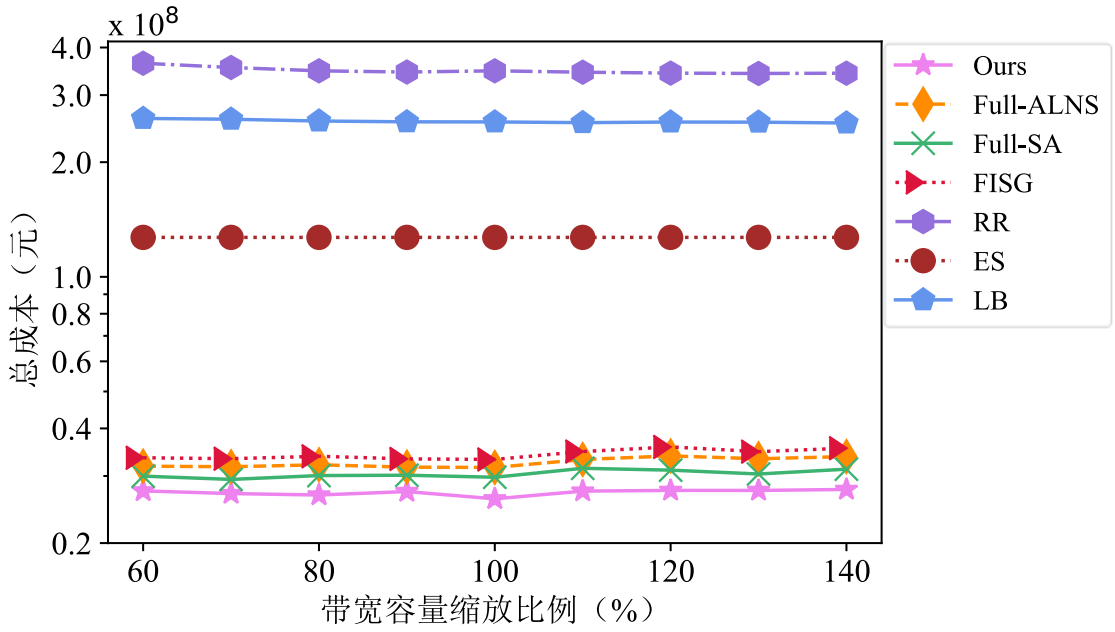


图 3.9 不同带宽容量下的总成本比较

Fig.3.9 Comparison of total cost under different bandwidth capacities

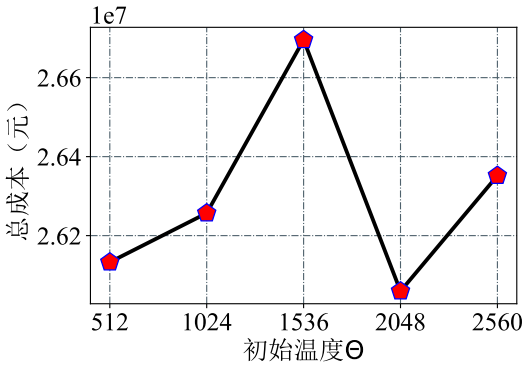
不同容量约束下的比较结果：图 3.9 比较了不同带宽容量下各个算法的总成本。总成本最高的前四种算法分别是 RR、LB、ES 和 FISG。Full-ALNS 的总成本相比于 FISG 平均下降了 4.83%，Full-SA 的总成本相比于 FISG 平均下降了 11.13%。本文提出的 DEBA 算法的总成本最低，与 FISG 相比，平均降低了 20.50%。

分析：由于复杂的 95 百分位优化问题的搜索空间较大，因此单一的搜索策略很难找到一条合适的优化路径。相比之下，基于 ALNS 的算法通过引入多个算子和自适应机制拓宽了解的优化范围。此外，SA 机制在搜索的过程中有一定的概率接受更差的邻域解，因此能够跳出局部最优。本文提出的 DEBA 算法由于同时结合了 SA 和 ALNS 的优势，因此能够获得更优的解决方案。

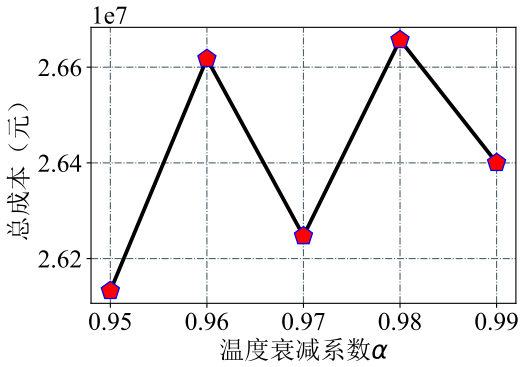
上述结果表明，无论如何放缩用户数量、用户的带宽需求、网络时延约束或服务器的带宽容量，本文提出的 FISG 算法均可获得高质量的初始解，且 DEBA 算法总能够实现较大的优化效果。此外，消融实验表明了各个模块的必要性。由于爬山算法并不会导致总成本的上升，因此本节并没有将该模块加入到消融实验中。

3.4.5 参数敏感性实验

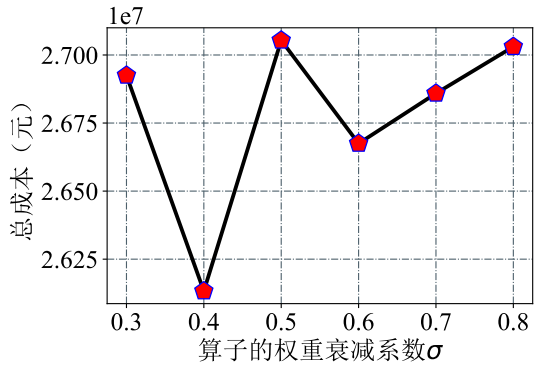
本节研究了一些重要的超参数对实验结果的影响，如图 3.10 所示。从图 3.10（g）、（h）、（i）、（j）中可以看出，DEBA 算法对 ω_1、ω_2、ω_3、ω_4 这四个参数的敏感程度最低，这是因为无论这四个参数的具体数值如何变化，其中的大小关系始终保持不变，即 $\omega_1 > \omega_2 > \omega_3 > \omega_4$，所以导致搜索过程中算子的选择并不会发生改变，最终获得的邻域解也不变。从图 3.10（a）、（b）中可以看出，DEBA 算法对模拟退火的初始温度 Θ、温度衰减系数 α 这些参数较不敏感，但是由于这些参数影响了邻域解搜索的方向、搜索的步长，所以对结果还是有略微的影响。从图 3.10（c）、（d）、（e）、（f）中可以看出，DEBA 算法对算子的权重衰减系数 σ、模拟退火算法的执行频率 λ、外循环次数 Γ、内循环次数 Φ 这四个参数比较敏感。因为算子的权重衰减系数 σ 直接决定了 ALNS 模块的重要性程度，影响了算子的选择过程，而模拟退火算法的执行频率 λ 决定了 SA 模块的重要性程度。此外，无论是外循环还是内循环，随着轮数的增大，实验的结果朝着更优的方向变化。这是因为随着搜索轮数的变大，搜索的范围也将变大，更有可能找到一个更优的邻域解。但是搜索轮数的增大，同时意味着搜索时间的延长，所以这两个参数的设置需要在运行时间和实验效果两者之间做权衡。



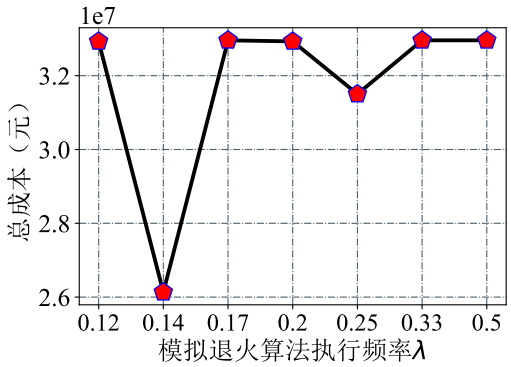
（a）初始温度 Θ 对结果的影响



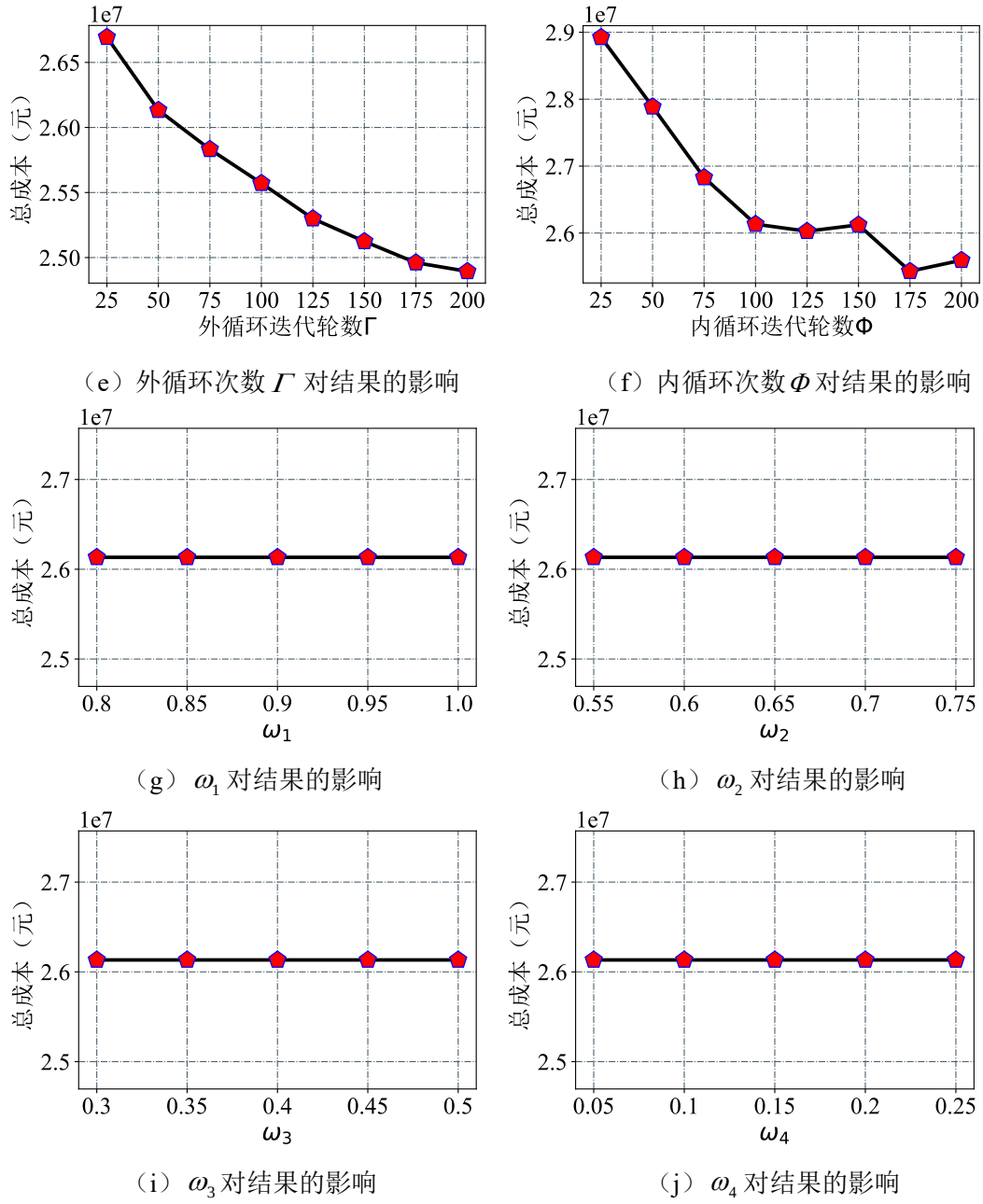
（b）温度衰减系数 α 对结果的影响



（c）算子的权重衰减系数 σ 对结果的影响



（d）模拟退火算法的执行频率 λ 对结果的影响



（e）外循环次数 Γ 对结果的影响　　（f）内循环次数 Φ 对结果的影响

（g） ω_1 对结果的影响　　（h） ω_2 对结果的影响

（i） ω_3 对结果的影响　　（j） ω_4 对结果的影响

图 3.10 实验中一些重要参数的敏感性结果

Fig.3.10 Sensitivity results of some important parameters in the experiment

3.5 本章小结

对于带宽需求可分割假设下的边缘带宽分配问题，本章建立了考虑 95 计费模式和网络时延约束的问题模型，证明了其 NP 难特性，并提出了 DEBA 算法予以解决。紧接着，对该算法的各个流程步骤进行详细地介绍。最后，本章在基于真实边缘带宽需求数据集上进行了一系列对比实验和消融实验，实验结果表明了 DEBA 算法的有效性以及各个模块的必要性。

4 基于带宽需求不可分割假设下的边缘带宽分配问题

4.1 概述

上一章主要研究的是在带宽需求可分割假设下的边缘带宽分配问题，即边缘用户的带宽需求可以被分割成为任意整数大小，然后这些带宽需求被分配到各个满足网络时延约束的边缘服务器中。但是在现实生活中，更多的是带宽需求不可分割的场景。例如，当某个用户在使用某个进程时，为了保证体验的连续性，通常会将整个进程的带宽需求分配到同一个服务器中。此外，带宽需求的不可拆分性还可以实现端到端的可追溯性，为性能调优、程序调试等提供便利。除此之外，与上一章将所有边缘服务器的带宽租用成本设置为相同值这一操作不同，本章引入了更加复杂的边缘服务器带宽成本计算公式，既考虑了服务器的开机成本，又体现出不同服务器之间的带宽租用成本差异性，更贴近真实应用场景。因此，针对带宽需求不可分割假设和引入边缘服务器成本计算公式的边缘带宽分配问题，本章同样首先建立了对应的模型，给出其形式化定义。其次，针对带宽需求不可分割这一特点，本章引入了流的概念，将流作为带宽需求调度分配的最小单位。接着，本章提出了两阶段的 IEBA 算法对该 NP 难问题进行了解决。具体而言，在第一阶段，通过流的迁移交换，结合 95 计费模式以及问题背景，生成一个可行的初始解。在第二阶段，重新设计了适合该场景的爬山算法和模拟退火算法，并基于自适应大邻域搜索算法框架对解的质量进行优化。同时由于边缘服务器存在着开机成本，本章还设计了对应的服务器关停算法，并将其与 IEBA 算法相结合，提出了适合带宽需求较低场景的 IEBA-L 算法。最后，本章基于真实边缘带宽需求数据集生成了多个数据集，并在这多个数据集上进行了一系列实验，实验结果表明了 IEBA 算法和 IEBA-L 算法在各自场景下的有效性。

4.2 系统模型和问题定义

本节在上一章的模型的基础上，结合带宽需求不可分割假设和边缘服务器成本计算公式，对上一章的模型进行了对应的修改和重构，然后给出其形式化定义。为了避免赘述，表 4.1 仅列出在本章研究中新引入的一些符号以及对应的描述信息。

表 4.1 符号列表

Table 4.1 List of notations

符号	描述
f_h	F 中第 h 种流。
F	流的集合。
φ	流的数量。
y_{jh}^k	u_j 的第 h 种流 f_h 在 t_k 处的带宽需求。
Y	所有边缘用户的所有流在整个计费周期 T 内的带宽需求矩阵。
x_{ijh}^k	在 t_k 处，u_j 的第 h 种流 f_h 是否分配给 s_i，1 表示分配，0 表示不分配。

4.2.1 系统模型

由于带宽需求的不可分割性，并且在现实场景中一个用户会使用多个进程，因此，本章引入流的概念。其中，流与边缘服务器之间的网络时延跟所属的边缘用户保持一致。假设在整个计费周期 T 内，一共存在着 φ 种不同的流，令 $F=\{f_1,f_2,\ldots,f_\varphi\}$ 来表示流的集合。在每个采样时段，对于每个边缘用户，会有若干种流，流的种类和带宽需求大小不一定相同。本章使用 y_{jh}^k 来表示在第 k 个采样时段 t_k，边缘用户 u_j 对第 h 种流 f_h 的带宽需求（这里用 0 表示该采样时段该边缘用户对该种流无带宽需求）。此外，流作为带宽需求分配调度的最小单位，体现着不可分割性，即：在每个采样时段，一个边缘用户对一种流的带宽需求，需要不可拆分地分配到某个边缘服务器中。这里，本章使用 X 表示一种带宽分配方案，即：在第 k 个采样时段 t_k，边缘用户 u_j 的第 h 种流 f_h 是否分配给第 i 个边缘服务器用 x_{ijh}^k 表示。x_{ijh}^k 的取值范围为 $\{0,1\}$，其中 1 表示分配，0 表示不分配。

因此，在第 k 个采样时段 t_k，边缘服务器 s_i 接收到的带宽需求总和可以通过如下公式进行计算：

$$w_i^k=\sum_{j=1}^{n}\sum_{h=1}^{\varphi}x_{ijh}^k\times y_{jh}^k \tag{4.1}$$

由此形成的关于 s_i 的带宽序列可表示为 $W_i=\{w_i^1,w_i^2,\ldots,w_i^{|T|}\}$。此时，该计费周期内 s_i 的计费值可以表示为：

$$q_i=O_{95}(W_i) \tag{4.2}$$

为了体现不同边缘服务器带宽租用成本的差异性，以及增加服务器的开机成本，本章引入边缘服务器成本计算公式，如下所示：

$$c_i=\begin{cases}0, \sum_{k=1}^{|T|} w_i^k = 0 \\ V\times p, \ \sum_{k=1}^{|T|} w_i^k > 0 \ and \ 0\le q_i \le V \\ \left(1/b_i\times\left(q_i-V\right)^2+q_i\right)\times p, \ q_i > V\end{cases} \tag{4.3}$$

其中，c_i 表示边缘服务器 s_i 在整个计费周期内的带宽开销成本，V 表示服务器的初始设定计费值。上述公式表明：对于某个边缘服务器，只有当整个计费周期内都没有被使用过时，其成本才为 0，对应于现实场景中该服务器未开机的情况。若其真实计费值小于等于初始设定计费值，则最终的带宽开销成本设置为初始设定计费值乘以带宽的单位成本，对应于现实场景中服务器存在开机成本的情况。若其真实计费值大于初始设定计费值，则最终的带宽开销成本为真实计费值的二次函数，对应于现实场景中阶梯计费的计费方式。并且对于同一个真实计费值，带宽容量上限大的边缘服务器的带宽开销成本要小一些，对应于现实场景中租用共享带宽要比独占带宽便宜的情况。由此可以看出，通过引入更加复杂的边缘服务器成本计算公式，使得模型能够更加贴近现实应用场景。

最后，在整个计费周期内，边缘应用程序提供商为了满足边缘用户的各种流的带宽需求，租用边缘服务器所花费的总带宽开销成本的计算公式如下：

$$C=\sum_{i=1}^{m} c_i \tag{4.4}$$

4.2.2 问题定义

基于上节提到的相关公式，本节所提出的边缘带宽分配问题可以形式化地定义如下：

$$\begin{aligned} &min \ C \\ &s.t. \ \sum_{i=1}^{m} x_{ijh}^k = 1, \ \forall y_{jh}^k > 0 \\ &\quad \sum_{i=1}^{m} x_{ijh}^k = 0, \ \forall y_{jh}^k = 0 \\ &\quad w_i^k \le b_i \\ &\quad l_{ij} < \tau, \ \forall x_{ijh}^k > 0 \\ &\quad x_{ijh}^k \in \{0,1\} \\ &\quad i\in[1,m], \ j\in[1,n], \ k\in[1,|T|], \ h\in[1,\varphi] \end{aligned} \tag{4.5}$$

其中，公式（4.5）的第一行表明这个问题的目标是最小化边缘应用程序提供商的总带宽开销成本；第二行表明每个时段每个边缘用户的每种流的带宽需求只能分配给一个边缘服务器；第三行表明带宽需求为 0 的流不分配给任何边缘服务器；第四行表明边缘服务器接收到的带宽需求不能超过其带宽容量上限；第五行表明边缘用户上的流的带宽需求只能分配到满足网络时延约束的边缘服务器中。

对于本节所提出的问题，如果将 95 计费模式松弛到百分之一百计费模式（取带宽序列的最大值，也就是最大值计费模式），并且令边缘服务器的带宽开销成本表示为其计费值的线性函数，此时讨论的问题就退化为了广义指派问题。而广义指派问题作为运筹学中的一类经典的组合优化问题，已被证明是 NP 难问题[56]。因此，本节所提出的边缘带宽分配问题也是一个 NP 难问题。

4.3 IEBA 算法

为了解决上述问题，本节提出了两阶段的 IEBA 算法，其流程步骤如图 4.1 所示。

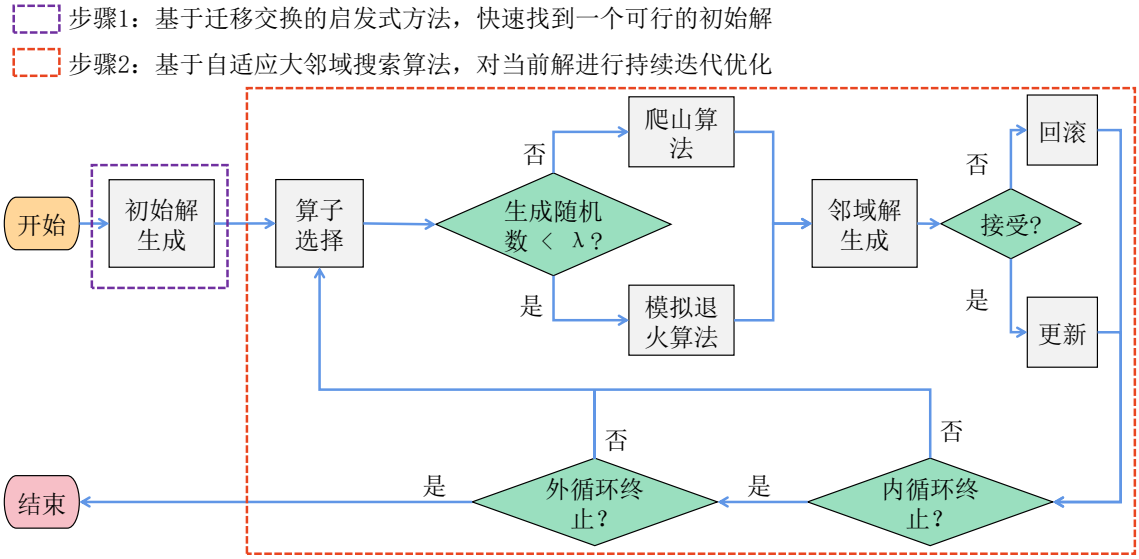


图 4.1 IEBA 算法的执行流程

Fig.4.1 The process of the proposed IEBA algorithm

4.3.1 流的迁移交换

图 4.2 给出了在不增加成本的情况下，迁入服务器的六种带宽序列变化结果。从图 4.2（a）可以看出，若当前采样点上的原采样值为 0 且带宽序列长度小于免费时段的数量，当某个流从其他边缘服务器迁入过来时，当前采样点将被设置为新的免费时段。从图 4.2（b）、（c）可以看出，若当前采样点为免费时段，无论旧计费值和初始设定计费值之间大小关系如何，当某个流从其他边缘服务器迁入过来时，只需额外维护免费时段集合。从图 4.2（d）、（e）、（f）可以看出，若当前采样点为计费时段，当某个流从其他边缘服务器迁入过来导致当前采样点上的新采样值小于旧计费值或者小于初始设定计费值时，在额外维护计费时段集合的同时需要更新计费值，并且若当前采样点变成了免费时段，需要额外维护免费时段集合。

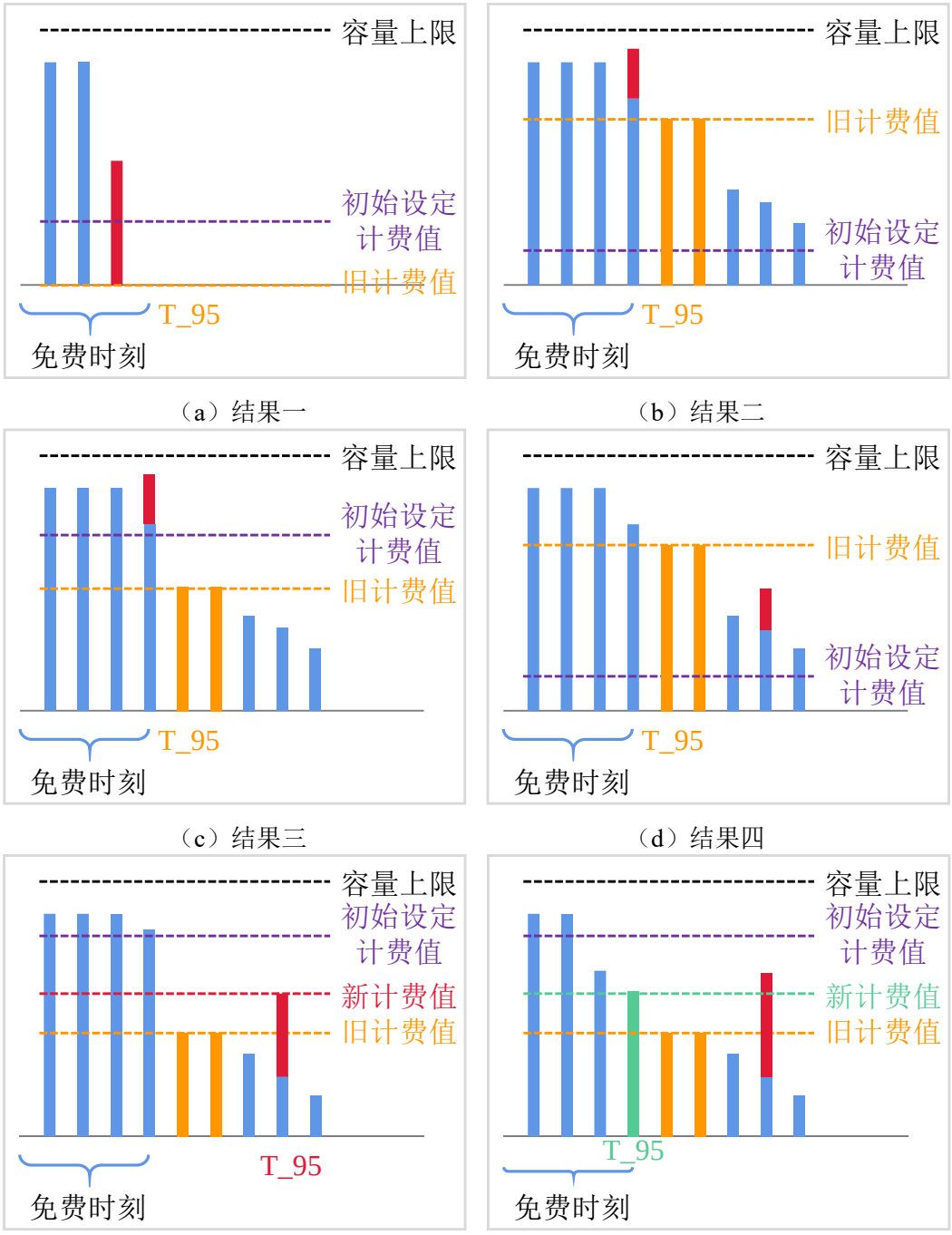


图 4.2 在不增加成本的情况下，迁入服务器的六种带宽序列变化结果

Fig.4.2 Six bandwidth sequence change results for the migrated server without increasing costs

图 4.3 给出了在增加成本的情况下，迁入服务器的五种带宽序列变化结果。图 4.3（a）对应原带宽序列长度为 0 即该服务器未开机的情况。从图 4.3（b）、（c）可以看出，若当前采样点为计费时段，无论旧计费值和初始设定计费值之间大小关系如何，当某个流从其他边缘服务器迁入过来后当前采样点还是计费时段时，

在额外维护计费时段集合的同时需要更新计费值。从图 4.3（d）、（e）可以看出，若当前采样点为计费时段，无论旧计费值和初始设定计费值之间大小关系如何，当某个流从其他边缘服务器迁入过来导致当前采样点变成免费时段时，需要在额外维护免费时段集合和计费时段集合的同时更新计费值。

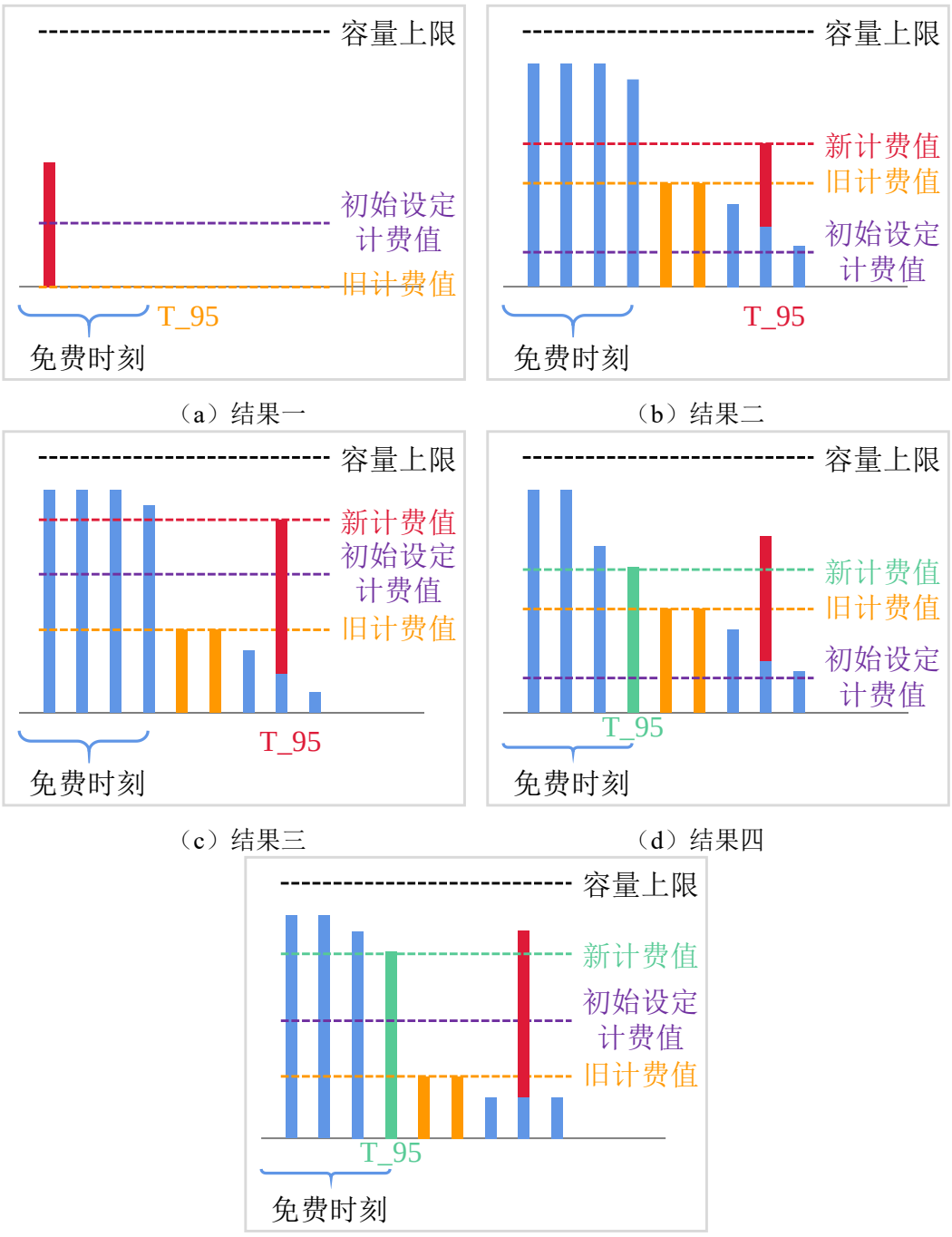


图 4.3 在增加成本的情况下，迁入服务器的五种带宽序列变化结果

Fig.4.3 Five bandwidth sequence change results for the migrated server while increasing costs

4.3.2 基于迁移交换的可行初始解生成算法

算法 4.1：基于迁移交换的可行初始解生成算法 *FISG*

输入：计费周期 T，边缘服务器集合 S，边缘用户集合 U，流集合 F，带宽容量上限集合 B，带宽需求矩阵 Y，网络时延矩阵 L，网络时延上限 τ，带宽单位成本 p

输出：带宽需求分配矩阵 X，总开销成本 C

```
/*根据 95 计费填充每个边缘服务器的免费时段*/
按连通性将 S 进行升序排序;
for i ← 1 to m do
    按剩余带宽需求总和将带宽序列进行降序排序;
    for k ← 1 to |T| − ⌈|T| × 0.95⌉ do
        D ← 获取与 s_i 连通的所有用户的流，以 (f_h, u_j) 形式保存，并按流的带宽需求进行降序排序;
        capacity ← b_i ;
        foreach (f_h, u_j) ∈ D do
            if capacity < y_jh^k then continue;
            x_ijh^k ← 1 ; capacity ← capacity − y_jh^k ;
/*分配剩余的流的带宽需求*/
Q_T ← 按剩余带宽需求总和将带宽序列进行降序排序;
while Q_T.empty() = False do
    k ← Q_T.poll() ;
    Q_F ← 获取 t_k 时段，所有用户上剩余的流，按流的带宽需求进行降序排序;
    while Q_F.empty() = False do
        (f_h, u_j) ← Q_F.poll() ;
        i ← 获取与 u_j 连通、能够接收 f_h、增加成本最低且负载率最低的边缘服务器;
        if i! = −1 then x_ijh^k ← 1 ; break;
        else
            i ← 选择某边缘服务器与 u_j 连通且容量上限大于 f_h 的带宽需求;
            将 s_i 上已分配的流随机释放直至可以接收 f_h 并更新 Q_F ;
            x_ijh^k ← 1 ; break;
    更新 Q_T ;
C ← 通过公式（4.4）计算其总开销成本;
return X, C ;
```

由于带宽需求的不可分割性，所以网络流算法在本节不再适用。对于 IEBA 算法的第一阶段，本节重新设计了一种基于迁移交换的可行初始解生成算法，如算法 4.1 所示。该算法主要包括两个步骤：（1）针对每个边缘服务器的带宽序列中的免费时段，为其尽可能多地分配带宽需求。这里本节设计了三种排序策略，分别是边缘服务器按照其连通性升序排序，带宽序列按照剩余带宽需求总和降序排序以及对于候选待分配的流按照其带宽需求大小降序排序。因为连通性低的边缘服务器在后续的分配过程中更不容易满足条件，所以需要提前进行分配。并且出于节省带宽开销成本的考虑，那些带宽需求高的时段更适合作为带宽序列中的免费时段。同理，类似于装箱问题，带宽需求高的流在后续的分配过程中更不容易找到能够容纳的边缘服务器，所以也应该提前分配（如第 1-10 行所示）。（2）每次选择剩余带宽需求总和最高的采样时段，然后选择带宽需求最高的那个流，为其分配目标边缘服务器。目标边缘服务器需要与待分配的流连通且其剩余的带宽容量需要大于等于流的带宽需求大小。如果找到多个候选服务器，首先选择增加成本最小的，如果成本相同再选择负载率最低的边缘服务器，这是出于一种负载均衡的思路。如果此时找不到一个可以接收的边缘服务器，说明之前的分配过程出了问题。这时候通过将某个边缘服务器上已分配的流进行释放操作，直至可以接收该轮次中待分配的流（这相当于一种兜底机制）。更新存放待分配的流的队列以及存放采样时段的队列，直到两个队列全为空，也就是整个计费周期内所有用户的流的带宽需求都已被分配至目标服务器（如第 11-26 行所示）。该算法的时间复杂度为$O\left(|T|n^3\varphi^3 log\left(n\varphi\right)\right)$。但是由于队列中存放的元素数量会逐渐减少，对于每个采样时段也不是每次都要循环遍历所有剩余的流，对于每个待分配的流也不是每次都要运行兜底机制，所以实际的运行时间会快得多。

4.3.3 基于自适应大邻域搜索的持续迭代优化算法

在 IEBA 算法的第二阶段，本节延续上一章的思路，在新的问题模型中设计了一种基于自适应大邻域搜索的算法来对当前解进行持续迭代优化。同样，为了加快优化的收敛速度和避免陷入局部最优解，还引入了基于爬山的优化算法和基于模拟退火的优化算法。

如算法 4.2 所示，虽然这一阶段的整体框架与上一章相同，但是由于带宽需求的不可分割性，网络流算法在这一节不再适用，所以在各个模块的具体实现上通过流的迁移交换策略进行替换。此外，对于 IEBA-L 算法，在每个温度下会首先尝试将某些服务器关停，从而使得总带宽开销成本降低。这一部分将在下一节进行详细介绍。值得注意的是，当初始设定计费值V 为 0 时，服务器关停算法就没有执行的必要了。如果当前解的成本要优于最优解，则将当前解保存至最优解（如第 5-8 行所示）。

算法 4.2：基于自适应大邻域搜索的持续迭代优化算法

输入：初始解 X，初始开销成本 C，初始温度 Θ，外层循环迭代轮数 Γ，内层循环迭代轮数 Φ，温度衰减系数 α，执行模拟退火算法的频率 λ，初始设定计费值 V，带宽单位成本 p

输出：最优解 $\bar{X}$，最优开销成本 $\bar{C}$

1 $\hat{X},\hat{C} \leftarrow X,C$；//当前解

2 $\bar{X},\bar{C} \leftarrow \hat{X},\hat{C}$；//最优解

3 $\rho^- \leftarrow (1,1,\ldots,1)$；$\rho^+ \leftarrow (1,1,\ldots,1)$；

4 **for** $e \leftarrow 1$ *to* Γ **do**

5 　/*对于 IEBA-L 算法，在内循环搜索前会先运行服务器关停算法

6 　**if** $V > 0$ **then** $\hat{X},\hat{C} \leftarrow SS(\hat{X},\hat{C},V,p)$；

7 　**if** $\hat{C} < \bar{C}$ **then** $\bar{X},\bar{C} \leftarrow \hat{X},\hat{C}$；

8 　*/

9 　**for** $g \leftarrow 1$ *to* Φ **do**

10 　　根据公式（2.3）分别选出一个破坏算子 $d \in \Omega^-$ 和一个修复算子 $r \in \Omega^+$；

11 　　$\delta \leftarrow random(0,1)$；

12 　　**if** $\delta \geq \lambda$ **then** $\hat{X},\hat{C} \leftarrow HCO(\hat{X},\hat{C},d,r,\Theta,V,p)$；//基于爬山的优化算法

13 　　**else** $\hat{X},\hat{C} \leftarrow SAO(\hat{X},\hat{C},d,r,\Theta,V,p)$；//基于模拟退火的优化算法

14 　　**if** $\hat{C} < \bar{C}$ **then** $\bar{X},\bar{C} \leftarrow \hat{X},\hat{C}$；

15 　　根据公式（2.4）获取算子得分，并根据公式（2.5）更新 ρ^- 和 ρ^+；

16 　$\Theta \leftarrow \Theta \times \alpha$；

17 **return** $\bar{X},\bar{C}$；

给定一个经破坏算子挑选出来的边缘服务器 s_i，基于爬山的优化算法的目标是：在不增加其他边缘服务器的成本（可能增加实际计费值，但是由于实际计费值小于初始设定计费值，所以成本依旧保持不变）的前提下，尝试降低 s_i 的实际计费值至少 1Mbps。如算法 7 所示，基于爬山的优化算法主要包括三个步骤：（1）初始化，包括根据破坏算子挑选出待优化的边缘服务器 s_i 并计算其当前实际计费值，如果当前实际计费值不大于初始设定计费值 V，那么说明 s_i 没有优化的必要，直接返回当前解。否则，根据修复算子获取边缘服务器的优先级排序以及备份当前解，便于尝试失败的时候直接回滚（如第 1-4 行所示）。（2）在关于 s_i 的带宽序列中，找到满足采样值等于当前实际计费值的所有采样点，如果该采样点不是 s_i 的免费时段，则 s_i 需要下探的采样点计数加一（如第 5-8 行所示）。（3）遍历候选待优化的采样点，尝试将 s_i 在该采样时段上的某个流迁移到其他的边缘服务器上，只要有一个流迁移成功，则该采样点成功优化，cnt 计数减一。当 $cnt = 0$ 时，

说明 s_i 上的实际计费值可以成功下探至少 1Mbps，此时接受该邻域解，计算总带宽开销成本并返回。如果失败，从备份中回滚（如第 9-19 行所示）。该算法的时间复杂度为 $O\left(|T|^2 n\varphi\right)$。

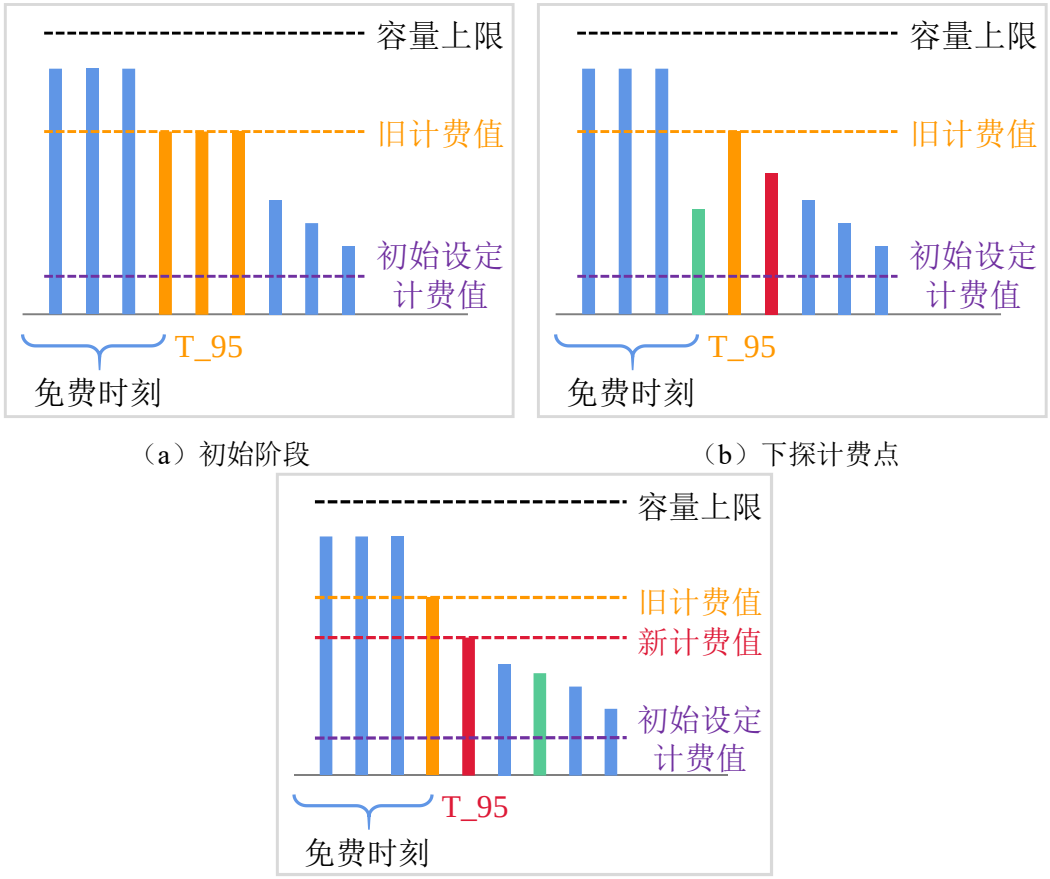
算法 4.3：基于爬山的优化算法 $HCO\left(\hat{X},\hat{C},d,r,\Theta,V,p\right)$

输入：当前解 $\hat{X}$，当前开销成本 $\hat{C}$，破坏算子 d，修复算子 r，当前温度 Θ，初始设定计费值 V，带宽单位成本 p

输出：当前解 $\hat{X}$，当前开销成本 $\hat{C}$

1 $s_i \leftarrow d$；$order \leftarrow r$；$cnt \leftarrow 0$；$G \leftarrow \varnothing$；
2 $q_i \leftarrow$ 通过公式（4.2）计算挑选出来的边缘服务器 s_i 目前的计费值；
3 **if** $q_i \le V$ **then return** $\hat{X},\hat{C}$；
4 $X,C \leftarrow \hat{X},\hat{C}$；//备份
5 **for** $k \leftarrow 1$ *to* $|T|$ **do**
6 　**if** $\sum_{j=1}^{n}\sum_{h=1}^{\varphi} x_{ijh}^{k} \times y_{jh} = q_i$ **then**
7 　　$G \leftarrow G \cup \{t_k\}$；
8 　　**if** t_k 属于 s_i 的计费时段 **then** $cnt \leftarrow cnt + 1$；
9 **foreach** $t_k \in G$ **do**
10 　$D \leftarrow$ 获取 $x_{ijh}^{k} = 1$ 的所有边缘用户的流；
11 　**foreach** $\left(f_h,u_j\right) \in D$ **do**
12 　　$result \leftarrow$ 按照 $order$ 依次尝试在不增加成本的情况下将 f_h 迁入其他边缘服务器；
13 　　**if** $result$ = **True then**
14 　　　$cnt \leftarrow cnt - 1$；
15 　　　break；
16 　**if** $cnt = 0$ **then**
17 　　$\hat{C} \leftarrow$ 通过公式（4.4）计算其总带宽开销成本；
18 　　**return** $\hat{X},\hat{C}$；
19 **return** X,C；

图 4.4 给出了一个目标边缘服务器在算法 7 中的优化过程。从图 4.4（a）看出该边缘服务器想要优化成功，需要下探 2 个采样点。图 4.4（b）选择若干个采样点进行了负载的下探。图 4.4（c）给出该边缘服务器成功优化后的带宽序列结果。



（a）初始阶段　　（b）下探计费点

（c）将带宽序列重新排序

图 4.4 目标边缘服务器在爬山算法中的优化过程

Fig.4.4 Optimization process of target edge server in hill climbing algorithm

如算法 4.4 所示，基于模拟退火的优化算法主要包括三个步骤：（1）初始化（如第 1-4 行所示）。（2）尝试在不增加总开销成本的情况下跳出局部最优。如果成功，计算总开销成本后直接返回（如第 5-14 行所示）。（3）如果在（2）步优化失败，则尝试在增加总开销成本的情况下跳出局部最优。若新的邻域解可行且总成本根据 Metropolis 准则判断可以接受，则返回该邻域解。如果尝试失败，从备份中回滚（如第 15-26 行所示）。该算法的时间复杂度为 $O\left(|T|^2 n\varphi\right)$。因此，所提出的 IEBA 算法的总时间复杂度为 $O\left(|T|n^3\varphi^3 log\left(n\varphi\right)+\Gamma\Phi|T|^2 n\varphi\right)$。

图 4.5 给出了一个待优化的目标边缘服务器在算法 4.4 中的优化过程。

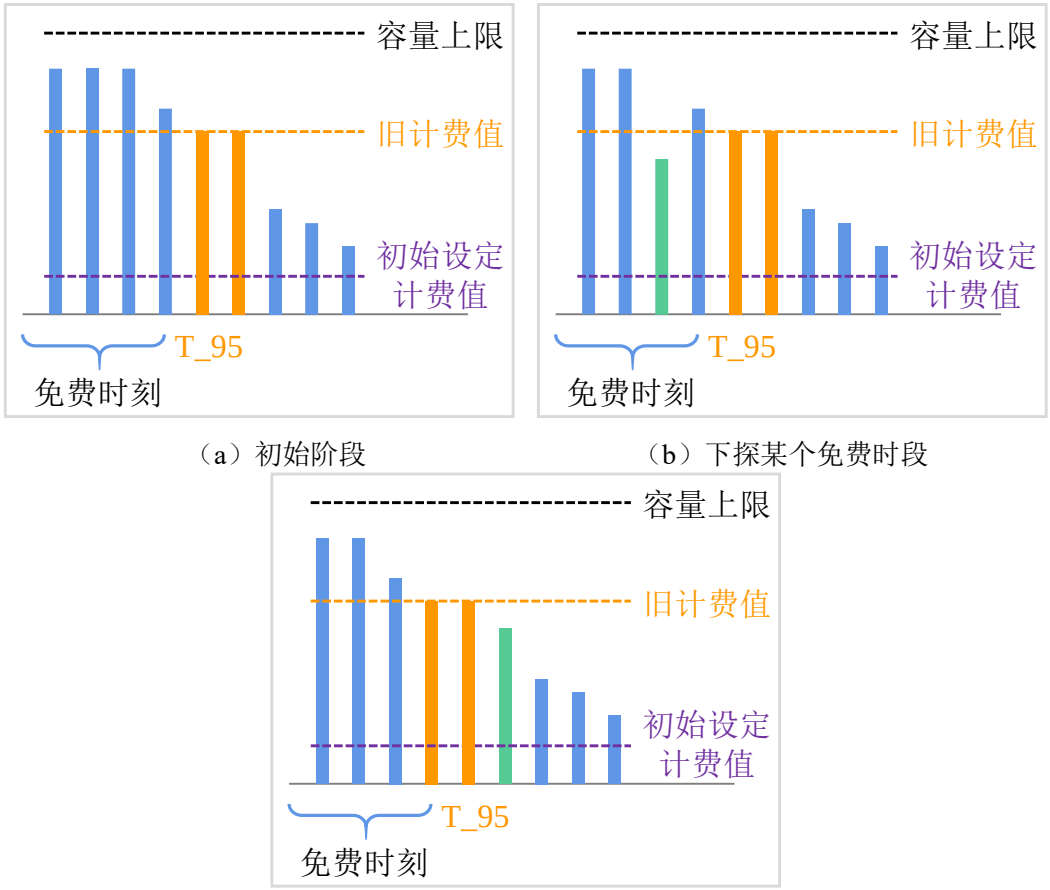
算法 4.4：基于模拟退火的优化算法 $SAO\left(\hat{X},\hat{C},d,r,\Theta,V,p\right)$

输入：当前解 $\hat{X}$，当前开销成本 $\hat{C}$，破坏算子 d，修复算子 r，当前温度 Θ，初始设定计费值 V，带宽单位成本 p

输出：当前解 $\hat{X}$，当前开销成本 $\hat{C}$

算法 4.4：基于模拟退火的优化算法 $SAO\left(\hat{X},\hat{C},d,r,\Theta,V,p\right)$

$s_i \leftarrow d$； $order \leftarrow r$；
$q_i \leftarrow$ 通过公式（4.2）计算挑选出来的边缘服务器 s_i 目前的计费值；
if $q_i \le V$ **then return** $\hat{X},\hat{C}$；
$X,C \leftarrow \hat{X},\hat{C}$；
/*尝试在不增加成本的情况下跳出局部最优*/
foreach t_k 属于 s_i 的免费时段 **do**
    $D \leftarrow$ 获取 $x_{ijh}^k = 1$ 的所有边缘用户的流；
    foreach $\left(f_h,u_j\right) \in D$ **do**
        $result \leftarrow$ 按照 $order$ 依次尝试在不增加成本的情况下将 f_h 迁入其他边缘服务器；
        if $result$ = **True then**
            if $\sum_{j=1}^{n}\sum_{h=1}^{\varphi} x_{ijh}^k \times y_{jh} < q_i$ **then**
                $\hat{C} \leftarrow$ 通过公式（4.4）计算其总开销成本；
                return $\hat{X},\hat{C}$；
    $\hat{X},\hat{C} \leftarrow X,C$； //从备份中回滚
/*尝试在增加成本的情况下跳出局部最优*/
foreach t_k 属于 s_i 的免费时段 **do**
    $D \leftarrow$ 获取 $x_{ijh}^k = 1$ 的所有边缘用户的流； $\delta \leftarrow random(0,1)$；
    foreach $\left(f_h,u_j\right) \in D$ **do**
        $result \leftarrow$ 按照 $order$ 依次尝试在增加成本的情况下将 f_h 迁入其他边缘服务器；
        if $result$ = **True then**
            if $\sum_{j=1}^{n}\sum_{h=1}^{\varphi} x_{ijh}^k \times y_{jh} < q_i$ **then**
                $\hat{C} \leftarrow$ 通过公式（4.4）计算其总开销成本；
                if $\delta < exp\left(\left(C-\hat{C}\right)/\Theta\right)$ **then return** $\hat{X},\hat{C}$；
                else break；
    $\hat{X},\hat{C} \leftarrow X,C$； //从备份中回滚
return X,C；



（a）初始阶段　　（b）下探某个免费时段

（c）将带宽序列重新排序

图 4.5 目标边缘服务器在模拟退火算法中的优化过程

Fig.4.5 Optimization process of target edge server in simulated annealing algorithm

4.4 IEBA-L 算法

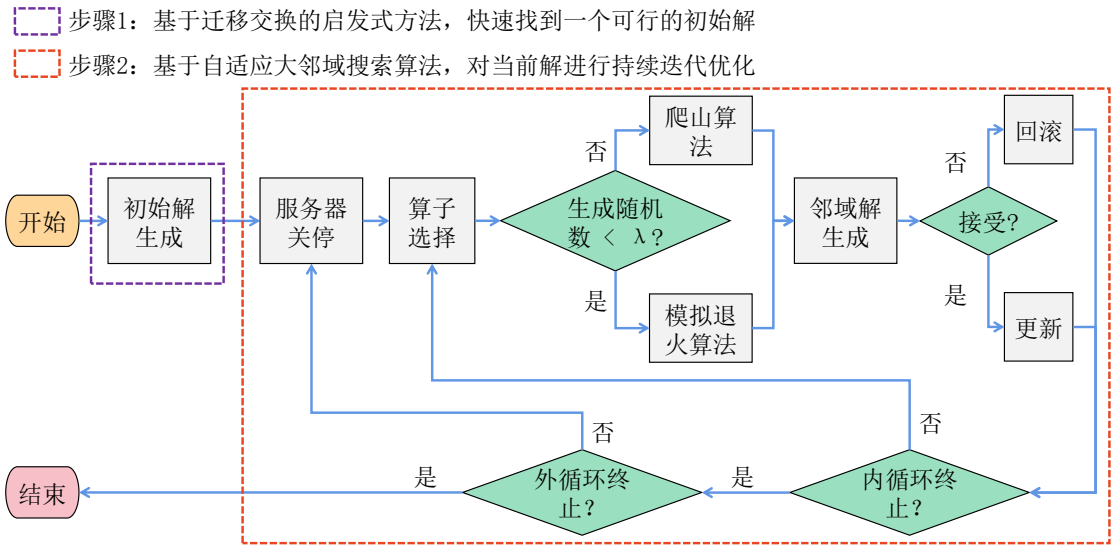


图 4.6 IEBA-L 算法的执行流程

Fig.4.6 The process of the proposed IEBA-L algorithm

本节在 IEBA 算法的基础上，结合服务器关停算法，提出了 IEBA-L 算法，用于在带宽需求较低的场景中进一步节省总带宽开销成本。其流程步骤如图 4.6 所示。

算法 4.5：服务器关停算法 $SS\left(\hat{X},\hat{C},V,p\right)$

输入：当前解 $\hat{X}$ ，当前开销成本 $\hat{C}$ ，初始设定计费值 V ，带宽单位成本 p

输出：当前解 $\hat{X}$ ，当前开销成本 $\hat{C}$

1 $X,C \leftarrow \hat{X},\hat{C}$ ；//备份

2 $Q_S \leftarrow$ 将边缘服务器按照整个计费周期内已分配的总带宽需求升序排序；

3 **while** $Q_S.empty() = \textbf{\textit{False}}$ **do**

4 　$i \leftarrow Q_S.poll()$ ；

5 　$D \leftarrow$ 获取与 s_i 连通的所有用户的流，以 $\left(f_h,u_j\right)$ 形式保存，并按流的带宽需求进行降序排序；

6 　$total\_cost \leftarrow 0$ ； $is\_success \leftarrow$ **True**；

7 　**foreach** $\left(f_h,u_j\right) \in D$ **do**

8 　　$result,cost \leftarrow$ 尝试将 f_h 迁移至其他边缘服务器；

9 　　**if** $result =$ **True and** $total\_cost + cost < V \times p$ **then**

10 　　$total\_cost \leftarrow total\_cost + cost$ ；

11 　　continue；

12 　　**else** $is\_success \leftarrow$ **False**；

13 　**if** $is\_success =$ **True then**

14 　　将 s_i 关停；

15 　　$X,C \leftarrow \hat{X},\hat{C}$ ；//更新备份

16 　**else** $\hat{X},\hat{C} \leftarrow X,C$ ；//从备份中回滚

17 　更新 Q_S ；

18 **return** $\hat{X},\hat{C}$ ；

对于服务器关停算法，其目标是通过尽可能多地关停某些边缘服务器，从而节省这些边缘服务器的开机成本。如算法 4.5 所示，其主要步骤如下：

（1）备份当前解，使得迁移失败时能够直接从备份中回滚恢复（如第 1 行所示）。

（2）依次尝试能否将某个边缘服务器关停，这里按照边缘服务器在整个计费周期内已分配的流的总带宽需求升序排序。因为无论关停哪一个边缘服务器，其最大的收益都为 $V \times p$ ，即服务器的开机成本，但是总带宽需求少的边缘服务器更容易关停成功。在确定待优化的目标边缘服务器之后，需要对整个计费周期内已

分配的流进行重新分配，这里按照流的带宽需求大小降序排序，能够减少运行的次数，提升效率。因为带宽需求大的流更容易出现迁移失败的情况，并且一旦出现某个流迁移失败的情况，剩余的流就没有继续尝试的必要（如第 2-5 行所示）。

（3）对于流的迁移操作，有可能会导致迁入的边缘服务器的带宽开销成本增大。所以对于待优化目标边缘服务器，其成功的判断依据为：整个计费周期内已分配的流都能成功地迁入其他边缘服务器，并且其他边缘服务器可能增加的带宽开销成本总和小于服务器关停后节省的开机成本。如果关停成功，则更新服务器状态以及重新备份当前解。如果关停失败，则从备份中重新回滚。更新 Q_S 中剩余的其他边缘服务器上的负载，然后继续尝试（如第 6-18 行所示）。该算法的时间复杂度为 $O\left(|T|nm\varphi log\ m\right)$。

4.5 实验结果与分析

在本节，基于上一章采样到的真实边缘带宽需求数据集，根据流的数量和带宽需求矩阵的方差生成多个数据集，并在这多个数据集上进行了一系列的实验，验证提出的 IEBA 算法和 IEBA-L 算法在各自场景中的有效性。

4.5.1 实验数据

由于本章引入了流的概念，所以需要将上一章中每个采样点上的边缘用户的带宽需求拆分到各个流中。本节根据如下方式生成关于流的带宽需求数据集：对于每个待拆分的带宽需求 y_j^k，首先通过随机数生成一个整数数组 $A=\left\{a_1,a_2,a_3,...,a_\varphi\right\}$，其中 φ 表示流的数量，随机数生成过程中的 lower_bound 和 upper_bound 用于控制数组 A 和带宽需求矩阵的方差。

对于前 $\varphi-1$ 个流，其分配到的带宽需求如下所示：

$$y_{jh}^k=\left\lfloor y_j^k\times a_h/\left(\sum_{g=1}^{\varphi}a_g\right)\right\rfloor \tag{4.6}$$

对于第 φ 个流，其分配到的带宽需求如下所示：

$$y_{j\varphi}^k=y_j^k-\sum_{h=1}^{\varphi-1}y_{jh}^k \tag{4.7}$$

因此，本节使用到的数据集如表 4.2 所示。其中，方差小表示 lower_bound 和 upper_bound 分别取值为 1、1，方差中表示 lower_bound 和 upper_bound 分别取值为 1、10，方差大表示 lower_bound 和 upper_bound 分别取值为 1、100。对于这九个数据集，初始设定计费值均设置为 800，其他的设置与上一章中的实验部分保持一致。

表 4.2 数据集描述

Table 4.2 Datasets description

名称	流的数量φ	方差
数据集一	20	小
数据集二	20	中
数据集三	20	大
数据集四	30	小
数据集五	30	中
数据集六	30	大
数据集七	50	小
数据集八	50	中
数据集九	50	大

4.5.2 模型参数设置

本节给出一些重要的超参数的设置结果，如表 4.3 所示。

表 4.3 超参数的设置

Table 4.3 Setting of hyper-parameters

名称	数值
Θ	2048
α	0.97
σ	0.3
λ	0.33
Γ	200
Φ	200
ω_1	1.0
ω_2	0.75
ω_3	0.5
ω_4	0.25

4.5.3 对比算法与评价指标

（1）对比算法

由于目前已有的文献中的算法并不能直接应用于本章提出的问题模型，因此，本节将常见的分配算法比如 Load Balance[53]、Round Robin[54]以及装箱算法 First Fit、Best Fit[57]结合本章问题场景稍作修改后作为对比算法。因为在带宽需求不可分割场景下的 Round Robin 算法与装箱算法中的 Next Fit 比较类似，Load Balance 算法与 Worst Fit 比较类似，所以本节就不再将 Next Fit 和 Worst Fit 加入到对比算法中。下面将对这些对比算法进行详细介绍：

Load Balance（LB）：针对计费周期内每个流的带宽需求，LB 将会启发式地随机选择 d 个连通的候选边缘服务器。值得注意的是，考虑到 95 计费的特点，当前采样点是其免费时段的边缘服务器会被优先选择。然后，整个流的带宽需求将会被分配到负载最低的候选边缘服务器。如果当前不存在能够分配的边缘服务器，则随机选择一个连通且带宽容量上限大于待分配的流的带宽需求的目标边缘服务器，释放目标边缘服务器上已分配的流，直至能够接收待分配的流（也称为兜底机制）。

Load Balance-95（LB-95）：LB 算法去除 95 计费模块，此时就变成了与问题无关的原始 LB 算法。

Round Robin（RR）：针对计费周期内每个流的带宽需求，RR 将会以轮询的方式选择其要分配给的目标边缘服务器。结合 95 计费的特点，优先选择当前采样点是其免费时段的边缘服务器。如果当前不存在能够分配的边缘服务器，加入兜底机制。

Round Robin-95（RR-95）：RR 算法去除 95 计费模块。

First Fit（FF）：针对计费周期内每个流的带宽需求，FF 将会以一个固定的顺序（按照数据集中的输入顺序）寻找目标边缘服务器，将流分配给第一个能够接收的边缘服务器。同时结合 95 计费的特点和兜底机制。

First Fit-95（FF-95）：FF 算法去除 95 计费模块。

Best Fit（BF）：针对计费周期内每个流的带宽需求，BF 会把流分配给该时段剩余带宽容量最小的边缘服务器。同时结合 95 计费的特点和兜底机制。

Best Fit-95（BF-95）：BF 算法去除 95 计费模块。

FISG：本章 4.3.2 节提出的一种基于迁移交换的可行初始解生成算法，能够充分利用 95 计费模式的特点，降低总带宽开销成本。

（2）评价指标

本节选择计费周期内所有边缘服务器的总带宽开销成本作为最终的评价指标。

4.5.4 性能对比分析

不同数据集下的对比实验结果：本节比较了在 9 个不同的数据集下，各个算法获得的最终解的质量，如表 4.4 所示。其中，加粗的数字表示本次实验中的最优

结果，带下划线的数字表示次优结果。对于 LB、RR、FF、BF 算法而言，在选择目标边缘服务器的时候将 95 计费考虑在内能够极大地降低总带宽开销成本。其中，LB 算法的总成本大致为 LB-95 算法的 43.95%，RR 算法的总成本大致为 RR-95 算法的 66.80%，FF 算法的总成本大致为 FF-95 算法的 74.13%，BF 算法的总成本大致为 BF-95 算法的 69.89%。在考虑 95 计费的前提下，无论数据集如何变化，BF、FF、RR、LB 算法始终为总成本最高的前四位。本章提出的 FISG 算法能够获得一个质量不错的初始解，其总成本大致为 LB 算法的 17.25%，为 RR 算法的 10.69%，为 FF 算法的 6.26%，为 BF 算法的 5.73%。此外，本章提出的 IEBA 算法和 IEBA-L 算法能够在 FISG 的基础上分别平均下降 21.02%和 19.50%。IEBA-L 算法在数据集七、数据集八和数据集九上取得了最优的结果，而 IEBA 算法在其余六个数据集上取得最优。最后，对于流数量相同的数据集，随着带宽需求矩阵方差的增大，各个算法的总成本大致成上升趋势，如数据集一二三组、四五六组和七八九组。对于带宽需求矩阵方差相同的数据集，随着流数量的增加，各个算法的总成本大致成下降趋势，如数据集一四七组、二五八组和三六九组。

表 4.4 不同数据集下的对比实验结果（单位：元）

Table 4.4 Comparative experimental results under different datasets (Yuan)

数据集	一	二	三	四	五	六	七	八	九
LB	5.58e7	6.78e7	7.04e7	5.44e7	5.31e7	5.34e7	5.22e7	5.11e7	5.10e7
LB-95	1.29e8	1.29e8	1.30e8	1.28e8	1.29e8	1.29e8	1.28e8	1.28e8	1.28e8
RR	9.63e7	1.09e8	1.10e8	7.92e7	8.64e7	8.74e7	8.19e7	8.44e7	8.62e7
RR-95	1.29e8	1.43e8	1.46e8	1.29e8	1.39e8	1.41e8	1.29e8	1.36e8	1.37e8
FF	1.55e8	1.55e8	1.55e8	1.55e8	1.56e8	1.55e8	1.55e8	1.55e8	1.55e8
FF-95	2.09e8	2.09e8	2.10e8	2.09e8	2.10e8	2.10e8	2.09e8	2.10e8	2.10e8
BF	1.69e8	1.69e8	1.69e8	1.70e8	1.70e8	1.70e8	1.69e8	1.69e8	1.69e8
BF-95	2.42e8	2.42e8	2.42e8	2.42e8	2.43e8	2.43e8	2.43e8	2.43e8	2.43e8
FISG	1.03e7	1.09e7	1.11e7	9.22e6	9.60e6	9.91e6	8.69e6	8.83e6	8.83e6
IEBA-L	7.41e6	9.07e6	9.83e6	7.23e6	7.89e6	8.00e6	**7.04e6**	**7.02e6**	**7.01e6**
IEBA	**7.40e6**	**8.72e6**	**9.31e6**	**7.21e6**	**7.55e6**	**7.75e6**	**7.04e6**	7.04e6	7.04e6

分析：由于本章的问题模型建立在 95 计费模式的基础上，所以在算法中结合 95 计费的特点，优先选择当前时段为免费时段的那些边缘服务器能够有效地降低总成本。在装箱问题中，Best Fit 算法一般要比 First Fit 算法更优，因为能够产生更少的内部碎片。然而在本章的问题模型中，却产生了相反的结果，这是因为 BF

会导致每个边缘服务器都存在若干个几乎满载的时段。由于本章使用的边缘带宽需求数据集在每个采样时段上的需求都比较高，这导致了 BF 算法中的边缘服务器的满载时段数量远远超过带宽序列的 5%，造成每个边缘服务器的计费值都很高。而 FF 算法以一个固定的顺序去寻找目标边缘服务器，使得每个边缘服务器上的带宽序列都比较平均，能够被充分利用，尽管顺序靠前的边缘服务器的计费值依旧很高。RR 算法类似于 Next Fit 算法，通过目标边缘服务器的轮换，避免了某几个边缘服务器的计费值偏高的现象。LB 算法类似于 Worst Fit 算法，通过负载均衡进一步让各个边缘服务器的计费值趋于平均。而本章提出的 FISG 算法能够充分挖掘 95 计费的特点，在免费时段让边缘服务器的负载尽可能高，在计费时段让边缘服务器的负载尽可能平均，因此获得了一个比较好的初始解。由于引入了自适应大邻域搜索算法、爬山算法和模拟退火算法，因此 IEBA 算法和 IEBA-L 算法能够在 FISG 的基础上，实现总成本的进一步下探。最后，对于流数量相同的数据集，随着带宽需求矩阵方差的增大，带宽需求偏高的流的数量也随之增多，这些流的带宽需求更不容易被优化，最终导致边缘服务器的计费值增加。而对于带宽需求矩阵方差相同的数据集，随着流数量的增加，由于每个采样时段上的总带宽需求保持不变，因此每个流的带宽需求随之降低，在流的分配过程中更不容易造成成本的增加。

表 4.5 IEBA 算法的消融实验结果（单位：元）

Table 4.5 Ablation results of IEBA algorithm (Yuan)

数据集	一	二	三	四	五	六	七	八	九
IEBA	**7.40e6**	**8.72e6**	**9.31e6**	**7.21e6**	**7.55e6**	**7.75e6**	**7.04e6**	**7.04e6**	**7.04e6**
Full-ALNS	8.70e6	9.22e6	9.49e6	7.87e6	8.26e6	8.47e6	7.45e6	7.61e6	7.63e6
Full-SA	1.01e7	1.09e7	1.11e7	9.14e6	9.60e6	9.91e6	8.60e6	8.83e6	8.83e6

表 4.6 IEBA-L 算法的消融实验结果（单位：元）

Table 4.6 Ablation results of IEBA-L algorithm (Yuan)

数据集	一	二	三	四	五	六	七	八	九
IEBA-L	**7.41e6**	**9.07e6**	9.83e6	**7.23e6**	**7.89e6**	**8.00e6**	**7.04e6**	**7.02e6**	**7.01e6**
Full-ALNS	8.63e6	9.24e6	**9.49e6**	7.94e6	8.15e6	8.58e6	7.56e6	7.61e6	7.70e6
Full-SA	1.01e7	1.09e7	1.11e7	9.04e6	9.60e6	9.91e6	8.60e6	8.83e6	8.83e6

不同数据集下的消融实验结果：为了验证各个模块的有效性，本节对 IEBA 算

法和 IEBA-L 算法中的各个模块分别去除后进行了消融实验，实验结果如表 4.5 和表 4.6 所示。可以看出，无论在去除 SA 模块还是 ALNS 模块后，IEBA 算法和 IEBA-L 算法的总成本都出现了明显的增加。其中，去除 ALNS 模块后，IEBA 算法的总成本平均增加 8.39%，IEBA-L 算法的总成本平均增加 6.76%。去除 SA 模块后，IEBA 算法的总成本平均增加 26.16%，IEBA-L 算法的总成本平均增加 23.74%。

分析：因为 ALNS 算法采用了多种修复算子和破坏算子，并且引入了自适应选择这一特征，使得邻域搜索的空间变得更大，更有机会找到质量更优的邻域解。而 SA 算法具有跳出局部最优的特点，所以无论去除哪一个模块，IEBA 算法和 IEBA-L 算法获得的最终解的质量几乎都出现了下降。对于表 4.6 所示的数据集三上的实验，由于 SA 模块有一定的概率接收质量更差的邻域解，导致 IEBA-L 算法的总成本反而比 Full-ALNS 要高，但是这是极少数的个例情况。

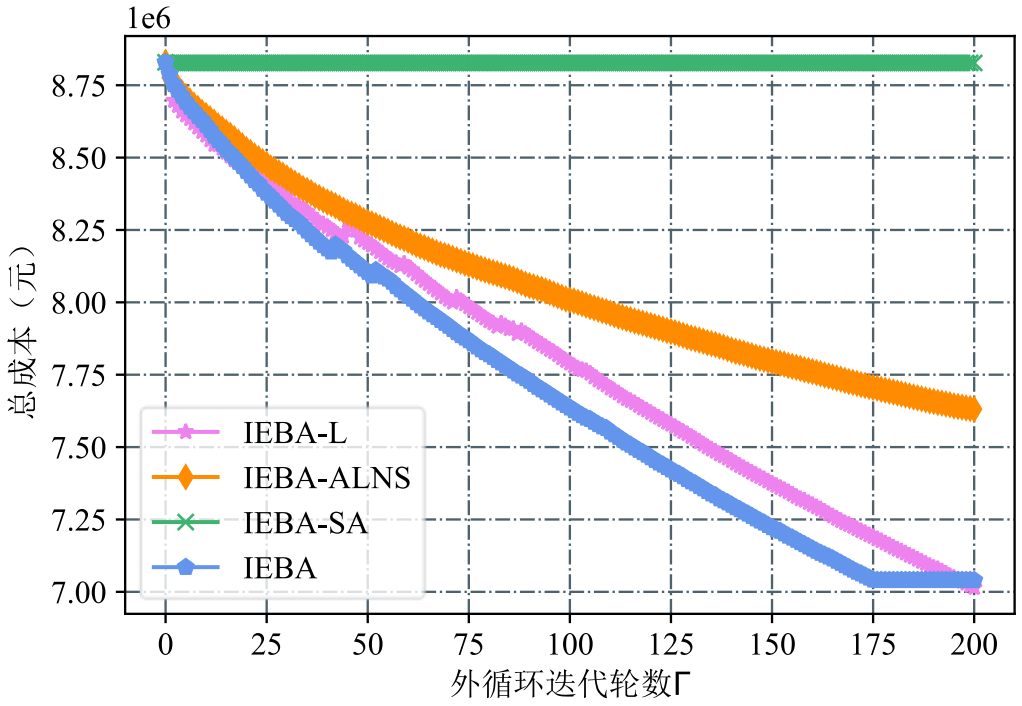


图 4.7 总成本优化过程

Fig.4.7 Total cost optimization process

以 IEBA 算法在数据集九上的消融实验为例，图 4.7 给出了各种算法的总成本优化过程。可以看出，无论是去除 ALNS 模块还是 SA 模块，不但导致最终解的质量下降，而且会减缓解的优化速度。此外，相比于 IEBA-L 算法，IEBA 算法由于去除了服务器关停模块，使得在解的优化过程中可用的边缘服务器数量更多，进

而加快了解的优化速度，更快地达到收敛。而 IEBA-L 算法能够获得更优的解，并且随着迭代轮数的增加，解的质量还能够进一步改善。

但是，如表 4.4 所示，在本节的 9 个数据集中，在 6 个数据集上的结果显示 IEBA 算法要比 IEBA-L 算法更优，而 IEBA-L 算法只在另外的 3 个数据集上取得略微优势，并且由图 4.7 可知 IEBA 算法的优化速度要快于 IEBA-L 算法。于是，为了验证本章提出的 IEBA-L 算法的作用，本节比较了在不同的带宽需求下 IEBA 算法和 IEBA-L 算法各自的总成本和运行边缘服务器数量，如图 4.8 和图 4.9 所示。

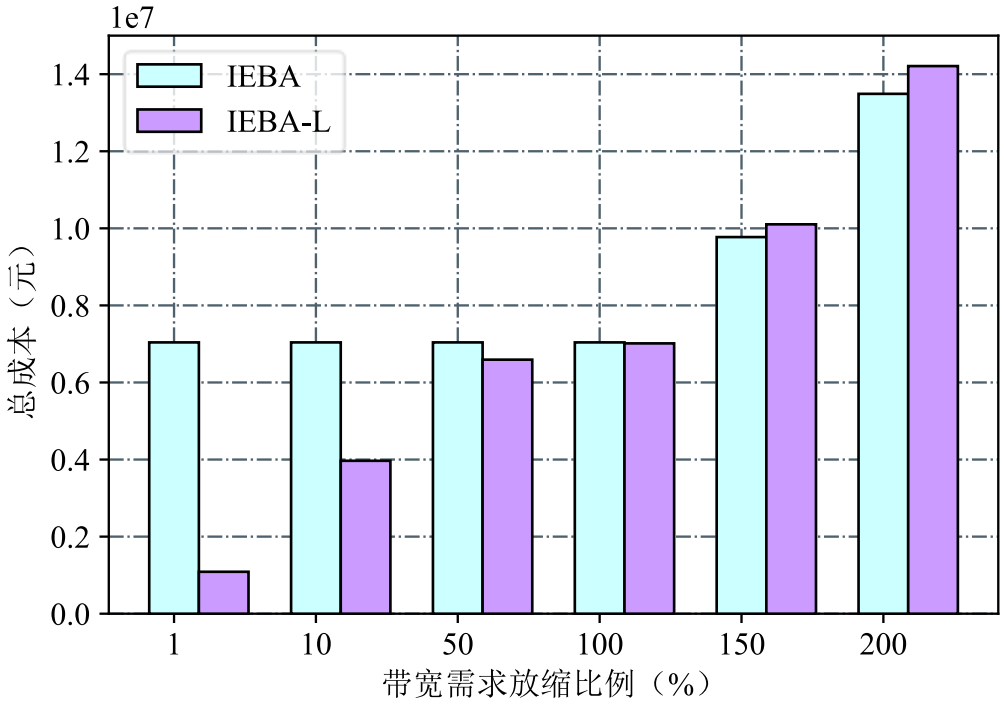


图 4.8 不同带宽需求下的总成本比较

Fig.4.8 Comparison of total cost under different bandwidth demands

可以看出，无论是 IEBA 算法还是 IEAB-L 算法，随着带宽需求的增加，总成本都呈现出上升的趋势。在带宽需求较低的场景中，IEBA-L 算法由于增加了服务器关停算法，能够有效地降低运行服务器数量，从而降低总带宽开销成本。而在 IEBA 算法中由于所有的服务器始终保持开机状态，导致总成本较高。在带宽需求较高的场景中，IEBA 算法由于可用的服务器数量更多，优化的成功率更高，解的质量反而更优，并且随着带宽需求的进一步增加，优势反而更加明显。由于 IEBA 算法与 IEBA-L 算法唯一的区别就在于是否增加了服务器关停算法，因此可以看出，本章提出的服务器关停算法更适合应用于带宽需求较低的场景。

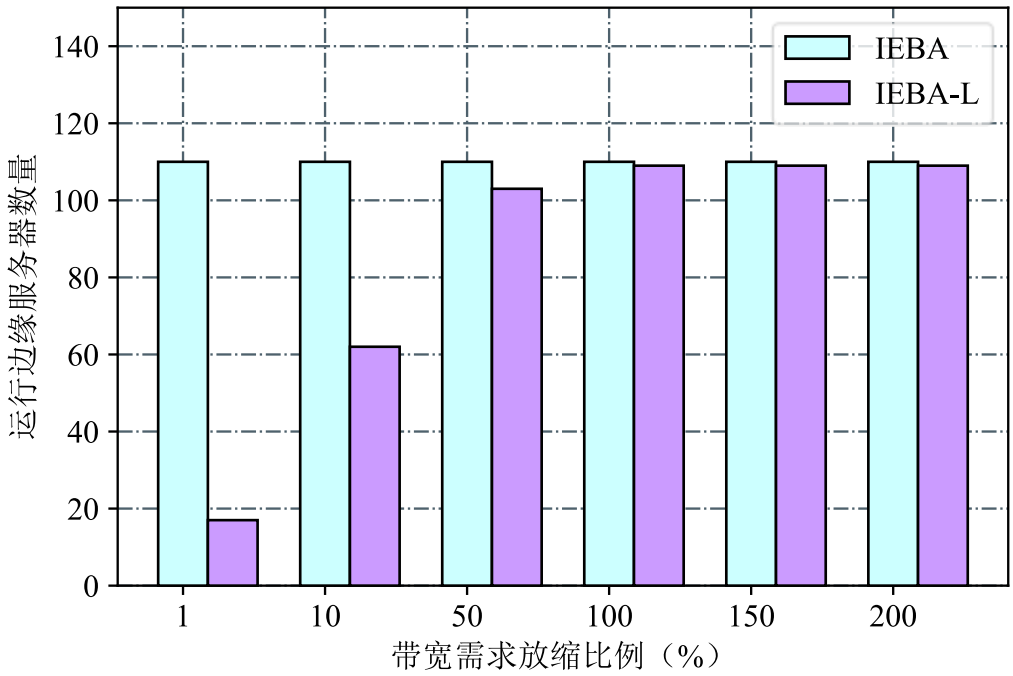


图 4.9 不同带宽需求下的运行边缘服务器数量比较

Fig.4.9 Comparison of the number of running edge servers under different bandwidth demands

此外，以数据集九为例，本节比较了在不同的计费周期下 IEBA 算法的总成本与平均迭代次数，如图 4.10 所示。在实验的设置中，算法的外层循环迭代轮数最高为 200 轮，当总成本连续 5 轮保持不变时则退出优化过程。以计费周期为 12 小时为例，IEBA 算法针对每 12 小时的边缘带宽需求数据集给出一个解决方案，将 60 个这样的子方案按时间顺序拼接得到最终一个月的分配结果。以此类推，当计费周期为 1 天时，则需要运行 30 次 IEBA 算法；当计费周期为 1 周时，总共需要运行 4 次 IEBA 算法。显然，由于每个子分配方案都是一个可行解，拼接得到的最终解也是满足公式（4.5）中的一系列约束条件。

从图 4.10 可以看出，随着计费周期的延长，最终解的总成本呈现下降的趋势，而算法的平均迭代次数呈现上升的趋势。当计费周期为 12 小时时，解的质量迅速变差，这是因为带宽需求往往以天为单位呈现出周期性波动的特征，如 3.4.1 节所述。此时，若以 12 小时为周期运行 IEBA 算法，则会导致一天内无论是高需求还是低需求都有可能分配至同一批边缘服务器，而对剩下的边缘服务器造成了资源的浪费，最终导致总成本的上升。此外，由 4.3.3 节可知，IEBA 算法的优化部分的时间复杂度与平均迭代次数成正比，与计费周期长度的平方成正比。因此当选择计费周期为 1 天时，不仅能获得一个质量较好的最终解，还能够极大地缩短运行时间，使得 IEBA 算法应用于实际场景中成为了可能。具体而言，在离线场景中，可以通过并行化技术使得在较短的时间内能够获得一个质量不错的解。在在线场

景中，现有的研究往往去预测未来整个月的带宽需求，导致预测的精度不高，最终影响解的质量。现如今只需去预测比如未来一天的带宽需求，不仅提升了预测的准确度，改善了解的质量，而且提高了算法的运行效率。这也是本文未来的研究方向。

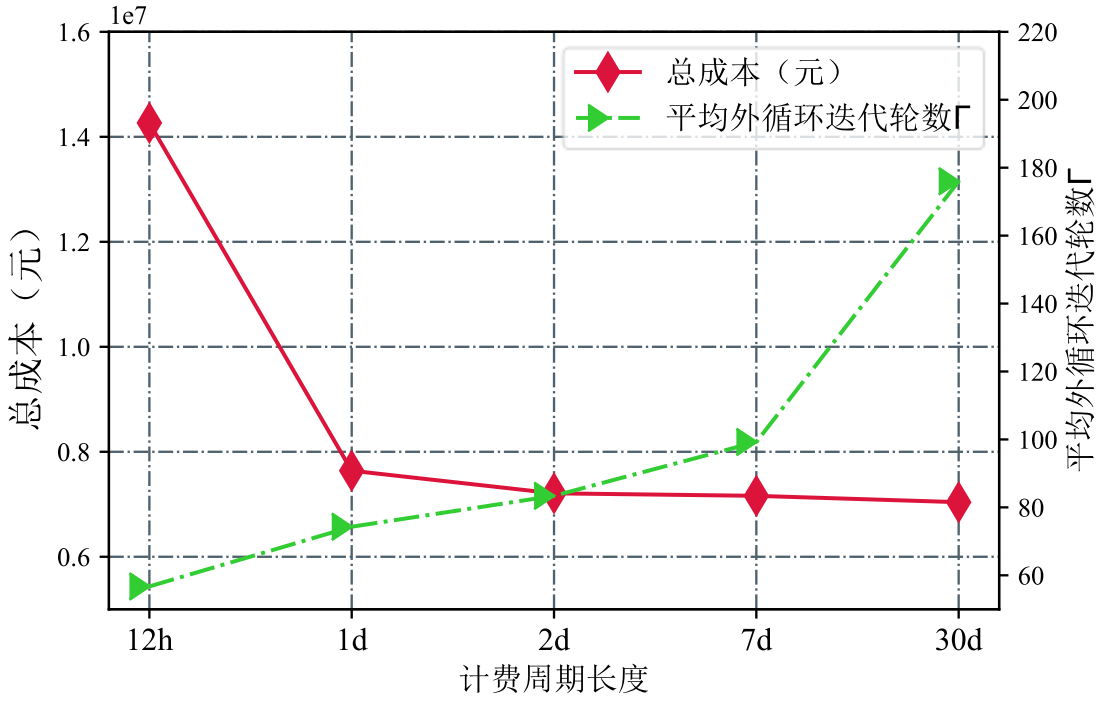
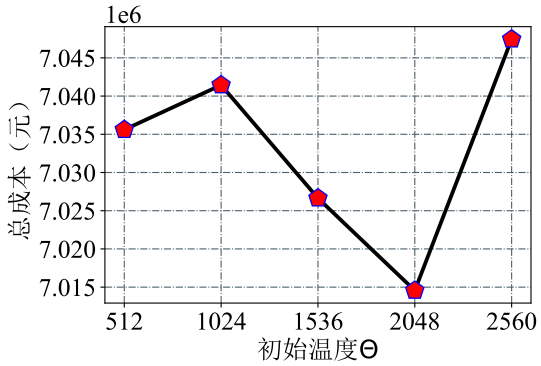


图 4.10 不同计费周期下算法的总成本与平均迭代次数

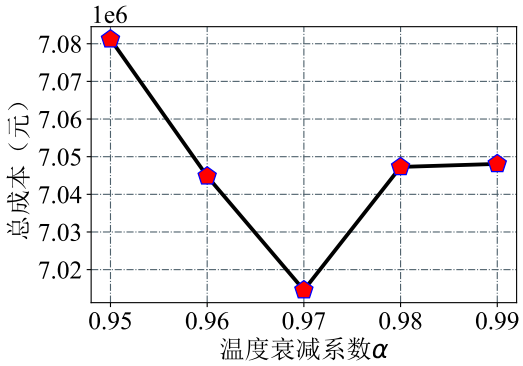
Fig.4.10 The total cost and average number of outer loops under different billing periods

4.5.5 参数敏感性实验

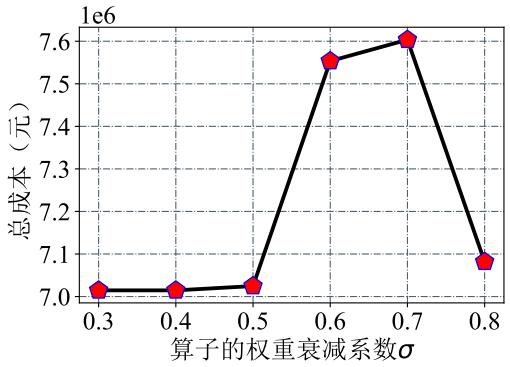
本节同样研究了一些重要的超参数对实验结果的影响，如图 4.11 所示。可以看出，IEBA 算法对模拟退火的初始温度、温度衰减系数这两个参数较不敏感，虽然算法的总成本随着这两个参数的取值变化而变化，但是波动范围较小。而算子的权重衰减系数、模拟退火算法的执行频率、外循环迭代次数、内循环迭代次数这四个参数对实验结果的影响较大。其中，算子的权重衰减系数直接决定了自适应大邻域搜索过程中算子的选择过程，权重衰减系数越低，自适应大邻域搜索起到的作用越明显。此外，无论是外循环次数还是内循环次数，随着次数的增多，对邻域解的搜索也变得更加充分，解的质量也随之提升。由 3.4.5 节可知，由于 ω_1、ω_2、ω_3、ω_4 这四个参数的大小关系保持不变，所以算法对这四个参数并不敏感，故本节不再进行重复实验。并且从图 4.11 可以看出，本节的实验参数设置都是较优的选择。



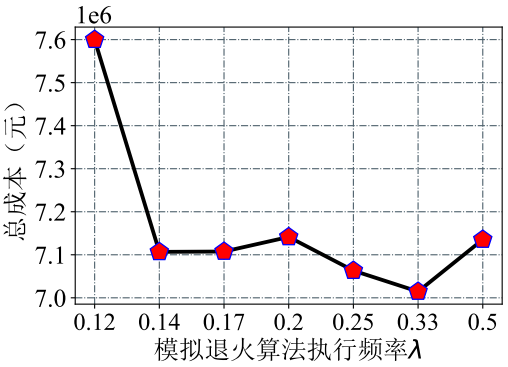
（a）初始温度Θ对结果的影响



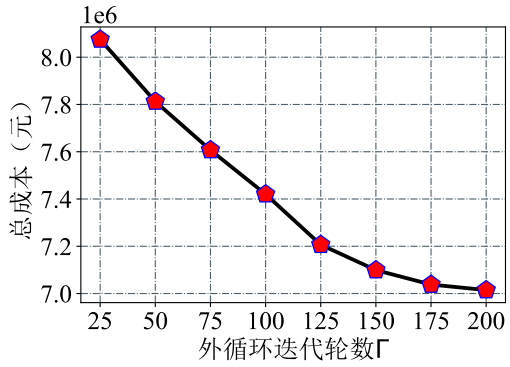
（b）温度衰减系数α对结果的影响



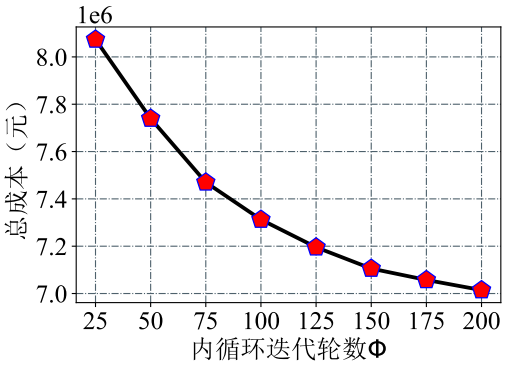
（c）算子的权重衰减系数σ对结果的影响



（d）模拟退火算法的执行频率λ对结果的影响



（e）外循环次数Γ对结果的影响



（f）内循环次数Φ对结果的影响

图 4.11 实验中一些重要参数的敏感性结果

Fig.4.11 Sensitivity results of some important parameters in the experiment

4.6 本章小结

针对带宽需求不可分割假设下的边缘带宽分配问题，本章首先构建了问题模型并给出其形式化定义。其次，通过引入流的概念，将流作为带宽需求分配调度的最小单位。接着，提出 IEBA 算法和 IEBA-L 算法分别针对高需求场景和低需求场景对问题进行了求解。其中，IEBA-L 算法通过在 IEBA 算法中引入服务器关停

算法，在带宽需求较低的场景中通过节省边缘服务器的开机成本使得总成本进一步下探。最后，本章在九个数据集上进行了一系列实验，对比实验结果验证了 IEBA 算法和 IEBA-L 算法在各自场景中的有效性，消融实验结果表明了各个模块的必要性，并且通过实验发现引出未来的研究方向。

5 总结与展望

5.1 全文总结

近年来，随着边缘计算的繁荣，将边缘用户的带宽需求分配到合适的边缘服务器使得总带宽开销成本最小化，对于边缘应用程序提供商而言具有非常重要的现实意义。然而现有的研究大都忽略了网络时延约束条件以及未考虑边缘服务器的成本定价差异。在现实场景中，并不是所有的边缘服务器都是可用的，边缘服务器存在开机成本，不同边缘服务器之间的带宽租用价格存在差异性。针对这些问题，本文基于一个两阶段的框架，分别对带宽需求可分割场景和带宽需求不可分割场景中的边缘带宽分配算法进行了研究。

针对带宽需求可分割场景中的边缘带宽分配问题，本文首先构建了考虑 95 计费和网络时延约束的问题模型并给出其形式化定义，接着提出 DEBA 算法予以解决。该算法在第一阶段通过基于网络流模型的启发式方法，充分挖掘 95 计费的特点，快速找到一个质量不错的可行初始解。在第二阶段，基于自适应大邻域搜索算法、爬山算法和模拟退火算法，对解进行持续迭代优化。此外，本文设计了 OptDinic 算法来有效地产生邻域解以及验证其可行性。在真实边缘带宽需求数据集上进行了一系列对比试验和消融实验，验证了 DEBA 算法的有效性和各个模块的必要性。

针对带宽需求不可分割场景中的边缘带宽分配问题，本文在第三章问题模型的基础上进行了修改和重构。具体而言，针对带宽需求不可分割的特点，引入了流的概念，将流作为带宽需求分配调度的最小单位。同时引入边缘服务器成本计算公式，将服务器的开机成本和带宽租用成本差异性考虑在内。接着，本文针对高需求和低需求场景分别提出了 IEBA 算法和 IEBA-L 算法。IEBA 算法在第一阶段通过流的迁移交换生成一个可行初始解，在第二阶段通过邻域搜索算法对解进行优化。而 IEBA-L 算法通过在 IEBA 算法的第二阶段引入服务器关停算法，能够在带宽需求较低的场景中通过节省边缘服务器的开机成本从而使得总成本进一步下探。本文在九个数据集上进行了一系列对比实验和消融实验，验证了 IEBA 算法和 IEBA-L 算法在各自场景中的有效性和各个模块的必要性。

5.2 展望

尽管本文针对多个场景下的边缘带宽分配问题给出了相应的解决方案，但是未来仍可以从以下几个方面做进一步的研究。

（1）针对同样是离线场景下的边缘带宽分配问题，由 4.5.4 节实验可知，当算法运行的计费周期缩短为一天时，能够在不显著降低解的质量的同时，极大地缩短算法的运行时间。因此，在未来的工作中，将进一步研究多核环境下边缘带宽分配算法的并行化实现，通过运用并行化处理技术，以达到在短时间内给出质量较优的分配方案的目的。

（2）研究离线场景中的边缘带宽分配算法最终都是为了能够在在线场景中更好地发挥作用。现有的针对在线场景中的带宽分配问题，都是通过预测未来整个月内的带宽需求数据集，然后在预测的数据集上运行离线算法。但是由于预测的时间跨度过大，往往导致预测的准确度下降，最终影响解的质量。受 4.5.4 节实验启发，在未来的工作中，将进一步研究基于时间序列模型的边缘带宽分配问题短期预测研究，通过每次预测未来一天的带宽需求，达到提升预测准确度、改善最终解的质量和提高算法运行效率的目的。

参考文献

[1] SHI W S, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[2] XIA Q F, REN W H, XU Z C, et al. When edge caching meets a budget: near optimal service delivery in multi-tiered edge clouds[J]. IEEE Transactions on Services Computing, 2022, 15(06): 3634-3648.

[3] LIYANAGE M, PORAMBAGE P, DING A Y, et al. Driving forces for multi-access edge computing (MEC) IoT integration in 5G[J]. ICT Express, 2021, 7(2): 127-137.

[4] SATYANARAYANAN M. The emergence of edge computing[J]. Computer, 2017, 50(1): 30-39.

[5] HONG C H, VARGHESE B. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms[J]. ACM Computing Surveys (CSUR), 2019, 52(5): 1-37.

[6] BITTMAN B, GILL B, ZIMMERMAN T, et al. Predicts 2022: the distributed enterprise drives computing to the edge[EB/OL]. [2023-03-21]. https://www.gartner.com/en/documents/4007176.

[7] SHAKIL K A, ALAM M, KHAN S. A latency-aware max-min algorithm for resource allocation in cloud[J]. International Journal of Electrical and Computer Engineering, 2021, 11(1): 671.

[8] SHOKRNEZHAD M, TALEB T. Near-optimal cloud-network integrated resource allocation for latency-sensitive B5G[C]//GLOBECOM 2022-2022 IEEE Global Communications Conference. Rio de Janeiro, Brazil: IEEE, 2022: 4498-4503.

[9] DIMITROPOULOS X A, HURLEY P, KIND A, et al. On the 95-percentile billing method[C]//PAM. Seoul, Korea: Springer, 2009: 207-216.

[10] MUKERJEE M K, NAYLOR D, JIANG J C, et al. Practical, real-time centralized control for CDN-based live video delivery[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. New York, NY, USA: ACM, 2015: 311-324.

[11] STANOJEVIC R, LAOUTARIS N, RODRIGUEZ P. On economic heavy hitters: shapley value analysis of 95th-percentile pricing[C]//Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. New York, NY, USA: ACM, 2010: 75-80.

[12] RAJA V R, DHAMDHERE A, SCICCHITANO A, et al. Volume-based transit pricing: is 95 the right percentile?[C]//PAM. Los Angeles, CA, USA: Springer, 2014: 77-87.

[13] ODLYZKO A. Internet pricing and the history of communications[J]. Computer Networks,

2001, 36(5-6): 493-517.

[14] ALTMANN J, DAANEN H, OLIVER H, et al. How to market-manage a QoS network[C]//Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. New York, NY, USA: IEEE, 2002: 284-293.

[15] ODLYZKO A. Paris metro pricing: the minimalist differentiated services solution[C]//1999 Seventh International Workshop on Quality of Service. London, UK: IEEE, 1999: 159-161.

[16] HOSANAGAR K. CDN pricing[J]. Content Delivery Networks, 2008, 9: 211-224.

[17] ADLER M, SITARAMAN R, VENKATARAMANI H. Algorithms for optimizing bandwidth costs on the internet[C]//2006 1st IEEE Workshop on Hot Topics in Web Systems and Technologies. Boston, MA, USA: IEEE, 2006: 1-9.

[18] ADLER M, SITARAMAN R K, VENKATARAMANI H. Algorithms for optimizing the bandwidth cost of content delivery[J]. Computer Networks, 2011, 55(18): 4007-4020.

[19] DIMITROPOULOS X A, HURLEY P, KIND A, et al. On the 95-percentile billing method[C]//PAM. Seoul, Korea: Springer, 2009: 207-216.

[20] JAIN S, FALL K, PATRA R. Routing in a delay tolerant network[C]//Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York, NY, USA: ACM, 2004: 145-158.

[21] WANG J P. Traffic regulation under the percentile-based pricing policy[C]//Proceedings of the 1st International Conference on Scalable Information Systems. New York, NY, USA: ACM, 2006: 4-es.

[22] WANG J P, CHEN J, YANG M, et al. Traffic regulation with single-and dual-homed ISPs under a percentile-based pricing policy[J]. Journal of Combinatorial Optimization, 2009, 17: 247-273.

[23] LAOUTARIS N, SMARAGDAKIS G, RODRIGUEZ P, et al. Delay tolerant bulk data transfers on the internet[C]//Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems. New York, NY, USA: ACM, 2009: 229-238.

[24] GOLUBCHIK L, KHULLER S, MUKHERJEE K, et al. To send or not to send: reducing the cost of data transmission[C]//2013 Proceedings IEEE INFOCOM. Turin, Italy: IEEE, 2013: 2472-2478.

[25] KHARE V, ZHANG B C. CDN request routing to reduce network access cost[C]//37th Annual IEEE Conference on Local Computer Networks. Clearwater Beach, FL, USA: IEEE, 2012: 610-617.

[26] GOLDENBERG D K, QIUY L L, XIE H Y, et al. Optimizing cost and performance for multihoming[J]. ACM SIGCOMM Computer Communication Review, 2004, 34(4): 79-92.

[27] FENG Y, LI B C, LI B. Jetway: minimizing costs on inter-datacenter video

traffic[C]//Proceedings of the 20th ACM International Conference on Multimedia. New York, NY, USA: ACM, 2012: 259-268.

[28] HONG C Y, KANDULA S, MAHAJAN R, et al. Achieving high utilization with software-driven WAN[C]//Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. New York, NY, USA: ACM, 2013: 15-26.

[29] LIN Y H, SHEN H Y, CHEN L H. Ecoflow: an economical and deadline-driven inter-datacenter video flow scheduling system[C]//Proceedings of the 23rd ACM International Conference on Multimedia. New York, NY, USA: ACM, 2015: 1059-1062.

[30] JALAPARTI V, BLIZNETS I, KANDULA S, et al. Dynamic pricing and traffic engineering for timely inter-datacenter transfers[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York, NY, USA: ACM, 2016: 73-86.

[31] LI W X, ZHOU X B, LI K Q, et al. TrafficShaper: shaping inter-datacenter traffic to reduce the transmission cost[J]. IEEE/ACM Transactions on Networking, 2018, 26(3): 1193-1206.

[32] WENDELL P, JIANG J W, FREEDMAN M J, et al. Donar: decentralized server selection for cloud services[C]//Proceedings of the ACM SIGCOMM 2010 Conference. New York, NY, USA: ACM, 2010: 231-242.

[33] ZHANG Z, ZHANG M, GREENBERG A G, et al. Optimizing cost and performance in online service provider networks[C]//NSDI. San Jose, CA, USA: USENIX, 2010: 33-48.

[34] CLEGG R G, LANDA R, ARAÚJO J T, et al. Tardis: stably shifting traffic in space and time[J]. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(1): 593-594.

[35] ZHAN Y, GHAMKHARI M, AKHAVAN-HEJAZI H, et al. Optimal response to burstable billing under demand uncertainty[EB/OL]. [2023-03-21]. https://arxiv.org/pdf/1603.05752.pdf.

[36] SINGH R, AGARWAL S, CALDER M, et al. Cost-effective cloud edge traffic engineering with cascara[C]//NSDI. San Jose, CA, USA: USENIX, 2021: 201-216.

[37] 陈寰. 内容分发网络在 95 计费下的流量分配[D]. 安徽: 中国科学技术大学, 2022.

[38] YANG C P, YOU J T, YUAN X M, et al. Network bandwidth allocation problem for cloud computing[EB/OL]. [2023-03-21]. https://arxiv.org/pdf/2203.06725.pdf.

[39] YOU Y Y, FENG B B, DING Z J. Q-percentile bandwidth billing based geo-scheduling algorithm[C]//2022 IEEE 15th International Conference on Cloud Computing (CLOUD). Barcelona, Spain: IEEE, 2022: 219-229.

[40] XU X P, LI W X, QI H, et al. Latency-constrained cost-minimized request allocation for geo-distributed cloud services[J]. IEEE Open Journal of the Communications Society, 2020, 1: 125-132.

[41] CHU P C, BEASLEY J E. A genetic algorithm for the generalised assignment problem[J].

Computers & Operations Research, 1997, 24(1): 17-23.

[42] YAGIURA M, YAMAGUCHI T, IBARAKI T. A variable depth search algorithm for the generalized assignment problem[J]. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, 1999: 459-471.

[43] YAGIURA M, IBARAKI T, GLOVER F. An ejection chain approach for the generalized assignment problem[J]. INFORMS Journal on Computing, 2004, 16(2): 133-151.

[44] VANDERBEI R J. Linear programming[M]. Cham, Switzerland: Springer, 2020: 6.

[45] CORMEN T H, LEISERSON C E, RIVEST R L, et al. Introduction to algorithms[M]. 3rd ed. Massachusetts, USA: MIT Press, 2009: 709.

[46] FORD L R, FULKERSON D R. Maximal flow through a network[J]. Canadian Journal of Mathematics, 1956, 8: 399-404.

[47] EDMONDS J, KARP R M. Theoretical improvements in algorithmic efficiency for network flow problems[J]. Journal of the ACM (JACM), 1972, 19(2): 248-264.

[48] DINIC E A. Algorithm for solution of a problem of maximum flow in networks with power estimation[J]. Soviet Math Dokl, 1970, 11(8): 1277-1280.

[49] KIRKPATRICK S, GELATT Jr C D, VECCHI M P. Optimization by simulated annealing[J]. Science, 1983, 220(4598): 671-680.

[50] ROPKE S, PISINGER D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows[J]. Transportation Science, 2006, 40(4): 455-472.

[51] GARY M R, JOHNSON D S. Computers and intractability: a guide to the theory of NP-completeness[M]. New York, NY, USA: W. H. Freeman, 1979: 245.

[52] ALIYUN. Aliyun ecs pricing-calculator[EB/OL]. [2023-03-21]. https://www.aliyun.com/pricing-calculator?#/ecs/detail/vm.

[53] GHOMI E J, RAHMANI A M, QADER N N. Load-balancing algorithms in cloud computing: a survey[J]. Journal of Network and Computer Applications, 2017, 88: 50-71.

[54] BALHARITH T, ALHAIDARI F. Round robin scheduling algorithm in CPU and cloud computing: a review[C]//2019 2nd International Conference on Computer Applications & Information Security (ICCAIS). Riyadh, Saudi Arabia: IEEE, 2019: 1-7.

[55] BERGANTIÑOS G, GÓMEZ-RÚA M, Llorca N, et al. Allocating costs in set covering problems[J]. European Journal of Operational Research, 2020, 284(3): 1074-1087.

[56] SAHNI S, GONZALEZ T. P-complete approximation problems[J]. Journal of the ACM (JACM), 1976, 23(3): 555-565.

[57] BAYS C. A comparison of next-fit, first-fit, and best-fit[J]. Communications of the ACM, 1977, 20(3): 191-192.

C. 学位论文数据集

<table>
<tr><td>关键词</td><td colspan="2">密级</td><td>中图分类号</td></tr>
<tr><td>边缘计算；95 计费模型；边缘带宽分配；邻域搜索；网络时延约束</td><td colspan="2">公开</td><td>TP</td></tr>
<tr><td>学位授予单位名称</td><td>学位授予单位代码</td><td>学位类别</td><td>学位级别</td></tr>
<tr><td>重庆大学</td><td>10611</td><td>学术学位</td><td>硕士</td></tr>
<tr><td>论文题名</td><td colspan="2">并列题名</td><td>论文语种</td></tr>
<tr><td>考虑网络时延约束的多场景边缘带宽分配算法研究</td><td colspan="2">无</td><td>中文</td></tr>
<tr><td>作者姓名</td><td>胡庆弘</td><td>学号</td><td>202014021052t</td></tr>
<tr><td colspan="2">培养单位名称</td><td colspan="2">培养单位代码</td></tr>
<tr><td colspan="2">重庆大学</td><td colspan="2">10611</td></tr>
<tr><td>学科专业</td><td>研究方向</td><td>学制</td><td>学位授予年</td></tr>
<tr><td>计算机科学与技术</td><td>边缘计算</td><td>3 年</td><td>2023 年</td></tr>
<tr><td>论文提交日期</td><td>2023 年 6 月</td><td>论文总页数</td><td>80 页</td></tr>
<tr><td>导师姓名</td><td>尚家兴</td><td>职称</td><td>副教授</td></tr>
<tr><td colspan="2">答辩委员会主席</td><td colspan="2">唐亮贵 教授</td></tr>
<tr><td colspan="4">电子版论文提交格式
文本（√） 图像（） 视频（） 音频（） 多媒体（） 其他（）</td></tr>
</table>