Robust Resource Provisioning in Time-Varying Edge Networks

Ruozhou Yu
North Carolina State University
Raleigh, NC

Guoliang Xue, Yinxin Wan
Arizona State University
Tempe, AZ

Jian Tang
Syracuse University
Syracuse, NY

Dejun Yang
Colorado School of Mines
Golden, CO

Yusheng Ji
National Institute of Informatics
Tokyo, Japan

ABSTRACT

Edge computing is one of the revolutionary technologies that enable high-performance and low-latency modern applications, such as smart cities, connected vehicles, etc. Yet its adoption has been limited by factors including high cost of edge resources, heterogeneous and fluctuating demands, and lack of reliability. In this paper, we study resource provisioning in edge computing, taking into account these different factors. First, based on observations from real demand traces, we propose a time-varying stochastic model to capture the time-dependent and uncertain demand and network dynamics in an edge network. We then apply a novel robustness model that accounts for both expected and worst-case performance of a service. Based on these models, we formulate edge provisioning as a multi-stage stochastic optimization problem. The problem is NP-hard even in the deterministic case. Leveraging the multi-stage structure, we apply nested Benders decomposition to solve the problem. We also describe several efficiency enhancement techniques, including a novel technique for quickly solving the large number of decomposed subproblems. Finally, we present results from real dataset-based simulations, which demonstrate the advantages of the proposed models, algorithm and techniques.

CCS CONCEPTS

• **Networks** → **Network resources allocation**; **Traffic engineering algorithms**; **Network management**; **Mobile networks**.

KEYWORDS

Edge Computing, Resource Allocation, Time-Varying, Robustness, Multi-Stage Stochastic Optimization

ACM Reference Format:
Ruozhou Yu, Guoliang Xue, Yinxin Wan, Jian Tang, Dejun Yang, and Yusheng Ji. 2020. Robust Resource Provisioning in Time-Varying Edge Networks. In *The Twenty-first ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20), October 11–14, 2020, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397166.3409146

Mobihoc '20, October 11–14, 2020, Boston, MA, USA

ACM ISBN 978-1-4503-8015-7/20/10…$15.00
https://doi.org/10.1145/3397166.3409146

1 INTRODUCTION

Advances in machine learning, computer vision, big data and more have enabled many new applications that employ connected intelligence to improve human lives, some of the most promising ones including smart home, smart city, and autonomous driving, to name a few. These applications gather the massive data generated by networked devices, and use centralized computing for data analytics and decision making. With more and more devices connected via the Internet-of-Things, however, our existing communication and computing infrastructures are being challenged. How to meet application performance goals while facing excessive load is becoming a critical problem both technically and commercially, which hinders the adoption of new life-changing applications.

Our current Internet-based cloud platforms fail to provide high-throughput, low-latency and reliable services due to unpredictability of the Internet. This has motivated the development of edge computing, where computing resources are provided via distributed nodes in the edge network. Edge computing enables services deployment proximal to end devices and users, both improving throughput and reducing latency. Yet edge computing has its own limitations, such as high cost, limited capacity, unreliable networks, etc. Efficient utilization of edge resources thus becomes a crucial problem.

In this paper, we study resource provisioning for edge services. We consider an integrated provisioning problem that involves service deployment, network provisioning, and simultaneously estimating and optimizing service performance. Different factors are accounted for during provisioning, including cost budget of the service provider, limited network resources, service robustness, and fluctuating demands and network condition. For the last consideration, we base our study on analysis of real-world datasets. We observe that some applications have very heterogeneous demands across both time and space. Moreover, in many cases the demands follow temporal-spatial repeating patterns, which is commonly due to the repeating behaviors of its users and devices. Such patterns can be crucial in performance enhancement and resource utilization. We propose a time-varying stochastic model to account for these patterns, which allows finer-grained resource provisioning. We further consider service performance robustness in terms of average user quality-of-service (QoS). We apply a balanced robustness metric, named Conditional Value-at-Risk (CVaR), which can flexibly trade-off between the expected and worst-case service performance.

Based on these models, we formulate robust edge provisioning as a three-stage stochastic optimization problem. The problem is a large-scale mixed integer linear program (MILP), and is NP-hard even in the deterministic case. We propose an optimal algorithm

based on *nested Benders decomposition*, which utilizes the decomposable structure of the problem. The original Benders decomposition suffers from slow convergence. We then describe efficiency enhancement techniques, including both established ones that apply to our problem, and a novel analytical solution to the decomposed subproblems. To validate our study, we have conducted simulations based on datasets synthesized from real data traces. The results show that our model enables fine-grained and robust provisioning of edge resources, the proposed algorithm generates superior solutions over heuristic approaches, and the described efficiency enhancement techniques are effective in reducing convergence time. To summarize, our contributions are as follows:

- We observe temporal-spatial repeating demand patterns from real-world traces, and formulate a three-stage edge resource provisioning problem with cost, resource, and robustness considerations. Edge provisioning with time-varying and temporal-spatial repeating demands has yet been studied to the best of our knowledge.
- We propose a nested Benders decomposition-based optimal approach to this NP-hard problem, as well as several efficiency enhancement techniques.
- We present results from real data trace-based simulations, which show the advantages of the proposed models, algorithm, and techniques.

The rest of this paper is organized as follows. Sec. 2 presents our observations and our system model. Sec. 3 presents our problem and states its complexity. Sec. 4 presents our optimal algorithm and the efficiency enhancement techniques. Sec. 5 presents our simulation results. Sec. 6 discusses related work in the literature. Finally, Sec. 7 concludes this paper.

2 SYSTEM MODEL AND OVERVIEW

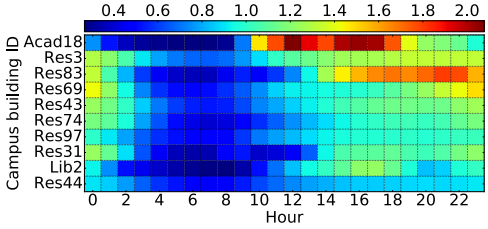
2.1 Edge System Model

The edge network is modeled as a directed graph $G = (N, L)$, where N is the set of nodes, and L is the set of links between nodes. The network consists of *computing nodes* (hosts) denoted by $H \subseteq N$, and *access points* (APs) denoted by $A \subseteq N$. Each link $l \in L$ has a bandwidth capacity b_l. We assume the network is strongly connected. A weakly connected or disconnected network can be made strongly connected by adding artificial zero-bandwidth links.

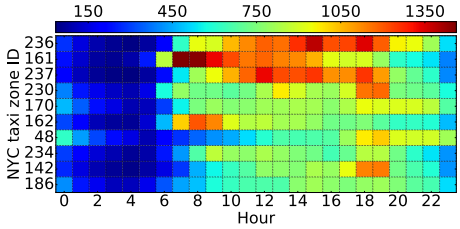
Three parties exist in the system. The *network manager* (NM) manages the network, including all network nodes and links. The *edge computing manager* (ECM) manages the computing resources on computing nodes. Finally, the *edge service provider* (ESP) operates an *edge service*. The ESP's goal is to deploy its service in the edge network to serve external user demands. We consider an edge service receiving and processing continuous data streams from APs. Typical examples include crowdsensing services that aggregate and analyze sensing data from mobile users, smart transportation services that receive data from sensors and vehicles and perform traffic scheduling, and emergency services that monitor and extract eventual information. The service can be deployed on one or multiple distributed computing nodes. However, as edge resources are expensive, universal service deployment may be impractical. Formally, let us use c_h to denote the cost of deploying and operating the edge service on node $h \in H$. The cost budget of the ESP is C.

2.2 Edge Demand Model

While demand patterns differ across applications, a common characteristic of edge demand is its frequent temporal-spatial dynamics. Such dynamics usually follow time-varying and repeating patterns over time. This could be due to many factors such as repeating user activities, user mobility, and duty-cycling services, and it can be observed on both small-scale and large-scale systems. For example, Fig. 1(a) shows the average-per-hour number of mobile devices connected to top-10 popular building APs in a campus network from 09/2002 to 09/2003 [15]. Fig. 1(b) shows the average number of passengers dropped-off by Yellow Taxi at top-10 popular taxi zones in NYC in 2018 [27]. Though measuring different data, both datasets show the different demands in different hours over a day, which most likely attributes to the repeating daily activities of users.



(a) Top 10 Dartmouth College buildings: average connected devices



(b) Top 10 New York City (NYC) taxi zones: average passenger drop-offs

Figure 1: Time-varying demands from two different datasets: (a) average number of devices conected to campus APs, and (b) average passenger drop-offs at NYC taxi zones.

Motivated by the observation, we adopt a time-varying demand model. We divide time into T time slots, each representing a demand pattern observed over time. For simplicity, we assume all slots are of equal lengths and are successively repeating, and denote the course of time over all slots as a *period*. Our model and solution can be trivially extended to handle time slots with different lengths and repeating patterns. From Fig. 1, demand distributions can be neither identical nor independent across APs in one time slot, or across different slots at the same AP. Specifically, demand at AP $a \in A$ in slot $t = 1 \dots T$ is assumed to follow a (possibly unknown) distribution, which we denote by random variable $\delta_{t,a}$; the total demand in t is defined as $\delta_t = \sum_a \delta_{t,a}$. This reflects the fact that even in the same slot, the exact demand may still fluctuate across periods due to unattended factors such as special events, seasonal changes, etc. Under this model, provisioning is done for each slot across different periods, instead of being fixed at all time.

2.3 Network Provisioning Model

Network has a critical impact on edge service performance. Ideally, the ESP wishes to attain optimal network performance via frequent

or real-time network optimization. However, the network is managed by the NM (*e.g.*, an ISP) and hence out of ESP's control. We assume the NM can process network optimization requests from the ESP, but not frequently or in real-time due to the complexity of network management and the large number of supported services.

Based on the time-varying demand model, the ESP employs the following strategy. During initial provisioning, the ESP submits a network optimization request for each time slot with demand information. The NM provisions a dedicated networking policy for each slot based on the requests. After the service starts operating, the NM updates its policy based on the time slots. If the demand patterns change, the ESP can re-submit those requests, and the NM will derive new policies and operate based on them thereafter.

Routing. The NM's network policy can greatly impact the service performance. Ideally, flow-based optimization (*e.g.*, min-cost multi-commodity flow [1]) can lead to optimal routing and bandwidth allocation for maximum service performance. Practically, the high complexity of flow-based algorithms discourages their usage in real networks. Fortunately, recent advances in *pathbook routing* have shown that a small set of carefully selected paths (*e.g.*, 3–5 paths per commodity) can achieve close-to-optimal throughput in real networks, while traffic engineering overhead can be drastically reduced compared to flow-based algorithms [16]. Since the NM may serve many ESP requests at the same time, we believe that pathbook routing is suitable as an efficient yet adequate routing solution for the NM. Formally, we use $P = \bigcup\{P_{a,h} \mid a \in A, h \in H\}$ to denote the pathbook used by the NM. We only consider paths that connect APs to hosts for a specific ESP. Based on ESP requests, the NM derives bandwidth allocation across paths for each time slot, subject to the available bandwidth on network links.

Performance of an edge service is largely dependent on network performance of data streams. We assume ESP's goal is to optimize the QoS of its service, notably the network delay, when serving external demands. Network conditions may fluctuate due to congestion, failures and maintenance, malicious traffic, etc. To model such dynamics, we use random variable $d_{t,l}$ to denote the *delay* of a link $l \in L$ in time slot t. We note that link delay is independent of demand due to explicit demand allocation; see next section.

3 PROBLEM STATEMENT AND COMPLEXITY

The initial edge provisioning consists of the following steps. First, the ESP selects a set of computing nodes to deploy the service. Since service deployment incurs manual cost for setup and testing, it is *fixed* once the provisioning is done; live service migration is considered harmful due to interrupted service to users, and hence is not considered in this paper. Second, the ESP submits requests to the NM for network optimization based on demand and network statistics. After the provisioning is done, the ESP is presented with a service deployment plan, as well as a set of paths with pre-determined per-time slot bandwidth quotas. Unlike service deployment by the ECM, the NM can change bandwidth quotas by simply adjusting weights in the network without interrupting the service. Hence the bandwidth quotas can be updated in each time slot. In operations, the ESP adjusts demand allocation across paths in real-time, subject to the bandwidth quotas of the paths, to continuously serve user demands with optimized network performance.

3.1 Basic Problem Formulation

3.1.1 Service Deployment (SD). The first stage, service deployment, is formally denoted by variables $X = \{x(h) \in \{0,1\} \mid h \in H\}$, where $x(h) = 1$ means that edge node h hosts a service instance and vice versa. The deployment is bounded by the cost budget of the ESP:

$$\sum\nolimits_{h\in H} c_h x(h) \le C. \tag{1}$$

Further, the service must have at least one instance deployed:

$$\sum\nolimits_{h\in H} x(h) \ge 1. \tag{2}$$

3.1.2 Network Provisioning (NPR). Given fixed SD, the next step is to provision the network for each time slot. Define $Y = \bigcup_{t=1}^{T} Y_t$, where $Y_t = \{y(t,p) \ge 0 \mid p \in P\}$ is the set of variables for *pathbook bandwidth allocation* for slot t. First, the allocation is constrained by the deployed service instance, *i.e.*, a path can have positive bandwidth only if it connects an access point to a deployed instance:

$$y(t,p) \le b_p^{\max} x(h_p), \quad \forall t, p \in P, \tag{3}$$

where $b_p^{\max} = \min_{l\in p}\{b_l\}$ is the bottleneck bandwidth of the path, and h_p is the host destination of path p. Further, in each t, all paths using a link must obey the link's bandwidth capacity:

$$\sum\nolimits_{p\in P: l\in p} y(t,p) \le b_l, \quad \forall t, l \in L. \tag{4}$$

3.1.3 Network Estimation (NE). In normal operations, the ESP would flexibly allocate demands across paths subject to the pre-determined bandwidth allocation, in order to optimize instantaneous network performance. This information can also be used in initial provisioning, when the ESP estimates the *expected network performance* for a bandwidth allocation plan, and optimizes both service deployment and bandwidth allocation accordingly. Formally, define $Z = \bigcup_t Z_t$, where $Z_t = \{z(t,p) \ge 0 \mid p \in P\}$ is the set of (random) variables denoting demand allocation on path p in time t. The allocation is upper bounded by the bandwidth on each path:

$$z(t,p) \le y(t,p), \quad \forall t, p \in P. \tag{5}$$

The allocation must also serve all demand from each AP:

$$\sum\nolimits_{p\in P_a} z(t,p) \ge \delta_{t,a}, \quad \forall t, a \in A, \tag{6}$$

where $P_a = \bigcup_{h\in H} P_{a,h}$ contains all paths from a to a host.

3.1.4 Network Objective. With Constraint (6) serving all user demands, the ESP's goal is to optimize network QoS, *i.e.*, transmission delay of all user demands. Given the stochastic model of demands and delays, we use D_t to denote the weighted average delay of all demands at time t, which is also a random variable, where

$$D_t \triangleq \frac{1}{\delta_t} \sum\nolimits_{p\in P} d_{t,p} z(t,p). \tag{7}$$

and $d_{t,p} = \sum_{l\in p} d_{t,l}$ is the delay of a path. Consistent service performance is preferred, hence ESP can seek to optimize the highest delay across time. Informally, the problem can be expressed as

$$\min_{\mathcal{X}\in\mathcal{F}} \max_t \{D_t\}, \tag{8}$$

where $\mathcal{X} = (X, Y, Z)$, and $\mathcal{F}$ is the feasibility region by (1)–(6).

3.2 Stochastic Optimization and Robustness

Given the stochastic nature of D_t, a common practice is to optimize the *expectation* of the objective. However, expectation optimization has limited ability in handling infrequent adverse scenarios [36]. For the provider, an elastic and fault-tolerant system produces more consistent revenue and lowers business risk in the long run. For

the users, a service with more stable performance leads to better user experience and higher satisfaction. Both perspectives coincide on the importance of *robustness* in performance optimization.

To counter performance fluctuation, we use a model called the *Conditional Value-at-Risk* (CVaR), a common risk measure in financial risk management. CVaR is widely used to measure the expected investment loss that an investor faces when market undergoes unfavorable variations. Formally, denote random variable $\mathbf{R}$ as the investor's loss, then given a confidence value $\alpha \in [0,1]$, define

$$\mathrm{VaR}_\alpha(\mathbf{R}) \triangleq \min\{r \mid \Pr[\mathbf{R} \le r] \ge \alpha\}, \tag{9}$$

$$\mathrm{CVaR}_\alpha(\mathbf{R}) \triangleq \mathbb{E}[\mathbf{R} \mid \mathbf{R} \ge \mathrm{VaR}_\alpha(\mathbf{R})]. \tag{10}$$

$\mathbb{E}(\cdot)$ denotes the expectation of a random variable. Eq. (9) defines the *Value-at-Risk* (VaR) with confidence α, representing the minimum value such that with α confidence, the investment loss will not exceed this value. Eq. (10) then defines the CVaR as the expected value of loss in all scenarios where the loss exceeds the VaR, *i.e.*, the expected loss in the worst $(1-\alpha)$ scenarios.

Notably, CVaR defines a trade-off between two objectives, controlled by α. If $\alpha = 0$, Eq. (10) equals to expectation of $\mathbf{R}$ over the entire distribution. If $\alpha \to 1$, CVaR converges to the maximum loss over all possible scenarios, and minimizing CVaR equals to *minimax* in robust optimization. While expectation lacks robustness at all, minimax addresses only the worst case; in practice, minimax could reduce worst-case loss by an arbitrarily small amount, but drastically increase loss in all other cases. CVaR avoids both issues.

3.3 CVaR-based Formulation

Back to our problem, the ESP may seek to optimize both the expected average delay and CVaR of the average delay. To strike a balance, we formulate a *scalarized* two-objective problem:

$$\min_{X \in \mathcal{F}} \max_t \{\rho_1 \cdot \mathbb{E}[D_t] + \rho_2 \cdot \mathrm{CVaR}_\alpha(D_t)\}, \tag{11}$$

Parameters ρ_1 and ρ_2 define the trade-off between expectation and robustness, and can be tuned based on the ESP's actual objective.

In Eq. (10), the computation of CVaR requires first computing VaR, which itself does not have a closed form. Fortunately, Rockafellar and Uryasev have shown in [24] that the CVaR can be directly computed using the following formula:

$$\mathrm{CVaR}_\alpha(D_t) = \min_{r(t)\in\mathbb{R}} \left\{ r(t) + \frac{1}{1-\alpha}\mathbb{E}\left[(D_t - r(t))^+\right] \right\}, \tag{12}$$

where $(D_t - r(t))^+ = \max\{D_t - r(t), 0\}$, and $\mathbb{R}$ is the real number set. Note that in (12), the VaR is automatically computed as the minimizing value of $r(t)$. Merging (12) into Program (11), we have

$$\min_{X\in\mathcal{F}, R} \max_t \left(\rho_1 \mathbb{E}[D_t] + \rho_2 \left(r(t) + \frac{1}{1-\alpha}\mathbb{E}\left[(D_t - r(t))^+\right] \right) \right). \tag{13}$$

where $R = \{r(t)\}$ is the set of VaR variables for all time slots.

3.4 Sample Average Approximation (SAA)

Program (13) is hard to solve even with known distributions, since the solution cannot be derived in a closed form. Hence for each t, we use a set of independently and identically distributed (i.i.d.) samples $\mathcal{S}_t = (S_t^1, \ldots, S_t^K)$ to approximate the distributions, where each $S_t^k = \{\delta_{t,a}^k \mid a \in A\} \cup \{d_{t,l}^k \mid l \in L\}$ contains a realization of the demands and delays in time t. With fixed SD and NPR, the optimal average delay of a sample can be computed by solving (5)–(8).

Let $Z_t^k = \{z(t,k,p)\}$ be the demand allocation variables for sample k of time slot t, and D_t^k be the optimal average delay (under (5)–(6)) for k. We use $\mathcal{Z}_t = \bigcup_k Z_t^k$ and $\mathcal{Z} = \bigcup_t \mathcal{Z}_t$ to denote the extended rate allocation sets over all samples of a time slot and all slots, respectively. The expected average delay $\mathbb{E}[D_t]$ is approximated by the sample average $\frac{1}{K}\sum_{k=1}^K D_t^k$. Similarly, $\mathrm{CVaR}_\alpha(D_t)$ is approximated by $\min_{r(t)\in\mathbb{R}} \left\{ r(t) + \frac{1}{1-\alpha}\frac{1}{K}\sum_{k=1}^K \left(D_t^k - r(t)\right)^+ \right\}$. To completely re-formulate Program (13) using sample averages, we define auxiliary variables $W = \bigcup_t W_t$, where $W_t = \{w(t,k) \ge 0 \mid k = 1 \ldots K\}$. We also linearize the maximum term in the objective function (13) using an additional variable $D \in \mathbb{R}$. Then Program (13) can be re-formulated as a mixed integer linear program (MILP):

$$\min_{\substack{X,Y,\mathcal{Z},\\ R,W,D}} \quad D \tag{14a}$$

$$\text{s.t.} \quad D \ge \rho_1 \frac{1}{K}\sum\nolimits_{k=1}^K D_t^k + \rho_2 \left(r(t) + \frac{1}{1-\alpha}\frac{1}{K}\sum\nolimits_{k=1}^K w(t,k) \right), \forall t; \tag{14b}$$

$$w(t,k) \ge D_t^k - r(t), \quad \forall t,k; \tag{14c}$$

$$(1)\text{–}(4), \text{ and } \forall t,k, (5)\text{–}(6).$$

We call this the **Robust Edge Provisioning (REP)** problem. It is well-known that the SAA problem converges arbitrarily close to the original problem with a sufficient number of i.i.d. samples [14].

3.5 Computational Complexity

Theorem 3.1. *REP is NP-hard.* □

Proof. Neglecting stochastics, the problem can be reduced from Knapsack, where $\mathbf{n}$ items each with a cost $\mathbf{c}_i$ and a value $\mathbf{v}_i$ are to be put into a knapsack with maximum cost $\mathbf{C}$ and minimum value $\mathbf{V}$. Consider an instance of REP with $\mathbf{n}$ edge computing nodes in H, all connected to a single AP $a \in A$, each with deployment cost $\mathbf{c}_i$ and bandwidth $\mathbf{v}_i$. The ESP has a budget of $\mathbf{C}$, and a has a constant demand of $\mathbf{V}$. Then deciding if the REP instance is feasible is equal to solving the Knapsack instance. REP's NP-hardness follows. □

4 REP WITH MULTI-STAGE OPTIMIZATION

Due to problem NP-hardness and its scale, a decomposition-based algorithm is proposed based on the structure of the problem.

4.1 Problem Decomposition

From its definition, the problem can be vertically decomposed into three stages. At the later stages (NPR and NE), the corresponding subproblems can further be horizontally decomposed. The first-stage SD problem is an MILP:

$$\mathcal{O} \triangleq \min_{X,D} D \text{ s.t. } \forall t, D \ge \mathcal{P}_t(X); (1)\text{–}(2). \tag{15}$$

The second-stage NPR subproblem given fixed X is an LP:

$$\mathcal{P}_t(X) \triangleq \min_{Y_t, r(t)} \rho_2 r(t) + \frac{1}{K}\sum\nolimits_k \mathcal{Q}_{t,k}(Y,R) \quad \text{s.t. } (3)\text{–}(4). \tag{16}$$

The third-stage NE subproblem given (Y,R) is also an LP:

$$\mathcal{Q}_{t,k}(Y,R) \triangleq \min_{\mathcal{Z},W} \rho_1 \cdot D_t^k + \rho_2 \cdot \frac{1}{1-\alpha} \cdot w(t,k) \quad \text{s.t. } (14c);\ \forall t,k,(5)\text{–}(6). \tag{17}$$

We use $Q_{t,k}(Y,R)$ to omit index t at variables Y and R as function arguments. Similar notations are used hereafter to simplify notations. Eqs. (15)–(17) show that REP has a natural vertical-horizontal three-stage decomposition, where each $\mathcal{P}_t(X)$ subproblem is independent from its siblings, and so is each $Q_{t,k}(Y,R)$. Note that variables R and W, though both used to formulate CVaR, have been separated by the NPR–NE decomposition to enable full independence among $Q_{t,k}(Y,R)$ subproblems. This decomposition motivates us to adopt a decomposition-based algorithm to address the REP problem.

4.2 Benders Decomposition Basics

Benders decomposition is an iterative algorithm for solving multi-stage decomposable optimization problems [3]. Consider a general bounded two-stage problem in the following equal forms (assume that for any $\mathbf{X} \in \boldsymbol{\mathcal{X}}$ the problem is bounded):

$$\min_{\mathbf{X}\in\boldsymbol{\mathcal{X}},\mathbf{Y}\geq 0}\{\mathbf{dX}+\mathbf{eY} \mid \mathbf{AX}+\mathbf{BY}\geq \mathbf{f}\} \tag{18}$$

$$= \min_{\mathbf{X}\in\boldsymbol{\mathcal{X}}}\{\mathbf{dX}+\min_{\mathbf{Y}\geq 0}\{\mathbf{eY} \mid \mathbf{BY}\geq \mathbf{f}-\mathbf{AX}\}\} \tag{19}$$

$$= \min_{\mathbf{X}\in\boldsymbol{\mathcal{X}}}\{\mathbf{dX}+\max_{\boldsymbol{\lambda}\geq 0}\{(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda} \mid \mathbf{B}'\boldsymbol{\lambda}\leq \mathbf{e}\}\}, \tag{20}$$

where $\boldsymbol{\lambda}$ denotes the dual variables for the subproblem with only $\mathbf{Y}$. The inner subproblem is called the *slave*, while the outer problem with only $\mathbf{X}$ is the *master*. Eq. (19) is via decomposition. Eq. (20) is via strong duality of the slave as an LP. Note in (20), the dual slave's feasibility region is independent of $\mathbf{X}$. Let $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}\geq 0 \mid \mathbf{B}'\boldsymbol{\lambda}\leq \mathbf{e}\}$ be the dual feasibility region of the slave. We can re-write (20) using only the extreme points $\boldsymbol{\Lambda}_{\mathrm{o}}^*$ and extreme rays $\boldsymbol{\Lambda}_{\mathrm{f}}^*$ in $\boldsymbol{\Lambda}$:

$$\min_{\mathbf{X}\in\boldsymbol{\mathcal{X}},\boldsymbol{\theta}} \quad \mathbf{dX}+\boldsymbol{\theta} \tag{21a}$$

$$\text{s.t.} \quad \boldsymbol{\theta}\geq(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda}_{\mathrm{o}}, \quad \forall\boldsymbol{\lambda}_{\mathrm{o}}\in\boldsymbol{\Lambda}_{\mathrm{o}}^*, \tag{21b}$$

$$(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda}_{\mathrm{f}}\leq 0, \quad \forall\boldsymbol{\lambda}_{\mathrm{f}}\in\boldsymbol{\Lambda}_{\mathrm{f}}^*. \tag{21c}$$

An extreme ray $\boldsymbol{\lambda}_{\mathrm{f}}$ is a direction to which the dual slave objective can increase without bound, which exists only if the primal slave is infeasible for some $\mathbf{X}\in\boldsymbol{\mathcal{X}}$. Eq. (21c) eliminates such an infeasible $\mathbf{X}$ by bounding such a ray to be below 0.

It is impractical to enumerate all extreme points and extreme rays for a given problem. Hence, Benders decomposition works on a relaxed version of (21) that includes only a subset of the extreme points $\boldsymbol{\Lambda}_{\mathrm{o}}\subseteq\boldsymbol{\Lambda}_{\mathrm{o}}^*$ and extreme rays $\boldsymbol{\Lambda}_{\mathrm{f}}\subseteq\boldsymbol{\Lambda}_{\mathrm{f}}^*$:

$$\min_{\mathbf{X}\in\boldsymbol{\mathcal{X}},\boldsymbol{\theta}} \quad \mathbf{dX}+\boldsymbol{\theta} \tag{22a}$$

$$\text{s.t.} \quad \boldsymbol{\theta}\geq(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda}_{\mathrm{o}}, \quad \forall\boldsymbol{\lambda}_{\mathrm{o}}\in\boldsymbol{\Lambda}_{\mathrm{o}}, \tag{22b}$$

$$(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda}_{\mathrm{f}}\leq 0, \quad \forall\boldsymbol{\lambda}_{\mathrm{f}}\in\boldsymbol{\Lambda}_{\mathrm{f}}. \tag{22c}$$

If a solution $(\mathbf{X},\boldsymbol{\theta})$ obtained by solving (22) fully satisfies (21b) and (21c), then $(\mathbf{X},\boldsymbol{\theta})$ is optimal to the original problem. This can be checked by solving the dual slave subproblem $\overline{\boldsymbol{\theta}} = \max_{\boldsymbol{\lambda}\geq 0}\{(\mathbf{f}-\mathbf{AX})\boldsymbol{\lambda} \mid \mathbf{B}'\boldsymbol{\lambda}\leq\mathbf{e}\}$ and checking whether $\boldsymbol{\theta}\geq\overline{\boldsymbol{\theta}}$. If $\overline{\boldsymbol{\theta}}=\infty$, then $\mathbf{X}$ is infeasible, and a new *feasibility cut* $(\mathbf{f}-\mathbf{AX})\widetilde{\boldsymbol{\lambda}}\leq 0$ can be added, where $\widetilde{\boldsymbol{\lambda}}$ is an extreme ray of the unbounded dual. Otherwise if $\boldsymbol{\theta}<\overline{\boldsymbol{\theta}}$, an *optimality cut* $\boldsymbol{\theta}\geq(\mathbf{f}-\mathbf{AX})\bar{\boldsymbol{\lambda}}$ can be added, where $\bar{\boldsymbol{\lambda}}$ is the optimal dual solution. Following this intuition, the Benders decomposition starts with the relaxed problem (22) with no cuts, and then gradually solves the relaxed problem and the corresponding dual slave to progressively add cuts to (22). In any iteration, the value $\mathbf{dX}+\boldsymbol{\theta}$ obtained by solving (22) is a lower bound of the optimal value (since the problem is relaxed), while any $\mathbf{dX}+\overline{\boldsymbol{\theta}}$ where $\overline{\boldsymbol{\theta}}\leq\infty$ obtained by solving the dual slave is an upper bound (since it represents a feasible solution of the problem). The algorithm stops either when $\boldsymbol{\theta}\geq\overline{\boldsymbol{\theta}}$ which yields an optimal solution, or when (22) is infeasible which means the original problem is infeasible as well.

4.3 Nested Benders Decomposition

The original Benders decomposition works for two-stage problems. For our problem, we need the *nested Benders decomposition* [6]. As the name suggests, the algorithm stacks two layers of the original Benders decomposition: the *Phase-I algorithm* for SD–NPR, and the *Phase-II algorithm* for NPR–NE. As the middle layer, NPR is both the *slave* in Phase-I, and the *master* in Phase-II. In Phase-I, each iteration solves the relaxed SD denoted as O^{re}, and then solves the resulting $\mathcal{P}_t(X)$ for each t through the Phase-II; a feasibility cut as in (22c) or a set of optimality cuts as in (22b) is added to O^{re} at the end of a Phase-I iteration. In Phase-II, each iteration solves the relaxed NPR denoted as $\mathcal{P}_t^{\mathrm{re}}(X)$, and then solves the resulting $Q_{t,k}(Y,R)$ for each sample k; a feasibility cut or an optimality cut is added to $\mathcal{P}_t^{\mathrm{re}}(X)$ at the end of a Phase-II iteration. After Phase-II converges, the final $\mathcal{P}_t^{\mathrm{re}}(X)$ is equivalent to $\mathcal{P}_t(X)$, and hence the optimal dual solution of $\mathcal{P}_t^{\mathrm{re}}(X)$ is used to construct the Phase-I optimality cuts. If Phase-II is infeasible for some X, then the dual (which has an extreme ray) is used to construct the Phase-I feasibility cut. The process stops when Phase-I converges or is deemed infeasible.

Below we formally derive the Benders cuts, starting from Phase-II. Given fixed $\overline{X}$, let $(\overline{Y}_t,\overline{r}(t))$ be an arbitrary Phase-II master solution. Assume $Q_{t,k}(\overline{Y},\overline{R})$ is feasible for $\forall k$. Let $\overline{\boldsymbol{\pi}} = \{\overline{\pi}_{t,k}^{(5)}(p)\}\cup\{\overline{\pi}_{t,k}^{(6)}(a)\}\cup\{\overline{\pi}_{t,k}^{(14c)}\}$ be the optimal dual multipliers for Constraints (5), (6) and (14c) respectively. Define q as the auxiliary variable in $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$ as $\boldsymbol{\theta}$ in (22). The optimality cut to be added to $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$ is

$$q\geq\sum\nolimits_k \mathcal{L}_{t,k}^{\mathcal{P}}(\overline{\boldsymbol{\pi}},Y,R) \tag{23}$$

where

$$\mathcal{L}_{t,k}^{\mathcal{P}}(\boldsymbol{\pi},Y,R)=\sum_{a\in A}\delta_{t,a}^k\pi_{t,k}^{(6)}(a)-\sum_{p\in P}\pi_{t,k}^{(5)}(p)y(t,p)-\pi_{t,k}^{(14c)}r(t).$$

If for some k, $Q_{t,k}(\overline{Y},\overline{R})$ is infeasible. Let $\widetilde{\boldsymbol{\pi}}$ be an extreme ray of the dual of $Q_{t,k}(\overline{Y},\overline{R})$. The feasibility cut to be added to $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$ is

$$\mathcal{L}_{t,k}^{\mathcal{P}}(\widetilde{\boldsymbol{\pi}},Y,R)\leq 0. \tag{24}$$

Let dual solutions $\overline{\boldsymbol{\Pi}}$ and extreme rays $\widetilde{\boldsymbol{\Pi}}$ be involved in $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$ in a Phase-II iteration. Then $\mathcal{P}_t^{\mathrm{re}}(\overline{X})\triangleq$

$$\min_{Y_t\geq 0,r(t),q}\ \rho_2 r(t)+q/K$$
$$\text{s.t. (3)–(4); (23) } \forall\overline{\boldsymbol{\pi}}_i\in\overline{\boldsymbol{\Pi}};\ \text{(24) } \forall\widetilde{\boldsymbol{\pi}}_j\in\widetilde{\boldsymbol{\Pi}}. \tag{25}$$

Now, given $\overline{X}$, assume $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$ is feasible for $\forall t$. Let $\overline{\boldsymbol{\gamma}} = \{\overline{\gamma}_t^{(3)}(p)\}\cup\{\overline{\gamma}_t^{(4)}(l)\}\cup\{\overline{\gamma}_t^{(23)}(i)\}\cup\{\overline{\gamma}_t^{(24)}(j)\}$ be the optimal dual multipliers for (3), (4), (23) and (24) respectively. Define

$$\mathcal{L}_t^{O}(\boldsymbol{\gamma},X)=\sum\nolimits_{\overline{\boldsymbol{\pi}}_i\in\overline{\boldsymbol{\Pi}}}\gamma_t^{(23)}(i)\sum\nolimits_k\sum\nolimits_{a\in A}\delta_{t,a}^k\overline{\pi}_{t,k}^{(6)}(a)+$$
$$\sum\nolimits_{\widetilde{\boldsymbol{\pi}}_j\in\widetilde{\boldsymbol{\Pi}}}\gamma_t^{(24)}(j)\sum\nolimits_{a\in A}\delta_{t,a}^{k_j}\widetilde{\pi}_{t,k_j}^{(6)}(a)-$$
$$\left(\sum\nolimits_{p\in P}b_p^{\max}\gamma_t^{(3)}(p)x(h_p)+\sum\nolimits_{l\in L}b_l\gamma_t^{(4)}(l)\right),$$

where k_j is the infeasible sample corresponding to the j-th extreme ray added to $\mathcal{P}_t^{\mathrm{re}}(\overline{X})$. The optimality cuts to O^{re} (Phase-I) is

$$D\geq\mathcal{L}_t^{O}(\overline{\boldsymbol{\gamma}},X),\quad \forall t=1\ldots T. \tag{26}$$

Note in Phase-I, variable D serves exactly as the auxiliary variable θ in O^{re}. Instead of adding one optimality cut per iteration, T cuts are added in the form of (26). Later in Sec. 4.4, we show that this idea can be extended to Phase-II to expedite algorithm convergence. Now suppose for some t, $\mathcal{P}_t^{\text{re}}(\overline{X})$ is infeasible. Let $\widetilde{\boldsymbol{\gamma}}$ be an extreme ray of the dual of $\mathcal{P}_t^{\text{re}}(\overline{X})$. The feasibility cut to O^{re} is

$$\mathcal{L}_t^{\mathcal{O}}(\widetilde{\boldsymbol{\gamma}}, X) \le 0. \tag{27}$$

We finally address convergence. For Phase-II, let $Q_{t,k}^*$ be the optimal value of $Q_{t,k}(\overline{Y}, \overline{R})$, then an *upper bound* of $\mathcal{P}_t(\overline{X})$ is $\rho_2 \overline{r}(t) + \sum_k Q_{t,k}^*$ (since $\{Q_{t,k}^* \mid k\}$ represents a feasible solution to NE); let $\mathcal{P}_t^*$ be the optimal value of $\mathcal{P}_t^{\text{re}}(\overline{X})$, then $\mathcal{P}_t^*$ is a *lower bound* of $\mathcal{P}_t(\overline{X})$ (since it is the optimal value of the *relaxed* problem). Phase-II convergence thus happens when $\mathcal{P}_t^* \ge \rho_2 \overline{r}(t) + \sum_k Q_{t,k}^*$. Similarly for Phase-I, an *upper bound* of O is $\max_t \mathcal{P}_t^*$ at Phase-II convergence; a *lower bound* is the optimal value O^* of O^{re}. Phase-I convergence happens when $O^* \ge \max_t \mathcal{P}_t^*$. Note that for either phase, the lower bound is non-decreasing through iterations (since the master becomes "less" relaxed in each iteration), yet the upper bound may not be non-increasing (since different feasible solutions may be found with various values). The algorithm thus needs to track the best upper bound ever found in all iterations to verify convergence.

Algorithm 1 shows the full algorithm. Lines 2–30 show Phase-I algorithm for solving O, while Lines 7–24 show Phase-II algorithm for solving $\mathcal{P}_t(\overline{X})$ for each t. Note that for middle-layer $\mathcal{P}_t^{\text{re}}(\overline{X})$, the Phase-II cuts *sc* are shared across Phase-I iterations. This is because although a Phase-II cut generated in a previous Phase-I iteration may not be a valid cutting plane for the current $\mathcal{P}_t(\overline{X})$, it is a valid cut for the overall SD–NPR problem. Sharing cuts across Phase-I iterations can prune suboptimal X solutions without running the full Phase-II on them, and hence can speed-up convergence.

Algorithm 1 exploits the horizontal decomposition in both Phase-I and Phase-II, where $\mathcal{P}_t^{\text{re}}(\overline{X})$ and $Q_{t,k}(\overline{Y}, \overline{R})$ subproblems are solved independently for each t and/or k respectively. This is enabled by separating all *complicating variables and constraints* into the corresponding master problems, *i.e.*, variables (X, D) and constraints (1)–(2) for SD–NPR, and $(Y_t, r(t))$ and (3)–(4) for NPR–NE. Remember we separated variables R and W when defining $\mathcal{P}_t(X)$ and $Q_{t,k}(Y, R)$ to enable such a decomposition. With the fastest LP algorithm running in no less than cubic time [35], such decompositions can reduce the dependence of time complexity over T and K from cubic to linear for solving the subproblems. which is specifically important when the sample size K is large. This is our main motivation for applying Benders decomposition.

Since (14) is an MILP with finite master feasibility set and LPs as subproblems, we state Algorithm 1's convergence as below, whose proof is established in [3, 6] and hence omitted due to page limit.

Theorem 4.1. *Algorithm 1 converges to the optimal solution of a feasible REP instance or returns "Infeasible" for an infeasible instance in finite iterations.* □

4.4 Efficiency Enhancement Techniques

Though Algorithm 1 converges, it may have a slow convergence as a cutting plane method. Below, we describe three computational techniques to improve its efficiency. We start with a novel analytical dual solving technique for the NE subproblems. We then describe

Algorithm 1: Nested Benders Decomposition for REP

Input: Network G, pathbook P, cost budget C, samples $\{\mathcal{S}_t\}$, convergence tolerance ϵ.
Output: Optimal $(X, Y, \mathcal{Z}, D, R, W)$.

1 $\text{LB}^{\text{m}} \leftarrow -\infty$, $\text{UB}^{\text{m}} \leftarrow \infty$, $mc \leftarrow \emptyset$, $sc \leftarrow \emptyset$;
2 **repeat** // Phase-I iteration
3 Solve $\text{LB}^{\text{m}} = O^{\text{re}}$ with cuts mc for $(\overline{X}, \overline{D})$;
4 **if** O^{re} *infeasible* **then return** *Infeasible.*;
5 **for** $t = 1 \dots T$ **do**
6 $\text{LB}_t^{\text{s}} \leftarrow -\infty$, $\text{UB}_t^{\text{s}} \leftarrow \infty$;
7 **repeat** // Phase-II iteration
8 Solve $\text{LB}_t^{\text{s}} = \mathcal{P}_t^{\text{re}}(\overline{X})$ with cuts sc, and denote the solution as $(\overline{Y}_t, \overline{r}(t))$;
9 **if** $\mathcal{P}_t^{\text{re}}(\overline{X})$ *infeasible* **then**
10 Get extreme ray of dual $\mathcal{P}_t^{\text{re}}(\overline{X})$;
11 Add Phase-I feasibility cut to mc;
12 **break**;
13 **for** $k = 1 \dots K$ **do**
14 Solve $Q_{t,k}^* = Q_{t,k}(\overline{Y}, \overline{R})$ for $(\overline{\mathcal{Z}}, \overline{W})$;
15 **if** $Q_{t,k}(\overline{Y}, \overline{R})$ *feasible* **then**
16 Record optimal dual of $Q_{t,k}(\overline{Y}, \overline{R})$;
17 **else**
18 Get extreme ray of dual $Q_{t,k}(\overline{Y}, \overline{R})$;
19 Add Phase-II feasibility cut to sc;
20 **break**;
21 **if** $\forall k, Q_{t,k}(\overline{Y}, \overline{R})$ *is feasible* **then**
22 $\text{UB}_t^{\text{s}} = \min\{\text{UB}_t^{\text{s}}, \rho_2 \overline{r}(t) + \sum_k Q_{t,k}^*\}$;
23 Construct Phase-II optimality cut from $Q_{t,k}(\overline{Y}, \overline{R})$ duals, and add cut to sc;
24 **until** $UB_t^{\text{s}} - LB_t^{\text{s}} \le \epsilon$;
25 **if** $\mathcal{P}_t^{\text{re}}(\overline{X})$ *infeasible* **then break**;
26 **else** Record optimal dual of $\mathcal{P}_t^{\text{re}}(\overline{X})$;
27 **if** $\forall t, \mathcal{P}_t^{\text{re}}(\overline{X})$ *is feasible* **then**
28 $\text{UB}^{\text{m}} = \min\{\text{UB}^{\text{m}}, \max_t\{\text{LB}_t^{\text{s}}\}\}$;
29 Construct Phase-I optimality cut from recorded $\mathcal{P}_t^{\text{re}}(\overline{X})$ duals, and add cut to mc;
30 **until** $UB^{\text{m}} - LB^{\text{m}} \le \epsilon$;
31 **return** $\overline{X}$ *and best recorded subproblem solutions.*

two established techniques, one applying to the original Benders decomposition, and one applying to the nested algorithm.

Analytical Solving for Phase-II Slave Dual: This technique applies due to the structure of the NE subproblem, *i.e.*, finding optimal per-sample demand allocation across paths is easy. Given a sample k, for each $a \in A$, one can enumerate its paths towards all hosts in increasing delays, and then allocate demand to the shortest paths until all demand is allocated. This leads to a linear-time optimal algorithm for primal NE, assuming pre-sorted paths in all samples.

Yet, dual information is needed to add cut(s). In general, obtaining the dual of a primal solution is equivalent to solving a system of linear equations, which takes cubic time, not much faster than solving the dual LP directly. For our problem, though, we propose a *linear time analytical solution* (with pre-sorting) for dual NE.

Consider an instance of $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$. Let $\boldsymbol{\pi} = \{\pi^{(5)}_{t,k}(p)\}\cup\{\pi^{(6)}_{t,k}(a)\}\cup\{\pi^{(14c)}_{t,k}\}$ be the dual variables for Constraints (5), (6) and (14c) respectively. The dual of $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ is

$$\max_{\boldsymbol{\pi}\geq 0}\sum_{a\in A}\left(\delta^k_{t,a}\pi^{(6)}_{t,k}(a)-\sum_{p\in P_a}\overline{y}(t,p)\pi^{(5)}_{t,k}(p)\right)-\overline{r}(t)\pi^{(14c)}_{t,k} \tag{28a}$$

$$\text{s.t. } \pi^{(14c)}_{t,k}\leq\frac{\rho_2}{1-\alpha}; \tag{28b}$$

$$\pi^{(6)}_{t,k}(a)-\pi^{(5)}_{t,k}(p)\leq\frac{1}{\delta^k_t}d^k_{t,p}\left(\rho_1+\pi^{(14c)}_{t,k}\right), \quad \forall a\in A, p\in P_a. \tag{28c}$$

Now consider an AP $a\in A$. Let its path set P_a be sorted as $\mathbf{P}_a = (p^a_1, p^a_2,\ldots,p^a_{\mathbf{p}})$ in increasing order of path delays $d^k_{t,p}$. For simplicity we assume all path delays are distinct. Define the *critical path index* as $\mathbf{j}_a = \arg\min_j\{d^k_{t,p^a_j} \mid p^a_j\in\mathbf{P}_a, \sum_{i=1}^{j}\overline{y}(t,p^a_i)\geq\delta^k_{t,a}\}$ and $d^a_{\text{crit}} = d^k_{t,p^a_{\mathbf{j}a}}$, *i.e.*, the min-delay path p^a_j such that all paths with delay no greater than d^k_{t,p^a_j} can fully satisfy the demand $\delta^k_{t,a}$. Based on the aforementioned optimal primal solution, any path with $d^k_{t,p} < d^a_{\text{crit}}$ will have a saturated allocation $z(t,k,p)=\overline{y}(t,p)$, while all paths with $d^k_{t,p} > d^a_{\text{crit}}$ will have $z(t,k,p)=0$.

Further define $\overline{y}^a_{\text{sum}} = \sum_{j<\mathbf{j}_a}\overline{y}(t,p^a_j)$ and $\overline{y}^a_{\text{crit}} = \delta^k_{t,a}-\overline{y}^a_{\text{sum}}$. From the primal solution, the optimal average delay can be computed as $D^*_{t,k} = \frac{1}{\delta^k_t}\sum_a\left(\sum_{j<\mathbf{j}_a}d^k_{t,p^a_j}\overline{y}(t,p^a_j)+d^a_{\text{crit}}\overline{y}^a_{\text{crit}}\right)$. We then compute an optimal dual solution by the following:

$$\pi^{(14c)}_{t,k} = \begin{cases}\frac{\rho_2}{1-\alpha} & D^*_{t,k}\geq\overline{r}(t)\\ 0 & \text{otherwise}\end{cases}, \tag{29a}$$

$$\pi^{(6)}_{t,k}(a) = \frac{1}{\delta^k_t}\left(\rho_1+\pi^{(14c)}_{t,k}\right)d^a_{\text{crit}}, \tag{29b}$$

$$\pi^{(5)}_{t,k}(p) = \frac{1}{\delta^k_t}\left(\rho_1+\pi^{(14c)}_{t,k}\right)\left(d^a_{\text{crit}}-d^k_{t,p}\right)^+. \tag{29c}$$

To compute these values, the paths need to be sorted, which takes $O(\mathbf{p}\log\mathbf{p})$ time. The sorting only needs to be performed once for every sample. Therefore, with a $O(TKP\log P)$ pre-sorting, each dual of $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ can be analytically solved in linear time using Eqs. (29). We note that feasibility of $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ can be addressed during the computation: if $\sum_{p\in P_a}\overline{y}(t,p)\geq\delta^k_{t,a}$ for $\forall a\in A$, then $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ is feasible. Otherwise, (29) also presents a valid extreme ray when d^a_{crit} takes any value that is greater than the longest path delay with positive $\overline{y}(t,p)$ and $\pi^{(14c)}_{t,k}=\frac{\rho_2}{1-\alpha}$; the proof is similar to that of Theorem 4.2 and hence is omitted. An alternative for addressing infeasible $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ is to add a new constraint

$$\sum\nolimits_{p\in P_a}y(t,p)\geq\max_k\{\delta^k_{t,p}\},\quad\forall a\in A$$

to the Phase-II master $\mathcal{P}^{\text{re}}_t(X)$, and modify the cuts in (26) and (27). This ensures that $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$ is always feasible. The following theorem states the optimality of Eqs. (29) when the dual NE subproblem is feasible given $\overline{Y}$ and $\overline{R}$:

Theorem 4.2. *For a feasible instance of $\mathcal{Q}_{t,k}(\overline{Y},\overline{R})$, Eqs. (29) denote an optimal solution to Program (28).* □

Proof. Let us consider a specific $a\in A$ and its path set P_a. To start with, we fix $\pi^{(14c)}_{t,k}$ and define $\eta_{t,k}=\frac{1}{\delta^k_t}\left(\rho_1+\pi^{(14c)}_{t,k}\right)$. For a given $\pi^{(6)}_{t,k}(a)$, divide P_a into four sets:

- $P^1_a=\{p\mid\overline{y}(t,p)>0,\eta_{t,k}\cdot d^k_{t,p}\leq\pi^{(6)}_{t,k}(a)\}$,
- $P^2_a=\{p\mid\overline{y}(t,p)>0,\eta_{t,k}\cdot d^k_{t,p}>\pi^{(6)}_{t,k}(a)\}$,
- $P^3_a=\{p\mid\overline{y}(t,p)=0,\eta_{t,k}\cdot d^k_{t,p}\leq\pi^{(6)}_{t,k}(a)\}$, and
- $P^4_a=\{p\mid\overline{y}(t,p)=0,\eta_{t,k}\cdot d^k_{t,p}>\pi^{(6)}_{t,k}(a)\}$.

For $p\in P^2_a$, $\pi^{(5)}_{t,k}(p)=0$ by complementary slackness of Constraint (5). For $p\in P^3_a\cup P^4_a$, $\pi^{(5)}_{t,k}(p)$ takes arbitrary value no less than $\max\{\pi^{(6)}_{t,k}(a)-\eta_{t,k}d^k_{t,p},0\}$, which ensures (28c); note that none of these paths contribute to the objective value since $\overline{y}(t,p)=0$. For $p\in P^1_a$, we must have a strict equality:

$$\pi^{(5)}_{t,k}(p)=\pi^{(6)}_{t,k}(a)-\eta_{t,k}d^k_{t,p},\quad\forall p\in P^1_a. \tag{30}$$

If any such $\pi^{(5)}_{t,k}(p)$ is more, then we can solely decrease $\pi^{(5)}_{t,k}(p)$ to increase the objective value, contradicting optimality.

Taking (30) into the objective function (28a), we can re-write the first part of (28a) for each $a\in A$ as:

$$\mathcal{D}^k_t(a)=\delta^k_{t,a}\pi^{(6)}_{t,k}(a)-\sum_{p\in P_a}\overline{y}(t,p)\pi^{(5)}_{t,k}(p) \tag{31a}$$

$$=\delta^k_{t,a}\pi^{(6)}_{t,k}(a)-\sum_{p\in P^1_a}\overline{y}(t,p)\left(\pi^{(6)}_{t,k}(a)-\eta_{t,k}d^k_{t,p}\right) \tag{31b}$$

$$=\left(\delta^k_{t,a}-\sum_{p\in P^1_a}\overline{y}(t,p)\right)\pi^{(6)}_{t,k}(a)+\sum_{p\in P^1_a}\overline{y}(t,p)\eta_{t,k}d^k_{t,p}. \tag{31c}$$

In (31c), the right term is only related to the set P^1_a, while the left term is decided by both P^1_a and $\pi^{(6)}_{t,k}(a)$. Recall we define the *critical delay* as $d^a_{\text{crit}}=\min\{d\mid\sum_{p:d^k_{t,p}\leq d}\overline{y}(t,p)\geq\delta^k_{t,a}\}$ (equivalent to the definition by a critical path). If $\pi^{(6)}_{t,k}(a)>\eta_{t,k}d^a_{\text{crit}}$, by definition $\delta^k_{t,a}-\sum_{p\in P^1_a}\overline{y}(t,p)\leq 0$. Then we can reduce $\pi^{(6)}_{t,k}(a)$ until $\pi^{(6)}_{t,k}(a)=\eta_{t,k}d^a_{\text{crit}}$ to increase (or not to decrease if $\delta^k_{t,a}=\sum_{p\in P^1_a}\overline{y}(t,p)$) the value of (31c). If $\pi^{(6)}_{t,k}(a)<\eta_{t,k}d^a_{\text{crit}}$, by definition $\delta^k_{t,a}-\sum_{p\in P^1_a}\overline{y}(t,p)>0$. Then we can increase $\pi^{(6)}_{t,k}(a)$ until $\pi^{(6)}_{t,k}(a)=\eta_{t,k}d^a_{\text{crit}}$ to again increase (31c). The only flexibility $\pi^{(6)}_{t,k}(a)$ can have is when $\delta^k_{t,a}=\sum_{p:d^k_{t,p}\leq d^a_{\text{crit}}}\overline{y}(t,p)$, in which case $\pi^{(6)}_{t,k}(a)$ can take any value between $\eta_{t,k}d^a_{\text{crit}}$ and $\eta_{t,k}d^k_{t,p_{\text{n}}}$ where p_{n} is the next path after the critical path that has a positive $\overline{y}(t,p)$.

The above has addressed the optimal values of $\pi^{(5)}_{t,k}(p)$ and $\pi^{(6)}_{t,k}(a)$ as in (29b) and (29c). Now if we take these values and $\eta_{t,k}$ into (31c), then $\sum_{a\in A}\mathcal{D}^k_t(a)=D^*_{t,k}\cdot(\rho_1+\pi^{(14c)}_{t,k})$ where $D^*_{t,k}$ is the optimal sample average delay. The full dual objective can be re-written as

$$\mathcal{D}^k_t=\sum\nolimits_{a\in A}\mathcal{D}^k_t(a)-\overline{r}(t)\pi^{(14c)}_{t,k} \tag{32a}$$

$$=\rho_1\cdot D^*_{t,k}+(D^*_{t,k}-\overline{r}(t))\pi^{(14c)}_{t,k}. \tag{32b}$$

It is now clear that if $D^*_{t,k}-\overline{r}(t)\geq 0$, then $\pi^{(14c)}_{t,k}$ can take the maximum value $\frac{\rho_2}{1-\alpha}$ constrained by (28b); otherwise $\pi^{(14c)}_{t,k}=0$. □

Multiple Cuts: In each Phase-II iteration in Algorithm 1, at most one optimality cut (in the form of (22b)) is added. Birge and Louveaux [7] first noticed that by dividing a single optimality cut into multiple cuts, one for each independent subproblem for example, the convergence speed can be improved. Intuitively, this is because multiple cuts can prune more sub-optimal region of the master solution space in each iteration, hence the number of iterations is reduced. We can apply this technique to our algorithm. Instead of having one auxiliary variable q in Eq. (23), K variables are defined as $\{q(k)\}$. The optimality cuts to replace Eq. (23) are

$$q(k) \geq \mathcal{L}^{\mathcal{P}}_{t,k}(\overline{\boldsymbol{\pi}}, Y, R), \quad \forall k, \tag{33}$$

and the objective function of $\mathcal{P}^{\text{re}}_t(\overline{X})$ in (25) is updated as

$$\min\ \rho_2 r(t) + \frac{1}{K}\sum\nolimits_k q(k). \tag{34}$$

The Phase-I algorithm has already employed a multi-cut formulation as in (26). We highlight that this technique does not change the execution in Algorithm 1: all subproblems $\mathcal{Q}_{t,k}(\overline{Y}, \overline{R})$ must still be solved before the cuts in (33) can be added, in order to avoid adding optimality cuts for an infeasible $(\overline{Y}, \overline{R})$. This is specifically important when implementing the algorithm distributedly. We show in Sec. 5 that this can achieve significant speed-up of the algorithm.

Fast Forward Fast Backward (FFFB): In Algorithm 1, each Phase-I iteration consists of a full Phase-II convergence: only dual of the optimal Phase-II master solution is passed onto Phase-I. This is the *Fast Forward (FF)* approach proposed by Birge [5]. Another choice is *Fast Backward (FB)*: its main loop consists of the Phase-II algorithm, where each Phase-II iteration consists of a full convergence of Phase-I [31]. However, neither approach achieves the best practical performance. A balanced approach, called *Fast Forward Fast Backward* (FFFB), is shown to be the most efficient in practice [6]. In each main iteration, the algorithm performs a full forward sweep (solving Phase-I master, Phase-II master and Phase-II slave), followed by a full backward sweep (adding Phase-II cut(s), solving Phase-II master *once*, and then adding Phase-I cut(s) based on it); if in the forward sweep the Phase-II master is infeasible, then such information is directly passed to the Phase-I master. FFFB speeds-up convergence similarly as sharing Phase-II cuts: cuts based on non-optimal (but feasible) NPR help eliminate suboptimal Phase-I master solutions early, which reduces the total number of Phase-I master solutions visited. We show its effectiveness also in Sec. 5.

4.5 Discussions

Complexity. Algorithm 1 converges to optimal (Theorem 4.1) but has exponential worst-case complexity, and techniques in Sec. 4.4 do not reduce it to polynomial. However, the techniques (especially dual analytical solving) present two main advantages in practice. First, we found that practically NE solving is the efficiency bottleneck due to the large number of samples needed for accurate sampling. Our technique speeds-up worst-case per-iteration NE solving by at least two orders of magnitudes, greatly improving efficiency without sacrificing solution quality. Second, iterative algorithms are commonly implemented with a time bound in practice. Faster per-iteration solving greatly increases the solution space searched in bounded time, thus improving solution quality.

Distributed computation. Algorithm 1 can be distributed among ECM, NM and ESP. Each party can solve its own subproblem (SD for ECM, NPR for NM, and NE for ESP itself), and exchange solution information. Distributed computation may not only reduce per-party computational overhead, but also preserves privacy of certain parties, *e.g.*, the computing resource distribution of the ECM, and the network topology of the NM. We will explore distributed computation and its implications in our future work.

5 PERFORMANCE EVALUATION

5.1 Evaluation Method

We synthesized real-world demand traces with randomly generated data for evaluation. For demands, we used the NYC 2018 Yellow Taxi Trip Data [27], which contains around 112 million taxi trips including drop-off times and taxi zones. We divided the year into days as periods, and each day was divided into 2–24 (4 by default) time slots. We regarded each taxi zone as a single AP, and took the average number of passengers dropped-off in each slot as a demand sample at the AP, where each passenger represented 1 Mbps of fixed demand. We picked the top 5 (*small*) or 20 (*medium*) most popular drop-off zones over the entire 262 zones depending on the desired problem size, which accounted for 18% or 55% of all demands respectively. We used the first 100 days as our *training samples*, *i.e.*, samples used to solve the optimization problem to derive SD and NPR. We then used the next 265 days as our *test samples*, *i.e.*, samples used to evaluate our solution once it is employed.

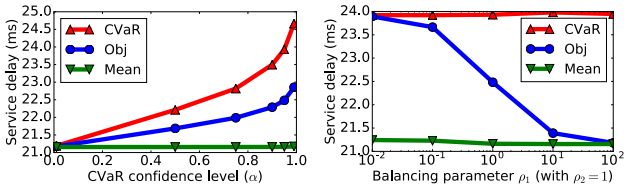
We synthesized the other inputs of the problem. First, we built random topologies among APs using the Watts-Strogatz model [30] with $k = 4$ and $p = 0.3$. For the small-sized problem with 5 APs, we added 20 artificial network nodes to construct the topology. We then randomly picked 5 nodes, each connected to an edge computing node through an infinite-capacity link. Deployment costs were generated in a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 1000$ and $\sigma = 200$, while the ESP's cost budget was 3300. For the pathbook, we ran the k-shortest path algorithm to generate 3 min-hop paths for each AP-host pair. Such a number has been reported in [16] to be able to admit over 90% of the maximum flow traffic in a typical network. We simulated two scenarios: a *normal* scenario where link capacities (except for links to edge nodes) were fixed at 5 Gbps and delays were generated in $\mathcal{N}(10, 4)$ ms, and a *congested* scenario where capacities were downgraded to 2 Gbps and half edge nodes randomly selected in each time slot had 50× prolonged connection delays (simulating about half packets experiencing 1s retransmission timeouts due to host congestion).

By default, we set $\rho_1 = \rho_2 = 0.5$ and $\alpha = 0.95$. For convergence, we set $\epsilon = 10^{-3}$. To reduce random noise, we averaged results over 10 independent runs on inputs generated with different seeds. We implemented all algorithms in Python, where LPs/MILPs were solved using MOSEK [2]. Running time-related simulations were run on a Windows PC with Hex-Core 3.2GHz CPU and 16GB memory.

5.2 Experiment Results

5.2.1 Mean vs. CVaR. Fig. 2 shows the trade-off between expectation and CVaR with α and (ρ_1, ρ_2) for the training set. We conducted simulations on the *medium* problem size and with *normal* load. In Fig. 2(a), we varied α from 0.01 to 0.99. With higher confidence α, the CVaR that measures the worst-case delay beyond such a confidence is higher. When α is close to 0, the CVaR converges

to the expectation; when α is close to 1, the CVaR approaches the worst-case performance observed among all samples. However, optimizing the solution with a higher α has little impact on the expected value. In Fig. 2(b), by changing ρ_1 (with fixed ρ_2), we can optimize different combinations of expectation and CVaR. A higher ρ_1 results in a lower expected value but a higher CVaR. In practice, ESP can fine-tune these parameters to find its best trade-off.



(a) Service delay with varying α **(b) Service delay with varying ρ_1**

Figure 2: Trade-off between sample mean and CVaR

5.2.2 Time-Varying vs. Time-Agnostic. Fig. 3 compares two modeling choices: time-varying (**T-Var**) and time-agnostic (**T-Ago**). Both models were solved using Algorithm 1. In **T-Ago**, all samples were flatten out across time slots, and the problem was solved as if there was just one time slot. We ran our simulations in the *small* setting with *congested* load, and varied the number of time slots from 2 to 24. Remember we used average demands in a time slot as a sample. With more slots, the input includes more samples (representing more dynamics), and hence both models lead to higher CVaR and thus higher objective value. **T-Var** has a growing performance advantage over **T-Ago**, because of the former's ability to adjust to different demand and network patterns at different times and perform finer-grained optimization with more slots.

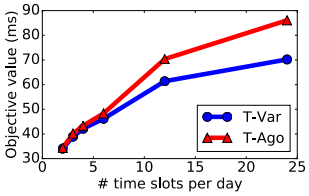
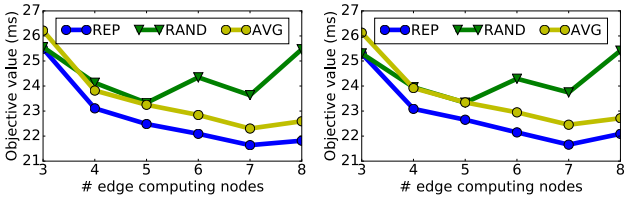


Figure 3: Objective with time-varying or fixed provisioning

5.2.3 Optimal vs. Heuristic. Fig. 4 shows the training and testing objective values obtained with different host numbers, where we compared our algorithm to two heuristics in terms of the quality of solution (objective value). Here, **REP** refers to our algorithm; **RAND** refers to a random host selection heuristic with NPR–NE solved using our Phase-II; **AVG** is a heuristic that simply optimizes the average delay for each time slot when delays and demands are simply averaged over all samples, which is solved in one integrated MILP. We ran our simulations in the *medium* setting with *normal* load. With more hosts, the objectives of **REP** and **AVG** are generally decreasing. **RAND**, however, has fluctuating objective values due to random deployment. Among the algorithms, **REP** naturally achieves the lowest objective value due to its optimality, though **AVG** has performance not significantly worse than ours. However, we emphasize that our advantageous solution will constantly impact service performance in the long run after deployment, based on the test sample results in Fig. 4(b). Hence it is worthwhile to improve performance using our algorithm in initial provisioning.



(a) Training objective value **(b) Testing objective value**

Figure 4: Comparison of our algorithm with heuristics

5.2.4 Efficiency Enhancement. Finally, we evaluate the efficiency enhancement techniques in Fig. 5. We compared **Full** (our algorithm with all three techniques), to **!Multi** (**Full** without multi-cut), **!FFFB** (**Full** without FFFB), and **!Ana** (**Full** without the analytical dual solution and solves Phase-II slave duals through direct LP solving). We ran our simulations in the *small* setting with *normal* load. As we can see, each technique achieves multifold improvement of convergence speed. Since these techniques are orthogonal to each other, the overall speed-up is in two orders of magnitude, which demonstrates the effectiveness of these techniques.

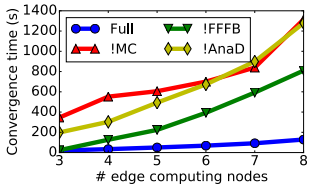


Figure 5: Convergence with/without efficiency techniques

6 RELATED WORK

6.1 Resource Allocation in Edge Computing

There has been two categories of efforts on edge resource allocation. One focuses on allocating non-network resources, stemming from existing research on distributed computing. Most such efforts consider fixed network allocations and statistics (*e.g.*, delay). Examples include task scheduling [8, 26, 38], task offloading / service placement [10, 28, 29, 32], resource allocation [9], caching [34], etc. The other recognizes criticality of the network in service performance, and employs active network planning to best utilize network information and/or allocate network resource. For instance, Gao *et al.* [12] explored how access networks selection can improve service QoS. Josilo *et al.* [13] and Liu *et al.* [19, 20] studied multi-dimensional resource allocation including both computing and network resources. Yu *et al.* [37] first identified the importance and challenge of complex edge network topologies, and proposed approximation schemes for QoS-guaranteed joint service deployment and network provisioning for real-time applications. Besides, Poularakis *et al.* [23], Ouyang *et al.* [22] and Meng *et al.* [21] also explored joint service placement and routing with storage, migration cost, and task deadline considerations, respectively.

This paper uses a similar service model as [37], where a service processes continuous data streams. However, this paper considers more practical factors, including deployment costs, time-varying and uncertain demands and network condition, robustness, etc. We note that robustness in edge resource allocation has yet been well explored, and has only been studied in some very recent work [18].

6.2 Other Related Work

Besides edge resource allocation, other related problems include virtual network or infrastructure embedding (VNE/VIE) [11] and service function chaining (SFC) [4]. Few efforts on these problems have explored the time-varying demands and network condition. Exceptions include Xie *et al.* [33] for VIE and Li *et al.* [17] for SFC. Xie *et al.* [33] extracted time-varying characteristics of peak application demands via profiling, and then made bandwidth allocation accordingly. Li *et al.* [17] used a time-varying model to compute correlations between service functions for joint deployment. Both regarded the time-varying demands as deterministic or estimatable, and did not consider robustness when facing uncertain dynamics.

Conventionally, CVaR is a risk measure widely used in economics and finance, notably in portfolio optimization [24]. However, it has recently found use in computing and networking for robust optimization as well. For example, Rullo *et al.* [25] and Yu *et al.* [36] both used CVaR to represent and optimize security risks in IoT.

7 CONCLUSIONS

In this paper, we studied robust edge resource provisioning with time-varying demands and network conditions. For an edge service, the service provider employs a three-stage approach, including service deployment, network provisioning, and performance estimation. We proposed a time-slotted stochastic model to capture the temporal-spatial demand distributions observed in real-world datasets. We introduced a widely used risk model in economics, Conditional Value-at-Risk, which enables trade-off between expected performance and robustness. Provisioning was formulated as a three-stage sample-based stochastic optimization problem. Based on the problem structure, we proposed a nested Benders decomposition approach to optimally solve the problem, and described several efficiency enhancement techniques to drastically speed-up convergence. We presented results from real dataset-based simulations, which demonstrated the advantages of the time-varying and robustness models, the proposed algorithm compared to heuristics, and the efficiency enhancement techniques, respectively.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants 1704092, 1704662, 1717197, and 1717315, and JSPS KAKENHI Grant No. JP20H00592. The information reported here does not reflect the position or the policy of the funding agency.

REFERENCES

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms and Applictions.*
[2] MOSEK ApS. 2019. MOSEK Optimizer API for Python Release 9.0.98. https://docs.mosek.com/9.0/pythonfusion/index.html
[3] Jacques F Benders. 1962. Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numer. Math.* 4, 1 (dec 1962), 238–252.
[4] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. 2016. A Survey on Service Function Chaining. *J. Netw. Comput. Appl.* 75 (nov 2016), 138–155.
[5] John R. Birge. 1985. Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *Oper. Res.* 33, 5 (oct 1985), 989–1007.
[6] John R. Birge and François Louveaux. 2011. *Introduction to Stochastic Programming.*
[7] John R. Birge and François V. Louveaux. 1988. A Multicut Algorithm for Two-Stage Stochastic Linear Programs. *Eur. J. Oper. Res.* 34, 3 (mar 1988), 384–392.
[8] Luiz F Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F Rana, and Manish Parashar. 2017. Mobility-Aware Application Scheduling in Fog Computing. *IEEE Cloud Comput.* 4, 2 (mar 2017), 26–35.
[9] Gabriele Castellano, Flavio Esposito, and Fulvio Risso. 2019. A Distributed Orchestration Algorithm for Edge Computing Resources with Guarantees. In *Proc. IEEE INFOCOM.* 2548–2556.
[10] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom Hao Luan, and Hao Liang. 2016. Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption. *IEEE Internet Things J.* 3, 6 (2016), 1171–1181.
[11] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. 2013. Virtual Network Embedding: A Survey. *IEEE Commun. Surv. Tutorials* 15, 4 (jan 2013), 1888–1906.
[12] Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu. 2019. Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing. In *Proc. IEEE INFOCOM.* 1459–1467.
[13] Sladana Josilo and Gyorgy Dan. 2019. Wireless and Computing Resource Allocation for Selfish Computation Offloading in Edge Computing. In *Proc. IEEE INFOCOM.* 2467–2475.
[14] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. 2002. The Sample Average Approximation Method for Stochastic Discrete Optimization. *SIAM J. Optim.* 12, 2 (jan 2002), 479–502.
[15] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. [n. d.]. CRAWDAD Dataset Dartmouth/Campus (V. 2009-09-09). https://crawdad.org/dartmouth/campus/20090909
[16] Mathieu Leconte, Apostolos Destounis, and Georgios Paschos. 2018. Traffic Engineering with Precomputed Pathbooks. In *Proc. IEEE INFOCOM.* 234–242.
[17] Defang Li, Peilin Hong, Kaiping Xue, and Jianing Pei. 2018. Virtual Network Function Placement Considering Resource Optimization and SFC Requests in Cloud Datacenter. *IEEE Trans. Parallel Distrib. Syst.* 29, 7 (jul 2018), 1664–1677.
[18] Haijun Liao, Zhenyu Zhou, Shahid Mumtaz, and Jonathan Rodriguez. 2019. Robust Task Offloading for IoT Fog Computing Under Information Asymmetry and Information Uncertainty. In *Proc. IEEE ICC.* 1–6.
[19] Qiang Liu and Tao Han. 2019. DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing. In *Proc. ACM MobiHoc.*
[20] Qiang Liu and Tao Han. 2019. VirtualEdge: Multi-Domain Resource Orchestration and Virtualization in Cellular Edge Computing. In *Proc. IEEE ICDCS.*
[21] Jiaying Meng, Haisheng Tan, Chao Xu, Wanli Cao, Liuyan Liu, and Bojie Li. 2019. Dedas: Online Task Dispatching and Scheduling with Bandwidth Constraint in Edge Computing. In *Proc. IEEE INFOCOM.* 2287–2295.
[22] Tao Ouyang, Rui Li, Xu Chen, Zhi Zhou, and Xin Tang. 2019. Adaptive User-managed Service Placement for Mobile Edge Computing: An Online Learning Approach. In *Proc. IEEE INFOCOM.* 1468–1476.
[23] Konstantinos Poularakis, Jaime Llorca, Antonia M. Tulino, Ian Taylor, and Leandros Tassiulas. 2019. Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks. In *Proc. IEEE INFOCOM.* 10–18.
[24] R. Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of Conditional Value-at-Risk. *J. Risk* 2 (2000), 21–41.
[25] Antonino Rullo, Edoardo Serra, Elisa Bertino, and Jorge Lobo. 2017. Shortfall-Based Optimal Placement of Security Resources for Mobile IoT Scenarios. In *Proc. ESORICS.* 419–436.
[26] Haisheng Tan, Zhenhua Han, Xiang-Yang Li, and Francis C M Lau. 2017. Online Job Dispatching and Scheduling in Edge-Clouds. In *Proc. IEEE INFOCOM.* 1–9.
[27] Taxi and Limousine Commission (TLC). 2019. 2018 Yellow Taxi Trip Data. https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq
[28] Liang Tong, Yong Li, and Wei Gao. 2016. A Hierarchical Edge Cloud Architecture for Mobile Computing. In *Proc. IEEE INFOCOM.* 1–9.
[29] Lin Wang, Lei Jiao, Ting He, Jun Li, and Max Muhlhauser. 2018. Service Entity Placement for Social Virtual Reality Applications in Edge Computing. In *Proc. IEEE INFOCOM.* 468–476.
[30] Duncan J. Watts and Steven H. Strogatz. 1998. Collective Dynamics of 'Small-World' Networks. *Nature* 393, 6684 (jun 1998), 440–442.
[31] Christian Wolf. 2013. *Advanced Acceleration Techniques for Nested Benders Decomposition in Stochastic Programming.* Ph.D. Dissertation. University of Paderborn.
[32] Yong Xiao and Marwan Krunz. 2017. QoE and Power Efficiency Tradeoff for Fog Computing Networks with Fog Node Cooperation. In *Proc. IEEE INFOCOM.* 1–9.
[33] Di Xie, Ning Ding, Y Charlie Hu, and Ramana Kompella. 2012. The Only Constant Is Change: Incorporating Time-Varying Network Reservations in Data Centers. In *Proc. ACM SIGCOMM.* 199–210.
[34] Jie Xu, Lixing Chen, and Pan Zhou. 2018. Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks. In *Proc. IEEE INFOCOM.* 207–215.
[35] Yinyu Ye. 1991. An O(n3L) Potential Reduction Algorithm for Linear Programming. *Math. Program.* 50, 1-3 (mar 1991), 239–258.
[36] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, and Xiang Zhang. 2018. Deploying Robust Security in Internet of Things. In *Proc. IEEE CNS.* 1–9.
[37] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. 2018. Application Provisioning in Fog Computing-enabled Internet-of-Things: A Network Perspective. In *Proc. IEEE INFOCOM.* 783–791.
[38] Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. 2016. Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System. *IEEE Trans. Comput.* 65, 12 (dec 2016), 3702–3712.