Spring 2023

Resource Coordination Learning for End-to-end Network Slicing Under Limited State Visibility

Xiang Liu
San Jose State University

Resource Coordination Learning for End-to-end Network Slicing Under Limited State Visibility

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Xiang Liu

May 2023

The Designated Project Committee Approves the Project Titled

Resource Coordination Learning for End-to-end Network Slicing Under Limited State Visibility

by

Xiang Liu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2023

Dr. Genya Ishigaki	Department of Computer Science
Dr. Mike Wu	Department of Computer Science
Dr. Navrarti Saxena	Department of Computer Science

ABSTRACT

Resource Coordination Learning for End-to-end Network Slicing Under Limited State Visibility

by Xiang Liu

This paper discusses a resource coordination problem under limited state visibility to realize end-to-end network slices that are hosted by multiple network domains. We formulate this resource coordination problem as a special type of the multi-armed bandit (MAB) problem called the combinatorial multi-armed bandit (CMAB) problem. Based on this formulation, we convert the problem to a regret minimization problem with a linear objective function and solve it by adapting the Learning with Linear Rewards (LLR) algorithm. In this paper, we present a new hybrid approach that incorporates state reports, which include partial resource information in each domain, into the existing LLR algorithm. Our experiment results show that the proposed algorithm performs significantly better than other baseline learning algorithms. Furthermore, the proposed algorithm outperforms the LLR algorithm especially when the traffic patterns of network slices are more dynamic and unstable.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my research supervisor Professor Ishigaki for his unwavering support and guidance throughout my master's program. I would also like to thank Professor Wu and Professor Saxena for serving on my thesis committee and providing valuable feedback and suggestions. Finally, I am deeply thankful to my family. As a first-generation student, I really appreciate what they have done for me.

TABLE OF CONTENTS

CHAPTER

1 Introduction 1

1.1 5G Network Slicing 1

1.2 Project Statement 2

2 Background 4

2.1 Multi-Armed Bandit Problem 4

2.1.1 Regret 4

2.1.2 Action-value Function 5

2.1.3 Stochastic Multi-Armed Bandit Problem 5

2.1.4 Epsilon-Greedy Approach 6

2.1.5 Upper Confidence Bound 7

2.2 Super Arm 8

2.3 Related Work 9

3 An Global Resource Coordination Architecture 11

3.1 Network Domains and Slice Coordination 11

3.2 Slice Request 13

3.3 Domain Report 13

4 Problem Formulation 15

5 Combinatorial Multi-Armed Bandit with Augmented Semi-Feedback 17

5.1 Combinatorial Multi-Armed Bandit 17

5.2 Learning with Linear Rewards (LLR) in the Domain-level Service Stitching . . . 18

5.3 Our Hybrid Approach . . . 20

6 Experiment . . . 21

6.1 Network, Control Domains and E2E Slice Requests . . . 21

6.1.1 ATT North America . . . 21

6.1.2 Synthesized Topology . . . 21

6.1.3 Domain Assignment . . . 22

6.1.4 E2E Slice Request Generation . . . 24

6.2 Periodic Domain Reports and Traffic Observation . . . 24

6.3 Baseline Comparison . . . 26

6.4 Results . . . 27

7 Conclusion . . . 31

LIST OF REFERENCES . . . 32

APPENDIX

CHAPTER 1

Introduction

1.1 5G Network Slicing

As a step up from the 4G system, it is anticipated that the 5G system will support a variety of new use cases. One of the key enabling technologies that will facilitate the deployment of these applications is network slicing. Network slicing, in essence, allows for the creation of multiple virtual networks on a shared infrastructure, thus providing separate and isolated environments for different types of traffic or applications [1]. Through network slicing, each virtual network can be tailored to a specific use case, such as low-latency communication for critical applications or high-speed internet for consumers. As shown in Figure 1, three different types of customized network slices are built on the same physical infrastructure. Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), which collectively provide flexible control and management of communication and computing resources, are the enablers of network slicing. SDN refers to the network architecture capable of separating the control plane from the data plane and providing programmability for the development of network application [2]. This separation allows for centralized network management and control, which results in more efficient network operations and better network performance. SDN has gained popularity in recent years due to its ability to simplify network management, reduce costs, and improve network security. NFV is a technology that allows network functions to be extracted from the underlying hardware by utilizing virtualization technologies and off-the-shelf hardware [3]. NFV can reduce both capital and operational expenses and offer horizontal scalability due to the fact that only inexpensive commodity hardware is required.

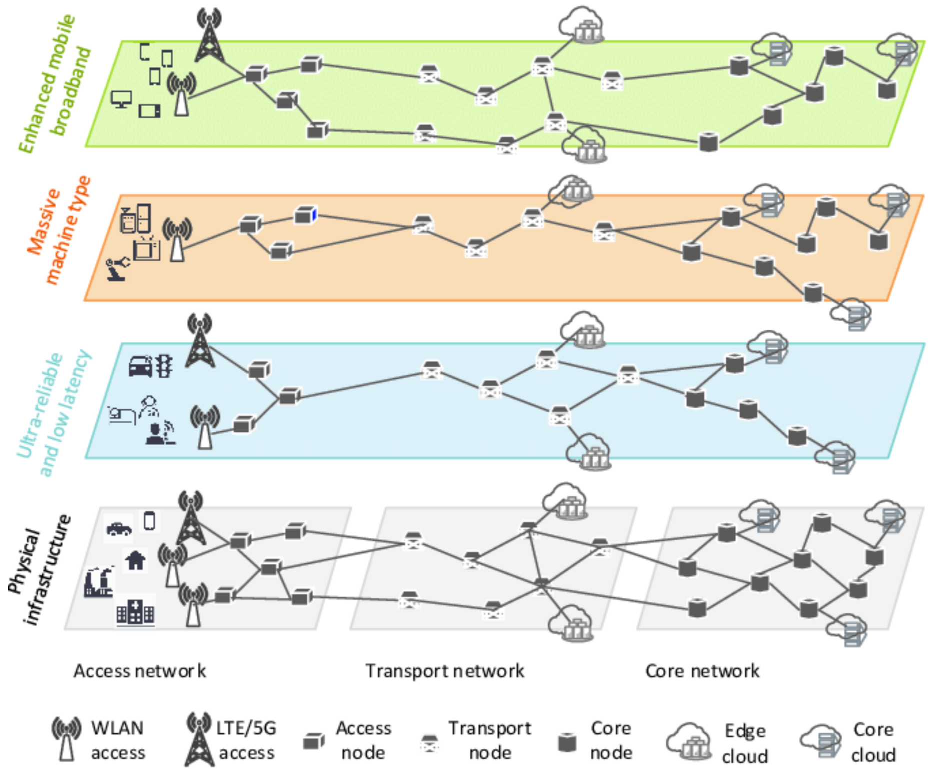


Figure 1: 5G Network Slicing Architecture based on [4].

1.2 Project Statement

In this research, we are particularly interested in end-to-end (E2E) network slicing that spans multiple administrative domains. Even though resource allocation problems for network slicing in single domains have been widely discussed in recent literature [5, 6, 7], the coordinated resource reservation among multiple network domains to compose E2E network slices still poses many challenges [8].

Based on the generic architecture of E2E resource coordination in the next-generation networks [9], we propose a resource reservation function of a centralized global slice coordinator that communicates with the domain orchestrators and stitches a set of domain-level services to compose E2E network slices. A major technical challenge in this global resource coordination is the limited visibility of domain-level topology and the associated performance. Due to its administrative independence, each resource orchestrator, which would be a composite of an SDN controller and NFV MANO [10], is assumed to only reveal limited information about its domain states to the global coordinator. The selective information revealing enhances security and scalability by limiting the information passed outside of each domain. However, at the same time, it makes the global resource coordination task more complex with the latent states of underlying domain infrastructure, especially with stringent Service-Level Agreement (SLA) of E2E network slices [11].

Hence, this paper discusses a resource allocation problem of a global slice coordinator for E2E network slicing. The problem assumes limited domain visibility where the coordinator has access to only a list of aggregated domain-level services and their expected performance metrics such as delay. The allocation problem is solved by a multi-armed bandit (MAB) algorithm that learns the performance of domain-level services by combining *semi-bandit* feedback from the actual performance measurement and the expected service performance reported by domain orchestrators. Our evaluation experiment demonstrates that the proposed approach outperforms other baseline MAB algorithms in terms of *regret*.

CHAPTER 2

Background

2.1 Multi-Armed Bandit Problem

The multi-armed bandit problem is a classic decision-making problem that is first initiated by Lai and Robbins [12]. It involves a situation in which a gambler must choose between multiple slot machines with varying payout rates. The goal is to maximize the total payoff over time. To maximize reward, the agent must strike a balance between exploration (attempting new arms to learn about their rewards) and exploitation (selecting the arm with the highest expected reward based on current knowledge). Several algorithms, including the ϵ-greedy algorithm, the Upper Confidence Bound (UCB) algorithm, and Thompson Sampling, have been developed to address the multi-armed bandit problem. This chapter will primarily focus on the first two algorithms which are employed in our experiment. Then we will explain the concept of a super arm and discuss related works. To begin with, we will discuss regret and the action-value function.

2.1.1 Regret

In reinforcement learning, particularly in the multi-armed bandit problem, regret is used to describe the difference between the expected cumulative reward of the optimal policy and the actual cumulative reward obtained by the agent's policy. As previously stated, our goal is to maximize the overall reward. In other words, we want to minimize regret. As a result, regret is a crucial metric for evaluating reinforcement learning algorithms and is often used as a benchmark for comparisons between different methods. In our research, we use the concept of cumulative regret which sums up regret over a fixed time horizon (see Figure 2).

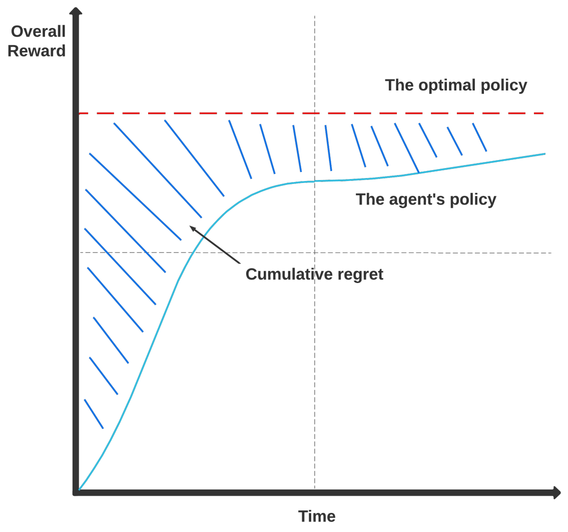


Figure 2: Cumulative Regret.

2.1.2 Action-value Function

An action-value function (also known as Q-function) is a function that measures the expected return of a action in a given state. More specifically, it informs us whether or not we should take a particular action in a given state. The action-value function is mostly used in a Markov decision process. It is also utilized in the context of the MAB problem, as the stochastic MAB problem can be interpreted as a Markov decision process without the state S. We will explain more about the stochastic MAB problem in the next section. Mathematically, the action-value function can be expressed as:

$$Q(a) = E\left[R_t \mid A_t = a\right], \tag{1}$$

where R_t is the reward received at time step t after taking action a.

2.1.3 Stochastic Multi-Armed Bandit Problem

Based on the characteristics of the reward generation process, the bandit problem can be further divided into three categories: stochastic, adversarial, and Markovian [13]. Our problem formulation belongs to the stochastic MAB problem. Here are the basic definitions of the stochastic MAB problem. There are K independent arms. Each arm $i = 1, ..., K$ corresponds to an unknown probability distribution for the reward. At each time step t, we choose an arm I_t and observe the reward $X_{I_t,t}$ based on its probability distribution. Thus the pseudo-regret after n plays can be defined as:

$$R_n = \max_{i=1,...,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}. \tag{2}$$

As stated previously, our objective is to minimize this regret.

2.1.4 Epsilon-Greedy Approach

The ϵ-greedy approach is a simple algorithm for balancing exploration and exploitation in the multi-armed bandit problem. At each time step t, the agent chooses a random action with probability ϵ (the exploration rate) and chooses the action with highest estimated reward $Q(a)$ with probability (1 - ϵ) (the exploitation rate).

Notably, ϵ is typically set to a small constant (e.g., 0.1) to encourage exploration in the early stages of learning. It then gradually decreases over time as the agent gains more experience and converges on more accurate action-value function estimates. This technique is called decayed ϵ-greedy which we have adopted in our implementation. Specifically we used a decay function for ϵ which is shown in Figure 3. This decay function combined with the ϵ-greedy algorithm will enforce the agent to explore in the beginning phase and smoothly transition to exploiting in later episodes. The ϵ-greedy approach provides a simple and effective solution for the MAB problem and can be easily implemented in practice. However, it has some major drawbacks. For example, we may still choose those arms that we have confirmed are bad for exploration, thus not maximizing our potential overall reward.

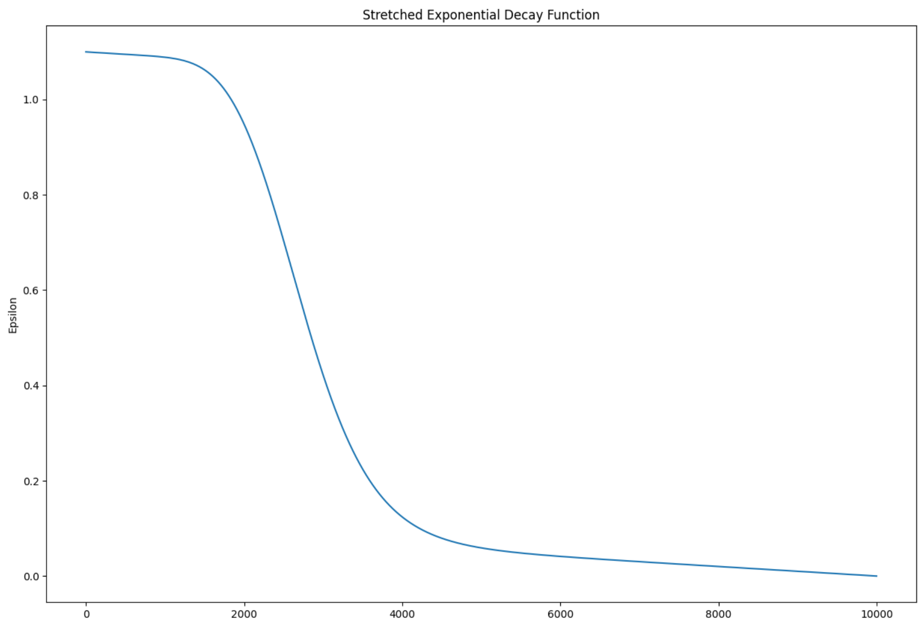


Figure 3: The Decay Function for ϵ.

2.1.5 Upper Confidence Bound

The UCB algorithm proposed by Agrawal [14] is another popular technique for solving the multi-armed bandit problem based on the principle of *optimism in face of uncertainty*. It does not assume any prior knowledge on the reward distribution and can avoid the drawbacks of the ϵ-greedy approach. The basic idea behind the UCB algorithm is to estimate the potential reward of each option and choose the one with the highest expected reward while also taking into account for the uncertainty in the estimates. Furthermore, the UCB algorithm balances exploration and exploitation by choosing actions with higher uncertainty (as reflected by the upper confidence bound) more frequently early on, and gradually shifting towards exploiting actions with higher estimated values as more data is collected. The formula for UCB can be represented as:

$$A_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]. \tag{3}$$

- $Q_t(a)$ represents the estimated value of action a at current timestep t.
- $N_t(a)$ represents the number of times that action a has been selected prior to current timestep.
- c is a constant that controls the confidence interval.

The action-value function $Q_t(a)$ is the exploitation part. Using this part alone, we will choose the action with the highest expected reward at the moment which is a greedy approach. The second half of the formula represents the exploration part. In other words, if an arm has not been drawn frequently enough, this value will increase over time to encourage exploration. In our research implementation, we used UCB1 which belongs to the UCB family. The formula for UCB1 is shown here:

$$A_t = \arg\max_a \left[Q_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}} \right]. \tag{4}$$

Note in UCB1, we don't have the constant c found in the original UCB formula. Instead, we have the factor of 2 in the numerator.

2.2 Super Arm

In our model, each domain-level service can be considered as an arm of the MAB problem. We choose a set of arms for every round from an ordered set of domains. Chen et al. [15] consider such a group of arms as a *super arm* because they are played together. By treating every super arm as an arm, we can apply the classical MAB framework as desired. We will go into more depth about how we generate these super arms as a preprocessing step in the experiment section. There are a couple of well-studied methods as mentioned above to solve the MAB problem. However, this super arm approach has some limitations. According to Chen et al. [15], the number of super arms grows exponentially as the number of domains increases due to combinatorial explosion and hence classical MAB algorithms may require an exponential number of steps only to traverse all super arms. Additionally, distinct super arms may have some underlying dependencies because they share some common arms in each domain. These dependencies are not taken into account in the traditional MAB framework.

2.3 Related Work

As the resource coordination problem for E2E network slicing deals with the optimization of combinatorial resources, we survey several existing techniques in combinatorial MAB problems and related network optimization problems in the existing literature. Our work in this paper is based on the work by Gai et al. [16], which studied stochastic combinatorial multi-armed bandits with linearly weighted rewards. They presented a new efficient policy called the *Learning with Linear Rewards* that has been shown to produce regret that grows logarithmically with time and polynomially with the number of unknown variables [16].

While the study mentioned above assumes a network-structured action set, Chen et al. [15] proposes a similar approach, CMAB, with a more general action set. The CMAB framework improves the regret bound and includes both general linear and non-linear reward functions. However, the paper assumes that the expected outcomes from base arms determine the expected reward of playing a super arm. The work by Chen et al. [17] allows a general nonlinear reward function and does not require the assumption on the expected rewards. In the CMAB problem, there are three categories of feedback: the semi-bandit, the full information, and the bandit types. As will be discussed later, our resource coordination problem belongs to the semi-bandit type, where the outcomes of selected arms in each round can be observed. The full information type implies that the outcomes of all arms are revealed in each round, which is not realistic for our network setting where the domain orchestrators only reveal limited information. As a natural application of the CMAB problem, stochastic online shortest path routing with end-to-end feedback was studied by Zhu and Modiano [18]. Their end-to-end feedback refers to the bandit type where the aggregated outcome of all selected arms is observed while the outcome of each individual arm is not observable. They designed efficient algorithms that can utilize the network structure to achieve nearly optimal regrets for the problem.

The same problem was addressed by Liu and Zhao [19], where they proposed an adaptive shortest-path routing algorithm. However, their approach requires a brute-force search over the action space. Kovacevic et al. [20] presented another framework where the delay requirements on each network domain can be redefined depending on the conditions in the rest of the network. This differs from our network setting.

CHAPTER 3

An Global Resource Coordination Architecture

This chapter introduces a global resource coordination architecture. Section 3.1 describes the network architecture as well as the coordination mechanism between the global slice coordinator and domain orchestrators. Section 3.2 defines the slice request format and discusses about its generality. Finally, section 3.3 concludes with an explanation of domain reports.

3.1 Network Domains and Slice Coordination

An E2E network slice is instantiated over multiple control network domains spanning access, transport, and core networks. The physical infrastructure of each part of the network would have different types of resources. For example, access networks cover relatively small areas and hold capacity-limited computing resources like edge servers. In contrast, transport and core networks cover wider areas and sustain larger computing facilities including cloud data centers. Such network domains are operated by different infrastructure operators who do not share information, such as topology and routing policy, with each other. Such domain-specific information is usually monitored and managed by domain orchestrators.

Therefore, the instantiation of E2E slices is coordinated by a global slice coordinator who communicates with the domain orchestrators. In this paper, we assume a communication model where the global coordinator has a more active role in planning slice deployments, while the domain orchestrators focus on maintaining a resource status view and mapping requests that are communicated from the global coordinator onto the actual infrastructure (See Figure 4). More precisely, domain orchestrators report available resource information, such as available *domain-level services* and their expected delay, to the global coordinator. Each domain-level service can be characterized by a pair of ingress and egress nodes, but the global coordinator does

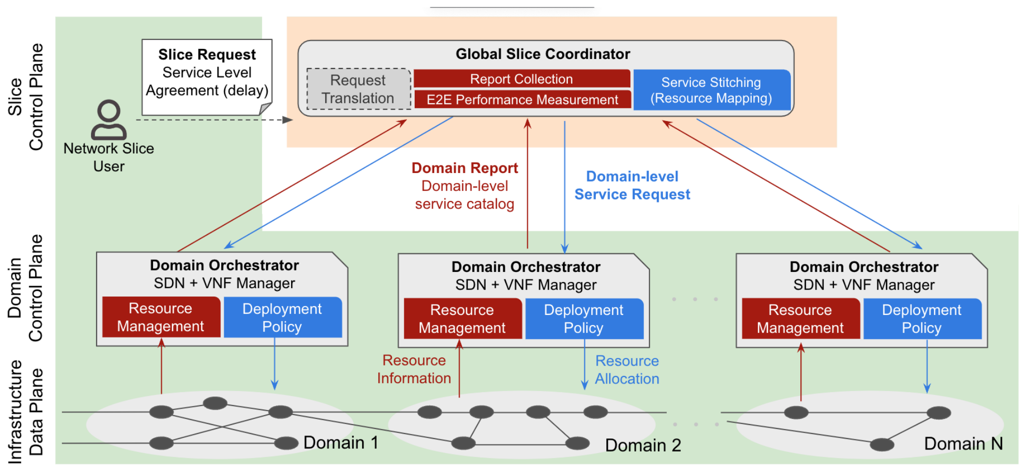


Figure 4: An Global Resource Coordination Architecture based on [9].

not know how the two endpoints are connected internally in the domain. While domain orchestrators may choose different routing policies to provide the domain-level services, a general assumption is that a logical service link between a pair of ingress and egress nodes within each domain is supported by technologies like MPLS tunnels over IP/MPLS. Therefore, even when two links on network slices use the same domain-level service, the associated traffic does not necessarily travel the same physical path in the domain. Also, traffic patterns of network slices mapped on the service would have an influence on the performance metrics of the services.

The global coordinator also serves the role of the entry point of user slice requests. A slice user sends service specification, which includes E2E end-nodes and an E2E delay requirement, to the coordinator, and the coordinator translates the E2E requests into domain-level deployment requests. Each domain orchestrator reserves an appropriate set of resources in its corresponding domain to satisfy the domain-level requests when receiving them from the global coordinator.

3.2 Slice Request

Each slice request consists of requirements for communication. In general, these requirements are carried over from the slice user's SLA with the provider. These requirements include data throughput, traffic capacity, number of users, latency, reliability, and availability.

In this paper, we represent an E2E request in the following format:

$$r = (s, t, C_r), \tag{5}$$

where C_r represents the upper bound for the sum of edge weights on a selected path p from node s to t. To put it simply, the following path weight requirement needs to be satisfied:

$$\sum_{e \in p_{s,t}} w(e) \leq C_r, \tag{6}$$

where w is an edge weight function. The simplest interpretation of the request format is the delay requirement. Each communication link on a path incurs a delay, and the total delay needs to be less than the requested E2E delay bound. In contrast, many other metrics and their combinations can be represented as edge weights. For example, the availability metrics can be represented as edge weights after taking the logarithm of availability probability [21]. Therefore, this representation in our paper still holds some level of generality though we discuss the simplest scenario with the delay constraints in the rest of this paper.

3.3 Domain Report

Each domain orchestrator manages communication and computation resources. It is reasonable to assume that the orchestrators do not reveal all the internal details, including domain topology and individual resource capacity, while they provide the resources in the domain as a service. Hence, it can be assumed that the domain orchestrators periodically report only the essential information about the domain state, so slice users (through the global coordinator) can select a set of appropriate services from multiple domains to realize E2E slices. We assume that the domain reports from domain orchestrators include available domain-level path services and their expected delay performance. The delay performance is the expected one-way latency for a domain-level service based on past data, which can be inaccurate due to the network's dynamic nature and other administrative factors.

CHAPTER 4

Problem Formulation

The global slice coordinator can be interpreted as a request interface between slice users and domain orchestrators, who actually own the infrastructure. From the domain orchestrators' perspective, the global coordinator bridges them and multiple slice users. This request delegation to the global coordinator by slice users suggests a view that the global coordinator becomes a pseudo-user for domain orchestrators. Therefore, the deployment problem of E2E slices can be formulated as an optimization problem solved by the global coordinator, given partial domain state views from the domain orchestrators. For each slice request, the global coordinator needs to solve the following problem: How should it select a set of available services from these domains to meet the requirements?

The whole network, including access, transport, and core networks, is denoted as:

$$G = (V, E). \tag{7}$$

A network domain in G is denoted as:

$$d_i = (V_i, E_i). \tag{8}$$

- V_i represents a set of communication nodes.
- E_i represents a set of communication links in the domain.
- The end nodes of an edge must be in V_i for the edge to belong to E_i.

The node set V_i is partitioned into two subsets; namely, internal nodes V_i^{internal} and border nodes V_i^{border} that have direct links to border nodes in other domains. A set of available domain-level services $P(d_i)$ in each domain d_i is reported in the form of pairs of border nodes (ingress and egress nodes) as the pair specifies an underlying transit path. It can be represented as:

$$P(d_i) = \{(s, t)_m\}, \tag{9}$$

where $s, t \in V_i^{\text{border}}$ and m is an index.

Suppose that each domain orchestrator proposes a set of candidate paths within the corresponding domain (including communication resources), $P(d_i) = \{(s, t)_m\} \; \forall d_i \in D$. When receiving an E2E request r_k, the global slice coordinator selects a set of domain paths from the candidate paths to satisfy the request. The selected E2E path needs to be a path instance $p(r_k) := \langle (s^{(0)}, t^{(0)}), (s^{(1)}, t^{(1)}), ..., (s^{(n)}, t^{(n)}) \rangle \in P(d_i) \times P(d_j) \times \cdots \times P(d_m)$ spanning a subset of domains $d_i, d_j, ..., d_m \in D(r_k) \subseteq D$, where $D(r_k)$ is an ordered set of domains that are selected to host the requested E2E slice. Note that the superscript (x) indicates the ordinal number of the domain path in the E2E path instance. The ordered set of domains could be generated by many different path-finding algorithms. For example, in this paper, we generate the ordered set by running Dijkstra's algorithm on a snapshot of a global abstract network graph that is constructed based on the collection of domain service reports. After finding a shortest path between a given pair of end nodes of an E2E network slice, we can order the domains according to the order in which they appear on the path. To guarantee the connectivity as an E2E path, an edge $(t^{(x)}, s^{(x+1)}) \; \forall x \in [n-1]$ must exist in E. The edge does not belong to any domain ($\{E_i\} \; \forall i$) since it is a communication link between two domains. After deploying the E2E slice using the path $p(r_k)$, the global coordinator monitors the actual performance during the slice holding time.

CHAPTER 5
Combinatorial Multi-Armed Bandit with Augmented Semi-Feedback

In this chapter, we briefly talk about our methodologies. Section 5.1 explains the general settings for the CMAB framework. Section 5.2 introduces the Learning with Linear Rewards algorithm and explains how it fits in our scenario. Finally, section 5.3 explains our hybrid approach and its advantages.

5.1 Combinatorial Multi-Armed Bandit

In this paper, we introduce a version of the CMAB framework. We have a set of domains $D = \{d_i\}$ where $i \in \{1, ..., N\}$ and N is the maximum number of domains in this network. At each decision period t with a newly arriving E2E slice request, an N-dimensional action vector $a(t)$ is selected. Each element $a_i(t)$ in this vector $a(t)$ is binary (0 or 1). When $a_i(t) = 1$, a service in the corresponding domain d_i is selected at time step t.

When a service is selected within a domain, its actual delay performance will be revealed. We refer to this type of feedback as *semi-bandit* feedback because the coordinator only examines the outcomes of selected services in a selected subset of domains in a single round of play. We denote the actual observed delay performance for the selected service in the domain d_i as $X_i(t)$. Thus, the cumulative delay is defined as:

$$L(t) = \sum_{i=1}^{N} a_i(t) X_i(t). \tag{10}$$

The global coordinator is trying to choose the optimal set of services at round t to satisfy the request r_k from a predefined set of domain-level services based on the measured delay of previously selected services from previous rounds, without knowing the whole network topology and their delay metrics.

When observing the actual delay incurred by each domain-level service at time t, the regret of the global coordinator's service selection $a(t)$ can be represented as:

$$R_t = \left(\sum_{i=1}^{N} a_i(t) X_i(t) \right) - L^*(t), \tag{11}$$

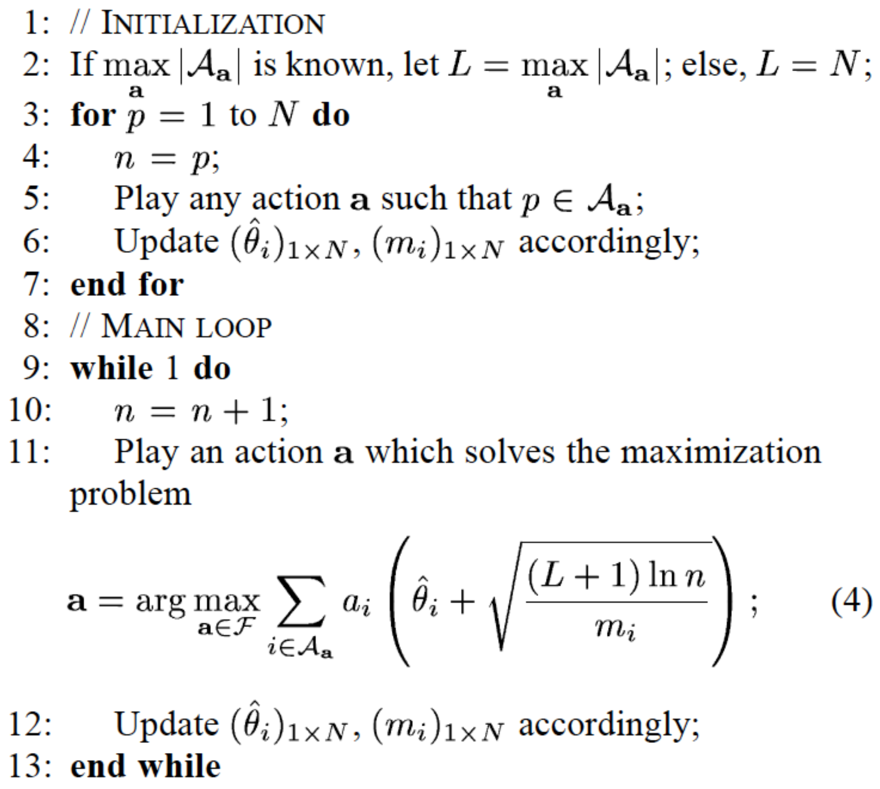
where $L^*(t) := \max_a \{ \sum_{i=1}^{N} a_i(t) X_i(t) \}$ represents the optimum path with the minimum delay, which can be obtained only when the global coordinator has access to full information of the entire topology and actual delay.

5.2 Learning with Linear Rewards (LLR) in the Domain-level Service Stitching

When the global coordinator stitches a set of domain-level services to compose an E2E network slice, the delay of the E2E path and individual links only can be estimated based on historical patterns. Since each communication link would host multiple slices on it, the delay of domain-level services, which are composites of the communication links, could change very dynamically over time. However, it is reasonable to assume that the delay follows a certain statistical pattern or a distribution as the relative popularity of each domain-level service should be relatively stable. From this perspective, we can treat the delay of each domain-level service as a random variable with an unknown mean θ. In our context, estimating the mean accurately is equivalent to knowing the average delay performance of each domain-level service. Thus, slice resource optimization can be statistically solved based on the average performance.

Gai et al. proposed the Learning with Linear Rewards (LLR) algorithm that fits the aforementioned property of domain-level service stitching. The LLR algorithm makes use of the UCB1 policy, which employs the Chernoff-Hoeffding bound [16]. We have the following formula in the LLR algorithm:

$$\hat{\theta}_i + \sqrt{\frac{(L+1)\ln n}{m_i}}, \tag{12}$$



```
1: // INITIALIZATION
2: If $\max_{\mathbf{a}} |\mathcal{A}_{\mathbf{a}}|$ is known, let $L = \max_{\mathbf{a}} |\mathcal{A}_{\mathbf{a}}|$; else, $L = N$;
3: for $p = 1$ to $N$ do
4:     $n = p$;
5:     Play any action $\mathbf{a}$ such that $p \in \mathcal{A}_{\mathbf{a}}$;
6:     Update $(\hat{\theta}_i)_{1\times N}$, $(m_i)_{1\times N}$ accordingly;
7: end for
8: // MAIN LOOP
9: while 1 do
10:     $n = n + 1$;
11:     Play an action $\mathbf{a}$ which solves the maximization problem
            $\mathbf{a} = \arg\max_{\mathbf{a}\in\mathcal{F}} \sum_{i\in\mathcal{A}_{\mathbf{a}}} a_i \left(\hat{\theta}_i + \sqrt{\frac{(L+1)\ln n}{m_i}}\right)$;   (4)
12:     Update $(\hat{\theta}_i)_{1\times N}$, $(m_i)_{1\times N}$ accordingly;
13: end while
```

Figure 5: The Learning with Linear Rewards Algorithm by Gai et al. [16].

which is essentially an adaptation of the UCB1 formula that we have shown before. Note that in step 11 of the LLR algorithm, the maximization problem needs to be solved in polynomial time in order for the algorithm to have polynomial computation at every step [16]. In our research setting, we aim to minimize the overall latency rather than maximize the rewards. Thus the formula is modified into:

$$\hat{\theta}_i - \sqrt{\frac{(L+1)\ln n}{m_i}}. \tag{13}$$

Additionally, the maximization problem is converted into a minimization problem and is solved with the shortest path algorithm during implementation. By maintaining the moving average $\hat{\theta}$, the observed times m_i and the uncertainty value, like the Upper Confidence Bound (UCB) value, during MAB rounds, the LLR estimates the true average in an environment with semi-bandit feedback. Unlike other super arm approaches where each arm corresponds to an E2E path, the LLR scales better by avoiding the enumeration of an exponentially large set of paths.

5.3 Our Hybrid Approach

Our hybrid approach augments the efficient LLR algorithm with additional information, the domain report, as extra feedback. Even though the expected delay values included in the domain reports are not necessarily accurate, the augmented feedback with them changes the CMAB problem from the semi-feedback type to *pseudo-full* feedback type in which a part of feedback is derived from the actual delay and the rest of it includes partial information about the actual delay. Using the additional report-based weights in the LLR algorithm, we try to further improve the performance to estimate the true means of each delay random variable.

CHAPTER 6

Experiment

We conduct experiments on both a realistic network topology and randomly generated network graphs to validate the performance of the Hybrid approach. The proposed approach is compared against the ϵ-greedy algorithm, the UCB1 algorithm, a report approach, and the LLR algorithm in terms of total regret over time.

6.1 Network, Control Domains and E2E Slice Requests

Our network simulator was implemented with Python with the NetworkX package [22].

6.1.1 ATT North America

To test the performance of the proposal in a practical network topology, we utilized a network graph constructed based on the ATT North America (See Figure 6).

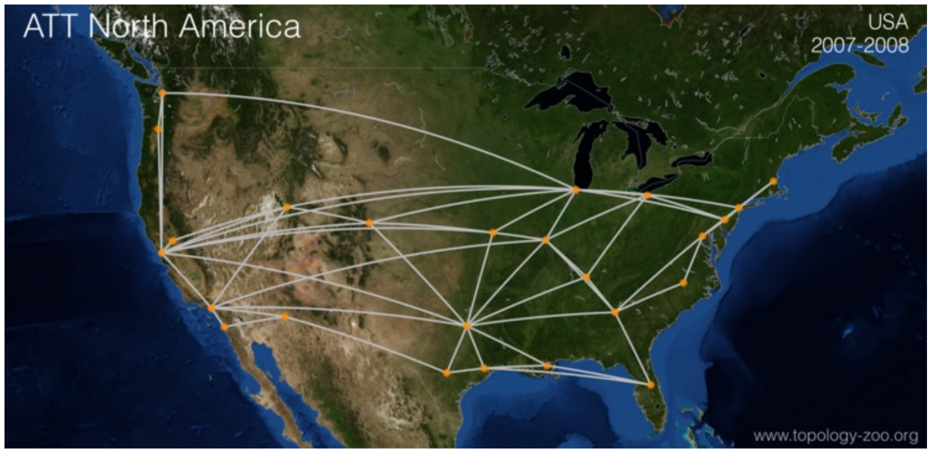


Figure 6: The ATT North America (2007 and 2008) Network Graph by the Internet Topology Zoo Network [23].

6.1.2 Synthesized Topology

In addition to the realistic graph, our simulation uses randomly generated undirected graphs to evaluate the performance of the algorithms in larger networks with more domains. The graph generator takes two parameters, the number of nodes and the probability of edge creation. An experimental graph is configured to 100 nodes with an edge generation probability of 0.018 (See Figure 7).

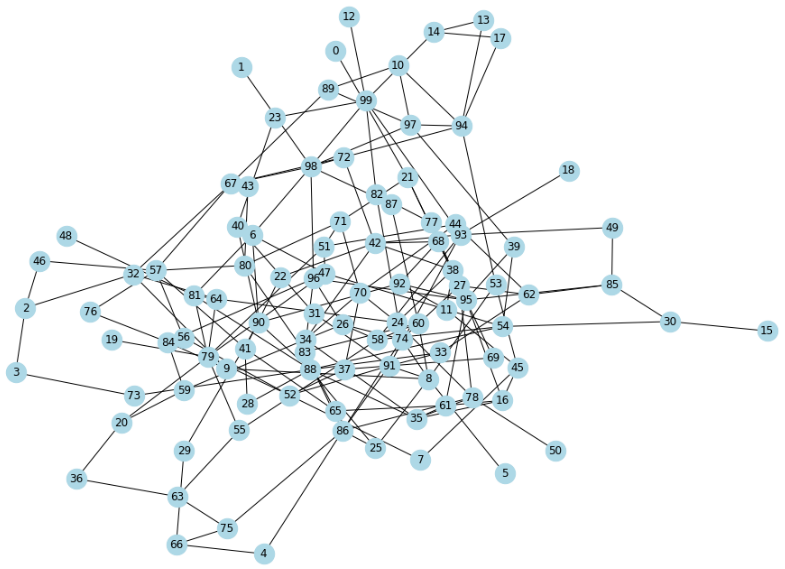


Figure 7: A Synthesized 100-node Network Typology G.

6.1.3 Domain Assignment

Control domains for both the ATT North America network and the synthesized network graph are determined by a clustering algorithm based on node proximity. Note that for both types of graphs, the initial weight of every edge is set to a random integer in the range of 5 to 15. We need to assign nodes in the network into different groups (control domains). This is a classic community detection problem. A number of methods have been developed that return good results in practical situations. In our case, we use the Clauset-Newman-Moore algorithm which mainly considers greedy modularity maximization [24]. Applying the community detection algorithm to a given network graph G, we are able to identify control domains which are represented by sets of vertex numbers. Figure 8 illustrates four different control domains defined

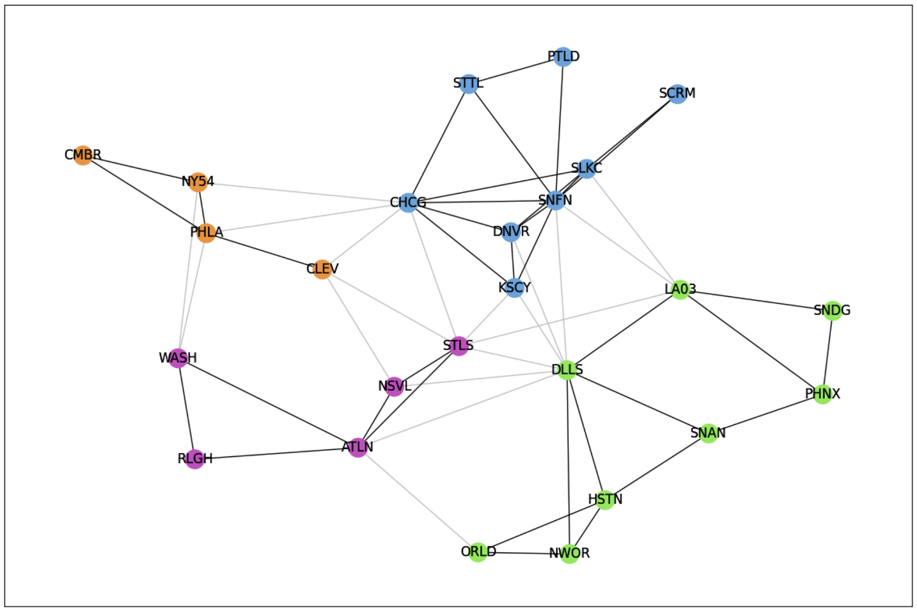


Figure 8: The ATT North America Typology with Domains.

for the ATT North America network graph. Figure 9 is a visualization of different domains for the synthesized network graph. Nodes in a control domain are colored with the same color. The domains automatically define the inter-domain links, which are represented in gray lines.

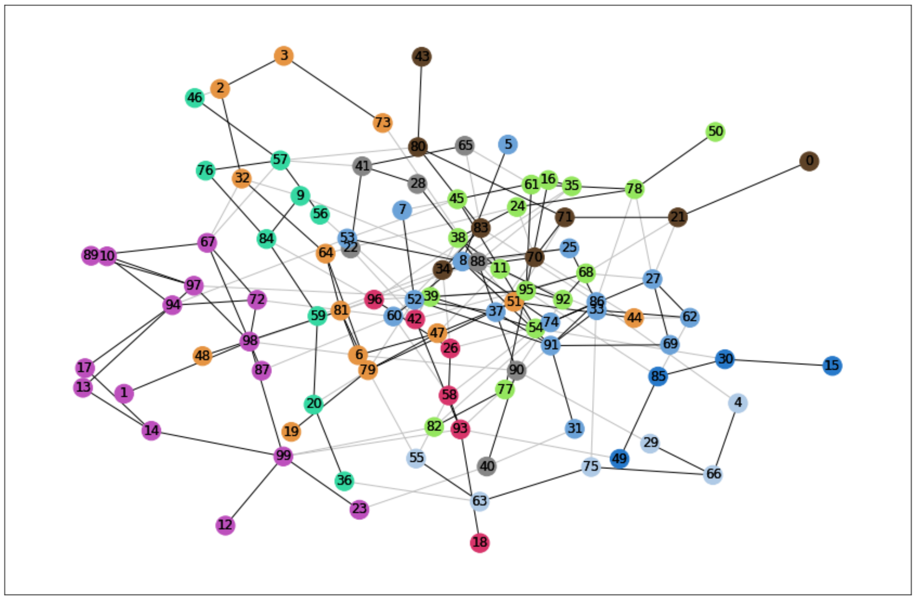


Figure 9: The Synthesized Network Typology with Domains.

6.1.4 E2E Slice Request Generation

A set of E2E requests are also randomly generated. A pair of source and destination nodes (s, t) are uniformly randomly selected from the border nodes in two district domains in a given graph G. When it comes to decomposing an E2E request into domain-level requests, we compute the weighted shortest path between s and t and extract the domains on the shortest paths as candidate domains.

6.2 Periodic Domain Reports and Traffic Observation

A domain-level service graph is created based on the selected candidate domains (See Figure 10). In the service graph, all nodes are border nodes from the domains, and other internal nodes will be aggregated. Therefore, an edge in the service graph could be a path in the underlying domain.

All edge weights are derived from the network graph G by employing the shortest path algorithm. By running Dijkstra's algorithm between the service end nodes (ingress and egress border nodes), each weight will be computed as the sum of edge weights on the shortest path. This service graph is what we refer to as the abstract view of domain services. The steps involved in creating a service graph are illustrated in Figure 10.

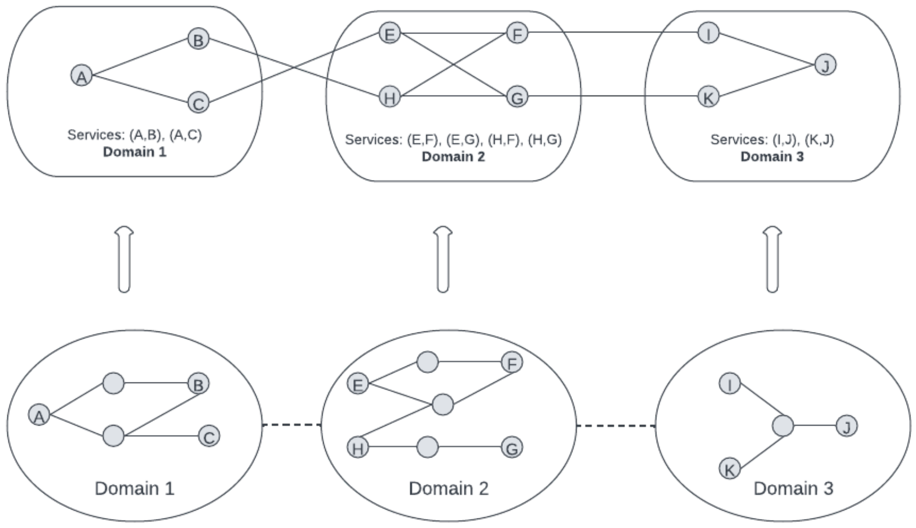


Figure 10: Composition of a Domain-level Service Graph for Slice Requests between Node A and J.

First, we have a randomly generated E2E request (A, J). After decomposing this E2E request, we have determined that domain 1, 2 and 3 are candidate domains. Based on this information, we then construct the corresponding domain-level service graph.

To simulate the dynamics of the delay and traffic pattern of each communication link, the weights of a service graph are updated every round. For each edge, we use the Gaussian distribution that produces its weight based on its edge weight from the moving average from the previous round and a standard deviation parameter σ_1. The value of σ_1 simulates the network's stability. When σ_1 is small, the delay of domain-level services is more stable. In contrast, the delay fluctuates dynamically when σ_1 increases.

The domain reports are generated by adding a random noise value to the true delay value on the service graph. In practice, even a domain orchestrator does not know the true delay and needs to estimate the expected delay to report. Also, in some scenarios, it is possible to assume that a domain orchestrator intentionally reports inaccurate expected delays to manipulate the service selection phase. The noise values added to produce the simulated domain reports represent these scenarios. The noise is generated based on Gaussian distribution where its mean is the true delay in the service graph and its standard deviation is a parameter σ_2. Similar to σ_1, the value of σ_2 indicates the accuracy of the domain reports as the increase in σ_2 more likely diverts the report from the true delay values.

6.3 Baseline Comparison

The performance of the Hybrid approach is evaluated against the ϵ-greedy algorithm, the UCB1 algorithm, a report approach, and the LLR algorithm in terms of total regret. The ϵ-greedy algorithm and the UCB1 algorithm both adopt the super arm approach, where each arm is a predefined path. In the initialization phase of these two algorithms, sampling-based uniform exploration is executed to obtain a set of valid paths. Then, the algorithms choose their service path selection from the set of valid paths and update the weights accordingly, utilizing the feedback of selected services. These algorithms only get their service path selections from a set of these pre-computed paths. For the Report approach, we select the shortest path from the service graph in each time slot using the state reports, assuming the reports are completely accurate. Note that here we assume the global coordinator has access to the service graph's typology but not its actual edge weights. For the LLR algorithm, we store and use feedback for each domain-level service (an edge in a service graph) instead of a pre-computed path. The weights of each domain-level service is computed by:

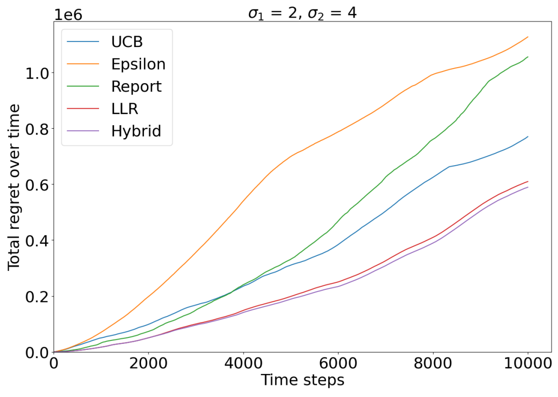
$$\hat{\theta}_i - \sqrt{(L+1)\ln n/m_i}, \tag{14}$$

where L refers to the number of edges in the service graph, $\hat{\theta}_i$ is the average of all the observed values for the edge up to the current time slot, and m_i is the number of times this edge has been observed up to the current time slot. Lastly, our Hybrid approach incorporates the LLR algorithm's method. We also take advantage of the domain reports to update the weights of domain-level services that are not selected.

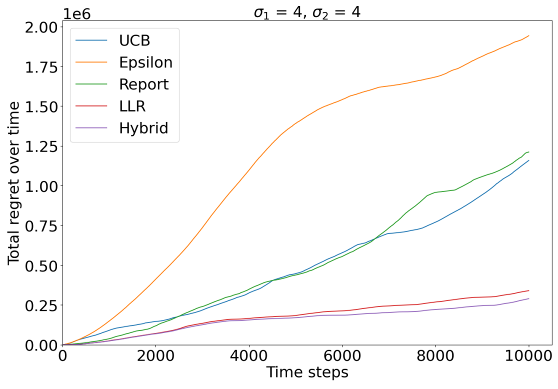
6.4 Results

The experiments in both the ATT North America and synthesized graphs demonstrate quite similar results. As the problem is inherently more difficult in larger networks with more domains, we show the results in the synthesized graphs in this paper. Figure 11 shows the results of total regret over time for networks with different levels of delay fluctuations. As discussed above, the stability of the network (or the link delay metric) is tuned by a parameter, σ_1. The experiment was executed over 10,000 time steps with σ_2 set to 4.

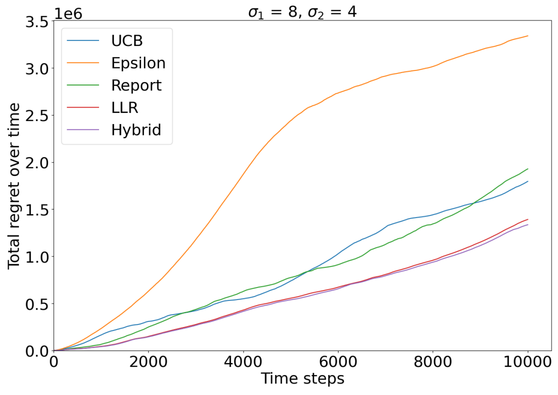
Our Hybrid approach performs the best, while the ϵ-greedy algorithm performs the worst in all cases. The simple report approach, however, eventually catches up to the UCB1 algorithm and surpasses it as our network is becoming more unstable. This is anticipated for several reasons. First, applying the UCB1 algorithm directly to super arms is a relatively naive strategy. We compute the uncertainty value for each super arm based on its moving average and frequency of play. The moving average in this case cannot capture the true mean of each super arm because the network is highly unstable and each basic arm may fluctuate considerably throughout each round. Secondly, as stated in the background chapter, the super arm approach disregards the interdependencies between super arms. Thus, when updating in each round, only one



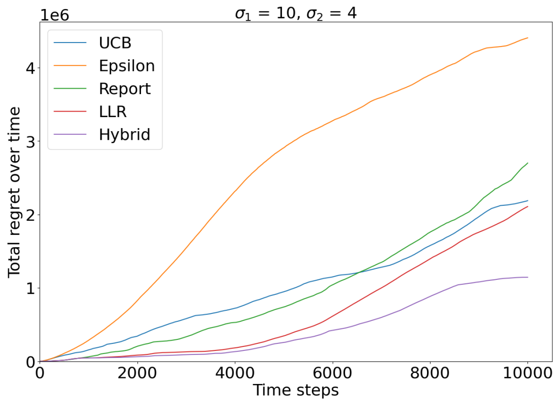
(a) $\sigma_1 = 2$



(b) $\sigma_1 = 4$



(c) $\sigma_1 = 8$



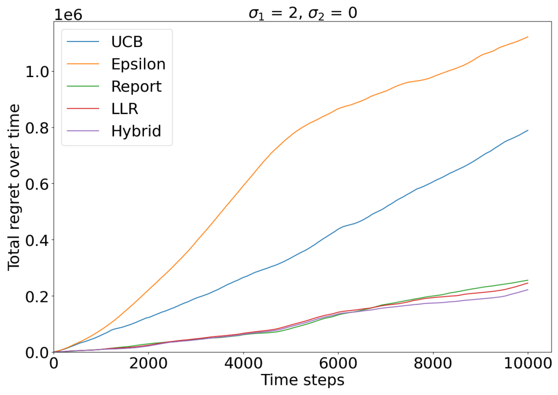
(d) $\sigma_1 = 10$

Figure 11: Total Regret Under Different Levels of Network Stability.

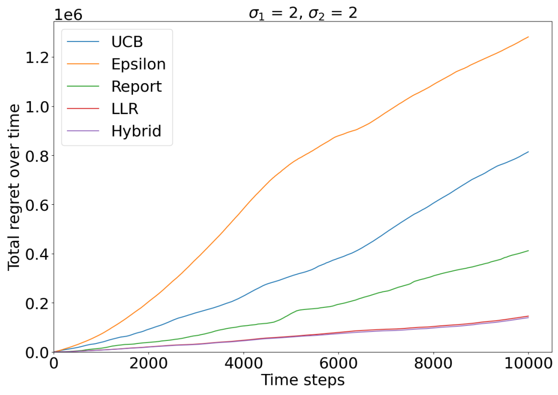
super arm is updated, whereas in the LLR, a collection of arms that can be used to compose other super arms is updated. Lastly, the straightforward report approach implies access to the service graph's typology, which the UCB1 approach does not have.

The results of total regret over time for different domain report accuracy are shown in Figure 12. The simulation was executed over 10,000 time steps with σ_1 set to 2. Each plot point is obtained by averaging the result from 10 different requests.

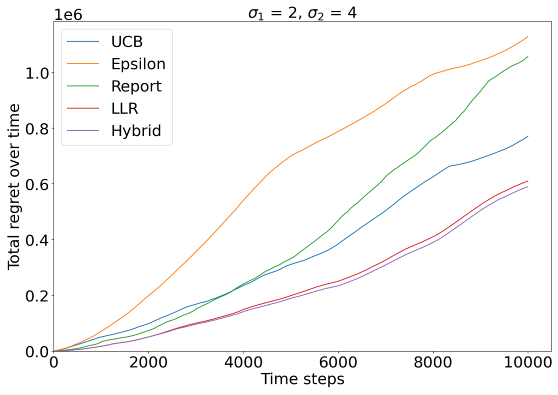
Our Hybrid approach achieves the best results, while the ϵ-greedy algorithm achieves the lowest levels of performance in every scenario. When $\sigma_2 = 0$, the performance of the report approach, the LLR, and the Hybrid approach are all



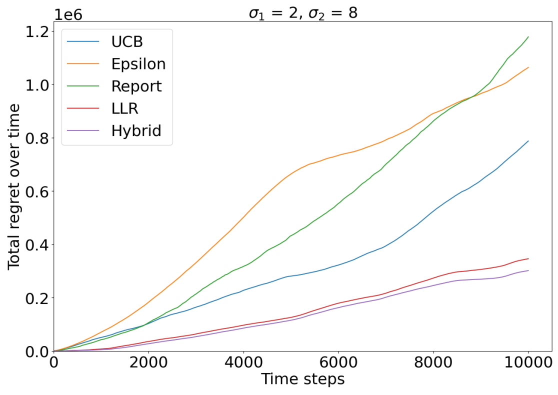
(a) $\sigma_2 = 0$



(b) $\sigma_2 = 2$



(c) $\sigma_2 = 4$



(d) $\sigma_2 = 8$

Figure 12: Total Regret with Different Levels of Domain Report Accuracy.

comparable. This is to be expected due to the fact that, when the network is relatively stable and the report accuracy is high enough, the report approach and the Hybrid approach will gain the most from the domain reports. It is worth mentioning that even if the report accuracy is 100%, we still receive a positive regret since the service graph does not necessarily represent the weights of the next time step but does represent the average performance.

A surprising result is that our hybrid method outperforms the LLR even in situations where the report accuracy is relatively worse, as demonstrated in Figures 12(c) and 12(d). This result indicates the potential benefit of incorporating extra report information in the bandit algorithm. According to this result, some information useful to estimate the overall performance of domain-level services can be extracted, even when the domain report does not quite reflect the delay of the next time step.

CHAPTER 7

Conclusion

The resource coordination problem for E2E network slicing is computationally intractable. In particular, resource coordination under the limited visibility posed by the independent administrative network domains demands a new learning-based approach to realize efficient resource utilization across multiple domains. In this paper, we present a new hybrid approach that incorporates abstract domain reports into the existing Learning with Linear Rewards algorithm. Experiment results show that our proposed algorithm performs significantly better than other algorithms. In recent years, machine learning algorithms, particularly graph neural networks (GNNs), have been proposed to solve the scalability issue of combinatorial optimization problems [25, 26]. To improve the scalability of our method, we plan to employ GNNs in future research to encode complex network structures. We are also interested in determining whether the technique developed in this study can be applied to combinatorial bandits with general nonlinear reward functions. In addition, we wish to empirically validate and demonstrate the efficacy of our algorithm in practice.

LIST OF REFERENCES

[1] S. Zhang, ‘‘An overview of network slicing for 5g,’’ *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111--117, 2019.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, ‘‘A survey on software-defined networking,’’ *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27--51, 2015.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, ‘‘Network function virtualization: Challenges and opportunities for innovations,’’ *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90--97, 2015.

[4] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, ‘‘A service-oriented deployment policy of end-to-end network slicing based on complex network theory,’’ *IEEE Access*, vol. 6, pp. 19 691--19 701, 2018.

[5] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, ‘‘On radio access network slicing from a radio resource management perspective,’’ *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166--174, 2017.

[6] P. L. Vo, M. N. H. Nguyen, T. A. Le, and N. H. Tran, ‘‘Slicing the edge: Resource allocation for ran network slicing,’’ *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 970--973, 2018.

[7] Y. Sun, M. Peng, S. Mao, and S. Yan, ‘‘Hierarchical radio resource allocation for network slicing in fog radio access networks,’’ *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3866--3881, 2019.

[8] A. Kaloxylos, A. Gavras, D. Camps Mur, M. Ghoraishi, and H. Hrasnica, ‘‘Ai and ml – enablers for beyond 5g networks,’’ Dec. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4299895

[9] M. Buhrgard, ‘‘GR ZSM 011 V1.1.1zero-touch network and service management (ZSM); intent-driven autonomous networks; intent-driven autonomous networks,’’ February 2023.

[10] ETSI, ‘‘GS NFV-MAN 001 V1.1.1 network function virtualisation (NFV); management and orchestration,’’ 2014.

[11] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, ‘‘Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models,’’ *IEEE Network*, vol. 33, no. 6, pp. 172--79, 2019.

[12] T. Lai and H. Robbins, ‘‘Asymptotically efficient adaptive allocation rules,’’ *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4--22, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0196885885900028

[13] S. Bubeck and N. Cesa-Bianchi, ‘‘Regret analysis of stochastic and nonstochastic multi-armed bandit problems,’’ 2012.

[14] R. Agrawal, ‘‘Sample mean based index policies by o(log n) regret for the multi-armed bandit problem,’’ *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054--78, 1995.

[15] W. Chen, Y. Wang, and Y. Yuan, ‘‘Combinatorial multi-armed bandit: General framework, results and applications,’’ in *30th International Conference on Machine Learning*, ser. 1, 2013, pp. 151--159.

[16] Y. Gai, B. Krishnamachari, and R. Jain, ‘‘Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations,’’ *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1466--1478, 2012.

[17] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, ‘‘Combinatorial multi-armed bandit with general reward functions,’’ 2018.

[18] R. Zhu and E. Modiano, ‘‘Learning to route efficiently with end-to-end feedback: The value of networked structure,’’ 2018.

[19] K. Liu and Q. Zhao, ‘‘Adaptive shortest-path routing under unknown and stochastically varying link states,’’ 2012.

[20] I. Kovacevic, A. S. Shafigh, S. Glisic, B. Lorenzo, and E. Hossain, ‘‘Multi-domain network slicing with latency equalization,’’ *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2182--2196, 2020.

[21] R. Gour, G. Ishigaki, J. Kong, and J. P. Jue, ‘‘Availability-guaranteed slice composition for service function chains in 5g transport networks,’’ *J. Opt. Commun. Netw.*, vol. 13, no. 3, pp. 14--24, Mar 2021. [Online]. Available: https://opg.optica.org/jocn/abstract.cfm?URI=jocn-13-3-14

[22] A. Hagberg, P. Swart, and D. S Chult, ‘‘Exploring network structure, dynamics, and function using networkx,’’ Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[23] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, ‘‘The internet topology zoo,’’ *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 --1775, october 2011.

[24] A. Clauset, M. E. J. Newman, and C. Moore, ‘‘Finding community structure in very large networks,’’ *Physical Review E*, vol. 70, no. 6, dec 2004.

[25] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Velickovic, ‘‘Combinatorial optimization and reasoning with graph neural networks,’’ in *IJCAI International Joint Conference on Artificial Intelligence*, 2021, pp. 4348--4355, cited By :15. [Online]. Available: www.scopus.com

[26] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman, ‘‘Learning combinatorial optimization on graphs: A survey with applications to networking,’’ *IEEE Access*, vol. 8, pp. 120 388--120 416, 2020.