Competitive and Regret Analysis for Online Optimization

YANG, Lin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
September 2018

Abstract of thesis entitled:
Competitive and Regret Analysis for Online Optimization
Submitted by YANG, Lin
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in September 2018

For optimization problems involving time-varying input, if the entire input is available from the start, decisions of an algorithm can be determined offline. However, in practice, knowledge on the system is usually limited, and input for the algorithm can be only acquired piece by piece. In this case, optimization for algorithm decisions must be conducted in an *online* fashion. Motivated by this, there are increasing research interests on online optimization and efficient online algorithms, whose decision making only depends on current or past inputs. This thesis focuses on several influential online optimization paradigms in computer science and operations research: the online QoS (Quality of Service) buffer management problem, the online trading problem, and the online learning problem.

The online QoS buffer management problem is a standard online paradigm in resource allocation. In its basic setting, packets with different values defined according to their QoS requirements arrive in an online fashion to a switching node. To maximize cumulative profit, the switch needs to decide whether to admit the incoming packet based on the packet value and its buffer availability. Even though this problem was proposed more than a decade ago, no optimal online solution has been proposed in the literature. In this thesis, we define a new online algorithm by leveraging a novel construction involving virtual queues, and prove that it can achieve the lower bound of the competitive ratio for any online algorithms.

This thesis also investigates another important application of online optimization in online trading. Nowadays, the electricity market has become more deregulated, resulting in individual participants to design more intelligent trading algorithms. The individual consumers and electricity suppliers both need to bid to offer or procure along with other competitors, faced with unpredictable market price fluctuation. Compared to traditional study, the novelty of our work is that we study a more general scenario where the cargo can be stored somewhere, as in a pumped storage system, for more desirable prices. In the presence of storage, it is more challenging to design online trading strategies, due to the additional design space enabled by the storage. In this thesis, we study the online trading problems from both sides of the electricity market (energy suppliers and consumers) and provide optimal online solutions, respectively.

The last investigated online optimization problem is the online learning problem, where regret is taken as an important performance metric for an online algorithm. Much efforts have been devoted to the Expert learning problem and the Online Convex Optimization (OCO) problem etc. Those traditional frameworks are limited in some recent application scenarios, especially when the learning set is from a metric space or the learning algorithm is faced with non-convex losses. Motivated by this, we focus our efforts on the Online Non-Convex Learning problem (ONCL)[1], which generalizes the OCO problem by removing the convexity assumption on both the cost function and the decision set. In this thesis, we propose a novel online algorithm, which improves the known result $O(\sqrt{T \ln T})$ to $O(\sqrt{T})$, achieving the lower bound for the OCO problem and filling the regret gap between the state-of-the-art results in online convex and non-convex learning problems.

[1] It is also called the Lipschitz Expert problem in some literature.

摘要

在实际中，系统的特性通常是未知的，而且算法的输入只能随着时间逐步获取，使得算法的决策必须以在线的方式确定。近来，很多研究集中在在线优化问题及相应的在线算法上。本文研究了计算机科学和运筹学中几种经典的在线优化问题，即在线QoS （Quality of Service）队列管理问题，在线交易问题，以及在线学习问题等。

在线队列管理问题的具体场景如下：根据服务质量，网络中每个数据包都被分配一个特定的正值来量化其被成功转发时交换机所获得的收益。这些包以任意的规则到达网络节点。为了最大化转发数据包获得的收益，交换机需要根据数据包的值以及当前的队列状况决定是否接收数据包。该问题已十多年的历史，但仍没有最佳的在线解决方案。在本文中，我们提出了一个基于虚拟队列的在线算法，并证明其可以达到最优的competitive ratio。

此外，本文还探讨了在线交易问题。目前，电力市场越来越自由化，个人用户以及电力供应商都可以在电力市场中以很大的自由度来进行交易，但同时也要面对未知的市场价格波动所带来的风险。本文的创新之处在于，我们研究了一种比以往更一般化的场景，即电力可以暂时存储在储能系统中，使得市场参与者可以以更理想的价格进行交易。这大大增加了问题的挑战性，因为这样大大扩展了算法设计的复杂度。本文针对可存储的在线交易问题（包括能源供应商和个人消费者），我们分别提出了最优的在线算法。

这篇论文研究的最后一个问题是在线学习问题。近来，很多工作致力于研究所谓的专家问题和在线凸优化问题等等。尽管对这些问题的研究取得了很大的成功，但是在有些特定的场合下，这些传统方法受到了一定的限制。特别是当决策集是连续的或者目标函数是非凸的时候。基于此，我们研究了更一般化的非凸在线优化问题。以往的结果表明，经典的在线学习方法的收敛速度是$O(\sqrt{T}\ln T)$。在本文中，我们提出了一种新的在线学习算法，该算法的收敛速度是$O(\sqrt{T})$，达到了非凸在线优化问题的性能下界。

ACKNOWLEDGEMENT

Except the research topics which could have a predictable result, I was rarely able to foresee the things to happen in my life. However, one thing that I can make sure is that, behind each success of my life, there is always generous help from the people that I have met. This time is no exception.

First of all, I would like to give my ineffable thanks to my advisor, Professor Wing Shing Wong, one of the nicest people that I have met. I have to admit that it is really a lucky thing for me to work with Professor Wong during my PhD study. He taught me limited knowledge, but revealed me an unlimited world of research. And more importantly, he guided me to cherish the passion of exploring the unexplored, enjoy the moment of achievements and build myself interests in any interesting ideas. This is one of the most valuable gifts I have received in the past three years and will definitely benefit my future research career.

I would also thank the outstanding faculty members in CUHK who offer me generous help in my study, including Professor Minghua Chen, Jianwei Huang, Soung-Chang Liew, John C.S. Lui, Shiqian Ma, Chandra Nair, Wingki Shum, Man-Cho So and Pascal Olivier Vontobel et al. I would also give my thanks to the staff members in IE department, including Connie, Kammy and Winnie et al., for their nice help.

I also thank all my collaborators including Professor Enrique Mallada and Dr. Mohammad H. Hajiesmaili et al. Especially, I value the extensive, unforgettable and helpful discussions with Ying and Lei et al.

I also thank all of my friends and labmates in CUHK. They are Yuanpei Cai, Chung Shue Chen, Lei Deng, Yixuan Ding, Yuzhe Li, Qiulin Lin, Fang Liu, Zhongchang Liu, Yuan-Hsun Lo, Jingjing Luo, Cheng Tan, Suzhen Wang, Hanling Yi, Yiding Yu, Jianing Zhai, Jinbei Zhang, Ying Zhang, Yijin Zhang, and Yuan Zhao et al. Each time when I think of the life in the past three years, my mind is filled with their laughters and faces. I would also give my thanks to Professor Hui Cheng and Professor Huanshui Zhang who generously hosted me when I visited them.

At last, I would give my deepest thanks and love to my parents, who gave me eyes to look at the beauty, brought me up with selfless love, and taught me the first lesson to explore the world. I will never forget the happy time when I was growing up.

This work is dedicated to my parents and grandfather.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Online Optimization and Online Algorithms

With the entire input available from the start, decisions of an algorithm during the investigated time interval can be optimized offline. While in many practical areas such as computer science and operations research, it is infeasible to either lay out a comprehensive deterministic/stochastic model valid from the start to the end or acquire the whole input sequence by some idealistic predictor. In other words, the knowledge for the system or the input for the algorithm has to be revealed piece-by-piece as time goes forward, and the optimization process must be conducted in an *online* fashion. With this motivation, much of the recent research is focused on online optimization problems and efficient online algorithms, whose decisions depend on only the current or past input. To illustrate the concept and methodology of the online optimization framework, we provide two simple examples in the following, which are the one-way trading problem and the Expert problem.

Paradigm 1: (One-Way Trading) Consider the fundamental one-way trading (OneTrad) or k-max search problem [58, 59], that the online player is required to convert one asset into another, e.g., dollars for yen, given a time-varying exchange rates $p(t) \in [p_{\min}, p_{\max}]$ arriving online. [1] The offline formulation of the one-way trading problem is as follows,

$$\begin{aligned} \textsf{OneTrad} \quad \max \quad & \sum_{t=1}^{T} p(t)a(t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} a(t) \leq K. \\ \text{var.} \quad & a(t) \in \{0,1\}. \end{aligned} \tag{1.1}$$

In Problem (1.1), K is an integer and used to denote the total number of asset units of the trader, and $a(t)$ is her binary action at round t. $a(t) = 1$ means selling out one unit of asset and 0 is not. The trader, without knowing the future price and ending time[2], has to decide to accept the

[1] The time series search problem and secretary problem [59, 112] are also quite similar to this context.

[2] The setting for the ending time of the game is slightly different from that of the traditional one-way

current price or wait for the more attractive prices in the future.

Paradigm 2: (Expert problem) Consider a repeated game composed of T iterations, where an online player is assisted with a finite number of "experts", labelled by an index set denoted by $\mathcal{I}$. At each round t, the online player, without knowing current and future costs on the expert, has to choose one expert as her representative. After committing to one choice, the cost on each expert, which is arbitrary and unavailable to the online player in advance, will be revealed. The cost on the chosen expert will be committed. The goal is to minimize the cumulative cost.

Optimizing above problems offline is fairly simple. Indeed, the optimal solution in the one-way trading problem is to find the time slots of which the prices are among the maximum K ones to sell one unit of the asset; and in the Expert problem, one just simply chooses the expert of the minimum cost at each round for the optimal performance. However, the future information in practice is usually unaccessible and rarely satisfies any known distributions; in extreme cases, the input trajectory may even fit more an *adversary* model, which assumes the input sequence is generated by a powerful opponent. For example, in the one-way trading problem, there is no way for the trader to access the market price in the future, and it is even difficult to only predict whether the price will go up (or down). The case is similar in the Expert problem, where the online player in many application scenarios has limited knowledge on the expert. Without the entire knowledge on the market price or the cost function of the expert, it is impossible to figure out the optimal solution. Motivated by this fact, one prefers to designing algorithms whose decisions are based on the real-time scenario rather than an elaborate mathematical model. This idea results in the methodology of online optimization which aims at providing a comparable performance guarantee under any circumstances, when compared with the optimal offline solution. The next section introduces two fundamental metrics quantifying the performance of the online algorithm.

1.2 Competitive and Regret Analysis for Online Computation

Generally speaking, the performance of an online algorithm can be measured in two aspects, i.e., *competitive ratio* and *regret*, which refer to the maximum performance ratio and gap between the optimal offline

trading problem or k-max search problem. Specifically, in the one-way trading problem, the game ends when all the units of assets are sold; and in the k-max search setting, the player is required to complete the transaction in a given time interval. While here we assume that the ending time is determined by the adversary. In Chapter 2, we will introduce an optimal online algorithm for the new setting.

solution and a particular online algorithm under any input instance, respectively. Even though both competitive ratio and regret focus on the "worst-case" performance when an algorithm is faced with arbitrary inputs, one should decide which one of them is more relevant depending on the application scenario. Generally speaking, in areas of resource management and job scheduling etc., one is usually interested in the competitive performance of an online algorithm, which measures the robustness of the online algorithm within an unknown and unpredictable environment. While in some areas as online learning, the regret metric might be more appropriate than the other candidate in reflecting the convergence performance of a learning algorithm. Taking the one-way trading problem and the Expert problem as examples, we will introduce the details on how to define the competitive ratio and regret, respectively.

For online optimization problems such as one-way trading, the goal of the online algorithm is to provide a performance bound under any input instance and the performance of an online algorithm can be measured by using competitive ratio. For the one-way trading problem, an instance constructed by an adversary, denoted by $\boldsymbol{\omega} \in \Omega$, can be defined as an input series including the price $p(t)$ over $[1, T]$, i.e.,

$$\boldsymbol{\omega} \overset{\text{def}}{=} [\omega(t) = (p(t))]_{t \in \mathcal{T}},$$

and Ω is used to denote the set of all possible instances, i.e.,

$$\Omega \overset{\text{def}}{=} \Big\{ [(p(t))]_{t=1:T} : p(t) \in [p_{\min}, p_{\max}], T \in \mathbb{Z}^+ \Big\}.$$

Given the definition of instance, the performance of an online algorithm can be measured using competitive ratio, which is referred to as the maximum ratio of the profit earned by the OPTimal offline solution (OPT) and a particular online algorithm $\mathfrak{A}$ under any input instance, i.e.,

$$\text{CR}(\mathfrak{A}) \overset{\text{def}}{=} \max_{\boldsymbol{\omega} \in \Omega} \frac{\mathsf{Prof}_{\mathsf{OPT}}(\boldsymbol{\omega})}{\mathsf{Prof}_{\mathfrak{A}}(\boldsymbol{\omega})},$$

where $\mathsf{Prof}_{\mathsf{OPT}}(\boldsymbol{\omega})$ and $\mathsf{Prof}_{\mathfrak{A}}(\boldsymbol{\omega})$ are the profit obtained by the respective optimal offline solution and the online algorithm $\mathfrak{A}$ when the input instance is $\boldsymbol{\omega}$. For a randomized online algorithm $\mathfrak{R}$, we assume the adversary is *oblivious*[3] and the competitive ratio is of the following form:

$$\text{CR}(\mathfrak{R}) \overset{\text{def}}{=} \max_{\boldsymbol{\omega} \in \Omega} \frac{\mathsf{Prof}_{\mathsf{OPT}}(\boldsymbol{\omega})}{\boldsymbol{E}[\mathsf{Prof}_{\mathfrak{R}}(\boldsymbol{\omega})]}.$$

[3] An adversary is said to be oblivious when the adversary is unaware of the action of the online player. This is opposite to the term, "non-oblivious".

In subsequent analysis in Section 2, readers will find that by carefully designing an online algorithm, the profit ratio between the optimal offline algorithm and the online algorithm is always upper bounded by $\ln\theta + 1$, where $\theta = p_{\max}/p_{\min}$. That is, the competitive ratio of the algorithm referred to is $\ln\theta + 1$, or the algorithm is $(\ln\theta + 1)$-competitive. For the above mentioned problem, $\ln\theta + 1$ is the best result that an online algorithm can achieve.

For the Expert problem, a commonly accepted goal for the online player is to choose among those experts as well as possible, i.e., minimize the *pseudo-regret*[4], which is formally defined as the difference between the online cumulative cost and the cumulative cost using an optimal offline choice in hindsight, i.e.,

$$\mathsf{regret}_T(\mathfrak{A}) \overset{\text{def}}{=} \max_{c_1,c_2,\ldots,c_T} \left\{ \sum_{t=1}^{T} \mathbb{E}[c_t(I_t)] - \min_{i\in\mathcal{I}} \sum_{t=1}^{T} c_t(i) \right\},$$

where $I_t \in \mathcal{I}$ is the index of the chosen expert at time slot t, and $c_t(\cdot)$ is the cost function of experts. This is a critical difference compared with the one-way trading problem.

A sublinear regret for an online algorithm is of practical significance, since it implies that the time average of the cost difference converges to zero through learning as time approaches infinity. For the Expert problem, by using the Hedge (or Multiplicative Weight) [108] algorithm which will be introduced in Chapter 5, one can attain a regret of $O(\sqrt{T})$, touching the known regret lower bound for any online algorithm.

In addition to the above classic paradigms, in the last three decades, the idea of online computation has rapidly spread over various areas, such as caching, resource management, server scheduling in computing systems, online learning in artificial intelligence, online search, one-way trading and portfolio selection in financial engineering etc. It is impossible for this thesis to explore the broad applications of online optimization in all areas. This thesis mainly focuses on several popular and influential online optimization problems, which are the online QoS buffer management problem, the online trading problem and the online learning problem. All of those are classic paradigms in computer science or operation research, receiving increasing research interests from the community.

[4] We call it regret in short in the rest of this thesis.

1.3 Overview of Studied Problems and Our Contributions

1.3.1 Online QoS Buffer Management

The online QoS buffer management problem, which is related to many existing works in caching, scheduling and other online allocation settings [20, 26, 91], can be viewed as a classic online optimization problem in the presence of resource constraints. The classic example is the DiffServ (differentiated service) networks, in which packets with different QoS requirements are associated with different quantized values, which characterize the profit earned by switches if the packets are successfully delivered. When the network is congested, the switches are not able to admit all arriving packets due to limited buffer capacity. Hence, to maximize its profit, the switch must decide to admit packets with higher values. Despite its classic application, the QoS buffer management problem could be considered as a general admission control problem in several state-of-the-art applications. As an example, one can consider the problem of the value-based cloud resource allocation with limited computation capacities [23, 146], in which the online jobs with different valuations must be either admitted or rejected upon their arrival based on their values and the utilization of the cloud servers.

In offline scenario in which the arrival and departure of packets are known in advance, the QoS buffer management is simple and can be solved using a linear program. However, in a real network environment, many unpredictable factors impact the profile of arriving packets, hence, offline algorithms are not practical. Thus, the main research effort has been focused on online settings in which the arrival and departure profile of packets are not known in advance. The existing algorithms usually follow the competitive analysis framework that tries to achieve a bounded performance as compared to the offline optimum without relying on future information.

The first study on QoS buffer management with multiple values appeared as early as in 2000 [20, 91]. The work was motivated by rising trends of differentiated services in networks. The early study was rapidly augmented by proposals of various problem settings [24, 31, 53, 61, 66, 67, 92, 100, 113, 121]. These problem settings can be roughly categorized according to the operation rules on the buffer or queue into the FIFO preemptive model and the non-preemptive model. In the FIFO preemptive model, packets which have been buffered in the queue are served in a FIFO manner, and can be discarded. In the non-preemptive model, the admitted packets cannot be ejected. There are also many interesting works investigating extensions of the FIFO preemptive or non-preemptive model. For example, in [53], the authors further take into account the heterogeneous packet processing time for different traffic. In [61, 66, 67], the authors address a model which involves a departure deadline for each transferred packet. In [31, 100, 121], the basic model is extended to the multiple input queues.

In this thesis, we mainly focus on the non-preemptive QoS buffer management problem in online setting. Even though the non-preemptive QoS buffer management problem was proposed more than a decade ago, no optimal online solution has been proposed in the literature. This thesis aims to provide a more complete picture for this problem by proposing: 1) A fixed threshold-based online algorithm with smaller competitive ratio than the existing results; 2) an optimal deterministic online algorithm under fractional admission model in which a packet could be admitted partially; and 3) an optimal randomized online algorithm for the general problem. We consider the last result being the most important contribution among the three. For details of those algorithms and adopted techniques, one can refer to Chapter 2.

1.3.2 Online Trading in a Deregulated Market

Another important application studied in this thesis is online trading. In daily financial activities, there are plenty of scenarios where one needs to decide how to trade in a market with an aim of pursuing profit or saving money. For example, the electricity market nowadays have become more and more deregulated, that motivates individual participants to design intelligent trading algorithms. In a deregulated market, the individual consumers and electricity suppliers both need to bid to offer or procure with other competitors, faced with a unpredictable market price due to unknown reasons.

The novelty of our work lies that we studied a more general scenario where the cargo can be stored somewhere, for example in a pumped storage system, such that the cargo can be traded at a more intriguing price. The electricity storage system for an energy supplier or individual user is necessitated by the unpredictable energy output or demand. And now it accounts for a considerable percentage of the infrastructure of the electrical power system. In the presence of storage, it is more challenging to design the online trading strategy, because of the additional design space enabled by the storage. In this thesis, we study the online trading problems from both sides of a deregulated electricity market, i.e., the energy suppliers and individual consumers.

Online Offering of Energy Suppliers

Renewable power producers, such as wind farms and solar plants, are being rapidly integrated to the power system due to increasing environment concerns. In 2015, investment on renewables sets a record of 296 billion US dollars, more than double the amount for fossil fuels [12]. In order to hedge against the inherent uncertainty of renewable generation, renewable power producers commonly equip the renewable plants with energy storage systems. In several existing electricity markets, renewable power producers receive guaranteed feed-in tariffs, e.g., "take-all-wind" policy in California's electricity market [3], which guarantees that the market absorbs the entire renewable supply at favorable fixed prices. This policy is not viable in the future as reduction in price [2] along with the environmental concerns pushes rapid penetration of the renewables. For example, in Denmark, the plan is to achieve 50% and 100% renewable generation in 2020 and 2050, respectively [10]. Hence, in eventual market with the considerable renewables' market share, it is inevitable to treat renewable producers the same as other traditional generation companies [94].

Several electricity markets such as NYISO [8], CAISO [3], and Nord Pool [9] operate in a multi-settlement manner and settle transactions at multiple timescales, i.e., day-ahead, hour-ahead [63, 87], and real-time. Considering the uncertainty of renewable output, the renewable power producers tend to participate in short-term market, specifically hour-ahead market, without suffering profit reduction caused by long-term forecasting errors [94]. In reality, CAISO's Participating Intermittent Resource Program (PIRP [3]) is already requiring wind power plants to bid into their hour-ahead market. In hour-ahead market operation, the generation companies including renewable producers submit their *offers* (including offering price and offering volume) for selling the electricity in the next hour (see Sec. 3.4.1 for more details on the hour-ahead market operation).

Motivated by this, we focus on designing profit maximization offering strategies, i.e., the strategies that determine the offering price and volume, for a storage-assisted renewable power producer that participates in hour-ahead electricity market. Designing the strategies is challenging since (i) the underlying problem is coupled across time due to the evolution of the storage level, and (ii) inputs to the problem including the renewable output and market clearing price are unknown when submitting offers. Following the competitive online algorithm design approach, we first study a basic setting where the renewable output and the clearing price are known for the next hour. Then, we consider the case in which the clearing price is unknown. Finally, we extend the above approach to the scenario where the renewable output has forecasting error. The the theoretical competitive analysis shows the online algorithm guarantees a bounded performance ratio of $O(\ln \hat{\theta})$ ($\hat{\theta}$ is the ratio between the maximum and minimum market price), compared to the optimal offline solution. In addition to that, The trace-driven experiments are provided to demonstrate that the algorithms achieve performance close to the offline optimal and outperform a baseline alternative significantly. For details, readers can refer to Chapter 4.

Online Procuring of Individual Consumers

The electricity cost constitutes a significant portion of a data center's operating expenses. For example, Google and Microsoft's energy bills are more than 30% of total cost of their data centers [125]. In recent years, there has been an unprecedented growth in the number and size of data centers, as the infrastructures of the Internet cloud services. A direct consequence is a rapid increase in the energy footprint of data centers worldwide; in 2015, the global electricity usage of data centers reached 416.2TWh, significantly higher than the UK's total consumption of about 300TWh [5]. Consequently, managing the power consumption of data center has become critically important for the operators of data centers [125]. In the recent years, there have been substantial research on reducing power consumption of data centers, e.g., incorporating on-site renewable sources [110], dynamic right sizing [107], geographical load balancing [127, 111], and on-site energy storage systems [78, 76, 48, 93], among others.

This thesis introduces a methodology on how to design energy procurement and storage management strategies to minimize the electricity bill of storage-assisted data centers. This investigation is motivated by two new developments in power management, namely, the popularization of on-site storages and the option of real-time pricing in deregulated electricity markets. Empowering data centers with on-site storages can reduce the electricity bill by shaping the energy procurement from deregulated electricity markets with real-time price fluctuations. In a data centers with on-site storage, the net energy demand could be satisfied by either purchasing the energy from the real-time market, or by discharging the local storage. When the electricity is cheap, the data center can purchase electricity to charge the storage for future usage during high price intervals.

Designing such strategies is challenging since the net energy demand of the data center and electricity market prices are not known in advance,

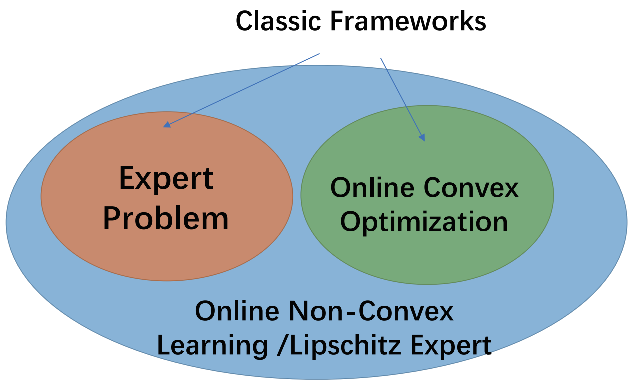

Classic Frameworks
Expert Problem
Online Convex Optimization
Online Non-Convex Learning /Lipschitz Expert

Fig. 1.1: The Online Non-Convex Learning Problem

and the underlying problem is coupled over time due to evolution of the storage level. In this thesis, we model the battery management of data centers as a novel online trading problem, which has not been studied in depth. Using the construction of virtual storage introduced in Chapter 2, we design an online algorithm that can achieve the optimal competitive ratio as a function of the price fluctuation ratio. In addition to theoretical analysis, this thesis also validates the algorithm using data traces from electricity markets and data-center energy demands. For details, readers can refer to Chapter 3.

1.3.3 Online Learning

In this thesis, we also investigate a class of fundamental online optimization frameworks, such as the classic Expert problem [72] and Online Convex Optimization problem [149]. Those frameworks have significantly influenced the machine learning community with many recent applications in artificial intelligence and financial engineering, such as online routing [30, 136], spam email filtering [79, 130], online metric learning [86], ad selection and content ranking in search engines [54, 115, 138], etc.

These problems stem from the classic Expert learning problem which was introduced by N. Littlestone and Y. Freund et al. (see [108] and [72], respectively) and rapidly enriched with many variants proposed. As an example, we have introduced the basic version of the Expert problem in Paradigm 2.

In addition to the Expert learning problem, much recent efforts have been devoted to studying the so-called Online Convex Optimization (OCO) problem since the seminal work by Zinkevich [149] et al. To some degree, the OCO problem can be seen as a generalization of the

Expert problem to a metric learning space. Specifically, the online player in the OCO framework chooses a decision point, denoted by $\boldsymbol{x}_t$, from a bounded convex learning set $\mathcal{K} \subset \mathbb{R}^n$; after the choice is committed, a bounded convex cost function $f_t \in \mathcal{F} : \mathcal{K} \mapsto \mathbb{R}$ is revealed to the player. Similar to the Expert problem, the regret is of the following form:

$$\mathsf{regret}_T(\mathfrak{A}) \stackrel{\text{def}}{=} \sup_{f_1, f_2, \ldots, f_T \in \mathcal{F}} \left\{ \sum_{t=1}^{T} \mathbb{E}[f_t(\boldsymbol{x}_t)] - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(\boldsymbol{x}) \right\}, \quad (1.2)$$

It is well known that the lower bound of the regret for the OCO problem is $\Omega(\sqrt{T})$ [80] and researchers have proposed a large number of online algorithms whose regret can attain this lower bound, (see the recent survey paper [80], and the references therein).

Despite the succuss of the Expert problem and the OCO problem, both of them have limitations in many practical scenarios, especially when the learning set is from a continuum and the cost function is non-convex. Motivated by this, our work generalizes the OCO problem by removing the convexity assumption on the decision set $\mathcal{K}$ and the cost function f_t (we generalize it to a general Lipschitz continuous function). This generalization brings out the Online Non-Convex Learning (ONCL) problem, which is necessitated by many state-of-the-art applications. For example, in the portfolio selection problem [55, 88], the decision maker (e.g., the trader) chooses a distribution of her wealth allocation over n assets $\boldsymbol{x}_t$, at each round. By the end of each round, the adversary chooses the market returns of the assets with positive values. In some specific settings [27, 104, 137], the online portfolio selection problem is a non-convex one due to the non-convex diversification constraints and non-convex transaction costs, and thus the traditional OCO framework fails in modeling such case. In addition, there are extensive machine learning research focusing on non-convex loss functions in large margin classifiers [64, 119, 144]. In [64, 119], non-convex online Support Vector Machine (SVM) models has been studied which adopts a non-convex loss function, called Ramp Loss, to suppress the influence of outliers. In [144], a special non-convex penalty, called the smoothly clipped absolute deviation penalty, is imposed on the hinge loss function in the SVM. Such a new SVM is applied to identify important genes for cancer classification [144].

For the general framework, we propose a novel online algorithm, called the Online Recursive Weighting (ORW) algorithm, and prove that the regret of the ORW is upper bounded by $O(\sqrt{T})$. The obtained regret bound matches the well-known lower bound of the regret for the

OCO [80], so the ORW algorithm is asymptotically optimal for the general ONCL problem. To the best knowledge of ours, this is the first optimal result for the ONCL/Lipschitz Expert problem.

The general idea of the ORW algorithm is to divide the decision set into multiple subsets according to a *grid layered* structure. Any subset in the upper layer is divided into multiple smaller subsets in the lower layers. Our algorithm recursively selects the subset from the topmost layer to the bottommost layer until a decision point is identified. In each layer, the ORW algorithm leverages the classic Exponential Weighting algorithm [29] to select a subset in the lower layer. As the core technical contribution, the ORW properly partitions the subsets and sets the subset-selecting probabilities, thereby, it asymptotically achieves $O(\sqrt{T})$ regret. Moreover, the ORW algorithm is extended to an adaptive version (the AORW algorithm), which increases the granularity parameter gradually as time goes on. The AORW guarantees $O(\sqrt{T})$ regret, reduces the computational complexity of the ORW, and can still work properly when the duration of time horizon is unknown to the online player. For details of the proposed algorithms, one can refer to Chapter 5.

1.4 Summary of Other Classic Online Paradigms: Known Results and Open Problems

In addition to the above introduced online optimization problems, we provide a simple account for other classic online problems, as well as related results. For more online paradigms, the readers are recommended to refer to the introductory books [39], [47] and [80].

In many practical problems, one is usually faced with a sub-problem, which is the so-called rent/buy problem. Among all kinds of variants of the rent/buy problem, a classic example is Ski rental [89, 131] where a decision maker needs to decide whether to stay in the current state, paying a samll amount of cost per time unit, or switch to another state, paying some fixed large cost but with no further payment needed. Mathematically, the problem definition is as follows:

Paradigm 3: (Ski Rental) A skier is going skiing for an unknown number of days. Assume that it costs σ per day to rent skis at a ski shop and buying skis costs Σ. At the beginning of each day, the skier has to choose between two options: continuing renting skis for one more day or buying a pair of skis. If the skier knows in advance how many days she will go skiing, say k, she can spend the minimum money, i.e., $\min\{k \cdot \sigma, \Sigma\}$. The question is what to do when she does not know in advance how many days she will ski.

For the Ski Rental problem, the break-even algorithm which advices to buy the skis at the $\lfloor \Sigma/\sigma \rfloor$-th day [90] is known to be the best deterministic algorithm. Running the break-even algorithm, one spends the cost which is always less than 2 times of that of the optimal strategy no matter the optimal strategy rents or buys the skis. In other words, the break-even algorithm is 2-competitive. By using randomization, [89] reduced the competitive ratio to $e/(e-1)$, touching the lower bound for all online algorithms, either deterministic or randomized.

Another classic paradigm where one needs to decide between switching to a new state and staying the current state is the List Update problem, whose definition is as follows:

Paradigm 4: (List Update Problem) Given a set of items in a list for users to query. We assume the inquires comes arbitrarily without satisfying any stochastic distribution. Search begins at the head of the list, and thus each inquiry incurs a cost which is proportional to the distance of the accessed item from the head of the list. One is required to come up with a strategy of reordering the list so that the total cost of accesses is minimized. The reordering can be done at any time but incurs a cost. The standard model includes two reordering actions: (a) a free transposition of the item being accessed anywhere ahead of its current position; (b) a paid transposition of a unit cost for exchanging any two items in the list.

For the List Update problem, there were many algorithm proposed, such as the MTF (Move to front) algorithm, the TRANS (Transpose) algorithm and the FC (Frequency Count) algorithm. Among them, the optimum competitive ratio for the list update problem can be attained by the MTF algorithm, which simply moves the accessed item to the front of the list without changing the order of other items. In [135], Sleator and Tarjan proved that MTF is 2-competitive by using Potential method analysis.

The Ski Rental problem and the List Update problem can be extended to a more general model, called the metrical task system [40].

Paradigm 5: (Metrical Task System) A task system is defined to a pair consisting of a set of states and and a metric distance function mapping from each state pair to a real value. An input to the task system is a sequence of cost vectors of non-negative entries that determine the processing costs for states when processing the arriving task. One is required to come up with an online algorithm for the task system which produces a schedule to determine the sequence of states. The processing cost for each time slot is the processing cost of arriving task plus the transition cost determined by the distance function. The objective of the algorithm is to find a schedule such that the cost is minimized.

As the above online problems, the most common measure to analyze algorithms for metrical task systems is the competitive analysis. For deterministic online algorithms, there is a tight bound $2n - 1$ on the competitive ratio by Borodin et al. [40], where n is the number of states. For randomized online algorithms, the competitive ratio is lower bounded by $\Omega(\log n/\log\log n)$ and upper bounded by $O\left((\log n \log\log n)^2\right)$. The lower bound is due to Bartal et al. [35]. The upper bound is due to Fiat and Mendel [68] who improved upon a result of Bartal et al. [34].

In addition to the metrical task systems, the k-server problem [117] is another fundamental problem of theoretical computer science which captures the scheduling problem in metric spaces. Its definition is as follows:

Paradigm 6: (k-server Problem) In this problem, an online algorithm must control the movement of a set of k servers, lying in a metric space consisting of n points, and to satisfy requests that arbitrarily appears in this space. As each request arrives, the algorithm must determine which server to move to the requested point. The goal of the algorithm is to keep the total distance all servers move small, relative to the total distance the servers could have moved by an optimal one who knows in advance the entire sequence of requests.

In 1990, Fiat et al. [69] first proved that there exists an algorithm with finite competitive ratio for any constant k and any metric space, and finally Koutsoupias and Papadimitriou [102] proved that Work Function Algorithm (WFA) has competitive ratio $2k - 1$ in 1995. However, despite the efforts of many other researchers, reducing the competitive ratio to k or providing an improved lower bound remains open as of 2014. The most common believed scenario is that the Work Function Algorithm is k-competitive. To this direction, in 2000, Bartal and Koutsoupias showed that this is true for some special cases (if the metric space is a line, a weighted star or any metric of $k + 2$ points). In 2011, a randomized algorithm with competitive bound $\tilde{O}(\log^2 k \log^3 n)$ was found [33]. Moreover, for a special case (the topology is a line), [106] provided a constant 3-competitive algorithm.

All of above problems are fundamental in computing system with unpredictable features. There are also a lot of classic problems in machine learning which can be formulated as an online optimization framework. Besides the Expert problem and the online convex optimization problem we have introduced, another classic framework is the multi-armed bandit problem, together with its variant versions.

Paradigm 7: (Multi-Armed Bandit Problem: Nonstochastic Version) In this problem, each arm provides a reward which is arbitrary and unavailable to the online player. Through lever pull, only the cost of the chosen arm is revealed to the player. The objective of the online player is to maximize the sum of rewards earned compared to the "best" arm over the investigated time interval.

The multi-armed bandit problem can be seen as a generalization to the Expert problem with partial information feedback. To this end, $O(\sqrt{T})$ characterizes one lower bound of regret for any online algorithms. By equipping an unbiased predictor, Auer et al. [29] defines the classic EXP3 algorithm which attains the optimal regret.

There is also a stochastic version for the multi-armed bandit problem where the reward on the arm is from a fixed distribution specific to that arm. In order to minimize the cost, the online player has to balance the tradeoff between "exploitation" of the arm that has the highest expected payoff and the "exploration" of new arms to get more information of other options. Also by Auer et al., an algorithm called the Upper Confidence Bound (UCB) [28] was designed to achieve the optimal regret, $O(\sqrt{T})$.

In addition to the above basic version, there is also extensive research studying the bandit problem in a metric space as [97, 98, 99]. Different from the discrete case, tight results for the continuous case are fairly limited, needing more investigation from the community. For more details on the bandit problem, the readers are recommended to refer to the survey paper [43].

1.5 Thesis Organization and Published Contents

The rest of the thesis is organized as follows.

Chapter 2 introduces details of the Online QoS Buffer Management problem as well as the optimal online solution we proposed. The optimal online algorithm is based on a critical technique, virtual queue, which also shows potentials to address a class of online problems. The results in Chapter 2 are mainly from my published manuscript [143] in journal of POMACS, whose conference version can be found in proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'2018).

Chapter 3 introduces details on the online trading problem for the supply side of the electricity market. The main adopted idea is the thresholding policy partly introduced in [143]. However, we wish to point out that we are considering a scenario in which the supplier has to participate in the market by online auction. This is different from the problem studied in [143]. The results in Chapter 3 are mainly from my published manuscript [142], which is accepted as an extended abstract by SIGMETRICS'2017. For more details, one can refer to the unpublished technical report [141] in arXiv.

Chapter 4 introduces details on the online trading problem for the demand side of the electricity market. The main adopted idea is the virtual queue we have introduced in [143]. The theoretical problem is novel and firstly formulated in this manuscript. The algorithm and results in Chapter 3 are mainly from my unpublished manuscript coauthored with Mohammad H. Hajiesmaili (JHU), Enrique Mallada (JHU), and my advisor, Wing Shing Wong.

Chapter 5 introduces details on a more general online learning framework than the traditional Expert and Online Convex Optimization problem. The algorithm and results in Chapter 5 are mainly from my published manuscript [140] in journal of POMACS, whose conference version can be also found in proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'2018).

The last chapter discusses possible research directions in the future.

□ **End of chapter.**

2. ONLINE QOS BUFFER MANAGMENT

2.1 Problem Background

The QoS buffer management problem, with significant and diverse computer applications, e.g., in online cloud resource allocation problems, is a classic online admission control problem in the presence of resource constraints. In its basic setting, packets with different values according to their QoS requirements, arrive in online fashion to a switching node with limited buffer size. Then, the switch needs to make an immediate decision to either admit or reject the incoming packet based on the value of the packet and its buffer availability. The objective is to maximize the cumulative profit of the admitted packets, while respecting the buffer constraint. Even though the QoS buffer management problem was proposed more than a decade ago, no optimal online solution has been proposed in the literature. This chapter contributes to this problem by firstly proposing the optimal online algorithm.

2.2 Related Results

2.2.1 FIFO Preemptive Model

The paper [91] deals with a preemptive single-queue model where the admitted packets can be discarded. The authors study a class of greedy algorithms which discard packets with the lowest value when an overflow occurs. Then, competitive ratio of the greedy algorithm is analyzed. Following [91], many other papers aimed to find better algorithms with lower competitive ratios. Generally speaking, the state-of-the-art result on the competitive ratio for a preemptive model is 1.732 [60], yet no optimal solution has been proposed. For a special case with only two different packet values, [60] introduces a deterministic strategy and proves that this strategy achieves an optimal competitive ratio of 1.282. In addition to the single-queue model, [31] and [100] study QoS buffer management within multiple queues, achieving a competitive ratio smaller than 2 for a special case where only two packet values are involved. For other work on preemptive buffer management, readers can refer to [24, 92, 113, 118], as well as the survey paper [74].

2.2.2 Non-Preemptive Model

For non-preemptive buffer management, The authors in [20] provided the first study of a two-value model. In their problem setting, packets are tagged as either being a high priority packet or a low priority packet. Specifically, they assign a benefit of $\alpha \geq 1$ to every high priority packet and a benefit of 1 to every low priority packet. Then, a general lower bound of $(2\alpha - 1)/\alpha$ for the two-value setting is proved. Then, [26] proposes an algorithm that can achieve the above lower bound for the competitive ratio. The two-value setting can be characterized as a special case of a general buffer management problem where packets are allowed to take arbitrary value in a particular region $[v_{\min}, v_{\max}]$. This generalization makes the algorithm design far more challenging. In [25], the authors prove that the optimal competitive ratio for deterministic online algorithms is lower bounded by $\ln \tilde{\theta} + 1$, where $\tilde{\theta} = v_{\max}/v_{\min}$ is the ratio between the maximum and minimum packet value. In [148], the author provides a lower bound of the competitive ratio for any online algorithm (deterministic or randomized), which is $\frac{1}{2} \ln \tilde{\theta} + 1$. [26] presents two online policies, the *Round-Robin* and *Selective Barrier* policy which set a linear non-decreasing threshold function with respect to the queue length, showing that they are both $e\lceil \ln \tilde{\theta} \rceil$-competitive. [25] improves the above results for small $\tilde{\theta}$ by proposing the *smooth selective barrier* policy. When $\tilde{\theta}$ is small (specifically, $\tilde{\theta} < e$), the competitive ratio of *smooth selective barrier* is proved to be $\ln \tilde{\theta} + 2 + O(\ln^2 \tilde{\theta}/B)$, where B is the buffer capacity. In spite of the above works, no optimal online algorithms, either deterministic or randomized have been proposed since this problem was first formulated. In this thesis, one of our most important contributions is to introduce a randomized online algorithm for the non-preemptive model which can be proved to be optimal.

2.2.3 Related Theoretical Problems and Timely Applications in a Broader Background

The Online QoS Buffer Management problem has intrinsic relations to many classic computer science problems, such as the *time series search problem* [112], the *one-way trading problem* [58, 59], the *multiple-choice secretary problem* [21, 71] and the *online knapsack problem* [32, 38]. The substantial difference is that in addition to the uncertainty in packet arrival (which is the same in above problems), the

QoS buffer management problem comes with another uncertainty in the network resource supply (that corresponds to the packet departure). This makes the adversary have more flexibility to make worst input by using two sources of uncertainty.

Almost all aforementioned classic problems have timely applications in the recent active research topics. For example, [146] is an example of a natural extension of online knapsack problem in the cloud resource allocation problem. The pricing strategy presented in [146] is close to our profit-based admission control with a social welfare maximization goal. Similarly, [134] represents the first online combinatorial auction designed for the cloud computing paradigm. Both [146] and [134] are the natural extensions of the online knapsack problem. Both works, however, fail to incorporate the latter uncertainty in provisioning of the network resource.

2.3 Summary of Main Results and Adopted Techniques

The following summarizes the main results and the conceptual framework of the solution design. All the introduced algorithms have a common structure which set a threshold value to either admit or reject the packets. We refer to these algorithms as *threshold-based* online algorithms.

▷ In Section 2.5, we introduce a simplified problem that assumes no packets departure, i.e., the admitted packets stay at the buffer permanently. For the simplified problem which theoretically is quite similar to online knapsack problem [32, 38], we design an algorithm to adopt a fixed threshold value for every state of the queue length of the buffer. An arriving packet is admitted only when its value is above or equal to the corresponding threshold. By optimizing the threshold values, we ultimately achieve a closed-form competitive ratio, which is proved to be the optimal among all deterministic online algorithms. While using several tailored techniques suitable for the remaining part of the thesis, our results match to the state-of-the-art result for online knapsack problem [32]. Subsequently, we extend the threshold-based strategy of the simplified problem to the original problem, with packet departure. By optimizing the threshold values, we achieve an online algorithm that attains a better competitive ratio than the existing works [20, 26, 91] and hence is of interest by itself.

▷ In Section 2.6, our goal is to propose a randomized online algorithm with the optimal competitive ratio. Toward this, in Section 2.6.1, we first relax the assumption of the discrete packet admission, i.e., either admit or reject the packet. In the relaxed model, the switching node can admit a fraction of the packet and in turn receive the reward proportionally. Then, we propose a novel strategy that builds a set of virtual sub-queues with unit capacity for each departed packet. By defining virtual sub-queues, we can track the history of packet departure, facilitating to potentially improve the competitiveness of the online algorithm.

By leveraging the idea of sub-queue construction, we then propose an algorithm that maintains a state vector consisting of the queue length of each sub-queue and associates each virtual sub-queue with a threshold-based admission strategy, a scaled version of the one proposed in Section 2.5. Then, the admitted portion of the arriving packet will be properly allocated among sub-queues in a *water-filling* manner. The analysis shows that the proposed algorithm achieves a competitive ratio of $\left[1 + (\ln\tilde{\theta} + 1)\tilde{\theta}/d\right] \cdot (\ln\tilde{\theta} + 1)$, where $\tilde{\theta}$ is the packet value fluctuation ratio and W is a parameter that determines the granularity of the fractional model. With sufficiently large W, the competitive ratio approximates the known lower bound of $\ln\tilde{\theta} + 1$ for deterministic online algorithms [25]. Different from the *Selective Barrier* policy in [25] (with competitive ratio of $O(\ln^2\tilde{\theta}) + \ln\tilde{\theta} + 2$ for small $\tilde{\theta}$) and [26] (with competitive ratio of $e \times \lceil\ln\tilde{\theta}\rceil$), which set a fixed threshold value only based on the state of queue length, the proposed algorithm adopts an independent threshold-based admission strategy for each virtual sub-queue, which corresponds to one unit of buffering budget.

$\triangleright$ In Section 2.6.3, we propose a randomized rounding approach to extend the result of the fractional admission model to the original discrete model. First, we prove that the optimal competitive ratio of any randomized online algorithm for the discrete case is lower bounded by $\ln\tilde{\theta} + 1$, which is also the tight lower bound of the online algorithms for the fractional admission model as well. This result shows that the randomization will not "outperform" the optimal online algorithm for the fractional admission model. Based on this observation, it is natural to seek a randomized scheme to keep the expected queue length equal to that of the fractional case. Our proposed randomized algorithm achieves the same competitive ratio as the optimal online algorithm for the fractional case, meeting the optimal lower bound, $\ln\tilde{\theta} + 1$, thereby it is the optimal online algorithm. The main endeavor is to properly design the admission probabilities based on the actions taken by the optimal online algorithm for the fractional admission model.

2.4 Problem Formulation and Preliminaries

In the networks supporting DiffServ, each packet is associated with a profit-related value according to its importance. We assume each switch node is equipped with a non-preemptive First-In-First-Out (FIFO) queue. With the goal to maximize the cumulative profit, the switch node has to make an instant decision on whether to buffer the incoming packet. In the following, we give a formal formulation for such problem.

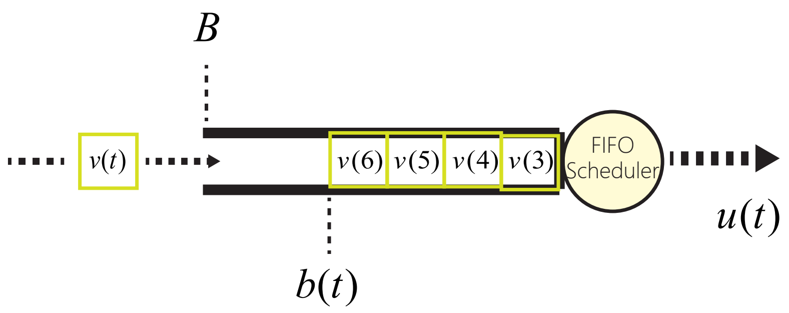


Fig. 2.1: FIFO QoS Buffer Management.

We partition the time horizon into slots according to the arrival time of the packets. Specifically, a time slot begins just before a new packet arrives to the switching node, and ends before the next packet arrival. By this definition, each time slot contains exactly one packet arrival.

Suppose that the size of packets are identical and the buffer can store at most B packets. The number of packet departures at time slot t is denoted by $u(t)$. By $v(t) \in [v_{\min}, v_{\max}]$, we denote the value of the arriving packet at the beginning of t-th time slot, and $v_{\max}$ and $v_{\min}$ denote the maximum and minimum packet values, respectively. Both $v(t)$ and $u(t)$ are exogenous inputs controlled under an adversary strategy. The online algorithm must decide to either admit or reject a packet upon its arrival. We represent the binary decision variable by abusing the notation $a(t) \in \{0, 1\}$, where $a(t) = 1$ represents admission of the packet; 0 otherwise. Finally, $b(t)$ denotes the buffer level, i.e., the number of packets in the buffer, at the end of time slot t, and is expressed by

$$b(t) = [b(t-1) + a(t) - u(t)]^{+},$$

where $[\cdot]^{+}$ denotes projection onto the nonnegative orthant. The number of packets in the buffer must satisfy the buffer capacity constraint, i.e.,

$$b(t-1)+a(t) \le B.$$

The objective is to maximize the profit of the switching node, i.e., the sum of values of admitted packets, over the time horizon, $\mathcal{T} = [1,T]$. Mathematically, the Online Buffer Management (OnBuff) problem is formulated as follows:

$$\begin{aligned}
\textsf{OnBuff} \quad \max \quad & \sum_{t\in\mathcal{T}} v(t)a(t) \\
\text{s.t.} \quad & b(t) = [b(t-1)+a(t)-u(t)]^+, \quad \forall t \in \mathcal{T}, \\
& b(t-1)+a(t) \le B, \quad \forall t \in \mathcal{T}, \\
\text{var.} \quad & a(t) \in \{0,1\}.
\end{aligned} \tag{2.1}$$

In our analysis, we assume that the initial state of the buffer is 0, i.e., $b(0)=0$. In the online context, the exogenous inputs $v(t)$ and $u(t)$, and the ending time T are not known in advance, and we do not rely on any stochastic modeling of the exogenous inputs.

Note that Problem (2.1) can be considered as a natural extension of the *online knapsack* problem[1] [32, 147, 38], and the category of conversion problems in financial markets [122]; some well-known variants are the *time series search* problem [112], the *one-way trading* problem [58, 59] and the *secretary problem* [21, 71]. The exogenous input $v(t)$ in Problem (2.1) is similar to the item values in the online knapsack problem and sequential online price in the conversion problems. However, $u(t)$ is another exogenous input to Problem (2.1) which does not exist in the aforementioned problems. In terms of competitive design, existence of two sets of exogenous inputs, i.e., $v(t)$ and $u(t)$, empowers the adversary to construct worst-case instances in a larger space, potentially resulting in a worse competitive ratio. Hence, online algorithm design becomes more challenging, since the adversary is more powerful. We refer to Section 2.2.3 for detailed discussions regarding similar problems.

In the following, we introduce some definitions and notations which are used for competitive analysis in this chapter. By $b^{\mathfrak{A}}_{\boldsymbol{\omega}}(t)$, we denote the queue length at slot t under $\mathfrak{A}$ and a particular instance $\boldsymbol{\omega}$. Let b denote the maximum queue length that $\mathfrak{A}$ reaches under $\boldsymbol{\omega}$ over the time horizon, i.e., $\max_{t\in\mathcal{T}} b^{\mathfrak{A}}_{\boldsymbol{\omega}}(t) = b$. By this definition, we can partition the universal set of input instances, denoted by Ω, into multiple separate

[1] In the online knapsack problem, the items with different weights and values, i.e., $(\delta(t), v(t))$ arrive online and a feasible solution is any subset $\mathcal{S}$ of items such that $\sum_{t\in\mathcal{S}} \delta(t) < B$, where B is the knapsack capacity. The goal is to maximize the value of selected items, i.e., $\sum_{t\in\mathcal{S}} v(t)$.

Tab. 2.1: Summary of key notations related to the Online QoS Buffer Management problem

Notation	**Description**
t	Index of each time slot
T	The number of time slots, $T \geq 0$
$\mathcal{T}$	Set $\mathcal{T} = \{1, 2, \ldots, T\}$
$v_{\max}$, $v_{\min}$	The maximum and minimum packet values
$v(t)$	Packet value at t, $v_{\min} \leq v(t) \leq v_{\max}$, known for t, unknown for $\tau \in \mathcal{T} : \tau > t$
v_i	Threshold value to admit the i-th packet, $i = 1, 2, \ldots, B$
$\tilde{\theta}$	The value fluctuation ratio, $\tilde{\theta} = v_{\max}/v_{\min}$
$u(t)$	The number of packet departures at time slot t
$b(t)$	The number of packets buffered packets at time slot t
B	The capacity of the buffer equipped in the switch node
$a(t)$	$a(t) \in \{0, 1\}$. The decision variable of the online algorithm at time slot t, 1 represents admission and 0 is rejection

subsets as follows:

$$\Omega = \bigcup_{b \in \{1,2,\ldots,B\}} \Omega_b^{\mathfrak{A}},$$

where

$$\Omega_b^{\mathfrak{A}} \overset{\text{def}}{=} \left\{ \boldsymbol{\omega} \in \Omega : \max_{t \in \mathcal{T}} b_{\boldsymbol{\omega}}^{\mathfrak{A}}(t) = b \right\},$$

represents the set of all input instances that result in the maximum queue length b by executing algorithm $\mathfrak{A}$.

Definition 2.4.1: Define the local competitive ratio $\text{CR}_b(\mathfrak{A})$ under the subset of input instances $\Omega_b^{\mathfrak{A}}$ as

$$\text{CR}_b(\mathfrak{A}) \overset{\text{def}}{=} \max_{\boldsymbol{\omega} \in \Omega_b^{\mathfrak{A}}} \frac{\mathsf{Prof}_{\mathsf{OPT}}(\boldsymbol{\omega})}{\mathsf{Prof}_{\mathfrak{A}}(\boldsymbol{\omega})}.$$

Given Definition 3.5.1, we redefine $\text{CR}(\mathfrak{A})$ as

$$\text{CR}(\mathfrak{A}) = \max_{b \in \{1,2,\ldots,B\}} \text{CR}_b(\mathfrak{A}).$$

We also use the following expressions for an input instance $\boldsymbol{\omega} = [(v_1, u_1), (v_2, u_2), \ldots, (v_T, u_T)]$:

1. The notation $\times n$ is used to represent repeated input segments. Specifically, $(v, u) \times n$ (or $\boldsymbol{\omega} \times n$) signifies the input tuple (v, u) (or input segment $\boldsymbol{\omega}$) will repeatedly appear n times in the subsequent time slots.

2. The concatenation of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ is denoted by $\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2$ and

expressed by

$$\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 = [\omega_1(1), \omega_1(2), \ldots, \omega_1(T_1), \omega_2(1), \omega_2(2), \ldots, \omega_2(T_2)] \,.$$

2.5 A Simplified Online Problem and the Threshold-Based Algorithm

The QoS buffer management problem involves two exogenous inputs $v(t)$, $u(t)$. To analyze Problem (2.1), we first investigate a simplified version of the original problem by setting the exogenous input $u(t)$ to be zero in the entire time horizon. In other words, we assume that there is no packet departure during the time horizon, i.e., $u(t) = 0$, for $t = 1, 2, \ldots, T$. In this way, the original problem reduces to the one-way-trading-like problem which has been introduced in the introduction section.

To design an online solution for the simplified scenario, we follow a well-established design approach for these problems and explore a class of deterministic online algorithms which are *threshold-based*. The main idea of the threshold-based strategies is that for any state of the buffer, there is a fixed threshold and the incoming packet will be admitted if its value is greater than or equal to the corresponding threshold. The analysis is then focused on demonstrating that by optimizing the threshold values, the threshold-based strategy can achieve the optimal competitive ratio among all deterministic online algorithms for the simplified problem.

2.5.1 Threshold-Based Policy Design for a Simplified Problem

By simplifying Problem (2.1) and setting $u(t) = 0, \ t \in \mathcal{T}$, we formulate the OneTrad problem introduced in the introduction section.

For Problem (1.1), we device a *threshold-based* online algorithm called sBuffAlg, which is defined by means of a series of non-decreasing threshold values $v_i, \ i = 1, 2, \ldots, B$. For convenience, we categorize those v_is of the same value into a single *step*. The goal is to design the optimal values of v_i as a function of queue length to specify the minimum value to admit the i-th packet.

Recall that $\Omega_b^{\mathsf{sBuffAlg}}$ is the subset of input instances that result in the maximum queue length b upon executing the online algorithm sBuffAlg. The following lemma characterizes a critical property for the worst instance in $\Omega_b^{\mathsf{sBuffAlg}}$.

Lemma 2.5.1: Assume $\text{CR}_b(\mathsf{sBuffAlg}) > 1$ and $\boldsymbol{\omega} = [v(t)]_{t \in \mathcal{T}}$ is a worst instance in $\Omega_b^{\mathsf{sBuffAlg}}$. Then at any slot t when sBuffAlg and OPT buffer packet simultaneously, $v(t)$ is exactly equal to the threshold of sBuffAlg.

Proof of Lemma 2.5.1. We prove Lemma 2.5.1 by contradiction.

Assume that a worst instance is $[v(t)]_{t\in\mathcal{T}}$. Then suppose there exists a time slot t such that sBuffAlg and OPT buffer packet simultaneously and $v(t)$ is larger than the threshold value of sBuffAlg, i.e., $v(t) > v_i$, $i = b(t-1)+1$. Now we present the following input instance to sBuffAlg:

$$[v(1),\dots,v(t-1)] + [v_i] + [v(t+1),\dots,v(T)].$$

Under the new instance, the total number of buffered packets and buffering time slots of sBuffAlg keep unchanged. The profit earned by sBuffAlg decreases by $v(t) - v_i$, while profit decrement of OPT is less than or equal to $v(t) - v_i$. The local competitive ratio $\text{CR}_b(\mathsf{sBuffAlg})$ is larger than 1, so the above instance results in a larger competitive ratio. This contradicts the assumption that $\boldsymbol{\omega}$ is the worst instance in $\Omega_b^{\mathsf{sBuffAlg}}$. We complete the proof. □

The next lemma characterizes upper bounds for local competitive ratios of sBuffAlg.

Lemma 2.5.2: Assume the *threshold-based* algorithm sBuffAlg is defined by non-decreasing threshold values v_i that satisfies the condition $v_1 = v_{\min}$. If the length of the first step is λ, then

1. for the subset $\Omega_b^{\mathsf{sBuffAlg}}$ where $b < \lambda$, the local competitive ratio is 1, and
2. for $b \geq \lambda$, the local competitive ratio within the subset $\Omega_b^{\mathsf{sBuffAlg}}$ satisfies
$$\text{CR}_b(\mathsf{sBuffAlg}) \leq \frac{v_{b+1}B}{\sum\limits_{i=1}^{b} v_i}.^{2} \qquad (2.2)$$

Proof of Lemma 2.5.2. For Lemma 2.5.2, we have the following analysis:

1. For $b < \lambda$, the threshold value of sBuffAlg is always equal to m over $\mathcal{T}$. That means sBuffAlg buffers all packets and obtain the same profit as OPT. In this case, the local competitive ratio is 1.
2. Suppose $b \geq \lambda$. If $\text{CR}_b(\mathsf{sBuffAlg}) = 1$, the case is trivial and Equation (2.2) definitely holds. We only consider the case that $\text{CR}_b(\mathsf{sBuffAlg}) > 1$. For an instance within $\Omega_b^{\mathsf{sBuffAlg}}$, the threshold values of sBuffAlg are always less than or equal to

[2] For consistence, we define $v_{B+1} = v_{\max}$.

v_{b+1} $(v_{b+1} > v_{\min})$, since the maximum queue length is b and v_i are non-decreasing. Assume $\boldsymbol{\omega}$ is a worst instance lying in $\Omega_b^{\textsf{sBuffAlg}}$. Then, according to Lemma 2.5.1, a packet value under $\boldsymbol{\omega}$ will be exactly equal to the threshold value of sBuffAlg when OPT and sBuffAlg buffer this packet simultaneously. Moreover, when only OPT buffers a packet, it is obvious that the packet value is less than the threshold value of sBuffAlg. That means, under $\boldsymbol{\omega}$, all the packets buffered by OPT are of values less than or equal to v_{b+1}. Thus, the profit earned by OPT under $\boldsymbol{\omega}$ is at most $v_{b+1}B$. Meanwhile, the minimum profit earned by sBuffAlg is at least $\sum\limits_{i=1}^{b} v_i$ due to its threshold-based admission strategy. Thus, we have proved that the largest profit ratio within subset $\Omega_b^{\textsf{sBuffAlg}}$ is at most $v_{b+1}B/\sum\limits_{i=1}^{b} v_i$.

This completes the proof. □

For $b > \lambda$, consider the following instance with increasing packet values:

$$[v_1, v_2, \ldots, v_b, (v_{b+1} - \delta) \times B],$$

under which the profit earned by OPT is $(v_{b+1} - \delta)B$ and the profit earned by sBuffAlg is $\sum_{i=1}^{b} v_i$. The worst-case profit ratio shown in Equation (2.2) can be realized by the above instance with $\delta \to 0$, i.e.,

$$\textsc{cr}_b(\textsf{sBuffAlg}) \geq \lim_{\delta \to 0} \frac{(v_{b+1} - \delta)B}{\sum\limits_{i=1}^{b} v_i}.$$

Combining with Lemma 2.5.2, we get

$$\textsc{cr}_b(\textsf{sBuffAlg}) = \frac{v_{b+1}B}{\sum\limits_{i=1}^{b} v_i}, \text{ for } b \geq \lambda. \tag{2.3}$$

According to Lemma 2.5.2 and Equation (2.3), the worst case occurs among subsets $\Omega_b^{\textsf{sBuffAlg}}$, $b \geq \lambda$, so the competitive ratio of sBuffAlg is

$$\textsc{cr}(\textsf{sBuffAlg}) = \max_{b=\lambda,\lambda+1,\ldots,B} \frac{v_{b+1}B}{\sum\limits_{i=1}^{b} v_i}.$$

Conditioning on the length of the first step λ, the minimum competitive ratio, which is denoted by $\textsc{cr}(\mathsf{sBuffAlg}|\lambda)$ can be obtained by optimizing the threshold values $v_{\lambda+1}, v_{\lambda+2}, \dots, v_B$:

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & y \geq (v_{b+1}B)/\sum_{i=1}^{b} v_i, \quad b = \lambda, \lambda+1, \dots, B. \\ \text{vars.} \quad & y,\ v_{\min} \leq v_b \leq v_{\max}, \quad b = \lambda+1, \lambda+2, \dots, B. \end{aligned} \tag{2.4}$$

The following lemma gives a necessary condition for $\textsc{cr}(\mathsf{sBuffAlg}|\lambda)$ to achieve its minimum value.

Lemma 2.5.3: $\textsc{cr}(\mathsf{sBuffAlg}|\lambda)$ achieves its minimum value only if the following expression holds:

$$\frac{v_{\lambda+1}B}{v_{\min}l} = \frac{v_{\lambda+2}B}{v_{\min}\lambda + v_{\lambda+1}} = \cdots = \frac{v_{\max}B}{v_{\min}l + \sum\limits_{i=\lambda+1}^{B} v_i}. \tag{2.5}$$

In the next step, we show that the minimum value of problem (2.4) is at least $\ln\tilde{\theta} + 1$, which provides a lower bound for the competitive ratio of $\mathsf{sBuffAlg}$. By Equation (2.5), we can represent B as

$$B = \left(\frac{v_{\lambda+1}}{v_{\min}} + \frac{v_{\lambda+2} - v_{\lambda+1}}{v_{\lambda+1}} + \frac{v_{\lambda+3} - v_{\lambda+2}}{v_{\lambda+2}} + \cdots + \frac{v_{\max} - v_B}{v_B}\right)\frac{v_{\min}}{v_{\lambda+1}}l.$$

Thus, the competitive ratio of $\mathsf{sBuffAlg}$ with the first step length being λ can be expressed as

$$\textsc{cr}(\mathsf{sBuffAlg}|\lambda) = 1 + \frac{v_{\lambda+1} - v_{\min}}{v_{\min}} + \frac{v_{\lambda+2} - v_{\lambda+1}}{v_{\lambda+1}} + \cdots + \frac{v_{\max} - v_B}{v_B}.$$

Let

$$\Sigma = \frac{v_{\lambda+1} - v_{\min}}{v_{\min}} + \frac{v_{\lambda+2} - v_{\lambda+1}}{v_{\lambda+1}} + \frac{v_{\lambda+3} - v_{\lambda+2}}{v_{\lambda+2}} + \cdots + \frac{v_{\max} - v_B}{v_B}.$$

Since the value of Σ is equal to the size of the shaded area in Figure 2.2, we have $\textsc{cr}(\mathsf{sBuffAlg}|\lambda) > 1 + \ln(v_{\max}/v_{\min}) = 1 + \ln\tilde{\theta}$.

According to the results in Lemma 2.5.3, the following theorem follows.

Theorem 2.5.1: Given a non-decreasing threshold values with the length of the first step being λ, the optimal competitive ratio that can be achieved by $\mathsf{sBuffAlg}$, denoted by γ_λ satisfies

$$\left(\frac{\gamma_\lambda + B}{B}\right)^{B-\lambda+1} - \left(\frac{\gamma_\lambda + B}{B}\right)^{B-\lambda} - \frac{\tilde{\theta}}{l} = 0. \tag{2.6}$$

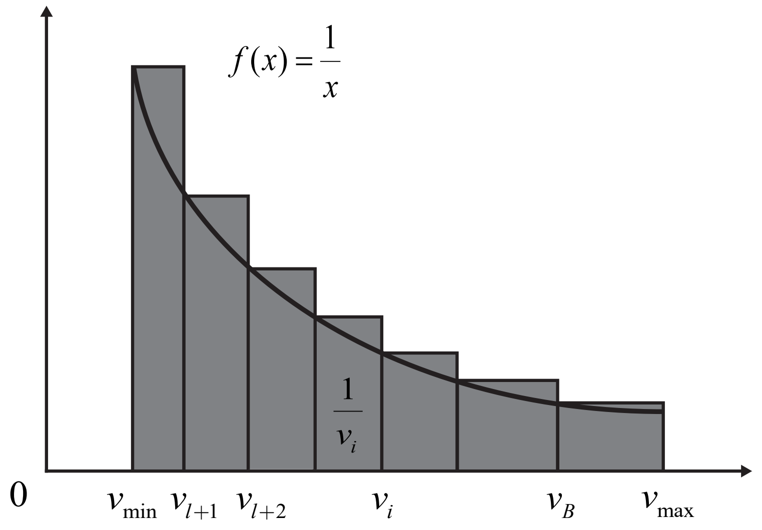


Fig. 2.2: Visualized expression of Σ.

Assuming λ^* is the optimal solution to minimize the above function and γ^* is the corresponding optimal value, the optimal threshold values are

$$v_{\lambda^*+1} = \frac{\gamma^* v_{\min} \lambda^*}{B},$$
$$v_{\lambda^*+i+1} = v_{\lambda^*+1} \left(\frac{v_{\max}}{v_{\lambda^*+1}} \right)^{\frac{i}{B-\lambda^*}}, \quad \text{for } i = 1, 2, \ldots, B - \lambda^*.$$

Proof of Theorem 2.5.1. According to Lemma 2.5.3, we have

$$v_{\lambda+1} = \frac{\gamma_\lambda v_{\min} \lambda}{B}.$$

Moreover,

$$\frac{v_{\lambda+2} - v_{\lambda+1}}{v_{\lambda+1}} = \frac{v_{\lambda+3} - v_{\lambda+2}}{v_{\lambda+2}} = \cdots = \frac{v_{B+1} - v_B}{v_B},$$

hence,

$$\frac{v_{i+1}}{v_i} = \left(\frac{v_{\max}}{v_{\lambda+1}} \right)^{\frac{1}{B-\lambda}}, \; i = \lambda + 1, \lambda + 2, \ldots, B.$$

The above equations give the threshold values. According to Lemma 2.5.3, we have that

$$\frac{v_{i+1} - v_i}{v_i} \frac{v_{\min}}{v_{\lambda+1}} \lambda = 1, \; i = \lambda + 1, \lambda + 2, \ldots, B.$$

Thus, combining the above equations, we have

$$\frac{v_{\lambda+1}}{v_{\min}} = \left[\left(\frac{v_{\max}}{v_{\lambda+1}}\right)^{\frac{1}{B-\lambda}} - 1\right]\lambda.$$

Replacing $v_{\lambda+1}$ with $\gamma_\lambda v_{\min}\lambda/B$ yields Equation (2.6). $\square$

We have introduced a threshold-based strategy sBuffAlg with a set of non-decreasing threshold values. The next theorem shows that sBuffAlg is optimal among all deterministic online algorithms.

Theorem 2.5.2: The *threshold-based* algorithm sBuffAlg is optimal among all deterministic online algorithms.

Proof of 2.5.2. Let $\mathfrak{A}$ be any deterministic online algorithm. In order to prove this theorem, we shall show that $\mathfrak{A}$ cannot achieve a lower competitive ratio than that of sBuffAlg. Toward this, consider the following instance with increasing packet values

$$[\underbrace{(v_{\min}) \times B}_{\text{first round}}, \underbrace{(v_{\min}+\delta) \times B}_{\text{second round}}, \ldots, \underbrace{(v_{\min}+(n-1)\delta) \times B}_{\text{penultimate round}}, \underbrace{(v_{\min}+n\delta) \times B}_{\text{last round}}],$$

where $\delta = \frac{v_{\max}-v_{\min}}{n}$. By $\bar{v}_i$, we denote the packet value of the i-th packet that $\mathfrak{A}$ admits under the above instance. In the first round, $\mathfrak{A}$ is presented packets of packet value $v_{\min}$ for B times. If $\mathfrak{A}$ has never accepted any packet, the adversary can stop the process right after the first round and construct an instance with the profit ratio being infinite. Assume $\mathfrak{A}$ accepts $\bar{\lambda} > 0$ packets in the first round. Let $\bar{B}$ be the total number of packets buffered during the above process. Then, the profit ratio between OPT and $\mathfrak{A}$ when the adversary presents the entire instance to $\mathfrak{A}$ is $v_{\max}B/\sum_{i=1}^{\bar{B}}\bar{v}_i$.

Moreover, we assume during the above process, b-th ($b \geq \bar{\lambda}$) and $(b+1)$-th packets are admitted in the j-th and j'-th round, respectively. If $j = j'$, the adversary can stop right after $\mathfrak{A}$ admits the b-th packet. In this case, the profit obtained by $\mathfrak{A}$ will be $\sum_{i=1}^{b}\bar{v}_i$, and that of OPT will not be less than $(\bar{v}_{b+1}-\delta)B$ (buffering packets during the $(j-1)$-th round). In this case, the profit ratio can be $(\bar{v}_{b+1}-\delta)B/\sum_{i=1}^{b}\bar{v}_i$. When $j \neq j'$, the adversary can stop the input instance right after the $(j'-1)$-th round and get a profit ratio of $(\bar{v}_{b+1}-\delta)B/\sum_{i=1}^{b}\bar{v}_i$. Thus, for any $\bar{\lambda} \leq b \leq \bar{B}-1$, we can always construct an instance under which the profit ratio between OPT and $\mathfrak{A}$ is at least $(\bar{v}_{b+1}-\delta)B/\sum_{i=1}^{b}\bar{v}_i$.

With $\delta \to 0$, we have

$$\begin{aligned}
\textsc{cr}(\mathfrak{A}) &\geq \max_{\lambda \leq b \leq \bar{B}} \left\{ \lim_{\delta \to 0} \frac{(\bar{v}_{b+1} - \delta)B}{\sum_{i=1}^{b} \bar{v}_i}, \frac{v_{\max} B}{\sum_{i=1}^{\bar{B}} \bar{v}_i} \right\} \\
&\geq \max_{\lambda \leq b \leq \bar{B}} \left\{ \frac{\bar{v}_{b+1} B}{\sum_{i=1}^{b} \bar{v}_i}, \frac{v_{\max} B}{\sum_{i=1}^{\bar{B}} \bar{v}_i} \right\} \\
&\geq \max_{\lambda \leq b \leq B} \frac{\bar{v}_{b+1} B}{\sum_{i=1}^{b} \bar{v}_i} \\
&\geq \min_{\lambda, v_i} \max_{\lambda \leq b \leq B} \frac{v_{b+1} B}{\sum_{i=1}^{b} v_i} = \textsc{cr}(\mathsf{sBuffAlg}),
\end{aligned}$$

where in the penultimate inequality, we set $\bar{v}_i = \bar{v}_{\bar{B}}$ for $\bar{B} < i \leq B$ and $\bar{v}_{B+1} = v_{\max}$.

This completes the proof. □

In [147], an online optimal solution to the online knapsack problem is derived by assuming that the item size is much smaller than the capacity of the knapsack. Using the solution in [147], a similar result can be obtained to problem (1.1) by allowing the number of packets to take fractional values. The following corollary restates this result simply for the convenience of the readers since it is required in our subsequent analysis for the general problem.

Corollary 2.5.1: In the special case of allowing the number of packets to take fractional values, the optimal competitive ratio of Problem (1.1) is

$$\textsc{cr}(\mathsf{sBuffAlg}) = \ln \tilde{\theta} + 1,$$

and the optimal threshold function $g(b) : [0, B] \to [v_{\min}, v_{\max}]$ (which is a continuous analogue of the discrete threshold values) is

$$g(b) = \begin{cases} v_{\min}, & b \leq \frac{B}{\ln \tilde{\theta} + 1}, \\ v_{\min} e^{\left(\ln \tilde{\theta} + 1\right) \frac{b}{B} - 1}, & \text{otherwise}. \end{cases} \tag{2.7}$$

2.5.2 A Threshold-Based Strategy for the General Problem

In this section, we apply the *threshold-based* strategy to the original problem which allows packet departure, i.e., $u(t) \geq 0, t \in \mathcal{T}$. In order to analyze the performance of the *threshold-based* strategy, we can divide the investigated time period into multiple *cycles*.

Definition 2.5.1: Given a *threshold-based* online algorithm $\mathsf{thBuffAlg}$, a *cycle* is defined to be the time interval beginning and ending whenever the buffer under thBuffAlg becomes empty. Specifically, let $0 \leq t_1 < t_2 < \cdots < t_n \leq T$ denote the time slots when the queue length under thBuffAlg goes to 0, then the time interval $[t_i + 1, t_{i+1}]$, $i \in \{1, 2, \ldots, n-1\}$ forms a *cycle*.

The following observation implies that the analysis for the competitive ratio can be conducted within a *cycle*.

Lemma 2.5.4: Let $\boldsymbol{\omega}$ be a worst instance and $\mathcal{C}$ be any cycle realized by thBuffAlg under $\boldsymbol{\omega}$, then the profit ratio between OPT and thBuffAlg during $\mathcal{C}$ is equal to the competitive ratio of thBuffAlg.

Proof of Lemma 2.5.4. Assume under the worst instance $\boldsymbol{\omega} = [(v(\tau), u(\tau))]_{\tau \in \mathcal{T}}$, there is a cycle $\mathcal{C} = [s+1, t]$ during which the maximum profit ratio is smaller than CR (thBuffAlg). At time slots s and t, the buffer under thBuffAlg is emptied. We just increase $u(s)$ and $u(t)$ by a large number δ such that the buffer under OPT also becomes empty. This operation never changes subsequent operations of thBuffAlg, as well as the obtained profit. Also, the profit obtained by OPT is unchanged since $\boldsymbol{\omega}$ has resulted in the worst-case profit ratio. Then, we can "remove" the input segment over $\mathcal{C}$ and present the instance $[(v(\tau), u(\tau))]_{\tau=1:s-1} + [(v(s), u(s) + \delta)] + [(v(\tau), u(\tau))]_{\tau=t+1:T}$ to thBuffAlg and get a larger profit ratio, contradicting the assumption on the worst instance. Similarly, if the profit ratio during one cycle is larger than the competitive ratio, presenting the following input instance to thBuffAlg yields the increase of the profit ratio:

$$[(v(\tau), u(\tau))]_{\tau=1:s-1} + [(v(s), u(s)+\lambda)] + \boldsymbol{\omega}_{\mathcal{C}} \times 2 + [(v(\tau), u(\tau))]_{\tau=t+1:T},$$

where $\boldsymbol{\omega}_{\mathcal{C}} = [(v(\tau), u(\tau))]_{\tau=s+1:t-1} + [(v(t), u(t) + \delta)]$. This also contradicts the assumption on the worst instance. $\square$

Based on the above lemma, our analysis on the competitive ratio of thBuffAlg can be reduced to instances which only contain one cycle. By $\Omega_b^{\text{thBuffAlg}}$, we denote the subset of single-cycle input instances with the maximum queue length being b. The following lemma characterizes the local competitive ratio within such a subset.

Lemma 2.5.5: Assume the *threshold-based* online algorithm thBuffAlg is equipped with a series of non-decreasing threshold values v_i satisfying $v_1 = v_{\min}$. If the length of the first step is λ, we have:

1. For the subset $\Omega_b^{\text{thBuffAlg}}$ where $b < \lambda$, the local competitive ratio is 1.

2. For $b \geq \lambda$, the local competitive ratio within the subset $\Omega_b^{\mathsf{thBuffAlg}}$ is

$$\mathrm{CR}_b(\mathsf{thBuffAlg}) \leq \frac{v_{b+1}B + \sum_{i=l+1}^{b} v_i}{\sum_{i=1}^{b} v_i}. \tag{2.8}$$

Inequality (2.8) holds with equality when the local competitive ratio within $\Omega_b^{\mathsf{thBuffAlg}}$ satisfies $\mathrm{CR}_b(\mathsf{thBuffAlg}) \geq \mathrm{CR}_{b'}(\mathsf{thBuffAlg})$ for all $b' < b$.

By Lemma 2.5.5, we have

$$\begin{aligned}\mathrm{CR}(\mathsf{thBuffAlg}) &= \max_{b=\lambda,\lambda+1,\ldots,B} \mathrm{CR}_b(\mathsf{thBuffAlg}) \\ &= \max_{b=\lambda,\lambda+1,\ldots,B} \frac{v_{b+1}B + \sum_{i=\lambda+1}^{b} v_i}{\sum_{i=1}^{b} v_i}.\end{aligned}$$

Similar to Lemma 2.5.3, the next lemma explains how to optimize the threshold values v_i.

Lemma 2.5.6: If the length of the first step l is determined, $\mathrm{CR}(\mathsf{thBuffAlg}|\lambda)$ maximizes if and only if the following equalities hold:

$$\frac{v_{\lambda+1}B}{v_{\min}\lambda} = \frac{v_{\lambda+2}B + v_{\lambda+1}}{v_{\min}\lambda + v_{\lambda+1}} = \cdots = \frac{v_B B + \sum_{i=\lambda+1}^{B-1} v_i}{v_{\min}\lambda + \sum_{i=\lambda+1}^{B-1} v_i} = \frac{v_{\max}B + \sum_{i=\lambda+1}^{B} v_i}{v_{\min}\lambda + \sum_{i=\lambda+1}^{B} v_i}.$$

Theorem 2.5.3: Assuming λ is given, the minimum competitive ratio of $\mathsf{thBuffAlg}$, denoted by γ_λ satisfies

$$\frac{B\tilde{\theta}}{\gamma_\lambda \lambda} = \left(\frac{\gamma_\lambda(B-\lambda)}{B^2} + 1\right)^{B-\lambda}. \tag{2.9}$$

Assuming λ^* is the optimal solution to minimize the above function and γ^* is the corresponding optimal value, then the optimal threshold values are

$$\begin{aligned}&v_{\lambda^*+1} = \tfrac{\gamma^* v_{\min}\lambda^*}{B}, \\ &v_{\lambda^*+i+1} = v_{\lambda^*+1}\left(\tfrac{v_{\max}}{v_{\lambda^*+1}}\right)^{\frac{i}{B-\lambda^*}}, \quad i = 1, 2, \ldots, B-\lambda^*.\end{aligned}$$

Proof of Theorem 2.5.3. By induction, we can derive that

$$\frac{v_{\min}}{v_{\lambda+1}}\frac{B\lambda}{B-\lambda}\frac{v_{i+1}-v_i}{v_i}=1,\ i=\lambda+1,\lambda+2,\ldots,B. \tag{2.10}$$

That implies

$$\frac{v_{\lambda+2}-v_{\lambda+1}}{v_{\lambda+1}}=\frac{v_{\lambda+3}-v_{\lambda+2}}{v_{\lambda+2}}=\cdots=\frac{v_{B+1}-v_B}{v_B},$$

so

$$\frac{v_{i+1}}{v_i}=\left(\frac{v_{\max}}{v_{\lambda+1}}\right)^{\frac{1}{B-\lambda}},\quad i=\lambda+1,\lambda+2,\ldots,B. \tag{2.11}$$

Moreover, according to Lemma 2.5.6, we have

$$v_{\lambda+1}=\frac{\gamma_\lambda v_{\min}\lambda}{B}.$$

Combining the above equation and (2.11), we can obtain the threshold values shown in the theorem.

According to Equation (2.10), we have

$$\frac{v_{\lambda+1}}{v_{\min}}=\left[\left(\frac{v_{\max}}{v_{\lambda+1}}\right)^{\frac{1}{B-\lambda}}-1\right]\frac{B\lambda}{B-\lambda}.$$

Replacing $v_{\lambda+1}$ with $\gamma_\lambda v_{\min}\lambda/B$ yields Equation (2.9). □

Remark 2.5.1: By Equation (2.9), one can conclude that when B is large enough, the competitive ratio approximates $\frac{(2+\ln\tilde{\theta})+\sqrt{\ln^2\tilde{\theta}+4\ln\tilde{\theta}}}{2}$, which is superior to the existing results ($e\lceil\ln\tilde{\theta}\rceil$ in [26] and $\ln\tilde{\theta}+2+O(\ln^2\tilde{\theta}/B)$ in [25]).

2.6 Optimal Randomized Online Algorithm

In the discrete case, the deterministic online algorithm can only admit or reject an incoming packet. It is beneficial for online algorithms to buffer a fraction of the packet if this is a viable option. Motivated by this observation, we focus on finding an optimal online algorithm for the fractional admission model. In this section, by considering fractional model, we design a novel online strategy and show it achieves the lower bound for competitive ratio of $\ln\tilde{\theta}+1$. Then, in Section 2.6.3, we extend the result into the original discrete setting.

2.6.1 Optimal Online Algorithm for the Fractional Admission Model

In Section 2.5.1, we devised an algorithm for a simplified online problem without packet departure over the time horizon. We showed that for the discrete and fractional cases, the optimal competitive ratio can be obtained by a threshold-based algorithm, which maintains a fixed threshold value for each state of the queue length. For the general case with packet departure, the available space for buffering packets changes over time. Thus, the threshold-based strategy whose threshold only depends on the queue length may yield a suboptimal competitive ratio. In this section, we aim to find the optimal competitive ratio for the general case. Toward this goal, we propose a novel online algorithm fBuffAlg. In a nutshell, we introduce virtual sub-queues which track the history of packet departures and maintain a threshold-based strategy for each virtual sub-queue. The details of fBuffAlg are as follows.

State of fBuffAlg

Without loss of generality, we assume the initial state of the queue length is 0. fBuffAlg initially sets B empty sub-queues, each of which with capacity 1. This corresponds to the initial buffer budget of B. Note that packet departure may or may not occur at any time slot. Hence, let t_i, $i = 1, 2, \dots, h$ denote the time slots that at least one packet departure occurs. If there are $u(t_i)$ packets departure in slot t_i, then $u(t_i)$ virtual sub-queues, each with capacity 1, will be created by the end of slot t_i. When the original queue length $b(t)$ becomes 0, the virtual sub-queues created due to packet departures is deleted and the queue lengths of the initial B sub-queues are set to 0. Let W be a sufficiently large integer value. A packet is conceptually divided into W equal *sub-packets*. We denote the discretized queue length of the i-th sub-queue by $\bar{b}_i(t)$ which can only take integer values between 0 and W. Using the above setups, we define fBuffAlg algorithm by extending the original system state $b(t)$ to a state vector $\bar{\boldsymbol{b}}(t) = [\bar{b}_1(t), \bar{b}_2(t), \dots, \bar{b}_N(t)]$, where N is the number of virtual sub-queues at time slot t. It is easy to see that $b(t)$ equals to $\sum_{i=1}^{N} \bar{b}_i(t) - N + B$. Figure 2.3 depicts the state of fBuffAlg at time slot t.

Admission Policy of fBuffAlg

In fBuffAlg, admission policy determines the fraction of the packet to be admitted, i.e., the number of sub-packets in $[0, W]$ that must be admitted. The number of admitted sub-packet depends on the system state $\bar{\boldsymbol{b}}(t)$. Toward this, for each sub-queue i, fBuffAlg maintains a fixed threshold

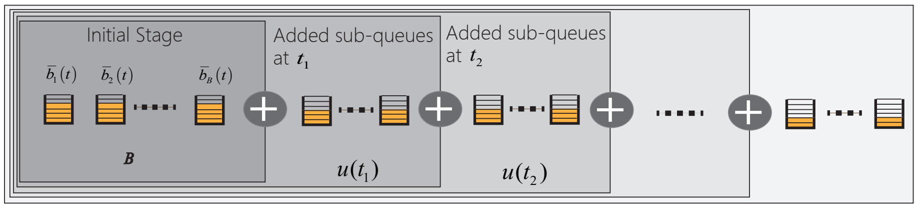


Fig. 2.3: Description of the state of the fBuffAlg algorithm at time slot t.

function $g_W(\bar{b}_i)$ that can be used to determine the number of admitted sub-packets of an arriving packet. The function is defined by

$$g_W(\bar{b}_i) = \begin{cases} v_{\min}, & \frac{\bar{b}_i}{d} \leq \frac{1}{\ln\tilde{\theta}+1}, \\ v_{\min} e^{\left(\ln\tilde{\theta}+1\right)\frac{\bar{b}_i}{W}-1}, & \text{otherwise.} \end{cases} \tag{2.12}$$

Equation (2.12) is a scaled version of (2.7). Based on Equation (2.12), the number of sub-packets that the i-th sub-queue can admit, denoted by $\bar{a}_i(t)$ is determined by

$$\begin{aligned} \bar{a}_i(t) = \quad & \max \quad \epsilon \\ & \text{s.t.} \quad g_W(\bar{b}_i(t-1)+\epsilon) \leq p(t), \\ & \text{var.} \quad \epsilon \in \mathbb{N}. \end{aligned}$$

By the above admission policy, the value of admitted sub-packets is guaranteed to be larger than the threshold of each sub-queue at any time slot. Then, the total number of aggregated sub-packets is simply the aggregation of sub-packets in each sub-queue truncated by W, i.e.,

$$\bar{a}(t) = \min\left\{\sum_{i=1}^{n} \bar{a}_i(t), W\right\}.$$

The value of $\bar{a}(t)$ is the admitted amount of an arriving packet by fBuffAlg under state $\bar{\boldsymbol{b}}(t-1)$ and packet value $v(t)$. We use $\bar{g}_W(\bar{\boldsymbol{b}})$ to denote the minimum threshold for fBuffAlg to admit portion of packet when the algorithm state is $\bar{\boldsymbol{b}}$, i.e.,

$$\bar{g}_W(\bar{\boldsymbol{b}}) \overset{\text{def}}{=} \min_{i=1,2,\cdots,N} g_W(\bar{b}_i+1).$$

$\bar{g}_W(\bar{\boldsymbol{b}})$ only depends on the algorithm state $\bar{\boldsymbol{b}}$.

Allocation Policy of fBuffAlg

Given that the number of sub-packets to be buffered is $\bar{a}(t)$, fBuffAlg allocates these sub-packets among the N sub-queues in a *water-filling* way (as shown in Figure 2.4). In this way, the value of an allocated sub-packet is always larger than the threshold value of the sub-queue. Specifically, the allocation algorithm is as follows.

Algorithm 2.1 Allocation Policy of fBuffAlg (Water-filling policy)

```
1  for $i = 1$ to $\bar{a}(t)$
2      $\mathcal{I} \leftarrow$ The sub-queues with smallest queue length
3      Allocate $i$-th sub-packet to the sub-queue with the smallest index in $\mathcal{I}$, and increase its
       queue length by 1.
```

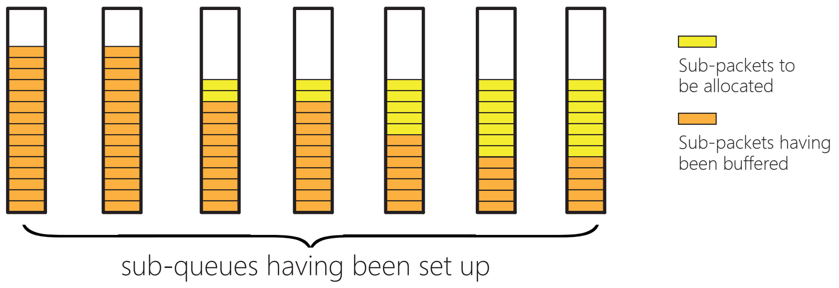

Sub-packets to be allocated
Sub-packets having been buffered
sub-queues having been set up

Fig. 2.4: Allocate admitted sub-packets in a *water-filling* manner.

2.6.2 *Performance Analysis of fBuffAlg*

When a packet departs, a unit of buffer space is released and the available buffering budget increases accordingly. The conceptual idea of fBuffAlg is to build a virtual sub-queue for each unit of released buffering space. An arriving packet is broken into multiple sub-packets and admitted to the original buffer and sub-queues under a thresholding policy such that the packet value of a buffered sub-packet is not less than the threshold. In order to analyze the competitive ratio of fBuffAlg, we face fBuffAlg with a more "powerful" adversary that can transmit a portion of packet aiming to construct a worse instance. Let $w(t),\ 1 \leq w(t) \leq W$ denotes the number of sub-packets transmitted to the switch node at time slot t. Under the powerful adversary, an input instance has the following extended form

$$[\bar{\omega}(t) = (v(t), u(t), w(t))]_{t\in\mathcal{T}},$$

where $v(t)$ and $u(t)$ are as the previous exogenous inputs. Note that when $w(t) = W,\ t \in \mathcal{T}$, the input instance will be reduced to the previous one defined in Section 2.4. Thus, the competitive ratio of fBuffAlg when faced with the powerful adversary characterizes an upper bound for the competitive ratio of fBuffAlg (under a standard adversary defined in Section 2.4).

By analyzing the admission and allocation policies of fBuffAlg, we observe the following two critical properties:

1. When queue lengths are equal, fBuffAlg will allocate the sub-packet to the sub-queue with smaller index. Thus, at any slot t, the following inequalities always hold:
$$\bar{b}_1(t) \geq \bar{b}_2(t) \geq \cdots \geq \bar{b}_N(t).$$
2. When a sub-packet is allocated to a sub-queue, the packet value is always larger than or equal to the threshold value of the sub-queue.

Under the powerful adversary, the following Lemma characterizes a critical property of the worst instance for fBuffAlg.

Lemma 2.6.1: Assume $\bar{\boldsymbol{\omega}} = [\bar{\omega}(t) = (v(t), u(t), w(t))]_{t\in\mathcal{T}}$ is a worst instance for fBuffAlg under the powerful adversary. Then, at any time slot t when fBuffAlg buffers portion of packet, $v(t)$ is exactly equal to the threshold of fBuffAlg, i.e.,
$$v(t) = \bar{g}_W(\bar{\boldsymbol{b}}(t-1)).$$

Proof of Lemma 2.6.1. Assume during the worst instance, there exists a time slot that $v(t)$ is larger than the threshold value $\bar{g}_W(\bar{\boldsymbol{b}}(t-1))$ and $\bar{a}(t)$ sub-packets are admitted by fBuffAlg. We first present the instance segment $[(v(1), u(1), w(1)), (v(2), u(2), w(2)), \ldots, (v(t-1), u(t-1), w(t-1))]$ to fBuffAlg. After that, we further present the instance segment $[(\bar{g}_W(\bar{\boldsymbol{b}}(t-1)), 0, 1), (v(t), 0, \bar{a}(t)-1)]$ to fBuffAlg, and the number of buffered sub-packets is equal to $\bar{a}(t)$. In this way, the profit earned from the buffered $\bar{a}(t)$ sub-packets decreases by $[v(t) - \bar{g}_W(\bar{\boldsymbol{b}}(t-1))]/W$. After that, we present the remaining instance segment $[(v(t+1), u(t+1), w(t+1)), (v(t+2), u(t+2), w(t+2)), \ldots, (v(T), u(T), w(T))]$ to fBuffAlg. Under the above instance, the profit earned by OPT decreases by at most $[v(t) - \bar{g}_W(\bar{\boldsymbol{b}}(t-1))]/W$, which is also the decrement amount of fBuffAlg. In this way, we construct a new instance that results in a larger competitive ratio, contradicting the assumption on the worst instance. □

The next Lemma implies that the competitive analysis of fBuffAlg can be conducted within a *cycle* (see Definition 2.5.1).

Lemma 2.6.2: Let $\bar{\boldsymbol{\omega}}$ be a worst instance and $\mathcal{C}$ be a cycle realized by fBuffAlg under $\bar{\boldsymbol{\omega}}$, then the profit ratio during $\mathcal{C}$ is equal to the competitive ratio of fBuffAlg.

The proof is analogous to that for Lemma 2.5.4 and is omitted. The following theorem shows the competitive ratio of fBuffAlg.

Theorem 2.6.1: Under the fractional admission assumption, fBuffAlg achieves the competitive ratio $\left[1+(\ln\tilde{\theta}+1)\tilde{\theta}/W\right]\cdot(\ln\tilde{\theta}+1)$.

Proof of Theorem 2.6.1. We assume there is a worst instance for fBuffAlg which only contains one cycle and at time t, the state of fBuffAlg is as shown in Figure 2.3. By t_i, we denote the time slot at which the i-th sub-queue is built up. From the allocation policy of fBuffAlg, we have

$$\bar{b}_1(t)\geq\bar{b}_2(t)\geq\cdots\geq\bar{b}_N(t).$$

During $[t_i+1,t]$, the threshold values of fBuffAlg are always less than or equal to $g_W(\bar{b}_i(t)+1)$. Thus, during the worst instance, packet values in $[t_i+1,t]$ are always less than or equal to $g_W(\bar{b}_i(t)+1)$. Otherwise, the packet value will be larger than the threshold of fBuffAlg, contradicting Lemma 2.6.1.

Above analysis demonstrates that under the worst instance for fBuffAlg, packet values are not larger than $g_W(\bar{b}_i(t)+1)$ after the i-th sub-queue is built up. Thus, the profit earned by OPT over $[1,t]$ is less than or equal to

$$\mathsf{Prof}_{\mathsf{OPT}}\leq g_W(\bar{b}_1(t)+1)+g_W(\bar{b}_2(t)+1)+\cdots+g_W(\bar{b}_N(t)+1) \quad (2.13)$$

On the other hand, according to the admission policy of fBuffAlg, a sub-packet is admitted by a sub-queue only when its packet value is larger than or equal to the threshold, so the aggregated profit of the buffered sub-packets in the i-th sub-queue is not less than $\sum_{k=1}^{\bar{b}_i(t)}g_W(k)$. By the property of the threshold function g_W (see the analysis in Section 2.5.1), we have

$$\sum_{k=1}^{\bar{b}_i(t)}g_W(k)\geq\frac{g_W\left(\bar{b}_i(t)\right)}{\ln\tilde{\theta}+1}.$$

Hence, the profit earned by fBuffAlg over $[1,t]$ is larger than or equal to

$$\mathsf{Prof}_{\mathsf{fBuffAlg}}\geq\frac{g_W(\bar{b}_1(t))+g_W(\bar{b}_2(t))+\cdots+g_W(\bar{b}_N(t))}{\ln\tilde{\theta}+1}. \quad (2.14)$$

Putting together Equations (2.13) and (2.14), we have

$$\begin{aligned}
&\textsc{cr}(\mathsf{fBuffAlg}) \\
&\leq \frac{g_W(\bar{b}_1(t)+1)+g_W(\bar{b}_2(t)+1)+\cdots+g_W(\bar{b}_N(t)+1)}{g_W(\bar{b}_1(t))+g_W(\bar{b}_2(t))+\cdots+g_W(\bar{b}_N(t))}\cdot(\ln\tilde{\theta}+1) \\
&\leq \max_{1\leq i\leq n}\frac{g_W(\bar{b}_i(t)+1)}{g_W(\bar{b}_i(t))}\cdot(\ln\tilde{\theta}+1) \\
&\leq \left[1+(\ln\tilde{\theta}+1)\frac{\tilde{\theta}}{W}\right]\cdot(\ln\tilde{\theta}+1).
\end{aligned}$$

The last inequality is according to the fact that the largest derivative of $g_W(\bar{b}_i),\ 0\leq\bar{b}_i\leq W$ is $v_{\min}(\ln\tilde{\theta}+1)\tilde{\theta}/W$ and $g(\bar{b}_i)\geq v_{\min}$. □

Corollary 2.6.1: Assuming $W\to\infty$, the competitive ratio of fBuffAlg approximates the lower bound of $\ln\tilde{\theta}+1$.

Remark 2.6.1: We note the following property concerning the admission amount when $W\to\infty$. For notational convenience, let us denote by $\tilde{b}_i(t)$ the actual amount of packet in the i-th sub-queue and use $\tilde{a}_i(t)$ to denote the actual amount of packet that can be admitted by the i-th sub-queue. According to the admission policy of fBuffAlg, we can deduce that when $W\to\infty$, the amount of packet that the i-th sub-queue can admit is

$$\tilde{a}_i(t)=\left[\frac{\ln\frac{v(t)}{v_{\min}}+1}{\ln\tilde{\theta}+1}-\tilde{b}_i(t-1)\right]^+.$$

Similarly, the actual admitted amount at time slot t, denoted by $a^{\mathsf{fBuffAlg}}(t)$, can be obtained by truncating the aggregation of $\tilde{a}_i(t)$ for all sub-queues. That is,

$$a^{\mathsf{fBuffAlg}}(t)=\min\left\{\sum_{i=1}^{n}\tilde{a}_i(t),1\right\}.$$

Then, a fraction of $a^{\mathsf{fBuffAlg}}(t)$ of the packet will be allocated among sub-queues in a water-filling way as shown in Figure 2.4.

2.6.3 Optimal Randomized Strategy for the Discrete Admission Model

In this section, we extend the fractional algorithm in the previous section and design a randomized online algorithm for the general discrete model. Intuitively, the expected competitive ratio can be improved to be equal to that of fBuffAlg by properly designing the buffering probability. In the following, we introduce a randomized strategy rBuffAlg (as for randomized online algorithm) which can attain this goal by online rounding. Given that the amount of packet buffered by fBuffAlg assuming $W \to \infty$ at time slot t is $a^{\text{fBuffAlg}}(t)$, we propose the following operation rules for rBuffAlg.

At time slot t, our proposed randomized rounding algorithm computes the admission probability according to the admission amount of fBuffAlg with $W \to \infty$ (summarized as Algorithm 2.2).

Algorithm 2.2 Randomized Rounding Algorithm rBuffAlg, at slot t

```
prob$(t) \leftarrow 0$ the admission probability of packet at slot $t$
if $\lfloor b^{\text{fBuffAlg}}(t-1) + a^{\text{fBuffAlg}}(t)\rfloor = \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor$ do
    if $b^{\text{rBuffAlg}}(t-1) = \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor$ do
        $\text{prob}(t) \leftarrow \frac{a^{\text{fBuffAlg}}(t)}{\lfloor b^{\text{fBuffAlg}}(t-1)\rfloor + 1 - b^{\text{fBuffAlg}}(t-1)}$    (2.15)
    else
        $\text{prob}(t) \leftarrow 0$
    end if
end if
if $\lfloor b^{\text{fBuffAlg}}(t-1) + a^{\text{fBuffAlg}}(t)\rfloor = \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor + 1$ do
    if $b^{\text{rBuffAlg}}(t-1) = \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor$ do
        $\text{prob}(t) \leftarrow 1$
    else
        $\text{prob}(t) \leftarrow \frac{b^{\text{fBuffAlg}}(t) + b^{\text{fBuffAlg}}(t-1) - \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor - 1}{b^{\text{fBuffAlg}}(t-1) - \lfloor b^{\text{fBuffAlg}}(t-1)\rfloor}$
    end if
end if
Admit the packet with probability prob$(t)$
```

Before analyzing the competitive ratio of above randomized policy, we first provide a general lower bound for the competitive ratio of randomized online algorithms. Subsequent analysis demonstrates that the obtained lower bound is tight.

Theorem 2.6.2: The competitive ratio for any randomized online algorithm is lower bounded by $\ln\tilde{\theta} + 1$.

Proof of Theorem 2.6.2. Given the initial state of the buffer being empty, an adversary can present enough packets with packet value $\bar{v}_1 = v_{\min}$ to a randomized online algorithm $\mathfrak{R}$. It is obvious that the number of packets admitted by $\mathfrak{R}$ will increase and definitely converge to some constant value, denoted by λ_1. Note that λ_1 can be regarded as a random variable for a randomized online algorithm. After that, we further present enough packets with packet value $\bar{v}_2 > \bar{v}_1$ to $\mathfrak{R}$, and assume ultimately there are λ_2 packets buffered. We repeat this process for n times and let $\bar{v}_n = v_{\max}$. The adversary can choose to stop the above input instance at any time. Hence, for such an input instance, it is possible for any $\mathfrak{R}$ to achieve a profit ratio equal to $(\bar{v}_i B)/\left(\sum_{k=1}^{i} \bar{v}_k \lambda_k\right)$, $i = 1, 2, \ldots, n$. The expected competitive ratio is larger than or equal to the maximum one among those ratios. That is

$$\text{ECR}(\mathfrak{R}) \geq \max_{i=1,2,\ldots,n} \frac{\bar{v}_k B}{\boldsymbol{E}\left[\sum\limits_{k=1}^{i} \bar{v}_k \lambda_k\right]} = \max_{i=1,2,\ldots,n} \frac{\bar{v}_k B}{\sum\limits_{k=1}^{i} \bar{v}_k \boldsymbol{E}\left[\lambda_k\right]}.$$

Obviously, $\sum\limits_{k=1}^{i} \boldsymbol{E}\left[\lambda_k\right] \leq B$. From the previous analysis, we can verify that

$$\max_{i=1,2,\ldots,n} \frac{\bar{v}_i B}{\sum_{k=1}^{i} \bar{v}_k \boldsymbol{E}\left[\lambda_k\right]} \geq \ln \tilde{\theta} + 1,$$

and thus $\text{ECR}(\mathfrak{R}) \geq \ln \tilde{\theta} + 1$, the proof is completed. □

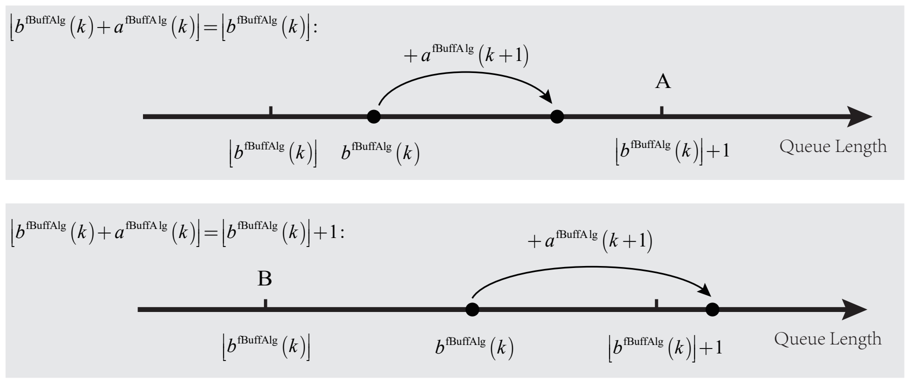
Lemma 2.6.3: $\left\lfloor b^{\mathsf{fBuffAlg}}(t) \right\rfloor \leq b^{\mathsf{rBuffAlg}}(t) \leq \left\lfloor b^{\mathsf{fBuffAlg}}(t) \right\rfloor + 1$.

Proof of Lemma 2.6.3. Assuming the initial state of the buffer is empty, we proceed to prove by induction. For the base case we have

$$\left\lfloor b^{\mathsf{fBuffAlg}}(t) \right\rfloor \leq b^{\mathsf{rBuffAlg}}(t) \leq \left\lfloor b^{\mathsf{fBuffAlg}}(t) \right\rfloor + 1,$$

which holds when $t = 0$.

Now, assume the lemma holds for $t \leq k$, i.e., $b^{\mathsf{rBuffAlg}}(t)$ is equal to either $\left\lfloor b^{\mathsf{fBuffAlg}}(k) \right\rfloor$ or $\left\lfloor b^{\mathsf{fBuffAlg}}(k) \right\rfloor + 1$. As illustrated in Figure 2.5, when $b^{\mathsf{rBuffAlg}}(t)$ lies at point "A", the arriving packet is buffered with probability 0; and when $b^{\mathsf{rBuffAlg}}(t)$ lies at point "B", the arriving packet is buffered with probability 1.



$\lfloor b^{\text{fBuffAlg}}(k)+a^{\text{fBuffAlg}}(k)\rfloor=\lfloor b^{\text{fBuffAlg}}(k)\rfloor$:
$+a^{\text{fBuffAlg}}(k+1)$
A
$\lfloor b^{\text{fBuffAlg}}(k)\rfloor$
$b^{\text{fBuffAlg}}(k)$
$\lfloor b^{\text{fBuffAlg}}(k)\rfloor+1$
Queue Length
$\lfloor b^{\text{fBuffAlg}}(k)+a^{\text{fBuffAlg}}(k)\rfloor=\lfloor b^{\text{fBuffAlg}}(k)\rfloor+1$:
$+a^{\text{fBuffAlg}}(k+1)$
B
$\lfloor b^{\text{fBuffAlg}}(k)\rfloor$
$b^{\text{fBuffAlg}}(k)$
$\lfloor b^{\text{fBuffAlg}}(k)\rfloor+1$
Queue Length

Fig. 2.5: The illustration of the proof for Lemma 2.6.3.

Putting together the observations in Figure 2.5, the following inequalities hold:

$$\begin{aligned}\left\lfloor b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1)\right\rfloor \leq & b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) \\ & \leq \left\lfloor b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1)\right\rfloor + 1.\end{aligned}$$

Hence,

$$\begin{aligned}&\left[\left\lfloor b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1)\right\rfloor - u(k+1)\right]^{+} \\ &\leq \left[b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) - u(k+1)\right]^{+} \\ &\leq \left[\left\lfloor b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1)\right\rfloor + 1 - u(k+1)\right]^{+},\end{aligned}$$

for $u(k+1) \in \mathbb{N}$, so

$$\left\lfloor b^{\mathsf{fBuffAlg}}(k+1)\right\rfloor \leq b^{\mathsf{rBuffAlg}}(k+1) \leq \left\lfloor b^{\mathsf{fBuffAlg}}(k+1)\right\rfloor + 1.$$

This completes the proof. □

Lemma 2.6.4: $\boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(t)\right] = a^{\mathsf{fBuffAlg}}(t)$.

Proof of Lemma 2.6.4. We prove by induction again. Assume the initial state of the queue length is 0, i.e., $b^{\mathsf{fBuffAlg}}(0) = b^{\mathsf{rBuffAlg}}(0) = 0$. If $a^{\mathsf{fBuffAlg}}(1) < 1$, then

$$\left\lfloor b^{\mathsf{fBuffAlg}}(0) + a^{\mathsf{fBuffAlg}}(1)\right\rfloor = \left\lfloor b^{\mathsf{fBuffAlg}}(0)\right\rfloor.$$

Hence, according to Equation (2.15), the packet is admitted with probability

$$\frac{a^{\mathsf{fBuffAlg}}(1)}{\left\lfloor b^{\mathsf{fBuffAlg}}(0)\right\rfloor + 1 - b^{\mathsf{fBuffAlg}}(0)} = a^{\mathsf{fBuffAlg}}(1).$$

If $a^{\mathsf{fBuffAlg}}(1) = 1$, then

$$\left\lfloor b^{\mathsf{fBuffAlg}}(0) + a^{\mathsf{fBuffAlg}}(1)\right\rfloor = \left\lfloor b^{\mathsf{fBuffAlg}}(0)\right\rfloor + 1.$$

The arriving packet will be admitted with probability 1. So when $t = 1$, we have $\boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(t)\right] = a^{\mathsf{fBuffAlg}}(t)$ and also $\boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(t)\right] = b^{\mathsf{fBuffAlg}}(t)$.

Now, assume $\boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(t)\right] = a^{\mathsf{fBuffAlg}}(t)$ and $\boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(t)\right] = b^{\mathsf{fBuffAlg}}(t)$ hold for $t \leq k$. Based on Lemma 2.6.3, we know that $b^{\mathsf{rBuffAlg}}(t)$ is equal to either $\left\lfloor b^{\mathsf{fBuffAlg}}(t)\right\rfloor$ or $\left\lfloor b^{\mathsf{fBuffAlg}}(t)\right\rfloor + 1$. When $\boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(k)\right] = b^{\mathsf{fBuffAlg}}(k)$, we can deduce that $b^{\mathsf{rBuffAlg}}(k)$ is equal to $\left\lfloor b^{\mathsf{fBuffAlg}}(k)\right\rfloor$ with probability $\left\lfloor b^{\mathsf{fBuffAlg}}(k)\right\rfloor + 1 - b^{\mathsf{fBuffAlg}}(k)$ and equal to $\left\lfloor b^{\mathsf{fBuffAlg}}(k)\right\rfloor + 1$ with probability

$b^{\mathsf{fBuffAlg}}(k) - \lfloor b^{\mathsf{fBuffAlg}}(k) \rfloor$. By verifying the operations of rBuffAlg, we can easily get that $\boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(k+1)\right] = a^{\mathsf{fBuffAlg}}(k+1)$. Moreover, if $b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1) \geq u(k+1)$, we must have $b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) \geq u(k+1)$ according to Lemma 2.6.3. In this case, we have

$$\begin{aligned}
&\boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(k+1)\right] \\
&= \boldsymbol{E}\left[[b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) - u(k+1)]^+\right] \\
&= \boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) - u(k+1)\right] \\
&= \boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(k)\right] + \boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(k+1)\right] - u(k+1) \\
&= b^{\mathsf{fBuffAlg}}(k+1).
\end{aligned}$$

If $b^{\mathsf{fBuffAlg}}(k) + a^{\mathsf{fBuffAlg}}(k+1) \leq u(k+1)$, we must have $b^{\mathsf{rBuffAlg}}(k) + a^{\mathsf{rBuffAlg}}(k+1) \leq u(k+1)$. In this case, $b^{\mathsf{rBuffAlg}}(k+1) = b^{\mathsf{fBuffAlg}}(k+1) = 0$. Then, we can conclude that $\boldsymbol{E}\left[a^{\mathsf{rBuffAlg}}(t)\right] = a^{\mathsf{fBuffAlg}}(t)$ and $\boldsymbol{E}\left[b^{\mathsf{rBuffAlg}}(t)\right] = b^{\mathsf{fBuffAlg}}(t)$ hold for $t = k+1$. By concluding the above all, we can get the final results as desired. □

Theorem 2.6.3: rBuffAlg achieves the optimal competitive ratio, i.e., $\ln \tilde{\theta} + 1$.

Proof of Theorem 2.6.3. As shown in Corollary 2.6.1, when $W \to \infty$, the competitive ratio of fBuffAlg achieves $\ln \tilde{\theta} + 1$. Moreover, from Lemma 2.6.4, we have that the expected amount of admitted packets under rBuffAlg is always equal to that of fBuffAlg for all times, and thus rBuffAlg achieves the same competitive ratio as fBuffAlg with $W \to \infty$. This proves the optimality of rBuffAlg. □

In this chapter, we studied the classic non-preemptive QoS buffer management problem, where its optimal online solution was an open problem for a decade. By relaxing the original discrete model to a fractional setting, we proposed a novel online algorithm that maintains a series of virtual sub-queues to record the available buffer budget over time. We proved it achieves the optimal competitive ratio. By devising a randomized rounding scheme, we then extended the fractional algorithm into the original discrete setting. Our analysis demonstrated that the randomized scheme achieves the optimal competitive ratio of $\ln \tilde{\theta} + 1$. Last but not the least, the problem is of interest in a general admission control problem that could be applied to the state-of-the-art applications. As an example, we stress the value-based resource allocation in data centers with limited computation capacities, in which the jobs must be either admitted or rejected upon their arrival based on their values and the utilization of the servers.

□ **End of chapter.**

3. ONLINE TRADING I: ONLINE OFFERING OF RENEWABLE SUPPLIES

The Online QoS Buffer Management Problem studied in Chapter 2 is a common problem and closely related to lots of practical problems in engineering and operation research. In addition to the basic setting of the Online QoS Buffer Management problem, we also investigate the recent problems which generalize the profit maximization problem in a deregulated market. Specifically, the extension focuses on designing bidding strategies that, with the goal of maximizing the profit, determine the offering price and volume, for SRGENCO that participates in hour-ahead market which adopts an auction mechanism.

3.1 Problem Background

Renewables are attractive in that they are clean, free (except capital and maintenance cost), and inexhaustible. Integration of renewables into the power system and particularly in the electricity market, however, is challenging since their generation is uncontrollable, intermittent, and unpredictable. A promising approach to facilitate renewable integration and hedge against the uncertainty, is to equip the renewable plants with the giant energy storage systems [52, 57]. Some examples are the storage stations at Southern California (with capacity of 40MWh), South Korea (16MWh), and Germany (15MWh) [1].

As depicted in Figure 3.1, we consider a scenario in which an SRGENCO, like other traditional generation companies, participates in hour-ahead market by (1) submitting the offer. After receiving the offers, (2) the market operator matches the offers with the bids from the demand-side and announces a market clearing price. If the offering price of SRGENCO is less than the clearing price, (3) its offering volume is considered as the *commitment* to the market for the next hour. In turn, SRGENCO is paid according to the clearing price. If the offering price of SRGENCO is less than the clearing price, its offering volume is considered as the *commitment* to the market for the next hour.

Finding profit maximization offering strategy for a renewable producer without storage is nontrivial due to the inherent uncertainty of the renewables and dynamics in the market clearing price. In the

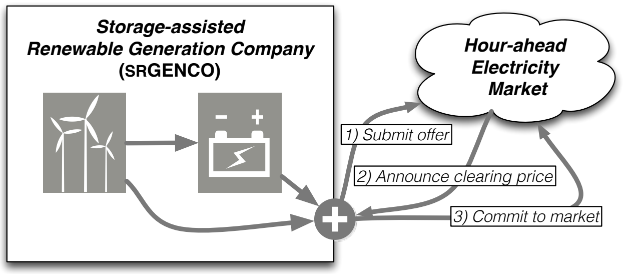

Storage-assisted Renewable Generation Company (SRGENCO)
Hour-ahead Electricity Market
1) Submit offer
2) Announce clearing price
3) Commit to market

Fig. 3.1: The energy offering and storage management scenario.

presence of storage, the offering strategy is even more challenging because of the additional design space enabled by the storage. More specifically, SRGENCO can use the storage to absorb the uncertainty of renewables and to compensate for the slots that the renewable output cannot fulfill the commitment. However, the storage provides another economic advantage. That is, it can shift the energy through absorbing the renewable output during low price periods, and then discharging during high price periods. In this way, designing profit maximization offering strategy in the presence of storage comes with wider design space than those without the storage and potentially can bring more profit for SRGENCO.

The future inputs to the problem, i.e., the renewable output and the clearing price, however, are unknown for SRGENCO when submitting offer. This emphasizes the need for online solution design which is challenging, since the problem is coupled across time due to the evolution of the storage. We note that some similar problems have been studied in literature using stochastic optimization approaches [87], however, the solution approach thesis is different since it has no assumption on the stochastic modeling of the future input. Our work could be considered as an extension of conversion problems [122]. Different from the online optimization problem formulated in Chapter 2, the scenario introduced in this chapter involves an additional auction process when the SRGENCO participates in the market.

3.2 Related Solutions

Due to the rapid increase in renewable energy deployment, recently, there has been a significant attention on designing profit-effective strategies for renewable producers to participate in the deregulated electricity market. The profit efficiency of a renewable producer is heavily restricted by the low accuracy of forecasting. [65] shows that the error prediction costs can reach as much as 10% of total income of a wind power producer. To this end, some works (e.g., [36, 123, 37, 41]) propose to bring the wind to the short-term market. We review the related work in the following.

Renewable power producer without storage in electricity market. In order to decrease the imbalance penalty imposed to wind farm, [120] proposes a new stochastic programming approach to generate optimal wind power production offers in short-term market. In their work, a stochastic process is assumed for the forecasting error of wind generation. In [41], Botterud et al., present a new model for optimal trading of wind power in day-ahead electricity markets under uncertainty in wind power output and market prices. In their model, the settlement mechanisms in markets with locational marginal prices are also taken into account. [37] also aims at designing optimal contract offerings in a competitive two-settlement market. The authors further take into account a different multi-period setting in which the wind power producer has a recourse opportunity to adjust its day-ahead commitment in an intra-day market. The performance of all above studies largely degrades as the forecasting errors increase. Actually, even with state-of-the-art forecasting methods, it is still complicated to predict even one-hour ahead power generation of a wind turbine, as well as the spot price in electricity market. In addition, none of the aforementioned works leverage competitive design framework, as a promising approach that relies on no stochastic modeling of market price and renewable output.

Renewable power producer with storage in electricity market. In order to mitigate the uncertainty of renewable production, several studies [45, 101, 42, 75, 94] advocate the idea of deploying grid-scale storage system to improve dispatchability of renewable power producer. In addition, the storage can absorb surplus or inexpensive energy during off-peak hours, and then discharge it during on-peak hours, when electricity prices are typically high. In view of these considerations, grid-scale storage has drawn the attention of power producers and utilities to address many challenges they are dealing with, especially at present, when the penetration of intermittent and inflexible renewable sources is on the rise. As a fundamental technical difference with the case without storage, in the case with storage, the offering strategy design is more involved because of the evolution of storage level. [45] proposes an hourly-discretized optimization algorithm to identify the optimum daily operational strategy to be followed by the wind turbines and the hydro generation pumping equipments, provided that a wind-power forecasting is available. In [75], the optimization model is formulated as a two-stage stochastic programming problem with two random parameters: market prices and wind generation. The optimal offers for day-ahead spot market are the "here and now" decisions while the optimal operation of the facilities are the recourse variables. [42] and [101] optimize the size of hydrogen storage system which can be used to jointly offer in the electricity market with the wind power plant. The above studies are all constrained by long-term forecasting errors, and their method cannot be extended to the case which involves large time horizon.

The most related problem to our study is [94]. J. Kim and W. Powell in [94] assume an auto-regressive energy generation process for the wind, and then the Markov Decision Process can be derived to solve the problem of making advance energy commitments in the presence of a storage system. There is a fundamental technical difference between the solution approach in our work and the one in [94]. Our approach is competitive analysis, in which neither the exact values nor the distribution of the future input is known in advance. In contrast, in [94], the solution relies on specific stochastic modeling of the input. While the performance of the solution in [94] largely depends on the accuracy of underlying stochastic modeling for renewable output and market price, our approaches characterizes the fundamental price of uncertainty without any underlying stochastic modeling assumptions.

3.3 Summary of Results and Adopted Techniques

We focus on designing profit maximization offering strategies, i.e., the strategies that determine the offering price and volume, for a storage-assisted renewable power producer that participates in hour-ahead electricity market. Designing the strategies is challenging since (i) the underlying problem is coupled across time due to the evolution of the storage level, and (ii) inputs to the problem including the renewable output and market clearing price are unknown when submitting offers. Following the competitive online algorithm design approach introduced in Chapter 2, we first study a basic setting where the renewable output and the clearing price are known for the next hour. We propose sOffAlg, a simple online offering strategy that achieves the best possible competitive ratio of $O(\log\hat{\theta})$, where $\hat{\theta}$ is the ratio between the maximum and the minimum clearing prices. Then, we consider the case in which the clearing price is unknown. By exploiting the idea of submitting multiple offers to combat price uncertainty, we propose mOffAlg, and demonstrate that the competitive ratio of mOffAlg converges to that of sOffAlg as the number of offers grows. Finally, we extend our approach to the scenario where the renewable output has forecasting error. We propose gOffAlg as the generalized offering strategy and characterize its competitive ratio as a function of the forecasting error. Our trace-driven experiments demonstrate that our algorithms achieve performance close to the offline optimal and outperform a baseline alternative significantly.

By introducing the system model and problem formulation in Sec. 3.4,

1. In Sec. 3.5, we study a basic setting where the exact values of renewable output and the clearing price for the next slot are known to SRGENCO before submitting the offer. We propose sOffAlg, a simple online offering strategy, in which the offering volume is calculated through a piecewise exponential/constant function of the renewable output and the current storage level, as well as the clearing price of the next slot. Following the technique used in Chapter 2, our analysis demonstrates that sOffAlg achieves the best possible competitive ratio of $O(\log \hat{\theta})$.

2. In Sec. 3.6.1, we study the case where the clearing price is unknown and propose mOffAlg. In mOffAlg, SRGENCO submits multiple offers, each of which conveys a portion of total offering volume, at different offering prices. Our analysis shows that the competitive ratio of mOffAlg converges to the ratio of sOffAlg as the number of offers grows. Moreover, in Sec. 3.6.2, our approach is extended to the case where SRGENCO knows renewable output with forecasting error. We propose gOffAlg as the generalized offering strategy and characterize the competitive ratio as a function of the maximum forecasting error.

3. In Sec. 4.6, by extensive numerical experiments based on real-world traces, we show that our online offering strategies can achieve satisfactory performance as compared to the offline optimum. In addition, gOffAlg with 10% forecasting error improves the profit of SRGENCO by 15% as compared to the baseline scenario without storage. As notable observations, our experiments demonstrate that when the market clearing price is unknown, submitting 3 offers is sufficient to achieve almost the same performance as the case that the market price is known. Moreover, forecasting error of more than 20% significantly degrades the performance results. In summary, our observations demonstrate that, while the uncertainty in market price can be effectively handled by multiple offer submissions, accurate short-term renewable forecasting is vital for SRGENCO to obtain a desired profit.

Tab. 3.1: Summary of key notations Related to the Formulation of the Online Offering Problem

Notation	Description
t	Index of one-hour time slot
T	The number of time slots, $T \geq 0$
$\mathcal{T}$	Set $\mathcal{T} = \{1, 2, \dots, T\}$
$p(t)$	Market clearing price at t, $p_{\min} \leq p(t) \leq p_{\max}$
$\hat{\theta}$	The ratio between the maximum and the minimum clearing prices, $\hat{\theta} = p_{\max}/p_{\min}$
$r(t)$	The renewable output of SRGENCO at t
S	The capacity of storage system
ρ_c	The maximum charging rate of storage system
ρ_d	The maximum discharging rate of storage system
$s(t)$	The storage level at the beginning of t, see Equation (4.1)
$\hat{p}(t)$	**optimization variable**, offering price of SRGENCO at t
$\hat{o}(t)$	**optimization variable**, offering volume of SRGENCO at t
$o(t)$	Commitment volume of SRGENCO at t, see Equation (3.1)
$\tilde{o}(t)$	Over-commitment volume of SRGENCO at t, see Equation (3.5)
$R(t)$	The net profit of SRGENCO at t, see Equation (3.6)

3.4 Problem Formulation

3.4.1 Hour-Ahead Electricity Market

The hour-ahead electricity market operates on an hourly basis[1] and SRGENCO along with other generation companies submits its offer, including the offering price and the offering volume (see Sec. 3.4.2 for details), for the forthcoming hour shortly before the operation time. The market operator (usually known as independent system operator, ISO) matches the offers collected from the generation companies with the received bids from the demand-side, e.g., utility companies. Then, using a well-established double auction mechanism [109] it determines the market clearing price for the next hour. The generation companies with the offering prices less than the clearing price are successful and the offering volume of electricity is considered as their *commitment* to be sold on the market at the clearing price. Thus, successful offers sell at prices at least as high as what they offered. All the remaining offers fail since their offering prices are greater than the clearing price.

[1] We emphasize that our model works in a short-term market in which the offers are submitted *before* actual market operation in hourly or even shorter scales, e.g., 15 or 5 minutes ahead. For real examples we refer to California ISO [3] and Nord Pool Markets [9]. For a survey of electricity markets, we refer to [63].

Formally, we consider a time-slotted model, such that the time horizon T is chopped into multiple slots with equal length, e.g., 1 hour, each of which is indexed by t. Shortly before slot t, SRGENCO along with other participants submits its offer, for the next slot. The ISO determines the clearing price $p(t)$ shortly after the participants submit their offers and bids. We assume $p_{\min} \leq p(t) \leq p_{\max}$, and parameter $\hat{\theta}$ is defined as the ratio between the maximum and the minimum clearing prices, i.e., $\hat{\theta} = p_{\max}/p_{\min}$. We later use $\hat{\theta}$ to analyze our algorithms.

3.4.2 The properties of SRGENCO

There is a Storage-assisted Renewable GENeration COmpany (SRGENCO) that produces electricity from the renewable sources such as wind farm or solar plant. At the same time SRGENCO is equipped with the storage systems to store the electricity for future commitment with potentially higher price. On the other hand, the storage could be discharged to compensate for the shortage of renewable output when the commitment to the market is beyond the renewable output.

Renewable Output

The renewable output of SRGENCO at slot t is denoted by $r(t)$ and we do not rely on any specific stochastic model of $r(t)$. In general, we assume that SRGENCO does not know the exact amount of $r(t)$ when submitting the offer. Note that $r(t)$ could be (i) directly committed to the market, (ii) committed to the market partially while the residual is stored on the storage, or (iii) entirely stored on the storage for future usage.

Offering Strategy

By offering strategy we mean the way that SRGENCO determines its offer that includes:

(i) **Offering price** denoted as $\hat{p}(t) \in [p_{\min}, p_{\max}]$, i.e., the minimum price at which SRGENCO desires to commit electricity to the market.

(ii) **Offering volume** denoted as $\hat{o}(t) \geq 0$, i.e., the amount of electricity in MWh at which SRGENCO offers to the market at slot t.[2]

[2] Note that there is no upper bound for $\hat{o}(t)$ because we assume that the market is big enough to absorb the offering volume of SRGENCO entirely.

We distinguish between the offering volume and the *commitment volume*. After the clearing price $p(t)$ is revealed, SRGENCO's offer may or may not be successful. If the offer is successful the offering volume is considered as the commitment volume. Otherwise, there would be no commitment. More specifically, we define $x(t)$ as the commitment volume of SRGENCO at slot t as

$$o(t) = \begin{cases} \hat{o}(t) & \text{if } p(t) \geq \hat{p}(t), \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The goal of this study is to design an offering strategy for SRGENCO to submit both offering price $\hat{p}(t)$ and offering volume $\hat{o}(t)$ so as to maximize its long-term profit.

Storage Model

We denote the maximum capacity of storage system of SRGENCO by S and let ρ_c and ρ_d be its maximum charging and discharge rates, respectively. In addition, let $s(t) \in [0, S]$ be the storage level at the *beginning* of slot t. Given the renewable output $r(t)$ and the commitment volume $o(t)$, the evolution of the storage level of SRGENCO is given by

$$s(t+1) = \Big[s(t) + o_c(t) - o_d(t)\Big]_{\mathcal{S}}, \tag{3.2}$$

where

$$o_c(t) = \min\Big\{\rho_c, \big[r(t) - o(t)\big]^+\Big\}, \tag{3.3}$$

is the charging amount of the storage at slot t,

$$o_d(t) = \min\Big\{\rho_d, \big[o(t) - r(t)\big]^+\Big\}, \tag{3.4}$$

and $o_d(t)$ is the discharging amount of the storage at slot t. Moreover, $[.]^+$ and $[.]_{\mathcal{S}}$ define the projections onto the positive orthant and set $\mathcal{S} = [0, S]$, respectively. Since SRGENCO is empowered by the storage, the commitment volume $o(t)$ might be either greater, less, or equal to the renewable output $r(t)$. The evolution of the storage for each case is as follows:

(i) $r(t) = o(t)$: in this case the entire renewable output is committed to the market and there is no change on the storage level, i.e., $s(t+1) = s(t)$.

(ii) $r(t) > o(t)$: in this case, $r(t) - o(t) > 0$ represents the amount of the surplus in the renewable output. Ideally, this surplus must be charged into the storage for the forthcoming commitments. However, because of the charging rate ρ_c it may not be possible, which is indeed captured in Equation (3.3).

(iii) $r(t) < o(t)$: in this case not only the entire renewable output is committed, but also the storage should contribute in fulfilling the residual commitment, i.e., $o(t) - r(t) > 0$. Again, given the storage level $s(t)$, and the discharge rate ρ_d, the residual commitment may not be satisfied.

Over-Commitment

Recall that SRGENCO does not know the exact renewable output $r(t)$ when submitting the offer, hence, it may fail to fulfill its commitment, which we refer to as the over-commitment. Let us denote $\tilde{o}(t)$ as the over-commitment volume at t expressed as

$$\tilde{o}(t) = \left[o(t) - \Big(r(t) + \min\big\{ s(t), \rho_d \big\} \Big) \right]^+. \tag{3.5}$$

Note that the maximum amount that SRGENCO can provide to the market in operation time t is the aggregation of the renewable output $r(t)$ and the maximum amount that could be discharged from the storage, i.e., $\min\big\{s(t), \rho_d\big\}$. Since $r(t)$ is unknown to SRGENCO when submitting the offer, the commitment volume $o(t)$ might be greater than the amount that SRGENCO can really output, i.e., $r(t) + \min\big\{s(t), \rho_d\big\}$.

3.4.3 Profit Model

By committing $o(t)$, the profit obtained by SRGENCO is $p(t)o(t)$. The consequence of over-commitment is captured in profit model by augmenting a penalty term. We adopt the penalty model in [94], in which the unit penalty payment in over-commitment is linearly proportional to the spot price $p(t)$ in the form of $\alpha_1 p(t) + \alpha_2$, where $\alpha_1, \alpha_2 \geq 0$ are constants.

Concluding above, the net profit obtained by SRGENCO at slot t, denoted by $R(t)$, is expressed as total profit subtracted by the (potential) penalty of the over-commitment, i.e.,

$$R(t) = p(t)o(t) - (\alpha_1 p(t) + \alpha_2)\tilde{o}(t). \tag{3.6}$$

3.4.4 Profit Maximization Problem

The objective is to maximize the cumulative profit obtained by SRGENCO over time horizon $\mathcal{T}$. The profit maximization offering strategy problem (OnOffer) is formally casted as

$$\begin{aligned} \textsf{OnOffer} \quad & \max \quad \sum_{t\in\mathcal{T}} R(t), \quad \text{s.t. Equation (4.1)}, \\ & \text{vars.} \quad \hat{p}(t) \in [p_{\min}, p_{\max}], \hat{o}(t) \geq 0, t \in \mathcal{T}. \end{aligned}$$

In offline scenario, in which the values of $r(t)$ and $p(t)$ as the time-varying inputs to the problem are known ahead of time, the problem is a linear one which is easy to solve. We refer to [45, 75] as related works that study related problems in offline settings. We note that in the offline scenario, the clearing prices are known to SRGENCO, hence it is not required to submit the offering price anymore. Consequently, the offering strategy reduces to announcing the commitment volume directly. In this way, the problem could be reformulated in an equivalent form with simpler structure.

In real-world, however, neither the renewable output $r(t)$ nor the clearing price $p(t)$ are revealed to SRGENCO when submitting the offer. Hence the focus in this thesis is to tackle the problem in online setting, so, we formulate the problem in a way that SRGENCO submits both offering price and offering volume. Solving OnOffer in online setting is challenging, since the problem is coupled over the time in the presence of the storage system. Recall that an important advantage of incorporating storage towards profit maximization is to (fully or partially) store the renewable supply in the storage when that market price is low, and discharge it when the market price is high. Without knowing the future values of $p(t)$ and $r(t)$, finding a profit maximization offering strategy, that implicitly determines how renewable supply and the stored electricity in the storage must be consumed is challenging.

Finally, we note that in [87, 94], by using the Markov decision process and approximate dynamic programming, different offering strategies are proposed given a particular probabilistic model of the clearing price and the renewable output. In these approaches, the solution is obtained in the sense of probabilistic expectation. In practice, however, real values might deviate from the underlying probabilistic models. Our general approach as explained in Sec. 3.4.5 has no assumptions on the stochastic modeling of the unknown time-varying inputs.

3.4.5 Online Competitive Algorithm Design

Our approach in this study is to follow competitive online algorithm design and propose online offering strategies in which the decision is made based on *only the current information*, and without any assumptions on the stochastic model on the renewable output and the clearing price. Similar to the Online QoS Buffer Management Problem, we use competitive ratio to evaluate how good is the online solution, which is defined similar to that of Online QoS Buffer Management.

Definition 3.4.1: When the underlying problem is a profit maximization one, for an online algorithm $\mathfrak{A}$, its competitive ratio is defined as the maximum ratio between offline optimum and the profit obtained by $\mathfrak{A}$, over all inputs, i.e.,

$$\textsc{cr}(\mathfrak{A}) \triangleq \max_{\boldsymbol{\omega}\in\Omega} \frac{R_{\mathsf{OPT}}(\boldsymbol{\omega})}{R_{\mathfrak{A}}(\boldsymbol{\omega})}, \tag{3.7}$$

where $\boldsymbol{\omega} \in \Omega$ refers to an instance of the online input parameters as

$$\boldsymbol{\omega} \triangleq \big[\omega(t) = (p(t), r(t))\big]_{t\in\mathcal{T}}, \tag{3.8}$$

and Ω is the set of all input instances. Moreover, $R_{\mathsf{OPT}}(\boldsymbol{\omega})$ and $R_{\mathfrak{A}}(\boldsymbol{\omega})$ are the profits earned by the optimal offline solution and the online algorithm $\mathfrak{A}$ respectively, when the input is $\boldsymbol{\omega}$.

By this definition, the smaller the competitive ratio, the better the performance is, since it guarantees no matter what the input is, the online optimal strategy obtains the profit close to the offline optimum.

3.5 Optimal Online Offering Strategy with Accurate Single-Slot Prediction

In this section, we propose online competitive algorithms for a simplified version of OnOffer, in which the accurate data for the next slot is available for both the renewable supply and the clearing price. Later in Sec. 3.6, based on the insights from result of this section on the simplified scenario, we tackle the general case and propose online algorithms with neither the renewable supply nor the clearing price known to SRGENCO when submitting the bids.

3.5.1 Simplified Problem with Accurate Single-Slot Prediction

For the sake of simplification in design, we first assume that both $p(t)$ and $r(t)$ values for the next slot are revealed to SRGENCO, perhaps by accurate short-term forecasting tools. In this way, OnOffer is largely simplified in two ways:

(i) since $p(t)$ is known, the offering strategy reduces to finding just the commitment amount $o(t)$. We relax this assumption in Sec.3.6.1.

(ii) since $r(t)$ is known, all the inputs and variables in Equation (3.5) are known to SRGENCO when submitting the offer, hence the over-commitment never happens, and the penalty term in the objective of OnOffer vanishes. We relax this assumption in Sec. 3.6.2.

Since in the new setting the price $p(t)$ is known for the next slot, and to be consistent with the general formulation, we set $\hat{p}(t) = p(t)$ and following Equation (3.1), we get $o(t) = \hat{o}(t)$. Then, given the above assumptions, the only optimization variable would be the commitment amount $o(t)$. Now, we cast the simplified offering strategy problem sOnOffer as

$$
\begin{aligned}
\textsf{sOnOffer} \quad & \max && \sum_{t\in\mathcal{T}} p(t)o(t) \\
& \text{s.t.} && o(t) \le \min\{s(t), \rho_d\} + r(t), \\
& && s(t+1) = \Big[s(t) + o_c(t) - o_d(t)\Big]_{\mathcal{S}}, \\
& \text{var}: && o(t) \ge 0, t \in \mathcal{T},
\end{aligned}
$$

where the first constraint ensures that over-commitment never happens. The second constraint involves the evolution of the storage level, where $x_c(t)$ and $x_d(t)$ defined in Eqs. (3.3)-(3.4) represent the charging and discharging amounts at time t.

3.5.2 Online Algorithm Design for sOnOffer

In this section, we propose a simple online offering strategy (sOffAlg) to solve sOnOffer. Then we analyze its competitiveness and show that sOffAlg achieves the best competitive ratio.

High Level Intuitions

Intuitively, a proper offering strategy must consider two issues in decision making:

(i) the clearing price $p(t)$ for the incoming slot t, the higher the price, the more the SRGENCO is willing to commit,

(ii) the storage level $s(t)$, if the storage level is almost full, SRGENCO would be more interested to commit to have more capacity for the forthcoming slots to store the electricity. On the other hand, if the storage level is almost unoccupied, SRGENCO might keep this electricity to commit with higher price in future slots.

Putting together both clearing price and the storage level, we design our algorithm following an *adaptive threshold-based* strategy, since it adaptively changes the offering volume based on the current storage level and the clearing price.

Main Algorithm

The main idea is to construct a function $g(s) : [0, S] \to [p_{\min}, p_{\max}]$.[3] The input to function $g(\cdot)$ is the aggregation of the incoming renewable supply $r(t)$ and the current storage level $s(t)$, projected into the capacity of the storage, i.e., $\min\{s(t) + r(t), S\}$. The output of $g(\cdot)$ is the *candidate offering price* $\hat{p}(t)$ for SRGENCO at slot t. Note that since we assume that SRGENCO knows clearing price $p(t)$, it calculates its candidate offering price $\hat{p}(t)$ and then based on the comparison between these two values decides how to submit its offer.

The function $g(\cdot)$ should be decreasing, i.e., with the increase in the current storage level, the offering price would be decreased (see intuition (ii) above). Given function $g(s)$, designing sOffAlg is rather straightforward and is summarized as Algorithm 3.1. At slot t, first, sOffAlg calculates candidate offering price $\hat{p}(t)$ based on the given $g(s)$. Then, it calculates the offering volume, whose details are explained in Sec. 3.5.3. In Line 4, the offering price is set to $\hat{p}(t) = p(t)$ to ensure that $\hat{o}(t)$ is committed to market anyway. In the next part, we explain how to design function $g(s)$.

Algorithm 3.1 sOffAlg $\big[p(t), r(t), s(t)\big]$

```
1  $s^+(t) \leftarrow \min\{s(t) + r(t), S\}$
2  $\hat{p}(t) \leftarrow g(s^+(t))$; see Eqs. (3.9) and (3.10) for the optimal design of $g(s)$
3  calculate $\hat{s}(t)$ according to $\hat{p}(t)$, see Equation (3.13) and the following analysis for the
   optimal design
4  $\hat{p}(t) \leftarrow p(t)$
5  submit offer $\big(\hat{p}(t), \hat{s}(t)\big)$;
```

3.5.3 The Design of Function $g(\cdot)$

On the Importance of Designing Function $g(\cdot)$

First, we highlight that $g(s)$ plays a critical role in sOffAlg, and the competitive ratio can be improved by optimizing function $g(s)$. To illustrate the impact of $g(s)$, we investigate the behavior of sOffAlg under different structures of function $g(s)$.

(i) $g(s) = \mathsf{con}$, where $\mathsf{con} \in [p_{\min}, p_{\max}]$, is a constant value as shown in Figure 3.5.2. If $\mathsf{con} > p_{\min}$, for the extreme case where $p(t) <$

[3] To be consistent to the notations in this thesis, ideally we must denote the function as $g(s(t))$. However, for simplicity we drop the slot index t for $s(t)$.

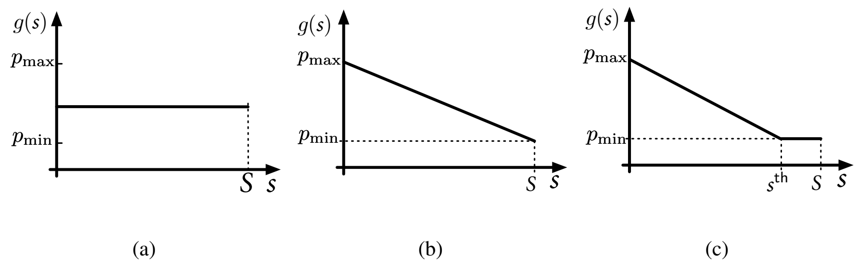


Fig. 3.2: Different structures of function $g(s)$.

con, $t \in \mathcal{T}$, it could be easily shown that the profit obtained by the sOffAlg is always 0, thereby the competitive ratio would be ∞. On the other hand, $c = p_{\min}$, the competitive ratio is upper bounded by $p_{\max}/p_{\min}$, which is not intriguing, since SRGENCO always commits the electricity to the market regardless of the clearing price and, it loses the opportunity of utilizing the potentials of storage in obtaining more profit.

Consequently, an intelligent function design aims to improve the competitiveness by reserving the storage for forthcoming slots with higher clearing price. This goal can be achieved by adopting a decreasing function.

(ii) $g(s) = ks + p_{\max}$, where $k < 0$, as depicted in Figure 3.5.2. Again, under the extreme case, CR(sOffAlg) $\to \infty$ if $p(t) = p_{\min}$. Note that this linear structure for $g(s)$ is one example and any strictly decreasing function with $g(0) = p_{\max}$ and $g(S) = p_{\min}$ behaves similarly in the worst case.

(iii) Another smart alternative is a piece-wise function as depicted in Figure 3.5.2. Function $g(s)$ again is decreasing initially. However, after the storage level reaches to a threshold value s^{th}, $g(s)$ changes to a constant function, i.e.,

$$g(s) = \begin{cases} \hat{g}(s) & \text{if } s \leq s^{\text{th}}, \\ p_{\min} & s \geq s^{\text{th}}. \end{cases} \tag{3.9}$$

Now, finding the optimal function $g(s)$ reduces to finding $\hat{g}(s)$ and s^{th}. In the next subsection, we introduce this function. Moreover in Sec. 3.5.4, we analyze the competitiveness of the algorithm and prove that the proposed function achieves the optimal CR.

Optimal Design of Function $g(s)$

The following theorem summarizes our main contribution in this section.

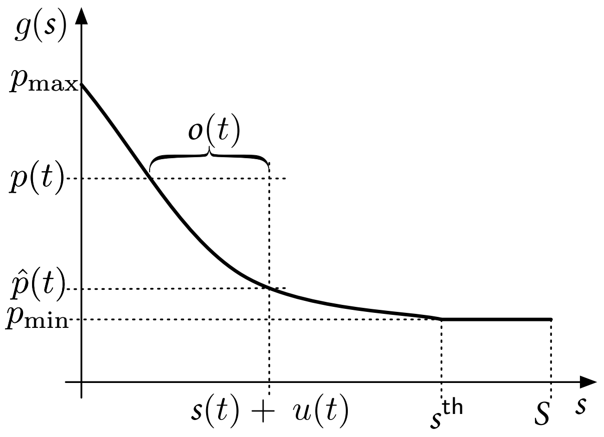


Fig. 3.3: Illustration of calculating the offering volume $\hat{o}(t)$ when $p(t) > \hat{p}(t)$.

Theorem 3.5.1: By setting $\hat{g}(s)$ in Equation (3.9) as

$$\hat{g}(s) = p_{\min} e^{\frac{(s^{\text{th}} - s)s^{\text{th}}}{S(S - s^{\text{th}})}}, \tag{3.10}$$

and s^{th} as

$$s^{\text{th}} = S - \frac{(2 + \log \hat{\theta})S - \sqrt{\log^2 \hat{\theta} + 4 \log \hat{\theta}} S}{2} > 0, \tag{3.11}$$

sOffAlg achieves the optimal competitive ratio for sOnOffer as

$$\text{CR}(\mathsf{sOffAlg}) = \frac{(2 + \log \hat{\theta}) + \sqrt{\log^2 \hat{\theta} + 4 \log \hat{\theta}}}{2}. \tag{3.12}$$

The proof is given as the competitive analysis in Sec. 3.5.4.

Remark. The theorem shows that the competitive ratio is proportional to a logarithmic function of $\hat{\theta}$ as the ratio between the maximum and minimum clearing prices. In practice, the scale of $\hat{\theta}$ varies from 2 to 50 in different markets, e.g., the clearing prices in PJM [11] and NYISO [8] in July, 2015 is in $[\$13.9, \$186.9]$ and $[\$8.1, \$43.1]$ per MWh. Given $\hat{\theta} = 50$, the competitive ratio is around 5.74. Our experimental results depict much lower empirical ratios using the real prices in different markets. For details we refer to Table 4.3.

Calculating the Offering Volume

Given the optimal function $g(s)$ in Theorem 4.4.1, we can finally find the offering volume $\hat{o}(t)$ as follows.

$$\hat{o}(t) = \begin{cases} [r(t) - \rho_c]^+, & \text{if } \hat{p}(t) > p(t), \\ s(t) + r(t) - \min\{s^{\mathsf{th}}, s(t) + \rho_c\}, & \text{if } \hat{p}(t) = p(t) = p_{\min}, \\ s(t) + r(t) - \min\{\hat{g}^{-1}(p(t)), s(t) + \rho_c\}, & \text{if } \hat{p}(t) < p(t), \end{cases} \tag{3.13}$$

where $\hat{g}^{-1}(p(t))$ is well-defined since $\hat{g}(s)$ defined in Equation (3.9) is monotonically decreasing. More specifically, when $\hat{p}(t) > p(t)$, SRGENCO stores the renewable supply as much as it can, considering the charging rate constraint ρ_c, then, the residual is committed to the market. If $\hat{p}(t) = p(t) = p_{\min}$, it means that the storage level exceeds the threshold level s^{th}, so, SRGENCO offers the minimum price $p_{\min}$. Since the market price is also $p_{\min}$ it commits the electricity until the storage level reaches the threshold s^{th} or $s(t) + \rho_c$, i.e., $\hat{o}(t) = s(t) + r(t) - \min\{s^{\mathsf{th}}, s(t) + \rho_c\}$. Finally, the last situation is $\hat{p}(t) < p(t)$, and $p(t) > p_{\min}$. In this case, we are in the exponential part of function $g(s)$ and $\hat{o}(t)$ is calculated as the total supply $r(t) + s(t)$ subtracted by $\min\{\hat{g}^{-1}(p(t)), s(t) + \rho_c\}$. An illustration of the offering volume in this case when $\hat{g}^{-1}(p(t)) \leq s(t) + \rho_c$ is depicted in Figure 3.3. Finally, to capture the maximum discharge rate of the storage, it suffices to modify offering volume as $\hat{o}(t) = \min\{r(t) + \rho_d, \hat{o}(t)\}$.

3.5.4 Competitive Analysis for Online Offering Algorithm

Our goal in this section is to design function $g(s)$ (especially $\hat{g}(s)$ and s^{th} in Equation (3.9)) so as to minimize the competitive ratio. By doing so we prove the result in Theorem 4.4.1. To simplify our explanation in this section and without loss of generality, we quantize the energy-related variables $o(t)$ and $s(t)$ and the input parameters $r(t)$ and S to take the integer values. More specifically, considering that S is a real number, we can define discretized capacity S_{d} as

$$S_{\mathsf{d}} = S/\delta, \tag{3.14}$$

where δ is the unit of electricity. In this way, S_{d} belongs to integer numbers. To avoid heavy notation complexity, however, in our analysis in this section, we abuse notation S to denote the discretized S_{d}. Similarly, we abuse $o(t), s(t)$, and $r(t)$ as their discretized versions. Finally, we assume that initially the storage is full.

An immediate consequence of the above discretization is that function $g(s)$ could be considered as a step function. We assume that there are N steps for $g(s)$, each of which indexed by i, and $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. Generally, the length of each step could be different, hence, let us denote by λ_i as the length of step i. We can characterize the threshold value $s^{\textsf{th}}$ defined in Equation (3.9) as the storage capacity subtracted by the length of the last step, i.e.,

$$s^{\textsf{th}} = S - \lambda_N. \tag{3.15}$$

Moreover, we define $\Lambda_i = \sum_{k=1}^{i} \lambda_k$ as the cumulative length until step i. Since $s \in [0, S]$ we get $S = \Lambda_N = \sum_{k=1}^{N} \lambda_k$. Finally, we denote $p_i = g(\Lambda_i)$ as the threshold price when the storage level is Λ_i under function $g(\cdot)$. By this definition, we get $p_1 = p_{\max}$ and $p_N = p_{\min}$.

By $s_{\boldsymbol{\omega}}^{\textsf{sOffAlg}}(t)$, we denote the storage level at slot t under $\textsf{sOffAlg}$, and a particular instance $\boldsymbol{\omega}$ as defined in Equation (3.8). Let us denote the minimum storage level that $\textsf{sOffAlg}$ reaches under $\boldsymbol{\omega}$ as $\bar{s} \in \{0, 1, \ldots, S\}$, i.e., $\min_{t\in\mathcal{T}} s_{\boldsymbol{\omega}}^{\textsf{sOffAlg}}(t) = \bar{s}$. Then, using this definition, we can partition the universal set of input instances Ω to multiple subsets as follows

$$\Omega = \bigcup_{\bar{s}\in\{0,1,\ldots,S\}} \Omega_{\bar{s}}^{\textsf{sOffAlg}},$$

where $\Omega_{b}^{\textsf{sOffAlg}} \triangleq \left\{\boldsymbol{\omega} \in \Omega : \min_{t\in\mathcal{T}} s_{\boldsymbol{\omega}}^{\textsf{sOffAlg}}(t) = \bar{s}\right\}$.

In particular, subset $\Omega_{\bar{s}}^{\textsf{sOffAlg}}$ represents the coalition of all input instances that results in the minimum storage level b upon executing the deterministic online algorithm $\textsf{sOffAlg}$.

Definition 3.5.1: Define the local competitive ratio $\textsc{cr}_{\bar{s}}(\textsf{sOffAlg})$ of $\textsf{sOffAlg}$ under the subset of input instances $\Omega_{\bar{s}}^{\textsf{sOffAlg}}$ as

$$\textsc{cr}_{\bar{s}}(\textsf{sOffAlg}) \overset{\text{def}}{=} \max_{\boldsymbol{\omega}\in\Omega_{\bar{s}}^{\textsf{sOffAlg}}} \frac{\textsf{Prof}_{\textsf{OPT}}(\boldsymbol{\omega})}{\textsf{Prof}_{\textsf{sOffAlg}}(\boldsymbol{\omega})}. \tag{3.16}$$

Given Definition 3.5.1, we can redefine $\textsc{cr}(\textsf{sOffAlg})$ as follows.

Definition 3.5.2: Define $\textsc{cr}$ $(\textsf{sOffAlg})$ as the maximum of $\textsc{cr}_{\bar{s}}(\textsf{sOffAlg})$ over all subsets $\Omega_{\bar{s}}^{\textsf{sOffAlg}}$, $\bar{s} \in \{0, 1, \ldots, S\}$, i.e.,

$$\textsc{cr}(\textsf{sOffAlg}) = \max_{\bar{s}\in\{0,1,\ldots,S\}} \textsc{cr}_{\bar{s}}(\textsf{sOffAlg}). \tag{3.17}$$

In [13], we justify that the based on the above definition the competitive ratio takes maximum value only among the subsets with

$\bar{s} = \Lambda_i$, $i \in \{1, 2, \ldots, N-1\}$, i.e.,

$$\textsc{cr}(\mathsf{sOffAlg}) = \max_{\bar{s} \in \{\Lambda_1, \Lambda_2, \ldots, \Lambda_{N-1}\}} \textsc{cr}_{\bar{s}}(\mathsf{sOffAlg}).$$

The following lemma characterizes a closed-form for $\textsc{cr}_{\bar{s}}(\mathsf{sOffAlg})$.

Lemma 3.5.1: For sOffAlg, if $\textsc{cr}_{\Lambda_i}(\mathsf{sOffAlg}) \geq \textsc{cr}_{\Lambda_k}(\mathsf{sOffAlg}), k \in \{i+1, i+2, \ldots, N-1\}$ and $i \leq N-1$, then we have:

$$\textsc{cr}_{\Lambda_i}(\mathsf{sOffAlg}) = \frac{p_i S + \sum_{k=i+1}^{N-1} p_k \lambda_k}{\sum_{k=i+1}^{N} p_k \lambda_k}. \tag{3.18}$$

otherwise,

$$\textsc{cr}_{\Lambda_i}(\mathsf{sOffAlg}) \geq \frac{p_i S + \sum_{k=i+1}^{N-1} p_k \lambda_k}{\sum_{k=i+1}^{N} p_k \lambda_k}. \tag{3.19}$$

We omit the proof of Lemma 3.5.1, since it is analogous to that in Chapter 2. Using the result in Lemma 2.5.2, we get the global competitive ratio of sOffAlg under universal set of instances as follows

$$\textsc{cr}(\mathsf{sOffAlg}) = \max_{i \in \{1, 2, \ldots, N-1\}} \frac{p_i S + \sum_{k=i+1}^{N-1} p_k \lambda_k}{\sum_{k=i+1}^{N} p_k \lambda_k}. \tag{3.20}$$

Our goal is to achieve the minimum possible value for $\textsc{cr}(\mathsf{sOffAlg})$. Our design space toward this goal is to find: (i) the optimal value for p_i which directly characterizes function $g(s)$, recall that by definition $p_i = g(\Lambda_i)$, and (ii) λ_N as the length of the last step in function which characterizes the threshold level s^{th}, recall that we have $s^{\mathsf{th}} = S - \lambda_N$.

The following lemma states that the minimum global competitive ratio is achieved when the value of local competitive ratios are all equal.

Lemma 3.5.2: Given a fixed $\lambda_1, \lambda_2, \ldots, \lambda_N$, $\textsc{cr}(\mathsf{sOffAlg})$ minimizes only if the following expression holds:

$$\frac{p_{N-1} S}{p_N \lambda_N} = \frac{p_{N-2} S + p_{N-1} \lambda_{N-1}}{p_N \lambda_N + p_{N-1} \lambda_{N-1}} = \cdots = \frac{p_1 S + \sum_{k=2}^{N-1} p_k \lambda_k}{\sum_{k=2}^{N} p_k \lambda_k}.$$

Using the result in Lemma 3.5.2 and by straightforward calculations, we can express λ_i as

$$\lambda_i = \frac{p_{i-1} - p_i}{p_i} \frac{p_n S \lambda_n}{p_{n-1} S - p_n \lambda_n}, i \in \{2, 3, \ldots, N-1\}. \tag{3.21}$$

Given $S = \sum_{i=1}^{n} \lambda_i$, and combining with (3.21), we get

$$S = \lambda_1 + \frac{p_N S \lambda_N}{p_{N-1} S - p_N \lambda_N} \sum_{i=2}^{N-1} \frac{p_{i-1} - p_i}{p_i} + \lambda_N.$$

Lemma 3.5.3: When $S \to \infty$, the competitive ratio of sOffAlg takes the minimum value only if $p_{N-1} \to p_N$.

Recall that S is the discretized version of the original storage capacity, and $S \to \infty$ could be achieved if we choose sufficiently small unit of electricity δ as in Equation (3.14). Given the results in Lemma 3.5.2 and Lemma 3.5.3, we have

$$\begin{aligned} \text{CR}(\mathsf{sOffAlg}) &= \frac{S}{\lambda_N} = \frac{\lambda_1}{\lambda_N} + \frac{S}{S - \lambda_N} \sum_{i=2}^{N-1} \frac{p_{i-1} - p_i}{p_i} + 1 \\ &\geq \frac{S}{S - \lambda_N} \log \hat{\theta} + 1. \end{aligned} \tag{3.22}$$

It can be verified that the above equation achieves the minimum value when $N \to \infty$, $\lambda_1 = 0$, and $\lambda_i = 1, i \in \{2, \ldots, N-1\}$. Now, if $s < S - \lambda_N$, we have

$$\begin{aligned} S - s &= \sum_{k=s}^{N} \lambda_k = \tfrac{p_N S \lambda_N}{p_{N-1} S - p_N \lambda_N} \textstyle\sum_{k=s}^{N-1} \tfrac{p_{k-1} - p_k}{p_k} + \lambda_N \\ &\approx \tfrac{S \lambda_N}{S - \lambda_N} \textstyle\int_{p_{\min}}^{p_{s-1}} \tfrac{1}{p} dp + \lambda_N \\ &= \tfrac{S \lambda_N}{S - \lambda_N} \log \tfrac{p_{s-1}}{p_{\min}} + \lambda_N, \end{aligned}$$

where the second-to-the-last equality holds since the difference between p_i and p_{i-1} would be arbitrarily small when $N \to \infty$ and $\lambda_i = 1, i \in \{2, \ldots, N-1\}$. Note that whenever $1 < s < S - \lambda_N$ and $g(s) = p_{s-1}$, solving the above equation we get the following closed-form for $g(s)$

$$g(s) = \begin{cases} p_{\min} & \text{if } s \geq S - \lambda_N, \\ p_{\min} e^{\frac{(S - \lambda_N - s)(S - \lambda_N)}{S l_N}} & \text{otherwise.} \end{cases} \tag{3.23}$$

Moreover, according to Equation (3.22), we have

$$\lambda_N = \frac{S}{\text{CR}(\mathsf{sOffAlg})} = \frac{S}{\frac{S}{S - \lambda_N} \log \hat{\theta} + 1},$$

and the closed form for λ_N is

$$\lambda_N = \frac{(2+\log\hat{\theta})S - \sqrt{\log^2\hat{\theta} + 4\log\hat{\theta}S}}{2}. \tag{3.24}$$

Putting together the results in Eqs. (3.23), (3.24), and (3.15), the result in Theorem 4.4.1 is proved.

3.6 Online Offering Strategy without Accurate Single-Slot Prediction

In Sec. 3.6.1, we first extend the previous result to the case that the clearing price $p(t)$ is unknown to SRGENCO when submitting the offer, however, the renewable output $r(t)$ is known accurately. Second, in Sec. 3.6.2, we extend the result to the general case that the renewable output is known to SRGENCO with forecasting error.

3.6.1 mOffAlg: sOffAlg with Multiple Offer Submissions; $p(t)$ Is Unknown, $r(t)$ Is Known

Our general approach in this scenario is to use the potentials of submitting *multiple offers* which is allowed in the current markets, e.g., in PJM market, the producers can submit at most 10 offers [4]. We again note that all the offers of producers with the offering prices less than the market clearing price are successful. Our approach in this case is to calculate total feasible commitment volume, and then partition this total amount into multiple offers, each of which conveying a portion of the offering volume, in different prices.

Let us define set $\mathcal{B}$ with ϵ elements as the offer set of the SRGENCO, where ζ is the maximum number of offers that is permitted by the market operator (e.g., $\zeta = 10$ in PJM market). Each element i in $\mathcal{B}$ consists of tuple (o_i, p_i) where o_i and p_i are the offering volume and offering price of offer $i = \{1, \ldots, \zeta\}$.

In Algorithm 3.2, we summarize the details of mOffAlg, as the multiple version of sOffAlg, when the clearing price is unknown. First, we present the formal definition of the total feasible commitment volume, denoted as $V(t)$ as follows:

$$V(t) = r(t) + \min\{s(t), \rho_d\}. \tag{3.25}$$

The total feasible commitment volume captures the maximum quantity that the SRGENCO can commit to the market for the next hour, which is the aggregation of the renewable output and the maximum amount that can be discharged from the storage, i.e., $\min\{s(t), \rho_d\}$.

Our general idea is to equally partition $V(t)$ into ϵ volumes, and submit one offer corresponding to each of them. The maximum charging rate ρ_c and the threshold value c^{th} are two important parameters that make the volume partitioning unequal in boundary points.

In Lines 3-8 of Algorithm 3.2, we set the offering volume and offering price for the first offer at the boundaries. The offer in Line 4 is for the case that total storage level after charging exceeds the s^{th} value. Hence, the surplus is offered with the minimum price. On the other hand, the offer in Line 6 offers the renewable output beyond the charging limit of storage level at the minimum price. Finally, the first offer is from the ones with minimum prices as constructed in Line 8.

Then in the second step as shown in Lines 10-16, in Line 12, the remaining quantity as declared by $V'(t)$ is divided into $\zeta - 1$ parts. Finally, for each offer we find the corresponding offering price according to Line 14 in which function $g(\cdot)$ is defined in Equation (3.10) and in Line 15 we add the offer to the set of offers.

Algorithm 3.2 mOffAlg $\left[r(t), s(t)\right]$

```
1  declare $\mathcal{B} \leftarrow \emptyset$ as the set of offers
2   Constructing the first offer at boundaries
3  if $\min\{r(t), \rho_c\} + s(t) > s^{\text{th}}$
4       $o_1(t) \leftarrow r(t) + s(t) - s^{\text{th}}$
5       else
6       $o_1(t) \leftarrow [r(t) - \rho_c]^+$
7  end if
8  $\mathcal{B} \leftarrow \{(o_1(t), p_{\min})\}$
9  Constructing the remaining offers
10 $V(t) \leftarrow r(t) + \min\{s(t), \rho_d\}$
11 $V'(t) \leftarrow V(t) - o_1(t)$
12 $\Delta o \leftarrow V'(t)/(\zeta - 1)$
13 for $i \leftarrow 1$ to $\zeta - 1$
14      $p_i \leftarrow g(V'(t) - i\Delta o)$
15      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\Delta o, p_i)\}$
16 end for
17 submit $\mathcal{B}$
```

Theorem 3.6.1 characterizes the competitive ratio of mOffAlg as a function of CR (sOffAlg) and the number of submitted offers ζ.

Theorem 3.6.1: The competitive ratio of mOffAlg is bounded by

$$\text{CR}(\mathsf{mOffAlg}) \leq \left(1 + \frac{\text{CR}(\mathsf{sOffAlg})\hat{\theta}}{\zeta^2}\right) \text{CR}(\mathsf{sOffAlg}).$$

Note that as $\zeta \to \infty$ we have $\text{CR}(\mathsf{mOffAlg}) \to \text{CR}(\mathsf{sOffAlg})$. In experiments, we evaluate the impact of the number of offers on the performance of mOffAlg.

3.6.2 gOffAlg: Generalized mOffAlg; $u(t)$ Is Given with Forecasting Error, $p(t)$ Is Unknown

Finally, we release the accurate forecasting assumption of the renewable output, and assume that the error of power generation in the forthcoming slot is bounded in a particular region. Namely the input to this algorithm is $\tilde{r}(t)$ as the predicted value and $\kappa(t)$ as the maximum error, such that

$$r(t) \in [(1-\kappa(t))\tilde{r}(t), (1+\kappa(t))\tilde{r}(t)],\ 0 \leq \kappa(t) \leq \kappa_{\max}.$$

Our algorithm gOffAlg is a simple extension of mOffAlg, with replacing $r(t)$ with $(1-\kappa(t))\tilde{r}(t)$ as the minimum possible value for the renewable output. We skip the other details since they are the same as mOffAlg. Note that with this input, the algorithm behaves in the most conservative way, such that the over-commitment never happens. Extending the algorithm to the more aggressive cases that takes into account the risk of over-commitment is part of our future work.

In following theorem we characterizes the competitive ratio of gOffAlg.

Theorem 3.6.2: Assuming $\kappa_{\max} < 0.5$, the competitive ratio of gOffAlg is bounded by $\textsc{cr}(\text{gOffAlg}) \leq \frac{1}{1-2\kappa_{\max}}\textsc{cr}(\text{mOffAlg})$.

Proof of Theorem 3.6.2. At each round, the bidding amount of gOffAlg is at least $1-2\kappa_{\max}$ times of that of mOffAlg. Therefore, the competitive ratio of gOffAlg is bounded by $\frac{1}{1-2\kappa_{\max}}\textsc{cr}(\text{mOffAlg})$. $\square$

Clearly, as $\kappa_{\max} \to 0$, we get $\textsc{cr}(\text{gOffAlg}) \to \textsc{cr}(\text{mOffAlg})$. In experiments, we investigate the impact of forecasting error on the result of gOffAlg.

3.7 Experimental Results

In this section, we evaluate the performance of our online strategies using the real-word traces for the renewable output and electricity market prices. Our objective is two-fold: (i) to compare the performance against the optimal offline, a comparison algorithm [112], and a baseline in which there is no storage, and (ii) to investigate the impact of the system model and algorithm parameters.

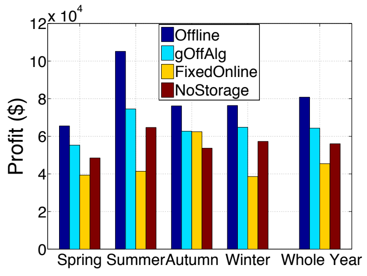


Fig. 3.4: The performance of different algorithms in different seasons

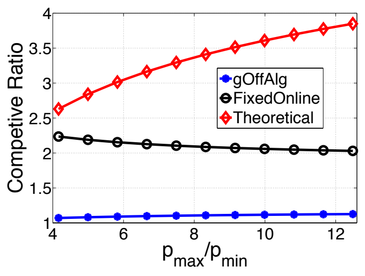


Fig. 3.5: The competitive ratio as a function of price volatility

3.7.1 Experimental Settings

Renewable Output and Electricity Market Prices

We use the wind generation data from PJM energy market [11] with the capacity of 10MW. We note that this is a wind farm in moderate size and the largest one has an operational capacity of 1020MW [15]. The hourly electricity price data are from PJM market for most of the experiments. We also demonstrated our results for Nord Pool [9] and NYISO [8] markets in Table 4.3. We note that the prices exhibit severe seasonal patterns. In particular, the prices are highly volatile, e.g., in summer the peak price can be as high as $\$396.9/$MWh. For this reason we evaluate the performance of our algorithms in different seasons as well.

Storage Capacity

Unless otherwise specified, the storage capacity is set to 20MWh. The maximum charge and discharge rates are 10MW. In reality, large scale Compressed Air Energy Storage (CAES) with similar parameters has been developed to cope with renewable uncertainty [114].

Parameters for the Algorithms

Unless otherwise specified, for mOffAlg and gOffAlg, the default value for the number of offers is 10. This is the common practice in PJM market [4]. Moreover, the maximum forecasting error $e_{\max}$ in gOffAlg is set to 10%. Finally, we note that each data point in figures is the average results of 100 different runs of the algorithms with $T = 360$ hours.

Tab. 3.2: Summary of considered algorithms

Algorithm	Description
	Our algorithms
sOffAlg	Simplified online offering strategy; $p(t)$ and $r(t)$ are known
mOffAlg	sOffAlg with multiple submissions; $p(t)$ is unknown, $r(t)$ is known
gOffAlg	Generalized mOffAlg; $p(t)$ is unknown, $r(t)$ is known with error
	Comparison Algorithms
OPT	Optimal offline solution with storage
NoStorage	Optimal offline solution without storage
FixedOnline	Simple online algorithm with fixed threshold price [112]

Comparison among Algorithms

We compare the performance of our algorithm gOffAlg with three other alternatives: (1) OPT, the optimal offline solution that is implemented as the benchmark to obtain the empirical competitive ratio; (2) NoStorage, the optimal offline cost when there is no storage. This is used to evaluate the economic advantage of integrating the storage; (3) FixedOnline, another simple online algorithm with a fixed threshold price. Specifically, we follow the approach in [112] and set the threshold of this simple online algorithm fixed at $\sqrt{p_{\min} p_{\max}}$, regardless of the storage level. In this algorithm, SRGENCO commits all the electricity whenever the price is not smaller than this threshold. The acronyms for all the algorithms are summarized in Table 4.2.

3.7.2 Experimental Results

Comparison of Results across Different Seasons

In this experiment we report the profit obtained by different algorithms in different seasons as well as the whole year. The result is depicted in Figure 3.4. The main observations are: (1) gOffAlg achieves 80% of the offline optimum, which shows that it is close to optimal. (2) gOffAlg outperforms NoStorage by 15%, which signifies the substantial economic benefit of incorporating the storage. (3) gOffAlg outperforms FixedOnline by 42%, which depicts the superiority of our online algorithms as compared to other "storage-level-oblivious" online alternatives.

Tab. 3.3: Summary of Theoretical and Empirical Competitive Ratios on Different Electricity Markets

Market	$\hat{\theta} = p_{\max}/p_{\min}$	Theoretical CR	Empirical CR
PJM	13.44	4.37	1.18
NYISO	5.32	3.38	1.14
Nord Pool	3.63	2.95	1.09

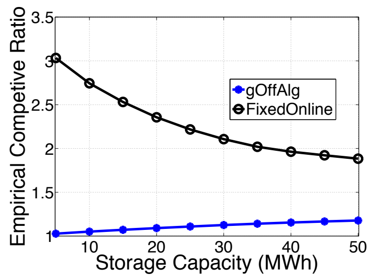


Fig. 3.6: The competitive ratio as a function of storage capacity

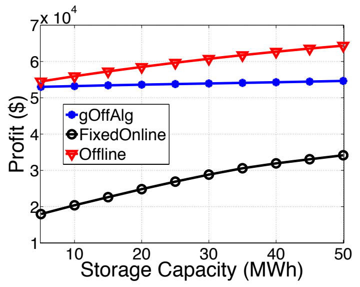


Fig. 3.7: The performance of different algorithms as a function of capacity

Impact of the Price Volatility

The electricity price in the deregulated electricity market exhibits large fluctuation. Theoretically, large price volatility will degrade the performance of the online algorithm, as the competitive ratio is an increasing function of $\hat{\theta} = p_{\max}/p_{\min}$. In this experiment, we present the result under different values of $\hat{\theta}$. As shown in Figure 3.5, gOffAlg is robust to price fluctuation with less than 5% increment, even though the theoretical competitive ratio increases by 44%. Meanwhile, we note that the empirical competitive ratio of FixedOnline decreases slightly as the $\hat{\theta}$ increases. However, it is on average 90% larger than that of gOffAlg, which further signifies the superiority of gOffAlg. In addition, we report the result of gOffAlg for the prices in different markets in Table 4.3. The result signifies that the larger the price volatility, the large theoretical and empirical competitive ratios.

Impact of the Storage Capacity

Storage capacity planning is an important issue that SRGENCO's owner needs to consider, since the storage is still expensive with the current technology. In this experiment, we vary the storage capacity from 5 to 50MWh to investigate its impact on the profit of SRGENCO. Figure 3.6 and Figure 3.7 show the empirical competitive ratios and the obtained profits, respectively. As the storage capacity increases, an increase in profit of both online algorithms is observed. However, the increase in gOffAlg is smaller (3%) than that of FixedOnline (90%). This is mainly because FixedOnline is completely oblivious to the storage level, and with the increase in capacity, there would be more room to mitigate this unawareness. Meanwhile, the empirical competitive ratio of gOffAlg increases with large storage capacity (from 1.03 to 1.18). This result depicts that when the capacity is in the order of the renewable capacity (say, $\times 0.5$ to $\times 2$), gOffAlg is close-to-optimal. However, when the storage capacity is much higher than the renewable capacity (say, $\times 5$), perhaps more sophisticated algorithms are required.

Impact of Uncertainty of Clearing Price and Renewable Output

In the last set of experiments we investigate the impact of the number of offers in mOffAlg (in Figure 3.8) and the forecasting error in gOffAlg (in Figure 3.9). In mOffAlg, we relax the assumption of sOffAlg and extend it to the case that the clearing price $p(t)$ is unknown. We proposed to submit multiple offers to alleviate its negative impact. To investigate how many offers are sufficient for mOffAlg to achieve the same performance level as sOffAlg, in Figure 3.8, we vary the number of offers from 1 to 15. The notable observation is that submitting 1 or 2 offers is not sufficient. However, with 3 or more offers the performance is quite similar to sOffAlg in which the price is known in advance. In the last experiment, we increase the maximum error of renewable output $e_{\max}$ and calculate the profit of gOffAlg. The result shows that gOffAlg is robust to forecasting error that belows 20%, and the obtained profit decreases rapidly as error increases beyond 20%. Concluding above, these experiments demonstrate that the negative impact of the uncertainty in the clearing price can be effectively mitigated by multiple offer submissions. However, accurate short-term renewable forecasting is vital for SRGENCO to obtain a desired profit, since the errors higher than 20% can severely degrade the performance.

□ **End of chapter.**

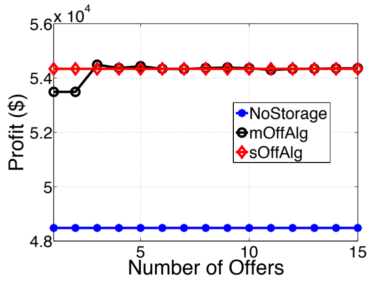


Fig. 3.8: The performance of mOffAlg as the number of bids increases

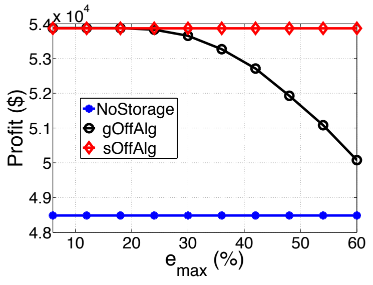


Fig. 3.9: The performance of gOffAlg as forecasting error increases

4. ONLINE TRADING II: ONLINE PROCURING OF INDIVIDUAL CONSUMERS

4.1 Problem Background

Electricity bill constitutes a significant portion of operational costs for large scale data centers. Empowering data centers with on-site storages can reduce the electricity bill by shaping the energy procurement from deregulated electricity markets with real-time price fluctuations. In this chapter, we focus on designing energy procurement and storage management strategies to minimize the electricity bill of storage-assisted data centers. Designing such strategies is challenging since the net energy demand of the data center and electricity market prices are not known in advance, and the underlying problem is coupled over time due to evolution of the storage level. Using competitive ratio as the performance measure, we propose an online algorithm that determines the energy procurement and storage management strategies using a threshold based policy. Our algorithm achieves the optimal competitive ratio as a function of the price fluctuation ratio. We validate the algorithm using data traces from electricity markets and data-center energy demands. The results show that our algorithm achieves close to the offline optimal performance and outperforms existing alternatives.

4.2 Summary of Main Results and Adopted Techniques

Generally speaking, the online procuring problem comes with two sets of uncertain input parameters, which allow the adversary to have more options in constructing the worst-case input. Therefore direct application of existing algorithms can only guarantee sub-optimal competitive ratios. The most relevant work, perhaps, is [48] where a similar problem with a slightly different system model was studied. Even though the solution in [48] is a competitive online algorithm, it is oblivious of the state of storage in decision making, leading to a sub-optimal competitive ratio.

By introducing the system model in Sec. 4.3, this thesis tackles the energy procurement and storage management problem by making the following constructions:

▷ In Subsection 4.4.4, we propose an online algorithm, the ONCOM, that determines the quantity of purchased energy from the real-time electricity market at each time slot based on the current energy price. To handle the uncertainty of energy demand, the ONCOM builds a set of virtual storages each of which corresponds to the demand at each slot. Then, we associate to each virtual storage a threshold-based admission strategy that determines the purchased quantity as a function of storage level. At each slot, the ONCOM procures electricity from the market only if the price is lower than the candidate price obtained from the threshold function. The ONCOM ensures fulfilling the energy demand and in some slots that the market price is cheap, it even buys electricity from the market to store it in the storage for future consumptions.

▷ In Subsection 4.4.5, we analyze the competitiveness of the ONCOM. We construct our analysis on top of the optimal competitive analysis for k-min search problem [112], as a simplified version of our problem by neglecting the uncertain demand. We show that by leveraging the notion of virtual storages, the ONCOM achieves the optimal competitive ratio of $1/(W(-\frac{\hat{\theta}-1}{\hat{\theta}\exp(1)})+1)$, where $\hat{\theta}$ is the price fluctuation ratio, and $W(\cdot)$ is Lambert-W function (defined as the inverse of $f(x) = x\exp(x)$, and grows in logarithmic order).

▷ In Sec. 4.5, we extend the basic solution to the case when the market price is not known even for the current slot. This makes the problem the one of designing online bidding strategy in real-time markets. We advocate the idea of submitting multiple bids, each with different bidding price and quantity. We also extend our solution to the more sophisticated battery models by incorporating maximum charging and discharging rates.

▷ In Sec. 4.6, the ONCOM is verified using real data traces from several electricity markets and data center energy demand. The results show that the cost of the ONCOM is $\times 1.23$ of the offline optimum, on average in different markets, which is much better than the theoretical "worst-case" competitive ratio. Our algorithm also outperforms the three other existing solutions [48, 78, 112] significantly.

4.3 Problem Description

4.3.1 Market Model

We consider that a data center participates in a deregulated real-time electricity market. We assume that the time horizon is divided into T real-time settlement intervals, indexed by t, each with fixed lengths, e.g.,

5 minutes in CAISO, NYISO, and SPP; 15 minutes in ERCOT; and 1 hour in ISO-NE, PJM, and MISO [16]. The independent system operator (ISO) posts the electricity price $p(t) \geq 0$ for the next slot shortly before the actual operation of the slot. The values of $p(t)$ are unknown for the future slots. Section 4.5.1, considers the setting in which the price $p(t)$ is unknown even for the current slot. This turns our formulation into an online bidding strategy design problem. Let $p_{\max}$ and $p_{\min}$ denote the maximum and minimum electricity prices, whose values are known.[1] Our analysis characterizes the competitive ratio as a function of $\hat{\theta}$.

4.3.2 Data Center Net Energy Demand

Let $d(t) \geq 0$ be the net energy demand of data center at slot t. By net demand, we mean the total energy demand subtracted by the (potential) local renewable supply. We assume that $d(t)$ is known for the current slot. However, the net demand is unpredictable for the future slots due to uncertainty in the workload of the datacenter and uncertainty of the renewable supply. Hence, we assume that the values of $d(t)$ are unknown for future slots beyond t. Furthermore, our model does not rely on any stochastic modeling of net demand and market price.

4.3.3 Energy Procurement and Storage Management Scenario

The scenario is depicted in Figure 4.1. We assume that the data center is equipped with an on-site storage to reduce the electricity bill by shaping the power consumption. Given demand $d(t)$ and price $p(t)$, the goal is to determine quantity $e(t) = e_s(t) + e_d(t)$ as the purchased energy from the electricity market to either satisfy the net demand of data center, denoted as $e_d(t)$, or charge the storage, denoted as $e_s(t)$, in cases that the electricity price is cheap. In addition, demand $d(t)$ can be covered by both electricity market and on-site storage, i.e., $d(t) = e_d(t) + s_d(t)$, where $s_d(t)$ is the energy discharged from storage to satisfy the demand.

4.3.4 Storage Model

Let S be the maximum storage capacity. In practice, storage systems have maximum charging and discharging rates. To avoid notation and solution complexities, we proceed to formulate the problem without these parameters. Our solution design, however, can be extended to the general storage models without fundamentally changing the algorithms as discussed in Section 4.5.2.

[1] This assumption is reasonable since by the historical data of electricity prices, the maximum and minimum market prices could be found.

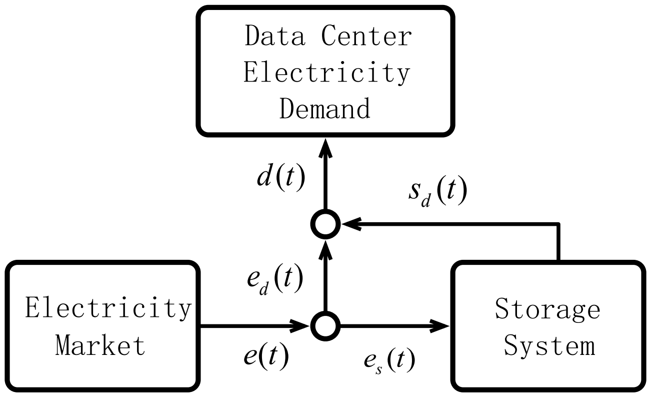


Fig. 4.1: The energy procurement and storage management scenario

Tab. 4.1: Summary of key notations related to the online procuring problem

Notation	**Description**
t	Index of each time slot
T	The number of time slots, $T \geq 0$
$\mathcal{T}$	Set $\mathcal{T} = \{1, 2, \ldots, T\}$
$p_{\max}, p_{\min}$	The maximum and minimum electricity price
$p(t)$	Electricity price at t, $p_{\min} \leq p(t) \leq p_{\max}$, known for t, unknown for $\tau \in \mathcal{T} : \tau > t$
$\hat{\theta}$	The price fluctuation ratio, $\hat{\theta} = p_{\max}/p_{\min}$
S	The capacity of storage system
$s(t)$	The storage level (state of charge) at the end of t
$d(t)$	The net energy demand (in kWh), known for t, unknown for $\tau \in \mathcal{T} : \tau > t$
$e(t)$	The total procured energy at t from market
$e_s(t)$	The amount of procured energy charged into the storage at t
$e_d(t)$	The amount of procured energy used to satisfy the demand at t

Let $s(t)$ be the storage level at the end of slot t. Given the net demand $d(t)$, the evolution of storage level is given as

$$s(t) = \Big[s(t-1) + e_s(t) - s_d(t)\Big]_{\mathcal{S}}, \tag{4.1}$$

where $[.]_{\mathcal{S}}$ defines the projection onto set $\mathcal{S} = [0, S]$. Note that at each slot either $e_s(t)$ or $s_d(t)$ is zero, which dictates either charging or discharging of the storage. More specifically, if $e(t) > d(t)$, then $e_s(t) > 0$ and $s_d(t) = 0$. On the other hand, if $e(t) < d(t)$, then $e_s(t) = 0$ and $s_d(t) > 0$. Hence, given $e(t)$, it is straightforward to calculate $e_d(t)$, $e_s(t)$, and $s_d(t)$, thereby we formulate the problem by considering $e(t)$ as the only optimization variable.

Problem Formulation

We formulate the energy procurement and storage management problem (called ECOM, as Energy COst Minimization problem) as follows

$$\begin{aligned}
\text{ECOM} \quad \min \quad & \sum_{t\in\mathcal{T}} p(t)e(t) \\
\text{s.t.}: \quad & \forall t \in \mathcal{T}: \\
& e(t) \geq d(t) - s(t-1), && (4.2)\\
& s(t) = \Big[s(t-1) + e(t) - d(t)\Big]_{\mathcal{C}}, && (4.3)\\
\text{vars.}: \quad & e(t) \geq 0, \quad t \in \mathcal{T}.
\end{aligned}$$

In the ECOM, the goal is to find $e(t)$ (as the optimization variable) at the beginning of each slot, such that the long-term electricity cost is minimized (as the objective) and at the same time data center net demand is covered and capacity constraint of storage is respected (as the constraints). Note that the storage level expression in Equation (4.3) is equivalent to Equation (4.1).

The ECOM is a linear problem that could be easily solved in offline setting when the entire inputs $p(t)$ and $d(t)$ are given ahead of time. The real-world practical setting, however, is online, due to online arrival of price $p(t)$ and net demand $d(t)$. In this thesis, we follow online competitive algorithm design [39] which does not rely on exact or stochastic modeling of future inputs. For the performance measure, we characterize the *competitive ratio* of our solution, defined as the maximum possible ratio between the cost of the online algorithm and the offline optimum, over the whole set of instances.

We focus on designing energy procurement and storage management strategies to minimize the electricity bill of storage-assisted data centers. Designing such strategies is challenging since the net energy demand of the data center and electricity market prices are not known in advance, and the underlying problem is coupled over time due to evolution of the storage level.

4.4 Online Solution

4.4.1 The ECOM as an Extension of k-min Search Problem

The ECOM could be considered as an extension of k-min search problem. In this problem, a player wants to buy $k \geq 1$ units of an asset with the goal of minimizing the cost. At any slot $t = \{1, \ldots, T\}$, the player is presented a price $p(t)$, and must immediately decide whether or not to buy one unit of the asset given $p(t)$.

In the ECOM, similar to k-min search, price $p(t)$ arrives in slot-by-slot fashion, and the storage capacity is similar to available budget of procuring k units of an asset (energy in our case). Different from k-min search, however, in the ECOM, the net energy demand is another uncertain input to the problem, which could be considered roughly as the *dynamics in available budget* for the asset in k-min search problem. The new uncertain input enables the adversary to have more flexibility to construct worst-case input sequence.

Our solution is built upon the optimal online algorithm proposed in [112] for k-min search problem in which a threshold function as a function of remaining budget is devised. In the algorithm proposed in [112], if the posted price $p(t)$ is *lower* than the value obtained form the threshold function, it buys one unit of the asset. In the context of the ECOM, the high level idea is to determine the procurement quantity based on the available energy in the storage using a threshold function. More specifically, if the storage level is almost full, it would be of interest to satisfy the demand locally by discharging the storage, hence, buying electricity is beneficial only if the market price is very cheap. On the other hand, if the storage level is almost empty, the only option is to buy the electricity form the market even with high price, since the local storage is unable to satisfy the entire net demand.

In the next section, we extend the optimal online algorithm for k-min search to the ECOM. Since the net energy demand of data center is similar to dynamics in available budget of buying asset in k-min search, our algorithm constructs several *virtual storages* each corresponding to the demand at each slot. Then, for each virtual storage we construct a specific threshold function which is a scaled version of the one for k-min search according to the capacity of virtual storages.

4.4.2 Constructing Virtual Storages

Without loss of generality, we assume that the initial state of the battery is 0. Our algorithm, named ONCOM, constructs multiple virtual storages as follows. Initially, it builds a virtual storage of capacity S, corresponding to the original battery. Afterwards, at each time slot t, the ONCOM creates a logical virtual storage of capacity $d(t)$. Consequently, at slot t, there are $t+1$ virtual storages; the first one corresponds to the original storage, and the remaining t virtual storages each corresponds to slots in $\{1, \dots, t\}$. Given the above virtual storage construction, we can record both the initial storage (of capacity S) and the storage spaces created due to the demand from the data center. In this way, the history of purchased energy and demand is logically "stored" among those virtual storages.

Putting together the storage level of all virtual storages, we define $\boldsymbol{s}(t)$ as the *algorithm state* at time slot t, whose elements are the states of the virtual storages, i.e.,

$$\boldsymbol{s}(t) = \left[s_0(t), s_1(t), \ldots, s_t(t)\right], \tag{4.4}$$

where $s_0(t)$ is the storage level of the original storage, and $s_i(t), i \in \{1, \cdots, t\}$ is virtually the amount of energy stored at slot t for the virtual storage created at slot i.

According to this construction, the aggregate amount of electricity in virtual storages, i.e., $\sum_{i=0}^{t} s_i(t)$, corresponds to total procured electricity, and the aggregate capacity of virtual storages except the first one, i.e., $\sum_{i=1}^{t} d(i)$, corresponds to the aggregate demand over $[1, t]$. Thus, we have

$$s(t) = \sum_{i=0}^{t} s_i(t) - \sum_{i=1}^{t} S_i, \tag{4.5}$$

as the storage level defined in Equation (4.3), where $S_i = d(i)$ is the capacity of i-th virtual storage. Note that when $s(t) = 0$, the algorithm will be set back to the initial state.

4.4.3 The Threshold-based Procurement Policy of the OnCoM

The energy procurement policy of the OnCoM for each single storage is determined using a threshold function $g_i(s_i(t))$ with respect to $s_i(t)$ as the state of charge of storage i. The threshold function generates a value such that if the market price $p(t)$ is less than the value, it purchases energy. Intuitively, the threshold function is decreasing, i.e., as the storage level increases, the threshold price decreases since there is sufficient energy in storage to fulfill the demand.

By optimizing threshold function $g_i(s_i)$[2], we can achieve an intriguing competitive ratio. The optimal design of the threshold function and the performance results are summarized in the following theorem. The proof is given in Sec. 4.4.5.

Theorem 4.4.1: Given the threshold function

$$g_i(s_i) = p_{\max}\left[1 - \left(1 - \frac{1}{\alpha^\star}\right)\exp\left(\frac{s_i}{\alpha^\star S_i}\right)\right], \; s_i \in [0, S_i], \tag{4.6}$$

for virtual storage $i \in [0, t]$, the admission policy of the OnCoM

[2] Note that we drop index t, to avoid notation complexity.

achieves the competitive ratio of $\alpha^\star$ given by

$$\alpha^\star = \left(W\left(-\frac{\hat{\theta}-1}{\hat{\theta}\exp(1)} \right) + 1 \right)^{-1}, \tag{4.7}$$

as the unique solution of

$$\frac{1-1/\hat{\theta}}{1-1/\alpha^\star} = \exp\left(\frac{1}{\alpha^\star}\right). \tag{4.8}$$

Moreover, the competitive ratio $\alpha^\star$ is optimal, meeting the optimal competitive ratio for the k-min search problem.

Remark 4.4.1: In Equation (4.7), W denotes *Lembert-W function* defined as inverse of $f(x) = x\exp(x)$. It is well known that $W(x) \approx \ln x$ [6]. The theorem shows that the competitive ratio is approximately proportional to a natural logarithm of θ as the price fluctuation ratio. In practice, the scale of θ varies substantially in different markets, e.g., the real-time prices in MISO [16] and NYISO [8] in June 2017 varied in $[\$11.09, \$73.08]$ and $[\$9.08, \$65.26]$. Given $\hat{\theta} = 40$, the competitive ratio is $\alpha^\star = 1 + W((40-1)/e) = 2.98$. Our experimental results (see Table 4.3) demonstrate much lower empirical ratios using the real prices in different markets.

4.4.4 Determining the Procurement Quantity

Given the threshold function in Equation (4.6), the final step is to determine quantity $e(t)$ as the optimization variable. Let $e_i(t)$ be the procured amount for storage i at slot t as follows

$$e_i(t) = \begin{cases} 0, & \text{if } p(t) \geq g_i(s_i(t-1)) \\ g_i^{-1}(p(t)) - s_i(t-1), & \text{if } p(t) < g_i(s_i(t-1)) \end{cases} \tag{4.9}$$

In Equation (4.9), when $p(t) \geq g_i(s_i(t-1))$, it means that the market price is above the candidate threshold price and it is not beneficial to purchase electricity form the market, hence, we set $e_i(t) = 0$. Otherwise, it purchases the electricity up to the level that the remaining state matches the market price. Intuitively, Figure 4.2 illustrates the calculation of $e_i(t)$ when $p(t) < g_i(s_i(t-1))$. Note that $g_i(s_i)$ is decreasing with respect to s_i, and we have $g_i(0) = p_{\max}/\alpha^\star$ and $g_i(S_i) = p_{\min}$.

Having the values of $e_i(t)$ for all virtual storages, the last step is to determine the aggregate energy procurement $e(t)$. To satisfy

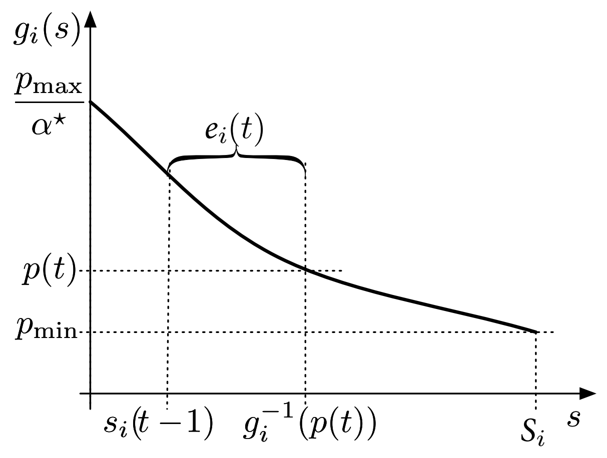


Fig. 4.2: The illustration of $g_i(s_i)$ and determining $e_i(t)$ when $p(t) < g_i(s_i)$.

constraint (4.2), i.e., $e(t) \geq d(t) - s(t-1)$, we calculate $e(t)$ as follows

$$e(t) = \max\left\{\sum_{i=0}^{t} e_i(t), d(t) - s(t-1)\right\}.$$

When $\sum_{i=0}^{t} e_i(t) < d(t) - s(t-1)$, the algorithm will be set back to the initial state, i.e., $s(t) = 0$. Otherwise, the ONCOM logically allocates $e_i(t)$ units of energy to storage i.

Based on the above construction, we conclude our proposed algorithm ONCOM and summarize it as Algorithm 4.1.

Algorithm 4.1 Energy Procurement Policy of ONCOM: $\forall t \in \mathcal{T}$

```
1  At each time slot $t$:
2  $s_t(t) \leftarrow 0$
3  for $i = 0$ to $t$
4      if $p(t) \geq g_i(s_i(t))$
5          $e_i(t) \leftarrow 0$
6      else
7          $e_i(t) \leftarrow g_i^{-1}(p(t)) - s_i(t-1)$
8          $s_i(t) \leftarrow s_i(t-1) + e_i(t)$
9      end if
10 end for
11 $e(t) \leftarrow \max\left\{\sum_{i=0}^{t} e_i(t), [d(t) - s(t-1)]^+\right\}$
12 if $s(t) = 0$
13     Initialize the algorithm state: $\boldsymbol{s}(t) \leftarrow [0]$
14 else
15     $\boldsymbol{s}(t) \leftarrow [s_0(t), s_1(t), \ldots, s_t(t)]$
16 end if
```

4.4.5 Competitive Analysis

Recall that without energy demand of data center, i.e., $d(t) = 0, t \in \mathcal{T}$, the ECOM degenerates to the classic k-min search problem [112], whose competitive ratio is lower bounded by $\alpha^\star$ as characterized in Equation (4.8). A direct consequence is that $\alpha^\star$ is a lower bound for the ECOM as a generalized k-min search problem. In this section, we demonstrate that under the non-trivial extension of uncertain demand, the ONCOM achieves the competitive ratio of $\alpha^\star$, meeting the lower bound for the competitive ratio of the ECOM such that $\alpha^\star$ is the optimal competitive ratio of the ECOM as well.

The key idea in competitive analysis is to intelligently partition the entire time horizon into several *cycles*. Then, we prove that to achieve the competitive ratio for the ECOM over entire time horizon, it suffices to find the competitive ratio over one cycle. Based on this observation, it is enough to search the worst instance among ones only containing one cycle. The detailed analysis is as follows.

Cycle Partitioning

Define a *cycle* $\mathcal{T}_c = [t' + 1, t] \subseteq \mathcal{T}$, as the time interval between two time slots when the storage begins to store energy until it gets fully discharged. Each cycle is further divided into two periods: Period I in which the storage level is strictly positive, i.e., $s(t) > 0, t \in \mathcal{T}_c$, and Period II, in which the storage keeps unoccupied, i.e., $s(t) = 0, t \in \mathcal{T}_c$. An example of a cycle and two periods is shown in Figure 4.3.

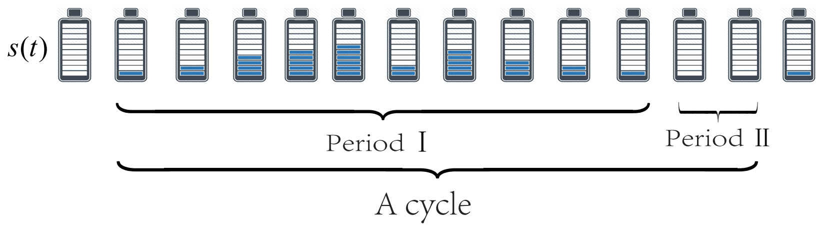


Fig. 4.3: Duration of a *cycle*.

Let $s^{\text{ONCOM}}(t)$ and $s^{\mathsf{OPT}}(t)$ be the storage level at slot t by executing ONCOM and optimal offline solution over the *worst case instance*, respectively.

Consider the following situation in which $s^{\text{ONCOM}}(t-1) = 0$ and $s^{\mathsf{OPT}}(t-1) > 0$. Then, the adversary constructs a worst case instance as follows. It sets the market price to the maximum value, i.e., $p(t) = p_{\max}$, and the energy demand to the remaining energy in the storage, i.e., $d(t) = s^{\mathsf{OPT}}(t-1) > 0$. In the above worst case instance, the remaining energy under OPT is fully utilized, while the ONCOM must procure the entire demand from the market at the maximum price. An important observation in above worst case instance construction is that the storage in both ONCOM and OPT gets fully discharged by the end of a cycle. The following result shows that the competitive analysis for the ONCOM can be reduced to one cycle.

Lemma 4.4.1: Let $\boldsymbol{\omega}$ be the worst instance and $\mathcal{C}$ be a cycle realized by the ONCOM under $\boldsymbol{\omega}$. The cost ratio during $\mathcal{C}$ is exactly equal to the competitive ratio of the ONCOM.

Proof of Lemma 4.4.1. Since $\boldsymbol{\omega}$ is the worst-case instance, we write the competitive ratio of the ONCOM as $\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}}/\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}}$ as the cost ratio between the cost of the ONCOM and the OPT given $\boldsymbol{\omega}$ as the input to the ECOM. In addition, let $\mathsf{cost}_{\mathcal{T}_c}^{\text{ONCOM}}/\mathsf{cost}_{\mathcal{T}_c}^{\mathsf{OPT}}$ be the cost ratio of the ONCOM and the OPT given cycle $\mathcal{T}_c = [t'+1, t]$ as the input. We prove the result by contradiction using the following two cases:

Case (1): $\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}}/\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}} > \mathsf{cost}_{\mathcal{T}_c}^{\text{ONCOM}}/\mathsf{cost}_{\mathcal{T}_c}^{\mathsf{OPT}}$.

In this case in cycle $\mathcal{C} = [t'+1, t]$ the maximum cost ratio is smaller than the competitive ratio of the ONCOM. At time slots t' and t, the storage under both ONCOM or OPT is empty. Then, we can "remove" the input segment over $\mathcal{C}$ and present the instance

$$\boldsymbol{\omega'} = [(p(\tau), d(\tau))]_{\tau=1:t'} \oplus [(p(\tau), d(\tau))]_{\tau=t+1:T},$$

where $\oplus$ in above equation denotes the concatenation of inputs over the slots. By the above replacement, the cost ratio of $\boldsymbol{\omega'}$ is

$$\frac{\mathsf{cost}_{\boldsymbol{\omega'}}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega'}}^{\mathsf{OPT}}} = \frac{\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}} - \mathsf{cost}_{\mathcal{T}_c}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}} - \mathsf{cost}_{\mathcal{T}_c}^{\mathsf{OPT}}} > \frac{\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}}}, \tag{4.10}$$

where the last inequality follows from $\frac{x_1-x_3}{x_2-x_4} > \frac{x_1}{x_2}$, if $\frac{x_1}{x_2} > \frac{x_3}{x_4}$, $x_1, x_2, x_3, x_4 > 0, x_1 > x_3, x_2 > x_4$. Consequently, Equation (4.10) shows that there is another instance $\boldsymbol{\omega'}$ whose cost ratio is larger than $\boldsymbol{\omega}$, contradicting that $\boldsymbol{\omega}$ is the worst case input.

Case (2): $\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}}/\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}} < \mathsf{cost}_{\mathcal{T}_c}^{\text{ONCOM}}/\mathsf{cost}_{\mathcal{T}_c}^{\mathsf{OPT}}$.

In this case, we construct another input instance by "inserting" one additional cycle in the middle of $\boldsymbol{\omega}$ as follows

$$\begin{aligned} \boldsymbol{\omega'} &= [(p(\tau), d(\tau))]_{\tau=1:t} \\ &\oplus [(p(\tau), d(\tau))]_{\tau=t'+1:t} \\ &\oplus [(p(\tau), d(\tau))]_{\tau=t+1:T}. \end{aligned}$$

By constructing $\boldsymbol{\omega}'$, its cost ratio is

$$\frac{\mathsf{cost}_{\boldsymbol{\omega}'}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega}'}^{\mathsf{OPT}}} = \frac{\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}} + \mathsf{cost}_{\mathcal{T}_c}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}} + \mathsf{cost}_{\mathcal{T}_c}^{\mathsf{OPT}}} > \frac{\mathsf{cost}_{\boldsymbol{\omega}}^{\text{ONCOM}}}{\mathsf{cost}_{\boldsymbol{\omega}}^{\mathsf{OPT}}}, \tag{4.11}$$

where the last inequality follows from $\frac{x_1+x_3}{x_2+x_4} > \frac{x_1}{x_2}$, if $\frac{x_1}{x_2} < \frac{x_3}{x_4}$, $x_1, x_2, x_3, x_4 > 0, x_1 > x_3, x_2 > x_4$. Similarly, Equation (4.11) shows that the cost ratio of $\boldsymbol{\omega}'$ is bigger than that of $\boldsymbol{\omega}$, contradicting that $\boldsymbol{\omega}$ is the worst case input.

Putting together cases (1) and (2), we infer that given the worst instance, the cost ratio in cycle $\mathcal{T}_c$ is equal to the competitive ratio of the ONCOM. □

Given the result in Lemma 4.4.1, in the rest of analysis, we consider the instances which only contains one cycle.

Worst-case Analysis

For notational convenience, we denote the minimum threshold obtained by the threshold functions as $\bar{g}(\boldsymbol{s}(t))$, i.e., $\bar{g}(\boldsymbol{s}(t)) = \min_{i=0,1\ldots,t} g_i(s_i(t))$.

Lemma 4.4.2: Suppose the worst instance for the ONCOM is $\boldsymbol{\omega}$. For an arbitrary positive value $\delta > 0$, we have $\bar{g}(\boldsymbol{s}(t)) - p(t) \leq \delta, \forall t \in \mathcal{T}$.

Proof of Lemma 4.4.2. We prove this by contradiction. Assume there exists a time slot in the worst case instance that $p(t) < \bar{g}(\boldsymbol{s}(t)) - \delta$. Let

$$n = \left\lfloor \frac{\bar{g}(\boldsymbol{s}(t)) - p(t)}{\delta} \right\rfloor .$$

Now, we insert n slots before time slot t and construct a new instance $\boldsymbol{\omega}'$ as follows:

$$\begin{aligned} \omega' &= [(p(\tau), d(\tau))]_{\tau=1:t-1} \\ &\oplus [(\bar{g}(\boldsymbol{s}(t)) - k\delta, 0)]_{k=1,\ldots,n} \\ &\oplus [(p(t), d(t))] \\ &\oplus [(p(\tau), d(\tau))]_{\tau=t+1:T}. \end{aligned}$$

Given $\boldsymbol{\omega}'$, the charging amount of the ONCOM during new inserted slots would be the same as the charging amount in $\boldsymbol{\omega}$ for slot t, however, the cost is higher than the previous cost. On the other hand, the cost of the OPT given $\boldsymbol{\omega}'$ remains intact. Hence, we constructed a new instance whose cost ratio is larger than $\boldsymbol{\omega}$, contradicting the assumption that $\boldsymbol{\omega}$ is the worst case instance. □

The following result is the key in proving the competitive ratio of the OnCoM.

Lemma 4.4.3: Given $g_i(s), i \in \{0,1,2,\ldots,t\}$ in Equation (4.6), we have

$$\frac{(S_i - s)p_{\max} + \int_{z=0}^{s} g_i(x)dx}{g_i(s)S_i} \leq \alpha^\star, \ \forall\ s \in [0, S_i]. \tag{4.12}$$

Proof of Lemma 4.4.3. By substituting Equation (4.6), we first calculate the second term in numerator of Equation (4.12) as follows

$$\begin{aligned}\int_{x=0}^{s} g_i(x)dx &= p_{\max}\left[x - \left(1-\frac{1}{\alpha^\star}\right)\exp\left(\frac{x}{\alpha^\star S_i}\right)\alpha^\star S_i\right]\Bigg|_{x=0}^{s} \\ &= p_{\max}\left[s - \left(1-\frac{1}{\alpha^\star}\right)\exp\left(\frac{s}{\alpha^\star S_i}\right)\alpha^\star S_i\right] \\ &+ p_{\max}\left(1-\frac{1}{\alpha^\star}\right)\alpha^\star S_i.\end{aligned}$$

Thus, we calculate the numerator

$$\begin{aligned}(S_i &- s)p_{\max} + \int_{x=0}^{s} g_i(x)dx \\ &= \alpha^\star S_i\left(p_{\max}\left[1-\left(1-\frac{1}{\alpha^\star}\right)\exp\left(\frac{s}{\alpha^\star S_i}\right)\right]\right) \\ &= \alpha^\star S_i g_i(s).\end{aligned} \tag{4.13}$$

Substituting (4.13) into (4.12) completes the proof. □

We proceed to prove Theorem 4.4.1 by showing that the competitive ratio of the OnCoM is upper bounded by $\alpha^\star$.

Given the result in Lemma 4.4.1, we only consider the instances which only contain one single cycle.

Assume t is the last time slot in Period I. At time slot t, there are $t+1$ virtual storages. An example of the algorithm state is shown in Figure 4.4.

By the construction of function g in Equation (4.6), we have $g_i(s_i(t)) = \min_{t\in[i,t]} p(t)$, and since it is a decreasing function during $[i,t]$ the minimum thresholds of the OnCoM over all virtual storages is $g_i(s_i(t))$. Moreover, the result in Lemma 4.4.2 states that with the worst instance, during $[i,t]$ there is no time slot with price lower than $g_i(s_i(t)) - \delta$, i.e., $p(\tau) \geq g_i(s_i(t)) - \delta, \forall\tau \in [i,t]$.

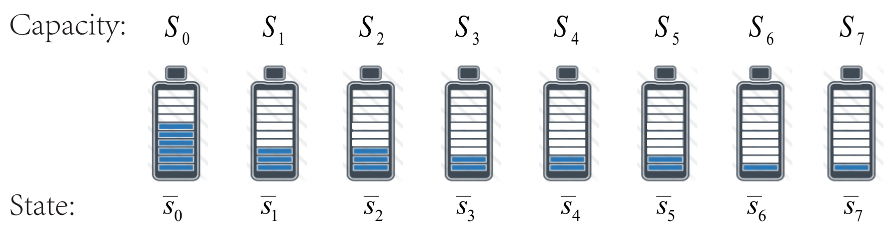

Capacity:
S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
State:
$\bar{s}_0$ $\bar{s}_1$ $\bar{s}_2$ $\bar{s}_3$ $\bar{s}_4$ $\bar{s}_5$ $\bar{s}_6$ $\bar{s}_7$

Fig. 4.4: The algorithm state at time slot $t = 7$.

Using the aforementioned results, we now calculate the costs of both ONCOM and OPT in Periods I and II.

(1) Cost in Period I: By definition, the aggregate demand is equal to the aggregate capacity of virtual storages except the first virtual storage during Period I. Thus, the minimum cost for the OPT is

$$\sum_{i=1}^{t} [g_i(s_i(t)) - \delta] \, S_i.$$

On the other hand, due to the threshold-based strategy, the maximum cost over Period I under the ONCOM is

$$\sum_{i=0}^{t} \int_{x=0}^{s_i(t)} g_i(x) dx.$$

(2) Cost in Period II: In Period II, the state of the storage is 0. Under the worst instance, the adversary sets the market price to the maximum value, and the energy demand is set to the remaining storage level in the OPT. To ensure the worst case competitive, we assume that the storage level in the OPT full, i.e., $s(t) = S$.

In this case, the OPT procures C units of energy in Period I, in addition to fulfilling the demand during Period I, whose cost is at least

$$[g_0(s_0(t)) - \delta] \, S.$$

The additional cost for the ONCOM to procure C units of energy is to buy directly from the market with the maximum price, i.e., $Sp_{\max}$. Using Equation (4.5), we can rewrite this cost as follows:

$$\sum_{i=0}^{t} (S_i - s_i(t)) \, p_{\max}.$$

Putting together the costs for Periods I and II, the cost ratio between the ONCOM and the OPT over one cycle is

$$\frac{\sum_{i=0}^{t}\left[(S_i - \bar{s}_i(t))\, p_{\max} + \int_{x=0}^{s_i(t)} g_i(x)dx\right]}{\sum_{i=0}^{t}\left[g_i(s_i(t)) - \delta\right] S_i},$$

where δ is an arbitrarily small positive value. Based on Lemma 4.4.3, we have

$$\lim_{\delta\to 0}\frac{(S_i - \bar{s}_i(t))\, p_{\max} + \int_{x=0}^{s_i(t)} g_i(x)dx}{\left[g_i(\bar{s}_i(t)) - \delta\right] S_i} \le \alpha^{\star},$$

It implies the the competitive ratio of the ONCOM is $\alpha^{\star}$, matching the lower bound competitive ratio for the ECOM as the optimal competitive ratio k-min search problem.

4.5 Extensions

4.5.1 Extension to Online Bidding Strategy Design

In this section, we extend our design to the case that the price $p(t)$ is unknown for the current slot t. Hereinafter, we drop index t from all notations for simplicity. This extension turns the problem into finding an online bidding strategy, in which the data center along with other customers, submits its bid, including the bidding price $\hat{p}$ and the bidding quantity $\hat{x}$, for the forthcoming slot shortly before the actual operation time. The ISO matches the offers collected from the suppliers with the received bids from the demand-side and using a double auction mechanism, it determines the market clearing price p for the next hour. Then, the bids with bidding price higher than the market clearing price become successful in purchasing submitted bidding quantity from the market.

In this new setting, calculating the value of Equation (4.9) becomes infeasible since p is unknown. Our general approach to tackle the extended scenario is to submit multiple bids. Note that submitting multiple bids from a single entity is allowed in the existing electricity markets, e.g., in PJM market, the each customer can submit at most 10 bids [11].

Let us denote the procurement quantity given clearing price p as $e(p)$. Note that $e(p)$ can be calculated in Line 10 of the ONCOM in Alg. 4.1 and for brevity we dropped index t and just state it as a decreasing function of p as the clearing price.

By defining $\mathcal{B}_k = \langle e_k, p_k\rangle$ as the k-th bid and assuming the maximum

Tab. 4.2: Summary of comparison algorithms

Algorithm	**Description**
OnCoM	The proposed online algorithm for the OnCoM
OnFix [48]	Simple online algorithm with fixed threshold price
OnAdpt [112]	Adaptive threshold algorithm with single storage level
LypOpt [78]	Lyapunov-based algorithm

number of bids is ζ, at each time slot t, we submit ζ bids as follows

$$\begin{aligned}\mathcal{B}_k &= \langle e(p_{\min} + (k-1)\delta) - e(p_{\min} + k\delta), p_{\min} + k\delta\rangle,\\ &\qquad\qquad\qquad\qquad \forall k \in \{1, 2, \ldots, \zeta - 1\},\\ \mathcal{B}_\zeta &= \langle e(\tilde{p}), p_{\max}\rangle,\end{aligned}$$

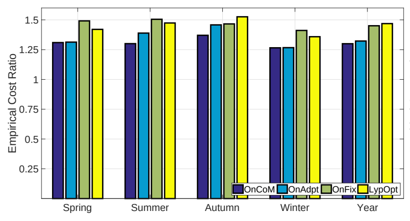
where $\tilde{p} = \bar{g}(\boldsymbol{s})$ is the minimum threshold obtained from all threshold functions as defined in Equation (4.6) and $\delta = (\tilde{p} - p_{\min})/(\zeta - 1)$. $\mathcal{B}_\zeta$ is defined to ensure that the net demand is satisfied by purchasing the electricity from the market. In experiments, we show that by submitting only 6 bids when the price is unknown, the performance becomes almost the same as the case when p is posted in advance.

4.5.2 Extension to Practical Storage Models

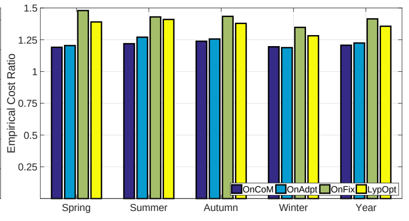
In practice in the storage systems, there are maximum charging and discharging rates. In this section, we extend the solution design in Subsection 4.4.4 to the general storage model with charging and discharging rates. Let ρ_c and ρ_d be the maximum charging and discharging rates of the on-site storage system. Given these limits, the minimum and maximum procurement quantities become $d - \min\{s, \rho_d\}$ and $\rho_c + d$, respectively. Given these values, we modify $e(p)$ as follows

$$e(p) = \begin{cases} \rho_c + d, & \text{if } e(p) \geq \rho_c + d \\ e(p), & \text{if } d - \min\{s, \rho_d\} \leq e(p) \leq \rho_c + d \\ d - \min\{s, \rho_d\}, & \text{if } e(p) \leq d - \min\{s, \rho_d\} \end{cases}$$

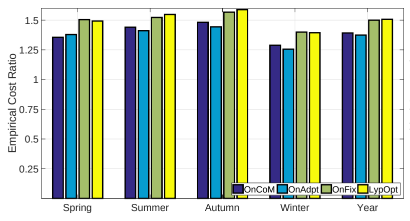
In experiments, we evaluate the performance of this extension.



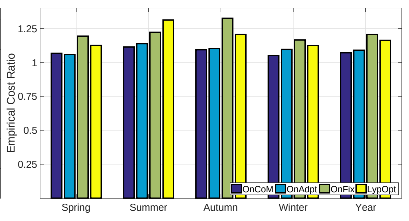
(a) PJM Market



(b) NYISO Market



(c) ECORT Market



(d) Nord Pool Market

Fig. 4.5: Comparison results of different algorithms in different seasons in different electricity markets

Tab. 4.3: Summary of Theoretical and Empirical Competitive Ratios of Different Algorithms in Different Markets

Market	$\hat{\theta}$	Theoretical competitive ratio	Empirical cost ratio			
			ONCOM	ONADPT	ONFIX	LYPOPT
PJM	21.93	3.63	1.29	1.32	1.45	1.47
NYISO	3.39	1.61	1.20	1.23	1.42	1.36
ERCOT	8.32	2.35	1.39	1.38	1.50	1.51
Nord Pool	1.95	1.29	1.07	1.09	1.21	1.16
Average	**8.90**	**2.21**	**1.23**	**1.26**	**1.39**	**1.38**

4.6 *Experimental Results*

In this section, we evaluate the performance of the proposed ONCOM algorithm using real-world traces for data center energy demand and electricity market prices. Our objective is three-fold: (i) to compare the performance against the offline optimum and existing alternatives [112, 48, 78] (as listed in Table 4.2); (ii) to investigate the impact of the system model parameters on the performance of the proposed algorithm; and (iii) investigate the performance of extensions in Sec. 4.5.

4.6.1 Experimental Settings

- **Data Center Energy Demand** The real-world workload data is based on the access traces published by Wikipedia [14]. One line of those traces corresponds to one web access, including the time-stamp of the request and the requested URL, among others. By counting the number of accesses every 1 hour, the energy demand in kWh is approximately calculated based on the model proposed in [78]. To create uncertainty, we inject the scaled output of a wind farm in California [7] such that on average 50% of demand is fulfilled by renewable supply. The remaining net demand must be satisfied by either purchasing from the market or discharging the storage.

- **Electricity Market Prices** We use electricity prices from PJM [11], NYISO [8], ERCOT [62], and Nord Pool [9], whose price fluctuation ratios are shown in Table 4.3. Since the prices exhibit severe seasonal patterns, we report the results in a seasonal basis. Toward this, we use the prices for October 2016 (autumn), January 2017 (winter), April 2017 (spring), and July 2017 (summer). The summer prices from PJM market are used for the rest of the experiments.

- **The Performance Metric and Comparison Algorithms** We report the “empirical cost ratio” as the performance metric, which is obtained by dividing the cost of each algorithm by the cost of offline optimal solution, thereby the lower the value, the better the performance. As listed in Table 4.2, we compare the result of the proposed OnCoM algorithm with existing algorithms as follows: (i) the OnFix [48] as a simple strategy that follows a fixed threshold of $\sqrt{p_{\max} \times p_{\min}}$ as the purchasing policy; (ii) the OnAdpt [112] as the simplified version of the OnCoM in which there is an adaptive policy with a *single* threshold function for the original storage similar to Equation (4.6); and finally (iii) the LypOpt [78] as a Lyapunov-based policy. Note that we ignore the QoS model in [78] and only consider the storage management similar to ours.

- **Parameter Settings** Unless otherwise mentioned, we set the length of each slot to 1 hour and $T = 60$, i.e., 5 days. The capacity is set to $S = 5 \times \max_{t\in\mathcal{T}} d(t)$. Finally, each data point in figures corresponds to the average results of 100 runs each of which with different random demands and market prices, picked from data traces.

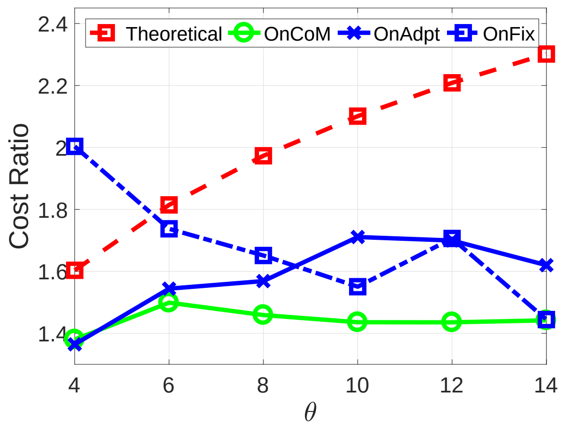
4.6.2 Experimental Results

- **Comparison Results across Different Seasons and Markets** The results are shown in Figure 4.5 and Table 4.3. The notable observations are: (i) the cost of the ONCOM is 1.23 times of the offline optimum, on average in different markets; (ii) the ONCOM outperforms the alternative algorithms ONFIX [48] and LYPOPT [78] significantly and its performance is slightly better than ONADPT [112] as the other adaptive threshold based policy; and finally (iii) the empirical cost ratio of the ONCOM is much better than the obtained theoretical competitive ratio.

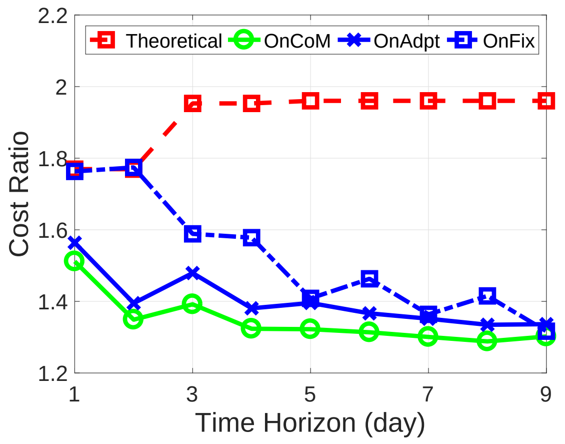
- **Impact of System Model Parameters** In Figures 4.6.2, 4.6.2, and 4.6.2, we investigate the impacts of price fluctuation ratio $\hat{\theta}$, the length of time horizon T, and the capacity of storage on the cost ratio of different algorithms. The result in Figure 4.6.2 demonstrates that while the theoretical competitive ratio increases as $\hat{\theta}$ increases (see Equation (4.7)), the empirical cost ratio of the ONCOM is almost the same with slight increase. More importantly, comparing the result of the ONCOM and ONADPT for large values of $\hat{\theta}$ shows that the performance of our algorithm when the price is highly fluctuating is considerably better than the ONADPT. The result in Figure 4.6.2 demonstrates that there is a decrease in the cost ratio for all the algorithms as the length of time horizon increases. Finally, the results in Figure 4.6.2 depict an increase in the cost ratio of all the algorithms as the capacity of storage increases. This is reasonable, since with larger capacity the offline optimum has more flexibility to utilize storage to shape the energy procurement from the market.

- **The Performance of the Extended Algorithms** In Figure 4.6.2, we report the performance of the proposed extension of the ONCOM (ONCOM-MultipleBids) by submitting multiple bids when the market price is unknown for the forthcoming slot. Toward this, we change the maximum number of submitted bids, and report the empirical cost ratios. The results show that as the number of submitted bids increases the empirical cost ratio decreases. By submitting more than 6 bids, the cost ratio is very close to the ONCOM where the price is known (1.22 vs. 1.19).

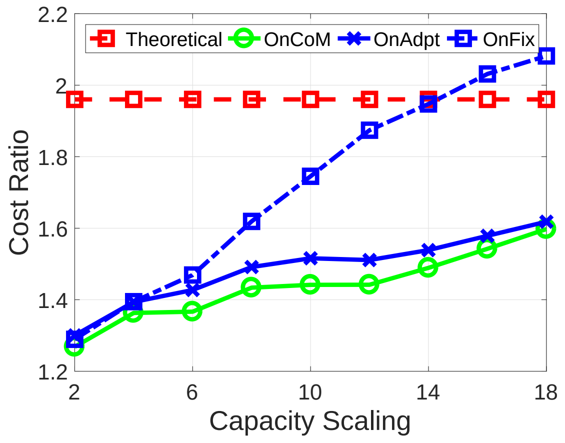
  In Figure 4.6.2, we set $\rho = \rho_c = \rho_d$ and change their restrictiveness by varying fraction ρ/S. The result shows that initially when the charging rate is very restricted, i.e., $\rho/S < 0.2$, the empirical cost ratio of the extended ONCOM (ONCOM-Rates) is smaller than that of original ONCOM. This is reasonable since



(a)

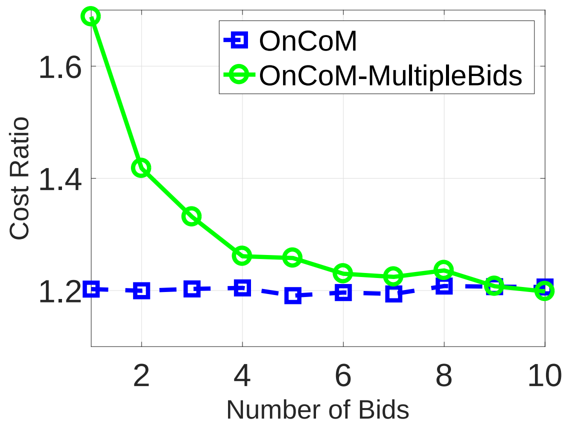


(b)

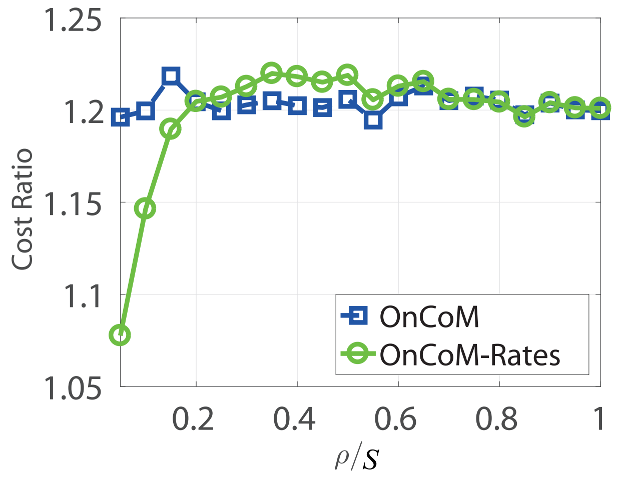


(c)

Fig. 4.6: The impacts of system model parameters on empirical cost ratio of different algorithms



(a)



(b)

Fig. 4.7: The results of extensions of the OnCoM

with more restriction in charging rate, the overall benefits of the storage is limited, hence both offline optimum and the OnCoM-Rates cannot reduce the cost substantially, thereby achieving almost the same cost. On the other hand, with $\rho/S > 0.2$, the cost ratio of OnCoM-Rates becomes very close to that of the OnCoM, which means that the extended algorithm works properly.

□ **End of chapter.**

5. ONLINE LEARNING

In this chapter, we will investigate a class of influential online learning problems, whose basic version stems from the classical Expert problem and Online Convex Optimization problem. Motivated by many recent applications, in this thesis, we investigate a more general framework, called the Online Non-Convex Leaning problem or the Lipschitz Expert problem in some literature. This framework can be applied to more general scenarios where the training set is in a metric space, and the cost function is not necessarily convex. By firstly introducing an optimal algorithm in the literature, this thesis provides a more complete picture for the online learning community. As a gentle start, we begin with introducing the Expert problem.

5.1 The Expert Problem

Consider a common scenario where an online player has to make a choice among available actions (for example, buy or not in a stock market). In the online learning context, we can assume there are a number of (e.g, N) "experts" who can offer advice to the player on making decisions. At each round, the online player will choose one expert and follow his advice. After committing to one choice, the online player will be aware of the experts who have offered the correct advice this round. The goal of the online player is to following the "correct" expert, i.e., making mistakes as few as possible.

For the expert prediction problem, [108] introduces a classic algorithm called the Weighted Majority algorithm which maintains exponential weights to evaluate the performance of the expert. In a more general scenario, we can consider the case where the expert's advice is evaluated by a quantified real value $c_t(i)$ instead of counting the number of mistakes. In this case, a closely related algorithm, called Hedge, was introduced in [72]. For each expert, there is an exponential weight, denoted by $\xi_t(i),\ i = 1, 2, \ldots, N$, and the Hedge algorithm choose the i-th expert with probability $q_t(i) = \xi_t(i)/\sum_{j=1}^{N} \xi_t(j)$. The details of Hedge are summarized by the following pseudo-code.

Algorithm 5.1 Hedge

```
1  Initialize: $\forall i \in \mathcal{N},\ \xi_1(i) = 1$
```

```
2 for $t = 1$ to $T$
3     Pick the $i$-th expert with probability $q_t(i) = \xi_t(i)/\sum_{j=1}^{N}\xi_t(j)$
4     Incur loss $c_t(I_t)$
5     Update weights $\xi_{t+1}(i) = \xi_t(i)\exp(-\eta c_t(i))$
6 end for
```

By properly designing the weighting strategy, the Hedge algorithm can guarantee the average cost to approach that of the best expert asymptotically. The performance of Hedge algorithm can be summarized in the following theorem (for the details of the proof, one can refer to the survey paper by Hazan [80]).

Theorem 5.1.1: Let $\boldsymbol{c}_t^2$ denote the N-dimensional vector of square losses, let $\eta > 0$, and assume all losses to be non-negative. The Hedge algorithm satisfies for any expert $i \in \mathcal{N}$:

$$\sum_{t=1}^{T} \boldsymbol{q}_t^\top \boldsymbol{c}_t \le \sum_{t=1}^{T} c_t(i) + \eta \sum_{t=1}^{T} \boldsymbol{q}_t^\top \boldsymbol{c}_t^2 + \frac{\log N}{\eta}.$$

Assume $c_t \le c_{\max}$. By setting $\eta = \sqrt{\frac{\log N}{T c_{\max}^2}}$, we have that

$$\sum_{t=1}^{T} \boldsymbol{q}_t^\top \boldsymbol{c}_t - \sum_{t=1}^{T} c_t(i) \le 2c_{\max}\sqrt{T \log N}.$$

In other words, the regret of the Hedge algorithm is $O(\sqrt{T \log N})$.

5.2 Online Convex Optimization

In addition to the Expert learning problem, there has been also extensive research focusing on the Online Convex Optimization problem since the seminal work by Zinkevich [149]. The OCO problem is modeled as a repeated game composed of T iterations. At iteration t, the player chooses a point $\boldsymbol{x}_t$ from a bounded convex decision set $\mathcal{K} \subset \mathbb{R}^n$; after the choice is committed, a bounded convex cost function $f_t : \mathcal{K} \mapsto \mathbb{R}^+$ is revealed to the player. The goal of the player is to minimize the *regret*, which is defined as the difference between the online cumulative cost and the cumulative cost using an optimal offline choice in hindsight. This model can be applied to many real-world problems, such as online routing [30, 136], spam email filtering [79, 130], online metric learning [86], ad selection and content ranking in search engines [138, 54, 115], etc.

A promising result related to the OCO model is that the regret of the state-of-the-art efficient algorithms [80] is $O(\sqrt{T})$, touching the well-known lower bound of the regret for the OCO problem [80]. Researchers have proposed a large number of online algorithms whose regret attains this lower bound, including the Online Gradient Decent method [149], the Stochastic Gradient Decent method [128, 126, 133, 82], the Online Newton Step, and many regularization-related methods [95, 77] (see the recent survey paper [80], and the references therein).

5.3 A More General Framework: Online Non-Convex Learning/Lipschitz Expert

Despite the success of the Expert problem and the Online Convex Optimization problem, there are still many limitations especially when the learning set is continuous but there is no convexity assumptions for the decision set or cost functions. For example, in the portfolio selection problem [55, 88], the decision maker (e.g., the trader) chooses a distribution of her wealth allocation over n assets $\boldsymbol{x}_t$, at each round. By the end of each round, the adversary chooses the market returns of the assets with positive values. In some specific settings [137, 104, 27], the online portfolio selection problem is a non-convex one due to the non-convex diversification constraints and non-convex transaction costs, and thus the traditional OCO framework fails in modeling such case. In addition, there is extensive machine learning research focusing on non-convex loss functions in large margin classifiers [119, 144, 64]. In [119, 64], non-convex online Support Vector Machine (SVM) models has been studied which adopts a non-convex loss function, called Ramp Loss, to suppress the influence of outliers. In [144], a special non-convex penalty, called the smoothly clipped absolute deviation penalty, is imposed on the hinge loss function in the SVM. Such a new SVM is applied to identify important genes for cancer classification [144].

This thesis tackles the ONCL problem, with non-convex L-Lipschitz cost functions and non-convex continuous decision set. The online non-convex learning problem demonstrates a structured repeated game, whereat each iteration t, the player chooses a decision $\boldsymbol{x}_t \in \mathcal{K}$, where $\mathcal{K} \subseteq \mathbb{R}^n$ is a bounded decision set whose diameter is D, i.e., $\sup_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{K}} ||\boldsymbol{x}-\boldsymbol{y}||_2 = D$. After the player commits to a decision point at iteration t, the adversary chooses a cost function $f_t(\boldsymbol{x}) \in \mathcal{F}$, where $\mathcal{F} : \mathcal{K} \mapsto \mathbb{R}^+$ is a bounded set of cost functions which are assumed to be non-negative and L-Lipschitz (as defined in Equation (5.1)).

Definition 5.3.1: A function $f : \mathcal{K} \to \mathbb{R}$ (where $\mathcal{K} \subset \mathbb{R}^n$) is L-Lipschitz if

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x}-\boldsymbol{y}\|_2, \quad \forall \boldsymbol{x},\boldsymbol{y} \in \mathcal{K}. \tag{5.1}$$

Tab. 5.1: Summary of notations related to the ONCL problem

Notation	**Description**
t	Index of iteration
T	The number of iterations, $T \geq 1$
$\mathcal{T}$	Set $\mathcal{T} = \{1, 2, \ldots, T\}$
n	The dimension of the decision space
$\boldsymbol{x}_t$	Chosen decision point at t
$\mathcal{K}$	The decision set available for the online player
D	Diameter of the decision set $\mathcal{K}$
$f_t(\boldsymbol{x})$	$f_t(\boldsymbol{x}) : \mathcal{K} \mapsto \mathbb{R}^+$. Cost function at iteration t, known for $t-1$, unknown for $\tau \geq t$
L	Lipschitz constant for the cost functions
$\mathcal{F}$	The set of bounded cost functions which are available for adversary
$\boldsymbol{H}_t$	$\boldsymbol{H}_t = (f_1, f_2, \ldots, f_{t-1})$. The historical cost functions available for the online algorithm at iteration t

At each iteration, the player needs to make online decisions without knowing the current and future cost functions. Fed with the historical cost functions $\boldsymbol{H}_t = (f_1, f_2, \ldots, f_{t-1})$, the decision of an online algorithm $\mathfrak{A}$ at iteration t is denoted as $\boldsymbol{x}_t = \mathfrak{A}(\boldsymbol{H}_t)$.

Our goal is to design an online algorithm for the ONCL problem and try to minimize the regret. Ideally, it is desired to have a sublinear regret with respect to T, i.e., $\mathsf{regret}_T(\mathfrak{A}) = o(T)$. The sublinear regret implies that time-average performance of the online algorithm is as good as the best fixed strategy as time goes to infinity.

5.4 Related Results

The ONCL problem is not a new problem and there are plenty prior works on it. Among them, [64] and [73] propose respective heuristic online training algorithms, but neither of them are rigorously shown to satisfy any regret bound. In [81], Hazan and Kale tackle the ONCL problem with submodular cost functions, and propose an online algorithm that attains the regret of $O(\sqrt{T})$. In [145], an online bandit learning problem with non-convex losses is investigated. The cost function is again a special non-convex function, defined as the composition of a non-increasing scalar function with a linear function of small variation. An online algorithm is developed with $\tilde{O}(\mathsf{poly}(d)T^{2/3})$ regret, where $\mathsf{poly}(d)$ stands for a polynomial with respect to the dimension of the decision set d. The most related works to ours are [103] and [116], where by applying the exponential

Tab. 5.2: Summary of key notations related to the ORW algorithm

Notation	Description
$\mathfrak{R}$	The notation used for the ORW algorithm in the equations (Algorithm 5.2)
$\mathcal{D}$	Cover cube of the decision set $\mathcal{K}$
$\boldsymbol{i}_\ell$	Index for a sub-cube/subset in ℓ-th layer, $\boldsymbol{i}_\ell = (i_{\ell,1}, i_{\ell,2}, \cdots, i_{\ell,n})$
$\mathcal{D}_\ell(\boldsymbol{i}_\ell)$	Layer-ℓ sub-cube indexed by $\boldsymbol{i}_\ell$
$\mathcal{K}_\ell(\boldsymbol{i}_\ell)$	Layer-l subset indexed by $\boldsymbol{i}_\ell$
$\mathcal{I}_\ell$	Index set for those nonempty layer ℓ subsets
$\boldsymbol{v}_s(\boldsymbol{i}_\ell)$	Index of layer s subset that contains $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$
$\mathcal{H}_s(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$	Index set of nonempty layer s subsets within the subset $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$
m	The number of layers where the subsets are sampled
$\boldsymbol{q}_m(\boldsymbol{i}_m)$	The sampled point for the subset $\mathcal{K}_m(\boldsymbol{i}_m)$
$c_{m,t}(\boldsymbol{i}_m)$	The cost on the sample point of the subset $\mathcal{K}_m(\boldsymbol{i}_m)$ at iteration t, i.e., $c_{m,t}(\boldsymbol{i}_m) \overset{\text{def}}{=} f_t(\boldsymbol{q}_m(\boldsymbol{i}_m))$
L_m	The maximum cost difference of two points in the same layer m subset
$\boldsymbol{I}_{l,t}$	The index of the subset chosen in layer l at iteration t
$\bar{c}_{\ell,t}(\boldsymbol{i}_\ell)$	The *normalized expected cost* conditioning on that subset $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ is chosen in layer ℓ at iteration t. For short, we also call it the expected cost of choosing $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ in layer l at iteration t
$\bar{C}_{\ell,t}(\boldsymbol{i}_\ell)$	The cumulative sum for $\bar{c}_{\ell,\tau}(\boldsymbol{i}_\ell)$ up to iteration t, i.e., $\bar{C}_{\ell,t}(\boldsymbol{i}_\ell) = \sum_{\tau=1}^{t} \bar{c}_{\ell,\tau}(\boldsymbol{i}_\ell)$. For short, we also call it the cumulative expected cost of choosing $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ in layer l up to iteration t

weighting method [29] the regret of $O(\sqrt{T \log T})$ is attained.

In addition, the ONCL problem has been broadly investigated under a similar problem, called the Lipschitz Expert problem [96, 98], which generalizes the traditional Expert problem to metric spaces. For such a problem, a regret of $O(\sqrt{T \log T})$ can be achieved [98]. However, to the best of our knowledge, for the general ONCL problem, no online algorithm can achieve the regret of $O(\sqrt{T})$ as the well-known lower bound for the OCO.

5.5 Online Recursive Weighting Algorithm

The Online Recursive Weighting (ORW) algorithm is based on the idea of dividing the entire decision set into several *sampling subsets* so that each subset contains highly correlated elements. An essential tuning parameter for the algorithm is the granularity of the sampling subsets, which determines the number of final sample points (details in Sections. 5.5.1 and 5.5.2).

To define the granularity of the sampling subsets, we construct a *layered grid structure*. The topmost layer consists of a single grid and hence a single subset including the entire decision set $\mathcal{K}$. The sampling subsets are the subsets in the bottommost layer. Hence, assuming there are m layers, there are 2^{nm} sampling subsets in total. In Section 5.5.1, we explain the details of constructing the layered grid structure.

In the next step (Section 5.5.2), a single element is randomly selected from each sampling subset called *sample point*, and by grouping all these points, we construct the set of sample points. At each iteration, the algorithm selects a *decision point* randomly from the set of sample points according to a probability distribution function.

Deriving the probability distribution function to select the decision point is the core technical contribution of the ORW algorithm and is proposed in Section 5.5.3.

5.5.1 A Layered Grid Structure

Recall that the diameter of bounded decision set $\mathcal{K}$ is D, hence, it is possible find a bounded cube of length D, denoted by $\mathcal{D}$, that can cover $\mathcal{K}$ entirely, i.e., $\mathcal{K} \subset \mathcal{D}$. In layer $\ell \in \mathbb{N}^+$, we partition $\mathcal{D}$ into smaller identical sub-cubes with edge length of size $D/2^\ell$. The decision set is n-dimensional, hence the total number of sub-cubes is equal to $2^{n\ell}$. For simplicity, each sub-cube is indexed by a distinct n-dimensional vector $\boldsymbol{i}_\ell = (i_{\ell,1}, i_{\ell,2}, \dots, i_{\ell,n})$, where $1 \leq i_{\ell,j} \leq 2^\ell, j = 1, 2, \dots, n$, and the sub-cube indexed by $\boldsymbol{i}_\ell$ is denoted by $\mathcal{D}_\ell(\boldsymbol{i}_\ell)$. One can refer to Figure 5.1 for an illustrative example with $n = 2$ and $\ell = 1$.

The notation $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ is used to represent the intersection of the decision set $\mathcal{K}$ and corresponding sub-cube $\mathcal{D}_\ell(\boldsymbol{i}_\ell)$, i.e.,

$$\mathcal{K}_\ell(\boldsymbol{i}_\ell) \overset{\text{def}}{=} \mathcal{D}_\ell(\boldsymbol{i}_\ell) \cap \mathcal{K}.$$

By convention, we regard the decision set $\mathcal{K}$ as the only layer 0 subset, denoted by $\mathcal{K}_0(\mathbf{1}) = \mathcal{K}$. By the above partitioning structure, in total $2^{n\ell}$ subsets at layer ℓ are constructed, whose union is the decision set $\mathcal{K}$. The subset can be empty and we denote the index set for nonempty layer l subsets by $\mathcal{I}_\ell$, i.e.,

$$\mathcal{I}_\ell \overset{\text{def}}{=} \{\boldsymbol{i}_\ell : \mathcal{K}_\ell(\boldsymbol{i}_\ell) \neq \emptyset\}.$$

Any subset indexed by $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$, i.e., $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, consists of a group of

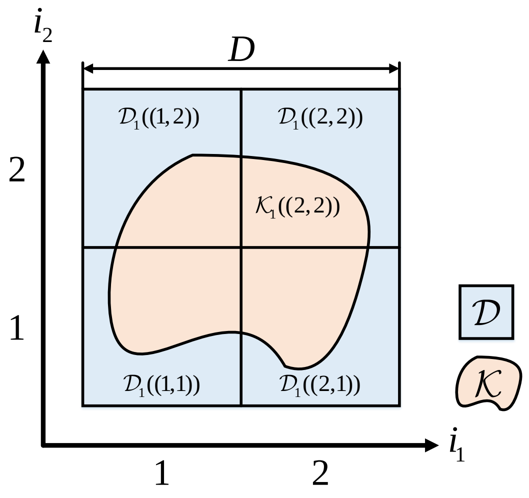


Fig. 5.1: Layer-1 set partitioning for a general decision set $\mathcal{K}$.

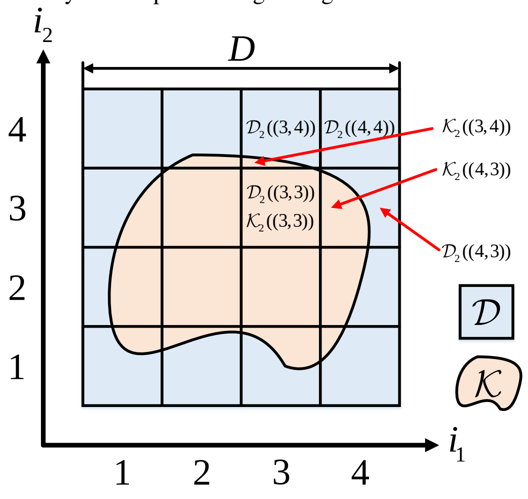


Fig. 5.2: The index set for the overlapped sub-cubes when $n = 2$ and $\ell = 2$.

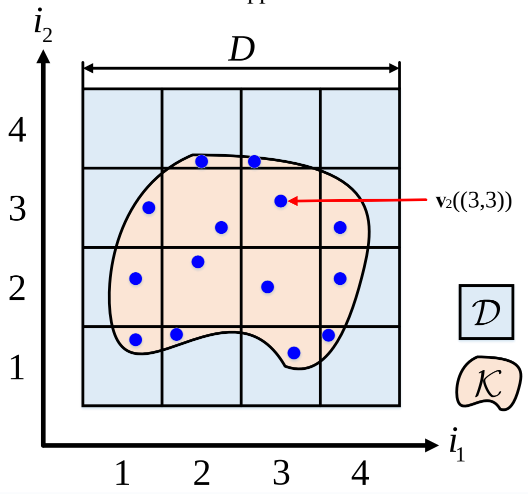


Fig. 5.3: Sampling for the case when $n = 2$ and $m = 2$.

neighboring lower-layer subsets. Specifically, for any $s > \ell$, we have

$$\mathcal{K}_\ell(\boldsymbol{i}_\ell) = \bigcup_{\substack{\boldsymbol{i}_s \,:\, i_{s,j} \geq 1 + (i_{\ell,j} - 1)2^{s-\ell}, \\ i_{s,j} \leq i_{\ell,j} 2^{s-\ell}, \\ j = 1,2,\ldots,n.}} \mathcal{K}_s(\boldsymbol{i}_s)$$

To ease the presentation, we define the following notation:

▷ We use $\mathcal{H}_s\left(\mathcal{K}_\ell(\boldsymbol{i}_\ell)\right),\ s \geq \ell$, to denote the index set of nonempty layer s subsets within $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, i.e.,

$$\mathcal{H}_s\left(\mathcal{K}_\ell(\boldsymbol{i}_\ell)\right) \overset{\text{def}}{=} \left\{\boldsymbol{i}_s \in \mathcal{I}_s : 1 + (i_{\ell,j} - 1)2^{s-\ell} \leq i_{s,j} \leq i_{\ell,j} 2^{s-\ell}\right\}. \quad (5.2)$$

Indeed, we have $|\mathcal{H}_s\left(\mathcal{K}_\ell(\boldsymbol{i}_\ell)\right)| \leq 2^{(s-\ell)n}$.

▷ Assume $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$ and $s \leq \ell$. We use $\boldsymbol{v}_s(\boldsymbol{i}_\ell)$ to denote an index for some layer s subset, and the elements of $\boldsymbol{v}_s(\boldsymbol{i}_\ell)$ satisfy

$$v_{s,j} = \left\lceil \frac{i_{\ell,j}}{2^{\ell-s}} \right\rceil,\ j = 1,2,\ldots,n.$$

Due to the above definition of $\boldsymbol{v}_s(\boldsymbol{i}_\ell),\ \boldsymbol{i}_\ell \in \mathcal{I}_\ell$, together with the set partitioning method we adopt, it follows that $\mathcal{K}_\ell(\boldsymbol{i}_\ell) \subset \mathcal{K}_s(\boldsymbol{v}_s(\boldsymbol{i}_\ell))$.

Example 1. Figure 5.1 illustrates a simple example of set partitioning when $n = 2$ and $l = 1$. At layer 1, the cover cube $\mathcal{D}$ is partitioned into four smaller sub-cubes, each of whom intersects the decision set $\mathcal{K}$, forming four nonempty subsets, respectively.

$$\mathcal{I}_1 = \{(1,1),(1,2),(2,1),(2,2)\}.$$

In Figure 5.2, the subsets in layer 1 are further divided and 14 fixed nonempty layer 2 subsets are constructed. Particularly, the subset $\mathcal{K}_1((2,2))$ contains three nonempty layer 2 subsets, i.e., $\mathcal{K}_2((3,3))$, $\mathcal{K}_2((3,4))$, and $\mathcal{K}_2((4,3))$. Thus, we have

$$\mathcal{H}_2\left(\mathcal{K}_1((2,2))\right) = \{(3,3),(3,4),(4,3)\}.$$

According to the definition of $\boldsymbol{v}_s(\boldsymbol{i}_l)$, we have

$$\boldsymbol{v}_1((3,3)) = \boldsymbol{v}_1((3,4)) = \boldsymbol{v}_1((4,3)) = (2,2),$$

and $\mathcal{K}_2((3,3)) \subset \mathcal{K}_1((2,2)), \mathcal{K}_2((3,4)) \subset \mathcal{K}_1((2,2))$, and $\mathcal{K}_2((4,3)) \subset \mathcal{K}_1((2,2))$.

5.5.2 Sampling

Assuming layer 0 contains the original decision set $\mathcal{K}$, we fix the number of layers to be $m+1$, where layer m contains at most 2^{nm} nonempty subsets. In the sampling procedure, a *sample point*, $\boldsymbol{q}_m(\boldsymbol{i}_m)$, is selected randomly as a representative, from each nonempty subset in layer m, $\mathcal{K}_m(\boldsymbol{i}_m)$. Figure 5.3 depicts a simple example for $n=2$ and $m=2$, where the sample points are colored in blue. Due to the bijective correspondence between the sample points and the subsets in layer m, we interchangeably use choosing a sample point from a subset and choosing the corresponding layer m subset.

In Section 5.5.3, we define a recursive, probabilistic algorithm to select subsets as we go down the layered structure. Correspondingly, this defines a probabilistic algorithm for choosing a *decision point* from the sample points as the final decision of the ORW algorithm.

Once the decision point is selected at each iteration, the cost of the decision, $f_t(\boldsymbol{q}_m(\boldsymbol{i}_m))$, along with the function f_t is revealed.

For convenience, we denote the cost of the sample point from the layer m subset, $\mathcal{K}_m(\boldsymbol{i}_m)$, $\boldsymbol{i}_m \in \mathcal{I}_m$, at iteration t by $c_{m,t}(\boldsymbol{i}_m)$, i.e.,

$$c_{m,t}(\boldsymbol{i}_m) \overset{\text{def}}{=} f_t(\boldsymbol{q}_m(\boldsymbol{i}_m)).$$

Note that by this sample point construction step, the proposed algorithm reduces the original problem in principle to an Expert problem with $|\mathcal{I}_m|$ experts. As compared to the classical setting of the Expert problem in [46], the difference is that the cost function in our problem is a non-convex L-Lipschitz continuous one. Finally, the following lemma characterizes the cost difference between two sample points.

Lemma 5.5.1: For $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathcal{I}_m$, we have

$$|c_{m,t}(\boldsymbol{\mu}) - c_{m,t}(\boldsymbol{\nu})| \leq 2L_m ||\boldsymbol{\mu} - \boldsymbol{\nu}||_1,$$

where $L_m = \sqrt{n}DL/(2^m)$ and $||\boldsymbol{\mu} - \boldsymbol{\nu}||_1 = \sum_{j=1}^n |\mu_j - \nu_j|$.

Proof of Lemma 5.5.1. The maximum distance between two points in any subsets at the same layer m is $\sqrt{n}D/(2^m)$. Considering the Lipschitz continuous condition, the maximum cost difference within the same layer m subset is $L_m = \sqrt{n}DL/(2^m)$. We define that, any two subsets of the same layer, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, are said to be neighboring subsets if their coordinates satisfy $\mu_j - \nu_j \leq 1$ for $j = 1, 2, \ldots, n$. Then, the maximum cost difference of two points in the union of any two neighboring layer m subsets is $2\sqrt{n}DL/(2^m)$. Thus, the maximum distance for any two points in the union of two layer m subsets, $\boldsymbol{\mu}$ and

$\boldsymbol{\nu}$, is $2\sqrt{n}DL/(2^m)||\boldsymbol{\mu}-\boldsymbol{\nu}||_1$. This yields the result in Lemma 5.5.1. □

5.5.3 Recursive Choosing Policy

In this subsection, we propose a novel recursive decision structure to determine the index point for the bottom-layer subset over the index set $\mathcal{I}_m$. By implementing the ORW algorithm, the regret is reduced to match the known lower bound, $O(\sqrt{T})$ [80] for the OCO problem.

At iteration t, $t = 1, 2, \ldots, T$, the final choice of the ORW algorithm is obtained by recursively choosing a nonempty subset from the topmost layer ($\ell = 0$) to the bottommost layer ($\ell = m$). In this way, the action of the ORW algorithm at each iteration t, consists of a sequence of indexes of subsets, i.e., $(\boldsymbol{I}_{0,t}, \boldsymbol{I}_{1,t}, \boldsymbol{I}_{2,t}, \ldots, \boldsymbol{I}_{m,t})$. By default, $\boldsymbol{I}_{0,t}$ is set to be $\mathbf{1}$, referring to the only decision set $\mathcal{K}$. Additionally, $\boldsymbol{I}_{\ell,t}$ should satisfy that $\boldsymbol{I}_{\ell,t} \in \mathcal{H}_\ell(\mathcal{K}_{\ell-1}(\boldsymbol{I}_{\ell-1,t}))$, i.e., $\mathcal{K}_\ell(\boldsymbol{I}_{\ell,t}) \subset \mathcal{K}_{\ell-1}(\boldsymbol{I}_{\ell-1,t})$ for $\ell = 1, 2, \ldots, m$.

At each layer, a stochastic policy is adopted to select a lower-layer subset. Suppose that, at iteration t and at layer l, $0 \le \ell < m$, the ORW algorithm has chosen subset $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$, i.e., $\boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell$. Then, the algorithm proceeds to choose a subset at the next layer (layer $(\ell+1)$) in $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ randomly according to a conditional probability.

In the ORW algorithm, and for $\boldsymbol{i}_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$, the conditional probability, $\Pr^t_{\ell,\ell+1}(\boldsymbol{i}_\ell, \boldsymbol{i}_{\ell+1}) \overset{\text{def}}{=} \Pr\left[\boldsymbol{I}_{\ell+1,t} = \boldsymbol{i}_{\ell+1} | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell\right]$, is based on the cumulative expected cost of the *previous* iterations.

For iteration τ, $0 \le \tau \le t-1$, the *expected normalized cost* of subset $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, denoted by $\bar{c}_{\ell+1,\tau}(\boldsymbol{i}_{\ell+1})$, is defined as follows

$$\begin{aligned}\bar{c}_{\ell,\tau}(\boldsymbol{i}_\ell) &\overset{\text{def}}{=} \mathbb{E}\left[\frac{c_{m,\tau}(\boldsymbol{I}_{m,\tau}) - \min\limits_{\boldsymbol{i}_m \in \mathcal{Q}} c_{m,\tau}(\boldsymbol{i}_m)}{2^{m-\ell+1}L_m} | \boldsymbol{I}_{\ell,\tau} = \boldsymbol{i}_\ell\right] \\ &= \sum_{\boldsymbol{j}_m \in \mathcal{H}_m(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \frac{c_{m,\tau}(\boldsymbol{j}_m) - \min\limits_{\boldsymbol{i}_m \in \mathcal{Q}} c_{m,\tau}(\boldsymbol{i}_m)}{2^{m-\ell+1}L_m} \cdot \Pr\left[\boldsymbol{I}_{m,\tau} = \boldsymbol{j}_m | \boldsymbol{I}_{\ell,\tau} = \boldsymbol{i}_\ell\right],\end{aligned} \tag{5.3}$$

where $\mathcal{Q} = \mathcal{H}_m(\mathcal{K}_{\ell-1}(\boldsymbol{v}_{\ell-1}(\boldsymbol{i}_\ell)))$. Note that $\bar{c}_{\ell+1,\tau}(\boldsymbol{i}_{\ell+1})$ can be computed at the end of iteration τ, once the cost function is revealed. In Equation (5.3), the conditional selection probability for subset $\boldsymbol{j}_m \in \mathcal{I}_m$, i.e., $\Pr\left[\boldsymbol{I}_{m,\tau} = \boldsymbol{j}_m | \boldsymbol{I}_{\ell,\tau} = \boldsymbol{i}_\ell\right]$ can be obtained by multiplying the selection probabilities of the corresponding subsets at each layer, i.e.,

$$\begin{aligned}&\Pr\left[\boldsymbol{I}_{m,\tau}=\boldsymbol{j}_m|\boldsymbol{I}_{\ell,\tau}=\boldsymbol{i}_\ell\right]\\&=\prod_{k=\ell}^{m-1}\Pr\left[\boldsymbol{I}_{k+1,\tau}=\boldsymbol{v}_{k+1}(\mathcal{K}_m(\boldsymbol{j}_m))|\boldsymbol{I}_{k,\tau}=\boldsymbol{v}_k(\mathcal{K}_m(\boldsymbol{j}_m))\right].\end{aligned}\tag{5.4}$$

For a subset $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, the *cumulative expected cost*, denoted by $\bar{C}_{\ell,t}(\boldsymbol{i}_\ell)$, is defined as the sum of the *expected normalized cost* from iteration 1 to iteration t, i.e.,

$$\bar{C}_{\ell,t}(\boldsymbol{i}_\ell)\overset{\text{def}}{=}\sum_{\tau=1}^{t}\bar{c}_{\ell,\tau}(\boldsymbol{i}_\ell).\tag{5.5}$$

By convention, let the initial value of the cumulative expected cost be zero, i.e., $\bar{C}_{\ell,0}(\boldsymbol{i}_\ell)=0$ for all $\boldsymbol{i}_\ell\in\mathcal{I}_\ell$ and for any layer $\ell=1,2,\cdots,m$.

At iteration t, the selection probability, $\Pr\left[\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell\right]$, is proportional to the exponent of the expected cumulative cost of choosing subset $\boldsymbol{i}_{\ell+1}$ at layer $(\ell+1)$ up to iteration $t-1$, i.e.,

$$\Pr\left[\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell\right]=\frac{\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))}\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)},\tag{5.6}$$

where η_t is a positive and decreasing parameter which implies that the change of $\bar{C}(\cdot)$ has decreasing influence on the choosing probability and thus the decisions of the online algorithm get more steady as time goes on. Note that the denominator in Equation (5.6) is a normalizer such that right hand side of Equation (5.6) is a probability mass function.

Note that the ORW algorithm updates the cumulative expected cost at the end of each iteration. Once the conditional probabilities are defined and the cost function at iteration t is revealed, we then calculate the expected normalized cost and further update the cumulative expected cost for each subset, which is

$$\bar{C}_{\ell,t}(\boldsymbol{i}_\ell)=\bar{C}_{\ell,t-1}(\boldsymbol{i}_\ell)+\bar{c}_{\ell,t}(\boldsymbol{i}_\ell).\tag{5.7}$$

Then, the newly computed cumulative expected cost will be used to calculate the choosing probability at the next iteration as shown in Equation (5.6). The summary of the proposed ORW algorithm is listed as Algorithm 5.2.

Algorithm 5.2 The Online Recursive Weighting (ORW) Algorithm

```
Input: index set $\mathcal{I}_m, T, \{\eta_t=\sqrt{\frac{n}{t}}\}$
Output: $\boldsymbol{q}_m(\boldsymbol{I}_{m,1}), \boldsymbol{q}_m(\boldsymbol{I}_{m,2}), \ldots$
```

```
    ▷ Output sample points of layer $m$ subsets
1   for $\ell = 1, 2, \ldots, m$
    ▷ Initialize the normalized expected cost for all subsets in all layers
2       for $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$
3           Set $\bar{C}_{\ell,0}(\boldsymbol{i}_\ell) = 0$
4       end for
5   end for
6   Set $\bar{C}_{\ell,0}(\boldsymbol{i}_\ell) = 0, \forall \boldsymbol{i}_\ell \in \mathcal{I}_\ell, \ell = 1, 2, \ldots, m$
7   for $t = 1, 2, \ldots, T$
8       $\boldsymbol{I}_{0,t} = \mathbf{1}$
        ▷ Recursively select subsets in all layers
9       for $\ell = 0, 1, \ldots, m-1$
10          Randomly select an index $\boldsymbol{I}_{\ell+1,t} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{I}_{\ell,t}))$ according to the probability
            distribution specified in Equation (5.6)
11      end for
12  end for
13  Choose the subset indexed by $\boldsymbol{I}_{m,t}$ at iteration $t$
14  Choose the sampled point for subset $\mathcal{K}_m(\boldsymbol{I}_{m,t})$, i.e., $\boldsymbol{q}_m(\boldsymbol{I}_{m,t})$, as the final decision
15  for $\ell = 1, 2, \cdots, m$
    ▷ Get the normalized expected cost for all subsets in all layers based on the revealed cost function $f_t(\boldsymbol{x})$
    at iteration $t$
16      for $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$
17          for $\boldsymbol{j}_m \in \mathcal{H}_m(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$
18              Calculate $\Pr[\boldsymbol{I}_{m,t} = \boldsymbol{j}_m | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell]$ according to Equation (5.4)
19          end for
20          Calculate $\Pr[\boldsymbol{I}_{m,t} = \boldsymbol{j}_m | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell]$ according to Equation (5.4)
21          Update $\bar{C}_{\ell,t}(\boldsymbol{i}_\ell) = \bar{C}_{\ell,t-1}(\boldsymbol{i}_\ell) + \bar{c}_{\ell,t}(\boldsymbol{i}_\ell)$
22      end for
23  end for
```

Remark 5.5.1: (Technical differences with the traditional weighting methods) For the expert learning problem, the intuitive idea to attain a sublinear regret is to allocate more preference to the expert of smaller cumulative cost in a stochastic manner [108], which can be realized by the Exponential Weighting algorithm (or Hedge algorithm on a continuum). The Hedge algorithm observes the costs on each point, and updates the weight of a point based on its own cost only. Such a point-by-point weighting method fails in utilizing the cost correlation among neighboring decision nodes. Different from the Hedge algorithm, the ORW algorithm is aware of the correlation of neighboring points and groups highly correlated decision points as a high-level decision. As expressed in Equation (5.3), a common evaluation is conducted for each decision group, and a particular amount of common preference is allocated to each point in the group.

5.6 Regret Analysis for the Online Recursive Weighting Algorithm

Since the ONCL problem is a more general model than the OCO, the lower bounds for the OCO are still valid for the non-convex case. In the

OCO framework, a well-known lower bound of the regret is $O(\sqrt{T})$, and the reader can refer to [80] for sketch of the proof. In this section, we analyze the regret of the ORW algorithm (represented by $\mathfrak{R}$ in equations) and demonstrate it matches the above lower bound. The main technical result is summarized in the following theorem.

Theorem 5.6.1: With $\eta_t = \sqrt{n/t}$, the ORW algorithm guarantees that

$$\mathsf{regret}_T(\mathfrak{R}) \leq 2(\ln 2 + 1)nDL\sqrt{T} + 2\ln 2 \cdot nDL + \frac{\sqrt{n}}{2^m}DLT.$$

If m is further set to $\lceil \log_2 \sqrt{nT} \rceil$, it follows that

$$\mathsf{regret}_T(\mathfrak{R}) < (4n+1)DL\sqrt{T} + 2nDL.$$

Remark 5.6.1: (Dimensional dependency) Theorem 5.6.1 implies that the ORW algorithm attains a regret of $O(n\sqrt{T})$, with a mild polynomial dependency on the dimension.

Remark 5.6.2: [103] shows that when the decision set is uniformly fat, the Hedge algorithm can attain a regret of $O(\sqrt{T \log T})$. An interesting result on the ORW algorithm is that the only assumption on decision set is that it is bounded.

To carry out the analysis, we split the regret analysis of the ORW algorithm into two parts:

▷ The first part is the regret due to the "imperfect choice" among sample points, i.e.,

$$\mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}) \overset{\text{def}}{=} \sup_{f_1,\ldots,f_T \in \mathcal{F}} \left\{ \sum_{t \in \mathcal{T}} \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})\right] - \min_{\boldsymbol{i}_m \in \mathcal{I}_m} \sum_{t \in \mathcal{T}} c_{m,t}(\boldsymbol{i}_m) \right\},$$

where the first term is the cumulative cost incurred by the online algorithm (whose choice at iteration t is denoted by $\boldsymbol{I}_{m,t}$), and the second term is the minimum cumulative cost among sampled points.

▷ The second part of the regret is introduced by "imperfect discretization", which is expressed as

$$\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}) \overset{\text{def}}{=} \sup_{f_1,\ldots,f_T \in \mathcal{F}} \left\{ \min_{\boldsymbol{i}_m \in \mathcal{I}_m} \sum_{t \in \mathcal{T}} c_{m,t}(\boldsymbol{i}_m) - \min_{\boldsymbol{x} \in \mathcal{K}} \sum_{t \in \mathcal{T}} f_t(\boldsymbol{x}) \right\},$$

where the second term is the minimum cumulative cost over the decision set $\mathcal{K}$. Since the supremum of sum is less than or equal to the sum of supremum, we have the following lemma.

Lemma 5.6.1: $\mathsf{regret}_T(\mathfrak{R}) \leq \mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}) + \mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R})$.

In the following proposition, we derive a bound for the first-part regret of the ORW algorithm, which is related to the subproblem of choosing a point over the index set $\mathcal{I}_m$.

Lemma 5.6.2: With $\eta_t = \sqrt{n/t}$, the ORW algorithm guarantees that

$$\mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}) \leq 2(\ln 2 + 1)nDL\sqrt{T} + 2\ln 2 \cdot nDL.$$

Proof of Lemma 5.6.2. The ORW algorithm has a recursive decision-making structure to determine the final decision. Correspondingly, the regret due to "imperfect choice" can be further splited into multiple pieces which are introduced at each layer.

Suppose a nonempty subset $\mathcal{K}_l(\boldsymbol{i}_l)$ is chosen at layer $l \in \{0, 1, 2, \ldots, m-1\}$. In the next step, the ORW algorithm will further choose a subset whose index lies in $\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$. Among the subsets of $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$, there exists a local optimal subset (for example, $\boldsymbol{i}''_{\ell+1}$), whose performance is known in hindsight and potentially a regret loss due to imperfect choice at layer ℓ will be incurred by the online algorithm. Equation (5.8) bounds such cost difference (or regret loss) between $\sum_{t\in\mathcal{T}} \mathbb{E}[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell]$ and $\sum_{t\in\mathcal{T}} \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell+1,t} = \boldsymbol{i}''_{\ell+1}\right]$ at the ℓ-th layer, where $\boldsymbol{i}''_{\ell+1}$ is any element in $\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$.

In Equation (5.8), $\tilde{\mathcal{Q}} = \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$ and $\bar{\mathcal{Q}} = \mathcal{H}_m(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$. Equality $(E1)$ is obtained simply by the law of total probability. Equality $(E2)$ is due to the definition for $\bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})$ in (5.3). Equality $(E3)$ is based on the conditional probability in (5.6). Inequality $(E4)$ is due to the following inequality,

$$\sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell)} \frac{\exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right) \cdot \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)} \leq$$

$$-\tfrac{1}{\eta_t} \ln \sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \frac{\exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right) \cdot \exp(-\eta_t \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1}))}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_l(\boldsymbol{i}_\ell))} \exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)} + \tfrac{\eta_t}{2}. \tag{5.9}$$

The proof of Equation (5.9) is given in Appendix B.1. Equality $(E5)$ is due to the fact that $\bar{C}_{\ell+1,t}(\boldsymbol{i}_{\ell+1}) = \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1}) + \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})$. Equality $(E6)$ simplifies the expression for the log-sum-exp function by defining

$$\Phi_t(\alpha) \stackrel{\text{def}}{=} -\frac{1}{\alpha} \ln \sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\alpha \bar{C}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right),$$

$$
\begin{aligned}
&\sum_{t\in\mathcal{T}} \mathbb{E}[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell] \\
\overset{(E1)}{=}&\sum_{t\in\mathcal{T}} \sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} P^t_{\ell,\ell+1}(\boldsymbol{i}_\ell,\boldsymbol{i}_{\ell+1})\cdot \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}\right] \\
\overset{(E2)}{=}&\sum_{t\in\mathcal{T}} \sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} P^t_{\ell,\ell+1}(\boldsymbol{i}_\ell,\boldsymbol{i}_{\ell+1})\cdot \left[\bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\cdot 2^{m-\ell}L_m+\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m)\right] \\
=&\sum_{t\in\mathcal{T}}\left\{\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} P^t_{\ell,\ell+1}(\boldsymbol{i}_\ell,\boldsymbol{i}_{\ell+1})\cdot \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\cdot 2^{m-\ell}L_m+\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} P^t_{\ell,\ell+1}(\boldsymbol{i}_\ell,\boldsymbol{i}_{\ell+1})\cdot\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m)\right\} \\
=&\sum_{t\in\mathcal{T}}\left\{\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} P^t_{\ell,\ell+1}(\boldsymbol{i}_\ell,\boldsymbol{i}_{\ell+1})\cdot \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\cdot 2^{m-\ell}L_m+\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m)\right\} \\
\overset{(E3)}{=}&\sum_{t\in\mathcal{T}}\underbrace{\left[\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}}\frac{\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)}{\sum_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}\cdot\bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right]}_{\text{See (5.9), which is proved in Appendix B.1}}\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E4)}{\le}&\sum_{t\in\mathcal{T}}\left\{\underbrace{\left[-\frac{1}{\eta_t}\ln\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}}\frac{\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)\cdot\exp\left(-\eta_t\bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right)}{\sum_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}+\frac{\eta_t}{2}\right]}_{\text{See (5.9), which is proved in Appendix B.1}}\cdot 2^{m-\ell}L_m+\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m)\right\} \\
\overset{(E5)}{=}&\sum_{t\in\mathcal{T}}\left[-\frac{1}{\eta_t}\ln\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}}\frac{\exp\left(-\eta_t\bar{C}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right)}{\sum_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}+\frac{\eta_t}{2}\right]\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E6)}{=}&\sum_{t\in\mathcal{T}}\left[\Phi_t(\eta_t)-\Phi_{t-1}(\eta_t)+\frac{\eta_t}{2}\right]\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E7)}{=}&\left\{\Phi_T(\eta_T)+\underbrace{\sum_{t=1}^{T-1}\left(\Phi_t(\eta_t)-\Phi_t(\eta_{t+1})\right)}_{\text{See (5.10) which is proved in Appendix B.2}}-\Phi_0(\eta_1)+\sum_{t=1}^{T}\frac{\eta_t}{2}\right\}\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E8)}{\le}&\left\{\Phi_T(\eta_T)+\underbrace{\ln 2\cdot\sqrt{nT}}_{\text{See (5.10) which is proved in Appendix B.2}}-\Phi_0(\eta_1)+\sum_{t=1}^{T}\frac{\eta_t}{2}\right\}\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E9)}{\le}&\left\{\underbrace{\Phi_T(\eta_T)}_{\text{See (5.11) which is proved in Appendix B.3}}+\ln 2\cdot\sqrt{nT}+\ln 2\cdot\sqrt{n}+\sqrt{nT}\right\}\cdot 2^{m-\ell}L_m+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E10)}{\le}&\left\{\underbrace{\sum_{t\in\mathcal{T}}\mathbb{E}\left[\frac{c_{m,t}(\boldsymbol{I}_{m,t})-\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}}c_{m,t}(\boldsymbol{i}_m)}{2^{m-\ell}L_m}|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]}_{\text{See (5.11) which is proved in Appendix B.3}}+(\ln 2+1)\sqrt{nT}+\ln 2\cdot\sqrt{n}\right\}\cdot 2^{m-\ell}L_m \\
&+\sum_{t\in\mathcal{T}}\min_{\boldsymbol{i}_m\in\bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m) \\
\overset{(E11)}{=}&\sum_{t\in\mathcal{T}}\mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]+\left((\ln 2+1)\sqrt{nT}+\ln 2\cdot\sqrt{n}\right)\cdot 2^{m-\ell}L_m \\
\overset{(E12)}{=}&\sum_{t\in\mathcal{T}}\mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]+\underbrace{\left((\ln 2+1)nDL\sqrt{T}+\ln 2\cdot nDL\right)}_{\text{Denoted as }\phi}\cdot\frac{1}{2^\ell},\quad\forall\boldsymbol{i}''_{\ell+1}\in\tilde{\mathcal{Q}}
\end{aligned}
\tag{5.8}
$$

where $\alpha > 0$. Equality $(E7)$ rearranges the first set of terms and Inequality $(E8)$ uses the following bound result

$$\sum_{t=1}^{T-1}[\Phi_t(\eta_t) - \Phi_t(\eta_{t+1})] \leq \sum_{t=1}^{T-1} n \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) \leq \ln 2 \cdot \sqrt{nT}, \quad (5.10)$$

which is proved in Appendix B.2. Inequality $(E9)$ entails the following two results:

$$\begin{aligned}
&- \Phi_0(\eta_1) = \frac{1}{\eta_1} \ln |\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))| = \frac{1}{\sqrt{n}} \ln |\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))| \\
&\leq \frac{1}{\sqrt{n}} \ln 2^n \leq \ln 2 \cdot \sqrt{n},
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{\eta_t}{2} < \sum_{t=1}^{T} \frac{\sqrt{n}}{\left(\sqrt{t} + \sqrt{t+1}\right)} = \sum_{t=1}^{T} \sqrt{n} \cdot \left(\sqrt{t+1} - \sqrt{t}\right) \\
&= \sqrt{n} \cdot \left(\sqrt{T+1} - 1\right) \leq \sqrt{nT}.
\end{aligned}$$

Inequality $(E10)$ is due to the following inequality

$$\begin{aligned}
&\Phi_T(\eta_T) \\
&\leq \sum_{t \in \mathcal{T}} \mathbb{E}\left[\frac{c_{m,t}(\boldsymbol{I}_{m,t}) - \min\limits_{\boldsymbol{i}_m \in \bar{\mathcal{Q}}} c_{m,t}(\boldsymbol{i}_m)}{2^{m-\ell} L_m} | \boldsymbol{I}_{\ell+1,t} = \boldsymbol{i}''_{\ell+1}\right], \qquad (5.11)
\end{aligned}$$

which is proved in Appendix B.3. Equality $(E11)$ combines the first term and the last term. Equality $(E12)$ follows from the fact that $L_m = \sqrt{n}DL/2^m$.

For any cost functions $(f_1, f_2, \cdots, f_T)$, let $\boldsymbol{i}^*_m$ be the index of which the sample point has the smallest cumulative cost, i.e.,

$$\boldsymbol{i}^*_m \overset{\text{def}}{=} \arg\min_{\boldsymbol{i}_m \in \mathcal{I}_m} \sum_{t \in \mathcal{T}} c_{m,t}(\boldsymbol{i}_m).$$

By repeatedly applying the result in Equation (5.8), we have

$$\begin{aligned}
&\sum_{t\in\mathcal{T}} \mathbb{E}[c_{m,t}(\boldsymbol{I}_{m,t})] - \min_{\boldsymbol{i}_m\in\mathcal{I}_m} \sum_{t\in\mathcal{T}} c_{m,t}(\boldsymbol{i}_m) \\
=&\sum_{t\in\mathcal{T}} \mathbb{E}[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{0,t}=\mathbf{1}] - \sum_{t\in\mathcal{T}} c_{m,t}(\boldsymbol{i}_m^*) \\
\le&\sum_{t\in\mathcal{T}} \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{1,t}=\boldsymbol{v}_1(\boldsymbol{i}_m^*)\right] + \phi\cdot\frac{1}{2^0} - \sum_{t\in\mathcal{T}} c_{m,t}(\boldsymbol{i}_m^*) \\
\le&\sum_{t\in\mathcal{T}} \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{2,t}=\boldsymbol{v}_2(\boldsymbol{i}_m^*)\right] + \phi\cdot\frac{1}{2^1} + \phi\cdot\frac{1}{2^0} \\
&-\sum_{t\in\mathcal{T}} c_{m,t}(\boldsymbol{i}_m^*) \qquad\qquad (5.12)\\
\le&\cdots \\
\le&\sum_{t\in\mathcal{T}} \mathbb{E}\left[c_{m,t}(\boldsymbol{I}_{m,t})|\boldsymbol{I}_{m,t}=\boldsymbol{v}_m(\boldsymbol{i}_m^*)\right] + \phi\cdot\left(\frac{1}{2^{m-1}}+\cdots\frac{1}{2^0}\right) \\
&-\sum_{t\in\mathcal{T}} c_{m,t}(\boldsymbol{i}_m^*) \\
=&\phi\cdot\left(\frac{1}{2^{m-1}}+\cdots\frac{1}{2^0}\right) = 2\phi\cdot\left[1-\frac{1}{2^m}\right] \le 2\phi.
\end{aligned}$$

Since (5.12) holds for any cost functions $(f_1, f_2, \cdots, f_T)$, we have

$$\mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}) \le 2\phi = 2(\ln 2+1)nDL\sqrt{T} + 2\ln 2\cdot nDL.$$

This completes the proof. □

Remark 5.6.3: Even though a larger m leads to an exponential increase of the size of the sample set, the above equation implies that $\mathsf{regret}_{\mathsf{ImC},T}$ is always upper bounded by $O(\sqrt{T})$ in time scale, no matter what value m takes. On the other hand, the regret loss due to imperfect discretization can be reduced with larger m.

Now, we state the the following lemma.

Lemma 5.6.3: The ORW algorithm guarantees that

$$\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}) \le \frac{\sqrt{n}}{2^m} DLT.$$

Proof of Lemma 5.6.3. For any cost functions $(f_1, f_2, \cdots, f_T)$, let

$$\boldsymbol{x}^* \overset{\text{def}}{=} \arg\min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t\in\mathcal{T}} f_t(\boldsymbol{x}).$$

Suppose that the layer m subset containing point $\boldsymbol{x}^*$ is indexed by $\boldsymbol{i}_m^*$. The sample point for subset $\mathcal{K}_m(\boldsymbol{i}_m^*)$ is $\boldsymbol{q}_m(\boldsymbol{i}_m^*) \in \mathcal{K}$. Clearly, $\boldsymbol{q}_m(\boldsymbol{i}_m^*) = (q_1, q_2, \cdots, q_n)$ and $\boldsymbol{x}^* = (x_1, x_2, \cdots, x_n)$ are in the same sub-cube whose edge length is $D/2^m$. Thus, we have

$$\begin{aligned}
&||\boldsymbol{q}_m(\boldsymbol{i}_m^*) - \boldsymbol{x}^*||_2 \\
&= \sqrt{(q_1 - x_1)^2 + (q_2 - x_2)^2 + \cdots (q_n - x_n)^2} \\
&\leq \sqrt{\left(\frac{D}{2^m}\right)^2 + \left(\frac{D}{2^m}\right)^2 + \cdots + \left(\frac{D}{2^m}\right)^2} = \frac{D\sqrt{n}}{2^m}.
\end{aligned}$$

Then, we have

$$\begin{aligned}
&\min_{\boldsymbol{i}_m \in \mathcal{I}_m} \sum_{t \in \mathcal{T}} c_{m,t}(\boldsymbol{i}_m) - \min_{\boldsymbol{x} \in \mathcal{K}} \sum_{t \in \mathcal{T}} f_t(\boldsymbol{x}) \\
&\leq \sum_{t \in \mathcal{T}} c_{m,t}(\boldsymbol{i}_m^*) - \sum_{t \in \mathcal{T}} f_t(\boldsymbol{x}^*) \\
&= \sum_{t \in \mathcal{T}} f_t(\boldsymbol{q}_m(\boldsymbol{i}_m^*)) - f_t(\boldsymbol{x}^*) \\
&\overset{(E1)}{\leq} TL||\boldsymbol{q}_m(\boldsymbol{i}_m^*) - \boldsymbol{x}^*||_2 \leq \frac{\sqrt{n}}{2^m} DLT,
\end{aligned} \tag{5.13}$$

where inequality $(E1)$ is due to the Lipschitz condition. Since (5.13) holds for any cost functions $(f_1, f_2, \cdots, f_T)$, we conclude that $\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}) \leq \sqrt{n}DLT/2^m$. This completes the proof. □

Putting together the results in lemmas 5.6.1, 5.6.2, and 5.6.3, the results in Theorem 5.6.1 are proved.

5.7 Adaptive Sampling

The ORW algorithm described in Section 5.5, specifies a granularity parameter, m, and samples the costs on the bottommost layer (layer m). With m larger than $\log_2 \sqrt{nT}$, the ORW algorithm guarantees the regret to be $O(\sqrt{T})$. In many cases, however, the duration interval is long and unknown to the online player, and setting a large m to guarantee $O(\sqrt{T})$ regret might be costly to compute. In order to handle cases with unknown T and reduce this complexity, in this section, we devise an adaptive version of the ORW algorithm (called the AORW algorithm), which increases the granularity parameter gradually. The AORW algorithm denoted by $\mathfrak{R}^{\mathsf{adpt}}$ in equations.

$$\begin{aligned}\bar{c}'_{\ell,t}(\boldsymbol{i}_\ell) &\overset{\text{def}}{=} \mathbb{E}\left[\frac{c_{m_t,t}(\boldsymbol{I}_{m_t,t}) - \min\limits_{\boldsymbol{i}_{m_t}\in\mathcal{H}_{m_t}(\mathcal{K}_{\ell-1}(\boldsymbol{v}_{\ell-1}(\boldsymbol{i}_\ell)))} c_{m_t,t}(\boldsymbol{i}_{m_t})}{2^{m_t-\ell+1}L_{m_t}} | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell\right] \\ &= \sum_{\boldsymbol{j}_{m_t}\in\mathcal{H}_{m_t}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \frac{c_{m_t,t}(\boldsymbol{j}_{m_t}) - \min\limits_{\boldsymbol{i}_{m_t}\in\mathcal{H}_{m_t}(\mathcal{K}_{\ell-1}(\boldsymbol{v}_{\ell-1}(\boldsymbol{i}_\ell)))} c_{m_t,t}(\boldsymbol{i}_{m_t})}{2^{m_t-\ell+1}L_{m_t}} \cdot \Pr\left[\boldsymbol{I}_{m_t,t} = \boldsymbol{j}_{m_t} | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell\right].\end{aligned} \tag{5.15}$$

5.7.1 Online Recursive Weighting Algorithm with Adaptive Sampling

The AORW algorithm adopts an increasing granularity parameter m_t. We denote the time interval that satisfies $m_t \geq l$ as $\mathcal{T}_l$ or $[t_l, T]$. With increasing m_t, the index and the index set for the sampling subsets are updated to the new notation, $\boldsymbol{i}_{m_t}$ and $\mathcal{I}_{m_t}$, respectively. $\boldsymbol{q}_{m_t}(\boldsymbol{i}_{m_t})$ is used to denote the sample point for the subset of index $\boldsymbol{i}_{m_t}$, and the cost on the sampled point of the subset $\boldsymbol{i}_{m_t} \in \mathcal{I}_{m_t}$ at iteration t is denoted as

$$c'_{m_t,t}(\boldsymbol{i}_{m_t}) \overset{\text{def}}{=} f_t(\boldsymbol{q}_{m_t}(\boldsymbol{i}_{m_t})).$$

The AORW algorithm maintains the *cumulative expected cost*, $\bar{C}'_{\ell,t}(\boldsymbol{i}_\ell)$, for each subset in layers $\{1, 2, \ldots, m_t\}$. The cumulative expected cost is initialized to be zero, i.e., $\bar{C}'_{\ell,0}(\boldsymbol{i}_\ell) = 0$. During run time, the cumulative expected cost is used to determine the choosing probability of subsets in layers $\{1, 2, \ldots, m_t\}$. The selection probability for the subset $\mathcal{K}_{\ell+1}(\boldsymbol{i}_{\ell+1})$ conditioning on that $\mathcal{K}_\ell(\boldsymbol{i}_\ell)$ has been selected is

$$\begin{aligned}&\Pr\left[\boldsymbol{I}_{\ell+1,t} = \boldsymbol{i}_{\ell+1} | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell\right] \\ &= \frac{\exp\left(-\eta_t \bar{C}'_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\eta_t \bar{C}'_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}.\end{aligned} \tag{5.14}$$

Since there is a subset in each layer chosen in a recursive manner from the top layer to the bottom layer, the action sequence of the AORW algorithm at iteration t can be correspondingly denoted by a vector, $(\boldsymbol{I}_{0,t}, \boldsymbol{I}_{1,t}, \boldsymbol{I}_{0,t}, \ldots, \boldsymbol{I}_{m_t,t})$, where $\boldsymbol{I}_{0,t}$ is set to be $\mathbf{1}$ by default.

After the costs on the sampled points of the bottom-layer subsets are revealed, the AORW algorithm calculates the *normalized expected cost* for each subset in layers $\{1, 2, \ldots, m_t\}$, which is defined in Equation (5.15).

Then, the calculated *normalized expected cost* is used to update the cumulative expected cost, i.e.,

$$\bar{C}'_{\ell,t}(\boldsymbol{i}_\ell) = \bar{C}'_{\ell,t-1}(\boldsymbol{i}_\ell) + \bar{c}'_{\ell,t}(\boldsymbol{i}_\ell). \tag{5.16}$$

For the case when $m_{t+1} = m_t+1$, the AORW algorithm will maintain the cumulative expected cost for those subsets in the new layer, with previous values over $[1, t_{m_{t+1}} - 1]$ set to be 0, i.e.,

$$\bar{c}'_{m_{t+1},\tau}(\boldsymbol{i}_{m_{t+1}}) = 0, \text{ for } \tau = 0, 1, 2, \ldots, t_{m_{t+1}} - 1.$$

The summary of the AORW algorithm is listed in Algorithm 5.3.

5.7.2 Regret Analysis

By adaptive sampling, the algorithm parameter does not depend on the length of time interval. Meanwhile, an interesting result of the regret analysis is that adaptive sampling does not degrade the regret bound. We state the main result in the following theorem.

Theorem 5.7.1: With $\eta_t = \sqrt{n/t}$ and m_t being set to $\lceil \log_2 \sqrt{nt} \rceil$, the AORW algorithm achieves the following regret bound

$$\mathsf{regret}_T(\mathfrak{R}^{\mathsf{adpt}}) \leq \left(6n + 2\sqrt{n}\right) DL\sqrt{T}.$$

Let $\boldsymbol{i}^*_{m_t},\ t \in \mathcal{T}$ indicate the bottom subset that contains the optimal decision point $\boldsymbol{x}^*$ at iteration t, i.e.,

$$\boldsymbol{i}^*_{m_t} : \boldsymbol{x}^* \in \mathcal{K}_{m_t}(\boldsymbol{i}^*_{m_t}),\ t \in \mathcal{T}.$$

Similar to the regret analysis in Section 5.6, we define the regret due to the imperfect choice among sample points as

$$\mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}^{\mathsf{adpt}}) \overset{\text{def}}{=} \sup_{f_1,\ldots,f_T \in \mathcal{F}} \left\{ \sum_{t \in \mathcal{T}} \mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})\right] - \sum_{t \in \mathcal{T}} c'_{m_t,t}(\boldsymbol{i}^*_{m_t}) \right\},$$

and that due to the imperfect sampling as

$$\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}^{\mathsf{adpt}}) \overset{\text{def}}{=} \sup_{f_1,\ldots,f_T \in \mathcal{F}} \left\{ \sum_{t \in \mathcal{T}} c'_{m_t,t}(\boldsymbol{i}^*_{m_t}) - \min_{\boldsymbol{x} \in \mathcal{K}} \sum_{t \in \mathcal{T}} f_t(\boldsymbol{x}) \right\}.$$

Similar to Lemma 5.5.1, we have

$$\mathsf{regret}_T(\mathfrak{R}^{\mathsf{adpt}}) \leq \mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}^{\mathsf{adpt}}) + \mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}^{\mathsf{adpt}}).$$

In the following two lemmas, we bound the regret loss due to imperfect choice and sampling, respectively.

Lemma 5.7.1: With $\eta_t = \sqrt{n/t}$ and the $m_t = \lceil \log_2 \sqrt{t} \rceil$, the AORW algorithm guarantees that

$$\mathsf{regret}_{\mathsf{ImC},T}(\mathfrak{R}^{\mathsf{adpt}}) \le 6DLn\sqrt{T}.$$

The proof of Lemma 5.7.1 is analogous to that of Lemma 5.6.2. The details are given in Appendix B.4.

Lemma 5.7.2: With $m_t = \lceil \log_2 \sqrt{t} \rceil$, the AORW algorithm guarantees that

$$\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}^{\mathsf{adpt}}) \le 2DL\sqrt{nT}.$$

Proof of Lemma 5.7.2. For any cost functions $(f_1, f_2, \cdots, f_T)$, let

$$\boldsymbol{x}^* \overset{\text{def}}{=} \arg\min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t\in\mathcal{T}} f_t(\boldsymbol{x}).$$

Suppose that the bottom subset containing point $\boldsymbol{x}^*$ is indexed by $\boldsymbol{i}^*_{m_t}$, whose sample point is $\boldsymbol{q}_{m_t}(\boldsymbol{i}^*_{m_t}) \in \mathcal{K}$. Clearly, $\boldsymbol{q}_{m_t}(\boldsymbol{i}^*_{m_t}) = (q_{m_t,1}, q_{m_t,2}, \cdots, q_{m_t,n})$ and $\boldsymbol{x}^* = (x_1, x_2, \cdots, x_n)$ are in the same sub-cube whose edge length is $D/2^{m_t}$. Thus, we have

$$\begin{aligned}
&||\boldsymbol{q}_{m_t}(\boldsymbol{i}^*_{m_t}) - \boldsymbol{x}^*||_2 \\
&= \sqrt{(q_{m_t,1} - x_1)^2 + (q_{m_t,2} - x_2)^2 + \cdots (q_{m_t,n} - x_n)^2} \\
&\le \sqrt{\left(\frac{D}{2^{m_t}}\right)^2 + \left(\frac{D}{2^{m_t}}\right)^2 + \cdots + \left(\frac{D}{2^{m_t}}\right)^2} = \frac{D\sqrt{n}}{2^{m_t}} \\
&\le D\sqrt{\frac{n}{t}}.
\end{aligned}$$

Then we have

$$\begin{aligned}
&\sum_{t\in\mathcal{T}} c'_{m_t,t}(\boldsymbol{i}^*_{m_t}) - \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t\in\mathcal{T}} f_t(\boldsymbol{x}) \\
&= \sum_{t\in\mathcal{T}} c'_{m_t,t}(\boldsymbol{i}^*_{m_t}) - \sum_{t\in\mathcal{T}} f_t(\boldsymbol{x}^*) \\
&= \sum_{t\in\mathcal{T}} f_t(\boldsymbol{q}_m(\boldsymbol{i}^*_m)) - f_t(\boldsymbol{x}^*) \\
&\overset{(E1)}{\le} L \sum_{t\in\mathcal{T}} ||\boldsymbol{q}_{m_t}(\boldsymbol{i}^*_{m_t}) - \boldsymbol{x}^*||_2 \le 2DL\sqrt{nT},
\end{aligned} \tag{5.17}$$

where inequality $(E1)$ is due to the Lipschitz condition. Since (5.17) holds for any cost functions $(f_1, f_2, \cdots, f_T)$, we conclude that

$\mathsf{regret}_{\mathsf{ImD},T}(\mathfrak{R}) \leq 2DL\sqrt{nT}$. This completes the proof. □

The results in lemmas 5.7.1 and 5.7.2 immediately proves the result in Theorem 5.7.1.

Algorithm 5.3 Recursive Weighting with Adaptive Sampling

```
Input: $\{\eta_t = \sqrt{n/t}\}$, $\{m_t = \lceil \log_2 \sqrt{t} \rceil\}$
Output: $\boldsymbol{q}_{m_1}(\boldsymbol{I}_{m_1,1}), \boldsymbol{q}_{m_2}(\boldsymbol{I}_{m_2,2}), \ldots$
    ▷ Output sample points of layer-$m$ subsets
1  for $\ell = 1, 2, \cdots, m_1$
       ▷ Initialize the normalized expected cost for all subsets in all layers
2      for $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$
3          Set $\bar{C}_{\ell,0}(\boldsymbol{i}_\ell) = 0$
4      end for
5  end for
6  Set $\bar{C}_{\ell,0}(\boldsymbol{i}_\ell) = 0, \forall \boldsymbol{i}_\ell \in \mathcal{I}_\ell, \ell = 1, 2, \ldots, m_1$
7  for $t = 1, 2, \ldots, T$
8      $\boldsymbol{I}_{0,t} = \mathbf{1}$
       ▷ Recursively select subsets in all layers
9      for $\ell = 0, 1, \ldots, m_t - 1$
10         Select an index $\boldsymbol{I}_{\ell+1,t} \in \mathcal{H}_{\ell+1}(\boldsymbol{I}_{\ell,t})$ according to the probability distribution
           calculated in Equation (5.14)
11     end for
12 end for
13 Choose the subset indexed by $\boldsymbol{I}_{m_t,t}$ at iteration $t$
14 Choose the sampled point for subset $\mathcal{K}_{m_t}(\boldsymbol{I}_{m_t,t})$, i.e., $\boldsymbol{q}_{m_t}(\boldsymbol{I}_{m_t,t})$, as the final decision
15 for $\ell = 1, 2, \ldots, m_t$
       ▷ Get the normalized expected cost for all subsets in all layers based on the revealed cost function $f_t(\boldsymbol{x})$
       at iteration t
16     for $\boldsymbol{i}_\ell \in \mathcal{I}_\ell$
17         for $\boldsymbol{j}_{m_t} \in \mathcal{H}_{m_t}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$
18             Calculate Pr $\left[\boldsymbol{I}_{m,t} = \boldsymbol{j}_{m_t} | \boldsymbol{I}_{\ell,t} = \boldsymbol{i}_\ell\right]$ according to Equation (5.4)
19         end for
20         Calculate $\bar{c}_{\ell,t}(\boldsymbol{i}_\ell)$ according to Equation (5.15)
21         Update $\bar{C}'_{\ell,t}(\boldsymbol{i}_\ell) = \bar{C}'_{\ell,t-1}(\boldsymbol{i}_\ell) + \bar{c}'_{\ell,t}(\boldsymbol{i}_\ell)$
22     end for
23 end for
24 if $m_{t+1} > m_t$ do
25     for $\boldsymbol{i}_{m_{t+1}} \in \mathcal{I}_{m_{t+1}}$
26         Set $\bar{C}'_{m_{t+1},t}(\boldsymbol{i}_{m_{t+1}}) = 0$
27     end for
28 end if
```

5.8 Extensions: Expert Learning with Partial Information Feedback

We emphasize that our online non-convex optimization problem is based on *full information feedback*. That is, the whole cost function will be revealed at the end of each iteration. It is an interesting and important future direction to consider *partial information feedback* where only the cost value of the player's choice is revealed. The *partial information feedback* extension is motivated by many real-world systems in which the observer is not co-located with the controller and the feedback information is noisy, partial, or incomplete due to limited communication bandwidth. Some examples are online routing in data networks [30], power control in cellular networks [139], and the ad placement problem on a web page [138].

The bandit information feedback setting has been investigated in a huge number of works in the OCO framework such as [70, 17, 129, 56, 18, 83, 84], which is called the Bandit Convex Optimization (BCO) problem. Among those works, [70, 17, 129, 56] reported to attain a sublinear regret for the BCO problem, respectively, but none of them have attained the lower bound of the regret. For the special case of strongly convex and smooth losses, [18] obtained a regret of $\tilde{O}(\sqrt{T})$ in the unconstrained case, and [83] obtained the same rate even in the constrained case. Recently, a new algorithm was reported by Hazan *et al.* to attain a regret of $(\ln T)^{2n}\sqrt{T}$ [84]. This is the first algorithm to attain a $\tilde{O}(\sqrt{T})$ regret for the OCO model with bandit feedback. Different from the above works, [19, 99, 124, 97, 98] studied the model where the (expected) payoff function satisfies a Lipschitz condition with respect to the metric. [19, 99, 124] investigate this problem in a few specific metric spaces such as a one-dimensional real interval, while in [97, 98], the action set is from a general metric space. Their model is more general and the results in [98] show that there is an algorithm whose regret on any instance satisfies $\mathsf{regret}_T = \tilde{O}(T^{\frac{n+1}{n+2}})$, where n is the dimension of the action set.

Despite of the above results, the optimal online algorithm and tight regret bound for the online non-convex optimization problem with bandit feedback are still open, and attracts extensive recent interests in the community.

□ **End of chapter.**

6. CONCLUSION AND DISCUSSIONS

The past four decades have seen the great success of the online optimization methodology, which results in many classic online problems in diverse computational contexts, such as scheduling, caching, resource management, and online learning etc. With consistent efforts from the community, optimal online solutions for many online problems have been proposed. The problems having been solved include the Ski Rental problem, the metrical task system, the one-way trading problem, the Expert problem, the bandit problem and the Online Convex Optimization problem etc. In addition to that, this thesis has provided respective optimal online solutions for several fundamental online optimization problems, i.e., the Online QoS Buffer Management problem, the online trading problem, and the Online Non-Convex Learning problem, making a more complete picture for the online optimization area. Despite the above achievements, there are still many complicated online problems whose optimal algorithms are still open, such as the k-server problem, the preemptive QoS buffer management problem, the bandit problem in metric spaces and the reinforcement learning problem. In the following table, we summarize part of the classical solved and unsolved online problems in the literature. (There are also many diverse variants of the listed problems, for which I will not introduce the details.)

There is no doubt that the future efforts should focus on those unsolved online problems. In the following, we will introduce two additional promising research directions in the future.

6.1 General Analysis Methods for Online Optimization

As future work, a possible research direction is to come up with more advanced techniques to address complicated problems, which are applicable for general online optimization problems. There were several successful examples, such as the potential function method and the primal-dual analysis, which have been proved to be extremely useful methods for a broad class of online optimization problems, see the introductory book [39] and the survey paper [44]. Despite the success of the above frameworks, their applications in practice still

Online Problem	Application Area	Metric	Solved?
Ski Rental	Computing Systems	C. R.	Yes
Metrical Task System	Computing Systems	C. R.	Partially
k-Server	Computing Systems	C. R.	No
Page List Access	Computing Systems	C. R.	Yes
Job Shop Scheduling	Computing Systems	C. R.	Partially
Online Knapsack Problem	Computing Systems	C. R.	Yes
QoS Buffer Management	Computing Systems	C. R.	No$\rightarrow$Yes
Preemptive Buffer Management	Computing Systems	C. R.	No
Secretary Problem	Operations Research	C. R.	Yes
Online Search	Operations Research	C. R.	Yes
Expert Problem	Online Learning	Regret	Yes
Online Convex Optimization	Online Learning	Regret	Yes
Lipschitz Expert	Online Learning	Regret	No$\rightarrow$Yes
Non-/stochastic Bandit	Online Learning	Regret	Yes
Lipschitz Bandit	Online Learning	Regret	No
Reinforcement Learning	Online Learning	Regret	No

heavily depend on the properties of studied problem. Hence, it is promising to study online analysis frameworks which are general and easy to implement.

6.2 Other Practical Settings for Online Optimization

Currently, there are increasing research interests on many practical problem settings for online optimization, which are motivated by many practical applications. For example, Chen, in his several manuscripts [49, 50, 51], studied a class of online convex optimization problems with predictions. Prediction in online optimization is naturally motivated, since it is very common for a decision-maker to access partial or noisy information on the dynamics and states of studied systems. Compared with traditional online optimization framework, Chen's work involves a stochastic prediction error model. By using only a constant-sized prediction window, the proposed Averaging Fixed Horizon Control (AFHC) policy can simultaneously achieve sublinear regret and constant competitive ratio in expectation. Generally, the online optimization with prediction is a new area, and more investigations are needed to come up with general prediction frameworks and related methods to make full use of noisy prediction.

Moreover, motivated by the recent interests on distributed learning, there are also many works focusing on distributed online optimization problems. For example, Hosseini [85] and Lee [105] generalized the classic online convex optimization problem to a decentralized optimization framework within a network of agents. In their work, consensus-based gradient-descent algorithms were proposed for distributed online optimization. In their setting, each agent aims to drive its individual average regret, which is the average over time of the regret function evaluated at this agent's estimation for the choice that the whole network should make, to zero. An interesting message of their work is that the $O(\sqrt{T})$ regret can still be attained by leveraging the communication among agents. [132] addresses decentralized online optimization in non-stationary environments using mirror descent, and in [22], distributed online optimization is studied for strongly convex objective functions over time-varying networks. However, more general online optimization problems in a distributed environment are under-explored, needing more investigated from the community.

□ **End of chapter.**

APPENDIX

A. PART OF PROOFS OF CHAPTER 2

A.1 Proof of Lemma 2.5.3

By partially dualizing on the first set of nonequality constraints, we obtain the following Lagrangian function:

$$L(y, \boldsymbol{v}, \boldsymbol{\mu}) = y + \sum_{i=0}^{B-\lambda} \mu_i (v_{\lambda+i+1} B - y \sum_{j=1}^{\lambda+i} v_j),$$

where $\mu_i,\ i = 0, 1, \ldots, B - \lambda$ are the dual variables associated with the non-equality constraints.

The first-order optimality necessity condition can be expressed as

$$1 - \sum_{i=0}^{B-\lambda} \mu_i \sum_{j=1}^{\lambda+i} v_j = 0,$$

$$\mu_i B - y \sum_{k=i+1}^{B-\lambda} \mu_k = 0,\ i = 0, 1, \ldots, B - \lambda - 1.$$

Note that $v_{\min} \leq v_j \leq v_{\max}$ for $j = \lambda + 1, \lambda + 2, \ldots, B - 1$. Then, by the means of above equations, we can show that $\mu_i > 0,\ i = 0, 1, \ldots, B - \lambda$. Moreover, according to the complementary slackness condition,

$$\mu_i (v_{\lambda+i+1} B - y \sum_{j=1}^{\lambda+i} v_j) = 0,\ i = 0, 1, \ldots, B - \lambda.$$

This implies that the following equations always hold:

$$v_{\lambda+i+1} B - y \sum_{j=1}^{\lambda+i} v_j = 0,\ i = 0, 1, \ldots, B - \lambda.$$

Mathematically, this can be rewritten as

$$\frac{v_{\lambda+i+1}B}{\sum_{j=1}^{\lambda+i} v_j} = y,\ i = 0, 1, \ldots, B - \lambda.$$

This completes the proof.

A.2 Proof of Corollary 2.5.1

For the fractional case, a packet can be equally divided into arbitrarily small units. Here n is a large integer. Specifically, we divide a packet into n units. Let the *threshold-based* policy set a threshold value for each unit of packet. Then by optimizing the threshold values and setting the first step length $\lambda = \alpha B$, we attain a competitive ratio γ satisfying

$$\left(\frac{\gamma + Bn}{Bn}\right)^{Bn-\alpha Bn+1} - \left(\frac{\gamma + Bn}{Bn}\right)^{Bn-\alpha Bn} - \frac{\tilde{\theta}}{\alpha Bn} = 0.$$

Let $n \to \infty$, we have

$$\exp(\gamma(1-\alpha)) = \frac{\tilde{\theta}}{\alpha\gamma}. \tag{A.1}$$

Equation (A.1) defines an implicit function of γ with respect to α. By taking derivative, we have that when $\alpha = 1/\gamma$, γ takes the minimum value. Substituting $\alpha = 1/\gamma$ to Equation (A.1) yields

$$\gamma = \ln\tilde{\theta} + 1.$$

This also proves the optimality of the *threshold-based* online algorithm, as the competitive ratio for the fractional case is lower bounded by $\ln\tilde{\theta} + 1$.

Assuming b is an real number that is not less than λ, the threshold value to further admit packet when the queue length is b, denoted by $g(b)$, is

$$g(b) = \alpha\gamma v_{\min}\left(\frac{v_{\max}}{\alpha\gamma v_{\min}}\right)^{\frac{(b-\lambda)n}{Bn-\alpha Bn}}.$$

Let $n \to \infty$ and replace r with $\ln\tilde{\theta} + 1$. We have

$$g(b) = v_{\min}e^{\left(\ln\tilde{\theta}+1\right)\frac{b}{B}-1}.$$

A.3 Proof of Lemma 2.5.5

Here we only prove the nontrivial case where $b \geq l$. Assume the initial queue length is 0 and at time slot t^{max}, $b_{\omega}^{\mathsf{thBuffAlg}}(t)$ reaches the maximum value during the time period $[1, T]$, i.e., $b_{\omega}^{\mathsf{thBuffAlg}}(t^{\mathsf{max}}) = b$. We consider a special worst instance where there is no packet delivered during $[1, t^{\mathsf{max}}]$ and no packet buffered by thBuffAlg during $[t^{\mathsf{max}} + 1, T]$. In this case, we have the following claimed results:

1. The profit earned by thBuffAlg is at least $\sum_{i=1}^{b} v_i$. This is due to the fact that the investigated online algorithm is defined by a non-decreasing threshold values. It is easy to see that the minimum profit earned by thBuffAlg is $\sum_{i=1}^{b} v_i$ when the queue length increases from 0 to b.

2. The profit by OPT is at most $v_{b+1}B + \sum_{i=\lambda+1}^{b} v_i$. As analyzed in the proof for Lemma 2.5.2, there is no packet with value larger than or equal to v_{b+1} during $[1, T]$. So under the worst case, OPT will definitely buffer B packets with the value of $v_{b+1} - \delta$ where δ is an arbitrarily small positive value. This happens only when $b_{\omega}^{\mathsf{thBuffAlg}}(t)$ reaches b. Because we assume there are no packets delivered during $[1, t^{\mathsf{max}}]$, thus OPT will not buffer packet before t^{max}. After OPT buffers B packets with value $v_{b+1} - \delta$, the number of packets further buffered by OPT before t ($t > t^{\mathsf{max}}$) will not be larger than b minus the queue length under thBuffAlg, i.e., $b - b_{\omega}^{\mathsf{thBuffAlg}}(t)$, since there are no packets buffered by thBuffAlg during $[t^{\mathsf{max}} + 1, T]$ as assumed. The packet value will not be larger than the threshold of thBuffAlg, so the maximum profit earned by OPT during the time period when $b_{\omega}^{\mathsf{thBuffAlg}}(t)$ goes back to 0 from b is not larger than $\sum_{i=l+1}^{b} v_i$. Concluding the above, we can get that the maximum profit by OPT during $[1, T]$ will not be larger than $v_{b+1}B + \sum_{i=l+1}^{b} v_i$.

Based on the above analysis, we know that the maximum profit ratio of OPT and thBuffAlg when there is no packet delivered during $[1, t^{\mathsf{max}}]$ and no packet buffered during $[t^{\mathsf{max}} + 1, T]$ is at most $\left(v_{b+1}B + \sum\limits_{i=l+1}^{b} v_i\right) / \left(\sum\limits_{i=1}^{b} v_i\right)$. Actually, this ratio can be approximately realized by the input instance in Equation (A.2) (also shown in Figure A.1).

$$[(v_1, 0) \times \lambda, (v_2, 0), \ldots, (v_b, 0), \underbrace{(v_{b+1} - \delta, 0) \times B, (v_b - \delta, 1), (v_{b-1} - \delta, 1), \ldots, (v_{\lambda+1} - \delta, 1)}_{\text{The time slots that OPT selects to buffer.}}]. \tag{A.2}$$

For the case of allowing packet delivery during $[1, t^{\mathsf{max}}]$ or packet buffering during $[t^{\mathsf{max}}+1, T]$, we have the following analysis:

1. Assume under the worst case $\boldsymbol{\omega}$, there is a time period $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,end}}] \in [1, t^{\mathsf{max}}]$ when $b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t)$ keeps decreasing or unchanged due to the packet delivery. Without loss of generality, we assume $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,end}}]$ is the time period of which the beginning queue length $b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t^{\mathsf{Dec,beg}})$ (for simplicity, we denote $b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t^{\mathsf{Dec,beg}})$ as b^{Dec}) is the largest among decreasing periods, and assume at time slot $t^{\mathsf{Dec,rec}}$, the queue length returns to b^{Dec}. For details of the above instance, readers can also refer to Figure A.2. Because there's no time period with packet delivery after $t^{\mathsf{Dec,rec}}$ and OPT will definitely empty the buffer before the queue length under $\mathsf{thBuffAlg}$ reaches the maximum, thus $b_{\boldsymbol{\omega}}^{\mathsf{OPT}}(t^{\mathsf{Rec}}) = 0$ must hold. Further, there's no packet of value larger than $g(b^{\mathsf{Dec}})$ before $t^{\mathsf{Dec,beg}}$, so we have $b_{\boldsymbol{\omega}}^{\mathsf{OPT}}(t^{\mathsf{Dec,beg}}-1) = 0$ for some worst case. From the above analysis, we have that the queue length of $\mathsf{thBuffAlg}$ and OPT at $t^{\mathsf{Dec,rec}}$ will both return to the state at $t^{\mathsf{Dec,beg}}$. Then under some particular worst case, the profit ratio during $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,rec}}]$ must be larger than $\textsc{cr}_b(\mathsf{thBuffAlg})$ (otherwise, we just "delete" the input segment during $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,rec}}]$, and this will not decrease the profit ratio). Moreover, the adversary can repeat the input instance during $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,rec}}]$ for arbitrary times, since the queue length of OPT and $\mathsf{thBuffAlg}$ both go back to the state at $t^{\mathsf{Dec,beg}}$. Assume the input instance during $[1, t^{\mathsf{Dec,beg}}-1]$ is $\boldsymbol{\omega}_1$, and the input instance during $[t^{\mathsf{Dec,beg}}, t^{\mathsf{Dec,rec}}]$ is $\boldsymbol{\omega}_2$, we can construct the input instance $\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 \times j + [(v_{\min}), B]$ which lies in subset $\Omega_{b^{\mathsf{Dec}}}^{\mathsf{thBuffAlg}}$. When j is large enough, we can get a larger local competitive ratio than $\textsc{cr}_b(\mathsf{thBuffAlg})$, i.e.,

$$\textsc{cr}_b(\mathsf{thBuffAlg}) < \textsc{cr}_{b^{\mathsf{Dec}}}(\mathsf{thBuffAlg}),\ b^{\mathsf{Dec}} < b.$$

   In conclusion, the above analysis implies that when $\textsc{cr}_b(\mathsf{thBuffAlg}) \geq \textsc{cr}_{b'}(\mathsf{thBuffAlg})$ for $b' < b$, there will be no packets delivered during $[1, t^{\mathsf{max}}]$.

2. Assume under the worst case $\boldsymbol{\omega}$, there is a time period $[t^{\mathsf{Inc,beg}}, t^{\mathsf{Inc,end}}]$ when $b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t)$ keeps increasing or unchanged due to the packet buffering. Similar to the analysis in (1), we can also find the buffering time period with the largest *ending* queue length as shown in Figure A.3. We can also deduce that the buffer at time $t^{\mathsf{Inc,rec}}$ and $t^{\mathsf{Inc,end}}$ must be full, since there will be no packet

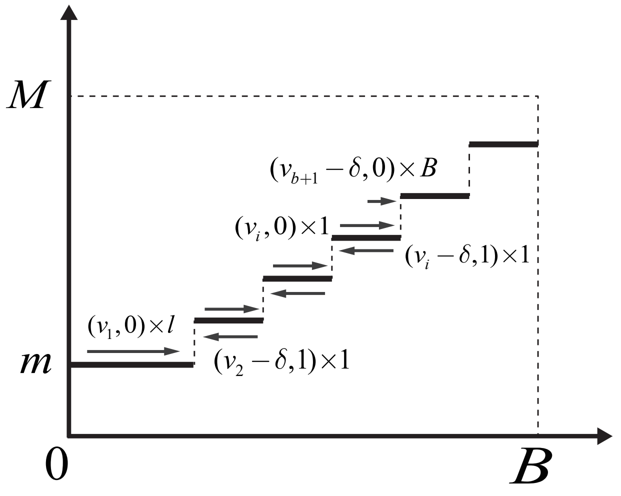


Fig. A.1: The constructed worst case.

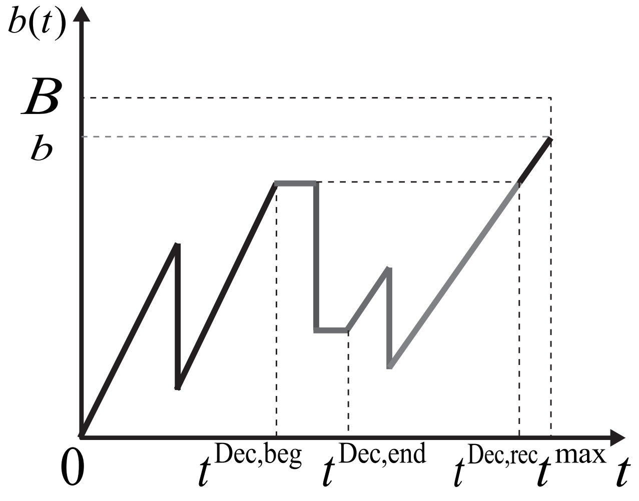


Fig. A.2: The delivery time period.

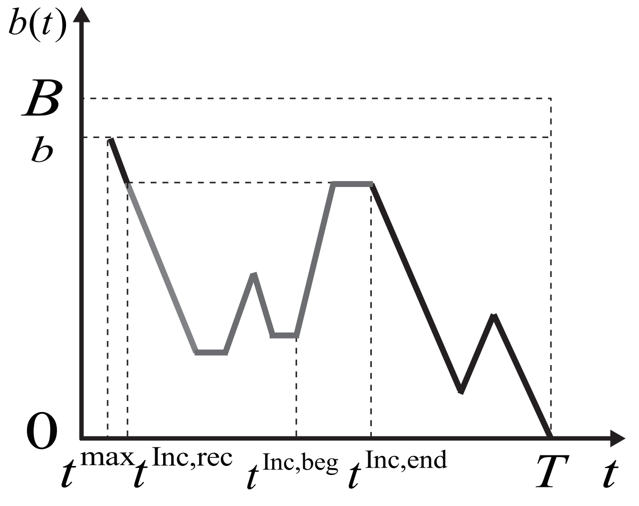


Fig. A.3: The buffering time period.

of value larger than $g(b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t^{\mathsf{Inc,beg}}))$ or $g(b_{\boldsymbol{\omega}}^{\mathsf{thBuffAlg}}(t^{\mathsf{Inc,rec}}))$ afterwards. Thus under the worst case, the profit ratio during $[t^{\mathsf{Inc,rec}}, t^{\mathsf{Inc,end}}]$ is larger than $\text{CR}_{\Omega_b^{\mathsf{thBuffAlg}}}(\mathsf{thBuffAlg})$. Then we can construct a case similar to (1) and prove that there is a subset whose local competitive ratio is larger than that of $\Omega_b^{\mathsf{thBuffAlg}}$. At last, we can get that no packets buffered during $[t^{\mathsf{max}}+1, T]$ when $\text{CR}_b(\mathsf{thBuffAlg}) \geq \text{CR}_{b'}(\mathsf{thBuffAlg})$ for $b' > b$.

Concluding (1) and (2), we can prove that when $\text{CR}_b(\mathsf{thBuffAlg}) \geq \text{CR}_{b'}(\mathsf{thBuffAlg})$, $b' > b$, there is no packet delivered during $[1, t^{\mathsf{max}}]$ and no packet buffered during $[t^{\mathsf{max}}+1, T]$ under the worst case. In this case, the worst input instance should be as shown in Figure A.1, and the corresponding competitive ratio is

$$\text{CR}_b(\mathsf{thBuffAlg}) = \frac{v_{b+1}B + \sum\limits_{i=\lambda+1}^{b} v_i}{\sum\limits_{i=1}^{b} v_i}.$$

This completes the proof.

A.4 Proof of Lemma 2.5.6

Given the length of the first step is λ, optimizing the threshold values of $\text{CR}(\mathsf{thBuffAlg})$ is equivalent to solving the following problem:

$$\begin{array}{lll} \min & y & \\ \text{s.t.} & y \geq \left(v_{b+1}B + \sum\limits_{i=\lambda+1}^{b} v_i\right) / \left(v_{\min}\lambda + \sum\limits_{i=\lambda+1}^{b} v_i\right), & b = \lambda, \lambda+1, \ldots, B. \\ \text{var} & y,\ v_{\min} \leq v_i \leq v_{\max}, & i = \lambda+1, \lambda+2, \ldots, B. \end{array}$$

The vector of variables is $[y, v_{\lambda+1}, v_{\lambda+2}, \ldots, v_B]$. By partially dualizing on the first set of nonequality constraints, we obtain the following Lagrangian function:

$$L(y, \boldsymbol{v}, \boldsymbol{\mu}) = y + \sum_{i=0}^{B-\lambda} \mu_i \left[\left(v_{\lambda+i+1}B + \sum_{j=\lambda+1}^{l+i} v_j\right) - y\left(v_{\min}\lambda + \sum_{j=\lambda+1}^{\lambda+i} v_j\right)\right],$$

where $\mu_i,\ i = 0, 1, \cdots, B-\lambda$ are the dual variables associated with the non-equality constraints.

The first-order optimality necessary condition can be expressed as

$$\begin{aligned}
&1 - \sum_{i=0}^{B-\lambda} \mu_i \left(v_{\min}\lambda + \sum_{j=\lambda+1}^{\lambda+i} v_j \right) = 0, \\
&\mu_i B + (1-y) \sum_{k=i+1}^{B-\lambda} \mu_k = 0, \qquad\qquad i = 0, 1, \ldots, B-\lambda-1.
\end{aligned}$$

Note that $v_{\min} \leq v_j \leq v_{\max}$ for $j = \lambda+1, \lambda+2, \ldots, B-1$. Note also that y is equal to $\textsc{cr}(\mathsf{thBuffAlg})$, so definitely, we have $1 - y < 0$. Combining the above equations, we establish the result that $\mu_i > 0,\ i = 0, 1, \ldots, B-\lambda$. Moreover, according to the complementary slackness condition

$$\mu_i \left[\left(v_{\lambda+i+1} B + \sum_{j=\lambda+1}^{\lambda+i} v_j \right) - y \left(v_{\min}\lambda + \sum_{j=\lambda+1}^{\lambda+i} v_j \right) \right] = 0,\ i = 0, 1, \ldots, B-\lambda,$$

the following equations hold:

$$\left(v_{\lambda+i+1} B + \sum_{j=\lambda+1}^{\lambda+i} v_j \right) - y \left(v_{\min}\lambda + \sum_{j=\lambda+1}^{\lambda+i} v_j \right) = 0,\ i = 0, 1, \ldots, B-\lambda.$$

Alternatively, theses equations can be rewritten as

$$\frac{v_{\lambda+i+1} B + \sum_{j=\lambda+1}^{\lambda+i} v_j}{v_{\min}\lambda + \sum_{j=\lambda+1}^{\lambda+i} v_j} = y,\ i = 0, 1, \ldots, B-\lambda.$$

This completes the proof.

B. PART OF PROOFS OF CHAPTER 5

B.1 Proof of Inequality (5.9)

The proof relies on the following Hoeffding's Lemma.

Lemma B.1.1: (Hoeffding's Lemma) Let Z be any real-valued random variable with expected value $\mathbb{E}[Z] = 0$ and such that $a \leq Z \leq b$ almost surely. Then, for all $\lambda \in \mathbb{R}$,

$$\mathbb{E}\left[\exp(\lambda Z)\right] \leq \exp\left(\frac{\lambda^2(b-a)^2}{8}\right).$$

We thus define a random variable Y whose probability mass function is specified as

$$\begin{aligned}&\Pr\left[Y = \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right] \\ &= \frac{\exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_l))} \exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)},\end{aligned} \tag{B.1}$$

for $\forall \boldsymbol{i}_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$.

Since $\bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1}) \in [0,1], \forall \boldsymbol{i}_{\ell+1}$, we have that $0 \leq Y \leq 1$ for sure. However, random variable Y may not have zero mean. We thus denote another random variable $Z = Y - \mathbb{E}[Y]$. Clearly, $\mathbb{E}[Z] = 0$ and $-1 \leq Z \leq 1$ for sure. We further let $\lambda = -\eta_t$. Thus, by Hoeffding's Lemma, we have

$$\mathbb{E}[\exp(-\eta_t Z)] \leq \exp\left(\frac{(-\eta_t)^2(1-(-1))^2}{8}\right) = \exp\left(\frac{\eta_t^2}{2}\right). \tag{B.2}$$

In addition, we have

$$\begin{aligned}&\mathbb{E}[\exp(-\eta_t Z)] = \mathbb{E}[\exp(-\eta_t(Y - \mathbb{E}[Y]))] \\ &= \mathbb{E}[\exp(-\eta_t Y)] \cdot \exp[\eta_t \mathbb{E}[Y]].\end{aligned} \tag{B.3}$$

Combining (B.2) and (B.3), we get

$$\mathbb{E}[\exp(-\eta_t Y)] \cdot \exp\left(\eta_t \mathbb{E}[Y]\right) \leq \exp\left(\frac{\eta_t^2}{2}\right). \tag{B.4}$$

Taking logarithm and rearranging the items in (B.4), we have

$$\mathbb{E}[Y] \leq -\frac{1}{\eta_t} \ln \mathbb{E}[\exp(-\eta_t Y)] + \frac{\eta_t}{2}. \tag{B.5}$$

Now applying the probability mass function (B.1) into (B.5), we get that

$$\begin{aligned}
\mathbb{E}[Y] &= \sum_{\boldsymbol{i}_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \frac{\exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right) \cdot \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})}{\sum_{\boldsymbol{i}'_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)} \\
&\leq -\frac{1}{\eta_t} \ln \mathbb{E}[\exp(-\eta_t Y)] + \frac{\eta_t}{2}. \\
&= -\frac{1}{\eta_t} \ln \sum_{\boldsymbol{i}_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \frac{\exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right) \cdot \exp(-\eta_t \bar{c}_{\ell+1,t}(\boldsymbol{i}_{\ell+1}))}{\sum_{\boldsymbol{i}'_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\eta_t \bar{C}_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)} + \frac{\eta_t}{2}.
\end{aligned}$$

B.2 Proof of Equality (5.10)

To prove (5.10), we only need to show that

$$\Phi_t(\eta_t) - \Phi_t(\eta_{t+1}) \leq \ln 2 \cdot n \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} \right). \tag{B.6}$$

The second inequality of (5.10) simply follows from $\eta_t = \sqrt{\frac{n}{t}}$. We next prove (B.6).

We call a vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_m) \in \mathbb{R}^m$ *a non-increasing non-negative vector* if $z_1 \geq z_2 \geq \cdots \geq z_m \geq 0$. We then define a function

$$\Psi(\delta, \boldsymbol{z}) \overset{\text{def}}{=} -\frac{1}{\delta} \ln \sum_{j=1}^{m} \exp\left(-\delta z_j\right), \quad \delta > 0,$$

where $\boldsymbol{z} = (z_1, z_2, \cdots, z_m)$ is a non-increasing non-negative vector in $\mathbb{R}^m$. We further define a function $\hbar(\cdot)$ mapping from a non-increasing non-negative vector to another non-increasing non-negative vector, where $\hbar(\boldsymbol{z})$ changes all largest elements of $\boldsymbol{z}$ into the second-largest element of $\boldsymbol{z}$. Namely, if the first k elements $\boldsymbol{x}$ are the largest, i.e., $z_1 = z_2 = \cdots = z_k > z_{k+1} \geq \cdots \geq z_m$, then

$$\hbar(\boldsymbol{z}) = (\underbrace{z_{k+1}, z_{k+1}, \ldots, z_{k+1}}_{\text{In total } k}, z_{k+1}, z_{k+2}, \ldots, z_m).$$

Clearly, if we apply function $\hbar(\cdot)$ to a non-increasing non-negative vector $\boldsymbol{z}$ for m times, we get a constant vector $\boldsymbol{z}_{\min} = (z_m, z_m, \cdots, z_m)$.

We have the following lemma.

Lemma B.2.1: For $\mu > \nu > 0$, any non-increasing non-negative vector $\boldsymbol{z}$, we have

$$\Psi(\mu, \boldsymbol{z}) - \Psi(\nu, \boldsymbol{z}) \leq \Psi(\mu, \hbar(\boldsymbol{z})) - \Psi(\nu, \hbar(\boldsymbol{z})). \tag{B.7}$$

Proof of Lemma B.2.1. (i) If $\boldsymbol{z}$ is a constant vector, i.e., $z_1 = z_2 = \cdots = z_m$, then $\hbar(\boldsymbol{z}) = \boldsymbol{z}$ and thus (B.7) holds as an equality.

(ii) If $\boldsymbol{z}$ is not a constant vector, we assume that the largest element of $\boldsymbol{z}$ is z and there are in total $k < m$ largest elements in $\boldsymbol{z}$, i.e., $z = z_1 = z_2 = \cdots = z_k > z_{k+1} \geq z_{k+2} \cdots \geq z_m$.

Then

$$\begin{aligned}\Psi(\mu, \boldsymbol{z}) - \Psi(\nu, \boldsymbol{z}) &= -\frac{1}{\mu} \ln \left(k \exp(-\mu z) + \sum_{j=k+1}^{m} \exp(-\mu z_j) \right) \\ &+ \frac{1}{\nu} \ln \left(k \exp(-\nu z) + \sum_{j=k+1}^{m} \exp(-\nu z_j) \right) \overset{\text{def}}{=} \Upsilon(z),\end{aligned} \tag{B.8}$$

where the last equality defines a function with respect to z satisfying $z > z_{k+1}$. Taking the derivative on $\Upsilon(z)$ with respect to z, we get

$$\frac{\partial \Upsilon(z)}{\partial z} = \frac{k \exp(-\mu z)}{k \exp(-\mu z) + \sum\limits_{j=k+1}^{m} \exp(-\mu z_j)} - \frac{k \exp(-\nu z)}{k \exp(-\nu z) + \sum\limits_{j=k+1}^{m} \exp(-\nu z_j)}.$$

Define

$$\Theta(\sigma) \overset{\text{def}}{=} \frac{k \exp(-\sigma z)}{k \exp(-\sigma z) + \sum_{j=k+1}^{m} \exp(-\sigma z_j)}.$$

Then we have that

$$\frac{\partial \Upsilon(z)}{\partial z} = \Theta(\mu) - \Theta(\nu). \tag{B.9}$$

Taking derivative on $\Theta(\sigma)$ with respect to σ, we get

$$\frac{\partial \Theta(\sigma)}{\partial \sigma} = \frac{k \exp(-\sigma z) \sum_{j=k+1}^{m} (z_j - z) \exp(-\sigma z_j)}{\Big(k \exp(-\sigma z) + \sum_{j=k+1}^{m} \exp(-\sigma z_j) \Big)^2}.$$

Since $z > z_j$, for $j = k+1, k+2, \ldots, m$, we have $\frac{\partial \Theta(\sigma)}{\partial \sigma} < 0$. Because $\mu > \nu$, we have

$$\frac{\partial \Upsilon(z)}{\partial z} = \Theta(\mu) - \Theta(\nu) < 0, \ \forall z > z_{i+1}.$$

Thus, $\Upsilon(z) < \Upsilon(z_{i+1})$. By noting that $\Upsilon(z_{i+1}) = \Psi(\mu, \hbar(\boldsymbol{z})) - \Psi(\nu, \hbar(\boldsymbol{z}))$, we thus have

$$\Psi(\mu, \boldsymbol{z}) - \Psi(\nu, \boldsymbol{z}) < \Psi(\mu, \hbar(\boldsymbol{z})) - \Psi(\nu, \hbar(\boldsymbol{z})).$$

Part (i) and part (ii) complete the proof. □

Based on Lemma B.2.1, we can immediately obtain the following result.

Lemma B.2.2: For $\mu > \nu > 0$ and any non-increasing non-negative vector $\boldsymbol{z} \in \mathbb{R}^m$, we have

$$\Psi(\mu, \boldsymbol{z}) - \Psi(\nu, \boldsymbol{z}) \leq \left(\frac{1}{\nu} - \frac{1}{\mu}\right) \ln m. \tag{B.10}$$

Proof of Lemma B.2.2. From Lemma B.2.1, we have

$$\begin{aligned}\Psi(\mu, \boldsymbol{z}) - \Psi(\nu, \boldsymbol{z}) &\leq \Psi(\mu, \hbar(\boldsymbol{z})) - \Psi(\nu, \hbar(\boldsymbol{z})) \\ &\leq \Psi(\mu, \hbar(\hbar(\boldsymbol{z}))) - \Psi(\nu, \hbar(\hbar(\boldsymbol{z}))) \\ &\leq \cdots \\ &\leq \Psi(\mu, \boldsymbol{z}_{\min}) - \Psi(\nu, \boldsymbol{z}_{\min}) \\ &= \left(\frac{1}{\nu} - \frac{1}{\mu}\right) \ln m,\end{aligned}$$

where $\boldsymbol{z}_{\min} = (z_m, z_m, \ldots, z_m)$. This completes the proof. □

We can sort all $\bar{C}_{\ell+1,t}(\boldsymbol{i}_{\ell+1})$'s where $\boldsymbol{i}_{\ell+1} \in \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$ in a descending order and construct a non-increasing non-negative vector $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))|}$. Then based on Lemma B.2.1, we have

$$\begin{aligned}&\Phi_t(\eta_t) - \Phi_t(\eta_{t+1}) = \Psi(\eta_t, \boldsymbol{z}) - \Psi(\eta_{t+1}, \boldsymbol{z}) \\ &\leq \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) \ln |\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))| \leq \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) n \ln 2,\end{aligned} \tag{B.11}$$

where the last inequality follows from $|\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))| \leq 2^n$.

This completes the proof for inequality (5.10).

B.3 Proof of Inequality (5.11)

First, we have

$$\begin{aligned}
&\exp\left(\eta_T \max_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\boldsymbol{i}_\ell)}[-\bar{C}_{\ell+1,T}(\boldsymbol{i}_{\ell+1})]\right)\\
&\le \sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\boldsymbol{i}_\ell)} \exp\left(-\eta_T\bar{C}_{\ell+1,T}(\boldsymbol{i}_{\ell+1})\right). \qquad \text{(B.12)}
\end{aligned}$$

Taking logarithm and dividing by $-\frac{1}{\eta_T} < 0$ in both sides of (B.12), we can get that

$$\begin{aligned}
\Phi_T(\eta_T) &= -\frac{1}{\eta_T}\ln \sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))} \exp\left(-\eta_T\bar{C}_{\ell+1,T}(\boldsymbol{i}_{\ell+1})\right)\\
&\ge \min_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_l(\boldsymbol{i}_\ell))} \bar{C}_{\ell+1,T}(\boldsymbol{i}_{\ell+1})\\
&\ge \bar{C}_{\ell+1,T}(\boldsymbol{i}''_{\ell+1}) \quad \left(\forall \boldsymbol{i}''_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))\right)\\
&= \sum_{t=1}^{T} \bar{c}_{\ell+1,t}(\boldsymbol{i}''_{\ell+1})\\
&= \sum_{t=1}^{T} \mathbb{E}\left[\frac{c_{m,t}(\boldsymbol{I}_{m,t}) - \min_{\boldsymbol{i}_m\in\mathcal{H}_m(\mathcal{K}_{\ell-1}(\boldsymbol{v}_{\ell-1}(\boldsymbol{i}_\ell)))} c_{m,t}(\boldsymbol{i}_m)}{2^{m-\ell+1}L_m} |\boldsymbol{I}_{\ell+1,t} = \boldsymbol{i}''_{\ell+1}\right],
\end{aligned} \qquad \text{(B.13)}$$

where the second last equality follows from the definition of $\bar{C}_{\ell,t}(\boldsymbol{i}_\ell)$ in (5.7) and the last equality follows from the definition of $\bar{c}_{\ell,t}(\boldsymbol{i}_\ell)$ in (5.3) and $\boldsymbol{v}_\ell(\boldsymbol{i}''_{\ell+1}) = \boldsymbol{i}_\ell$.

This completes the proof.

B.4 Proof of Lemma 5.7.1

Analogous to Equation (5.8), we have Equation (B.14) which characterizes the regret loss in each layer. In Equation (B.14), $\tilde{\mathcal{Q}} = \mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$ and $\bar{\mathcal{Q}} = \mathcal{H}_{m_t}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))$. Equality $(E1)$ is according to the definition for the normalized expected cost shown in Equation (5.15). Equality $(E2)$ is simply by the choosing probability for subsets in each layer. Inequality $(E3)$ has been proved in Section (B.1). Equality $(E4)$ follows from the update law for the cumulative expected cost. Inequality $(E5)$ is based on the Equation (5.10) and (5.11), as well as the following inequality

$$\begin{aligned}
&-\Phi_{t_{\ell+1}-1}(\eta_{t_{\ell+1}})\\
&=\frac{1}{\eta_{t_{\ell+1}}}\ln\sum_{\boldsymbol{i}_{\ell+1}\in\mathcal{H}_{\ell+1}(\mathcal{K}_l(\boldsymbol{i}_\ell))}\exp\left(-\eta_{t_{\ell+1}}\bar{C}_{\ell+1,t_{\ell+1}-1}(\boldsymbol{i}_{\ell+1})\right)\\
&=\frac{1}{\eta_{t_{\ell+1}}}\ln|\mathcal{H}_{\ell+1}(\mathcal{K}_\ell(\boldsymbol{i}_\ell))|\\
&\leq\sqrt{nt_{\ell+1}}.
\end{aligned}$$

Inequality $(E6)$ is proved in Section B.3.

$$
\begin{aligned}
&\sum_{t\in\mathcal{T}_{\ell+1}} \mathbb{H}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell] \\
&= \sum_{t\in\mathcal{T}_{\ell+1}} \sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \Pr[\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell]\cdot\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}\right] \\
&\overset{(E1)}{=} \sum_{t\in\mathcal{T}_{\ell+1}} \sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \Pr[\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell]\cdot\left[\bar{c}'_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\cdot 2^{m_t-\ell}L_{m_t}+\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t})\right] \\
&= \sum_{t\in\mathcal{T}_{\ell+1}} \left\{\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \Pr[\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}_{\ell+1}|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}_\ell]\cdot\bar{c}'_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\cdot 2^{m_t-\ell}L_{m_t}+\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t})\right\} \\
&\overset{(E2)}{=} \sum_{t\in\mathcal{T}_{\ell+1}} \left\{\left[\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \frac{\exp\left(-\eta_t\bar{C}'_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}'_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}\cdot\bar{c}'_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right]\cdot 2^{m_t-\ell}L_{m_t}+\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t})\right\} \\
&\overset{(E3)}{\le} \sum_{t\in\mathcal{T}_{\ell+1}} \left\{\left[-\frac{1}{\eta_t}\ln\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \frac{\exp\left(-\eta_t\bar{C}'_{\ell+1,t-1}(\boldsymbol{i}_{\ell+1})\right)\cdot\exp\left(-\eta_t\bar{c}'_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}'_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}+\frac{\eta_t}{2}\right]\cdot 2^{m_t-\ell}L_{m_t}+\min_{\boldsymbol{i}_{m}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t})\right\} \\
&\overset{(E4)}{=} \sum_{t\in\mathcal{T}_{\ell+1}} \left[-\frac{1}{\eta_t}\ln\sum_{\boldsymbol{i}_{\ell+1}\in\tilde{\mathcal{Q}}} \frac{\exp\left(-\eta_t\bar{C}'_{\ell+1,t}(\boldsymbol{i}_{\ell+1})\right)}{\sum\limits_{\boldsymbol{i}'_{\ell+1}\in\tilde{\mathcal{Q}}}\exp\left(-\eta_t\bar{C}'_{\ell+1,t-1}(\boldsymbol{i}'_{\ell+1})\right)}+\frac{\eta_t}{2}\right]\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&= \sum_{t\in\mathcal{T}_{\ell+1}} \left[\Phi_t(\eta_t)-\Phi_{t-1}(\eta_t)+\frac{\eta_t}{2}\right]\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&= \left\{\Phi_T(\eta_T)+\sum_{t=t_{\ell+1}}^{T-1}\left(\Phi_t(\eta_t)-\Phi_t(\eta_{t+1})\right)-\Phi_{t_{\ell+1}-1}(\eta_{t_{\ell+1}})+\sum_{t=t_{\ell+1}}^{T}\frac{\eta_t}{2}\right\}\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&\overset{(E5)}{\le} \left\{\Phi_T(\eta_T)+\ln 2\cdot n\left(\frac{1}{\eta_T}-\frac{1}{\eta_{t_{\ell+1}}}\right)-\Phi_{t_{\ell+1}-1}(\eta_{t_{\ell+1}})+\sum_{t=t_{\ell+1}}^{T}\frac{\eta_t}{2}\right\}\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&\le \left\{\Phi_T(\eta_T)+\sqrt{nT}+\sqrt{nt_{\ell+1}}+\sqrt{nT}\right\}\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&\overset{(E6)}{\le} \left\{\sum_{t\in\mathcal{T}_{\ell+1}}\mathbb{E}\left[\frac{c'_{m_t,t}(\boldsymbol{I}_{m_t,t})-\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t})}{2^{m_t-\ell}L_{m_t}}|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]+3\sqrt{nT}\right\}\cdot 2^{m_t-\ell}L_{m_t}+\sum_{t\in\mathcal{T}_{\ell+1}}\min_{\boldsymbol{i}_{m_t}\in\bar{\mathcal{Q}}} c'_{m_t,t}(\boldsymbol{i}_{m_t}) \\
&\le \sum_{t\in\mathcal{T}_{\ell+1}}\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]+3\sqrt{nT}\cdot 2^{m_t-\ell}L_{m_t} \\
&= \sum_{t\in\mathcal{T}_{\ell+1}}\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell+1,t}=\boldsymbol{i}''_{\ell+1}\right]+3DLn\sqrt{T}\cdot\frac{1}{2^\ell},\quad \forall\boldsymbol{i}''_{\ell+1}\in\tilde{\mathcal{Q}}
\end{aligned}
\tag{B.14}
$$

$$\begin{aligned}
&\sum_{t\in\mathcal{T}}\mathbb{E}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})]-\sum_{t\in\mathcal{T}}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})=\sum_{t\in\mathcal{T}}\mathbb{E}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{0,t}=\boldsymbol{i}^*_0]-\sum_{t\in\mathcal{T}}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})\\
&\overset{(E1)}{=}\sum_{t\in\mathcal{T}_1}\mathbb{E}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{0,t}=\boldsymbol{i}^*_0]-\sum_{t\in\mathcal{T}_1}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})\\
&\le\sum_{t\in\mathcal{T}_1}\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{1,t}=\boldsymbol{i}^*_1\right]-\sum_{t\in\mathcal{T}_1}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})+3DLn\sqrt{T}\cdot\frac{1}{2^0}\\
&\overset{(E2)}{=}\sum_{t\in\mathcal{T}_2}\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{1,t}=\boldsymbol{i}^*_1\right]-\sum_{t\in\mathcal{T}_2}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})+3DLn\sqrt{T}\cdot\frac{1}{2^0}\\
&\le\sum_{t\in\mathcal{T}_2}\mathbb{E}\left[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{1,t}=\boldsymbol{i}^*_2\right]-\sum_{t\in\mathcal{T}_2}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})+3DLn\sqrt{T}\cdot\left(\frac{1}{2^0}+\frac{1}{2^1}\right)\\
&\le\cdots\le 3DLn\sqrt{T}\cdot\left(\frac{1}{2^0}+\frac{1}{2^1}+\cdots+\frac{1}{2^m}\right)\le 6DLn\sqrt{T}.
\end{aligned}\tag{B.15}$$

With the results in Equation (B.14), we have Equation (B.15). In Equation (B.15), the equalities follow from the fact that

$$\begin{aligned}
&\sum_{t\in\mathcal{T}_\ell}\mathbb{E}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}^*_\ell]-\sum_{t\in\mathcal{T}_\ell}c'_{m_t,t}(\boldsymbol{i}^*_{m_t})\\
&=\sum_{t\in\mathcal{T}_{\ell+1}}\mathbb{E}[c'_{m_t,t}(\boldsymbol{I}_{m_t,t})|\boldsymbol{I}_{\ell,t}=\boldsymbol{i}^*_l]-\sum_{t\in\mathcal{T}_{\ell+1}}c'_{m_t,t}(\boldsymbol{i}^*_{m_t}).
\end{aligned}$$

This completes the proof.

□ **End of chapter.**

BIBLIOGRAPHY

[1] Battery storagae power station. https://en.wikipedia.org/wiki/Battery_storage_power_station.

[2] Bloomberg, the world nears peak fossil. www.bloomberg.com/.

[3] Caiso archive. http://www.caiso.com.

[4] Generator offer curve in PJM. http://www.pjm.com/~/media/documents/manuals/m15.ashx.

[5] Global warming: Data centres to consume three times as much energy in next decade, experts warn. available at http://www.independent.co.uk/.

[6] Lambert W function. available at https://en.wikipedia.org/wiki/Lambert_W_function.

[7] National renewable energy laboratory. http://wind.nrel.gov.

[8] New York ISO (NYISO). http://www.nyiso.com/.

[9] Nord Pool market. http://www.nordpoolspot.com/.

[10] The official website of Denmark. http://denmark.dk/en/green-living/.

[11] Pennsylvania-New Jersey-Maryland (PJM). http://www.pjm.com/.

[12] Renewables 2016 global status report. http://www.ren21.net/status-of-renewables/global-status-report/.

[13] Technical report. https://www.dropbox.com/s/ddebwmcki39923f/tech.report.pdf.

[14] Wikipedia access traces. http://www.wikibench.eu/.

[15] Wind capacity. http://www.power-technology.com/features/feature-biggest-wind-farms-in-the-world-texas/.

[16] Wholesale electricity market design initiatives in the United States: Survey and research needs. *EPRI,Technical Results, available at https://www.epri.com/#/pages/product/000000003002009273/*, 2016.

[17] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pages 263–274, 2008.

[18] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40, 2010.

[19] R. Agrawal. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33(6):1926–1951, 1995.

[20] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén. Competitive queue policies for differentiated services. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 431–440, 2000.

[21] M. Ajtai, N. Megiddo, and O. Waarts. Improved algorithms and analysis for secretary problems and generalizations. *SIAM Journal on Discrete Mathematics*, 14(1):1–27, 2001.

[22] M. Akbari, B. Gharesifard, and T. Linder. Distributed online convex optimization on time-varying directed graphs. *IEEE Transactions on Control of Network Systems*, 2015.

[23] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

[24] N. Andelman. Randomized qeue management for DiffServ. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2005.

[25] N. Andelman and Y. Mansour. Competitive management of non-preemptive queues with multiple values. *Distributed Computing*, 2003: 166-180.

[26] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proceedings of ACM-SIAM*

Symposium on Discrete Algorithms (SODA), pages 761–770, 2003.

[27] D. Ardia, K. Boudt, P. Carl, K. M. Mullen, and B. Peterson. Differential evolution (deoptim) for non-convex portfolio optimization. 2010.

[28] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[29] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[30] B. Awerbuch and R. Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

[31] Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. *Algorithmica*, 43(1-2):81–96, 2005.

[32] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 16–28, 2007.

[33] N. Bansal, N. Buchbinder, A. Madry, and J. Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 267–276. IEEE, 2011.

[34] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog (n)-competitive algorithm for metrical task systems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 711–719. ACM, 1997.

[35] Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric ramsey-type phenomena. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 463–472. ACM, 2003.

[36] G. Bathurst, J. Weatherill, and G. Strbac. Trading wind generation in short term energy markets. *IEEE Transactions on Power Systems*, 17(3):782–789, 2002.

[37] E. Bitar, R. Rajagopal, P. Khargonekar, and et al. Bringing wind energy to market. *IEEE Transactions on Power Systems*, 27(3):1225–1235, 2012.

[38] H. Böckenhauer, D. Komm, R. Královič, and P. Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014.

[39] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[40] A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 39(4):745–763, 1992.

[41] A. Botterud, Z. Zhou, J. Wang, and et al. Wind power trading under uncertainty in lmp markets. *IEEE Transactions on Power Systems*, 27(2):894–903, 2012.

[42] C. Brunetto and G. Tina. Optimal hydrogen storage sizing for wind power plants in day ahead electricity market. *IET Renewable Power Generation*, 1(4):220, 2007.

[43] S. Bubeck, N. Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[44] N. Buchbinder, J. S. Naor, et al. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.

[45] E. D. Castronuovo and J. Lopes. On the optimization of the daily operation of a wind-hydro power plant. *Power Systems, IEEE Transactions on*, 19(3):1599–1606, 2004.

[46] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44(3):427–485, 1997.

[47] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[48] C.-K. Chau, G. Zhang, and M. Chen. Cost minimizing online algorithms for energy storage management with worst-case guarantee. *IEEE Transactions on Smart Grid*, 7(6):2691–2702, 2016.

[49] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew. Online convex optimization using predictions. In *ACM SIGMETRICS Performance Evaluation Review*, volume 43, pages 191–204. ACM, 2015.

[50] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):193–206, 2016.

[51] N. Chen, L. Gan, S. H. Low, and A. Wierman. Distributional analysis for model predictive deferrable load control. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6433–6438. IEEE, 2014.

[52] M. Chowdhury, M. Rao, Y. Zhao, T. Javidi, and A. Goldsmith. Benefits of storage control for wind power producers in power markets. *IEEE Transactions on Sustainable Energy*, 2016.

[53] P. Chuprikov, S. Nikolenko, and K. Kogan. Priority queueing with multiple packet characteristics. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1418–1426, 2015.

[54] R. Combes, S. Magureanu, A. Proutiere, and C. Laroche. Learning to rank: Regret lower bounds and efficient algorithms. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):231–244, 2015.

[55] T. Cover. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

[56] O. Dekel, R. Eldan, and T. Koren. Bandit smooth convex optimization: Improving the bias-variance tradeoff. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2926–2934, 2015.

[57] B. Dunn, H. Kamath, and J.-M. Tarascon. Electrical energy storage for the grid: a battery of choices. *Science*, 334(6058):928–935, 2011.

[58] R. El-Yaniv, A. Fiat, R. Karp, and G. Turpin. Competitive analysis of financial games. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–333, 1992.

[59] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 2001.

[60] M. Englert and M. Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.

[61] M. Englert and M. Westermann. Considering suppressed packets improves buffer management in quality of service switches. *SIAM Journal on Computing*, 41(5):1166–1192, 2012.

[62] ERCOT electricity market. available at `http://www.ercot.com`.

[63] S. Eric, Z. Sam, G. Tony, and N. Grace. A primer on wholesale market design. *Harvard Electricity Policy Group*, 2011.

[64] S. Ertekin, L. Bottou, and C. Giles. Non-convex online support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):368–381, 2011.

[65] A. Fabbri, T. Roman, J. Lopes, and et al. Assessment of the cost associated with wind generation prediction errors in a liberalized electricity market. *IEEE Transactions on Power Systems*, 20(3):1440–1446, 2005.

[66] M. Feldman and J. S. Naor. Non-preemptive buffer management for latency sensitive packets. *Journal of Scheduling*, 20(4):337–353, 2017.

[67] A. Fiat, Y. Mansour, H. Yi, and U. Nadav. Competitive queue management for latency sensitive packets. In *Proceedings of ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 228–237, 2008.

[68] A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.

[69] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 454–463. IEEE, 1990.

[70] A. Flaxman, A. Kalai, and H. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 385–394, 2005.

[71] P. R. Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, pages 189–206, 1983.

[72] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[73] G. Gasso, L. Pappaioannou, M. Spivak, and L. Bottou. Batch and online learning algorithms for nonconvex neyman-pearson classification. *ACM Transactions on Intelligent Systems and Technology*, 2(3):28, 2011.

[74] M. Goldwasser. A survey of buffer management policies for packet switches. *ACM SIGACT News*, 41(1):100–128, 2010.

[75] J. González, D. Muela, R. Ruiz, L. Santos, and A. González. Stochastic joint optimization of wind generation and pumped-storage units in an electricity market. *Power Systems, IEEE Trans. on*, 23(2), 2008.

[76] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar. Aggressive datacenter power provisioning with batteries. *ACM Transactions on Computing Systems*, 31(1):2:1–2:31, 2013.

[77] A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

[78] Y. Guo and Y. Fang. Electricity cost saving strategy in data centers by using energy storage. *IEEE Transactions Parallel and Distributed Systems*, 24(6):1149–1160, 2013.

[79] T. Guzella and W. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[80] E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2016.

[81] E. Hazan and S. Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13(Oct):2903–2922, 2012.

[82] E. Hazan and S. Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[83] E. Hazan and K. Levy. Bandit convex optimization: Towards tight bounds. In *Advances in Neural Information Processing Systems (NIPS)*, pages 784–792, 2014.

[84] E. Hazan and Y. Li. An optimal algorithm for bandit convex optimization. *arXiv preprint arXiv:1603.04350.*

[85] S. Hosseini, A. Chapman, and M. Mesbahi. Online distributed convex optimization on dynamic networks. *IEEE Transactions on Automatic Control*, 61(11):3545–3550, 2016.

[86] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pages 761–768, 2009.

[87] D. R. Jiang and W. B. Powell. Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS Journal on Computing*, 27(3):525–543, 2015.

[88] A. Kalai and S. Vempala. Efficient algorithms for universal portfolios. *Journal of Machine Learning Research*, 3(Nov):423–440, 2002.

[89] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[90] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1-4):79–119, 1988.

[91] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, and B. Schieber. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[92] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. In *Proceedings of ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 591–600, 2001.

[93] A. Kim, V. Liaghat, J. Qin, and A. Saberi. Online energy storage management: an algorithmic approach. In *Proc. APPROX*, 2016.

[94] J. H. Kim and W. B. Powell. Optimal energy commitments with storage and intermittent supply. *Operations Research*, 59(6):1347–1360, 2011.

[95] J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. In *Advances in neural information processing systems (NIPS)*, pages 287–293, 1998.

[96] R. Kleinberg and A. Slivkins. Sharp dichotomies for regret minimization in metric spaces. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 827–846. Society for Industrial and Applied Mathematics, 2010.

[97] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

[98] R. Kleinberg, A. Slivkins, and E. Upfal. Bandits and experts in metric spaces. *arXiv preprint arXiv:1312.1277*, 2013.

[99] R. D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, pages 697–704, 2005.

[100] K. Kobayashi, S. Miyazaki, and Y. Okabe. Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 328–336, 2009.

[101] M. Korpaas, A. T. Holen, and R. Hildrum. Operation and sizing of energy storage for wind power plants in a market system. *International Journal of Electrical Power & Energy Systems*, 25(8):599–606, 2003.

[102] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.

[103] W. Krichene, M. Balandat, C. Tomlin, and A. Bayen. The Hedge algorithm on a continuum. In *the 32nd International Conference on Machine Learning (ICML-15)*, pages 824–832, 2015.

[104] P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68, 2002.

[105] S. Lee, A. Nedich, and M. Raginsky. Stochastic dual averaging for decentralized online optimization on time-varying communication graphs. *IEEE Transactions on Automatic Control*, 2017.

[106] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew. Online algorithms for geographical load balancing. In *Proc. IGCC*, 2012.

[107] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391, 2013.

[108] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[109] M. Liu and F. F. Wu. Risk management in a competitive electricity market. *Int. Journal of Elec. Power & Energy Systems*, 29(9), 2007.

[110] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser. Renewable and cooling aware workload management for sustainable data centers. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 175–186, 2012.

[111] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew. Greening geographical load balancing. In *Proc. ACM SIGMETRICS*, 2011.

[112] J. Lorenz, K. Panagiotou, and A. Steger. Optimal algorithms for k-search with application in option pricing. *Algorithmica*, 2009.

[113] Z. Lotker and B. Patt-Shamir. Nearly optimal FIFO buffer management for DiffServ. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*, pages 134–143, 2002.

[114] X. Luo, J. Wang, M. Dooner, and J. Clarke. Overview of current development in electrical energy storage technologies and the application potential in power system operation. *Applied Energy*, 137, 2015.

[115] S. Magureanu, A. Proutiere, M. Isaksson, and B. Zhang. Online learning of optimally diverse rankings. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):32, 2017.

[116] O.-A. Maillard and R. Munos. Online learning in adversarial Lipschitz environments. *Machine Learning and Knowledge Discovery in Databases*, pages 305–320, 2010.

[117] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[118] Y. Mansour, B. Patt-Shamir, and O. Lapid. Optimal smoothing schedules for real-time streams. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*, pages 21–29, 2000.

[119] L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255, 2000.

[120] J. Matevosyan and L. Soder. Minimization of imbalance cost trading wind power on the short-term power market. *IEEE Transactions on Power Systems*, 21(3):1396–1404, 2006.

[121] S. Moharir, S. Sanghavi, and S. Shakkottai. Online load balancing under graph constraints. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 363–364, 2013.

[122] E. Mohr, I. Ahmad, and G. Schmidt. Online algorithms for conversion problems: a survey. *Surveys in Operations Research and Management Science*, 19(2):87–104, 2014.

[123] J. Morales, A. Conejo, and J. Pérez-Ruiz. Short-term trading for a wind power producer. *IEEE Transactions on Power Systems*, 25(1):554–564, 2010.

[124] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 216–227. SIAM, 2007.

[125] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for Internet-scale systems. In *Proc. ACM SIGCOMM*, 2009.

[126] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 449–456, 2012.

[127] L. Rao, X. Liu, and W. Liu. Minimizing electricity cost: Optimization of distributed Internet data centers in a multi-electricity-market environment. In *Proc. IEEE INFOCOM*, 2010.

[128] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[129] A. Saha and A. Tewari. Improved regret guarantees for online smooth convex optimization with bandit feedback. In *AISTATS*, pages 636–642, 2011.

[130] D. Sculley and G. Wachman. Relaxed online svms for spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422, 2007.

[131] S. S. Seiden. A guessing game and randomized online algorithms. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 592–601. ACM, 2000.

[132] S. Shahrampour and A. Jadbabaie. Distributed online optimization in dynamic environments using mirror descent. *IEEE Transactions on Automatic Control*, 2017.

[133] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[134] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):71–83, 2014.

[135] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[136] M. S. Talebi, Z. Zou, R. Combes, A. Proutiere, and M. Johansson. Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 2017.

[137] S. Uryasev. Conditional value-at-risk: Optimization algorithms and applications. In *Computational Intelligence for Financial Engineering, 2000.(CIFEr) Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on*, pages 49–57. IEEE, 2000.

[138] F. Wauthier, M. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning (ICML)*, pages 109–117, 2013.

[139] W. Wong and C. Sung. Robust convergence of low-data rate-distributed controllers. *IEEE transactions on automatic control*, 49(1):82–87, 2004.

[140] L. Yang, L. Deng, M. H. Hajiesmaili, C. Tan, and W. S. Wong. An optimal algorithm for online non-convex learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):25, 2018.

[141] L. Yang, M. H. Hajiesmaili, H. Yi, and M. Chen. Online offering strategies for storage-assisted renewable power producer in hour-ahead market. *arXiv preprint arXiv:1612.00179*, 2016.

[142] L. Yang, M. H. Hajiesmaili, H. Yi, and M. Chen. Hour-ahead offering strategies in electricity market for power producers

with storage and intermittent supply. In *Proceedings of ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 21–22, 2017.

[143] L. Yang, W. S. Wong, and M. H. Hajiesmaili. An optimal randomized online algorithm for qos buffer management. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):36, 2017.

[144] H. H. Zhang, J. Ahn, X. Lin, and C. Park. Gene selection using support vector machines with non-convex penalty. *bioinformatics*, 22(1):88–95, 2005.

[145] L. Zhang, T. Yang, R. Jin, and Z. Zhou. Online bandit learning for a special class of non-convex losses. In *AAAI*, pages 3158–3164, 2015.

[146] Z. Zhang, Z. Li, and C. Wu. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):23, 2017.

[147] Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*, pages 566–576, 2008.

[148] A. Zhu. Analysis of queueing policies in QoS switches. *Journal of Algorithms*, 53(2):137–168, 2004.

[149] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003.