https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0030-364X, EISSN 1526-5463

Submitted to *Operations Research*

Solving Huge-Scale Bandwidth Allocation Problems with 95^{th} Percentile Billing Scheme at the Cloud Edge: The MIP Approach

(Authors' names are not included for peer review)

Abstract.

We model the bandwidth allocation problem under the standard 95^{th} percentile billing scheme, which emerges recently in the cloud computing industry, as a mixed integer programming (MIP) problem, and propose a GPU-adaptive matheuristic algorithm. The new algorithm could solve the MIP with the scale of millions of variables in seconds, which is a thousand-fold acceleration compared with traditional solvers. Our key technique is to solve the underlying traffic routing problem by the well-known alternating direction method of multipliers (ADMM) with sophisticated matrix representation skills so that all the subproblems are perfectly suitable for parallel computing and execution on GPUs. The proposed algorithm has been implemented in industry, demonstrating clear commercial value in saving operational costs, and our methodology can also be extended to design GPU-adaptive algorithms for other large-scale optimization problems with strong industry background.

Key words: MIP, GPU-adaptive, ADMM, Trafic Allocation, Cloud Computing

1. Introduction

Cloud computing has emerged as a transformative force in the technology sector, with companies like Amazon and Google leading the way as prominent cloud service providers (CSPs). The market is projected to exceed $1,600 billion in value by 2030 (Precedence Research 2022). This evolution has revolutionized both client and provider operations, facilitating efficient resource utilization that reduces costs for clients while enhancing operational efficiency for CSPs through economies of scale. Communication between clients and CSPs typically occurs via high-bandwidth, low-latency networks maintained by Internet service providers (ISPs). As reliance on high-quality internet connectivity intensifies, optimizing bandwidth costs while ensuring acceptable quality of service (QoS) has become a critical challenge for CSPs.

Within the entire cloud network, the cloud edge represents a substantial portion of the operational expenditure for CSPs and is vital for guaranteeing an acceptable QoS for customers. Figure 1 provides a demonstration of the cloud edge architecture, utilizing Domain Name System (DNS) based routing to ensure

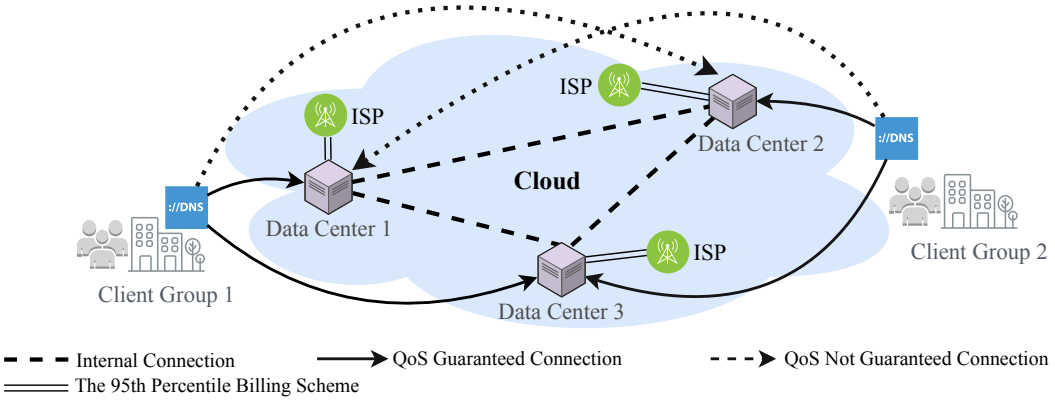
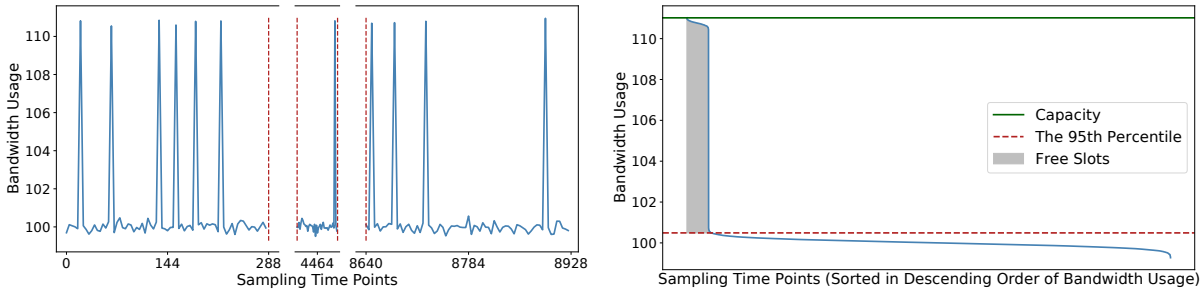


Figure 1 An illustrative example of traffic allocation at cloud edge

clients from a specific location (e.g., cities, states and provinces) share a common routing table and referred as client groups. The QoS between customers and computing clouds heavily depends on factors such as latency, location, and reliability. As a result, demands from a particular DNS can only be directed to a subset of cloud Data Centers (DCs). This means that certain client groups can only connect to a portion of the DCs at the cloud edge. In Figure 1, solid arrows indicate feasible communication pathways, dashed arrows highlight connections that are not feasible, and double-lines represent the economic transaction where the CSP compensate ISPs for Internet bandwidth resources, following the 95^{th} percentile billing scheme.

In DCs, Internet bandwidth usage is billed by ISPs based on the industry-standard monthly 95^{th} percentile burstable billing scheme. Under this scheme, Internet bandwidth consumption at a DC is sampled every five minutes, and all billed samples are ranked from highest to lowest according to bandwidth usage. The top 5% of these samples are excluded from the billing calculation. Figure 2 provides an illustrative example of the 95^{th} percentile billing scheme applied at a single DC, where the billing cycle is a 31-day month (i.e., 8928 five-minute time slots in total, and 446 free time slots). As depicted in Figure 2(b), the sampled bandwidth usage data, consisting of 8928 entries, is arranged in descending order. The highest 446 data points (the gray part) are subsequently excluded from the calculation. Consequently, the highest of remaining 8482 data points (the red dashed line), which represents the 95^{th} percentile, is the billed bandwidth usage. This example demonstrates that in scenarios characterized by significant data fluctuations and occasional spikes to exceptionally high values, the point of actual billing can deviate substantially from the point of peak demand. Consequently, by strategically allocating bandwidth across DCs, it is possible to achieve a significant reduction in the DC's operational costs.

Achieving optimal solutions for Bandwidth Allocation at Cloud Edge under the 95^{th} percentile billing scheme (abbreviated as "BACE-95") can significantly reduce operational expenditures for CSPs. However,



(a) A sampling of bandwidth usage during a 31-day month (i.e., 8928 five-minute time slots).

(b) Bandwidth usage data sorted in descending order with the 95^{th} percentile clearly marked.

Figure 2 A real-world example of the 95^{th} percentile billing scheme applied at a single data center

BACE-95 is mathematically NP-hard in the strong sense and computationally challenging, which poses significant challenges for the development of efficient solution methodologies (Adler et al. 2011, Jalaparti et al. 2016, Singh et al. 2021). Modern GPUs, such as the NVIDIA A100 and H100, equipped with thousands of tensor cores, provide unparalleled parallel processing power (NVIDIA Corporation 2024). For example, the NVIDIA A100 achieves up to 312 teraflops of performance, significantly accelerating tasks like matrix multiplication. Where high-end CPUs may take seconds to perform such operations, GPUs can complete them in a fraction of the time, delivering speedups of 10 to 100 times. This leap in computational power has revolutionized scientific computing, enabling the efficient handling of complex problems at scale. We are motivated to consider solving large-scale computational problems due to recent advancements in specialized hardware, particularly Graphics Processing Units (GPUs), which have made efficient solutions feasible.

We formulate the BACE-95 problem as a MIP model and decompose it into two interconnected subproblems: Traffic Routing (TR), which is a Linear Programming (LP) problem that determines how demands from client groups are routed to DCs, and Bandwidth Scheduling (BS), which manages the allocation of bandwidth capacity across DCs throughout a billing cycle. The TR problem is a critical component in tactical operations for evaluating the status of the Traffic Engineering (TE) system and holds greater significance compared to the BS problem. It can be treated as a transportation problem augmented by a high-dimensional time component, where intricate interdependencies among pricing variables substantially increase computational complexity. Notably, even a simplified representation of the problem, using its LP relaxation, mandates extensive computational effort due to its high dimensionality: standard commercial solvers can take up to 15 hours to produce a sub-optimal solution for even modest-sized instances, with deviations potentially reaching up to 15% from the optimal benchmark (Singh et al. 2021). The layered structure of TR problem presents considerable challenges for commercial solvers like CPLEX, which struggle to efficiently handle the problem's vast dimensionality and dense interconnectivity among variables. The BS problem is heavily dependent on the rapid evaluation and feedback of the TR problem. It is also more reliant on management strategies, which are modeled using binary variables that determine whether the DC's potential is fully

realized. We investigate how to customize the classic Alternating Direction Method of Multipliers (ADMM) (Glowinski and Marroco 1975) to solve the TR problem by leveraging its specific structural characteristics. The binary decision variables in BS problem introduce significant combinatorial complexity. To tackle this difficulty, we leverage a specialized Adaptive Large Neighborhood Search (ALNS) algorithm that employs strategic matheuristic destroy and repair mechanisms, informed by insights from the TR problem (Ropke and Pisinger 2006). The ADMM and ALNS together form a two-stage iterative algorithm that solves the problem in a step-by-step manner.

The main contributions of this paper can be summarized as follows:

1. This study pioneers the research into the bandwidth allocation problem in cloud computing from an operations research perspective. The BACE-95 problem is inherently complex and challenging due to its non-convexity, non-smoothness, and high dimensionality. However, by identifying key structural characteristics, we strategically reformulate the problem as a MIP model and decompose it into two stages: tactical traffic routing optimization and strategic bandwidth scheduling management. This decomposition not only simplifies the problem by allowing independent algorithm design and management analysis but also highlights that the primary computational challenge lies in solving the highly interconnected, high-dimensional TR subproblem.

2. We apply ADMM to decompose the TR problem into separable subproblems, which are further broken down into independent one-dimensional subproblems, each with a closed-form solution, enabling efficient parallel computation. By expressing the problem in matrix form, ADMM leverages modern GPU parallel capabilities, resulting in significant acceleration. These features make ADMM a scalable and effective solution for the TR problem, especially where traditional methods are inefficient.

3. This study demonstrates substantial cost savings in bandwidth allocation problems and provides a practical tool for decision-makers at CSPs. Numerical experiments conducted using anonymized real-world data from a leading CSP confirm that the proposed methods achieve up to a 70% reduction in total bandwidth costs, highlighting their tangible effectiveness and value in real operational scenarios.

The remainder of this paper is structured as follows: Section 2 provides an overview of related literature on bandwidth traffic engineering in cloud computing and algorithms for large-scale system optimization. Section 3 delves into the BACE-95, presents its mathematical model as a MIP, and conducts a mathematical analysis to inform our algorithm development. In Section 4, we detail our novel two-stage ADMM-ALNS matheuristic algorithm and its implementation. Section 5 discusses the results from numerical experiments, showcasing the effectiveness of our model and algorithm. Finally, Section 6 summarizes our findings and concludes the study.

2. Literature Review

In this section, we provide a comprehensive literature review, focusing on two main areas: traffic management in cloud computing, and algorithm design specifically for large-scale convex, linear and combinatorial optimization problems.

2.1. Bandwidth Traffic Management in Cloud Computing

Managing computer networks involves the complex interplay of diverse hardware and software elements like switches, routers, and firewalls, presenting various challenges (Feamster et al. 2014). Within this intricate landscape, network bandwidth allocation stands out as a pivotal task, aiming to optimize resource use while ensuring QoS. Existing literature mainly tackles this issue through the lens of network design. For example, Guerin et al. (1991) introduces a key metric for assessing effective bandwidth usage. Studies such as Assi et al. (2003), Luo and Ansari (2005), and McGarry et al. (2008) explore dynamic bandwidth allocation in Ethernet passive optical networks. Meanwhile, Kivanc et al. (2003) and Christodoulopoulos et al. (2011) offer computationally efficient methods tailored for orthogonal frequency division multiplexing based optical networks.

Recently, both the industry and academic realms have shown a keen interest in the centralized management of scheduling and bandwidth allocation. Liu et al. (2012) argue that a centralized controller can enhance the user experience and suggest a TE system to boost network utilization without compromising latency. Similarly, Hong et al. (2013) advocate for a centralized, software-driven Wide Area Network (SWAN) that orchestrates sending rates and network paths across global DCs. Google's B4, as presented by Jain et al. (2013), showcases a centralized TE approach that augments utilization across its global DCs by distributing flows over multiple paths to even out capacity. In the realm of content delivery, Mukerjee et al. (2015) introduce a video delivery network that provides content delivery network operators with the flexibility to manage stream placement and bitrate constraints dynamically. Additionally, Chou et al. (2019) detail Facebook's cloud-edge traffic allocation strategy, which has effectively reduced backend storage query loads by 17%. According to Yuan et al. (2024), utilizing well-designed TE systems, Huawei can save up to 30% of the bandwidth cost, equivalent to 49.6 million dollars.

Bandwidth cost is the principal expenditure for CSPs, with its management complexity exacerbated by the industry-standard 95^{th} percentile burstable billing scheme. Studies such as those by Jalaparti et al. (2016) and Adler et al. (2011) have demonstrated the NP-hardness of optimizing the 95^{th} percentile usage billing, drawing parallels to well-known computational problems. Furthermore, the complexity of the 95^{th} percentile billing scheme in bandwidth allocation has been explored extensively, highlighting the inadequacies of linear approximations and the limitations of commercial solvers in handling this complex objective function under variable traffic patterns. For instance, research by Singh et al. (2021) critiques these approximations and reports the use of a general-purpose commercial MILP solver, GUROBI, which required over 15

hours to reach a suboptimal solution even for modest datasets. This inefficiency underscores the need for more specialized algorithms tailored to the unique challenges of the 95^{th} percentile billing model in cloud computing. Meanwhile, Zhan et al. (2016) present a non-convex and stochastic programming model that simplifies the problem to a form more amenable to these solvers, emphasizing the ongoing need for innovation in computational strategies for bandwidth management.

The traffic routing part of the BACE-95 studied in this paper is highly related to the assignment problem (Pentico 2007) and transport problem (Munkres 1957). However, as mentioned earlier, this study differs from existing literature in that the problem involves complex multi-period decision-making, which presents computational challenges and structural complexity. Notably, despite its critical importance for all large CSPs, the literature has not yet thoroughly explored traffic allocation under the 95^{th} percentile burstable pricing scheme, nor has it yielded an algorithm specifically designed for this purpose.

2.2. Scalable Algorithm Design for Large-Scale Optimization

The core challenges addressed in this paper stem from two principal factors: the large scale inherent in cloud computing, and the NP-hard complexity introduced by our modeling approach. Consequently, our subsequent review concentrates on scalable algorithms adept at managing large-scale systems.

Classic ADMM deals with linearly constrained separable convex optimization problems, where the objective function can be split into independent convex functions. Groundbreaking work by Glowinski and Marroco (1975) and Douglas and Rachford (1956) laid the foundations for ADMM, while He and Yuan (2012) and He and Yuan (2015) further advanced the method by establishing rates of convergence. While Chen et al. (2016) cautioned against the direct application of ADMM to multi-block convex minimization problems, demonstrating potential for divergence, the technique has nonetheless been successfully leveraged in various fields. For instance, Mohan et al. (2013) applied it to the latent variable Gaussian graphical model selection, and Tao and Yuan (2011) extended it to robust principal component analysis in the context of noisy and incomplete data. He et al. (2012) extends ADMM to the general case of linearly constrained separable convex minimization problems, while ensuring convergence and demonstrating numerical efficiency through various application problems. He et al. (2015) propose a splitting method for separable convex minimization with linear constraints, demonstrating its global convergence and numerical efficiency.

The landscape of solving NP-hard combinatorial problems is dominated by exact algorithms such as branch and bound (Land and Doig 1960) and its derivatives: branch and cut (Padberg and Rinaldi 1991), branch and price (Barnhart et al. 1998), and branch and price and cut (Desrosiers and Lübbecke 2011). Dantzig-Wolfe decomposition, introduced by Dantzig and Wolfe (1960), and Benders decomposition, proposed by Benders (1962), are techniques designed to tackle large-scale decomposable problems. Both approaches require the problem to have a specific structural form where the constraint matrix exhibits a certain block structure that allows for decomposition into more manageable subproblems. Despite their theoretical elegance, these

algorithms often fall short when applied to real-world, large-scale instances, primarily due to the exhaustive nature of their search mechanisms.

To bridge this gap, researchers have turned to the development of heuristics and meta-heuristics, which offer more practical solutions. The large neighborhood search (LNS) method, proposed by Shaw (1998), was a significant milestone in this direction. It approached the classic vehicle routing problem (VRP) by iteratively destructing and reconstructing large portions of the current solution, thus exploring a substantial neighborhood space. Building on this, Ropke and Pisinger (2006) introduced an adaptive mechanism to the LNS, resulting in the ALNS, an heuristic that dynamically adjusts its parameters to improve performance. Pisinger and Ropke (2007) further expanded on the ALNS, tailoring it into a flexible heuristic framework adaptable to various specific scenarios. This flexibility was evidenced by the improvement of 183 out of 486 benchmark instances of the VRP. The versatility of the ALNS framework has been proven through its successful application across a range of fields, such as the pickup and delivery problem with time windows (Ropke and Pisinger 2006), the pollution-routing problem (Demir et al. 2012), the pickup and delivery problem involving transfers (Masson et al. 2013), and the production routing problem (Adulyasak et al. 2014).

Despite the efficiency gains from centralized TE systems and software-defined networks, cost-effective bandwidth allocation challenges persist due to the lack of algorithms for these large-scale, NP-hard problems. This paper pioneers research to bridge the identified gap from an operations research perspective and designs algorithms for large-scale problems that leverage modern GPU architectures.

3. Mathematical Modeling and Analysis

In this section, we formally define the BACE-95 as a mathematical problem and conduct a thorough analysis of its mathematical properties, thereby laying the groundwork for subsequent algorithmic design.

3.1. Problem Statement and Modeling

Figure 3 illustrates an example of BACE-95, which seeks to optimally allocate bandwidth requests from client groups, aggregated at DNSs, to suitable cloud edge nodes. The objective is to minimize the total bandwidth cost under the 95^{th} percentile billing scheme while satisfying QoS requirements, such as maintaining latency below 400 ms, over a billing cycle, typically one month. Table 1 lists the notations used for the mathematical formulation. Nodes $i \in I$ are virtual representative client group nodes, each with an aggregated demand D_{it} at a 5-minute time slot $t \in T$. Nodes $j \in J$ are virtual representative cloud edge nodes, each with a nonnegative capacity C_j. The QoS requirement is quantified by the binary parameter a_{ij}, where a value of 1 indicates a valid connection, and 0 indicates infeasible connection. The BACE-95 seeks to identify L burstable time intervals within the billing cycle T, ensuring the formulation of scheduling plans for each cloud edge node $j \in J$. This entails directing the traffic flow for each customer $i \in I$ through the edges

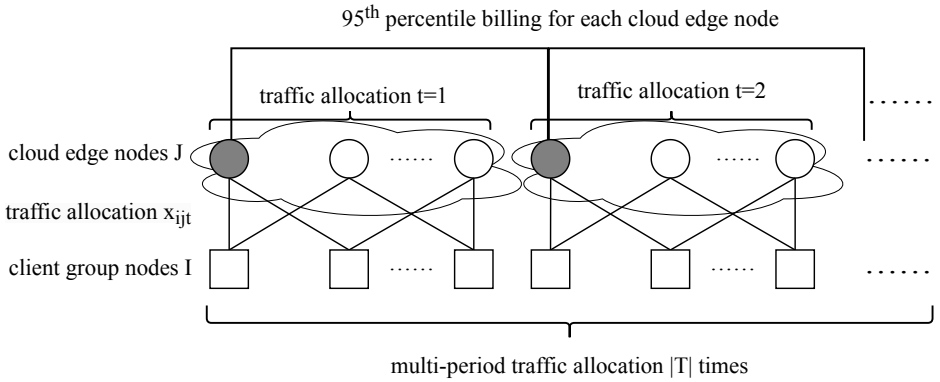


Figure 3 An illustrative example of traffic allocation within a billing cycle under the 95^{th} percentile billing scheme

Notation	Definition
$I = \{1, 2, \cdots m\}$	The set of client group nodes
$J = \{1, 2, \cdots n\}$	The set of cloud edge nodes
$T = \{1, 2, \cdots o\}$	The set of 5-minute time slots within a billing cycle
$E = \{(i, j) \mid i \in I, j \in J\}$	The set of edges that connects client group node i and cloud edge node j
a_{ij}	Indicator variable: 1 if traffic through edge (i, j) meets QoS requirements, 0 otherwise
C_j	The bandwidth capacity of cloud edge node $j \in J$
D_{it}	The traffic demand of client group node $i \in I$ during time slot $t \in T$
$m = \|I\|$	The number of client group nodes
$n = \|J\|$	The number of cloud edge nodes
$o = \|T\|$	The number of 5-minute intervals in the billing cycle
$L = \lfloor 0.05 \times o \rfloor$	The number representing 5% of all 5-minute intervals

Table 1 The summary of parameter notations

$(i, j) \in E$, while adhering to the QoS requirements, accommodating the demand D_{it} for all $i \in I$ and $t \in T$, and not exceeding the capacity C_j for each $j \in J$.

The notations and definitions of the variables used in this study are provided in Table 2.

Notation	Definition
$x_{ijt} \geq 0$	The traffic flow from client group node $i \in I$ to cloud edge node $j \in J$ within time interval $t \in T$
$y_{jt} \in \{0, 1\}$	Takes the value 1 if the bandwidth usage of $j \in J$ during $t \in T$ is within its top 5%; 0 otherwise

Table 2 The summary of variable notations

REMARK 1. The 95^{th} percentile of a set of $|T|$ numbers is equivalent to the *k-max* problem, where $k = \lfloor |T|/20 \rfloor$ (Singh et al. 2021).

REMARK 2. The BACE-95 problem is NP-hard (Singh et al. 2021, Zhan et al. 2016, Jalaparti et al. 2016, Adler et al. 2011).

Given the aforementioned problem statements, and with the notations specified in Tables 1 and 2, we can now formally formulate the BACE-95 as follows.

$$\min_{x_{ijt},y_{jt}} \quad \sum_{j} \max_{t} \left((1-y_{jt}) \cdot \sum_{(i,j)\in E} a_{ij} \cdot x_{ijt} \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{t\in T} y_{jt} = L, \qquad \forall j \in J, \tag{2}$$

$$\sum_{(i,j)\in E} a_{ij} x_{ijt} \le C_j, \qquad \forall j \in J, \forall t \in T, \tag{3}$$

$$\sum_{(i,j)\in E} a_{ij} x_{ijt} = D_{it}, \qquad \forall i \in I, \forall t \in T, \tag{4}$$

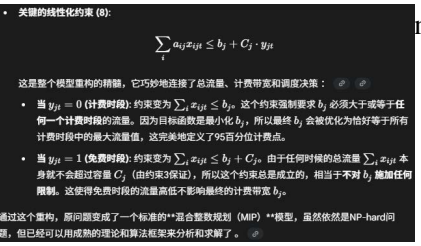
$$x_{ijt} \ge 0, \qquad \forall i \in I, \forall j \in J, \forall t \in T, \tag{5}$$

$$y_{jt} \in \{0,1\}, \qquad \forall j \in J, \forall t \in T. \tag{6}$$

The objective function (1) aims to minimize the total cost across all DCs, which is defined as the maximum bandwidth usage observed during the bottom 95 percent of all time slots for each DC. In this context, we have assumed that the unit price for each cloud edge node is identical and have subsequently omitted it from the calculations. Constraint (2) identifies the top 5% of time intervals for each cloud edge node throughout the billing cycle. Constraint (3) ensures that the total traffic allocated to each cloud edge node at any given time slot does not surpass its available capacity. Constraint (4) guarantees the fulfillment of traffic demand across all time slots. Constraints (5) and (6) define the variables.

REMARK 3. The 95^{th} percentile billing scheme of BACE-95 can be generalized to other less common pricing schemes, such as maximum traffic usage pricing and median traffic usage pricing, by setting L to 0 and $\lfloor 0.5 \times o \rfloor$, respectively.

The BACE-95 problem is inherently nonconvex and nonlinear. To reformulate BACE-95, we introduce the variable $b_j \ge 0$, representing the 95^{th} percentile bandwidth utilization of cloud edge node $j \in J$. This reformulation transforms BACE-95 into the following MIP formulation. It is important to note that this reformulation retains the complex nature of the original problem structure.



$$\min \quad \sum_{j\in J} b_j \tag{7}$$

$$s.t. \quad \sum_{(i,j)\in E} a_{ij} x_{ijt} \le b_j + C_j \cdot y_{jt}, \qquad \forall j \in J, \forall t \in T, \tag{8}$$

Equations (2) – (6),

$$b_j \ge 0, \qquad \forall j \in J. \tag{9}$$

The objective function, presented in Equation (7), calculates the total cost at the 95^{th} percentile across all cloud edge nodes. Constraint (8) establishes the 95^{th} percentile billing point for each cloud edge node, utilizing the selection made by Constraint (2). Note that Constraint (9) is redundant, as $b_j, \forall j \in J$, is implicitly guaranteed by the model to non-negative.

3.2. Structural and Mathematical Analysis of the BACE-95 Problem

As indicated in Remarks 1 and 2, the BACE-95 is NP-hard. In addition to its NP-hard problem structure, the challenge is further intensified by the high dimensionality of the problem (e.g. the studied CSP must allocate the demand of hundreds of client groups across hundreds of DCs over 8,928 or 8,640 time intervals within a month), posing substantial computational difficulties. This subsection presents a mathematical analysis that informs the design of efficient algorithms, addressing computational and structural challenges by treating continuous and binary variables separately.

3.2.1. Analysis of the MIP Formulation for the BACE-95 Problem Given that the structural NP-hardness of the BACE-95 problem primarily arises from its binary variables and the high dimensionality inherited from multi-period decision-making, we reformulate the BACE-95 model into a two-stage form. In essence, the TR stage determines the traffic flow between the client groups and DCs, and the BS stage determines the bandwidth plans of DCs.

$$\text{(TR)} \quad \min \quad \sum_{j\in J} b_j \tag{10}$$

$$s.t. \quad \sum_{(i,j)\in E} a_{ij}x_{ijt} \le C_j \cdot y_{jt} + (1-y_{jt})\cdot b_j, \qquad \forall j\in J, \forall t\in T, \tag{11}$$

$$\sum_{(i,j)\in E} a_{ij}x_{ijt} = D_{it}, \qquad \forall i\in I, \forall t\in T, \tag{12}$$

$$x_{ijt} \ge 0, \qquad \forall i\in I, \forall j\in J, \forall t\in T, \tag{13}$$

$$\text{(BS)} \quad y_{jt} \in \mathbb{Y}, \qquad \forall j\in J, \forall t\in T. \tag{14}$$

where $\mathbb{Y}$ defines the CSP's strategic plan for DCs at the cloud edge, and it is extended across the time dimension to include the top 5% of uncharged time intervals, which is defined as:

$$(\mathbb{Y}) \quad \sum_{t\in T} y_{jt} = L, \qquad \forall j\in J, \tag{15}$$

$$y_{jt} \in \{0,1\}, \qquad \forall j\in J, \forall t\in T. \tag{16}$$

REMARK 4. The constraints of the TR problem are all related to time intervals $t \in T$; however, the separable structure is impeded by the interdependency of variable $b_j, \forall j \in J$. In addition, the presence of a nonzero affine objective function renders the application of dual decomposition methods inapplicable, particularly due to the unbounded update of the variable $b_j, \forall j \in J$.

The TR deals with continuous variables, taking the binary variables as given inputs. Conversely, the BS focuses solely on binary variables and does not include an objective function, serving to define a feasible region for binary variables. In real-world applications, TR solver serves as an efficient evaluation tool for TE engineers to assess network status, quantify DC capacity pressures, identify potential improvements of BS plans, and perform other related tasks.

PROPOSITION 1. *The exploration complexity of finding the optimal values of* $y_{jt} \in \{0,1\}, \forall j \in J, \forall t \in T$ *without any prior information is* $O((\frac{o!}{\lfloor 0.05o \rfloor ! \lceil 0.95o \rceil !})^n)$, *where* $o = |T|$ *and* $n = |J|$.

Proof. According to the 95^{th} percentile billing scheme, a cloud edge node has $l = \lfloor 0.05o \rfloor$ free 5-minute intervals, where o is the total number of intervals in a billing cycle. For each cloud edge node $j \in J$, determining the values of y_{jt} is equivalent to choosing l 5-minute intervals from the set T. The possible number of combinations for each cloud edge node can be denoted by

$$C_o^L = \frac{o!}{L!(o-L)!} = \frac{o!}{\lfloor 0.05o \rfloor ! \lceil 0.95o \rceil !}$$

Given that there are n cloud edge nodes in total, the computational complexity of determining the optimal values for $y_{jt}, \forall j \in J, \forall t \in T$, without any prior information is $O\left(\left(\frac{o!}{\lfloor 0.05o \rfloor ! \cdot \lceil 0.95o \rceil !}\right)^n\right)$, where $o = |T|$ represents the total number of 5-minute sampling intervals. □

Proposition 1 shows that the computational complexity of finding an optimal solution for BS is exponential. Furthermore, the evaluation of a given solution for BS heavily relies on the efficient evaluation of TR. However, even though the binary-variables-fixed TR is a LP problem, solving it directly with a commercial solver is time-consuming due to its large dimensionality and interdependent structure. Solving a problem of this size with a commercial solver is impractical. Therefore, designing a specific algorithm for TR is a top priority to find optimal solutions for BACE-95.

3.2.2. ADMM Application to the TR Problem with Decomposable Structure To tackle the computational challenges posed by solving TR, we leverage the problem's decomposable structure by separating the decision variables $b_j, \forall j \in J$ and $x_{ijt}, \forall i \in I, j \in J, t \in T$. This decomposition reveals a structure conducive to parallel processing, laying the groundwork for efficient computation. We introduce slack variables $p_{jt} \geq 0, \forall j \in J, t \in T$ and $q_{ijt} \geq 0, \forall i \in I, j \in J, t \in T$ to transform the original constraints into equality constraints, thereby enabling the application of ADMM:

$$\min \quad \sum_{j \in J} b_j \tag{17}$$

$$\text{s.t.} \quad \sum_{(i,j) \in E} a_{ij} x_{ijt} + p_{jt} = C_j y_{jt} + (1 - y_{jt}) b_j, \qquad \forall j \in J, \forall t \in T, \tag{18}$$

$$\sum_{(i,j) \in E} a_{ij} x_{ijt} = D_{it}, \qquad \forall i \in I, \forall t \in T, \tag{19}$$

$$x_{ijt} - q_{ijt} = 0, \qquad \forall i \in I, \forall j \in J, \forall t \in T, \tag{20}$$

$$p_{jt}, q_{ijt} \geq 0, \qquad \forall i \in I, \forall j \in J, \forall t \in T. \tag{21}$$

The augmented Lagrangian function of (17)–(21) is as follows:

$$\begin{aligned}\mathcal{L}_\rho\Big(\mathbf{x},\mathbf{b},\mathbf{p},\mathbf{q},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}\Big) = &\sum_{j\in J} b_j + \sum_{j\in J}\sum_{t\in T}\alpha_{jt}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - C_j\,y_{jt} - (1-y_{jt})\,b_j + p_{jt}\Big)\\ &+ \sum_{i\in I}\sum_{t\in T}\beta_{it}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - D_{it}\Big) + \sum_{(i,j)\in E}\sum_{t\in T}\gamma_{ijt}\Big(x_{ijt} - q_{ijt}\Big)\\ &+ \frac{\rho}{2}\sum_{j\in J}\sum_{t\in T}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - C_j\,y_{jt} - (1-y_{jt})\,b_j + p_{jt}\Big)^2\\ &+ \frac{\rho}{2}\sum_{i\in I}\sum_{t\in T}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - D_{it}\Big)^2 + \frac{\rho}{2}\sum_{(i,j)\in E}\sum_{t\in T}\Big(x_{ijt} - q_{ijt}\Big)^2.\end{aligned} \tag{22}$$

where $\alpha_{jt}\in\mathbb{R}$, $\beta_{it}\in\mathbb{R}$, and $\gamma_{ijt}\in\mathbb{R}$ for $\forall i\in I$, $\forall j\in J$, and $\forall t\in T$ are the Lagrange multipliers, and positive parameter ρ penalty parameter.

We categorize the variables into three blocks: $\mathbf{x}$, $\mathbf{b}$, and $\mathbf{s}=(\mathbf{p},\mathbf{q})^\top$, and consider the direct extension of ADMM for (17)–(21). The resulting iterative scheme reads as

$$\begin{cases}\mathbf{x}^{k+1}\in\operatorname{argmin}\left\{\mathcal{L}_\rho(\mathbf{x},\mathbf{b}^k,\mathbf{s}^k,\boldsymbol{\lambda}^k)\,\middle|\,\mathbf{x}\in\mathbb{R}^{mn\times o}\right\}, & \text{(23a)}\\ \mathbf{b}^{k+1}\in\operatorname{argmin}\left\{\mathcal{L}_\rho(\mathbf{x}^{k+1},\mathbf{b},\mathbf{s}^k,\boldsymbol{\lambda}^k)\,\middle|\,\mathbf{b}\in\mathbb{R}^n_+\right\}, & \text{(23b)}\\ \mathbf{s}^{k+1}\in\operatorname{argmin}\left\{\mathcal{L}_\rho(\mathbf{x}^{k+1},\mathbf{b}^{k+1},\mathbf{s},\boldsymbol{\lambda}^k)\,\middle|\,\mathbf{s}\in\mathbb{R}^{n\times o}_+\cup\mathbb{R}^{mn\times o}_+\right\}, & \text{(23c)}\\ \boldsymbol{\lambda}^{k+1}\in\operatorname{argmin}\left\{\mathcal{L}_\rho(\mathbf{x}^{k+1},\mathbf{b}^{k+1},\mathbf{s}^{k+1},\boldsymbol{\lambda})\,\middle|\,\boldsymbol{\lambda}\in\mathbb{R}^{n\times o}\cup\mathbb{R}^{m\times o}\cup\mathbb{R}^{mn\times o}\right\}. & \text{(23d)}\end{cases}$$

Considering the dimensions of each decision variable in (23), the primary computational burden in the update procedure (23) lies in determining the values of x_{ijt} for all $i\in I$, $j\in J$, and $t\in T$, as specified in (23a). To enable efficient updates of x_{ijt}, we observe that the subproblem (23a) can be further decomposed into smaller subproblems for each $t\in T$, as outlined in the following proposition.

PROPOSITION 2. *The update procedure for x_{ijt}, for all $i\in I$, $j\in J$, and $t\in T$, can be decomposed into smaller and mutually independent subproblems, each corresponding to a distinct $t\in T$. In particular, each subproblem only involves the variables $\{x_{ijt}\}_{(i,j)\in E}$ at a fixed t.*

Proof. The key observation follows from examining the structure of $\mathcal{L}_\rho(\mathbf{x},\mathbf{b}^k,\mathbf{s}^k,\boldsymbol{\lambda}^k)$ in (22) and the corresponding update step in (23a). Rewriting the objective,

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^{mn\times o}}\ &\sum_{t\in T}\Bigg(\sum_{j\in J}\alpha^k_{jt}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt}\Big) + \sum_{i\in I}\beta^k_{it}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt}\Big)\\ &+ \sum_{(i,j)\in E}\gamma^k_{ijt}\,x_{ijt} + \frac{\rho}{2}\sum_{j\in J}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - C_j\,y_{jt} - (1-y_{jt})\,b^k_j + p^k_{jt}\Big)^2\\ &+ \frac{\rho}{2}\sum_{i\in I}\Big(\sum_{(i,j)\in E} a_{ij}\,x_{ijt} - D_{it}\Big)^2 + \frac{\rho}{2}\sum_{(i,j)\in E}\big(x_{ijt} - q^k_{ijt}\big)^2\Bigg).\end{aligned} \tag{24}$$

Noticing that each term in the summation is indexed only by t, and that $\mathbf{x}$ can be partitioned as $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|T|})$, where $\mathbf{x}_t$ collects all variables $\{x_{ijt}\}_{(i,j)\in E}$ for a fixed t, we see that minimizing the objective with respect to $\mathbf{x}$ separates into $|T|$ independent problems:

$$\min_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \ldots) = \sum_{t\in T} \min_{\mathbf{x}_t} (\text{terms involving } \mathbf{x}_t \text{ only}). \tag{25}$$

Thus, the update step for each $\mathbf{x}_t$ depends solely on the terms tied to time t, rendering the subproblems decoupled. Consequently, one can solve (or update) each $\mathbf{x}_t$ in parallel without affecting the other ones. This structural decomposition naturally enables a parallel (e.g., GPU-adaptive) implementation, as will be detailed in Subsection 4.1. □

Proposition 2 demonstrates that the update of x_{ijt} for all $i \in I$, $j \in J$, and $t \in T$ can be decomposed with respect to $t \in T$. This motivates a more in-depth analysis to derive a parallelizable computation structure that enables efficient GPU implementation, as will be detailed in Subsection 4.1.

Additionally, the updates for $\mathbf{b}$, $\mathbf{s}$, and $\boldsymbol{\lambda}$ can also be decomposed into one-dimensional optimization tasks, each of which has a closed-form solution, as demonstrated in the following proposition.

PROPOSITION 3. *After obtaining the solution values of x_{ijt} for all $i \in I$, $j \in J$, and $t \in T$, the closed-form solution to (23b)–(23d) can be derived by sequentially as follows:*

$$\begin{cases}
b_j^{k+1} = \left(\dfrac{\sum_{t\in T} \left(\rho\,(y_{jt}-1) \cdot \left(\rho\, C_j\, y_{jt} - \rho\, p_{jt}^k - \rho \sum_{(i,j)\in E} a_{ij}\, x_{ijt}^{k+1} - \alpha_{jt}^k \right) \right) - 1}{\sum_{t\in T} (y_{jt}-1)^2} \right)_+, & \forall\, j \in J, & (26a) \\
p_{jt}^{k+1} = \left(- \sum_{(i,j)\in E} a_{ij}\, x_{ijt}^{k+1} + C_j\, y_{jt} + (1-y_{jt})\, b_j^{k+1} + \dfrac{\alpha_{jt}^{k+1}}{\rho} \right)_+, & \forall\, j \in J,\ \forall\, t \in T, & (26b) \\
q_{ijt}^{k+1} = \left(x_{ijt}^{k+1} \right)_+, & \forall\, i \in I,\ \forall\, j \in J,\ \forall\, t \in T, & (26c) \\
\alpha_{jt}^{k+1} = \alpha_{jt}^k + \rho \left(\sum_{(i,j)\in E} a_{ij}\, x_{ijt}^{k+1} - C_j\, y_{jt} - (1-y_{jt})\, b_j^{k+1} + p_{jt}^{k+1} \right), & \forall\, j \in J,\ \forall\, t \in T, & (26d) \\
\beta_{it}^{k+1} = \beta_{it}^k + \rho \left(\sum_{(i,j)\in E} a_{ij}\, x_{ijt}^{k+1} - D_{it} \right), & \forall\, i \in I,\ \forall\, t \in T, & (26e) \\
\gamma_{ijt}^{k+1} = \gamma_{ijt}^k + \rho \left(x_{ijt}^{k+1} - q_{ijt}^{k+1} \right), & \forall\, i \in I,\ \forall\, j \in J,\ \forall\, t \in T. & (26f)
\end{cases}$$

Proof. The proof is completed by deriving the partial derivatives, setting them to zero to satisfy first-order optimality conditions, and solving the resulting linear equations with respect to each variable while holding the remaining variables constant. □

Observing Propositions 2 and 3, it becomes evident that the procedures for updating all variables can be executed in parallel. For instance, the updating of b_j for each $j \in J$ can be individually handled, thus facilitating parallel computation. This parallelization can significantly enhance computational efficiency, especially when dealing with a large set of variables. However, finding the solution for $x_{ijt}, \forall i \in I, \forall j \in J, \forall t \in T$, remains a challenging task.

4. Algorithmic Description

In this section, we provide a comprehensive description of the GPU-adaptive two-stage ADMM-ALNS algorithm framework. First, we detail the implementation of ADMM for solving TR, emphasizing its capability for parallel computation through GPU. Subsequently, we introduce the ALNS scheme, utilized for fixing the binary variables in BS. The local solutions generated in this process are efficiently evaluated using ADMM.

4.1. The GPU-Adaptive Implementation of ADMM for BACE-95 in Matrix Formulation

Given the scale of real-world applications, which often involve millions of continuous variables, the development of a specialized solver for TR is not merely beneficial but imperative. In this section, we illustrate our GPU-adaptive solver based on ADMM. To aid our discussion on algorithm development, we outline below a compact matrix formulation of the proposed BACE-95 model.

Define an indicator function $\sigma(\cdot)$ as follows:

$$\sigma(i,j) = \begin{cases} 1, & \text{if} \quad (i,j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

Then, define an $n \times mn$ matrix $\mathbf{\Phi}$ and a $m \times mn$ matrix $\mathbf{\Psi}$:

$$\mathbf{\Phi} := \begin{bmatrix} \sigma(1,1) & \cdots & 0 & \cdots & \sigma(m,1) & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma(1,n) & \cdots & 0 & \cdots & \sigma(m,n) \end{bmatrix}, \mathbf{\Psi} := \begin{bmatrix} \sigma(1,1) & \cdots & \sigma(1,n) & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & \sigma(m,1) & \cdots & \sigma(m,n) \end{bmatrix}. \tag{28}$$

We list the constructed compact forms of parameters and decision variables as follows:

$$\mathbf{b} := \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^\top, \mathbf{B} := \underbrace{[\mathbf{b}, \mathbf{b}, \cdots, \mathbf{b}]}_{o}{}_{n \times o}, \mathbf{c} := \begin{bmatrix} C_1 & C_2 & \dots & C_n \end{bmatrix}^\top, \mathbf{C} := \underbrace{[\mathbf{c}, \mathbf{c}, \cdots, \mathbf{c}]}_{o}{}_{n \times o}, \tag{29}$$

$$\mathbf{D} := \begin{bmatrix} D_{11} & \cdots & D_{1o} \\ \vdots & \ddots & \vdots \\ D_{m1} & \cdots & D_{mo} \end{bmatrix}, \mathbf{L} := \underbrace{[L, L, \cdots, L]}_{n}{}_{1 \times n}, \mathbf{X} := \begin{bmatrix} x_{111} & x_{112} & \cdots & x_{11o} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n1} & x_{1n2} & \cdots & x_{1no} \\ \vdots & \vdots & \ddots & \vdots \\ x_{mn1} & x_{mn2} & \cdots & x_{mno} \end{bmatrix}, \text{and } \mathbf{Y} := \begin{bmatrix} y_{11} & \cdots & y_{n1} \\ \vdots & \ddots & \vdots \\ y_{1o} & \cdots & y_{no} \end{bmatrix}. \tag{30}$$

Subsequently, the compact matrix formulation of BACE-95 is written as

$$\min \quad \mathbf{1}_n^\top \mathbf{b} \tag{31a}$$

$$\text{s.t.} \quad \mathbf{\Phi X} \le \mathbf{C}; \tag{31b}$$

$$\mathbf{\Psi X} = \mathbf{D}; \tag{31c}$$

$$\mathbf{1}_p^\top \mathbf{Y} = \mathbf{L}; \tag{31d}$$

$$\mathbf{\Phi X} - \mathbf{C} \odot \mathbf{Y}^\top \le \mathbf{B}; \tag{31e}$$

$$\mathbf{X} \ge \mathbf{0}_{mn\times o}; \tag{31f}$$

$$\mathbf{Y} \in \{0,1\}^{o\times n}, \tag{31g}$$

where $\mathbf{1}_n$ is an $n \times 1$ vector with all elements being one, $\mathbf{0}_{mn\times o}$ is a $mn \times o$ matrix with all elements being zero, and $\odot$ represents the element-wise multiplication.

4.1.1. ADMM for the TR Problem in Matrix Form As outlined in the preceding discussion, we first introduce slack variables to convert inequality constraints to equivalent equalities. Following this modification, we present a reformulated matrix representation of TR:

$$\min \quad \mathbf{1}_n^\top \mathbf{b} \tag{32a}$$

$$\text{s.t.} \quad \begin{bmatrix} \mathbf{\Psi} \\ \mathbf{\Phi} \\ \mathbf{I} \end{bmatrix} \mathbf{X} - \begin{bmatrix} \mathbf{0} \\ \left(\mathbf{1} - \mathbf{Y}^\top\right) \odot \mathbf{B} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{S} \\ -\mathbf{R} \end{bmatrix} = \begin{bmatrix} \mathbf{D} \\ \mathbf{Y}^\top \odot \mathbf{C} \\ \mathbf{0} \end{bmatrix}, \tag{32b}$$

where $\mathbf{S} \ge \mathbf{0}$ and $\mathbf{R} \ge \mathbf{0}$ are introduced slack variables. Their elements are defined as follows:

$$\mathbf{S} \coloneqq \begin{bmatrix} s_{11} & \cdots & s_{1o} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{no} \end{bmatrix}, \quad \mathbf{R} \coloneqq \begin{bmatrix} r_{111} & r_{112} & \cdots & r_{11o} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1n1} & r_{1n2} & \cdots & r_{1no} \\ \vdots & \vdots & \ddots & \vdots \\ r_{mn1} & r_{mn2} & \cdots & r_{mno} \end{bmatrix}. \tag{33}$$

We have chosen not to express the standard matrix-vector multiplication format in equation (32b) by vectorizing $\mathbf{X}$ and $\mathbf{B}$ for the sake of clarity, but it is worth noting that such a transformation is feasible. Denote the Lagrangian multiplier by

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{bmatrix}, \tag{34}$$

where $\boldsymbol{\alpha} \in \mathbf{R}^{m\times o}$, $\boldsymbol{\beta} \in \mathbf{R}^{n\times o}$, and $\boldsymbol{\gamma} \in \mathbf{R}^{mn\times o}$. Concretely, they are defined as follows:

$$\boldsymbol{\alpha} \coloneqq \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1p} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mo} \end{bmatrix}, \boldsymbol{\beta} \coloneqq \begin{bmatrix} \beta_{11} & \cdots & \beta_{1o} \\ \vdots & \ddots & \vdots \\ \beta_{n1} & \cdots & \beta_{no} \end{bmatrix}, \boldsymbol{\gamma} \coloneqq \begin{bmatrix} \gamma_{111} & \gamma_{112} & \cdots & \gamma_{11o} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{1n1} & \gamma_{1n2} & \cdots & \gamma_{1no} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{mn1} & \gamma_{mn2} & \cdots & \gamma_{mno} \end{bmatrix}. \tag{35}$$

Then, the augmented Lagrangian function can be rewritten as

$$\mathcal{L}_\rho(\mathbf{X},\mathbf{B},\mathbf{S},\mathbf{R},\boldsymbol{\lambda}) = \mathbf{1}^\top\mathbf{b} - \mathbf{1}^\top\Big(\boldsymbol{\alpha}\odot\big[\boldsymbol{\Psi}\mathbf{X}-\mathbf{D}\big]\Big)\mathbf{1} - \mathbf{1}^\top\Big(\boldsymbol{\beta}\odot\Big[\boldsymbol{\Phi}\mathbf{X}-(\mathbf{1}-\mathbf{Y}^\top)\odot\mathbf{B}+\mathbf{S}-\mathbf{Y}^\top\odot\mathbf{C}\Big]\Big)\mathbf{1}$$
$$-\mathbf{1}^\top\Big(\boldsymbol{\gamma}\odot\big[\mathbf{X}-\mathbf{R}\big]\Big)\mathbf{1}+\frac{\rho}{2}\Big\|\boldsymbol{\Psi}\mathbf{X}-\mathbf{D}\Big\|_F^2+\frac{\rho}{2}\Big\|\boldsymbol{\Phi}\mathbf{X}-(\mathbf{1}_{n\times p}-\mathbf{Y}^\top)\odot\mathbf{B}+\mathbf{S}-\mathbf{Y}^\top\odot\mathbf{C}\Big\|_F^2+\frac{\rho}{2}\Big\|\mathbf{X}-\mathbf{R}\Big\|_F^2. \quad (36)$$

where $\|\cdot\|_F$ denotes the Frobinuous norm. When the ADMM framework is applied to (36), the iterative scheme reads as $\|A\|_F=\sqrt{\sum_{i=1}^m\sum_{j=1}^n|a_{ij}|^2}$

$$\begin{cases}
\mathbf{X}^{k+1} = \operatorname{argmin}\Big\{\mathcal{L}_\rho(\mathbf{X},\mathbf{B}^k,\mathbf{S}^k,\mathbf{R}^k,\boldsymbol{\lambda}^k)\,\Big|\,\mathbf{X}\in\mathbb{R}^{mn\times o}\Big\}, & (37a)\\
\mathbf{B}^{k+1} = \operatorname{argmin}\Big\{\mathcal{L}_\rho(\mathbf{X}^{k+1},\mathbf{B},\mathbf{S}^k,\mathbf{R}^k,\boldsymbol{\lambda}^k)\,\Big|\,\mathbf{B}\geq\mathbf{0}_{n\times o}\Big\}, & (37b)\\
(\mathbf{S}^{k+1},\mathbf{R}^{k+1}) = \operatorname{argmin}\Big\{\mathcal{L}_\rho(\mathbf{X}^{k+1},\mathbf{B}^{k+1},\mathbf{S},\mathbf{R},\boldsymbol{\lambda}^k)\,\Big|\,\mathbf{S}\geq\mathbf{0}_{n\times o},\ \mathbf{R}\geq\mathbf{0}_{mn\times o}\Big\}, & (37c)\\
\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k-\rho\begin{bmatrix}\boldsymbol{\Psi}\mathbf{X}^{k+1}-\mathbf{D}\\ \boldsymbol{\Phi}\mathbf{X}^{k+1}-(\mathbf{1}-\mathbf{Y}^\top)\odot\mathbf{B}^{k+1}+\mathbf{S}^{k+1}-\mathbf{Y}^\top\odot\mathbf{C}\\ \mathbf{X}^{k+1}-\mathbf{R}^{k+1}\end{bmatrix}. & (37d)
\end{cases}$$

The last $\boldsymbol{\lambda}$-subproblem can be solved with relative ease. We now shift our focus to the remaining three subproblems. Notably, these are not only amenable to analytical solutions, but also offer an excellent opportunity for parallel computation. As such, they contribute significantly to the overall computational efficiency of our proposed approach. To formalize this claim, we present the following proposition:

PROPOSITION 4. *The triad of subproblems, namely (37a), (37b), and (37c), have analytical solutions and are inherently amenable to parallel computation.*

We will elucidate Proposition 4 in the ensuing Subsection 4.1.2.

4.1.2. GPU-Adaptive Parallel Computing for ADMM Subproblems From the definition of $\mathbf{X}$ given in (30) and the computational logic provided in (37a), it is clear that the update procedure for each column of $\mathbf{X}$ is independent. To illustrate this, let's denote $\mathbf{x}_t$ $(t\in T)$ as the t-th column of $\mathbf{X}$. According to (37a), there is

$$\mathbf{x}_t = \operatorname{argmin}\Big\{-(\boldsymbol{\alpha}_t^k)^\top\boldsymbol{\Psi}\mathbf{x}_t-(\boldsymbol{\beta}_t^k)^\top\boldsymbol{\Phi}\mathbf{x}_t-(\boldsymbol{\gamma}_t^k)^\top\mathbf{x}_t+\frac{\rho}{2}\|\boldsymbol{\Psi}\mathbf{x}_t-\mathbf{d}_t\|_2^2+\frac{\rho}{2}\|\boldsymbol{\Phi}\mathbf{x}_t-\mathbf{z}_t^k\|_2^2+\frac{\rho}{2}\|\mathbf{x}_t-\mathbf{r}_t^k\|_2^2\ \Big|\ \mathbf{x}_t\in\mathbb{R}^{mn\times 1}\Big\}. \quad (38)$$

where $\boldsymbol{\alpha}_t^k,\boldsymbol{\beta}_t^k,\boldsymbol{\gamma}_t^k,\mathbf{d}_t,\mathbf{z}_t^k$ and $\mathbf{r}_t^k$ are the t-th column of $\boldsymbol{\alpha}^k,\boldsymbol{\beta}^k,\boldsymbol{\gamma}^k,\mathbf{D},(\mathbf{1_{n\times p}}-\mathbf{Y}^\top)\odot\mathbf{B}^k-\mathbf{S}^k+\mathbf{Y}^\top\odot\mathbf{C}$ and $\mathbf{R}^k$, respectively; and $\|\cdot\|_2^2$ denotes the ℓ_2 norm. Subsequently, invoking the first-order optimality condition, it follows that

$$\mathbf{x}_t^{k+1} = \Big(\boldsymbol{\Psi}^\top\boldsymbol{\Psi}+\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\mathbf{I}\Big)^{-1}\Big(\boldsymbol{\Psi}^\top\mathbf{d}_t+\boldsymbol{\Phi}^\top\mathbf{z}_t^k+\mathbf{r}_t^k+\frac{1}{\rho}(\boldsymbol{\Psi}^\top\boldsymbol{\alpha}_t^k+\boldsymbol{\Phi}^\top\boldsymbol{\beta}_t^k+\boldsymbol{\gamma}_t^k)\Big). \quad (39)$$

Through the application of (39) and fundamental principles of linear algebra, we are able to construct an analytic solution for updating $\mathbf{X}$, which is a compact representation of $\mathbf{x}_t$'s. That is

$$\mathbf{X}^{k+1} = \Big(\boldsymbol{\Psi}^\top\boldsymbol{\Psi}+\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\mathbf{I}\Big)^{-1}\Big(\boldsymbol{\Psi}^\top\mathbf{D}+\boldsymbol{\Phi}^\top\mathbf{Z}^k+\mathbf{R}^k+\frac{1}{\rho}(\boldsymbol{\Psi}^\top\boldsymbol{\alpha}^k+\boldsymbol{\Phi}^\top\boldsymbol{\beta}^k+\boldsymbol{\gamma}^k)\Big), \quad (40)$$

where $\mathbf{Z}^k = (\mathbf{1}_{n\times p} - \mathbf{Y}^\top) \odot \mathbf{B}^k - \mathbf{S}^k + \mathbf{Y}^\top \odot \mathbf{C}$. Additionally, (40) suggests that the update process for all $\mathbf{x}_t$ elements can be executed in parallel. This is because a matrix-matrix multiplication operation can be broken down into multiple matrix-column vector multiplication operations. With the application of modern numerical computation packages, which are highly optimized for hardware resources, these operations can be executed in parallel. This approach significantly improves computational efficiency, particularly when supported by multi-core CPU or GPU architectures.

Then, we delve into the intricacies of the $\mathbf{B}$-subproblem. Analogous to our previous analysis, we elucidate the detailed optimization problem integral to this subproblem according to (37b):

$$\begin{aligned}\mathbf{B}^{k+1} = \operatorname{argmin}\Big\{ &\sum_{j\in J} b_j - \sum_{j\in J} b_j\Big(\sum_{t\in T}(\beta_j^{(t)})^k (y_j^{(t)}-1)\Big) + \frac{\rho}{2}\sum_{j\in J} b_j^2\Big(\sum_{t\in T}(y_j^{(t)}-1)^2\Big) \\ &+ \rho\sum_{j\in J} b_j\Big(\sum_{t\in T}(y_j^{(t)}-1)\left([\mathbf{\Phi}\mathbf{X}^{k+1} + \mathbf{S}^k - \mathbf{Y}^\top\odot\mathbf{C}]_j^{(t)}\right)\Big) \;\Big|\; b_j \ge 0,\ \forall j\in J\Big\}.\end{aligned} \tag{41}$$

It is evident that (41) can be disaggregated into n distinct subproblems, with each subproblem encapsulating a single decision variable b_j:

$$\begin{aligned}b_j^{k+1} = \operatorname{argmin}\Big\{ &b_j - b_j\Big(\sum_{t\in T}(\beta_j^{(t)})^k (y_j^{(t)}-1)\Big) + \frac{\rho}{2} b_j^2\Big(\sum_{t\in T}(y_j^{(t)}-1)^2\Big) \\ &+ \rho\, b_j\Big(\sum_{t\in T}(y_j^{(t)}-1)\left([\mathbf{\Phi}\mathbf{X}^{k+1} + \mathbf{S}^k - \mathbf{Y}^\top\odot\mathbf{C}]_j^{(t)}\right)\Big) \;\Big|\; b_j \ge 0\Big\}.\end{aligned} \tag{42}$$

This kind of problem also yields an analytic solution, presented as follows:

$$b_j^{k+1} = \left[\left(\sum_{t\in T}(y_j^{(t)}-1)\left([\mathbf{Y}^\top\odot\mathbf{C} - \mathbf{\Phi}\mathbf{X}^{k+1} - \mathbf{S}^k]_j^{(t)} + \frac{(\beta_j^{(t)})^k}{\rho}\right) - \frac{1}{\rho}\right) \Big/ \sum_{t\in T}(y_j^{(t)}-1)^2\right]_+. \tag{43}$$

where the operation $[\cdot]_+$ is referred to as the positive part function, defined such that $[a]_+ = \max(0, a)$. By (43), we have the capability to concurrently solve these n subproblems to update b_j for all $j \in J$. Besides, we present a compact formulation for this update step, which allows us to fully exploit the parallel computation capabilities offered by modern numerical computation packages as mentioned above:

$$\mathbf{B} \coloneqq \underbrace{[\mathbf{b}, \mathbf{b}, \cdots, \mathbf{b}]}_{o}{}_{n\times o} \qquad \mathbf{b}^{k+1} = \left[\left(((\mathbf{Y}^\top - \mathbf{1}) \odot (\mathbf{Y}^\top\odot\mathbf{C} - \mathbf{\Phi}\mathbf{X}^{k+1} - \mathbf{S}^k + \boldsymbol{\beta}^k/\rho))\,\mathbf{1} - \mathbf{1}/\rho\right) \Big/ (\mathbf{Y}^\top - \mathbf{1})^2\,\mathbf{1}\right]_+. \tag{44}$$

where both vector-vector division and the positive part function are applied element-wise. By the definition of $\mathbf{B}$ given in (29), we can construct $\mathbf{B}^{k+1}$ by replicating $\mathbf{b}^{k+1}$ column-wise p times.

Subsequently, $\mathbf{S}$ and $\mathbf{R}$ are updated by solving (37c). Given the independence of these two decision variables, we can deconstruct (37c) into two distinct subproblems respective to each variable:

$$\begin{aligned}\mathbf{S}^{k+1} = \operatorname{argmin}\Big\{ &-\mathbf{1}^\top(\boldsymbol{\beta}^k\odot\mathbf{S})\,\mathbf{1}_{p\times 1} + \frac{\rho}{2}\|\mathbf{S}\|_F^2 \\ &+ \rho\Big[\mathbf{\Phi}\mathbf{X}^{k+1} - (\mathbf{1}_{n\times p} - \mathbf{Y}^\top)\odot\mathbf{B}^{k+1} - \mathbf{Y}^\top\odot\mathbf{C}\Big]\odot\mathbf{S} \;\Big|\; \mathbf{S}\ge\mathbf{0}_{n\times p}\Big\}.\end{aligned} \tag{45}$$

$$\mathbf{R}^{k+1} = \operatorname{argmin}\Big\{ \mathbf{1}_{mn\times 1}^\top(\boldsymbol{\beta}^k\odot\mathbf{R})\,\mathbf{1}_{p\times 1} + \frac{\rho}{2}\|\mathbf{R}\|_F^2 - \rho\,(\mathbf{X}\odot\mathbf{R}) \;\Big|\; \mathbf{R}\ge\mathbf{0}_{mn\times p}\Big\}. \tag{46}$$

Each of these subproblems yields an analytic solution:

$$\mathbf{S}^{k+1} = \Big[\boldsymbol{\beta}^k/\rho - \mathbf{\Phi}\mathbf{X}^{k+1} + (\mathbf{1}_{n\times p} - \mathbf{Y}^\top)\odot\mathbf{B}^{k+1} + \mathbf{Y}^\top\odot\mathbf{C}\Big]_+. \tag{47}$$

$$\mathbf{R}^{k+1} = \left[\mathbf{X}^{k+1} - \frac{\boldsymbol{\beta}^k}{\rho}\right]_+. \tag{48}$$

It is noteworthy that the update processes for $\mathbf{S}$ and $\mathbf{R}$ can be executed concurrently.

For ease of understanding and implementation, all steps of the proposed GPU-adaptive TR solver are consolidated in Algorithm 1.

4.1.3. GPU-adaptive TR Solver Based on ADMM Based on the analysis presented in Section 3.2 and 4.1, we outline the implementation details of ADMM for TR in Algorithm 1, which avoids the need for specifically designed algorithms within the ADMM framework by utilizing closed-form solutions.

Algorithm 1: GPU-Adaptive TR Solver Based on ADMM

Inputs : $\mathbf{Y}^{k+1}$, $\mathbf{B}^k$, $\mathbf{S}^k$, $\mathbf{R}^k$, $\boldsymbol{\lambda}^k$, ρ, and the number of iteration K'.

Outputs: $\mathbf{X}^{k+1}$, $\mathbf{B}^{k+1}$, $\mathbf{S}^{k+1}$, $\mathbf{R}^{k+1}$, $\boldsymbol{\lambda}^{k+1}$.

#variable initialization from the k-th step of the Two-Stage ALNS-ADMM;

$\mathbf{B}^0, \mathbf{S}^0, \mathbf{R}^0, \boldsymbol{\lambda}^0 \leftarrow \mathbf{B}^k, \mathbf{S}^k, \mathbf{R}^k, \boldsymbol{\lambda}^k$;

for $k' \leftarrow 1$ **to** K' **do**

- Update $\mathbf{X}^{k'}$ with (40);
- Update $\mathbf{b}^{k'}$ ($\mathbf{B}^{k'}$) with (44);
- Update $\mathbf{S}^{k'}$ and $\mathbf{R}^{k'}$ with (47) and (48) concurrently;
- Update $\boldsymbol{\lambda}^{k'}$ with (37d);

end

$\mathbf{B}^{k+1}, \mathbf{S}^{k+1}, \mathbf{R}^{k+1}, \boldsymbol{\lambda}^{k+1} \leftarrow \mathbf{B}^{K'}, \mathbf{S}^{K'}, \mathbf{R}^{K'}, \boldsymbol{\lambda}^{K'}$;

Algorithm 1 plays a crucial role in this study, as it necessitates rapid evaluation of local solutions within TE systems. Parallel computing is facilitated through matrix manipulations, as detailed in Subsection 4.1.2. This approach allows for simultaneous computation of all vectors within a matrix, which is further optimized by leveraging the computational power of GPUs.

4.2. Exploring BS at Cloud Edge through Matheuristic ALNS

Considering that BS contains a large number of feasible solutions, as highlighted in Proposition 1, we utilize the ALNS method strengthened by mathematical analysis for effective exploration of its feasible region. ALNS operates as an iterative local search framework, wherein various heuristic methods compete to enhance the quality of the current solution at each iteration. Within this process, a local solution undergoes a two-step modification: initially, a destroy heuristic operator, chosen from a predefined set of such operators, is applied to remove a portion of the current solution. Subsequently, a repair heuristic operator is employed to reconstruct and repair the solution. The newly formed solution is accepted and becomes the current solution

for the next iteration if it meets the criteria established by the local search framework. Given the iterative nature of ALNS, it is imperative that the solution at each iteration is evaluated swiftly, ensuring a timely progression of the search process. This requirement serves as a significant motivation for the adoption of ADMM in the evaluation phase, offering a balanced approach that combines efficiency and effectiveness.

In this subsection, we outline the proposed two-stage ADMM and ALNS algorithms in detail. Let $\mathbf{N}^-$ represent the set of destroy operators, $\mathbf{N}^+$ the set of repair operators, h the percentage of the current solution to be destroyed, and $f(\mathbf{y})$ the objective function evaluated by Algorithm 1. Algorithm 2 provides an overview of the ADMM-ALNS framework, with detailed descriptions of the operators to follow in the subsequent subsections. The master level of the ALNS local search framework employs simulated annealing, generating one solution per iteration.

Algorithm 2: Two-stage ADMM-ALNS framework

Inputs : $\boldsymbol{y}^*$, MAX_ITER_2, h.

Outputs: $\boldsymbol{y}^*$.

Initialize $\boldsymbol{y}$ with a greedy heuristic, $\boldsymbol{y}^* := \boldsymbol{y}$, $k = 0$;

Evaluate current solution $\boldsymbol{y}^*$ using ADMM described in Algorithm 1;

while $k < MAX\_ITER_2$ **do**

- Randomly select a destroy operator $d \in \mathbf{N}^-$ with respect to weights ;
- Use operator d to set $h \cdot mL$ elements of $\boldsymbol{y}^*$ that are 1 to 0;
- Randomly select a repair operator $r \in \mathbf{N}^+$ with respect to weights ;
- Use operator r to construct a new solution $\boldsymbol{y}'$;
- Evaluate current solution $\boldsymbol{y}'$ using ADMM described in Algorithm 1;
- **if** $\boldsymbol{y}$ *can be accepted* **then**
  - $\boldsymbol{y} := \boldsymbol{y}'$
- **end**
- Update weights of d, r;
- **if** $f(\boldsymbol{y}) < f(\boldsymbol{y}^*)$ **then**
  - $\boldsymbol{y}^* := \boldsymbol{y}$
- **end**

end

4.2.1. Matheuristic Operators Based on Dual Information of TR Dual heuristic operators comprise *dual destroy* and *dual repair* operators. These operators are designed based on the principles of complementary slackness and dual prices. Complementary slackness posits that a constraint will have a non-zero dual value if it is binding, whereas if the constraint is not binding, its dual value will be zero. On the other

hand, dual prices indicate how much the objective value would improve with an increase in resources. It is important to note that the only parameter subject to change in this context is $\mathbf{y}$. Different values of $\mathbf{y}$ lead to different TR models, each with its own set of resources.

In each iteration of ALNS, the execution of Algorithm 1 yields the dual values $\boldsymbol{\lambda} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$, as well as the slack variables $\boldsymbol{s} = (\boldsymbol{p}, \boldsymbol{q})$. Drawing upon the principles of complementary slackness, we can theoretically deduce that

$$\begin{cases} \boldsymbol{p} \cdot \boldsymbol{\alpha} = 0; \\ \boldsymbol{q} \cdot \boldsymbol{\gamma} = 0. \end{cases} \tag{49}$$

However, the execution of Algorithm 1 often terminates before optimal solutions are reached, resulting in the conditions of (49) not being strictly satisfied. Given that the primary objective of the algorithm in this study is to identify a feasible solution with moderate accuracy, we leverage the insights gained from Algorithm 1 to devise the two heuristic operators described below.

For the *dual destroy* heuristic operator, our objective is to identify the "binding constraints" that are hindering further decrease in the objective value. These are characterized by slack values that are close to or equal to 0, accompanied by large dual values. To implement this, we sort the indices $j \in J, t \in T$ in descending order based on the dual values in $\boldsymbol{\alpha}$, and select the top $h\%$ of indices where $y_{jt} = 1$ for destruction.

Conversely, the *dual repair* heuristic operator aims to find constraints that offer no potential for improving the objective function, indicated by dual values close to 0 or large slack values. Here, we sort the indices $j \in J, t \in T$ in descending order with respect to the slack values in $\mathbf{p}$, and select the top $h\%$ of indices where $y_{jt} = 0$ to repair the currently destroyed solution.

4.2.2. Matheuristic Operators Based on Lower Bound Once y_{jt} is fixed for all $j \in J$ and $t \in T$, the objective function of BACE-95 can be expressed as follows:

$$\sum_{j \in J} \max_{t \in T} \{ \sum_{(i,j) \in E} (a_{ij} x_{ijt} (1 - y_{jt})) \}. \tag{50}$$

Proposition 5. *The inequality* (51) *indicates that the right hand side can serve as a lower bound of the total cost on the left hand side.*

$$\sum_{j \in J} \max_{t \in T} (\sum_{(i,j) \in E} a_{ij} x_{ijt} \cdot (1 - y_{jt})) \geq \max_{t \in T} (D_t - \sum_{j \in J} (y_{jt} \cdot C_j))). \tag{51}$$

Proof. The proof is finalized by leveraging the principle that the 'peak of the sum is never greater than the sum of the peaks', as observed by Weinman (2012).

$$\begin{aligned} &\max_{t \in T} (\sum_{j \in J} \sum_{(i,j) \in E} (a_{ij} x_{ijt} \cdot (1 - y_{jt}))) = \max_{t \in T} (\sum_{j \in J} \sum_{(i,j) \in E} a_{ij} x_{ijt} - \sum_{j \in J} \sum_{(i,j) \in E} a_{ij} x_{ijt} \cdot y_{jt})) \\ &= \max_{t \in T} (\sum_{i \in I} D_{it} - \sum_{j \in J} y_{jt} \cdot \sum_{(i,j) \in E} a_{ij} x_{ijt})) \geq \max_{t \in T} (\sum_{i \in I} D_{it} - \sum_{j \in J} (y_{jt} \cdot C_j))) = \max_{t \in T} (D_t - \sum_{j \in J} (y_{jt} \cdot C_j))). \end{aligned}$$

□

The *greedy destroy* operator can be defined based on inequality (51). We iteratively remove part of the current solution based on the current bandwidth demand distribution.

The *greedy repair* operator aims to identify the time intervals that hold the greatest potential for improving the objective value. Following this, we iteratively allocate bandwidth demand to the chosen time intervals and cloud edge nodes.

4.2.3. Random Operators for Diversification The *dual heuristic operators* and *greedy operators* are both designed to guide the solution search process towards convergence. In contrast, the *random destroy* and *random repair* operators aim to diversify the search.

The *random destroy* operator randomly selects $h \cdot mL$ elements, which have a value of 1, and removes them from the current solution.

The *random repair* operator, on the other hand, randomly selects $h \cdot mL$ elements, which have a value of 0, and inserts them back into the partially destroyed solution to form a new feasible solution.

4.2.4. Master Local Search of ALNS At the master level of the ALNS, we use simulated annealing as the local search algorithm to guide the search for better solutions. A candidate solution $\mathbf{y}'$ is accepted given the current solution $\mathbf{y}$ with probability

$$e^{-\frac{f(\mathbf{y}')-f(\mathbf{y})}{T}} \tag{52}$$

where $T > 0$ is the *temperature*. The initial temperature is T_0, and T decreases according to the expression $T := T \cdot \eta$, where $0 < \eta < 1$ is the cooling rate.

In the adaptive layer of ALNS, roulette wheel selection is utilized to choose the destroy and repair operators. Denoting the $weight$ of an operator i as π_i, and the operator set as $N = N^+ \bigcup N^-$, then in each iteration we can choose an operator $j \in N^- \vee j \in N^+$ with probability

$$p_j = \frac{\pi_j}{\sum_{i \in N^- \vee i \in N^+} \pi_i} \tag{53}$$

In each iteration, the $weights$ of operators are updated based on their performance. The more an operator i contributes to the solution procedure, the larger the $weight$ π_i it receives, and consequently, the higher the probability that it will be chosen. It is important to note that the destroy and repair operators are chosen and updated separately. The heuristic weight of i in iteration k is incremented with the following values depending on the solution $\mathbf{y}'$, ensuring a dynamic adjustment of operator weights based on their performance.

σ_1: The most recent pair of destroy and repair operators yielded a new global best solution $\mathbf{y}'$.

σ_2: The most recent pair of destroy and repair operators produced a solution $\mathbf{y}'$ that, despite not being accepted, has a cost better than the cost of the current solution.

σ_3: The most recent pair of destroy and repair operators resulted in an accepted solution $\mathbf{y}'$, even though the cost of this new solution is worse than the cost of the current solution.

σ_4: The current solution has been rejected.

At the end of each iteration, the $weight$ of operator $j \in \mathbf{N}^+/\mathbf{N}^-$ is updated as follows:

$$\pi_j^{k+1} = (1-\Delta)\cdot\sigma_i + \Delta\cdot\pi_j^k, \quad \forall i \in \{1,2,3,4\} \tag{54}$$

where Δ controls the speed of weight adjustment, and k denotes the iteration count.

5. Numerical Results Based on Real-world Data

In this section, we present numerical results to demonstrate the performance of the proposed two-stage ADMM-ALNS algorithm. We have implemented the algorithms using Python and PyTorch. The numerical experiments were carried out on a PC equipped with an Intel Core i7-8700 CPU @ 3.20 GHz and 32 GB of RAM. In addition, parallel computing was facilitated by PyTorch, running on an NVIDIA Tesla P100 GPU paired with a Dual Xeon E5-2690 v4 CPU @ 2.6 GHz.

The instances used in our experiments are based on real-world data from a top-tier CSP in China. To avoid conflicts of interest, we have perturbed the anonymized data to create instances with a uniform distribution of demand data, ranging in [0.8, 1.5]. Small-scale instances are artificially generated and can be solved by CPLEX as benchmark to test the efficiency of the algorithm. Middle-scale instances represent a single day, consisting of $o = 288$ time intervals, $m = 31$ client group nodes (each representing a province in China), and $n = 95/131$ cloud edge nodes. Large-scale instances are derived from the online data within a month, totaling $o = 8928/8640$ time intervals. Our evaluation is conducted in two primary parts: first, we assess the efficiency and convergence properties of ADMM-based solver for TR, and we compare its performance with that of the commercial solver CPLEX. Subsequently, we conduct experiments using the two-stage ADMM-ALNS framework to address the original NP-hard problem BACE-95. Additionally, we extract insights that are beneficial for management practices from the results of our experiments.

5.1. Numerical Experiments on GPU-Adaptive TR Solver

In this subsection, we conduct experiments to evaluate the performance of ADMM-based TR solver in relation to its parameter ρ, the number of iterations required, and its capability to solve large-scale instances in comparison to CPLEX. The ADMM-based TR solver is initialized with a BS solution obtained from the TE system. The convergence rate of ADMM is significantly influenced by the choice of the parameter ρ. The optimal value of ρ is dependent on specific problem settings and the scale of the data being used. Based on preliminary experiments, we have decided to set ρ to 0.1 for the subsequent experiments.

5.1.1. Convergence Experiments with Respect to ρ and Iteration Numbers To determine the optimal combination of ρ and iteration numbers for upcoming experiments, we performed a series of tests under various ρ values and maximum iteration counts. The results, which include running times and objective values corresponding to each combination of ρ and max iteration number, are summarized in Table 3. The

gap is defined as gap = $\frac{\text{optimal}-\text{obj}}{\text{optimal}} \cdot 100\%$, representing the percentage difference between the optimal and current objective values. Notably, when MAX_ITER_1 is set to 10000, ADMM achieves convergence to the optimal solution across all tested ρ values, proving the method's robustness. With MAX_ITER_1 reduced to 5000, the average gap across all instances rises slightly to 0.23%, but the computational time is cut to just 55.7% of what is required for 10000 iterations. At $\text{MAX\_ITER}_1 = 1000$, however, the performance becomes more variable: only half of the instances achieve a convergence with a gap smaller than 5%, though the required time drops dramatically to a mere 7.5 seconds, significantly faster than in the previous scenarios. This analysis indicates that while a higher iteration count can guarantee robust performance across various settings of ρ, there are more efficient configurations that still provide satisfactory results, albeit with a slight compromise in solution quality. The challenge lies in balancing the trade-off between computational speed and the precision of the solution.

Table 3 Running time and convergence results for different values of ρ and iteration counts

	$\text{MAX\_ITER}_1 = 100$			$\text{MAX\_ITER}_1 = 1000$			$\text{MAX\_ITER}_1 = 5000$			$\text{MAX\_ITER}_1 = 10000$		
ρ	obj	t(s)	gap	obj	t(s)	gap	obj	t(s)	gap	obj	t(s)	gap
0.1	60575	4.6	10.08%	65990	7.5	2.05%	67448	20.5	-0.12%	67361	36.8	0.01%
0.2	88475	4.6	-31.33%	66924	7.5	0.66%	67475	20.5	-0.16%	67330	36.8	0.06%
0.3	119562	4.6	-77.48%	66833	7.5	0.79%	67367	20.5	0.00%	67350	36.8	0.03%
0.4	142102	4.6	-110.93%	67329	7.5	0.06%	67381	20.5	-0.02%	67379	36.8	-0.02%
0.5	159501	4.6	-136.76%	70024	7.5	-3.94%	67431	20.5	-0.09%	67388	36.8	-0.03%
0.6	173020	4.6	-156.83%	79120	7.5	-17.44%	67331	20.5	0.05%	67367	36.8	0.00%
0.7	183740	4.6	-172.74%	90698	7.5	-34.63%	67391	20.5	-0.03%	67376	36.8	-0.01%
0.8	192682	4.6	-186.01%	101534	7.5	-50.72%	67363	20.5	0.01%	67364	36.8	0.01%
0.9	200338	4.6	-197.38%	111190	7.5	-65.05%	67559	20.5	-0.28%	67379	36.8	-0.02%
1.0	206910	4.6	-207.13%	119980	7.5	-78.10%	68498	20.5	-1.68%	67368	36.8	0.00%
Average		4.6	-126.65%		7.5	-24.63%		20.5	**-0.23%**		36.8	**0.00%**

5.1.2. Robustness Experiments of the TR Solver To assess the trade-off between iteration count and algorithm effectiveness, and to evaluate the robustness of our TR solver across various instances, we conduct further experiments based on the results presented in Table 3. The gaps are defined as follows: $gap_1 = \frac{obj_0 - obj_1}{obj_0} \cdot 100\%$, $gap_2 = \frac{obj_0 - obj_2}{obj_0} \cdot 100\%$, $gap_3 = \frac{t_0 - t_1}{t_0} \cdot 100\%$, and $gap_4 = \frac{t_0 - t_2}{t_0} \cdot 100\%$, where subscript 0 refers to the benchmark (CPLEX), and 1 and 2 refers to the results from $ADMM_1$ and $ADMM_2$, respectively.

1) For $ADMM_1$, whose $\text{MAX\_ITER}_1 = 1000$: Across seven tested instances, the TR solver achieved an average optimality gap of 0.28% while reducing computation time by 77.75% compared to CPLEX.

2) For $ADMM_2$, whose $\text{MAX\_ITER}_1 = 5000$: The average gap was 0.06%, with a 32.1% reduction in computation time compared to CPLEX, across the same set of instances.

Practically, setting MAX_ITER_1 to 1000 provides a satisfactory solution for middle-scale instances and requires only about a quarter of the computation time needed by CPLEX.

Table 4 Convergence experiments on middle-scale instances compared with CPLEX

		Objective Value					Time Consumption				
Instance	m/n/o	CPLEX	$ADMM_1$	$ADMM_2$	gap_1	gap_2	CPLEX	$ADMM_1$	$ADMM_2$	gap_3	gap_4
1	31/95/288	67368	65990	67448	2.05%	-0.12%	29.8	7.5	20.5	74.83%	31.21%
2	31/131/288	2771682	2815617	2785523	-1.59%	-0.50%	38.9	9.3	29.5	76.09%	24.16%
3	31/95/288	342424	342540	342607	-0.03%	-0.05%	51.1	7.5	20.5	85.32%	59.88%
4	31/95/288	64931	63308	64520	2.50%	0.63%	32.6	9.4	29.5	71.17%	9.51%
5	31/95/288	126211	126090	125568	0.10%	0.51%	40.8	9.3	29.5	77.21%	27.70%
6	31/95/288	142775	142718	142571	0.04%	0.14%	47.1	9.4	29.5	80.04%	37.37%
7	31/95/288	562700	568851	563730	-1.09%	-0.18%	45.5	9.3	29.5	79.56%	35.16%
Average					**0.28%**	**0.06%**	41	8.8	26.9	**77.75%**	**32.10%**

5.1.3. Scalability of the GPU-Adaptive TR Solver on Large-Scale Instances While TR is often viewed as less challenging compared to BACE-95, our preliminary tests suggest that solving TR using traditional methods like the interior point method and simplex method can be highly time-consuming. Moreover, in some instances, these approaches might lead to an out-of-memory error. The primary challenge here arises from the problem's high dimensionality, characterized by multi-period decision-making and complex interdependencies among variables. Table 5 presents the results on Instance 8, which comprises

Table 5 Experiments on a real-world large-scale instance

		CPLEX		TR solver				
	m/n/o	obj_1	t_1(s)	obj_2	t_2(s)	MAX_ITER$_1$	gap_1	gap_2
Instance 8	31/95/8928	357983	85905	358724	10.2	100	-0.21%	8422
				341553	21	200	4.59%	4091
				349075	85.2	1000	2.49%	1008
				351670	408	5000	1.76%	211
				356459	810	10000	0.43%	106

over 26 million variables and 1 million constraints. We only provide results for this single large-scale instance to show the computational limitations of CPLEX. The gaps are defined as follows: $gap_1 = \frac{obj_1 - obj_2}{obj_1} \cdot 100\%$, and $gap_2 = \lfloor \frac{t_1}{t_2} \rfloor$. As the data in Table 5 reveals, CPLEX takes a staggering 85,905 seconds to solve Instance 8. In contrast, TR solver proves its efficiency by consistently delivering solutions within 5% of the optimum across all five experimental groups. Notably, the time required by CPLEX to solve the problem is between 106 to 8,422 times longer than that taken by TR solver. This significant difference highlights how impractical it is to directly incorporate TR with commercial solvers into TE systems, underscoring the critical role that our TR solver serves in these situations.

5.2. Numerical Experiments on the Two-Stage ADMM-ALNS Algorithm for BACE-95

In this subsection, our objective is to evaluate the performance of the ADMM-ALNS algorithm, particularly within the context of real-world applications. Typically, the studied CSPs employ offline bandwidth allocation

plans on a daily and monthly basis. Daily plans involve 288 five-minute intervals, while monthly plans cover 8928 or 8640 five-minute intervals. The monthly plan operates at a strategic level, and billing from ISPs occurs on a monthly basis. In addition, TE system engineers frequently use daily plans to evaluate the status of Traffic Engineering (TE) systems. Since CPLEX often struggles to solve practical instances, we select artificial small-scale instances to validate the effectiveness of our proposed algorithms, as detailed in Subsection 5.2.2. We then generate daily plans for monitoring the TE system, with corresponding test results presented in Subsection 5.2.3. Additionally, we generate monthly plans for scheduling a full billing cycle, and the experimental results are detailed in Subsection 5.2.4.

5.2.1. Parameter Settings for the Two-Stage ADMM-ALNS Algorithm Table 6 presents the parameter settings for the ALNS framework, as determined by our preliminary experiments. We employ the maximum number of iterations as the stopping criteria for both ALNS and ADMM. The MAX_ITER_1 parameter for ADMM is adjustable to meet specific requirements; for instance, it can be set to 200 for rapid convergence or 1000 for greater accuracy. Meanwhile, MAX_ITER_2 for ALNS can be either 500 or 1000, depending on the time constraints and practical requirements.

Table 6 Parameter settings

Parameter	Δ	σ_1	σ_2	σ_3	σ_4	T	η	ρ	h
Value	0.8	1.8	1.2	0.8	0.5	1000	0.95	0.1	0.2

5.2.2. Experiments on Instances Solvable by CPLEX as a Benchmark Due to the strong NP-hard nature of BACE-95, commercial solvers like CPLEX are limited to addressing only small-scale instances. To assess the efficacy of the ADMM-ALNS scheme, we generated small-scale artificial instances using CPLEX as a benchmark. For all numerical experiments, we set CPLEX's maximum runtime to 7200 seconds. The gap is defined as $gap = \frac{obj_1 - obj_0}{obj_0} \cdot 100\%$.

Table 7 Experimental comparison between CPLEX and ADMM-ALNS

Instance		CPLEX		ADMM-ALNS		
Index	m/n/o	UB (obj_0)	time (s)	obj_1	time (s)	gap
9	5/10/60	14682	5.01	15042	148	2.45%
10	5/10/60	42785	183.57	44166	149	3.22%
11	5/10/80	13711	7200.00	14664	146	6.95%
12	10/30/20	14961	7200.00	15622	145	4.42%
13	10/30/40	9893	7200.00	10234	151	3.45%
14	10/30/80	17938	7200.00	19011	159	5.98%
15	10/30/288	72910	7200.00	63822	390	-12.46%
16	10/30/288	147639	7200.00	136791	391	-7.35 %

Table 7 presents the results for these small-scale artificial instances. Instances 15 and 16 correspond to daily bandwidth allocation adjustments for subsets of cloud edge nodes and customer nodes, respectively. CPLEX managed to solve only two out of the eight instances. For the remaining instances, we have provided the best integer solutions found by CPLEX within the predefined time limit. Table 7 reveals that the proposed ADMM-ALNS consistently delivers high-quality solutions that rival those of CPLEX for instances 9-14. Moreover, when tackling slightly larger-scale instances such as 15 and 16, the ADMM-ALNS begins to exhibit superior performance over CPLEX. This superiority of ADMM-ALNS relative to CPLEX becomes even more pronounced as the problem scale grows, as further demonstrated in Subsections 5.2.3 and 5.2.4.

5.2.3. Experiments for Practical Middle-Scale Daily Evaluation Scenarios In practical scenarios, daily BS plans and real-time results are assessed by TR solvers to evaluate current status of the TE systems. The results obtained from the ADMM-ALNS algorithm serve as benchmarks for quantifying system scheduling. The gaps are defined as $gap_1 = \frac{obj_1}{obj_0} \cdot 100\%$ and $gap_2 = \frac{obj}{obj_0} \cdot 100\%$. Table 8 presents the performance of the ADMM-ALNS algorithm on middle-scale instances. Out of the six instances tested, CPLEX is able to generate a feasible solution only for instance 17, while the remaining instances result in an out-of-memory (OOM) issue in CPLEX.

Table 8 Experiments for ADMM-ALNS on middle-scale instances

Instance	m/n/o	CPLEX(obj_1)	UB(obj_0)	obj	time	gap_1	gap_2
17	31/95/288	198740	1150017	97985	1492	17.28%	8.52%
18	31/131/288	OOM	4610514	95223	1489	-	2.07%
19	31/95/288	OOM	1855515	639037	1475	-	34.44%
20	31/131/288	OOM	1173119	185260	1168	-	15.79%
21	31/131/288	OOM	1171063	177605	1162	-	15.17%
22	31/95/288	OOM	1140468	370408	746	-	32.48%
Average				1255		18.08%	

As illustrated in Table 8, ADMM-ALNS demonstrates its efficiency by producing a solution for instance 17 in just 1492 seconds, achieving a cost that is only half of what CPLEX manages in a full runtime of 7200 seconds. To establish a comparison benchmark, we introduce new upper bounds, denoted as obj_0, based on the initial plans currently utilized by the CSP under study, where limited operational research methodologies and techniques are employed. Impressively, ADMM-ALNS is capable of generating scheduling plans that, on average, cost only 18.08% of these upper bounds, while requiring merely 1255 seconds of computation time.

5.2.4. Experimental Evaluation of Large-Scale Monthly Scheduling and Billing Scenarios Commercial solver CPLEX is incapable of handling large-scale instances. To validate the effectiveness of the proposed methodologies, we conduct a comparison between the two-stage ADMM-ALNS and a real-world

greedy scheduling plan implemented by the studied CSP. Gap is defined as $gap = \frac{obj}{obj_0} \cdot 100\%$. Table 9 demonstrates that the ADMM-ALNS algorithm consistently generates solutions at only 32.73% of the upper bound costs, with an average runtime of 25,184 seconds. Given that monthly plans are generated once per month, this computational time is considered practical.

Table 9 Experiments for ADMM-ALNS on real-world large-scale instances

Instance	m/n/o	UB(obj_0)	obj	time	gap
23	31/95/8928	1376538	452805	29643	32.89%
24	31/131/8640	4840307	3041257	27362	62.83%
25	31/95/8928	1652733	535133	28095	32.38%
26	31/95/8928	1205784	210967	25168	17.50%
27	31/95/8928	1203516	202189	25363	16.80%
28	31/95/8928	1102093	374788	15473	34.01%
Average				25184	32.73%

In summary, the numerical experiments illustrate that the GPU-adaptive ADMM-based TR solver efficiently addresses the TR problem and serves as a rapid evaluation tool within the TE system, achieving acceleration of thousands of times compared to CPLEX. Additionally, the ADMM-ALNS algorithm can serve as a quantifying benchmark for strategic planning of bandwidth allocation for CSPs. The experiments reveal significant cost savings potential for CSPs when centralized TE systems, automatic bandwidth allocation techniques, and specifically designed optimization algorithms are applied.

6. Conclusions

This paper investigates the Bandwidth Allocation at Cloud Edge under the 95^{th} percentile billing scheme (BACE-95) problem, a critical topic gaining traction in both academic research and industrial practice due to the surging demand for cloud computing resources. Despite its importance, there remains a significant research gap—particularly in operations research and algorithm design—regarding efficient methods to address the unique challenges posed by BACE-95. To bridge this gap, we reformulate BACE-95 as a mixed-integer programming (MIP) problem and propose a GPU-adaptive two-stage ADMM-ALNS matheuristic algorithm tailored to the decomposed subproblems of traffic routing (TR) and bandwidth selection (BS). The TR problem leverages ADMM to address its high-dimensional, interconnected structure, while the BS subproblem employs the adaptive large neighborhood search (ALNS) algorithm for efficient exploration of the binary solution space. By reformulating the TR problem with a matrix-based representation and deriving closed-form solutions for the ADMM updates, we enable parallel computation, achieving substantial acceleration through GPU processing. Our approach differs from traditional CPU-based linear programming solvers by fully leveraging the parallel computing capabilities of modern GPU architectures, providing substantial computational benefits. Numerical experiments using real-world data from a tier-one cloud service

provider (CSP) in China demonstrate the remarkable efficiency of our TR solver. It delivers feasible solutions within 10 seconds for large-scale problems involving over 20 million continuous and binary variables and 1 million constraints. Compared to CPLEX, our TR solver achieves a speedup of over 8000 times while maintaining a solution gap within 0.5%. Furthermore, the two-stage ADMM-ALNS algorithm consistently delivers near-optimal solutions for small instances and robust, scalable results for medium to large instances, where commercial solvers struggle to perform. Overall, the proposed methods substantially enhance the studied CSP's traffic engineering (TE) system by improving computational efficiency, supporting strategic decision-making, and significantly reducing operational costs. These findings underscore the practical value of combining advanced algorithmic design with GPU-adaptive parallel computing in addressing critical challenges in modern cloud computing systems.

Looking ahead, the integration of GPUs into large-scale problems can be extended to tackle issues that traditional solvers find intractable. Furthermore, within the realm of cloud computing, ADMM can be established as a standard tool for evaluating TR, while the ALNS component can be adapted or replaced to accommodate various network structures. This aims to address a broader spectrum of bandwidth scheduling challenges that are prevalent in cloud computing environments.

References

Adler M, Sitaraman RK, Venkataramani H (2011) Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks* 55(18):4007–4020.

Adulyasak Y, Cordeau JF, Jans R (2014) Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science* 48(1):20–45.

Assi CM, Ye Y, Dixit S, Ali MA (2003) Dynamic bandwidth allocation for quality-of-service over ethernet pons. *IEEE Journal on selected Areas in Communications* 21(9):1467–1477.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329, ISSN 0030364X, 15265463, URL http://www.jstor.org/stable/222825.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1):238–252, ISSN 0029-599X, URL http://dx.doi.org/10.1007/BF01386316.

Chen C, He B, Ye Y, Yuan X (2016) The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming* 155(1-2):57–79.

Chou D, Xu T, Veeraraghavan K, Newell A, Margulis S, Xiao L, Ruiz PM, Meza J, Ha K, Padmanabha S, et al. (2019) Taiji: managing global user traffic for large-scale internet services at the edge. *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 430–446.

Christodoulopoulos K, Tomkos I, Varvarigos EA (2011) Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology* 29(9):1354–1366.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Operations Research* 8(1):101–111, ISSN 0030364X, 15265463, URL http://www.jstor.org/stable/167547.

Demir E, Bektaş T, Laporte G (2012) An adaptive large neighborhood search heuristic for the pollution-routing problem. *European journal of operational research* 223(2):346–359.

Desrosiers J, Lübbecke M (2011) *Branch-Price-and-Cut Algorithms*. ISBN 9780470400531, URL http://dx.doi.org/10.1002/9780470400531.eorms0118.

Douglas J, Rachford HH (1956) On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society* 82(2):421–439.

Feamster N, Rexford J, Zegura E (2014) The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review* 44(2):87–98.

Glowinski R, Marroco A (1975) Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 9(R2):41–76.

Guerin R, Ahmadi H, Naghshineh M (1991) Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on selected areas in communications* 9(7):968–981.

He B, Tao M, Yuan X (2012) Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization* 22(2):313–340.

He B, Tao M, Yuan X (2015) A splitting method for separable convex programming. *IMA Journal of Numerical Analysis* 35(1):394–426.

He B, Yuan X (2012) On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis* 50(2):700–709.

He B, Yuan X (2015) On non-ergodic convergence rate of douglas-rachford alternating direction method of multipliers. *Numerische Mathematik* 130, URL http://dx.doi.org/10.1007/s00211-014-0673-6.

Hong CY, Kandula S, Mahajan R, Zhang M, Gill V, Nanduri M, Wattenhofer R (2013) Achieving high utilization with software-driven wan. *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 15–26.

Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, Venkata S, Wanderer J, Zhou J, Zhu M, et al. (2013) B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review* 43(4):3–14.

Jalaparti V, Bliznets I, Kandula S, Lucier B, Menache I (2016) Dynamic pricing and traffic engineering for timely inter-datacenter transfers. *Proceedings of the 2016 ACM SIGCOMM Conference*, 73–86.

Kivanc D, Li G, Liu H (2003) Computationally efficient bandwidth allocation and power control for ofdma. *IEEE transactions on wireless communications* 2(6):1150–1158.

Land AH, Doig AG (1960) An automatic method of solving discrete programming problems. *Econometrica* 28(3):497–520, ISSN 00129682, 14680262, URL http://www.jstor.org/stable/1910129.

Liu X, Dobrian F, Milner H, Jiang J, Sekar V, Stoica I, Zhang H (2012) A case for a coordinated internet video control plane. *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 359–370.

Luo Y, Ansari N (2005) Bandwidth allocation for multiservice access on epons. *IEEE communications magazine* 43(2):S16–S21.

Masson R, Lehuédé F, Péton O (2013) An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science* 47(3):344–355, ISSN 00411655, 15265447, URL http://www.jstor.org/stable/43666874.

McGarry MP, Reisslein M, Maier M (2008) Ethernet passive optical network architectures and dynamic bandwidth allocation algorithms. *IEEE Communications Surveys & Tutorials* 10(3):46–60.

Mohan K, London P, Fazel M, Witten DM, Lee S (2013) Node-based learning of multiple gaussian graphical models. *CoRR* abs/1303.5145, URL http://arxiv.org/abs/1303.5145.

Mukerjee MK, Naylor D, Jiang J, Han D, Seshan S, Zhang H (2015) Practical, real-time centralized control for cdn-based live video delivery. *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 311–324.

Munkres J (1957) Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* 5(1):32–38.

NVIDIA Corporation (2024) Gpu performance background. URL https://docs.nvidia.com/deeplearning/performance/dl-performance-gpu-background/index.html, accessed: 2024-11-17.

Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review* 33(1):60–100, ISSN 00361445, URL http://www.jstor.org/stable/2030652.

Pentico DW (2007) Assignment problems: A golden anniversary survey. *European Journal of Operational Research* 176(2):774–793.

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. *Computers & operations research* 34(8):2403–2435.

Precedence Research (2022) Cloud computing market size to hit us$ 1,614.1 billion by 2030. URL https://www.globenewswire.com/en/news-release/2022/05/13/2443081/0/en/Cloud-Computing-Market-Size-to-Hit-US-1-614-1-Billion-by-2030.html.

Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science* 40(4):455–472.

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. Maher M, Puget JF, eds., *Principles and Practice of Constraint Programming — CP98*, 417–431 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-540-49481-2.

Singh R, Agarwal S, Calder M, Bahl P (2021) Cost-effective cloud edge traffic engineering with cascara. *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 201–216 (USENIX Association), ISBN 978-1-939133-21-2.

Tao M, Yuan X (2011) Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization* 21(1):57–81, URL http://dx.doi.org/10.1137/100781894.

Weinman J (2012) *Cloudonomics: The business value of cloud computing* (John Wiley & Sons).

Yuan X, Zhao P, Hu H, You J, Yang C, Peng W, Kang Y, Teo KM (2024) Huawei cloud adopts operations research for live streaming services to save network bandwidth cost: The gsco system. *INFORMS Journal on Applied Analytics* 54(1):37–53.

Zhan Y, Ghamkhari M, Akhavan-Hejazi H, Xu D, Mohsenian-Rad H (2016) Optimal response to burstable billing under demand uncertainty.