DEMAND-SIDE MANAGEMENT FOR ENERGY-EFFICIENT DATA CENTER OPERATIONS WITH RENEWABLE ENERGY AND DEMAND RESPONSE

A Dissertation

by

SOONGEOL KWON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2017

Major Subject: Industrial Engineering

ABSTRACT

In recent years, we have noticed tremendous increase of energy consumption and carbon pollution in the industrial sector, and many energy-intensive industries are striving to reduce energy cost and to have a positive impact on the environment. In this context, this dissertation is motivated by opportunities to reduce energy cost from demand-side perspective. Specifically, industries have an opportunity to reduce energy consumption by improving *energy-efficiency* in their system operations. By improving utilization of their resources, they can reduce waste of energy, and thus, they are able to prevent paying unnecessary energy cost. In addition, because of today‘s high penetration of renewable generation (e.g. wind or solar), many industries consider *renewable energy* as a promising solution to reduce energy cost and carbon pollution, and they have tried to utilize renewable energy to meet their power demand by installing on-site generation facilities (e.g. PV panels on roof top) or making a contract with renewable generation farms. Moreover, it is becoming common for energy markets to allow industries to directly purchase electricity from them while providing the industries with day-ahead and real-time electricity price information. In this situation, industries have an opportunity to adjust purchase and consumption of energy in response to time-varying electricity price and intermittent renewable generation to reduce their energy procurement cost, which are called *demand response*.

Considering these opportunities, it is anticipated that the industrial sector can save a significant amount of energy cost, however, time-varying behavior, uncertainty and stochasticity in system operations, power demand, renewable energy, and electricity price make it challenging to determine optimal operational decision. Motivated by the aforementioned opportunities as well as challenges, this dissertation focuses on developing decision-making methodologies tailored for demand-side energy system operations to improve *energy-efficiency* based on energy-aware system operations and reduce energy procurement cost by utilizing *renewable energy* and *demand response* in an integrated fashion to optimally reduce energy cost.

For practical application, this dissertation considers real-world practices in data centers including their operations management and power procurement for the following research tasks: (i) develop a server provisioning algorithm that dynamically adapts server operations in response to heterogeneous and time-varying workloads to reduce energy consumption while providing performance guarantees based on time-stability; (ii) propose stochastic optimization models for optimal energy procurement to determine purchase and consumption of energy based on day-ahead and real-time energy market operations considering utilization of renewable energy based on demand response; (iii) suggest a decision-making model that integrate the proposed server provisioning algorithm with energy procurement to achieve energy-efficiency in data center operations to reduce both energy consumption and energy cost against variability and uncertainty. In terms of methodologies, this study uses operations research techniques including deterministic and stochastic models, such as, queueing analysis, mixed-integer program, Markov decision process, two-stage stochastic program, and probabilistic constrained program.

In conclusion, this dissertation claims that renewable energy, demand response, and energy storage are worth to be considered for data center operations to reduce energy consumption and procurement cost. Although variability and uncertainty in system operations, renewable generation, and electricity price make it challenging to determine optimal operational decisions, numerical results show that the proposed optimization problems can be efficiently solved by the developed algorithm. The proposed decision-making methodologies can also be extended to other industries, and thus, this dissertation study would be a good starting point to study demand-side management in energy system operations.

DEDICATION

I dedicate this dissertation to my family.

ACKNOWLEDGMENTS

Above all, thank you God for your mercy and blessing. My life belongs to you and I will follow your way through my entire life.

I would like to express my sincere gratitude to my advisor, Dr. Natarajan Gautam, for the continuous support of my Ph.D. study. Besides my advisor, I would like to thank my committee members, Dr. Yu Ding, Dr. Lewis Ntaimo, and Dr. Le Xie, for their guidance throughout the course of this research.

I would like to thank my wife, Moon Sun Jeon, for her faith in me, encouragement, and understanding. I would like to express my love to my daughters, Junseo Kwon and Junwoo Kwon. Thanks also go to my parents, Ki Joo Kwon and Moon Sook Back, and my parents-in-law, Euybai Jeon and Eun Ae Kim, for their endless support.

I would like to say many thanks to Pastor Sun Yeop Lee and Mrs. Sung Sook Lee for their prayer. I am thankful to my friends at Korean Church of A&M for their friendship.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

Faculty Committee Recognition

This work was supervised by a dissertation committee consisting of committee chair, Professor Natarajan Gautam, and committee members, Professors Yu Ding and Lewis Ntaimo of the Department of Industrial and Systems Engineering, and Professor Le Xie of the Department of Electrical and Computer Engineering.

Collaborator Contributions

Part of this dissertation (Chapter 4) was conducted in collaboration with Professor Yunjian Xu of the Department of Engineering Systems and Design at the Singapore University of Technology and Design. All other work for the dissertation was completed by the student independently.

Funding Sources

This dissertation is based upon work partially supported by the AFOSR under Contract No.FA9550-13-1-0008.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In recent years, many industries have witnessed a tremendous increase in energy consumption that has resulted in enormous expenses as well as carbon pollution. For this reason, energy-intensive industries, such as data centers, are striving to reduce energy cost and to have a positive impact on the environment. In this context at energy intensive industries, the so-called demand-side management, which addresses how to procure energy and how to manage system operation for minimizing energy cost, has received significant attention and is regarded as a promising research topic in both academia and industries. While considering energy consumer's perspective, this dissertation focuses on developing decision making methodologies designed (i) to improve energy-efficiency based on energy-aware system operations and (ii) to reduce energy procurement cost by utilizing renewable energy and demand response for demand-side management. Figure 1.1 captures various aspects of demand-side management, contrastively against supply-side management.

Although the methodologies developed in this dissertation can be extended to any industry, the first part of this dissertation comprising of *Sections 2* and *3*, mainly focuses on developing models and techniques to improve energy-efficiency in data center operations. In general, data centers have an issue of low utilization of servers to mitigate the effects of stochastic, heterogeneous and time-varying workloads, and thus they incur unnecessary cost for waste of energy, which is not used for processing workloads. While there are tremendous opportunities to conserve energy consumption in data centers, due to the inherent uncertainty and variability in the loads, developing provably effective methods to manage servers in data centers has been a challenge.

In Section 2, this dissertation models data centers as a system of multiple parallel single-server queues while considering a scenario where multiple classes of requests ar-

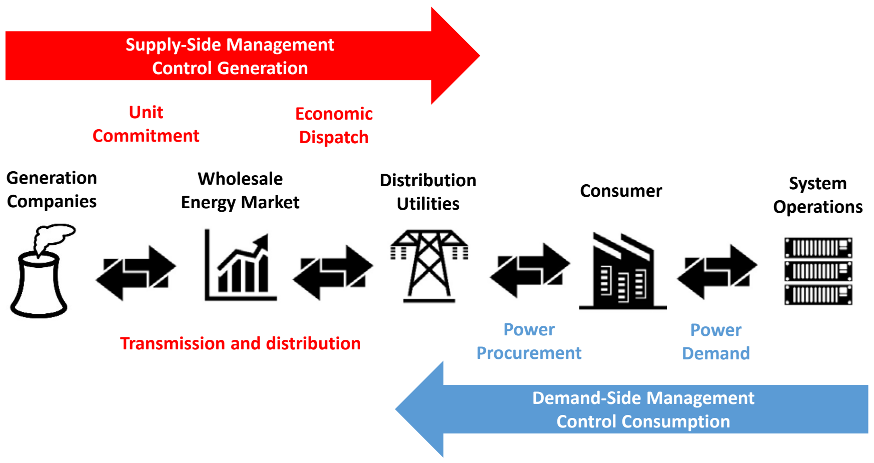


Figure 1.1: Demand-side management

rive at a dispatcher at time-varying rates (i.e. arrival rates are changing) and they are routed to one of single-server queues. In addition, we consider a fairly common situation in data centers that at all times the load from each class is very high and a large number of servers are necessary. For such a time-varying and fast-changing system, design, control and performance analysis under such heterogeneous and transient conditions are extremely difficult. This dissertation asks if time-stability can be attained and used to prevent waste of energy and improve energy-efficiency in data center operations, and Section 2 will investigate how sizing (i.e. number of active servers), assignment and routing are determined appropriately to ensure performance guarantees by enforcing time-stability for a time-varying and fast-changing system.

In Section 3, based on the general setting of data center operations considered in Section 2, this dissertation additionally considers a unique scenario that the number of application classes are much larger than the number of servers and many application classes have so little load that they would need to be hosted on just one server. This is fundamentally different from the setting considered in Section 2, where a large number of servers are necessary to host each application class. Also, it is assumed that servers are heterogeneous with different maximum processing speeds and different capacity limits of various resources, and moreover, processing speed of a server can be changed for energy efficiency. Note that Section 2 considers homogeneous servers running at the constant processing speed. For the aforementioned scenario and setting, Section 3 will address the following research questions: (i) how to stabilize aggregate workload distribution processed at each server; (ii) how to scale processing speed to achieve time-stable performance at each server; (iii) how to formulate the optimization problem to minimize energy cost while providing performance guarantees.

Next, the second part of this dissertation including *Sections 4 and 5*, addresses a demand-side energy procurement problem which is designed to determine when and how much energy to be purchased and consumed to minimize energy procurement cost. In recent years, as renewable penetration level has grown, industries have an opportunity to utilize on-site (or direct access to) renewable energy to serve their demand load to curtail expenses for procuring energy. In addition, there exists another opportunity for industries to adjust purchase and consumption of energy in response to time-varying price in the energy market. Traditionally, power consumers use electricity with a flat rate offered by utility companies or energy market for their usage. However, in recent years, it is becoming common for the energy market to allow consumers to directly purchase electricity while providing day-ahead and real-time price information. Therefore, industries get a chance to purchase electricity while being fully aware of the time-varying price, and thus, they have an opportunity to determine the amount of purchase and consumption of energy depending on electricity price and renewable generation. In this case, industries are motivated to use energy storage to mitigate fluctuations in electricity prices and renewable generations. Although it is anticipated that energy consumers are able to save a huge amount of cost by procuring energy with renewable energy and time-varying electricity prices, variability and uncertainty in power demand, renewable generation, and electricity prices make it challenging to determine optimal operational decisions.

For the aforementioned opportunities and challenges, in Section 4, this dissertation considers a consumer of electricity with inelastic demand that is met by: (i) purchasing from the energy market; (ii) on-site renewable generation (e.g. solar panels); and (iii) discharging from energy storage. Furthermore, it is assumed that there are limits and inefficiency associated with charging and discharging the energy storage. Based on the above scenario, Section 4 will focus on how to formulate and solve a sequential decision making problem tailored to real-time power procurement to minimize energy procurement cost while considering time-varying and stochastic power demand, electricity price, and renewable generation.

In Section 5, this dissertation specifically considers the operations of day-ahead and real-time energy markets and proposed a two-stage framework of power procurement such that: (i) the first stage determines a day-ahead purchase commitment for the forecated power demand and renewable generation; and (ii) the second stage determines real-time operational decisions, including purchasing electricity from real-time market, charging and discharging energy storage to adjust mismatch. In addition, this dissertation considers a unique opportunity, called demand-response, that is implemented to shift elastic and flexible power demand to use more energy at lower price and utilize more renewable energy. Specifically, from energy consumer's perspective, demand response can be implemented into the proposed two-stage power procurement so that a consumer assigns discrete time periods in day-ahead to allow demands to be shifted during real-time operations. Based on the above scenario, Section 5 will address the research question, such as how to formulate the stochastic optimization problem tailored to the two-stage power procurement with demand response to minimize power procurement cost in presence of variability and uncertainty.

At the conclusion, Section 6 will finalize this dissertation by presenting summary, con-

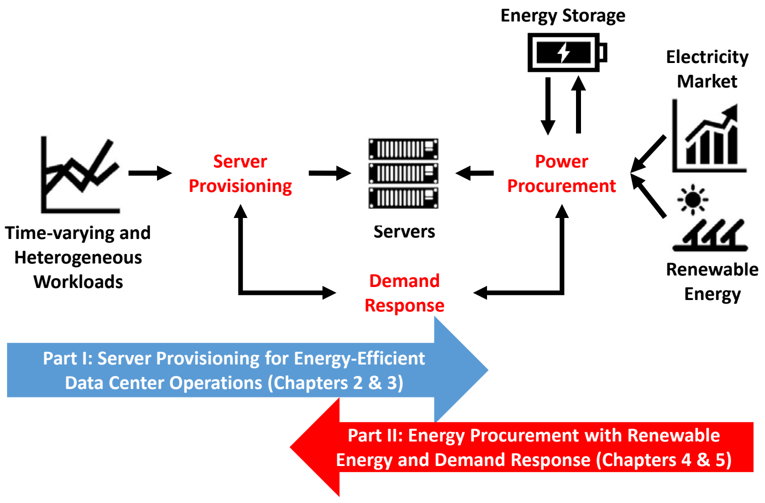

Energy Storage
Electricity Market
Server Provisioning
Servers
Power Procurement
Time-varying and Heterogeneous Workloads
Demand Response
Renewable Energy
Part I: Server Provisioning for Energy-Efficient Data Center Operations (Chapters 2 & 3)
Part II: Energy Procurement with Renewable Energy and Demand Response (Chapters 4 & 5)

Figure 1.2: Organization of chapters in terms of research topics

tribution, and future research work. Figure 1.2 shows how each of the following sections can be organized in terms of the research topic based on the big picture of this research. Note that this manuscript consists of four published/accepted journal articles, [2], [3], [4], and [5], and each of the journal articles corresponds to the Sections, 2, 3, 4, and 5. The copyright and publication information will be provided on the first page of the section. And, each section consists of individual sections including introduction, literature review, detailed description of problem and solution approach, numerical experiments, and conclusion.

2. TIME-STABLE PERFORMANCE IN PARALLEL QUEUES WITH NON- HOMOGENEOUS AND MULTI-CLASS WORKLOADS*

2.1 Introduction

Internet applications hosted by data centers are characterized by time-varying workloads with significant variations and uncertainties over multiple time scales (Menasce et al. [6]). Under such workloads it is challenging to appropriately manage resources to conserve energy consumption which is skyrocketing (see report [7]) while providing a reasonable level of performance and meeting service level agreement (SLA) (Chen et al. [8]). As explained and documented in Hamilton [9] and Koomey [10], data centers consume a phenomenal amount of power similar to what an entire city would use, albeit inefficiently; Barroso and Holzle [11] indicated that servers operate most of the time between 10 and 50 percent of their maximum utilization levels, and Vogels [12] reported that many of the large analyst firms estimate that resource utilization of 15 to 20 percent is common for operation of data centers. In addition recent studies, Abts et al. [13], Lin et al. [1], Feller et al. [14], Gandhi et al. [15], Lee and Zomaya [16] and Wang et al. [17] also mentioned low utilization of data centers and proposed approach for energy efficiency.

While there are tremendous opportunities to conserve energy consumption in data centers, due to the inherent uncertainty and variability in the loads, developing provably effective methods to manage resources in data centers has been a challenge. To address this shortcoming, a number of techniques have been proposed and most of these studies focus on developing algorithms to determine the right size of servers for non-stationary workloads. In particular, Singh et al. [18] suggested a mix-aware dynamic provisioning

*Reprinted with permission from “Time-Stable Performance in Parallel Queues with Non-Homogeneous and Multi-class Workloads" by Soongeol Kwon and Natarajan Gautam, 2016, *IEEE/ACM Transaction on Networking*, 24(3), 1322-1335, Copyright © 2016, IEEE.

technique using the k-means clustering algorithm to determine workload mix, Gandhi et al. [19] presented an approach to correctly allocate resources in data centers such that SLA violations and energy consumption are minimized and Lin et al. [1] proposed a new on-line algorithm for dynamic right sizing in data centers motivated by optimal offline solution for energy cost. Also Gandhi et al. [15] studied dynamic capacity management for multi-tier data centers, Wang et al. [17] provided an analytic framework that captures non-stationarities and stochastic variation of workloads for dynamic re-sizing in data centers and Gallego et al. [20] introduced a unified methodology that combines virtualization, speed scaling, and powering off servers to efficiently operate data centers while incorporating the inherent variability and uncertainty of workloads. It is worthwhile pointing out that most of aforementioned approaches [19], [1], [15], [17] and [20] use quasi-steady state approximations, i.e. the metrics are piecewise constant for time periods long enough for the system to reach steady-state.

Although the challenges for right-sizing in data centers for non-stationary workloads have received significant attention, the problem of achieving time-stability over time-varying workloads has not been effectively addressed. Achieving time-stability is essential for a non-homogeneous system because it enables the system to provide guaranteed quality of service. For example, one could compute the tail probability of sojourn times and probabilistically guarantee an incoming request for an appropriate SLA. Moreover, by stabilizing a non-homogeneous system, it is possible to effectively design and analyze the system and perform monitoring and control based on time-stability. In the context of data centers, time-stability has received little attention, although there have been some research studies in the queueing area. Foley et al. [21] and Barnes et al. [22] showed that the departure process from the $M_t/G_t/\infty$ queue can be made stationary. There is another body of literature which provides algorithms to determine appropriate staffing levels for call centers. Feldman et al. [23] proposed a simulation-based iterative-staffing algorithm for time-stable delay probability, and Liu and Whitt [24] suggested a formula-based algorithm to stabilize abandonment probabilities and expected delays using offered-load based approximations for a queueing model with the non-homogeneous Poisson arrival process and customer abandonment.

In our case, we have modeled data centers as a system of multiple parallel single-server queues, and considered a scenario where multiple classes of requests arrive at a dispatcher at time-varying rates that historically has daily or weekly patterns. For such a scenario, we develop an approach to simultaneously determine sizing, assignment and routing appropriately so that the resulting system performance is homogeneous over time and uncertainty is controlled despite the fact that the parameters can vary extremely quickly, not allowing the system to reach steady-state. Therefore, no matter how fast arrival rates vary, our approach can provide time-stable distribution of the number of requests in the system as well as sojourn times, and this is the crucial difference between our approach and other sizing algorithms dealing with time-varying workloads.

Objective of our study is to address needs of practitioners, such as providing performance guarantees while being prudent about energy consumption. Our suggested approach provides an analytic framework simplifying a multi-dimensional, transient and non-stationary problem by decomposing into individual simpler stationary ones based on the strategies for sizing, assignment, and routing in an integrated fashion which has seldom been implemented jointly. The main contribution of our study is providing performance guarantees and bounds which can be simply derived based on stationary analysis for time-varying and transient system while considering energy efficiency. The remainder of the paper is organized as follows: Section 2.2 describes the detailed scenario for the problem and various options for decisions and control that we would consider; Section 2.3 proposes a sequential procedure to determine assignment, sizing, and routing for the suggested scenario; Section 2.4 introduces an additional insight regarding assignment strategies; Section 2.5 describes the notion of time-stability and introduces our approach to obtain time-stability; Section 2.6 discusses details of time-stability including extension and limitations; Section 2.7 illustrates the experimental results to support our claims; and Section 2.8 presents conclusions and future research directions.

2.2 Analytical Framework

This section provides a detailed scenario for multi-class and non-homogeneous requests to servers that are considered in this paper followed by a stochastic model for the scenario. We then briefly state the asymptotic scaling where the arrival rates and number of servers are scaled. This section concludes with a description of various options for decisions and control such as assignment, sizing and routing.

2.2.1 Scenario and Problem Description

We have considered a system using a large number of servers with each server having its own queue with an infinite waiting area, and the servers and their queues are arranged in a parallel fashion with dispatcher depicted in Figure 2.1. Considering that today's data centers have hundreds or thousands of servers to process huge amount of traffic for cloud computing, an architecture with multiple servers and a single queue results in significant communication overload to update the state information of each server to dispatch requests from queue. Therefore, multiple parallel single server queue system where dispatcher routes incoming request to servers based on load balancing algorithm is indeed appropriate to design data centers. This is corroborated by recent studies, Chen et al. [25], Gandhi et al. [26] and [27] which used multiple parallel queue system to analyze data center operations. Note that we also assume that the servers are identical, however we would like to point out that the analysis can be extended to heterogeneous servers as well.

The servers process requests that belong to multiple classes. The requests that are part of a class are stochastically identical with a common non-homogeneous arrival process and

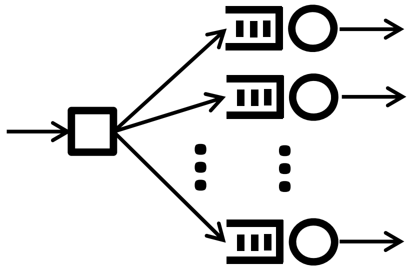


Figure 2.1: System of multiple parallel single server queues

also the amount of work they bring. It is assumed that a server can host multiple classes of requests and every class of request can be hosted on multiple servers. We have considered a scaled system where the arrival rate for every class is so large that several servers would need to be operational to respond to the requests of that class alone. However, the arrival rate for every class is time-varying both deterministically and stochastically. The variability is frequent-enough that in the general case one cannot expect the system to reach steady state before arrival rates change. For such a multi-class, transient and non-homogeneous system, our intent is to effectively manage resources to ensure time-stability while being mindful of energy costs. The following are issues that are considered explicitly for time-stability:

1. *Assignment*: Applications corresponding to each class of request can be assigned to servers such that each server hosts one or more classes and each class is hosted on multiple servers. One focus is to study the impact of host-server assignment on performance. We assume that there is no direct cost per se for the assignments as well as no costs for switching assignments.

2. *Sizing*: Each server could be dynamically powered on or off. Naturally more servers would be "on" during peak periods than during lean periods. Significant energy cost

savings can be attained by powering servers off. However, this analysis neither considers switching costs (from on to off and vice versa) nor considers reliability costs for on-off cycles. Note that some modern servers allow for “sleep" settings instead of completely turning off servers. From a mathematical standpoint, we consider them equivalent.

3. *Routing*: There is a dispatcher that is responsible for routing arriving requests to one of the queues that not only can serve the request but also has a server that is powered on. A key assumption is that the dispatcher cannot observe the real-time state of any of the servers (however the dispatcher knows whether a server is on or off, and what classes it hosts; as we will see in the model subsequently, these do not vary in real time).

2.2.2 Model and Notation

For the problem described in the previous sub-section, here we set the notation and develop a stochastic model that would form the inputs to our analysis. We consider a system of N parallel queues and each queue is served by a single server that could be dynamically powered on or off. The dispatcher is responsible for routing arriving requests to one of the queues that not only serves the request but also has a server that is powered “on.” An arriving request belongs to one of multiple classes in a discrete set $\mathcal{A}$ denoting a set of applications. The amount of work a class a (for all $a \in \mathcal{A}$) request brings is independent and identically distributed (IID) according to general distribution $H_a(\cdot)$ with mean $1/\theta_a$ and squared coefficient of variation (SCOV) C_a^2. Recall that the SCOV is the ratio of the variance to the square of the mean. For ease of exposition, as a probability distribution that can handle SCOV values greater than, equal to as well as less than one for analysis, we selected a Coxian-2 distribution for workload. Essentially, a Coxian-2 distribution is either a sum of two independent exponential distributions (with parameter $\theta_{a,1}$ and $\theta_{a,2}$)

with probability p_a, or just exponentially distributed (with parameter $\theta_{a,1}$) with probability $1 - p_a$. We chose the units of $1/\theta_a$ to be kB (kilo-Bytes) with the understanding that the analysis would not be affected in any way by choosing other units.

Requests of each class arrive to the dispatcher according to a piecewise constant non-homogeneous Poisson process. It is assumed that the environment process that drives arrival rates of the non-homogeneous Poisson process is cyclic. This is a fairly reasonable assumption as arrivals tend to have daily or weekly patterns that repeat in a cyclic fashion (Gmach et al. [28], Lin et al. [29], Liu et al. [30] and Lin et al. [1]). Using that assumption we modeled each cycle as divided into a set of phases $\mathcal{T}$ so that in each phase $\ell \in \mathcal{T}$, the arrival rate for every class $a \in \mathcal{A}$ remains a constant $\lambda_{a,\ell}$ per second. Although the intention is to convey the richness of the model (in that the analysis would work in such a general fashion), in practice one would typically choose something like the set of all disjointed 5-minute intervals (or hourly intervals) in a day (or a week) as the set of phases $\mathcal{T}$.

Let ϕ be a target operating speed of a powered on server in units of kB/s (kilo-Bytes per second). Therefore, a class a (for some $a \in \mathcal{A}$) arrival brings a random amount of workload W_a kB and is routed to an idle server that is capable of serving class a requests, then the service time (if all the processor capacity is allocated to this arrival) would be W_a/ϕ seconds with mean $E[W_a]/\phi = 1/(\theta_a \phi)$ and SCOV C_a^2. Note that the SCOV of service times is unaffected by the speed of service. At this time, we assume ϕ remains a constant. One easy way to accommodate heterogeneous servers is to have them all operate at ϕ (since the jobs are assumed to be CPU intensive).

In addition the analysis in this paper uses an asymptotic approach. In particular, we jointly scaled up the arrival rates and the number of servers so that together they approach infinity. Thus, we assume that we can write down for all $a \in \mathcal{A}$ and $\ell \in \mathcal{T}$

$$\lambda_{a,\ell} = N\alpha_{a,\ell}$$

where $\alpha_{a,\ell}$ is the normalized arrival rate, and study a sequence of systems by letting $N = 1, 2, \ldots$, which is similar in spirit to the scaling in Liu et al. [24]. However, this is not the traditional fluid or diffusion limit. All we have is that at any time there is a total of N servers (some powered on and the rest powered off) and class a requests arrive at rate $\lambda_{a,\ell} = N\alpha_{a,\ell}$, then we scale N. The next section describes how to tackle the aforementioned issues in a sequential manner.

2.3 Sequential Decision Procedure

As described in Section 2.2, our objective is to consider issues regarding assignment, sizing, and routing for the suggested scenario. These decisions are made at different time-granularities. Specifically, the assignments are made more-or-less one time, although it is assumed that at the beginning of each phase $\ell \in \mathcal{T}$ the assignments can be changed for some servers, possibly (but not necessarily) using virtual migration. We assume that sizing is done at the beginning of each phase $\ell \in \mathcal{T}$. In addition, there are real-time issues such as routing which is determined at every request arrival. The decision to be made is to determine the server to which an arriving request would be routed with conditions that (i) the server is powered on, and (ii) the server has been assigned the class of application that arrives.

2.3.1 Assignment

For each phase $\ell \in \mathcal{T}$ we consider two alternate extreme assignments for analysis:

- *all classes to all servers* (*pooled*) assignment
- *one class to one server* (*dedicated*) assignment

In Section 2.5 we will show that time-stability can be obtained by controlling non-homogeneous traffic based on assignment strategies. In fact it is possible to achieve time-stability by using *dedicated assignment*. Also we will introduce an additional insight about performance comparison between *dedicated assignment* and *pooled assignment* in Section 2.4.

2.3.2 Sizing

As described earlier, the objective is to provide time-stability while being mindful of saving energy. One of the greatest savings in energy costs results from powering servers off (or sending them to sleep states in more modern servers). Since the workload varies from phase to phase, we have evaluated the number of servers to be powered "on" in each phase, and appropriately power on or off the right number of servers. It is also assumed that there is an ample number of servers available, therefore running out of servers is out of the question. In fact, that is a reasonable assumption considering how poorly utilized some of the servers are, as the data centers are typically well over-provisioned. Recall that N is the total number of servers available. Based on the two alternate assignments described in the previous section, we have:

- *pooled assignment*: All applications assigned to all servers; let N_l be the number of servers powered "on" in phase ℓ, $\forall\ \ell \in \mathcal{T}$
- *dedicated assignment*: Only one application assigned to one server; let $N_{a,\ell}$ class a servers be powered on in phase ℓ, $\forall\ \ell \in \mathcal{T}$ and $a \in \mathcal{A}$.

We have considered a simple strategy of using enough servers so that the average load on servers that are powered on remains constant over time as well as across servers (the latter is indeed typical in load-balancing but not the former). To determine N_ℓ and $N_{a,\ell}$, we defined ρ as a desirable traffic intensity (ρ is dimensionless) for any server that is powered on during interval ℓ. While determining the number of servers to keep the energy consumption low, we aim to create enough residual capacity for unforeseen surges by restricting the utilization of each server to be ρ. In addition we control non-homogeneous traffic in a time-homogeneous fashion by implementing ρ into sizing algorithm defined as below. We will show how ρ can be used to achieve time-stability in Section 2.5. We select the number of "on" servers as follows:

- *Dedicated assignment*: Only one application assigned to one server

$$N_{a,\ell} = \left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \tag{2.1}$$

- Pooled assignment: All applications assigned to all servers

$$N_{\ell} = \left\lceil \frac{1}{\rho\phi} \sum_{a \in \mathcal{A}} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \tag{2.2}$$

for all $\ell \in \mathcal{T}$ and $a \in \mathcal{A}$. Note that under asymptotic scaling $N \to \infty$,

$$\left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \to \frac{\lambda_{a,\ell}}{\rho\phi\theta_a} \text{ and hence } \sum_{a \in \mathcal{A}} N_{a,\ell} \to N_{\ell}. \tag{2.3}$$

In such a way, the total number of servers powered on in any phase would be identical for both *pooled assignment* and *dedicated assignment*. By determining the size of powered-on servers based in Equation (2.1), each powered-on server is assigned a desirable traffic intensity ρ in either case. According to the above sizing rules, if it is necessary to power on more servers between successive phases, we randomly select candidate servers among the powered-off servers and power them on at the beginning of a time phase. Also, to power off servers we randomly select the powered-on servers and power them off at the end of a time phase. In this case if selected server is not idle, then we set state of server as “to be off" and do not assign any requests to those servers. We will wait until selected servers complete service for the remaining requests and power off when those servers become idle. Note that those requests remaining in “to be off" servers will also have the same sojourn time distribution since under a first-come-first-served (FCFS) the sojourn times are not affected by arrivals that come later.

2.3.3 Routing

In the sequential consideration, once the assignment of classes to servers and the number of servers to be powered “on” are made for each phase $\ell \in \mathcal{T}$, the next issue is to determine the routing strategy for the dispatcher. We assume that the dispatcher sends incoming requests to servers without information of real-time states of the queues in terms of number of jobs or amount of workload. However, we assume that the dispatcher knows the assignment of classes to servers as well as whether a server is powered on or off. In that light two routing policies are considered:

- Round-robin routing: The dispatcher routes job to queues with powered-on servers in a cyclical fashion. This is straightforward in the *pooled assignment* case, while round-robin is done within a class for *dedicated assignment*.

- Bernoulli routing: The dispatcher routes jobs to queues with eligible servers in a random fashion. In the pooled assignment case, in phase ℓ (for any $\ell \in \mathcal{T}$) select any of the N_ℓ servers with probability $1/N_\ell$ and route to that server. For the *dedicated case*, if the arriving job belongs to class a, then the dispatcher selects one of the $N_{a,\ell}$ servers with equal probability.

Harchol-Balter et al. [31] showed that round-robin routing results in better performance than Bernoulli routing. Clearly, other policies such as join the shortest queue and join the least workload queue would perform better, but they require real-time state information (which is assumed inappropriate for large-scale data centers setting). It is worthwhile noting that the round-robin policy works better because the dispatcher selects the queue which was the least recently selected (among candidate queues), and that queue naturally is also the one with the smallest expected number of jobs and smallest expected workload. We will continue to use both round-robin and Bernoulli policies for load balancing, although it is fairly clear that round-robin results in better performance. One of the reasons for continuing to use the Bernoulli policy is the convenience in analytic models, especially to obtain insights.

2.4 Additional Insight: Dedicated Is Better Than Pooling

This section describes an additional insight regarding assignment strategies based on our analytical framework. In general, because of the benefits of pooling resources mentioned in the literature, the intuition is that performance would be better when we assign as many applications as possible to a server. However, based on two alternate assignments defined in Section 2.3.1 we will show that *dedicated assignment* would be better. Although we have the same number of “on" servers for both *dedicated assignment* and *pooled assignment* in each time period, the queue lengths (or the sojourn times) of overall system would be higher when we use *pooled assignment* than use *dedicated assignment*. Consider a single server that is always on with time-homogeneous arrivals, i.e. $\lambda_{a,\ell}$ does not vary with ℓ for all $a \in \mathcal{A}$ and i.e. $\lambda_{a,\ell} = \lambda_a \; \forall \ell \in \mathcal{T}$. This may appear strange given that we started the article with non-homogeneous arrivals, however subsequently we will show that this setting is in fact what is realized in the main problem in Section 2.5.2.1. It is also assumed that the servers are identical. Consider two cases for the assignments mentioned above, *dedicated assignment* or *pooled assignment*. Recall that in either case, each powered-on server faces the same traffic intensity of ρ when we determine the number of servers to be powered on according to Equation (2.1) for *dedicated assignment* and

Equation (2.2) for pooled assignment.

Theorem 1. *If $\mathcal{C}_a^2$ is identical for all $a \in \mathcal{A}$ and Bernoulli routing is used, then the mean sojourn time (and total number in the system) of pooled assignment is higher than* dedicated assignment *in a steady state.*

Proof. An arriving class a job in steady state brings a workload W_a and service time $S_a = (W_a/\phi)$ for any $a \in \mathcal{A}$. Since we assume that $\mathcal{C}_a^2$ is identical for all $a \in \mathcal{A}$, we can use $\mathcal{C}^2$ as SCOV of the amount of work for all $a \in \mathcal{A}$. Note that each server has the same traffic intensity ρ. Based on our sizing strategies, we can calculate the total number of requests in the whole system for each assignment strategy by using the Pollaczek-Khintchine formula (P-K formula) (Gautam [32]) as follows:

- for the *dedicated assignment*, the number of servers for each application a is

$$N_a = \frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}$$

and arrival rate Λ_a for each server of application a is

$$\Lambda_a = \frac{\lambda_a}{\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}} = \rho\phi\theta_a. \tag{2.4}$$

Thus, the expected number of requests in each queue (server) of application a in steady state is

$$L = \rho + \frac{\Lambda_a^2}{2}\frac{(Var[S_a] + (E[S_a])^2)}{(1-\rho)}. \tag{2.5}$$

Then, we have the total number of requests in the whole system for *dedicated assignment* case given by

$$L_{dedicated} = \sum_{a\in\mathcal{A}} \frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}\left(\rho + \frac{\Lambda_a^2}{2}\frac{(Var[S_a] + (E[S_a])^2)}{(1-\rho)}\right)$$

$$=\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}+\frac{\rho}{2}\frac{(1+\mathcal{C}^2)}{(1-\rho)}\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}. \tag{2.6}$$

by substituting for (2.4), and realizing that $\mathcal{C}^2=\mathcal{C}_a^2=\frac{Var[W_a]}{\frac{1}{\theta_\alpha^2}}$.

- for the *pooled assignment*, the total number of servers is

$$N=\sum_{a\in\mathcal{A}}\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}.$$

In this case, we need to use the Pollaczek-Khintchine formula for multi-class queue, thus the number of requests in each queue (server) is

$$L=\rho+\frac{1}{2}\frac{\Lambda^2E[S^2]}{(1-\rho)} \tag{2.7}$$

where

$$\Lambda=\frac{\sum_{a\in\mathcal{A}}\lambda_a}{\sum_{a\in\mathcal{A}}\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}} \tag{2.8}$$

and

$$E[S^2]=\frac{\sum_{a\in\mathcal{A}}\lambda_aE[S_a^2]}{\sum_{a\in\mathcal{A}}\lambda_a}=\frac{\sum_{a\in\mathcal{A}}\lambda_a\left(Var[S_a]+\frac{1}{\phi^2\theta_a^2}\right)}{\sum_{a\in\mathcal{A}}\lambda_a}.$$

Then, we can calculate the total number of requests in the whole system for *pooled assignment* case as

$$\begin{aligned}L_{pooled}=&\left(\rho+\frac{1}{2}\frac{\Lambda^2E[S^2]}{(1-\rho)}\right)\sum_{a\in\mathcal{A}}\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}\\=&\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}+\frac{\rho}{2}\frac{(1+C^2)}{(1-\rho)}\left(\frac{\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi^2\theta_a^2}}{\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}}\right)\sum_{a\in\mathcal{A}}\lambda_a.\end{aligned} \tag{2.9}$$

by substituting for (2.8), and realizing that $\mathcal{C}^2=\mathcal{C}_a^2=\frac{Var[W_a]}{\frac{1}{\theta_\alpha^2}}$.

Based on Equation (2.6) and (2.9), $L_{dedicated} \leq L_{pooled}$ if

$$\left(\sum_{a\in\mathcal{A}} \frac{\lambda_a}{\phi\theta_a}\right)^2 \leq \left(\sum_{a\in\mathcal{A}} \frac{\lambda_a}{(\phi\theta_a)^2}\right)\sum_{a\in\mathcal{A}} \lambda_a. \tag{2.10}$$

We can represent left-hand side of Equation (2.10) as

$$\left(\sum_{a\in\mathcal{A}} \frac{\lambda_a}{\phi\theta_a}\right)^2 = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \lambda_i\lambda_j \frac{1}{\phi\theta_i}\frac{1}{\phi\theta_j}. \tag{2.11}$$

Likewise the right-hand side of Equation (2.10) as

$$\left(\sum_{a\in\mathcal{A}} \frac{\lambda_a}{(\phi\theta_a)^2}\right)\sum_{a\in\mathcal{A}} \lambda_a = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \lambda_i\lambda_j \frac{1}{(\phi\theta_i)^2}. \tag{2.12}$$

Now, using the fact that

$$\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \lambda_i\lambda_j \left(\frac{1}{\phi\theta_i} - \frac{1}{\phi\theta_j}\right)^2 \geq 0$$

we can show Equation (2.10) is true as since

$$2\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \lambda_i\lambda_j \frac{1}{\phi\theta_i}\frac{1}{\phi\theta_j} \leq 2\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \lambda_i\lambda_j \frac{1}{(\phi\theta_i)^2}. \tag{2.13}$$

Finally, by using Little's Law (Gautam [32]), the sojourn times of *dedicated assignment* case is better than the pooled case.

□

From Theorem 1, we can conclude that the *dedicated assignment* appears to be more effective than the pooled assignment for the total number of requests as well as the mean sojourn time.

Remark 1. *Based on Theorem 1, we make the following comments: (i) even if we assign a subset of applications to each server (not pooling* all *classes), it would still be worse than having a dedicated server for each application; (ii) we conjecture that if the $\mathcal{C}^2$ were different for the applications, the result would still remain (we will verify this conjecture in the numerical studies in Section 2.7.2.1); (iii) we require the arrival rates to be homogeneous across time for each application, and it turns out, as shown in the next section, that this requirement would be satisfied as we will create time-stationary queues as described in Section 2.5.*

2.5 Time-Stability

As we previously mentioned, the main objective is to suggest an approach which provides performance bounds and guarantees based on time-stability for the non-homogeneous and transient system. In this section, we describe our notion of time-stability and the approach to obtain time-stability based on the suggested analytical framework.

2.5.1 Notion of Time-Stability

As we described in Section 2.2.1, we consider a system of N parallel queues with a single dispatcher. Each queue has a single server that may be on or off at time t. For all $n \in \{1, \ldots, N\}$, at time t let $X_n(t)$ be the number of jobs in queue n and $O_n(t)$ be the status of the server (with $O_n(t) = 1$ denoting 'on' and $O_n(t) = 0$ denoting 'off'). Let $\lambda_a(t)$ be the arrival rate of class a jobs at the dispatcher at time t. We assume that we can divide time into arbitrarily small intervals (of appropriate time units) such that $\lambda_a(t) = \lambda_a([t])$, i.e. the arrival rate stays constant in the interval $[t, t+1)$ for all t (the notation $[t]$ denotes the integer part of t). Let $W_a(t)$ be the sojourn times experienced by a class a job that arrives into the dispatcher at time t.

As described in Section 2.3, we seek to obtain a policy for deciding (i) $\sum_n O_n([t])$, the total number of servers that would be 'on' in interval $[t, t+1)$ for all t (sizing); (ii) the allocation scheme of applications to servers (assignment); (iii) the policy for routing requests from despatcher to server queues (routing). Based on this, our key objective is to ensure time-stability of both queue lengths for powered-on servers as well as sojourn times for a class of application. In other words, for all $t \in [0, \infty)$, $s \in [0, \infty)$, and $i \in \{0, 1, 2, \ldots\}$,

$$\begin{aligned} P\{X_n(t) = i | O_n(t) = 1\} &= \pi_a(i) \\ P\{W_a(t) \leq s\} &= \Psi_a(s) \end{aligned}$$

where $\pi_a(i)$ and $\Psi_a(s)$ are computable constants that are not dependent on t. That is the sense of time-stability we aim to achieve. In the following sections we will show that we can achieve the aforementioned time-stability via (i) dedicated assignment of applications to servers, (ii) sizing rule for dedicated assignment in Equation (2.1), (iii) either Bernoulli routing or round-robin routing, (iv) dummy requests, and (v) adjusting the remaining work for the head-of-line job. In fact, if we use round-robin (or Bernoulli) routing, then $\pi_a(i)$ is the stationary probability that a $D/G/1$ (or $M/G/1$) queue has i jobs in the system and $\Psi_a(s)$ is the CDF of sojourn times of an arbitrary job of the corresponding queue.

2.5.2 Approach to Obtain Time-Stability

In the previous section, we introduce the notion of time-stability considered in this study. Based on our notion of time stability, in this section we suggest an approach to obtain time-stability which consists of two main procedures. First we decompose non-homogeneous, multiple, parallel single-server queue system into individual simple time-homogeneous queues, and then we add “dummies" to ensure the steady state of each class a server while powering servers on and off. We describe details of the procedure in the following subsections.

2.5.2.1 Non-Homogeneous Traffic Control

As described in Section 2.3.3, we consider two routing strategies, round-robin and Bernoulli, and the following theorem characterizes the arrival process for both round-robin and Bernoulli routing based on pooled assignment.

Theorem 2. *For the pooled assignment strategy, each server that is powered on during phase ℓ gets arrivals deterministically (exponentially) at rate*

$$\frac{\sum_{a\in\mathcal{A}}\lambda_{a,\ell}}{\frac{1}{\rho}\sum_{a\in\mathcal{A}}\frac{\lambda_{a,\ell}}{\phi\theta_a}}$$

for all $\ell \in \mathcal{T}$ and $a \in \mathcal{A}$, under Round-robin (Bernoulli) routing, as $N \to \infty$. And the expected workload (in KB/s) that each request brings (by conditioning on the class) is

$$\sum_{a\in\mathcal{A}}\left(\frac{\lambda_{a,\ell}}{\sum_{b\in\mathcal{A}}\lambda_{b,\ell}}\right)\frac{1}{\theta_a}.$$

Proof. The net arrival rate to the dispatcher in phase ℓ is $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Thus the time between request arrival at the dispatcher in phase ℓ is exponentially distributed with parameter $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Then, due to round-robin routing, each server that is powered on in phase ℓ observes inter-arrival time which is the sum of N_ℓ IID exponentially distributed times with parameter $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Thus, the inter-arrival times to a powered on server is according to an Erlang distribution with mean $N_\ell/\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$ and variance $N_\ell/\left(\sum_{a\in\mathcal{A}}\lambda_{a,\ell}\right)^2$. In the limit as $N \to \infty$, using the expression for N_ℓ in Equation (2.2), the mean term converges to

$$\frac{\sum_{a\in\mathcal{A}}\frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}}{\sum_{a\in\mathcal{A}}\lambda_{a,\ell}}$$

while the variance term converges to zero. Thus the time between arrivals become deterministic in the limit as $N \to \infty$ and each server that is on during phase ℓ gets arrivals deterministically at rate

$$\frac{\sum_{a\in\mathcal{A}} \lambda_{a,\ell}}{\sum_{a\in\mathcal{A}} \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}}.$$

Now, we can compute the expected workload (in KB) that each request brings (by conditioning on the class) as

$$\sum_{a\in\mathcal{A}} \left(\frac{\lambda_{a,\ell}}{\sum_{b\in\mathcal{A}} \lambda_{b,\ell}}\right)\frac{1}{\theta_a}$$

and thus by multiplying by the expected arrival rate the expected workload arrival is $\rho\phi$ KB/s. Now, if round-robin routing is replaced with Bernoulli routing, then the only change in the theorem would be to replace both occurrences of the word "deterministically" with "exponentially distributed." This is because after a Bernoulli split, the resulting processes are Poisson processes with identical rates as the deterministic arrivals (however, note that we do not require the $N \to \infty$ for this case). Otherwise, everything else remains the same. $\square$

Theorem 2 concludes that for either routing case, round-robin or Bernoulli, pooling all applications in one server (*pooled assignment*) would result in a non-homogeneous system without time-stability because each server has time-varying arrival rates for $\ell \in \mathcal{T}$ under both routing strategies. However, the next theorem shows that time-stability can possibly be obtained with *dedicated assignment* strategy.

Theorem 3. *For the* dedicated assignment *strategy, each server of application a that is powered on at any time gets arrivals deterministically (exponentially) at rate $\rho\phi\theta_a$ under round-robin (Bernoulli) routing strategies and each arrival brings work according to CDF $H_a(\cdot)$ as $N \to \infty$. Also, each powered "on" class a server faces an expected workload $\rho\theta_a$ KB/s at all times.*

Proof. During phase ℓ, requests of class a arrive according to a Poisson process with mean rate $\lambda_{a,\ell}$. First consider round-robin routing. For the *dedicated assignment*, each server hosting class a and is powered on in phase ℓ observes inter-arrival time which is the sum of $N_{a,\ell}$ IID exponentially distributed times with parameter $\lambda_{a,\ell}$ since for each class a the dispatcher performs a round-robin of the servers within the class. Thus the inter-arrival times to a class-a powered-on server is according to an Erlang distribution with mean $N_{a,\ell}/\lambda_{a,\ell}$ and variance $N_{a,\ell}/\lambda_{a,\ell}^2$. In the limit as $N \to \infty$, $N_{a,\ell} \to \infty$ the mean term converges to $1/(\rho\theta_a)$ by substituting for $N_{a,\ell}$ from Equation (2.1), while the variance converges to zero. Thus, the time between arrivals becomes deterministic in the limit as $N \to \infty$ and each class-a server that is on during phase ℓ gets arrivals deterministically at rate $\rho\phi\theta_a$. The expected workload (in KB) that each request brings (by conditioning on the class) to a class-a server is $1/\theta_a$, and thus the expected workload arrival rate is $\rho\phi$ KB/s. With Bernoulli routing instead of round-robin, the resulting split processes going into each powered-on server are Poisson process, and each server gets arrivals exponentially at rate $\rho\phi\theta_a$. $\square$

Based on Theorem 3, when only one application is assigned to a server (*dedicated assignment*), each server of application a that is powered on at any time phase gets homogeneous arrival process and also each arrival brings work according to CDF $H_a(\cdot)$. These will form the building blocks for creating time-stable queue length processes in powered-on servers. The next section describes how to obtain time-stability with powering on and off schemes.

2.5.2.2 Time-Stability by Adding Dummy Requests

The previous section showed that each individual server of application a gets time-homogeneous arrival process and workload distribution with *dedicated assignment*. However, our concern is whether powering servers on would cause problems for achieving time-stable performance. It is intuitive to think that stationarity would be affected during times when servers are powered on and off, i.e. between phases. In other words, homogeneous arrival process and workload distribution are not sufficient to achieve time-stable performance since the initial conditions in an interval are different when powering servers on. This is especially the case when time intervals are short and steady-state is not reached, then the initial conditions become significant.

To address this problem of initial conditions, we introduce dummy requests to adjust the initial number of requests in a queue of a newly powered-on servers. In order to ensure the steady-state of each class a server that is powered on afresh at the beginning of an interval, we generated dummy requests sampled from the stationary distribution of a $D/G/1$ queue for round-robin routing or an $M/G/1$ queue for Bernoulli routing. For $M/G/1$ queue case under a FCFS (note that the formulas have to be tweaked appropriately for other polices such as versions of processor sharing), we can use the probability generating function of the stationary queue length distribution (Gautam [32]) for class a server,

$$\pi_a(i) = \frac{(1-\rho)(1-i)\tilde{G}(\lambda_a - \lambda_a i)}{\tilde{G}(\lambda_a - \lambda_a i) - i} \tag{2.14}$$

where $\lambda_a = \rho\phi\theta_a$ and $\tilde{G}(s) = \int_0^\infty e^{-sx} dG(x)$, the Laplace-Stieltjes transform (LST) of the service time distribution $G(\cdot)$. Note that service time would be X/ϕ seconds with a random amount of workload X kB and processing speed ϕ kB/s, and we have $G(y) = P[Y \leq y] = P[\frac{X}{\phi} \leq y] = P[X \leq \phi y] = H(\phi y)$ where $H(\cdot)$ is cumulative distribution function for workload. From Equation (2.14), we derive moment-generating function of the stationary queue length distribution for class a server defined by workload distribution $H(\cdot)$,

$$\pi_a(i) = \frac{(1-\rho)(1-i)\tilde{H}(\theta_a\rho - \theta_a\rho i)}{\tilde{H}(\theta_a\rho - \theta_a\rho i) - i} \tag{2.15}$$

where $\tilde{H}(s) = \int_0^\infty e^{-sx} dH(x)$, the LST of workload distribution. Now, we can initially populate the number of requests in queue by sampling from the distribution in Equation (2.15). For $D/G/1$ queue case, we do not have an exact formula for the stationary queue length distribution, but instead, we can simulate a single $D/G/1$ queue offline and obtain the distribution numerically. Note that such a simulation is extremely inexpensive and straightforward.

In addition since the objective is to create a time-homogeneous system, at any given time the system characteristics must be stationary. In particular, at times when a server is powered on, not only the number of dummy requests be according to the stationary distribution but the amount of work completed for the request at the head of the line (if any) must also be stationary. Using results from renewal theory, we know that the remaining work for the job at the head of the line is according to its stationary excess distribution (Gautam [32]). Stationary excess distribution $F_e(t)$ associated with CDF $F(t)$ in terms of the mean $\tau = -\tilde{F}'(0)$ such that

$$F_e(t) = \frac{1}{\tau}\int_0^t [1 - F(u)]du. \tag{2.16}$$

We now illustrate the stationary excess distribution and its computation for the Coxian-2 random variable that will be used in Section 2.7. It results in the following theorem for the stationary excess distribution of Coxian-2 distribution.

Theorem 4. *The stationary excess distribution of Coxian-2 distribution is also Coxian-2 distribution albeit with different parameters.*

Idea of proof: By using the LST we can easily show that CDF of Coxian-2 distribution can be represented as a linear combination of two CDFs of exponential distribution. Moreover, stationary excess distribution of Coxian-2 distribution can be defined as a linear combination of two CDFs of exponential distribution which means that stationary excess distribution is also Coxian-2 distribution.

Next, we introduce dummy traffic to adjust the arrival rate to each powered-on server under *dedicated assignment*. Recall that we determined the number of class a servers powered on in phase ℓ, $N_{a,\ell}$ using Equation (2.1) as described in Section 2.3.2,

$$N_{a,\ell} = \left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil$$

to ensure that each powered-on server gets a desirable traffic intensity ρ in any time phase for both *pooled assignment* and *dedicated assignment*. In case N is finite, we need to adjust $\lambda_{a,\ell}$ by adding dummy traffic for class a so that the net arrival rate in phase ℓ is $N_{a,\ell}\rho\phi\theta_a$. Adding dummy traffic can ensure a homogeneous arrival process for each class with *dedicated assignment*. In this case the amount of additional dummy traffic would be

$$\left(\left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil - \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right) \rho\phi\theta_a \leq 1 \times \rho\phi\theta_a.$$

Note that the maximum amount of additional traffic into each powered on class a server would be less than $\rho\phi\theta_a / \left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil$ and if total number of N is large (which is fairly common in data centers), then the amount of additional traffic would be insignificant. We will compare actual arrivals with adjusted arrivals in Section 2.7.2.6). Now, based on the results from previous sections and strategy for dummies, we can arrive at the following theorem which shows that time-stable performance can be achieved by the suggested approach.

Theorem 5. *The number of requests in any powered on server processing class a requests at any time in an interval would be stationary according to the stationary distribution of an $D/G/1$ or $M/G/1$ queue depending on round-robin or Bernoulli routing, thereby resulting in a time-stable performance.*

Proof. We need to show that initial conditions of class a servers, especially those powered on afresh, in an every time interval ℓ are according to stationary queues with dummy requests. Considering an arbitrary class a and an arbitrary interval of time ℓ. For convenience, we let the beginning of this interval be time $t = 0$ and select an arbitrary class a

server that is powered on afresh at time $t = 0$, i.e. powered on in interval ℓ but powered-off in the previous interval. Clearly, by adding “dummy" jobs as described above, the number of jobs in the server as well as the amount workload at time $t = 0$ are according to those of a stationary $D/G/1$ ($M/G/1$) queue under round-robin (Bernoulli) routing. Also, since the arrival process and the amount of work an arrival brings remain unchanged throughout the time the server is on (even if it is over multiple intervals) with *dedicated assignment* as described in Section 2.5.2.1, the workload process is Markovian for Bernoulli routing and delayed semi-Markovian for Round-robin routing, due to stationarity and ergodicity properties which would result in time-stable performance. Thus the number in the system or the workload observed at any time t during the server's on-time sojourn would remain stationary regardless of powering servers on and off (note that this includes time intervals beyond ℓ). $\square$

In Section 2.6.1, we will show that time-stability of the number of requests in system could be extended to time-stability of sojourn times in a straightforward fashion.

2.5.2.3 Step-by-Step Procedure

The following is a procedure to achieve time-stability:

Step 1. Off-line Phase

Step 1.1. By using dedicated assignment, determine the number of servers for each class a and for each time period $N_{a,\ell}$ by using Equation (2.1).

Step 1.2. Obtain the queue length distribution $\pi_a(i)$ for $M/G/1$ queue analytically or $D/G/1$ queue via simulation to sample from for initial number of dummy requests for initial condition.

Step 1.3. Add dummy traffic so that arrival rate for class a for time ℓ is $N_{a,\ell}\rho\phi\theta_a$.

Step 2. On-line Phase

Step 2.1. At the beginning (or end) of each time period, compute the difference in the number of servers between consecutive time periods based on the number of servers computed in [Step 1.1].

Step 2.2. If $N_{a,\ell} < N_{a,\ell+1}$, then

Step 2.2.1. Select $N_{a,\ell+1} - N_{a,\ell}$ servers to be powered on randomly among the “off" servers.

Step 2.2.2. Add dummy requests to each newly powered on server by sampling the number of dummy requests from the queue length distribution $\pi_a(i)$ derived in [Step 1.3].

Step 2.2.3. Adjust the amount of remaining work of the very first dummy request of each newly powered on server based on the stationary excess distribution.

Step 2.3. If $N_{a,\ell} > N_{a,\ell+1}$, then

Step 2.3.1. Select $N_{a,\ell} - N_{a,\ell+1}$ servers randomly among the “on" servers.

Step 2.3.2. If selected server is idle, then just power off selected server.

Step 2.3.3. Otherwise, set status of server as “to be off" and do not route incoming requests to that server, then power off when server completes service of the last remaining request.

2.5.3 Performance Bounds and Guarantees

As we described in the previous Section 2.5.2.2, dummies are used to (i) adjust the initial number of requests in a queue of newly powered-on servers and (ii) adjust the class dependent arrival rate to each powered-on server. Although adding dummies is crucial to obtain time-stability, it also degrades performance and thus practitioners may have concerns about this issue. In this situation if the practitioners choose not to add dummy requests, then time-stability predictions would be an upper bound on actual performance. In other words, the mean queue length would be time-varying without using dummies, but strictly bounded by time-stable performance which can be obtained by adding dummies. From both theoretical and practical points of view, such performance bounds are extremely useful since bounds are provable and derived by stationary analysis of queueing model (e.g. P-K formula) for non-homogeneous and transient system. Note that it is difficult to yield time-stable performance or obtain the provable bounds on performance of time-varying system especially when steady-state cannot be reached.

Remark 2. *Time-stable and provable performance bounds cannot be obtained by simply assuming stationarity without adding dummies. To explain this, let S1 be an original system which determines $N_{a,\ell}$ by using* (2.1)*, but does not add both types of dummies. In fact in S1, arrival rates of class a requests into each powered-on server j, $\lambda_{a,j,\ell}$ which can be defined as $\lambda_{a,j,\ell} = \frac{\lambda_{a,\ell}}{N_{a,\ell}}$ where $\sum_{j \in \mathcal{N}} \lambda_{a,j,\ell} = \lambda_{a,\ell}$, would be time-varying across time intervals. In other words, $\lambda_{a,j,\ell} \neq \rho\phi\theta_a$ for all $\ell \in \mathcal{T}$, since $\lambda_{a,j,\ell} = \lambda_{a,\ell}/N_{a,\ell}$ but $N_{a,\ell} = \left\lceil \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \neq \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}$. In this case, we can use standard PK formula for $M/G/1$ queue model by assuming stationarity to compute the mean queue length. Mean queue length of class a server in time interval ℓ, $L_{a,\ell}$ can be computed as,*

$$L_{a,\ell} = \frac{\lambda_{a,j,\ell}}{\phi\theta_a} + \frac{1}{2}\left(\frac{\lambda_{a,j,\ell}}{\phi\theta_a}\right)^2 \left(\frac{1 + C_a^2}{1 - \frac{\lambda_{a,j,\ell}}{\phi\theta_a}}\right).$$

Based on above equation, our claim is that we cannot obtain time-stable upper bound on the mean queue length and thus $L_{a,1} \neq L_{a,2} \neq \cdots \neq L_{a,T} \neq \bar{L}_a$ where $\bar{L}_a$ is an upper bound on the mean queue length obtained by our suggested approach, since arrival rates $\lambda_{a,j,\ell}$ are different across time intervals without adding dummy traffic as described in

Section 2.5.2.2. In this case, $L_a^{\max} = max\{L_{a,1}, L_{a,2}, \ldots, L_{a,T}\}$ would be an upper bound, however $\bar{L}_a \leq L_a^{\max}$. In other words, assuming steady-state itself is not enough to obtain time-stable and provable upper bound $\bar{L}_a$, and our suggested approach provides essential conditions to obtain an upper bound $\bar{L}_a$ which is provable and can be applied to transient system without assuming steady state assumption (which is impossible for real system).

Although time-stable performance bounds provided by our suggested approach are useful, it is important to analyze the gap between time-varying actual performance with performance bounds. First of all, it is reasonable to expect that the gap between bounds and actual performance would be bigger when arrival rates are increasing more drastically since the actual performance is highly dependent with the increment of the number of servers. In other words, since every newly powered-on server starts serving incoming requests with empty queue, the mean queue length would be decreasing when the number of server is increasing. In addition the gap between bounds and actual performance is highly dependent with variance of workloads and also system utilization based on analysis of queueing model (e.g. P-K formula (2.5) and (2.7) used in Section 2.4). Considering that providing performance bounds and guarantees based on time-stability opposed to time-varying and transient system has not been addressed before our study, we believe that our study has both theoretical and practical contributions. In Section 2.7.2.3 we will introduce simulation results to compare the time-varying actual performance with time-stable bounds and analyze the gap for the different SCOV of workload distribution and the desired traffic intensity ρ.

2.6 Discussion on Time-Stability

In this section we discuss details of time-stability obtained by our suggested approach in terms of its extension and limitations.

2.6.1 Extension to Sojourn Times

As introduced in Section 2.5.1, our suggested approach stabilizes queue length distribution. Then we consider sojourn times as users of data center need to get Quality of Service (QoS) guarantees in terms of sojourn times. In fact, when the distribution of the queue lengths is stabilized, performance analysis of system is very straightforward in terms of sojourn times. Since the distribution of number of jobs in each powered-on class a server is time-stable, the amount of work brought by jobs is time-invariant, and service speed is constant for each server, the sojourn time distribution would also be time-stable. Therefore based on queue length distribution, we can derive time-stable sojourn time distribution which enables us to provide probabilistic guarantees of the response times for incoming requests. In other words, under a FCFS regime, distribution of sojourn time of class a at time t, $W_a(t)$, can be defined as $\Psi_a(w) = P[W_a(t) \leq w]$ (which is not dependent on time t). Providing probabilistic guarantees on sojourn times (as well as queue lengths) based on time-stability has significant benefits since for transient system with time-varying and non-stationary load, it is extremely difficult to provide guaranteed SLA without assumption for steady-state. For example, our approach is able to provide a bound τ on average sojourn time such that $E[W_a(t)] \leq \tau$, or tail probability of response time for bound τ such that $P[W_a(t) \leq \tau] \geq 1 - \epsilon$ which would remain unchanged across time. Without assuming that system reaches steady-state in each time interval, the only way to provide guarantees is running a large number of servers which causes a much higher energy consumption. In this context, achieving time-stability and providing performance bounds and guarantees based on dummies is the key benefit of our suggested approach.

As described in Section 2.5.2.1, under suggested framework we can decompose our system into simpler homogeneous queues, $D/G/1$ queue for round-robin routing and $M/G/1$ queue for Bernoulli routing. In this case, for the $M/G/1$ queue we have the

LST of the sojourn time distribution $\tilde{\Psi}_a(s)$ for class a request as (Gautam [32]),

$$\tilde{\Psi}_a(s) = \frac{(1-\rho)s\tilde{G}(s)}{s-\lambda_a(1-\tilde{G}(s))} \tag{2.17}$$

where $\lambda_a = \rho\phi\theta_a$ and $\tilde{G}(s) = \int_0^\infty e^{-sx} dG(x)$, the LST of service time distribution. Although we do not have a specific formula for the sojourn time distribution of $D/G/1$ queue case, we can derive the sojourn time distribution from simulation with $D/G/1$ queue setting. Note that it is not easy to derive continuous sojourn time distribution, thus we can derive it based on queue length distribution, $\pi_a(i)$ itself. Indeed, we can apply derived sojourn time distribution to each server under round-robin routing in time-stable manner. Based on our analysis, we can also obtain time-stable performance bound on sojourn times as well as queue lengths. In Section 2.7.2.2 we will introduce simulation results which show that the mean and standard deviation of sojourn times are stabilized with our suggested approach.

2.6.2 Time Interval Length

In order to model time-varying arrivals of requests, we assume that requests arrive according to a piecewise constant non-homogeneous Poisson process where arrival rates of requests of application classes stay constant in each time interval. In this situation, we need to carefully think about the effect of time interval length in terms of whether our suggested approach would be robust to time interval length. In other words, we need to check whether distribution of queue lengths or sojourn times would be time-stable with small time interval length when arrival rates change very fast. In this context, we would like to note that our suggested approach would perform well when time interval length too small to reach steady-state within each time interval and has a sense of the robustness to time interval length. Note that for implementation it is reasonable to assume that the service times and inter arrival times of requests are much smaller than time interval length since the case where the service times are longer than time interval length is unlikely in practice for data centers. In Section 2.7.2.4, we will compare the simulation results with different time interval lengths to show robustness of our approach.

2.6.3 System Size (Total Number of Servers)

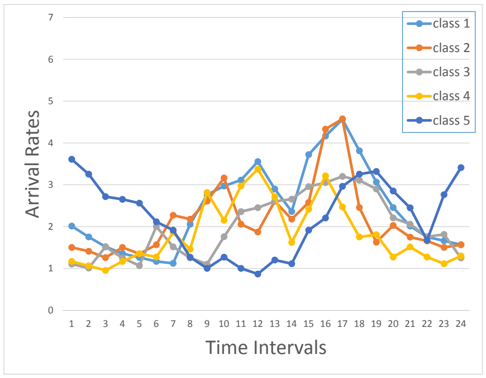
In this study, we consider a fairly common situation in data centers where the traffic of requests is very high and a large number of servers are necessary, and thus we use the asymptotic scaling where both the arrival rates and number of servers are scaled with size N. In fact, our suggested approach itself has limitation with small size N, since for round-robin routing arrival rate into each powered-on server would not be time-homogeneous if size N is small as shown in proof of Theorem 3. Therefore queue length distribution is also non-homogeneous with small size N. Note that for the case of using Bernoulli routing, arrival rate into each powered-on server would be time-stable regardless of size N. In Section 2.7.2.5 we will compare simulation results with small size N for both round-robin and Bernoulli routing cases to check the limitation of our approach.

2.7 Numerical Evaluation

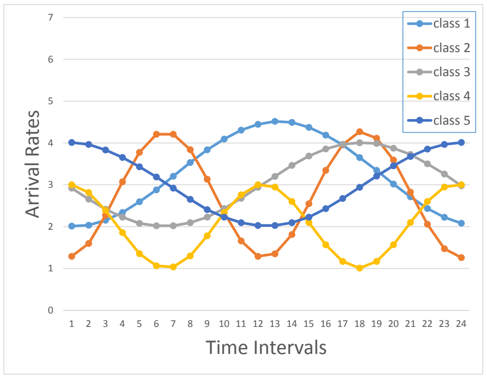
In this section we describe the simulation settings and then analyze the results of simulation experiments to evaluate our approach. We verify our additional insight for assignments and show that our suggested approach provides time-stability in both queue length distributions and sojourn time distributions based on simulation results. Also we analyze performance bounds and effects of both time interval length and system size N to time-stability.

2.7.1 Simulation Experiments

We developed a simulation on a Java platform with $N = 1000$ possible servers using two sets of input data for the arrival rates and two sets for the workloads. The input



(a) Pattern A



(b) Pattern B

Figure 2.2: Normalized arrival rate $\alpha_{a,\ell}$ for the 5 classes for 1 cycle of 24 equal-length phases

data will be discussed in the latter part of this section. We used 5 classes of requests, hence $\mathcal{A} = \{1, 2, 3, 4, 5\}$ and 24 equally spaced time intervals (time interval length is 60 minutes), hence $\mathcal{T} = \{1, 2, ..., 24\}$. We assume that the request inter-arrival times are much shorter than the time intervals and it is crucial to note that although for the analysis we do not require the intervals be equally spaced, it is that way to avoid a cumbersome presentation.

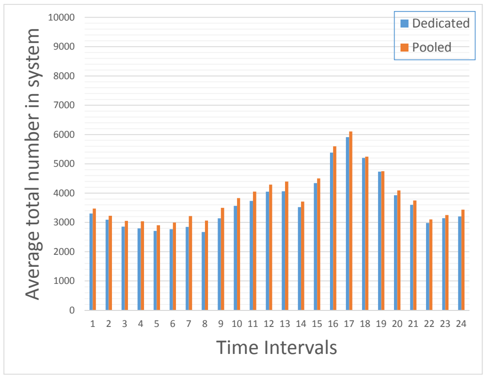
Next we describe the 5 classes' workload characteristics. Note that we used Coxian-2 distribution for the workload described in Section 2.2.2. We define two sets of amount of work data, both having the same mean amount of work $1/\theta_a$ for all $a \in \mathcal{A}$ as 20, 15.2381, 25, 17.619, 21 seconds. These two sets have different conditions for SCOV, one has the same value of SCOV, 0.7, for all $a \in \mathcal{A}$ and the other has SCOV of the amount of work for classes 1, 2, 3, 4 and 5 as 1, 0.8887, 2.2, 1.335, and 0.9501 respectively. Since we used different SCOV for amount of work (but the mean amount of work is the same), we needed to define the parameters of the Coxian-2 distribution, θ_1, θ_2 and p, differently for each set of amount of work. For the same SCOV case, we used probability p as 0.9375, 0.6099, 0.8, 0.9591 and 0.8748 for classes 1, 2, 3, 4 and 5 respectively. Also, for the different

SCOV case, we have probability p as 0.9, 0.95, 0.05, 0.1 and 0.55 for classes 1, 2, 3, 4 and 5 respectively. The θ_1 and θ_2 values can be obtained using the fact that the mean and SCOV of the Coxian-2 distribution are $\frac{1}{\theta_1} + \frac{p}{\theta_2}$ and $\frac{\frac{1}{\theta_1^2}+\frac{2p-p^2}{\theta_2^2}}{\frac{1}{\theta_1^2}+\frac{p^2}{\theta_2^2}+\frac{2p}{\theta_1\theta_2}}$ respectively. Note that we considered only one processing speed, $\phi = 0.52$.

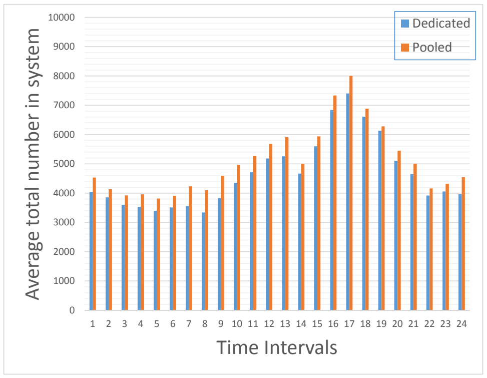
Also, we used two data sets for arrival rates, pattern A and B for performance analysis. Graphs of the arrival rates $\alpha_{a,\ell}$ for two arrival rate patterns are provided in Figures 2.2a and 2.2b respectively. In pattern A notice that arrival rate of some classes are correlated with others over time and the peak times are not necessarily the same. Our intention was to select a representative sample to illustrate both heterogeniety as well as issues such as correlation. Also, in pattern B, we defined arrival rate as sinusoidal function for $t \in \mathcal{T}$. The sinusoidal form of the arrival rate is clearly a mathematical abstraction which has the essential property of producing significant fluctuations over time (Liu and Whitt [24]). This particular arrival rate pattern is by no means critical for our approach; our approach applied to an arbitrary arrival rate but it is convenient to show how it achieved time-stable performance with time-varying arrival rates. The number of powered-on servers $N_{a,\ell}$ would be determined proportional to the arrival rate by our sizing rule in Equation (2.1) in Section 2.3.2. In all our simulations we only considered FCFS because implementing a processor sharing scheme with a large number of servers is extremely cumbersome with long run times. However, it is important to note that the time-stable results would be valid for any work-conserving policy such as processor sharing, limited processor sharing, etc. To enable a meaningful set of simulations in a reasonable time, we have only presented the FCFS case.

2.7.2 Results and Findings

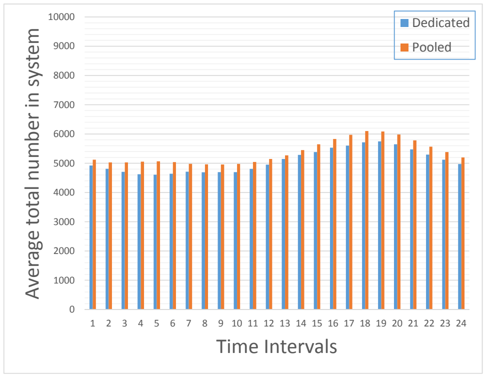
For the purpose of performance analysis, we define baseline condition which consists of the *dedicated assignment*, sizing as described in Section 2.3.2 and round-robin routing



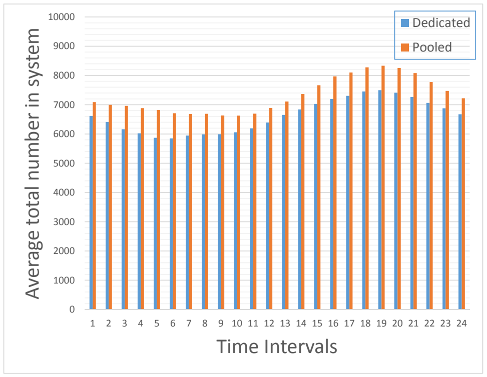
(a) Pattern A - same SCOV



(b) Pattern A - different SCOV



(c) Pattern B - same SCOV



(d) Pattern B - different SCOV

Figure 2.3: Comparing average total number in system across all classes over 1 cycle: dedicated vs pooled

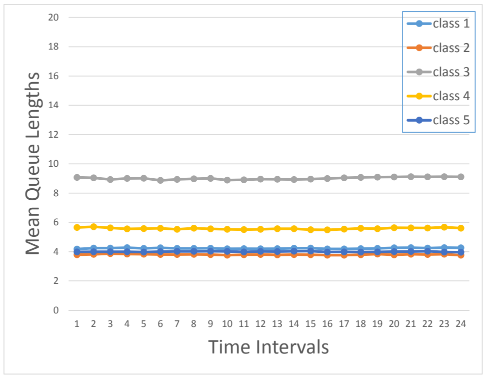
with traffic intensity $\rho = 0.9$. We will evaluate our approach by using baseline condition in following sections.

2.7.2.1 Performance Comparison Between Assignment Strategies

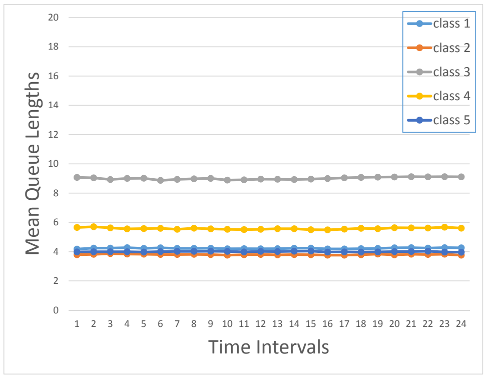
First, we compare the performance of assignment strategies to verify our insight described in Section 2.4. As described in Theorem 1, we use Bernoulli routing for the *dedicated assignment* and *pooled assignment*. Note that the total number of servers powered on at any time period is the same for both assignment strategies, we can make a fair comparison between assignment strategies. Since, as described in Section 2.5.2.1, the pooled assignment results in a non-homogeneous system, it would not be possible to use “dummy" requests for pooled assignment cases. Therefore, we compare the *dedicated assignment* case without using “dummy" requests. We compare the average total number of requests in the system by plotting it across time (also averaged across all classes) with constant SCOV in Figure 2.3a for arrival rate pattern A and in Figure 2.3c for pattern B. From the results, we can verify that *dedicated assignment* is better than pooled assignment. Moreover, we try to compare assignments with different SCOV defined in the Section 2.7.1 to check our conjecture that our insight can be extended to more general cases where SCOV is not constant and each class has a different SCOV value. Figures 2.3b and 2.3d show that *dedicated assignment* is also better than pooled assignment with a different SCOV value for each arrival rate pattern. Since cases with different SCOV values are regarded as more general, we will consider only different (and high) SCOV for further analysis.

2.7.2.2 Analysis of Time-Stability

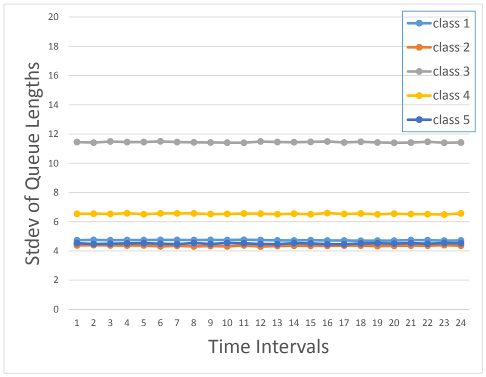
Next we analyze the time-stability of our suggested approach. As described in Section 2.5, our approach stabilizes the distributions of the queue lengths as well as the sojourn times (see Section 2.6.1). Based on both time-stable distributions, first we show that the mean and standard deviation of queue lengths for 5 classes are time-stable for round-robin (baseline) in Figure 2.4. Note that both round-robin and Bernoulli routing result in time-stable performance as mentioned in Section 2.5.2.1, but round-robin routing indeed results in better performance than Bernoulli routing as described in Section 2.3.3. For this reason we analyze time-stable performance of baseline (which use round-robin routing) for further analysis. Also based on Figures 2.4 and 2.5, it is worthwhile to indicate that our time-stable performance does not depend on arrival rate patterns which verifies the discussion in Section 2.5.2.1. In addition we check that distribution of sojourn times is also stabilized described in Section 2.6.1. As we indicated, Figure 2.5 show that the mean



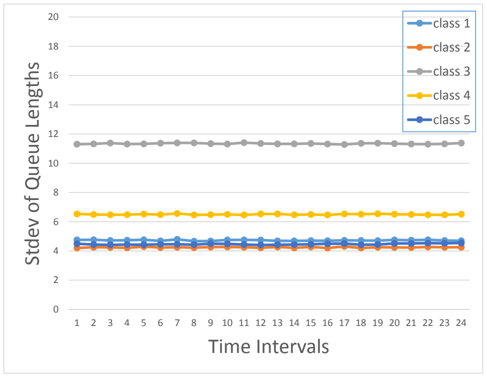
(a) Mean for pattern A



(b) Mean for pattern B



(c) Stdev for pattern A



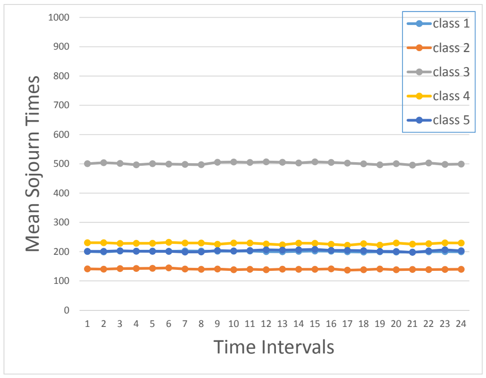
(d) Stdev for pattern B

Figure 2.4: Mean and standard deviation of queue length for the 5 classes across a cycle with round-robin routing

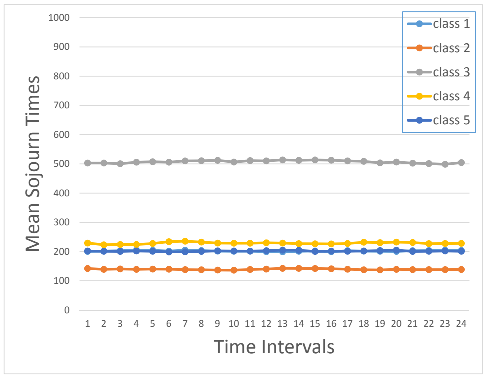
and standard deviation of sojourn times for 5 classes are time-stable.

2.7.2.3 Bounded Performance

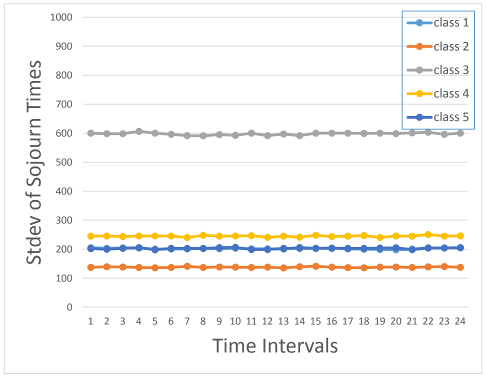
In Section 2.5.3 we mentioned that our time-stable performance measures would be an upper bound on actual performance without using dummies. Figure 2.6 compares the actual time-varying performance obtained our approach without dummies as opposed to time-stable performance bound. As we already mentioned, if dummies are not used then the mean queue lengths are time-varying across time intervals (due to empty queue of newly powered-on servers), but they are strictly bounded by the time-stable mean queue



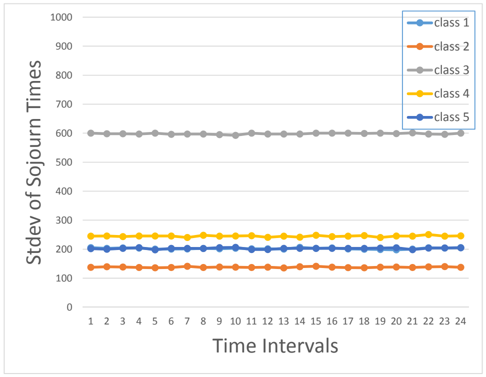
(a) Mean for pattern A



(b) Mean for pattern B



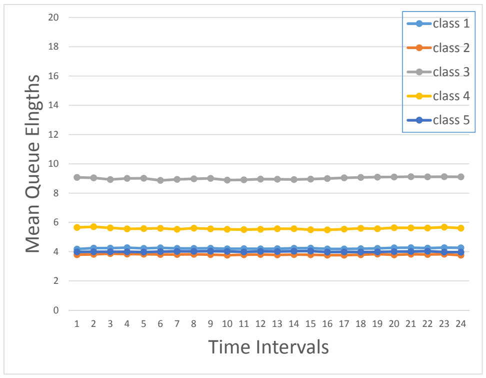
(c) Stdev for pattern A



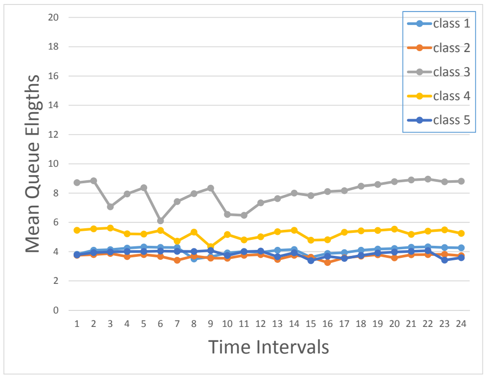
(d) Stdev for pattern B

Figure 2.5: Mean and standard deviation of sojourn times for the 5 classes across a cycle with round-robin routing

lengths obtained by adding dummies. In addition, we have claimed that the gap between actual performance and bound would be affected by both variance of workload (i.e. SCOV of workload distribution) and utilization (which can be controlled by the desired traffic intensity ρ in our approach), but it is not dependent on the time interval length. Figure 2.7 compares the differences between actual performance and bound for application class 3 (which shows the largest variation without dummies) according to the different conditions of SCOV, utilization and time interval length. As we expected, the performance gap would be bigger with bigger SCOV and higher utilization, but the same with smaller time interval length. Although the performance gap seems to be large for bigger SCOV of workload



(a) With dummies



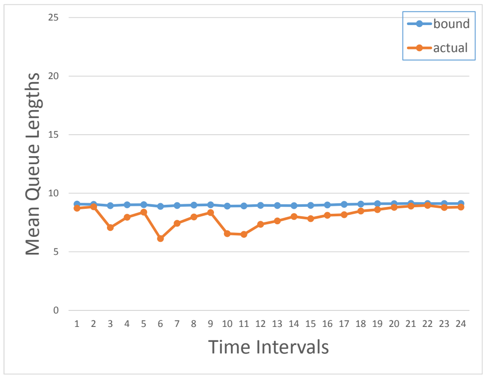
(b) Without dummies

Figure 2.6: Performance of the mean queue length for the 5 classes across 24 60-minutes time intervals

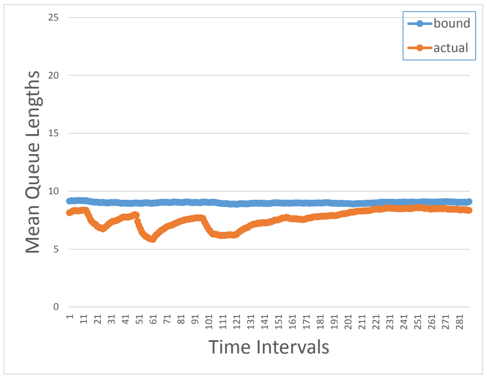
distribution and higher utilization, considering that it is also difficult to analyze dynamics of time-varying and transient system for both cases, we believe that our suggest approach still provide significant benefits based on time-stability.

2.7.2.4 The Effect of Time Interval Length

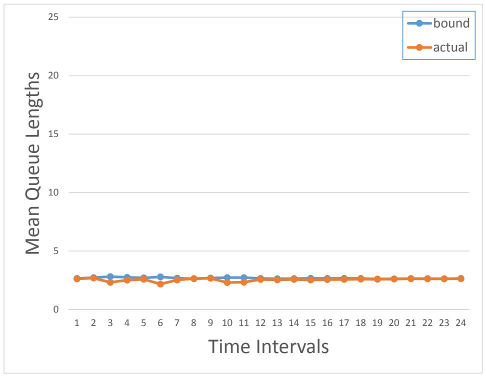
As we discussed in Section 2.6.2, in order to check whether our suggested approach performs well for the case with smaller time interval length, we run simulation for 288 5-minutes time intervals by decomposing 24 60-minutes interval into smaller ones with the same daily pattern. Recall that we used data set which has the mean service times of 5 classes as 38.46, 29.034, 48.0769, 33.8826, and 40.3846 seconds, and thus we believe that 5 minutes time intervals are appropriate to check the case of smaller time interval length. Figure 2.8 shows the bound and actual performance of the mean queue lengths for 288 5-minutes time intervals, and based on the comparison with Figure 2.6 we can conclude that time-stability obtained by our suggested approach is robust to time interval length.



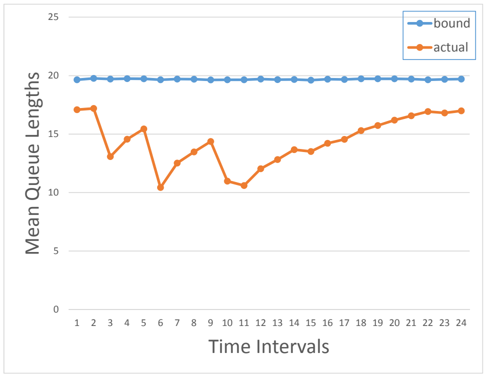
(a) 60-minutes, SCOV=2.2, $\rho = 0.9$



(b) 5-minutes



(c) SCOV=0.6

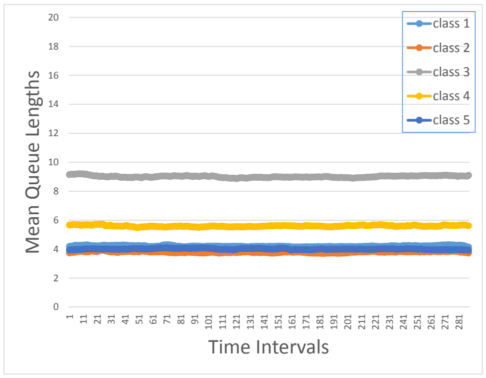


(d) $\rho = 0.95$

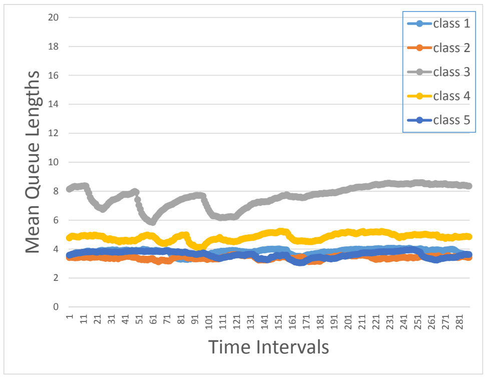
Figure 2.7: Analysis of the gap between bound and actual performance of class 3 for different conditions

2.7.2.5 *The Effect of System Size*

As we mentioned in Section 2.6.3, our suggested approach has a limitation that performance would not be stabilized with smaller size N for round-robin routing. To analyze the limitation of our suggested approach, we have run simulation by scaling with size $N = 100$ instead of $N = 1000$, summarized the results as shown in Figure 2.9. As shown in Figure 2.9, performance by using Bernoulli routing is stabilized with smaller size $N = 100$ (but as we mentioned performance is wore than round-robin), however round-robin does not yield time-stable performance for the case of size $N = 100$.

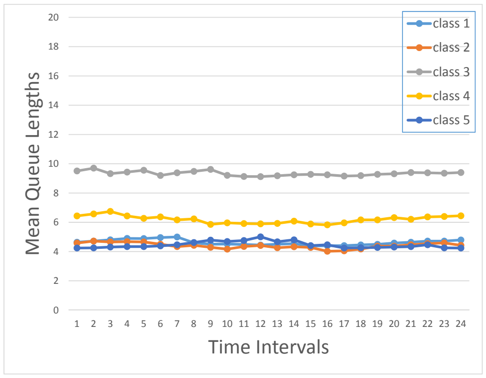


(a) With dummies

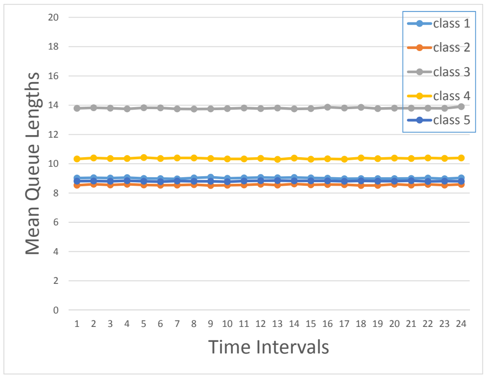


(b) Without dummies

Figure 2.8: Bounds and actual performance of the mean queue length for the 5 classes across 288 5-minutes time intervals



(a) Round-robin with $N = 100$



(b) Bernoulli with $N = 100$

Figure 2.9: Bounds on the mean queue length with $N = 100$ for both routing policies: Round-robin and Bernoulli

2.7.2.6 Dummy Traffic Analysis

In Section 2.5.2.2, we claimed that the amount of dummy traffic to adjust arrival rates of each application a is insignificant. We show percentile gap between the actual arrival rates and adjusted arrival rates in Table 2.1, and the additional dummy traffic is reasonably negligible to the actual arrivals.

Time	Class Indices				
Intervals	class 1	class 2	class 3	class 4	class 5
1	0.6200	0.3275	0.4073	1.6294	0.2857
2	0.2857	0.9125	1.0880	1.1895	0.1143
3	1.4000	0.7370	1.0880	0.6569	0.6984
4	0.5333	0.3275	0.3392	1.6294	0.0755
5	1.0880	0.1000	1.6229	0.3462	0.5042
6	1.7391	1.0540	0.1520	1.9990	0.8163
7	2.1091	1.0100	1.0880	0.5058	0.8722
8	0.4488	1.4226	0.3392	0.7533	1.6229
9	0.4073	0.4062	0.4073	0.5570	0.2857
10	0.7390	0.4250	0.5531	0.0717	1.6229
11	0.3935	0.3774	0.3711	0.8464	0.2857
12	0.1915	1.2682	0.0947	0.6986	2.2521
13	0.0552	0.4062	0.0800	0.3462	0.2857
14	0.5702	1.4226	0.3109	1.2685	1.2987
15	0.5568	1.1706	0.2631	0.7151	0.8722
16	0.3663	0.7084	0.0446	0.4384	0.2857
17	0.2857	0.5750	0.0350	0.8281	0.4746
18	0.3737	0.2857	0.2426	0.1776	0.1143
19	0.5049	1.7351	0.0552	0.3462	0.6234
20	0.2857	1.3512	0.4073	1.9990	0.0902
21	0.6200	0.0350	0.4488	0.9365	0.0583
22	0.2857	0.5136	0.5531	1.9990	1.2987
23	0.6909	0.3275	0.8800	1.4194	0.4883
24	1.1484	1.0540	0.3392	0.1192	0.2857
Overall	0.5318	0.7370	0.3752	0.7847	0.5165

Table 2.1: Percentage gap between the actual arrival rates (A) and adjusted arrival rates (B): $\frac{(B-A)}{A} \times 100(\%)$

2.8 Concluding Remarks and Future Work

A number of approaches have been studied to manage resources in data centers over non-homogeneous workloads; those approaches have mainly focused on determining right-sizing of servers to minimize energy cost while considering SLA violation conditions. However, the aforementioned studies ignore achieving time-stability which makes it convenient to analyze system, provide probabilistic guarantees and performance bound under transient conditions. To the best of our knowledge, achieving time-stability over time-varying workloads while considering sizing, assignment and load balancing in integrated fashion for data centers operations has not been addressed. In this context, we suggest an approach to effectively reduce energy consumption by powering on and off just the right number of servers while being able to provide performance bounds and guarantees over fast varying arrival rates that steady-state cannot typically be reached.

This paper asks if time-stability can be attained using a combination of sizing, assignment, and routing in an integrated fashion. We have suggested an analytic framework simplifying a large scale, multi-dimensional, and non-stationary problem by decomposing into individual simpler stationary ones, and have introduced dummy requests to achieve time stability based on decomposed settings. Performance bounds and probabilistic guarantees introduced in this study are provable and simply derived by stationary analysis based on suggested framework. Also, we have introduced additional insight regarding assignment strategies and addressed extension and limitation of our suggested approach. One could consider the following in the future: (i) suggest real-time speed scaling control by varying ϕ for time-stable performance, (ii) instead of all classes with a large number of servers some classes may need to be hosted on only one server, (iii) develop an optimization framework to holistically right-size, speed scale, route and assign classes for energy efficiency, and (iv) extend to multi-server queues.

3. GUARANTEEING PERFORMANCE BASED ON TIME-STABILITY FOR ENERGY-EFFICIENT DATA CENTERS*

3.1 Introduction

We consider a data center which consists of a set of heterogeneous servers with different maximum processing speeds and different capacity limits of various resources, such as memory and storage. The data center hosts heterogeneous application classes which have different workload distributions, for which time-varying requests arrive with resource requirements and levels of quality of service (QoS) guarantees. In this case, servers process requests that belong to multiple classes, whereas requests categorized into the same class are stochastically identical, and they arrive to data centers according to a piecewise constant non-homogeneous Poisson process. It is assumed that the environment process that drives arrival rates of the non-homogeneous Poisson process is cyclic. This is a fairly reasonable assumption as arrivals tend to have daily or weekly patterns that repeat in a cyclic fashion (Gmach et al. [28], Lin et al. [29], Liu et al. [30], Lin et al. [1] and Kwon and Gautam [2]). Using that assumption we model each cycle as divided into a set of intervals corresponding to the piecewise constant period for the arrival process. In addition, we assume that the number of classes is large (larger than the number of servers, e.g. 10 servers and 20 applications) but their workloads are so little that most servers host a mixture of heterogeneous classes. For the non-homogeneous arrival process, the arrival rate of each class is time-varying and also changes so fast that steady-state is not reached before arrival rates change. Based on the scenario mentioned above, we model a data center as a system of multiple parallel single server queues where a dispatcher routes arriving applications to

*Reprinted with permission from "Guaranteeing Performance based on Time-stability for Energy-efficient Data Centers" by Soongeol Kwon and Natarajan Gautam, 2016, *IIE Transactions*, 48(9), 812-825, Copyright © 2016 Taylor & Francis.

servers.

For such a time-varying and heterogeneous system, our fundamental aim is to manage servers of data centers in an energy-efficient manner while satisfying performance guarantees by achieving time-stability. For time-varying and heterogeneous systems, arrival rates of requests change so fast that a steady-state of system is not reached, therefore making it extremely difficult to provide performance guarantees. Without assuming that system reaches steady-state, one can provide performance guarantees by powering on large number of servers responding to peak workload; however, that causes a huge additional energy cost than necessary. In this context, achieving time-stability is extremely useful for managing a non-homogeneous and transient system because it enables operators to effectively design and analyze the time-varying system, provides guaranteed QoS as desired, and effectively performs monitoring and control. Achieving time-stability and providing performance guarantees, while being mindful of energy costs, is the main objective of this study. To achieve time-stability, we consider the following control decisions that can be tuned.

- *Assignment*: Applications corresponding to each class of request can be assigned to servers such that each server hosts one or more classes and each class is hosted on multiple servers. We assume that there is no cost for the assignments as well as switching between assignments.
- *Sizing*: Each server could be dynamically powered on or off across time intervals. However, we neither consider switching costs (from on to off and vice versa) nor consider reliability costs for on-off cycles. Note that some modern servers allow for “sleep" settings instead of completely turning off servers. From a mathematical standpoint, we consider them equivalent.
- *Routing Fraction*: The dispatcher routes arriving requests to one of the powered-

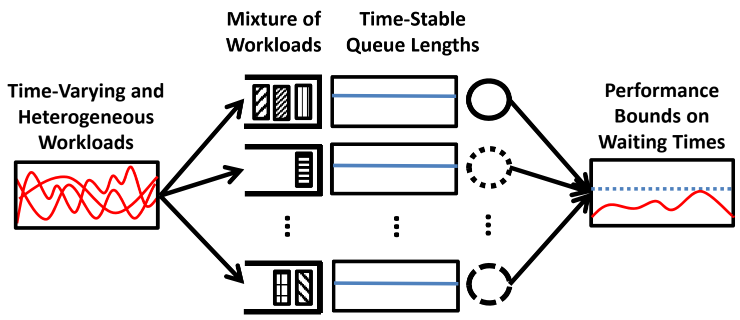

Mixture of Workloads
Time-Stable Queue Lengths
Time-Varying and Heterogeneous Workloads
Performance Bounds on Waiting Times

Figure 3.1: Achieving time-stability for multiple parallel single queues with time-varying and heterogeneous workloads

on servers based on predetermined routing fractions by considering assignment of classes to servers. A key assumption is that the dispatcher cannot observe the real-time state of any of the servers.

- *Speed Scaling*: For each server, it is possible to dynamically change the processing speed by scaling the voltage and frequency up to a maximum speed.

We will show that assignment, sizing, routing and speed scaling can be done appropriately in an integrated fashion to achieve both time-stability and energy-efficiency through the suggested framework described in Sections 3.2 and 3.3. Figure 3.1 briefly shows the scenario and the main objective of this study. Next we review the relevant literature and introduce notation used in this paper.

3.1.1 Literature Review

As there has been a surge in demand for cloud computing in recent years, server management in data centers has received tremendous attention in both industry and academia. Data centers provide benefits in cost reduction, flexibility, and accessibility by allowing enterprises to outsource resources for service rather than managing their own resources. As a result, data centers have challenging tasks to achieve energy efficiency and provide performance guarantees in cloud computing environments characterized by time-varying workloads with significant variation and uncertainties. In general, data centers need to provide strict QoS guarantees to users, thereby over-provisioning their servers to respond to peak loads due to inherent uncertainty and variability in demand. On one hand this over-provisioning of servers results in low utilization as reported in Barroso and Hölzle [33] and Vogels [12]. On the other hand they incur a significant amount of energy usage for operating and cooling servers as explained and documented in Hamilton [9] and Koomey [10]. Fortunately, to abate the skyrocketing energy consumption in data centers (see report [7]), there are tremendous opportunities to conserve energy consumption in data centers, such as powering servers off and running them at slower speeds.

In this context, server provisioning plays a key role in improving utilization by selecting active servers (e.g. powering off servers or allowing to enter a power-saving mode) in accordance to traffic changes, while considering performance guarantees, and thus a number of techniques for efficient server provisioning have been proposed to address the above problem. Gandhi et al. [19] presented an approach based on a combination of predictive and reactive provisioning to correctly allocate resources in data centers such that service level agreement (SLA) violations and energy consumption are minimized. Zhu et al. [34] presented a data center architectural design based on virtualized resources in order to reduce provisioning overhead; they also proposed a dynamic provisioning technique while satisfying user's SLA and maximizing overall profits. Also, Wang et al. [35] provided an analytic framework that captures non-stationarities and stochastic variation of workloads for dynamic re-sizing in data centers. Lin et al. [1] suggested a new on-line algorithm for dynamic right sizing in data centers motivated by optimal off-line solutions to minimize energy costs including switch costs. In addition, there have been some studies that considered heterogeneous workloads for server provisioning and allocation problems. There is another body of literature that proposed an optimization approach based on powering servers on/off and dynamic voltage/frequency scaling (DVFS) to minimize energy consumption. Bertini et al. [36] proposed a mixed integer program (MIP) for the problem of selecting the servers' states and processing speeds with QoS control, and Gallego Arrubla et al. [20] introduced a unified methodology that combines virtualization, speed scaling, and powering off servers to efficiently operate data centers while incorporating the inherent variability and uncertainty.

While all of the aforementioned research is complementary to our work, the key difference is that our study suggests an approach which aims to not only save energy consumption but also achieve time-stability for providing performance guarantees. Note that, to the best of our knowledge, the problem of achieving time-stability over time-varying traffic in data center operation has received little attention and not been effectively addressed. Although time-stability has received little attention in the context of data centers operation, there have been some research studies in the queueing area. Foley et al. [21] and Barnes et al. [22] showed that the departure process from the $M_t/G_t/\infty$ queue can be made stationary, and in recent days, Whitt [37] suggested the rate matching control algorithm, which stabilizes the queue length distribution for $G_t/G_t/1$ single-server queue where both arrival rate and service rate are time-varying. There is another body of literature which provides algorithms to determine appropriate staffing levels for call centers. Feldman et al. [23] proposed a simulation-based iterative-staffing algorithm for time-stable delay probability, and Liu and Whitt [24] suggested a formula-based algorithm to stabilize abandonment probabilities and expected delays using offered-load based approximations for a queueing model with the non-homogeneous Poisson arrival process and customer abandonment.

In fact, our previous work [2] suggests an approach to achieve time-stability in both queue lengths and sojourn times for data center operations where time-varying arrivals are cyclic. In [2], we considered a scenario of a company that owns a data center (e.g. Yahoo, Google or Facebook) and operates a large number of servers to host applications. We assumed that each application request has high squared coefficient of variation (SCOV) of workloads. Also, each application needs to be hosted on a large number of servers to handle the load (in fact, we use an asymptotic scaling, the number of servers $N_a \to \infty$ for each application a, which allowed us to attain time-stability). In comparison, this study considers a unique scenario for hosting data centers that provide hosting service to several other companies. In general, hosting data centers cluster applications of each company and assign the cluster to a group of servers for the purpose of security and confidentiality. It is important for the data centers to monitor the performance experienced by the applications, and without time-stable performance, monitoring would be difficult. This motivation is to consider time-stability. We assume that the number of application classes is much larger than the number of servers and many application classes have so little load that they would need to be hosted on just one server. Also, we assume that servers are heterogeneous and host a mixture of heterogeneous application requests, and processing speed of a server can be changed for energy efficiency (in [2], servers are homogeneous and every server runs at the same processing speed). Moreover, in this study, we formulate an MIP problem to optimally determine operational decisions including assignment, routing, and speed scaling for time-stability as well as energy efficiency, which were not considered in [2]. Therefore, our suggested model in this study is fundamentally different from the problem considered in [2].

For the purpose of this study, we provide an analytical framework which decomposes a complex and non-stationary system into individual simpler stationary ones based on multiple strategies for assignment, sizing, routing, and speed scaling. Based on the suggested framework, our objective is to stabilize queue length distributions of each powered-on server and provide performance guarantees on waiting time of each application class. Moreover, for energy efficiency we propose an optimization model to minimize total energy cost via powering on or off servers, routing, and speed scaling, while satisfying the time-homogeneity constraints. In fact, our suggested approach enables us to utilize standard stationary queueing analysis to obtain performance guarantees for time-varying, transient, and heterogeneous systems, and we believe that our study has significance and provides useful insights for practitioners. The main contributions of this study are to: (i) provide an integrated framework unifying sizing, assignment, routing, and speed scaling under heterogeneous conditions which has seldom been implemented jointly; (ii) define time-homogeneity constraints which ensures time-stability; (iii) suggest an approach to provide performance guarantees based on time-stability; and (iv) introduce an optimization problem with an MIP formulation to reduce energy cost while considering time-stability. The remainder of the paper is organized as follows: Section 3.2 introduces our notion of time-stability and suggests an approach to obtain time-homogeneity constraints; Section 3.3 proposes an optimization problem to determine various decisions for energy conservation and time-stability; Section 3.4 proposes an approach to provide performance guarantees; Section 3.5 reports and analyzes the results of numerical experiments; and Section 3.6 presents conclusions and future research directions.

3.1.2 Notations

For the scenario considered in this study, we set the notations to define the problem and describe our approach appropriately. An arriving request belongs to one of multiple classes in a discrete set $\mathcal{A}$. We categorize incoming requests into several class types based on their mean workload and each class has a small squared coefficient of variance. The amount of work a class $a \in \mathcal{A}$ request brings is independent and identically distributed (IID) according to a general distribution $H_a(\cdot)$ with mean $1/\theta_a$ and SCOV C_a^2. Also, each class a has requirements β_a^k for each resource type $k \in \mathcal{K}$ (this could be memory or storage, for example). Let $\mathcal{T}$ be a collection of contiguous time intervals and in each interval $\ell \in \mathcal{T}$ the arrival rate for each class $a \in \mathcal{A}$ remains a constant $\lambda_{a\ell}$. Also, we consider

$\mathcal{N}$ heterogeneous servers and each server $j \in \mathcal{N}$ has maximum processing speed ϕ^j_{max} and capacity limit for each resource type $k \in \mathcal{K}$, b^k_j. Note that instead of defining service time distribution, which is generally used in other studies based on queueing models, our approach uses the combination of workload distribution and processing speed (which could be varying). In addition, we define the desired traffic intensity ρ for each powered-on server. In fact, we use the desired traffic intensity to create enough residual capacity for unforeseen surges by restricting the utilization of each server to be ρ. Next, for time interval $\ell \in \mathcal{T}$, server $j \in \mathcal{N}$, class $a \in \mathcal{A}$, and resource type $k \in \mathcal{K}$ we define decision variables considered in this study as follows: (i) the assignment of class a to servers j for each time interval ℓ, $x_{aj\ell}$ (e.g. $x_{aj\ell} = 1$ if class a assigned to server j in time interval ℓ, otherwise $x_{aj\ell} = 0$.), (ii) whether server j must be powered on or off in time interval ℓ, $y_{j\ell}$ (e.g. $y_{j\ell} = 1$ if server j is powered on in time interval ℓ, otherwise, $y_{j\ell} = 0$), (iii) the processing speed for server j in time interval ℓ, $\phi_{j\ell}$, and (iv) the fraction of jobs of application a to be routed to server j in time interval ℓ, $v_{aj\ell}$. Each server j must be powered on or off in each time interval ℓ, and thus there will be $N_\ell = \sum_{j \in \mathcal{N}} y_{j\ell}$ powered-on servers in each time interval ℓ. We summarize the set of indices, parameters, and decision variables which are used to define optimization problem in Table 3.1.

3.2 Time-stability

Recall that our main objective is to provide performance guarantees for time-varying and heterogeneous data centers by achieving time-stability. In this section, first we introduce the notion of time-stability considered in this study and then we suggest an approach to achieve time-stability. We conclude this section by introducing a practical application of a suggested approach considered in this study. Recall that the benefit of time-stability has been discussed in Section 3.1.

Indices	
$\mathcal{T}$	index set of time intervals $\ell \in \mathcal{T}$
$\mathcal{A}$	index set of classes of requests $a \in \mathcal{A}$
$\mathcal{N}$	index set of servers $j \in \mathcal{N}$
$\mathcal{K}$	index set of types of resources $k \in \mathcal{K}$
Parameters	
$\lambda_{a\ell}$	arrival rate of class a in time interval ℓ
$1/\theta_a$	average amount of workload brought by class a
ρ	desirable traffic intensity for each powered-on server
β_a^k	requirement of resource k for class a
b_j^k	capacity limit of resource k for server j
ϕ_{max}^j	maximum processing speed of server j
Decision variables	
$x_{aj\ell}$	assignment of class a to server j in time interval ℓ
$y_{j\ell}$	whether server j must be powered-on or off in time interval ℓ
$v_{aj\ell}$	fraction of class a to route to server j in time interval ℓ
$\phi_{j\ell}$	processing speed of server j in time interval ℓ

Table 3.1: Indices, parameters, and decision variables of suggested optimization problem

3.2.1 Our Notion of Time-Stability

As described in Section 3.1, we consider a scenario where each server may host multiple classes of applications with aggregate workload defined by a mixture of heterogeneous application classes. For achieving time-stability, we seek to manage the system so that any powered-on server receives arrivals with a target workload distribution $H(\cdot)$ whose Laplace-Stieltjes transform (LST) is $\tilde{H}(s) = \int_0^\infty e^{-sx} dH(x)$. We denote $1/\theta$ as the target average amount of work brought by each arrival. We seek to keep the aggregate workload distribution at any instant of time in each powered-on server to be time-stable in order to ensure that the distribution of queue lengths at any time for every powered-on server is according to a stationary distribution of a homogeneous $M/G/1$ queue. In Section 3.2.2, we will propose an approach to obtain time-stable queue length distribution by stabilizing both the aggregate workload distribution and arrival rates, and also in Section 3.3 we will show that time-stability would be valid for speed scaling. For mathematical representation, for all $j \in \mathcal{N}$, at time t let $X_j(t)$ be the number of requests in queue of server j and $O_j(t)$ be the status of the server ($O_j(t) = 1$ if server j is powered-on at time t, otherwise $O_j(t) = 0$). Our approach indeed stabilizes queue length distribution $\pi(i)$ for all $i \geq 0$ which can be represented as $P\{X_j(t) = i | O_j(t) = 1\} = \pi(i)$ where $\pi(i)$ is constant and not dependent on t. Based on the time-stability in queue length distribution, performance analysis is straightforward and probabilistic guarantees on the waiting times can be provided.

3.2.2 Approach to Achieve Time-Stability

In this section we suggest an approach to achieve time-stability introduced in Section 3.2.1. For time-varying arrivals of requests of heterogeneous applications, the main idea of our suggested approach is to stabilize queueing process of each powered-on server based on (i) time-invariant aggregate workload distribution obtained by moment matching approximation and (ii) rate matching between arrival rates and processing speeds by selecting routing fractions appropriately. In the following subsections, we will derive time-homogeneity constraints for the proposed MIP problem.

3.2.2.1 Moment Matching Approximation for Time-invariant Workload Distribution

For achieving time-stability, we seek to stabilize an aggregate workload distribution of each powered-on server where the servers host incoming requests of multiple heterogeneous applications that have different workload distributions $H_a(\cdot)$. To achieve the invariant target workload distribution $H(\cdot)$ with LST $\tilde{H}(s)$ and average workload $1/\theta$, our objective is to control the system appropriately. This way each powered-on server receives arrivals of requests with homogeneous aggregate workload distribution $H(\cdot)$ based on moment matching approximation. To begin, for the desirable traffic intensity ρ and the target nominal speed ϕ, we determine the minimum number of powered-on servers for each time interval $\ell \in \mathcal{T}$, N_ℓ, which satisfies the following inequality:

$$\sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}_a(s) \leq N_\ell \tilde{H}(s)\rho\phi\theta \quad \forall s \geq 0 \tag{3.1}$$

where the left-hand side is the LST of aggregate workload distribution for the entire system and the right-hand side represents the LST of aggregate workload distribution of N_ℓ time-stable servers, where each of the N_ℓ servers has the target workload distribution $H(\cdot)$ and arrival rate as $\rho\phi\theta$. Based on the inequality (3.1), N_ℓ can be determined so that the workload served by each of the N_ℓ servers would be less than equal to the target workload (which is defined by traffic intensity ρ, processing speed ϕ, and mean workload $1/\theta$) when the incoming workload is equally distributed with the same routing fractions to the powered-on servers. In fact, we can redefine inequality (3.1) based on moment matching approximation for several moments as follows:

$$\begin{gathered}
\sum_{a\in\mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} \leq N_\ell\rho\phi \\
(-\tilde{H}'(0))\sum_{a\in\mathcal{A}} \lambda_{a\ell} \leq N_\ell\rho\phi \\
(-\tilde{H}'(0))\sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}_a''(0) \leq N_\ell\rho\phi\tilde{H}''(0) \\
(-\tilde{H}'(0))\sum_{a\in\mathcal{A}} \lambda_{a\ell}(-\tilde{H}_a'''(0)) \leq N_\ell\rho\phi(-\tilde{H}'''(0)) \\
\vdots
\end{gathered}$$

by taking the derivatives of the LST $\tilde{H}(s)$ at $s = 0$. Our aim is to introduce additional traffic to match the difference between the left-hand side and the right-hand side of each inequality (each moment) so that the aggregate workload distribution of each of the N_ℓ servers is approximately equal to the target workload distribution $H(\cdot)$. As discussed in

[38], [39], and [40], additional traffic can be thought of as low priority jobs (or delay-tolerant jobs) that do not have time constraints that data centers need to process. For additional traffic with index 0, arrival rate in time interval ℓ, $\lambda_{0\ell}$, and the moment of workload distribution for each time interval ℓ, $-\tilde{H}'_{0\ell}(0)$, $\tilde{H}''_{0\ell}(0)$, $-\tilde{H}'''_{0\ell}(0), \ldots$, can be determined appropriately through the following equalities based on N_ℓ:

$$\begin{aligned}
(-\tilde{H}'(0))\left(\lambda_{0\ell}+\sum_{a\in\mathcal{A}}\lambda_{a\ell}\right) &= N_\ell\rho\phi \\
\frac{\lambda_{0\ell}}{\theta_{0\ell}}+\sum_{a\in\mathcal{A}}\frac{\lambda_{a\ell}}{\theta_a} &= N_\ell\rho\phi \\
(-\tilde{H}'(0))\left(\lambda_{0\ell}\tilde{H}''_{0\ell}(0)+\sum_{a\in\mathcal{A}}\lambda_{a\ell}\tilde{H}''_a(0)\right) &= N_\ell\rho\phi\tilde{H}''(0) \\
(-\tilde{H}'(0))\left(-\lambda_{0\ell}\tilde{H}'''_{0\ell}(0)+\sum_{a\in\mathcal{A}}\lambda_{a\ell}(-\tilde{H}'''_a(0))\right) &= N_\ell\rho\phi(-\tilde{H}'''(0)) \\
&\vdots
\end{aligned}$$

and then we can approximately define $H_{0\ell}(\cdot)$ by combining the derivatives. Recall that there is additional traffic which has low priority of QoS guarantees (e.g. non-interactive jobs are less sensitive to response time) and is also CPU intensive (it does not need other resource requirements). In this case, it is reasonable to assume that this additional traffic can be assigned to the system according to a Poisson process with parameter $\lambda_{0\ell}$ and workload distribution $H_{0\ell}(\cdot)$ for each time interval ℓ from front-end proxy servers or other data centers. We would like to note that our suggested approach allows additional traffic so that the increment of workloads caused by an addition of traffic would not degrade the desired performance, but would reduce variability and uncertainty in aggregate workload while also improving the utilization of CPU for each powered-on server. Note that if we match sufficiently many moments, then the above equalities would result in

$$\lambda_{0\ell}\tilde{H}_{0\ell}(s) + \sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}_a(s) = N_\ell \tilde{H}(s)\rho\phi\theta \quad \forall s \geq 0 \tag{3.2}$$

where $H_{0\ell}(\cdot)$ has $-\tilde{H}'(0)$, $\tilde{H}''(0)$, $-\tilde{H}'''(0), \ldots$ as moments. By selecting additional traffic for several moments, our suggested approach stabilizes workload distribution for each powered-on server.

3.2.2.2 Selecting Routing Fractions for Time-stable Arrival Rates

In addition to stabilizing aggregate workload distribution as introduced in Section 3.2.2.1, next our suggested approach stabilizes an aggregate arrival rate to each powered-on server so that the queue length distribution of each powered-on server is a stationary time-homogeneous $M/G/1$ queue. Based on the minimum number of powered-on servers N_ℓ, arrival rates $\lambda_{0\ell}$ and workload distributions $H_{0\ell}(\cdot)$ of additional traffic for each time interval $\ell \in \mathcal{T}$, we seek to route arrivals of requests so that arrival rates into each powered-on server would be time-homogeneous (i.e. constant across time-intervals) while ensuring the aggregate workload distribution is also stabilized as the target workload distribution $H(\cdot)$. Let $v_{aj\ell}$ be the fraction of arriving requests of class a routed to server j in time interval ℓ, then $v_{aj\ell}$ can be appropriately selected based on the following time-homogeneity constraints:

$$\sum_{j=1}^{N_\ell} v_{aj\ell} = 1 \quad \forall a \in \mathcal{A} \cup 0 \tag{3.3}$$

$$\frac{\lambda_{0\ell}}{\theta_{0\ell}} v_{0j\ell} + \sum_{a\in\mathcal{A}} \frac{\lambda a\ell}{\theta_a} v_{aj\ell} = \rho\phi \quad \forall j \in \{1, 2, ..., N_\ell\} \tag{3.4}$$

$$\frac{\lambda_{0\ell} v_{0j\ell}\tilde{H}_{0\ell}(s) + \sum_{a\in\mathcal{A}} \lambda_{a\ell} v_{aj\ell} \tilde{H}_a(s)}{\sum_{a\in\mathcal{A}\cup 0} \lambda_{a\ell} v_{aj\ell}} = \tilde{H}(s) \quad \forall j \in \{1, 2, ..., N_\ell\}, s \geq 0. \tag{3.5}$$

Recall that ρ is the desired traffic intensity and ϕ is the target nominal speed. Equation (3.3) means that all application requests should be routed to servers; traffic intensity of each powered-on server would be ρ based on Equation (3.4); Equation (3.5) ensures that aggregate workload distribution at each powered-on server j is approximately equivalent to the target workload distribution $H(\cdot)$. In fact Equation (3.5) is directly derived from Equation (3.2) by applying $v_{aj\ell}$ to routing arriving requests instead of distributing equally to N_ℓ powered-on servers. Based on Equations (3.4) and (3.5), arrival rates into each powered-on server would remain constant at $\rho\phi\theta$ across time intervals, since the mean amount of workload brought by incoming request would be $1/\theta$ based on Equation (3.5) (i.e. $-\tilde{H}'(0) = 1/\theta$) and thus,

$$\sum_{a \in \mathcal{A} \cup 0} \frac{\lambda_{a\ell}}{\theta} v_{aj\ell} = \rho\phi, \quad \sum_{a \in \mathcal{A} \cup 0} \lambda_{a\ell} v_{aj\ell} = \rho\phi\theta \quad \forall j \in \{1, 2, ..., N_\ell\}.$$

We will formulate an MIP problem by using time-homogeneity constraints (3.3), (3.4), and (3.5) to optimally determine $v_{aj\ell}$ to achieve time-stability in Section 3.3.

3.2.2.3 Adjusting Initial Condition of Time Intervals

The fundamental idea for reducing energy cost is to power servers on and off as they respond to time-varying workloads. To that end, we will formulate an MIP problem in Section 3.3 to determine operational decisions (including powering servers on and off) so that the queue length distribution of every powered-on server is stabilized across time intervals. In this case, it is intuitive to think that time-stability (stabilized queue length distribution) would be affected when servers are powered-on afresh (from powered-off state) since newly powered-on servers start processing jobs with an empty queue; however in the other case, servers process jobs with a stationary queue. Thus, queue length distribution would not be stabilized when the length of each time interval is not long enough to reach steady-state. To address this problem, we add a batch of requests to servers which are powered-on afresh at the beginning of a time interval so that queue lengths of every powered-on server would be stochastically equivalent to that of a stationary $M/G/1$ queue. We adopt an approach proposed by our previous work [2] whose main idea is adding a number of jobs sampled from the stationary queue length distribution defined by the stabilized workload distribution ($H(\cdot)$) and arrival rates ($\rho\phi\theta$). If servers selected to be powered-off are not idle at the end of time interval, then incoming requests would not be routed to those servers, and we wait until those servers complete service for the remaining requests (and then power off the servers). As we discussed in [2], the key benefit of the suggested approach is that performance bounds based on time-stability are provable and also easily derived by using standard queueing theory results. In addition, for the general case, we have the following remark for an extension of our suggested approach.

Remark 3 (Extension of Time-stability to Processor Sharing)**.** *In Section 3.2.1, we introduce our notion of time-stability based on stationary $M/G/1$ queues where each server hosts a mixture of multiple heterogeneous applications with first-come first-served (FCFS) service discipline. Here our claim is that our suggested approach can be extended to a model using processor sharing (PS) service policy. Under PS regime, all the jobs or entities in the system are served simultaneously and equally share the processor at any given time; it is also fairly common to model computer servers based on PS service policy. In fact, $M/G/1$ queue with PS does produce closed form results which are identical to those of $M/M/1$ queues with FCFS discipline [32]. Thus, our suggested approach for time-stability can be applied to the model which uses PS instead of FCFS for service policy. For PS service policy, the mean queue length L of $M/G/1$ would be $L = \rho/(1-\rho)$, and thus the mean sojourn time would be $1/(\theta\phi(1-\rho))$.*

3.2.3 Practical Application of The Suggested Approach

In Section 3.2.2, we introduce the notion of time-stability considered in this study and suggest an approach to achieve time-stability based on moment matching approximation. As we mentioned, if we match sufficiently many moments, then aggregate workload distribution at every powered-on server would be approximately equivalent to $H(\cdot)$, and thus we could theoretically have stabilized queue length distribution. Moreover, for the practical application, we can choose to match only a few moments to achieve time-stability. In this section, we will show that the mean queue lengths of every powered-on server can be reasonably stabilized across time intervals by considering only first and second moments in practice. Recall that our aim is to manage each powered-on server as a stationary $M/G/1$ queue, and we use the Pollaczek-Khintchine (P-K) formula [32] to determine the mean queue length L as follows:

$$L = \rho + \frac{\rho^2(1 + C^2)}{2(1 - \rho)} \tag{3.6}$$

where C^2 is the SCOV of service time and ρ is the traffic intensity. Based on our suggested approach, we need to determine the target workload distribution $H(\cdot)$, and in fact, $H(\cdot)$ can be determined according to the desired mean queue length. For example, let $\bar{L}$ be the desired (i.e. targeted) mean queue length; then it is possible to select C^2 of $H(\cdot)$ based on Equation (3.6) for given $\bar{L}$ and ρ (recall that ρ is the desired traffic intensity). Since C^2 is solely determined by the first and second moments of $H(\cdot)$ ($-\tilde{H}'(0)$ and $\tilde{H}''(0)$, respectively), the mean queue length can be stabilized as desired by determining the first and second moments of $H(\cdot)$ and matching the first and second moments as follows:

$$\frac{\lambda_{0\ell}v_{0j\ell}(-\tilde{H}'_{0\ell}(0)) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}(-\tilde{H}'_{a}(0))}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = -\tilde{H}'(0) \quad \forall j \in \{1,2,...,N_\ell\}, s \geq 0 \tag{3.7}$$

$$\frac{\lambda_{0\ell}v_{0j\ell}\tilde{H}''_{0\ell}(0) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}\tilde{H}''_{a}(0)}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = \tilde{H}''(0) \quad \forall j \in \{1,2,...,N_\ell\}, s \geq 0. \tag{3.8}$$

In Section 3.3, we will formulate an MIP by using the constraints (3.7) and (3.8) instead of using (3.5) to stabilize the mean queue length and provide performance guarantees on the mean waiting time.

3.3 Optimization Problem

In this section, we suggest an optimization problem to determine operational decisions that correspond to decision variables $x_{aj\ell}$, $y_{j\ell}$, $v_{aj\ell}$, and $\phi_{j\ell}$ introduced in Section 3.1.2. The key idea is to select decision variables, $x_{aj\ell}$, $y_{j\ell}$, $v_{aj\ell}$, and $\phi_{j\ell}$ so that it not only enforces time-stability of the queue length distribution at every powered-on server but also results in the system being energy-efficient.

3.3.1 Assumption for Energy Cost

For considering the energy cost of data center operation, we define a fixed energy cost f_j for powered-on server j (i.e. energy cost for server j at “idle" state) and an operating cost c_j for processing jobs at server j at speed $\phi_{j\ell}$. In fact, we are assuming that there is no cost and no service delay for switching server operations between time intervals. Next we briefly describe the reason we assume there would be no switching cost for our suggested problem. According to [29] and [1], the switching cost mainly consists of (i) energy used for powering servers on and off, (ii) a time delay that the server is in setup, and (iii) increased wear-and-tear on the server toggling. For (i) and (ii), firstly, we would like to note that the length of time intervals used in our suggested model is much longer than the duration of setup time. For example, authors in [41], [42], and [43] reported that setup times are ranging from 20 seconds to 200 seconds, which are very small compared to the one hour (3600 seconds) time interval length. In this case, the power consumed during setup can be negligible compared with the power consumption for running a server during a one-hour time interval, and thus, it is reasonable to only consider operating cost without considering extra setup cost. Also our suggested approach is a static control algorithm, and server schedules (e.g. $y_{j\ell}$ and $\phi_{j\ell}$ for all $j \in \mathcal{N}$ and $\ell \in \mathcal{T}$) are pre-determined, as opposed to dynamic server provisioning considered in [29] and [1], which determines whether to put idle servers to sleep or wake up servers in real-time. Thus, based on the scenario considered in our problem, servers can be set up to process jobs in advance based on the pre-determined server schedules, and thus service delay also can be negligible. In terms of (iii), a body of literature (e.g. [44], [42], [29], [1], and [35]) considers the impact of server on/off cycles on the reliability of the server. For example, the recent studies [29], [1], and [35] proposed an approach for the dynamic server provisioning model by considering switching cost that includes server reliability cost for wear-and-tear caused by toggling servers. However, contrasting the dynamic server provisioning algorithms proposed by [29], [1], and [35], which change server state frequently, the servers in our problem would stay at powered-on or powered-off (or sleep) state for at least one hour and would not change server state frequently; thus the effect of toggling servers on reliability may not be significant. Moreover, as mentioned in [45] and [46], powering servers off may extend the lifetime of server components. Therefore, toggling servers may not significantly affect reliability of servers in our proposed problem. In addition, since the assignment of classes to servers are changing across time intervals in our suggested problem, one may argue that there may exist a cost for switching assignments. We would like to mention that cost for switching assignments can be easily ignored since all applications essentially reside in all servers and in a given time interval, the applications that are used are fired up while others are off. Also, it is reasonable to assume that the assignment of applications to servers can be changed without service delay since applications can be deployed on servers concurrently while they process other jobs. Based on the above description, we consider only fixed energy cost and operating cost without including server switching cost in our suggested problem.

3.3.2 Formulation

For time-stability, the target aggregate workload distribution $H(\cdot)$, and arrival rates of additional traffic $\lambda_{0\ell}$ and workload distribution $H_0(\cdot)$ are determined as described in Section 3.2. Recall that we define time-homogeneity constraints to stabilize the mean queue lengths by using constraints (3.7) and (3.8) based on the first and second moments of workload distribution of additional traffic (indexed as “0") $1/\theta_{0\ell}$ and $\tilde{H}''_{0\ell}(0)$. Based on the set of indices, parameters, and decision variables, we formulate an MIP optimization problem. For each time interval $\ell \in \mathcal{T}$, our suggested optimization problem can be formulated as follows:

$$\text{MIP-}\ell\text{: Minimize} \quad \sum_{j \in \mathcal{N}} (f_j y_{j\ell} + c_j \phi_{j\ell}) \tag{3.9}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{N}} v_{aj\ell} = 1 \qquad \forall a \in \mathcal{A} \tag{3.10}$$

$$\frac{\lambda_{0\ell}}{\theta_{0\ell}} v_{0j\ell} + \sum_{a \in \mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} v_{aj\ell} = \rho \phi_{j\ell} \qquad \forall j \in \mathcal{N} \tag{3.11}$$

$$\phi_{j\ell} \leq \phi^j_{max} y_{j\ell} \qquad \forall j \in \mathcal{N} \tag{3.12}$$

$$\frac{\lambda_{0\ell} v_{0j\ell}(-\tilde{H}'_{0\ell}(0)) + \sum_{a \in \mathcal{A}} \lambda_{a\ell} v_{aj\ell}(-\tilde{H}'_a(0))}{\sum_{a \in \mathcal{A} \cup 0} \lambda_{a\ell} v_{aj\ell}} = -\tilde{H}'(0) \quad \forall j \in \mathcal{N} \tag{3.13}$$

$$\frac{\lambda_{0\ell} v_{0j\ell} \tilde{H}''_{0\ell}(0) + \sum_{a \in \mathcal{A}} \lambda_{a\ell} v_{aj\ell} \tilde{H}''_a(0)}{\sum_{a \in \mathcal{A} \cup 0} \lambda_{a\ell} v_{aj\ell}} = \tilde{H}''(0) \quad \forall j \in \mathcal{N} \tag{3.14}$$

$$\sum_{a\in\mathcal{A}} \beta_a^k x_{aj\ell} \leq b_j^k \qquad \forall j \in \mathcal{N}, \forall k \in \mathcal{K} \tag{3.15}$$

$$v_{aj\ell} \leq x_{aj\ell} \qquad \forall a \in \mathcal{A}, \forall j \in \mathcal{N} \tag{3.16}$$

$$v_{aj\ell} \geq 0, x_{aj\ell} \in \{0,1\}, y_{j\ell} \in \{0,1\}, \phi_{j\ell} \geq 0 \qquad \forall a \in \mathcal{A}, \forall j \in \mathcal{N}. \tag{3.17}$$

In the above formulation, the objective function (3.9) is total energy cost in a cycle, which consists of fixed cost and operating cost that we wish to minimize. We define the cost function for energy usage as a combination of fixed energy cost and operating cost using a model of power usage of typical servers adopted by [1]. For operating cost, we would like to note that modern CPUs can be operated at different speeds during runtime by employing DVFS, and recent studies Gandhi et al. [26], Kusic et al. [47], and Raghavendra et al. [48] reported that DVFS results in a linear power and frequency relationship without additional cost for ramp up/down processing speed as defined in our objective function (3.9). Constraint (3.10) ensures that all class a traffic is divided across various servers, and constraint (3.11) ensures that the traffic intensity for server j be ρ since the average work that arrives at server j is $\rho\phi_{j\ell}$ if the server runs at speed $\phi_{j\ell}$. Constraint (3.12) forces the server j's speed to be zero when it is off and limits its speed by its maximum value when on. Constraints (3.13) and (3.14) match the first and second moments, respectively, to stabilize the mean queue length as described in Section 3.2.3. Constraints (3.10), (3.11), (3.13), and (3.14) are derived from the homogeneity constraints defined in Section 3.2.2.2. Recall that we define constraint (3.4) with constant speed ϕ, but here we define constraint (3.11) to allow speed scaling with $\phi_{j\ell}$ for minimizing energy cost based on theoretical foundations described in the online supplement. In fact, Theorem 1 in the online supplement shows that the queue length distribution of time-varying $M_t/G_t/1$ queue is stochastically identical to that of time-homogeneous $M/G/1$ queue when we adjust processing speed (service rate) responding to time-varying arrival rate based on constraint (3.4). In addition, constraint (3.15) limits resource k to b_j^k across classes assigned to server j with requirements β_a^k for class a, and constraint (3.16) ensures that if class a is not assigned to server j, then no fraction of arriving requests are assigned to that server. Constraint (3.17) ensures non-negativity and binary nature of the decision variables. The decision variables $x_{aj\ell}$, $y_{j\ell}$, $\phi_{j\ell}$, and $v_{aj\ell}$ (for all $\ell \in \mathcal{T}$, $j \in \mathcal{N}$, and $a \in \mathcal{A}$) can be determined by solving the above MIP-ℓ problem separately for each time interval $\ell \in \mathcal{T}$. Although we solve the optimization problem to determine operational decisions independently for each time interval $\ell \in \mathcal{T}$ (e.g. powering server on/off, assignment, routing, and speed scaling), the aggregate workload distribution and the queue length distribution of every powered-on server would be stabilized across time intervals. In other words, based on our suggested approach, we can simplify a large scale, transient, non-stationary problem by decomposing it into individual smaller problems (i.e. decompose overall problem into 24 MIP-ℓ problems) in order to achieve time-stability. This is the key benefit of our suggested approach since we can solve smaller problems instead of large-scale MIP problems while stabilizing the queue length distribution across time intervals. Although we decompose the overall problem into individual MIP-ℓ problems, each of the MIP-ℓ problems is indeed NP-hard (we can simply show that a well-known NP-hard capacitated facility location problem (CFLP) can be reduced to our suggested MIP-ℓ problem), and it is difficult to solve the problem for large-size instances; therefore, we need to develop an efficient algorithm to solve the proposed problem. Note that in this study we choose to focus on introducing the key idea of our approach to achieve time-stability and analyze numerical results for the smaller instance, and developing an algorithm to solve the suggested MIP problem is beyond the scope of this study. Based on our suggested approach, we will study approaches to solve large-scale problems in the future.

3.4 Performance Guarantee

Through Sections 3.2 and 3.3, we introduce the notion of time-stability and suggest an approach to stabilize the queue length distribution of each powered-on server. Specifically, we derive the constraints and formulate an MIP problem to stabilize the mean queue lengths. Moreover, in this section, we will show that performance bound on waiting time can be obtained based on the stabilized queue length distribution. In general, users of data centers commonly require QoS guarantees in terms of waiting times (or response times), or for that matter SLA and its violation are also defined based on waiting times. Thus, performance bound on waiting time would be a preferred performance measure and much more useful to design, monitor, and control data center operations. Recall that, for energy efficiency, we apply speed scaling techniques and show that the queue length distribution would be stabilized when the processing speed of each powered-on server changes in proportion to arrival rates in the online supplement. In this case, even if the queue length distribution is stabilized, waiting time distribution would not be stabilized since the processing speed is varying across time intervals. However, based on the stabilized queue length distribution, we can derive the waiting time distribution and compute the mean waiting time of each application request for each time interval. For each time interval $\ell \in \mathcal{T}$, let $X_{a\ell}$ be the waiting time of class a in time interval ℓ with cumulative distribution function (CDF) $W_{a\ell}(\cdot)$, and $X_{j\ell}$ be the waiting time of applications which are served by server j in time interval ℓ with CDF $W_{j\ell}(\cdot)$. Then $X_{a\ell}$ can be defined as $X_{a\ell} = X_{j\ell}$ with probability $v_{aj\ell}$, and the LST of the waiting time distribution for class a in time interval ℓ, $\tilde{W}_{a\ell}(s)$ can be defined as

$$\tilde{W}_{a\ell}(s) = \sum_{j} v_{aj\ell} \tilde{W}_{j\ell}(s)$$

where $\tilde{W}_{j\ell}(s)$ is the LST of the waiting time distribution at server j in time interval ℓ. For the stabilized $M/G/1$ queue, the P-K transform formula for $\tilde{W}_{j\ell}(s)$ is defined as

$$\tilde{W}_{j\ell}(s) = \frac{(1-\rho)s}{s - \lambda + \lambda \tilde{S}_{j\ell}(s)}.$$

Note that aggregate arrival rates $\lambda_{j\ell}$ at server j in time interval ℓ would be $\lambda_{j\ell} = \sum_a v_{aj\ell}\lambda_{a\ell}$. In addition, for a random variable Y with a target workload distribution $H(\cdot)$ and processing speed of server j in time interval ℓ, $\phi_{j\ell}$, the LST of service time distribution of server j in time interval ℓ, $\tilde{S}_{j\ell}(s)$ can be defined as $\tilde{S}_{j\ell}(s) = \tilde{H}(s/\phi_{j\ell})$ (since service time is defined as $Y/\phi_{j\ell}$). Now we have the LST of waiting time of application a in time interval ℓ as

$$\tilde{W}_{a\ell}(s) = \sum_j v_{aj\ell}\tilde{W}_{j\ell}(s) = \sum_j v_{aj\ell} \frac{(1-\rho)s}{s - \sum_a v_{aj\ell}\lambda_{a\ell} + \sum_a v_{aj\ell}\lambda_{a\ell}\tilde{H}(\frac{s}{\phi_{j\ell}})}.$$

Moreover, the mean waiting time of class a in time interval ℓ as

$$\bar{X}_{a\ell} = \sum_j v_{aj\ell}\bar{X}_{j\ell} = \sum_j \left(\frac{\lambda_{j\ell}E[Z_{j\ell}^2]}{2(1-\rho)}\right) = \sum_j v_{aj\ell}\left(\frac{\sum_a v_{aj\ell}\lambda_{a\ell}}{2(1-\rho)}\frac{(1+C^2)(E[Y])^2}{\phi_{j\ell}^2}\right) \tag{3.18}$$

where C^2 is SCOV of the target workload. Based on the analysis of waiting times, it is intuitive to think that waiting times of applications can be controlled in a straightforward manner by adjusting processing speeds of powered-on servers. Further, if we do not consider speed scaling and enforce that processing speed of powered-on servers would be constant across time intervals such that $\phi_{j\ell} = \phi \quad \forall j \in \mathcal{N}, \ell \in \mathcal{T}$, then the waiting time distributions are also stabilized as well as the queue length across time intervals. In addition, for speed scaling, the mean waiting time of each application class would be bounded by the minimum and maximum processing speeds of assigned servers. Since we have stabilized aggregate workload distribution and queue length distribution for each powered-on server, an upper bound on the mean waiting times of each class a, τ_a (e.g. $\bar{X}_{a\ell} \leq \tau_a \quad \forall a \in \mathcal{A}, \forall \ell \in \mathcal{T}$) can be easily obtained based on the desired mean queue length $\bar{L}$ which is introduced in Section 3.2.3 as follows:

$$\tau_a = \frac{E[Y]\bar{L}}{\phi_a^{\min}} \quad \text{where } \phi_a^{\min} = \min\{\phi_{j\ell}, j \in \mathcal{N}_a^{\text{on}}, \ell \in \mathcal{T}\}. \tag{3.19}$$

In other words, the upper bound τ_a is derived by assuming the worst-case scenario reflecting the situation that class a workloads are solely routed to server j with the minimum speed ϕ_{min}^j among the powered-on servers. Consequently, an upper bound on the mean waiting times of the overall system across time intervals, τ can be defined as

$$\tau = \max\{\tau_a, a \in \mathcal{A}\}. \tag{3.20}$$

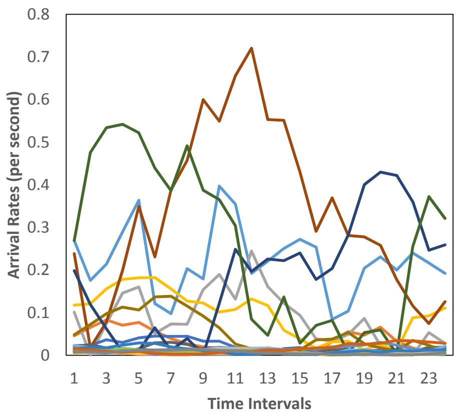
Based on the analysis of the waiting time, we define the constraints on the processing speed of each server j in time interval ℓ, $\phi_{j\ell}$, to define the upper bound on the mean waiting time as follows

$$r\phi_{\max}^j \leq \phi_j \qquad \forall j \in \mathcal{N}, \tag{3.21}$$

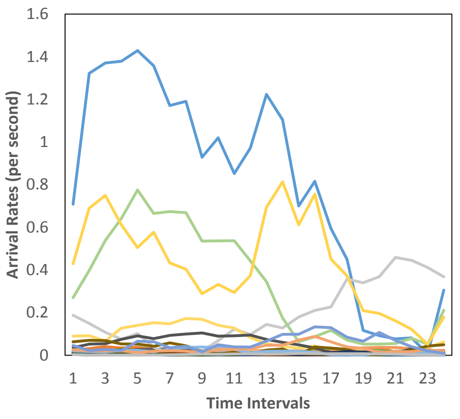
where r is the bound ratio of the minimum speed to the maximum speed of servers. We will solve our proposed MIP problem by adding constraint (3.21) and analyzing the results in Section 3.5.

3.5 Numerical Evaluation

In this section we introduce simulation results to show that our suggested approach stabilizes the mean queue length and we provide performance bound on the mean waiting time. For the numerical evaluation, we select two test instances; 10 servers with 20 appli-



(a) Arrival rates of 20 applications



(b) Arrival rates of 40 applications

Figure 3.2: Time-varying arrival rates across 24 1-hour time intervals

cations ($|\mathcal{N}| = 10$, $|\mathcal{A}| = 20$) and 20 servers with 40 applications ($|\mathcal{N}| = 20$, $|\mathcal{A}| = 40$) from [20]. Recall that, although data centers generally operate hundreds or thousands of servers, considering that in most data centers applications of a single client are clustered among servers for security and confidentiality, test instances of 10 or 20 servers would be large enough to show that our suggested approach can be applied successfully in practice. We model the arrival process of each application's request as a non-homogeneous Poisson process with time-varying arrival rates $\lambda_{a\ell}$ for 24 one-hour time intervals (i.e. 24 equally spaced time intervals for a one-day cycle) such that $\ell \in \mathcal{T} = \{1, 2, \ldots, 24\}$. In Figures 3.2a and 3.2b, we plot the arrival rates of both test instances (20 applications and 40 applications, respectively) across 24 one-hour time intervals. Also, for the workload distribution of each application a, $H_a(\cdot)$, we choose uniform distribution with mean workload $1/\theta_a$ and SCOV C_a^2. We also assume that each application has a requirement for three types of resources ($\mathcal{K} = \{1, 2, 3\}$) and that each server has a capacity limit for those resource types. For energy cost, we define fixed energy costs for powered-on server j as much higher than unit operation costs of server j, c_j, considering the fact that energy cost of servers is dominated by fixed costs as mentioned in [33]. In this study, we solved the suggested MIP problem by using CPLEX 12.6 Concert Technology on an Intel Core i7-3740QM 2.70GHz with 16GB memory, and we used Java to develop a discrete event simulation framework. We ran simulation experiments by using decision variables, $x_{aj\ell}$, $y_{j\ell}$, $\phi_{j\ell}$, and $v_{aj\ell}$, which are determined by solving the suggested MIP problem.

3.5.1 Computational Time

Before analyzing the simulation results, to describe the complexity of our suggested MIP problem, we summarize in Table 3.2 the objective function values and the computational time obtained by solving MIP-ℓ across time intervals $\ell \in \mathcal{T}$ for the two test instances. Note that we obtain the optimal objective function values for the instance of 10 servers with 20 applications within 3600 seconds; however, due to the complexity of the problem, we report the objective function values of the best feasible solution and MIP gap for 3600, 7200, and 10800 seconds time limits for the instance of 20 servers and 40 applications.

3.5.2 Analysis of Time-Stability

Firstly, we analyze the mean queue lengths of each powered-on server to check whether our suggested approach achieves time-stability. Recall that as described in Sections 3.2.3, our suggested approach can be utilized to stabilize the mean queue length by matching the first and second moments for the desired queue length $\bar{L}$. For the numerical evaluation, we select the desired mean queue length $\bar{L}$ and the first and second moments ($E(Y)$ and $E(Y^2)$, respectively) as $\bar{L} = 3$ with $E(Y) = 2.3$ and $E(Y^2) = 7.2737$ for 10 servers with 20 applications and $\bar{L} = 4$ with $E(Y) = 3.7$ and $E(Y^2) = 27.38$ for 20 servers with 40 applications. In both cases, we select the desired traffic intensity as $\rho = 0.8$. Figures 3.5a and 3.6a depict the mean queue length of powered-on servers across 24 time intervals of both test instances. Although there exist some variations, the mean queue lengths at

	10 Servers and 20 Classes		20 Servers and 40 Classes with 3600 Seconds Time Limit			20 Servers and 40 Classes with 7200 Seconds Time Limit			20 Servers and 40 Classes with 10800 Seconds Time Limit		
Time	Objective Function	CPU Time (seconds)	Objective Function	CPU Time (seconds)	Gap (%)	Objective Function	CPU Time (seconds)	Gap (%)	Objective Function	CPU Time (seconds)	Gap (%)
1	282.00	694.49	768.17	3603.86	13.38	760.33	7208.60	12.07	760.33	10825.67	12.01
2	255.00	148.59	969.50	3604.75	0.74	962.83	7201.29	0.05	962.33	7068.53	0.00
3	282.00	1534.94	1029.33	70.78	0.00	1029.33	70.78	0.00	1029.33	70.78	0.00
4	309.00	408.02	1037.50	3604.86	0.91	1033.17	7210.60	0.50	1033.17	10800.25	0.49
5	342.00	128.34	1099.67	100.06	0.00	1099.67	100.06	0.00	1099.67	100.06	0.00
6	280.00	1076.35	1029.33	1233.63	0.00	1029.33	1233.63	0.00	1029.33	1233.63	0.00
7	288.00	163.40	957.67	3603.56	6.51	952.33	7225.78	5.99	952.33	10863.99	5.99
8	315.00	2007.39	953.33	3610.75	6.08	953.33	7210.01	6.08	945.67	10831.73	5.32
9	327.00	843.70	833.67	3607.44	12.76	833.67	7206.97	12.58	825.00	10808.16	11.53
10	356.00	663.96	868.67	3603.47	10.65	874.00	7223.31	10.98	874.00	10815.67	10.98
11	356.00	2455.92	822.67	3606.51	12.17	835.17	7218.76	13.67	835.17	10817.75	13.67
12	345.00	91.56	828.83	3605.96	13.21	817.00	7219.87	11.75	817.00	10826.99	11.75
13	318.00	180.09	916.83	3609.30	2.35	902.50	4176.90	0.79	895.33	4176.90	0.00
14	310.00	245.81	853.17	3605.03	2.13	853.17	7204.36	2.13	853.17	10810.11	2.13
15	295.00	305.95	689.00	3602.81	8.75	689.00	7210.23	8.40	689.00	10816.66	8.31
16	256.00	196.19	757.00	3602.22	4.83	757.00	7218.20	4.76	754.17	10816.77	4.40
17	256.00	203.53	690.33	3608.10	11.70	688.00	7217.12	11.29	680.50	10869.92	10.20
18	260.00	121.98	647.17	3603.84	6.89	646.17	7203.58	6.24	639.50	10811.99	5.77
19	295.00	391.58	625.00	3607.68	11.81	622.67	7219.65	11.48	620.67	10864.83	11.20
20	273.00	170.18	614.50	3608.63	10.31	607.83	7209.07	9.32	607.83	10814.04	9.32
21	266.00	89.03	620.67	3605.70	10.87	620.67	7206.45	10.46	620.33	10809.39	9.93
22	263.00	142.93	613.50	3605.06	9.50	611.67	7209.62	9.02	610.17	10818.20	8.49
23	263.00	103.35	627.83	3613.53	15.24	627.83	7217.48	15.22	614.50	10839.07	13.38
24	255.00	214.84	671.00	3621.70	11.45	671.00	7226.45	11.29	670.17	10824.64	10.28

Table 3.2: Objective function value and CPU time: 10 servers and 20 applications and 20 servers and 40 applications

each powered-on server are approximately stabilized as desired across 24 time intervals as compared to time-varying arrival rates shown in Figures 3.2a and 3.2b. We would like to note that if we match more moments for the target workload distribution, then the mean queue lengths would be more stable across time intervals. Also, we compare the mean queue length resulted by using the decision variables determined by solving the modified MIP problem which uses the following constraint (3.22) instead of time-homogeneity constraints (3.11), (3.13), and (3.14),

$$\sum_{a \in \mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} v_{aj\ell} \leq \rho \phi_{j\ell} \qquad \forall j \in \mathcal{N}. \tag{3.22}$$

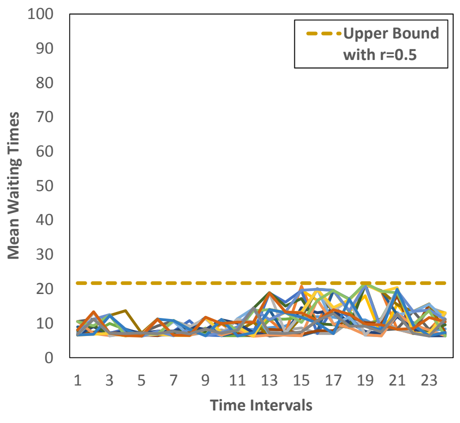
The above constraint (3.22) ensures that the average workload that arrives at every powered-on server must not exceed the desired traffic intensity ρ for fair comparison of our suggested approach. As shown in Figures 3.5b and 3.6b, the mean queue length would not be stabilized without using time-homogeneity constraints as compared to Figures 3.5a and 3.6a. Also, we compare the objective function values obtained by solving our suggested MIP problem with values obtained by solving the modified MIP problem to check how much additional energy cost is required for stabilizing the mean queue length, as well as provide performance guarantees on the mean waiting time. As we can see in Figures 3.7a and 3.7b, total energy cost for stabilizing the mean queue length is slightly higher, but it can be reasonably ignored since we can provide provable performance bounds.

3.5.3 Performance Bounds on The Mean Waiting Times

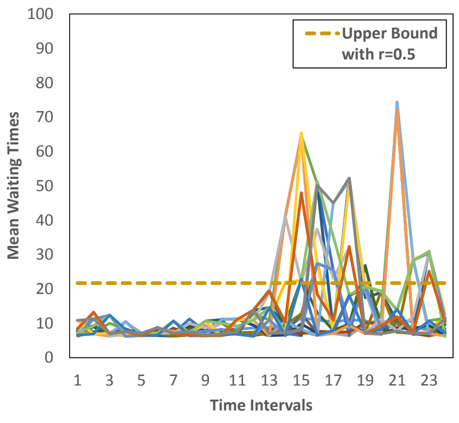
As mentioned in Section 3.4, performance bound on the mean waiting time of requests can be derived by using the bound ratio r of the minimum processing speed for each powered-on server based on constraint (3.21). Here, we select the bound ratio r as $r = 0.5$ for 10 servers with 20 applications and $r = 0.4$ for 20 servers with 40 applications, and then we contrast the mean waiting times obtained by using constraint (3.21) with the results obtained without using constraint (3.21) for both test instances. Figures 3.3a and 3.4a depict the mean waiting times and the performance bounds of both test instances, respectively. Note that the mean waiting time is strictly bounded by τ which can be derived as Equation (3.20) as opposed to the results obtained without using constraint (3.21) shown in Figures 3.3b and 3.4b.

3.5.4 Benchmark

In addition, we compare our approach against the algorithm proposed by Lin et al. [1] as a benchmark. Lin et al. [1] considered a data center optimization problem and proposed an off-line algorithm to determine the number of active servers for minimizing total energy cost, which consists of operating costs and switching costs. The main idea of their suggested approach is to minimize energy costs while satisfying the QoS constraints,

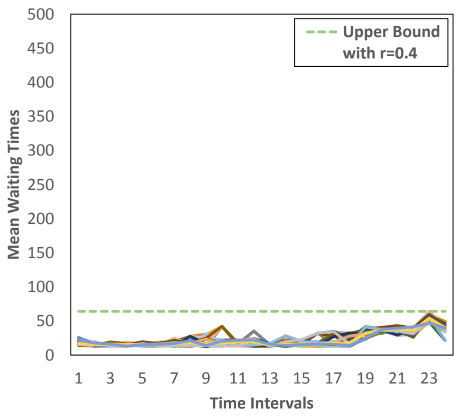


(a) With bound constraint with $r = 0.5$

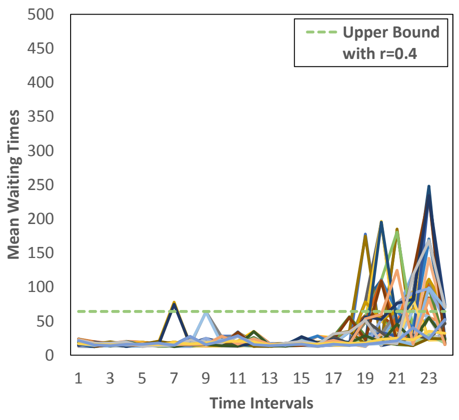


(b) Without bound constraint

Figure 3.3: Mean waiting times for the test instance of 10 servers with 20 applications



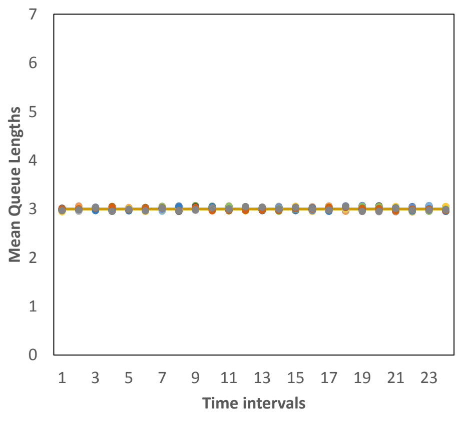
(a) With bound constraint with $r = 0.4$



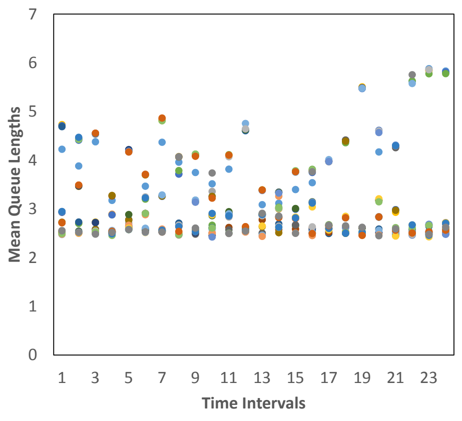
(b) Without bound constraint

Figure 3.4: Mean waiting times for the test instance of 20 servers with 40 applications

which are defined by using standard queueing theory results as shown in Van et al. [49] and Rao et al. [50]. Note that Lin et al. [1] implemented a dispatching rule which delivers equal amount of workload to each powered-on server (i.e. load balancing). Note that Lin et al. [1] suggested a model that uses quasi-steady-state approximations (i.e. the metrics are piecewise constant for each time interval long enough for the system to reach steady-

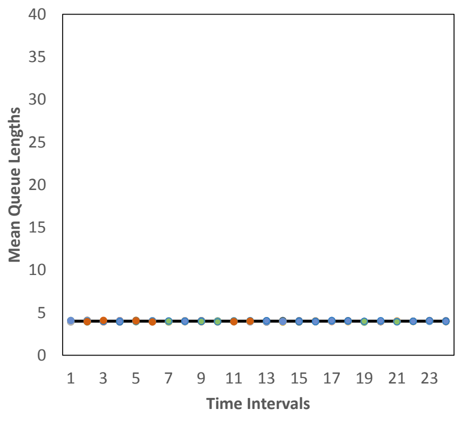


(a) With time-homogeneity constraints for $\bar{L} = 3$

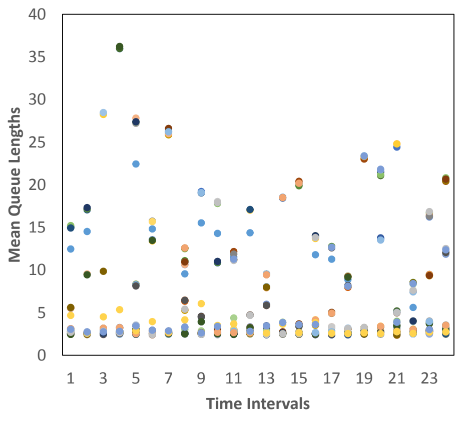


(b) Without time-homogeneity constraints

Figure 3.5: Mean queue lengths for the test instance of 10 servers with 20 applications



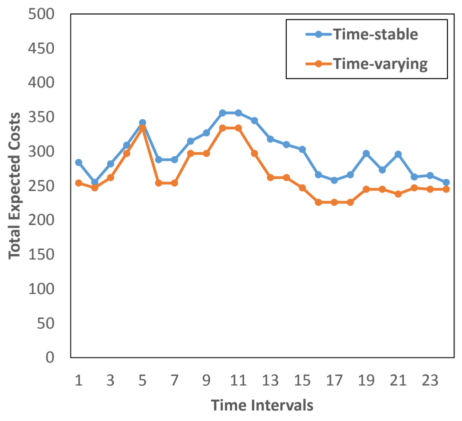
(a) With time-homogeneity constraints for $\bar{L} = 4$



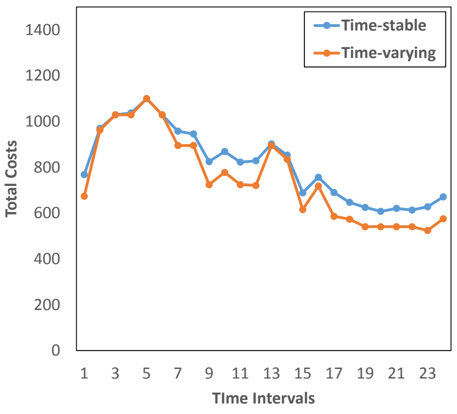
(b) Without time-homogeneity constraints

Figure 3.6: Mean queue lengths for the test instance of 20 servers with 40 applications

state) to define QoS constraints for non-stationary system, and it is worthwhile pointing out that our suggested approach can provide provable performance bound for time-varying and transient systems. Since Lin et al. [1] assumed that both workloads and servers are homogeneous and did not consider resource capacities and speed scaling, we slightly



(a) 10 servers with 20 applications



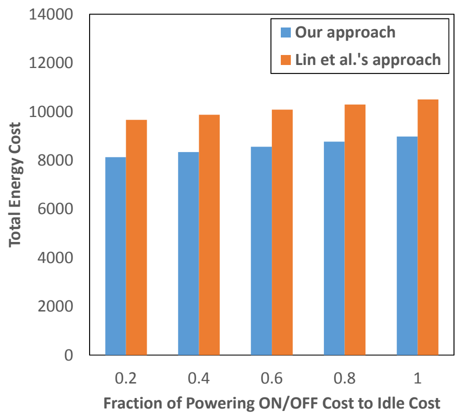
(b) 20 servers with 40 applications

Figure 3.7: Comparison of energy cost for two test instances: time-stable results vs time-varying results

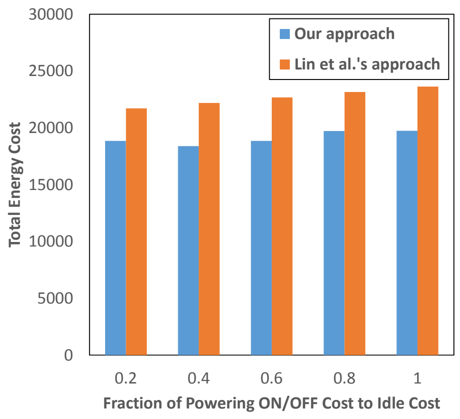
modified their model and ours, and compared the results. In fact, Lin et al. [1] proposed a QoS constraint that the average waiting time is bounded by certain thresholds based on $M/G/1$ processor sharing queue, and we re-defined their constraint by using multiclass $M/G/1$ queue with FCFS (Gautam [32]) for the heterogeneous workload. Also, since our suggested an MIP problem does not consider switching costs (i.e. cost for powering server on and off), we modified our model by adding the following constraints (3.23)-(3.24) and cost function (3.25) so that our MIP problem determines operational decisions while minimizing both operating costs and switching costs. In order to include cost for powering on/off, we define additional decision variables, $u^{\text{on}}_{j\ell}$ (e.g. $u^{\text{on}}_{j\ell} = 1$ if server j is turned on at the beginning of time interval ℓ) and $u^{\text{off}}_{j\ell}$ (e.g. $u^{\text{off}}_{j\ell} = 1$ if server j is turned off at the end of time interval $\ell - 1$). Then we can define the constraints to determine variables $u^{\text{on}}_{j\ell}$ and $u^{\text{off}}_{j\ell}$ as follows:

$$y_{j\ell} - y_{j(\ell-1)} \leq u^{\text{on}}_{j\ell} \quad \forall j \in \mathcal{N}, \forall \ell \in T \tag{3.23}$$

$$y_{j(\ell-1)} - y_{j\ell} + u^{\text{on}}_{j\ell} = u^{\text{off}}_{jt} \quad \forall j \in \mathcal{N}, \forall \ell \in T. \tag{3.24}$$



(a) 10 servers with 20 applications



(b) 20 servers with 40 applications

Figure 3.8: Comparison of total energy cost: our approach vs Lin et al.'s approach [1]

Recall that decision variable $y_{j\ell} = 1$ if server j is active in time interval ℓ; otherwise, $y_{j\ell} = 0$. By using $u^{\text{on}}_{j\ell}$ and $u^{\text{off}}_{j\ell}$, now we can define the cost function for turning servers on/off, which can be added to the objective function as follows:

$$\sum_{j\in\mathcal{N}}\sum_{\ell\in\mathcal{T}}(c^{\text{on}}_{j}u^{\text{on}}_{j\ell} + c^{\text{off}}_{j}u^{\text{off}}_{j\ell}), \tag{3.25}$$

where c^{on}_{j} and c^{off}_{j} are cost for turning servers on/off, respectively. To compare the approaches, we solved both problem test instances without considering resource capacity (as well as constraints (3.23)-(3.24)) and set cost for turning servers on/off so that the fractions of cost to power cost at "idle" state of each server j, f_j as 0.2, 0.4, 0.6, 0.8, and 1.0. For example, if the fraction is 0.2, then the cost for each time the servers turn on/off would be 20% of "idle" power cost. In the following Figures 3.8a and 3.8b, we compare total energy costs of our approach against that of Lin et al. [1] by setting thresholds for the mean waiting time as equal to the upper bound obtained by our suggested approach under the same traffic intensity. As shown in Figures 3.8a and 3.8b, the total energy cost of our suggested approach is smaller than that of the approach in Lin et al. [1] for the same upper bounds on the mean waiting time. This is because both the variability of the queue length and the waiting time get bigger for multiclass jobs (heterogeneous workloads), and thus more servers are needed to provide the same performance bounds can be derived by our suggested approach.

3.6 Concluding Remarks and Future Work

In this paper, we considered a fairly common scenario in cloud computing where requests of heterogeneous applications arrive at time-varying rates to a data center that consists of heterogeneous servers. For such a time-varying and heterogeneous system, it is difficult to achieve energy efficiency and provide performance guarantees simultaneously; therefore, to tackle this shortcoming, we suggested an approach to achieve time-stability. For performance guarantees, our suggested approach stabilizes (i) aggregate workload distribution based on moment matching approximation and (ii) arrival rates based on routing fractions to achieve time stable queue length distribution. We also derived time-homogeneity constraints based on our approach and formulated an MIP problem to optimally determine various decisions of sizing, assignments, routing, and speed scaling to minimize energy costs while considering time-stability. In fact, we can obtain time-stability of queue length distribution by matching sufficiently many moments based on the suggested approach. We showed that we can also obtain reasonable time-stability for the mean queue length by matching only the first and second moment. In addition, based on time-stable queue lengths distribution, the waiting times of applications are easily controlled by obtaining bounds on processing speeds. Our suggested approach indeed enables us to provide performance guarantee on the mean waiting times, which is an extremely useful measurement in terms of QoS for both users and service providers. To evaluate our approach, we developed a simulation model and summarized results that support our claim. In the present study we do not focus on developing an algorithm to efficiently solve the proposed MIP for the large-scale problem, we leave it for future work.

We acknowledge that our approach is purely based on historical information and further work needs to be done before being implemented in practice. We believe that results from this research can be used by practitioners to develop an initial cut for setting up servers in their data centers. Then an appropriate analysis tailored to the type of applications hosted by the data center would need to be performed to determine the number of stand-by servers to be positioned to handle unexpected surges in demand. Then, traffic can be monitored and significant deviations from time-stable queue lengths in real time can be used to trigger the use of stand-by servers. Further, one could develop control algorithms to dynamically change server settings on the fly and use our proposed approach as the baseline static setting. We propose to extend our study to develop an on-line algorithm for such real-time control problems in the future.

4. MEETING INELASTIC DEMAND IN SYSTEMS WITH STORAGE AND RENEWABLE SOURCES*

4.1 Introduction

Renewable generation capacity is expanding rapidly to potentially reduce carbon dioxide emissions and dependence on fossil fuels. Being a source of non-dispatchable generation, renewable energy introduces variability into the energy portfolio, and further amplifies the difficulty of matching demand with supply in real time. Energy storage is an environmentally friendly candidate that can provide flexibility to the system and mitigate the impact of volatile renewable generation.

The focus of this paper is on the operation of electric storages operated by the electricity consumers who own distributed renewable generation and face time-varying and stochastic electricity prices. Our motivation stems from the potential of electricity consumers to own and use storage devices (e.g., major consumers like data centers [51] and individual consumers who own PHEVs [52]), and from a recent study that shows consumer ownership of storage can be socially beneficial [53]. We also note that there is a growing trend for residential consumers and data centers to own distributed renewable generation [40, 54].

In this paper, we consider a *consumer* of electricity with *inelastic demand*, i.e., in each time period the consumer has to consume a certain (time-varying and possibly random) amount energy that is independent of the price of electricity. Part of the demand can be met by a renewable energy source (such as photo-voltaic (PV) solar panels) that is situated locally and owned by the consumer. Note that renewable power *supply* is time-varying and

*Reprinted with permission from “Meeting inelastic demand in systems with storage and renewable sources" by Soongeol Kwon, Yunjian Xu, and Natarajan Gautam, *IEEE Transactions on Smart Grid*, Accepted, Copyright © 2016, IEEE.

stochastic. Remaining demand (if any), beyond what the renewable source can supply, is satisfied either from the *grid* or by an in-house *Energy Storage Device* (ESD) or both. Like power demand and renewable supply, *price* for power from the grid is also time-varying and stochastic.

In the last few years this problem has received a lot of attention. There exists a substantial literature on the operation of energy storage owned by renewable generators or system operators. The joint scheduling of variable wind generation and energy storage systems is studied in order to maximize the joint profit of wind farms and energy storage systems, through a two-stage stochastic programming formulation [55], and a model predictive control (MPC)-based approach [56]. The authors of [57] derive an upper bound on the marginal value of storage (at small installed capacities) for a transmission-constrained power network. A few recent works study the optimal operation of energy storage devices with an objective of minimizing the mismatch between the available renewable generation and system load [58, 59, 60].

Another well studied application of energy storage is the use of storages to arbitrage [61, 52]. A few recent works conduct a dynamic programming approach to derive the arbitrage value of electric storage, in the presence of dynamic pricing [62, 63]. Different from the setting in the present paper, this aforementioned literature assumes that the operator of energy storages (e.g., an arbitrager) has zero demand for electricity and puts no value on its own electricity consumption.

There have been recent studies on the operation of consumer-owned ESDs. The authors of [64] study the day-ahead scheduling of energy storage by analyzing a noncooperative game among consumers. There is a growing literature that applies Lyapunov optimization based on-line algorithms on the operation of consumer-owned ESDs [65, 66, 67]. These on-line algorithms are shown to be asymptotically optimal, as the storage capacity increases to infinity. It is worth noting that these Lyapunov optimization based on-line algorithms may fail to achieve asymptotic optimality if the storage efficiency is less than 1, i.e., if the storage has non-negligible self-discharging [68]. Unlike the Markov decision process (MDP), which is computationally complex and requires substantial statistical information of the system dynamics, Lyapunov optimization based algorithms use simple linear programs to make storage operation decisions based only on the current system state (e.g., the current storage level). Our numerical results show that the algorithm proposed in [66] performs well when the storage capacity is significantly larger than the maximum charging/discharging rates, i.e., when it takes many hours to fully charge and discharge the storage.

Closely related to the present paper, a few recent papers establish structural properties on optimal storage operation policies in a variety of MDP settings that incorporate time-varying (and/or stochastic) cost and demand [69, 70, 71]. The main results of these theoretic works are the existence of an optimal policy that can be characterized by (time-varying and possibly state dependent) operational thresholds. The computation of these thresholds usually becomes intractable for practical settings with stochastic renewable generation and varying electricity prices, for example, in our MDP setting where time is explicitly incorporated into the system state, and each time period lasts for only 5 minutes (288 time periods per day).

We formulate the storage operation problem as an MDP with periodic cycles. The parameters of the MDP are trained using a set of real data on electricity prices, solar generation, consumer demand. The *main contribution* of this paper is to implement and numerically compare approaches ranging from Markovian models, to hybrid methods based on statistics and optimization, to those that are based on Lyapunov optimization and require no historical information, under a variety of parameter settings on the storage size, the level of solar generation, as well as the maximum charging/discharging rates.

We introduce and test two approximate dynamic programming (ADP) based heuristic policies, which usually yield the best performance among all tested heuristics. The first ADP-based policy, which is referred to as One-step Look-ahead Algorithm (OLA), chooses an action that minimizes the expected cost for the current and the next state. While OLA always treats the next stage as the terminal stage, the second ADP-based heuristic policy, which is referred to as One-step Roll-out algorithm (ORA), approximates the cost-to-go at every possible next system state by solving a deterministic (certainty-equivalent) optimization problem with future system stochasticity taking the expected value.

The insight we obtain from numerical experiments sheds some light on the effectiveness of different types of heuristic policies and the value of storage under various parameter settings. We summarize our key findings in the following.

1. Algorithms based on Lyapunov optimization (e.g., the one proposed in [66]) require minimum (almost negligible) computational efforts, and perform reasonably well when the storage capacity is significantly larger than the maximum charging/discharging rates. For fast-charging storage devices that can be fully charged within 2 hours, on the other hand, ADP-based algorithms (i.e., ORA and OLA) significantly outperform the one proposed in [66].

2. The value of storage (VoS, which is measured as the net benefit obtained by the consumer if she operates the storage according to an ADP algorithm) is much higher under 5-minute real-time pricing than that under hourly pricing, due the higher variability in cost in the former case. VoS increases sharply with the storage capacity only when the maximum charging/discharging rates grow in proportion to the storage capacity. In other words, the value of storage does not increase appreciably with increase in storage size, if the maximum charging/discharging rates remain fixed.

The rest of the paper is organized as follows. We describe our problem in Section 4.2. In Section 4.3 we develop a probabilistic model and suggest ADP-based approaches to solve it. We consider other heuristic algorithms in Section 4.4. We compare and discuss the performance of these algorithms in Section 4.5 by obtaining parameters using a real training data set and testing them. Finally, we make some brief concluding remarks in Section 4.6.

4.2 Problem Description

Before describing our model, we state a few key features of our problem setup. We consider inelastic demand, which must always be met instantaneously in real time and cannot be either postponed or cut back using incentives such as prices. We note that electricity consumption usually exhibits inelasticity in the short term [72], and that the setting of inelastic demand is used in many related works, e.g., [58, 69]. The ESD has a finite energy storage capacity. Price is non-negative and exogenous (not affected by the consumer's decisions). There are inefficiencies in charging and discharging, but no leakages (i.e., self-discharging) in the ESD. This assumption of 100% storage efficiency (no self-discharging) is reasonable since many popular types of modern batteries (e.g., Lead acid, Sodium Sulphur (NaS), Lithium ion, and Vanadium redox batteries) have negligible self-discharge ($0 - 5\%$ per month) [73], and further, the effective planning horizon for storage operation is usually no more than a week.

We now describe some notations used in this work (pictorially described in Fig. 4.1). We consider a discrete-time model where time periods are indexed by $t = 0, 1, \ldots$. The amount of energy in the ESD at the beginning of period t is denoted by U_t (in kWh). The *stochastic* uncontrollable variables are: D_t, the *demand* for energy in period t (in kWh); S_t, the energy *supply* from renewable source in period t (in kWh); C_t, the *cost* in period t for a unit of energy from grid (in \$/kWh). Table 4.1 summarizes the acronyms used in this paper.

There are constraints and inefficiencies in the ESD charging and discharging processes.

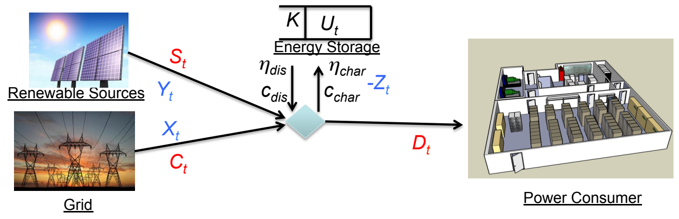


Figure 4.1: Schematic representation of scenario and notation

Acronym	Description
ESD	Energy Storage Device
MDP	Markov Decision Process
OLA	One-step Look-ahead Algorithm
ORA	One-step Roll-out Algorithm
TBA	Threshold-Based Approximation algorithm
NOA	Naive Opportunistic Algorithm
HWR	The on-line algorithm proposed in Huang, Walrand and Ramchandran [66]

Table 4.1: Summary of acronyms

A maximum of K kWh of energy can be stored in the ESD at any time. The ESD can be discharged and charged at a maximum rate of c_{dis} and c_{char} (in kW) respectively. Also, the ESD discharging and charging efficiencies are $\eta_{dis} \leq 1$ and $\eta_{char} \leq 1$ respectively (which we explain next). If ρ kWh of energy is used to charge the ESD in period t, then the increase in ESD level $U_{t+1} - U_t$ is $\rho\eta_{char}$ kWh. Likewise if ρ kWh of energy is needed from the ESD, then the increase in ESD level $U_{t+1} - U_t$ is $-\rho/\eta_{dis}$ kWh. Next we describe the *decision variables* under the control of the consumer. Let X_t, Y_t and Z_t be the energy drawn (in kWh) from the grid, renewable source and ESD respectively at period t. While $X_t \geq 0$ and $Y_t \geq 0$ for all t, Z_t can be positive or negative.

During every period t, given demand D_t, renewable supply S_t, cost C_t and ESD charge level U_t, we need to *determine* the supply from grid X_t, the draw from renewable source

Y_t, and contribution from the ESD Z_t so that the long-run expected total cost is minimized subject to satisfying demand, staying within ESD capacities and other constraints such as dynamics and non-negativity. As in [69, 70], we are interested in minimizing the total expected discounted cost. This sequential decision making problem can be formulated mathematically as follows,

$$\text{Minimize}_{(X_t,Y_t,Z_t)} \lim_{T\to\infty} \textstyle\sum_{t=0}^{T} \beta^t\, \mathbb{E}[C_t X_t] \tag{4.1}$$

$$\text{Subject to the following } \ \forall t \in \{0,1,2,\ldots\}, \tag{4.2}$$

$$X_t + Y_t + Z_t \geq D_t, \tag{4.3}$$

$$0 \leq Y_t \leq S_t, \quad \psi Z_t \leq c_{dis}, \tag{4.4}$$

$$-\psi \min\{Z_t, 0\} \leq c_{char}, \tag{4.5}$$

$$U_{t+1} - U_t = -\max\{Z_t, 0\}/\eta_{dis} - \eta_{char}\min\{Z_t, 0\}, \tag{4.6}$$

$$0 \leq U_t \leq K, \quad X_t \geq 0, \tag{4.7}$$

where $\beta \in (0,1)$ is a discount factor, and ψ is a constant for time-unit conversion, i.e. number of time units per hour (viz. since c_{char} is in kW and Z_t in kWh, if length of period t is 1 second then $\psi = 3600$). In constraint (4.3) we have implicitly assumed free disposal of renewable generation. It is optimal to use the renewable generation first to meet the demand, and then to charge the residual renewable generation to the ESD. If there is not enough storage capacity to absorb the residual renewable generation, then it is possible that $Y_t + Z_t > D_t$ (note that $X_t = 0$ and $Z_t \leq 0$ in this case). For all $t \geq 0$, C_t, D_t, and S_t are modeled as discrete random variables, and all constraints in the above optimization problem must hold for every trajectory of realized demand, renewable supply, and cost.

Remark 4. *We can reduce the above problem to a 1-dimensional control in X_t or Z_t by realizing that we can let $Y_t = S_t$ and $Z_t = D_t$ - X_t - Y_t, subject to the charging/discharging* *rate constraints [55, 65, 70]. However, for ease of presentation we will use all variables, not just X_t or Z_t.*

Remark 5. *It is shown in [65, 69, 71] that the optimal policy can be characterized by two thresholds. Given the system state at period t, namely t, D_t, S_t, C_t, and U_t, the optimal policy does the following: (i) if U_t lies between the two thresholds, do not charge or discharge the storage; (ii) if U_t lies below the lower threshold, greedily charge the storage up to this threshold; (iii) if U_t lies above the higher threshold, then discharge the storage to fulfil the demand. The threshold structure of optimal policies will provide some guidance on the design of heuristic policies (proposed in Sections 4.3.2 and 4.4). We finally note that when $\eta_{char} = \eta_{dis} = 1$ (no charging/discharging inefficiency), there exists a simpler optimal policy with a single threshold [63].*

4.3 MDP: Probabilistic Model with Cycles

In this section, we first introduce the way we fit real data into an MDP model, and then discuss approaches to solve it. Analyzing the data described in [74, 40] and the NREL labs, it is evident that demand, solar PV supply and cost are time-varying and stochastic. However, it is also not unreasonable to assume that there are daily or weekly effects. In other words, there is a deterministic variability as well as stochastic variability. To model such a phenomenon we consider what we call *probabilistic model with cycles*.

Definition 1. *An uncontrolled process $\{V_t\}_{t=1}^{\infty}$ is* cyclic *with cycle length N if the joint probabilistic distribution of $\{V_{\tau+\ell N}\}_{\tau=0}^{N-1}$ is identical for all $\ell \in \{1, 2, \ldots\}$, where N is the number of periods in a cycle. Each cycle lasts for T hours, and therefore has $N = \psi T$ periods.* □

Based on the above definition, we assume that $\{D_t\}_{t=1}^{\infty}$, $\{S_t\}_{t=1}^{\infty}$ and $\{C_t\}_{t=1}^{\infty}$ are *cyclic* with cycle length T (one cycle typically is the equivalent of one day). Further, we write down for all $t \in \{0, 1, \ldots\}$, with $n = (t \mod N)$,

$$D_t = d_n W_t^d + \delta_n, \quad S_t = s_n W_t^s, \quad C_t = c_n W_t^c + \gamma_n,$$

where $\{d_1, d_2, \ldots, d_N\}$, $\{\delta_1, \delta_2, \ldots, \delta_N\}$, $\{s_1, s_2, \ldots, s_N\}$, $\{c_1, c_2, \ldots, c_N\}$ and $\{\gamma_1, \gamma_2, \ldots, \gamma_N\}$, are sets of deterministic constants while $\{W_t^d\}_{t=1}^{\infty}$, $\{W_t^s\}_{t=1}^{\infty}$, and $\{W_t^c\}_{t=1}^{\infty}$, are stationary and independent *discrete time Markov chains* on state spaces $\mathcal{S}^d$, $\mathcal{S}^s$, and $\mathcal{S}^c$ and transition probability matrices $\mathcal{P}^d$, $\mathcal{P}^s$, and $\mathcal{P}^c$ respectively.

One can think of $\{s_1, s_2, \ldots, s_N\}$ as the power supplied by PV panels on a perfectly sunny day while $\mathcal{S}^s$ is a continuous set of values between 0 and 1. The *demand* and *cost* terms do not have such a nice interpretation and one would have to model them carefully based on data.

Note that since D_t, S_t and C_t are continuous, so are W_t^d, W_t^s and W_t^c. These random parameters W_t^d, W_t^s and W_t^c are assumed to be not correlated although D_t, S_t and C_t might themselves be correlated due to the correlation in their deterministic components. However, to model as an MDP, we need state spaces $\mathcal{S}^d$, $\mathcal{S}^s$, and $\mathcal{S}^c$ to be discrete. We discretize W_t^d, W_t^s and W_t^c using discrete random variables $\tilde{W}_t^d$, $\tilde{W}_t^s$ and $\tilde{W}_t^c$ each of which take $M+1$ different values. For example the aforementioned W_t^s would be mapped from a statespace $\mathcal{S}^s = [0, 1]$ to $\tilde{\mathcal{S}}^s = [0,\ 1/M,\ 2/M, \ldots, 1]$. Thus $\tilde{W}_t^d$, $\tilde{W}_t^s$ and $\tilde{W}_t^c$ would each be $M+1$ state discrete time Markov chains with transition probability matrices $\mathcal{P}^d$, $\mathcal{P}^s$, and $\mathcal{P}^c$ respectively. By some abuse of notations, for the rest of this paper we let D_t, S_t, and C_t denote the corresponding discretized values of demand, renewable generation, and cost at stage t.

In Section 4.5, we will use training data to estimate d_n, δ_n, s_n, c_n and γ_n for all $n \in \{1, \ldots, N\}$ as well as $\mathcal{P}^d$, $\mathcal{P}^s$, and $\mathcal{P}^c$. However, for the rest of this section we take a probabilistic approach assuming all the aforementioned parameters are known and formulate the system as an MDP.

We denote the **system state** at time t as a 5-tuple

$$\mathbf{x} = \{\{t/N\} + 1, W_t^d, W_t^s, W_t^c, U_t\}, \tag{4.8}$$

(where $\{t/N\}$ denotes $(t \mod N)$) with state space $\mathcal{S}$ given by the cartesian product

$$\{1, 2, \ldots, N\} \times \tilde{\mathcal{S}}^d \times \tilde{\mathcal{S}}^s \times \tilde{\mathcal{S}}^c \times \tilde{\mathcal{S}}^u,$$

where $\tilde{\mathcal{S}}^u$ is the discrete set of values between 0 and K that U_t can take. We note that the constructed MDP is *stationary*, because the time dependency and correlations of consumer demand and renewable generation are incorporated by including in the system state a periodic Markov chain that describes time evolution.

Let the **action** at time t denote the amount of power to be supplied from the grid, i.e. X_t, with action space $\mathcal{A}(\mathbf{x})$ corresponding to the set of all possible real numbers that lie in the following interval

$$\begin{aligned}\mathcal{A}(\mathbf{x}) = \Big[& \big(D_{\{t/N\}} - S_{\{t/N\}} - \min\{\eta_{dis} U_{\{t/N\}}, c_{dis}/\psi\}\big)^+, \\ & \Big(D_{\{t/N\}} - S_{\{t/N\}} + \min\Big\{\tfrac{K - U_{\{t/N\}}}{\eta_{char}}, c_{char}/\psi\Big\}\Big)^+ \Big],\end{aligned} \tag{4.9}$$

where $(\cdot)^+ = \max\{\cdot, 0\}$. Here, the lower bound of the action space is the amount of energy needed from the grid to fulfil the demand, when the storage is greedily discharged for consumption, and the upper bound is the amount of energy needed to meet the demand and to greedily charge the storage.

As noted in Remark 4, the energy drawn from renewable source and ESD at period t is determined by X_t, i.e., $Y_t = S_t$ and $Z_t = D_t$ - X_t - Y_t.

Given any $\mathbf{x} \in \mathcal{S}$, $X_t \in \mathcal{A}(\mathbf{x})$, and $\mathbf{y} \in \mathcal{S}$, we can compute the **transition probability**

$P_{\mathbf{xy}}(X_t)$ using appropriate Kronecker products of $\mathcal{P}^d$, $\mathcal{P}^s$, $\mathcal{P}^c$ and other matrices of zeros and ones (which are not explained due to space constraints). The next-stage storage level is given by

$$U_{t+1} = U_t - \max\{Z_t, 0\}/\eta_{dis} - \eta_{char} \min\{Z_t, 0\}, \tag{4.10}$$

where $Z_t = D_t - S_t - X_t$.

The **stage cost** at time t is the product of the corresponding power cost in state i times $X_t \in \mathcal{A}(\mathbf{x})$, $C_t X_t$.

By incorporating the time element into the state of the dynamic program, we have indeed formulated a "stationary" MDP (the quotes are because the state transition is stationary from one cycle of N values to the next cycle, but not within a cycle). For a given stationary policy which maps every possible system state $\mathbf{x}$ to a point in the action space $\mathcal{A}(\mathbf{x})$, the long-run discounted total cost corresponds to the objective function of our optimization problem defined in Section 4.2 (and also results in a feasible solution).

4.3.1 Exact MDP Solution

Note that the above MDP has a finite state-space and a finite action-space. There are many methods to obtain the optimal action $a \in \mathcal{A}(\mathbf{x})$ at state $\mathbf{x}$ for all $\mathbf{x} \in \mathcal{S}$. We consider a linear program (LP) based method. The following LP solves the optimal cost-to-go at each state $\mathbf{x}$, $\{J^*_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{S}}$ [75]:

$$\begin{aligned} &\mathrm{Max}_{\{J_{\mathbf{x}}\}} \quad \textstyle\sum_{\mathbf{x}\in\mathcal{S}} c_{\mathbf{x}} J_{\mathbf{x}} \\ &\text{s.t.} \quad g_{\mathbf{x}}(a) + \beta \sum_{\mathbf{y}\in\mathcal{S}} P_{\mathbf{xy}}(a) J_{\mathbf{y}} \geq J_{\mathbf{x}}, \;\; \forall \mathbf{x} \in \mathcal{S},\; \forall a \in \mathcal{A}(\mathbf{x}), \end{aligned} \tag{4.11}$$

where $\{c_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{S}}$ is a given vector with positive components, and $g_{\mathbf{x}}(a)$ is the stage cost at state $\mathbf{x}$ under action a. The optimal action to be taken at each state $\mathbf{x}$ can then be obtained by the following Bellman equation:

$$a^* \in \arg \min_{a \in \mathcal{A}(\mathbf{x})} \left\{ g_{\mathbf{x}}(a) + \beta \sum\nolimits_{\mathbf{y} \in \mathcal{S}} P_{\mathbf{x}\mathbf{y}}(a) J^*_{\mathbf{y}} \right\}. \tag{4.12}$$

In this manner it is possible to determine the optimal action in each state (in theory).

In summary, the MDP algorithm works as follows: given the demand, renewable generation, cost, and storage level at stage t, we first obtain the discretized values D_t, S_t, C_t and U_t. Using those we compute the optimal action Z_t prescribed by the MDP. Then we obtain the actions $Y_t = S_t$ and $X_t = \max(D_t - S_t - Z_t, 0)$.

We note that an optimal solution to the LP formulated in (4.11) must exist. However in practice, one could encounter difficulties known as the *curse of dimensionality*. The exact MDP can be solved (especially by packages such as MATLAB) only when the action space is small, $\eta_{char} = \eta_{dis} = 1$, and C_t, D_t and S_t belong to a small discrete set for all $t \in \{1, \ldots, N\}$. In the next subsection, we adopt a common procedure usually referred to as **Approximate Dynamic Programming (ADP)** to deal with the curse of dimensionality.

4.3.2 Approximate MDP Solution

In this section, we introduce two simple ADP-based algorithms that will be tested against other heuristics as well as the exact MDP solution in Section 4.5. An effective way to reduce the computation required by a dynamic program is to truncate the time horizon and at each stage make a decision based on look-ahead of a small number of stages [76, 77]. In particular, we will focus on the simplest ADP algorithms that look only a single stage ahead. Numerical results in Section 4.5 demonstrate that even these simplest ADP algorithms usually (significantly) outperforms Lyapunov optimization based algorithms. It is worth noting that, however, even these simplest ADP algorithms require much more computation than Lyapunov optimization based algorithms.

According to the simplest ADP-based policy, which is referred to as **One-Step Look-ahead algorithm** (OLA) in this paper, given the current state we determine the best action so that the expected cost for this state and the next state is minimized. Formally, given the current system state $\mathbf{x}$, the algorithm chooses an action $X_t \in \mathcal{A}_t$ that minimizes the following cost:

$$C_t X_t + \sum_{\mathbf{y}} \beta P_{\mathbf{xy}}(X_t) \cdot C_{t+1} \cdot (D_{t+1} - S_{t+1} - \eta_{dis} U_{t+1})^+ , \tag{4.13}$$

where $(\cdot)^+ = \max\{\cdot, 0\}$, and $\mathbf{y}$ is the system state at time $t+1$ that includes the parameters C_{t+1}, D_{t+1}, S_{t+1} and U_{t+1}. In this myopic version of one-step look-ahead policy, the stage $t+1$ is treated as the terminal stage, and therefore the policy fully discharges the storage to fulfill the demand at stage $t+1$. This would result in a stage cost at time $t+1$ of $C_{t+1} \left(D_{t+1} - S_{t+1} - \eta_{dis} U_{t+1}\right)^+$.

A natural way to improve OLA is to replace the myopic stage cost at state $\mathbf{y}$ by some approximated cost-to-go at this state. Formally, given the current system state $\mathbf{x}$, a **One-step Roll-out algorithm (ORA)** chooses an action $X_t \in \mathcal{A}_t$ that minimizes the following cost:

$$C_t X_t + \sum\nolimits_{\mathbf{y} \in \mathcal{S}} \beta P_{\mathbf{xy}}(X_t) \cdot \tilde{J}_{\mathbf{y}}, \tag{4.14}$$

where $\tilde{J}_{\mathbf{y}}$ is an approximation of the cost-to-go at system state $\mathbf{y}$, $J_{\mathbf{y}}$. We note that if the approximation is exact, i.e., if $\tilde{J}_{\mathbf{y}} = J_{\mathbf{y}}$, then the above Bellman recursion must yield the optimal action at the current system state $\mathbf{x}$.

The OLA simply treats stage $t+1$ as the terminal stage and let $\tilde{J}_{\mathbf{y}}$ be the stage cost at time $t+1$ in state $\mathbf{y}$. The ORA, on the other hand, solves a certainty equivalent optimization problem to approximate the cost-to-go at possible next-stage system states, where all random variables take the expected value, given that the system state at time $t+1$ is $\mathbf{y}$:

$$\tilde{J}_{\mathbf{y}} = \text{ Minimize}_{\{\bar{X}_\tau,\bar{Y}_\tau,\bar{Z}_\tau,\bar{U}_\tau\}} \sum_{\tau=t+1}^{t+N} \beta^{\tau-t-1}\mathbb{E}[C_\tau \mid \mathbf{y}] \cdot \bar{X}_\tau$$

$$\text{Subject to } \forall\tau \in \{t+1, t+2, \ldots, t+N\},$$

$$\bar{X}_\tau + \bar{Y}_\tau + \bar{Z}_\tau \geq \tag{4.15}$$

$mathbbE[D_\tau \mid \mathbf{y}],$

$$0 \leq \bar{Y}_\tau \leq \mathbb{E}[S_\tau \mid \mathbf{y}], \quad \psi\bar{Z}_\tau \leq c_{dis}, \tag{4.16}$$

$$-\psi \min\{\bar{Z}_\tau, 0\} \leq c_{char},$$

$$\bar{U}_{\tau+1} - \bar{U}_\tau = -\max\{\bar{Z}_\tau, 0\}/\eta_{dis} - \eta_{char}\min\{\bar{Z}_\tau, 0\},$$

$$0 \leq \bar{U}_\tau \leq K, \quad \bar{X}_\tau \geq 0,$$

where $\bar{U}_{t+1}$ is determined by the system state $\mathbf{y}$, $\mathbb{E}[\cdot\,|\mathbf{y}]$ denotes conditional expectation, and the minimization is taken over the variables $\bar{X}_\tau$, $\bar{Y}_\tau$, and $\bar{Z}_\tau$.

Certainty equivalent control is a simple and intuitive way to make sequential decisions under uncertainty. It is shown to be optimal for Linear-Quadratic-Gaussian (LQG) problems [78]. Certainty equivalence is the logic underlying the current practice of unit commitment (which is a deterministic optimization problem with random demand and renewable generation taking the expected value), and is proposed for the economic/environmental dispatch of power systems with intermittent renewable generation [79]. The certainty equivalent approximation results in significant computational savings by avoiding computing the exact cost-to-go at stage $t+1$, and on the other hand, makes the ORA suboptimal.

Since the system state space is continuous (due to the continuous storage level), given the current system state $\mathbf{x}$, it is impossible to evaluate the cost-to-go of every possible state $\mathbf{y}$ at stage $t+1$ (by solving the certainty equivalence optimization problem in (4.16)). In our simulation, we therefore discretize the space of demand, renewable supply, and cost at stage $t+1$, and restrict our attention to a finite set of actions X_t. Formally, given the current system state $\mathbf{x}$, ORA chooses an action X_t from a finite set of actions $\bar{\mathcal{A}}(\mathbf{x}) \in \mathcal{A}(\mathbf{x})$, in order to minimize

$$C_t X_t + \sum\nolimits_{\mathbf{y}\in\bar{\mathcal{S}}(\mathbf{x})} \frac{\beta P_{\mathbf{xy}}(X_t)}{\sum\nolimits_{\mathbf{y}\in\bar{\mathcal{S}}(\mathbf{x})} P_{\mathbf{xy}}(X_t)} \tilde{J}_{\mathbf{y}}, \tag{4.17}$$

where $\bar{\mathcal{S}}(\mathbf{x})$ is a finite subset of $\mathcal{S}$ that includes the discretized states of demand, renewable supply, and cost, as well as a finite set of storage levels U_{t+1} resulting from the finite set of actions in $\bar{\mathcal{A}}(\mathbf{x})$ (according to Eq. (4.10)). In our simulation, the discretized states (of demand, renewable supply, and cost) are uniformly distributed in a compact set that is estimated from real data. The set of actions $\bar{\mathcal{A}}(\mathbf{x})$ explored by ORA at state $\mathbf{x}$ includes the following: (i) charge the battery only if there is surplus in renewable generation, i.e., $X_t = (D_t - S_t)^+$, (ii) greedily discharge the battery to meet the demand, i.e., $Z_t = \min\left\{c_{dis}/\psi, \eta_{dis}U_t, (D_t - S_t)^+\right\}$, (iii) greedily charge the battery, i.e., $Z_t = -\min\left\{c_{char}/\psi, (K - U_t)^+ / \eta_{char}\right\}$, and several additional actions that are uniformly distributed between actions (ii) and (iii).

It is worth noting that unlike MDP, ORA does not discretize the storage level *in priori*. Under ORA, the (finite) set of next-stage storage levels in $\bar{\mathcal{S}}(\mathbf{x})$ is determined by the set of actions $\bar{\mathcal{A}}(\mathbf{x})$ as well as the current states U_t, D_t, and S_t.

4.4 Other Heuristics

As a one-step rollout policy on top of certainty equivalent control, ORA uses the Markovian structure of our model (via the state transition probability from stage t to $t+1$). To assess how our key algorithm ORA performs with real data, we compare it against other certainty equivalence based algorithms that use neither the discrete framework nor the Markovian structure. For that, in this section we leverage upon existing approaches to develop two heuristic policies: TBA (threshold-based approximation) algorithm and NOA (naive opportunistic algorithm).

For both the heuristic TBA and NOA algorithms, we first solve the following (unconditioned) certainty equivalence (UCE) problem to obtain the variables $\hat{X}_\tau$, $\hat{Y}_\tau$, $\hat{Z}_\tau$, and $\hat{U}_\tau$, for $\tau = 1, 2, \ldots, N$.

$$\text{UCE: Minimize}_{\{\hat{X}_\tau, \hat{Y}_\tau, \hat{Z}_\tau, \hat{U}_\tau\}} \sum_{\tau=1}^{N} \beta^{\tau-1}\, \mathbb{E}[C_\tau]\hat{X}_\tau,$$

$$\text{Subject to } \forall\tau \in \{1, \ldots, N\},$$

$$\hat{X}_\tau + \hat{Y}_\tau + \hat{Z}_\tau \geq \mathbb{E}[D_\tau],$$

$$0 \leq \hat{Y}_\tau \leq \tag{4.18}$$

$$mathbbE[S_\tau], \tag{4.19}$$

$$\psi \hat{Z}_\tau \leq c_{dis},$$

$$-\psi \min\{\hat{Z}_\tau, 0\} \leq c_{char},$$

$$\hat{U}_{\tau+1} - \hat{U}_\tau = -\max\{\hat{Z}_\tau, 0\}/\eta_{dis} - \eta_{char} \min\{\hat{Z}_\tau, 0\},$$

$$0 \leq \hat{U}_\tau \leq K,$$

$$\hat{X}_\tau \geq 0,$$

where $\hat{U}_{N+1} = \hat{U}_1$.

Heuristic TBA: Given the state at time t, namely, ($\{t/N\}$, D_t, S_t, C_t, U_t), we determine Z_t so that at time $t+1$, U_{t+1} is as close to $\hat{U}_{\{t/N\}+1}$ by appropriately charging or discharging. The goal is to reach threshold level $\hat{U}_{\{t/N\}+1}$ in the next time. Thus TBA is as follows (with $n = \{t/N\}$):
if $U_t < \hat{U}_{n+1}$, $Z_t = -\min\left\{\dfrac{\hat{U}_{n+1} - U_t}{\eta_{char}}, c_{char}/\psi\right\}$
else if $U_t = \hat{U}_{n+1}$, let $Z_t = 0$,
else $Z_t = \min\left\{\eta_{dis}(U_t - \hat{U}_{n+1}), c_{dis}/\psi\right\}$. Then, $X_t = \max\{D_t - S_t - Z_t, 0\}$ and

$Y_t = S_t$.

Heuristic NOA: Given the state at time t, namely, ($\{t/N\}$, D_t, S_t, C_t, U_t), we adopt a naive (but intuitive) strategy – if C_t is cheap, charge the ESD as much as possible; and if D_t is much higher than S_t, discharge as much as possible; otherwise do what the certainty equivalence model suggests. For that we use $\mathbb{E}[C_N]$ as the grand cost (computed over entire cycle N), and $Var[C_N]$ the corresponding grand variance; also ϕ_c and ϕ are parameters to be tuned. We note that both parameters ϕ_c and ϕ are non-negative and bounded from the above. Since NOA is computationally simple, one can use the training data to test the performance of NOA under a finite set of feasible parameters ϕ_c and ϕ. It leads to the following NOA (with $n = \{t/N\}$):

if $C_t < \mathbb{E}[C_N] - \phi_c\sqrt{Var[C_N]}$, then

$Z_t = -\min\{(K - U_t)/\eta_{char}, c_{char}/\psi\}$ otherwise,

if $D_t - S_t > \mathbb{E}[D_n] - \mathbb{E}[S_n] + \phi\sqrt{Var[D_n] + Var[S_n]}$,

$Z_t = \min\left\{D_t - S_t - \hat{X}_n, U_t\eta_{dis}, c_{dis}/\psi\right\}$

else

(i.e. $D_t - S_t < \mathbb{E}[D_n] - \mathbb{E}[S_n] + \phi\sqrt{Var[D_n] + Var[S_n]}$)

$Z_t = \min\left\{\max(S_t - D_t, \hat{Z}_n), c_{char}/\psi, (K - U_t)/\eta_{char}\right\}$. Then $X_t = \max\{D_t - S_t - Z_t, 0\}$ and $Y_t = S_t$.

Heuristic HWR: Before ending this section, we revisit the algorithm proposed in Huang, Walrand and Ramchandran [66], which is referred to as **HWR** in this paper. The algorithm will be numerically tested in the next section. It is a remarkable online algorithm that is based on Lyapunov optimization. The algorithm proposed in [66] does not use any historical information and makes (myopic) decisions based only on current state information (such as D_t, S_t, C_t and U_t).

We now briefly outline the algorithm HWR. At each stage $t = 0, 1, \ldots$, the algorithm solves the following LP:

$$\begin{aligned}
\text{Min}_{(\alpha_t^H,\beta_t^H,\gamma_t^H,\delta_t^H)} \qquad & H_t^c\beta_t^H - H_t^s\gamma_t^H + H_t^r\delta_t^H \\
\text{Subject to} \qquad & \beta_t^H + \delta_t^H \le c_{char}, \\
& \gamma_t^H \le c_{dis}, \\
& \delta_t^H \le \max(S_t - D_t, 0), \\
& \alpha_t^H + \gamma_t^H = \max(D_t - S_t, 0), \\
& \alpha_t^H \ge 0,\ \ \beta_t^H \ge 0,\ \ \gamma_t^H \ge 0,\ \ \delta_t^H \ge 0,
\end{aligned} \tag{4.20}$$

where $H_t^c = \eta_{char}(U_t - \theta) + C_t/\epsilon$, $H_t^s = (U_t - \theta)/\eta_{dis} + C_t/\epsilon$, and $H_t^r = (U_t - \theta)/\eta_{dis}$, with $\theta = K - \eta_{char}c_{char}/\psi$, and

$$\epsilon = \sup_{u\ge 0} C_u / \left[\eta_{char}(\theta - \min(\sup_{u\ge 0} D_u, c_{dis})/(\eta_{dis}\psi))\right].$$

The parameters (H_t^c, H_t^s, H_t^r) are designed to approximate the total operational cost. The relation between the decision variables of HWR and those used in this paper is

$$X_t = \alpha_t^H + \beta_t^H, \quad Z_t = \gamma_t^H - \beta_t^H - \delta_t^H.$$

For $t+1$ the algorithm updates the storage level as follows:

$$\psi U_{t+1} = \psi U_t - \gamma_t^H/\eta_{dis} + \eta_{char}(\beta_t^H + \delta_t^H).$$

It is shown in [66] that HWR achieves asymptotic optimality as the capacity of ESD K grows to infinity (when θ is big and ϵ is small). In the next section we will numerically compare the performance of HWR against the MDP and other heuristic policies in a setting with finite storage capacity.

4.5 Numerical Experimentation and Results

In Section 4.5.1, we compare the performance and computational time of heuristic algorithms against MDP, under different sizes of discretized state spaces (for demand, solar generation, and cost). Our numerical results show that compared to MDP, the ORA algorithm requires much less computational time and only slightly increases the total cost (by less than 2%). Motivated by the excellent performance of ORA, in Section 4.5.2 we benchmark the performance of OLA, HWR, TBA and NOA against ORA under a variety of parameter settings on storage capacity, charging/discharging rates, and average solar generation. In Section 4.5.3, we numerically explore the value of storage and solar generation under the same set of parameter settings considered in Section 4.5.2.

Before representing the numerical results, we would like to briefly discuss the data we obtained and the way we train our algorithms. Our main purpose was to get a representative sample that adequately captures the deterministic and stochastic variability over time. All algorithms are implemented in Matlab R2014a on an Intel Core i7-3740 2.70GHz PC with 16GB memory.

For 5-minute granularity we obtained 26 days of demand, solar generation and cost data in a single month. In that spirit we collected solar PV supply data from NREL (http://www.nrel.gov/midc/), 5-minute electricity prices from New England ISO (http://iso-ne.org/), and demand data from households (http://www.doc.ic.ac.uk/ dk3810/data/). Note that for 5-minute granularity we have $\psi = 12$. We used 16 days of collected data to train the model, i.e., estimate/fit parameters in the MDP model (described in Section 4.3) and the NOA algorithm (described in Section 4.4). We then use real data in the 10 remaining days (from the original 26 days) for testing. The length of the truncated horizons used in our simulation is long enough to evaluate the steady-state performance of the tested heuristic policies, with a discount factor $\beta = 0.99$.

For 1-hour granularity we obtained 5 years of 3 months' data of demand, solar generation and cost in June, July, and August of 2010-2014. We collected demand and (day-ahead hourly) price data from PJM (http://www.pjm.com/markets-and-operations/energy.aspx). For solar generation, we collected measurements of solar irradiance under the coverage of PJM, for the same 15-month period (http://www.nrel.gov/midc/bsc/). For 1-hour granularity we have $\psi = 1$. We used the first 12 months of collected data to train the model, and use real data in the 3 remaining months for testing.

To estimate d_t and δ_t for any $t \in [1, N]$, we use $D(1,t)$, $D(2,t)$, $\ldots$, $D(16,t)$, the realized demands in 16 days, to compute $\delta_t = \min_i[D(i,t)]$ and $d_t = \max_i[D(i,t)] - \delta_t$. Likewise for c_t and γ_t. In case of supply s_t, the minimum value is zero. Then for the DTMCs $\{\tilde{W}_t^d\}_{t=1}^{\infty}$, $\{\tilde{W}_t^s\}_{t=1}^{\infty}$, and $\{\tilde{W}_t^c\}_{t=1}^{\infty}$, we first select the number of states $M+1$. The state space of these three DTMCs is a set of discrete values 0, $1/(M-1)$, $2/(M-1)$, $\ldots$, 1. Then we estimate the elements of $\mathcal{P}^d$, $\mathcal{P}^s$, and $\mathcal{P}^c$ as the respective frequency of transition based on the 16 days' data (for the 5-min case) and the 12 months' data (for the 1-hour case). For all the 1-hour test cases, we use the weekdays' and the weekends' data (of demand and cost) to train two different transition matrices for weekdays and weekends, respectively. The objective is to capture the weekly fluctuation of demand and electricity prices through the constructed MDP model. For all the 1-hour test cases, each algorithm (MDP, ORA, OLA, HWR, TBA, NOA) makes storage operation decisions based on the corresponding transition matrix (of demand and cost) in weekdays and weekends.

For MDP, we consider 13 discretized actions that are uniformly distributed in the continuous action space expressed in Eq. (4.9). Given the current system state, ORA considers a finite number of possible next system states resulting from a set of 7 actions (note that the choice of these 7 actions depends on the current system state $U_t,\ D_t$ and S_t (cf. the discussion at the end of Section 4.3)), and chooses the action that minimizes the approximated cost-to-go. For the data set we use, increasing the number of explored actions (beyond 7) leads to negligible improvement in the performance of ORA. The OLA algorithm takes into account the same set of 7 actions as the ORA algorithm. The other three heuristic polices (HWR, TBA, and NOA), on the other hand, do not discretize the action space. For all the numerical experiments we use $c_{char} = c_{dis}$ and $\eta = \eta_{char} = \eta_{dis}$.

4.5.1 Benchmarking Heuristics against MDP

In this subsection, we will 1) compare the performance of MDP and ORA under the daily model (where a single transition matrix is trained using real data) and the weekly model (where two different transition matrices are trained using weekday's and weekends' data, respectively), and 2) benchmark the performance and computational time of heuristic algorithms against MDP under different sizes of discretized states.

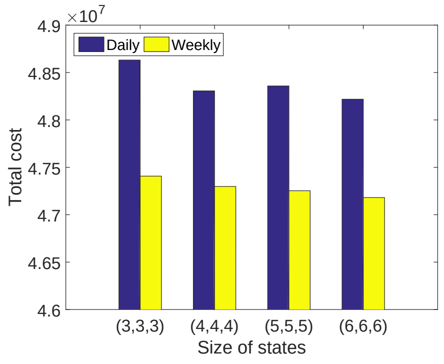
All numerical results presented in this subsection have a 1-hour interval. The two representative parameter settings considered in this subsection are:

1. $\eta = 1$, $K = 600$ kWh, $c_{char} = c_{dis}$ = 300 kW, $K/E[D_t] = 1.0572$, $E[S_t]/E[D_t] = 0.5357$;

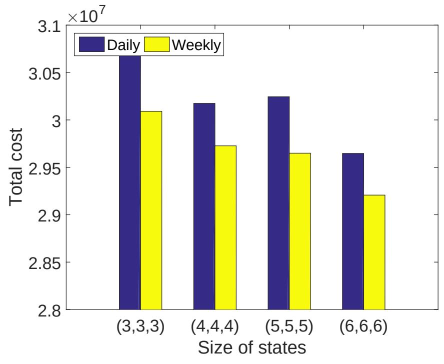
2. $\eta = 0.85$, $K = 800$ kWh, $c_{char} = c_{dis}$ = 100 kW, $K/E[D_t] = 2.17$, $E[S_t]/E[D_t] = 0.468$.

For MDP, we use 7 discretized storage levels under the first parameter setting, and 9 discretized storage levels under the second parameter setting. (Note that the other heuristic policies do not discretize storage level.)

In Fig. 4.2 and 4.3 we present the total discounted cost resulting from MDP and ORA under the daily model and weekly model. We note from these figures that the incorporation of weekly fluctuation mildly improves the performance of MDP and ORA (by about $1 -$

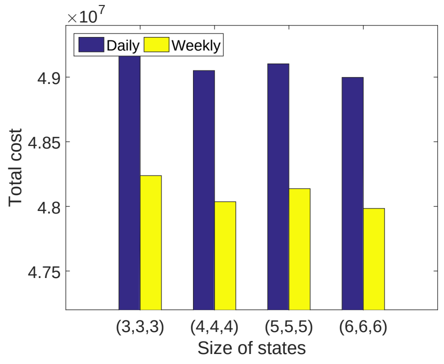


(a) The first parameter setting

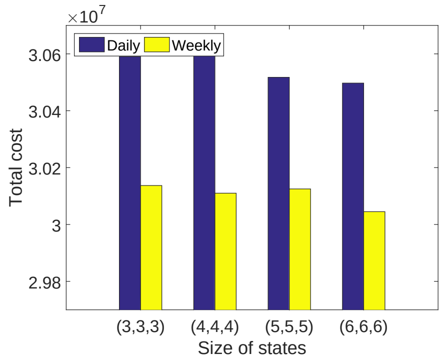


(b) The second parameter setting

Figure 4.2: Total cost resulting from MDP under the daily and weekly models



(a) The first parameter setting

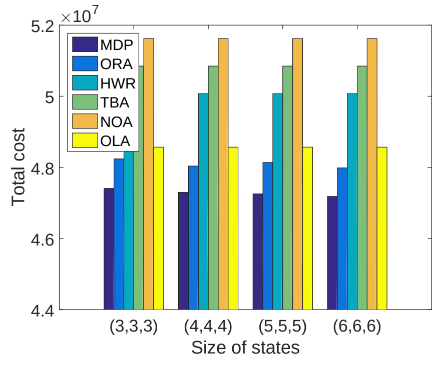


(b) The second parameter setting

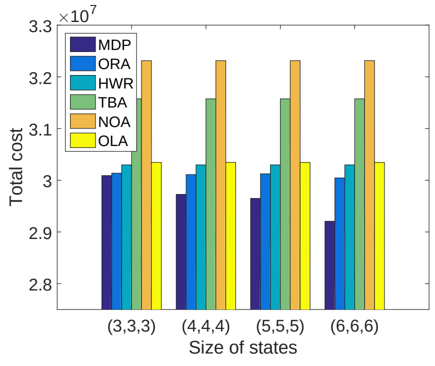
Figure 4.3: Total cost resulting from ORA under the daily and weekly models

2.5%). We will therefore apply the weekly model for MDP and ORA in all our 1-hour tests throughout the section. Here we do not include the comparison for OLA because incorporating weekly fluctuation leads to negligible performance improvement for OLA.

In Fig. 4.4 and 4.5 we compare the total discounted cost resulting from all algorithms under different sizes of the discretized states (of demand, solar generation, and cost). We observe from Fig. 4.4 that while the performance of MDP is slightly improved as the number of discretized states increases, the performance of other heuristic policies are not

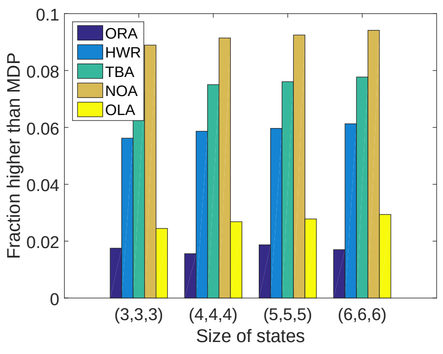


(a) The first parameter setting

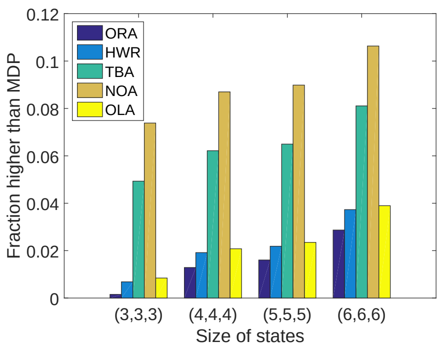


(b) The second parameter setting

Figure 4.4: Cost comparison under the weekly model and various sizes of discretized states



(a) The first parameter setting



(b) The second parameter setting

Figure 4.5: Fractional cost savings comparison under the weekly model and various sizes of discretized states

sensitive to the size of state space. We observe from Fig. 4.5 that ORA achieves the best performance: the gap between ORA and MDP is almost always less than 2% (except in the $(6, 6, 6)$ case under the second parameter setting). OLA is the second best, and results in about $1 - 4\%$ more cost than MDP. HWR achieves similar performance as OLA under the second parameter setting, but leads to much higher cost than OLA under the first parameter setting. This is in correspondence with the observation in Section 4.5.2: HWR

	(3,3,3)	(4,4,4)	(5,5,5)	(6,6,6)
MDP	25.72	345.48	4516.00	21880
ORA	14.46	22.57	45.3	63.08
HWR	2.023	2.023	2.023	2.023
TBA	0.001	0.001	0.001	0.001
NOA	0.0016	0.0016	0.0016	0.0016
OLA	0.069	0.07	0.071	0.073

Table 4.2: Computational time under the first parameter setting (in second)

	(3,3,3)	(4,4,4)	(5,5,5)	(6,6,6)
MDP	57.76	1376.6	29671	117081
ORA	11.07	21.89	65.72	94.5
HWR	1.834	1.834	1.834	1.834
TBA	0.001	0.001	0.001	0.001
NOA	0.0014	0.0014	0.0014	0.0014
OLA	0.072	0.0908	0.0962	0.1123

Table 4.3: Computational time under the second parameter setting (in second)

performs well when the number of hours needed to fully charge the storage, K/c_{char}, is big. We have implemented the ORA and OLA algorithms with much larger state space (up to $(40, 40, 40)$); the performance of both algorithms remains almost the same as the size of state space increases from $(6, 6, 6)$ to $(40, 40, 40)$.

In Table II and III we compare the computational time of all algorithms for the scheduling of one-week storage operation. While the size of discretized states significantly (mildly) increases the computational time of MDP (ORA, respectively), it has little influence on the computational time of the other heuristic algorithms. We note that the computational time of MDP is much higher under the second parameter setting, mainly because of the high value of K/c_{char} that leads to more discretized states of storage level. It is also worth noting that ORA takes much less computational time than MDP in all cases, and that somewhat surprisingly, OLA is faster than HWR.

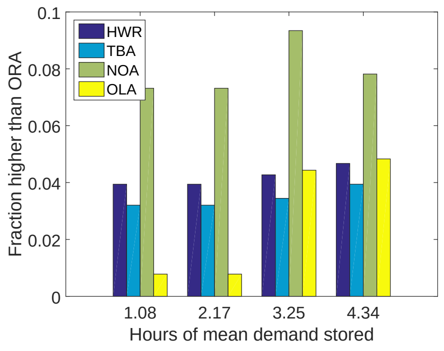
In summary, ORA is comparable to MDP: with problem sizes of $(3, 3, 3)$ and $(4, 4, 4)$, it leads to $0.2 - 2\%$ more total cost than MDP, and is faster than MDP in the $(4, 4, 4)$ case. OLA leads to $1 - 4\%$ more total cost than MDP, and requires negligible computational time. HWR achieves similar performance as OLA under the second parameter setting with $K/c_{char} = 8$, and leads to significantly higher cost than OLA under the first parameter setting with $K/c_{char} = 2$.

4.5.2 Benchmarking Heuristics against ORA

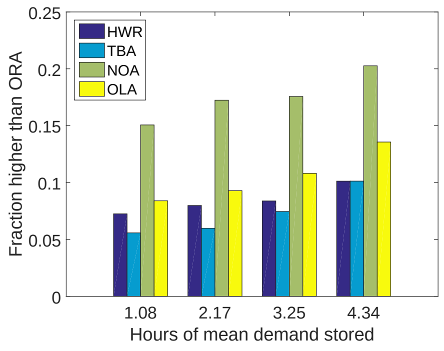
Here we compare the five heuristics ORA, OLA, HWR, TBA and NOA but without MDP. Motivated by the excellent performance of ORA (cf. Section 4.5.1), we compare the other four algorithms against ORA in the next set of experiments. In addition we let storage charging/discharging efficiency $\eta = 0.85$, and consider both 1-hour/5-min intervals (corresponding to $N = 24$ and $N = 288$, respectively). We vary the storage capacity K and scale the average PV supply $\mathbb{E}[S_t]$ based on the average demand $\mathbb{E}[D_t]$. While we will consider variations, we will mainly consider the baseline of: Hours of "average" demand in storage, i.e. $K/\mathbb{E}[D_t]$ as 2.17; Hours to fully charge/discharge, i.e. K/c_{char} as 8; Ratio of average PV supply to average demand, $\mathbb{E}[S_t]/\mathbb{E}[D_t]$, as 0.468.

We first estimate the state transition matrices (of demand, solar generation, and cost) using training data. We tried various alternatives for size of the state space. We chose number of states in demand, renewable generation, and cost Markov chain to be (4,4,4). Incidentally, when we increased the number of states to (10,10,10), the results approximately remain the same.

The simulation results are described in Fig. 4.6-4.9 where we compare the five heuristic algorithms with the y-axis denoting $(b - a)/a$ where a is the minimum total discounted cost (that is obtained by the ORA algorithm) and b is the corresponding heuristic's total discounted cost. While the left side displays correspond to the 1-hour case, the right side

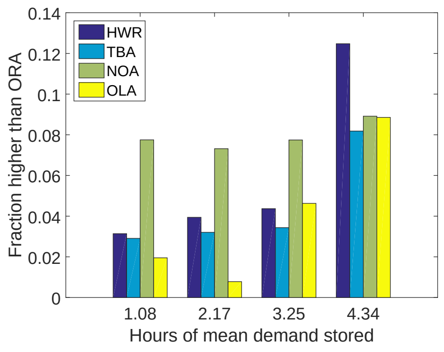


(a) 1-hour intervals

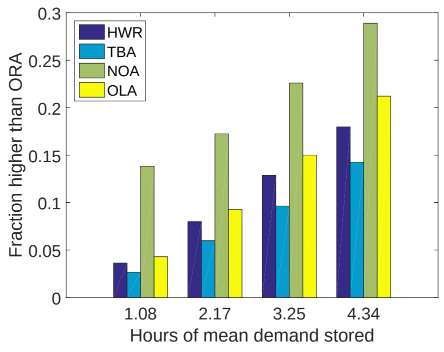


(b) 5-minute intervals

Figure 4.6: Heuristics' performance over varying storage capacity (via $K/\mathbb{E}[D_t]$) keeping c_{char} constant



(a) 1-hour intervals

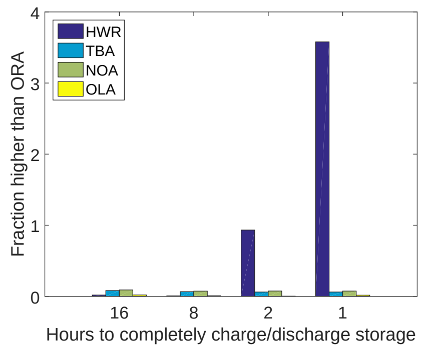


(b) 5-minute intervals

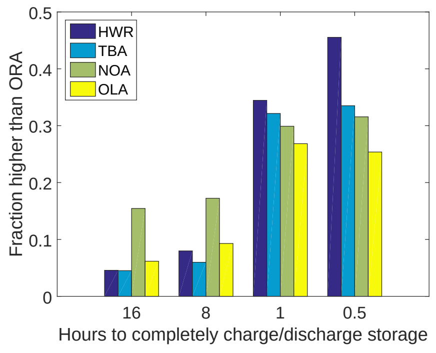
Figure 4.7: Heuristics' performance over varying storage capacity (via $K/\mathbb{E}[D_t]$) keeping $K/c_{char} = 8$

figures are based on 5-min real data.

For the testing we let $U_0 = K/2$, i.e., the initial storage level is 50% of storage capacity. For NOA, we selected tolerance parameters $\phi_c = \phi = 0.25$ by testing several options. Interestingly the 0.25 value is robust and the solutions do not change with much higher or lower values of ϕ. We observe from these four figures that for most all 1-hour cases, the ADP-based OLA algorithm performs slightly worse than ORA, and yields the minimum

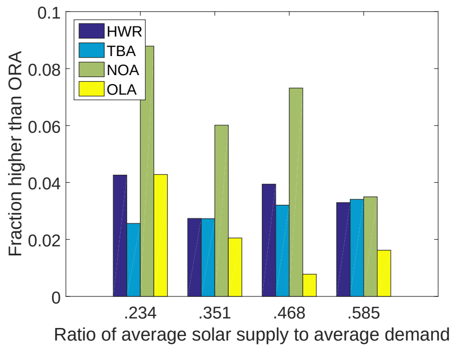


(a) 1-hour intervals

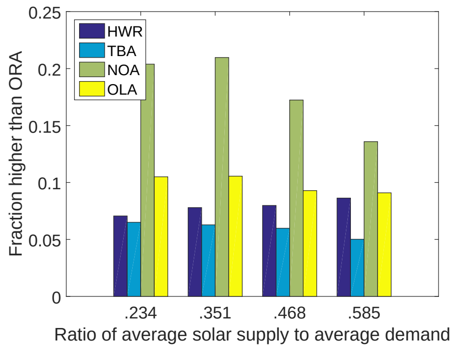


(b) 5-minute intervals

Figure 4.8: Heuristics' performance over varying hours to completely charge/discharge storage (via K/c_{char}) keeping K fixed



(a) 1-hour intervals



(b) 5-minute intervals

Figure 4.9: Heuristics' performance over varying ratio of average solar supply to average demand (via $\mathbb{E}[S_t]/\mathbb{E}[D_t]$)

total discounted cost among the four heuristics. In many 5-minute cases, however, OLA algorithm performs worse than some other heuristics (e.g., TBA). This is intuitive since OLA always treats the next stage (the next five minutes in 5-minute case) as the terminal stage and completely ignores the system dynamics after the next stage.

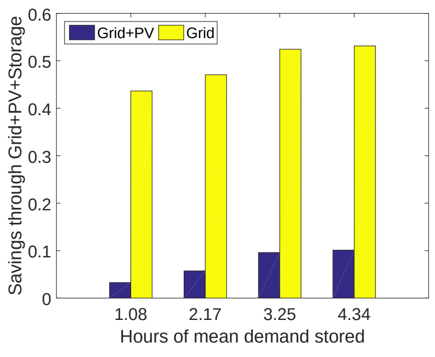
In Fig. 4.6, we fix the maximum charging rate c_{char} and vary the storage capacity K. Note that the ratio K/c_{char} is 4, 8, 16, and 32 hours for the four cases, respectively. We observe from Fig. 4.6 that HWR performs reasonably well in all cases, and yields $2\% - 13\%$ more cost than ORA. Also, we observe that TBA achieves the minimum cost (among the four heuristics) in the 5-minute case. In Fig. 4.7, we fix the ratio $K/c_{char} = 8$ and vary the storage capacity K. This is a more practical setting since the maximum charging rate usually grows (nearly) proportionally to the storage capacity. The performance gap between ORA and the four heuristics increases with storage capacity.

In Fig. 4.8 we fix the storage capacity K and vary the capacity to charging rate ratio K/c_{char}. The parameter setting in our simulation is motivated by the development of fast-charging batteries [80]. For example, the lithium-ion titanate batteries are capable of recharging to 95% of full capacity within approximately ten minutes [80]. We observe from Fig. 4.8 that the performance of HWR heavily depends on the ratio K/c_{char}: the performance gap between HWR and ORA is mild when this ratio is larger than 8 (i.e., it takes more than 8 hours to fully charge the storage); however, for fast-response storage devices with $K/c_{char} \leq 2$, both ORA and OLA significantly outperform HWR.

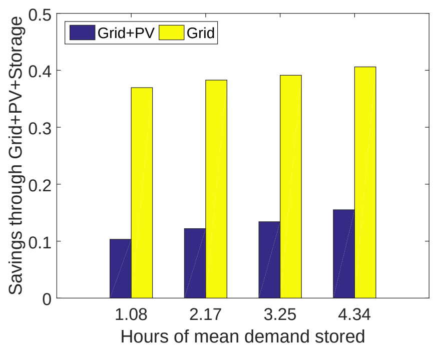
Finally, in Fig. 4.9, we vary the ratio of average solar supply to average demand ($\mathbb{E}[S_t]/\mathbb{E}[D_t]$) while fixing the other parameters. The parameter setting is motivated by the fast growing installment of solar panels on the consumer side. We note that as the solar penetration increases, ORA still outperforms the other four heuristics (including OLA), especially in the 5-minute case.

4.5.3 The Value of Storage and PV

There are costs to install a solar PV system and/or an ESD. A natural question to ask is whether the PV and/or ESD installation was worth it. For that we consider two parameters: value of storage and value of PV and storage. We use the same test data as the previous sub-section and policy based on ORA. In Fig. 4.10-4.13, the y-axis denotes $(a - b)/a$ where b is the total discounted cost using both PV and storage, while a in the left bars

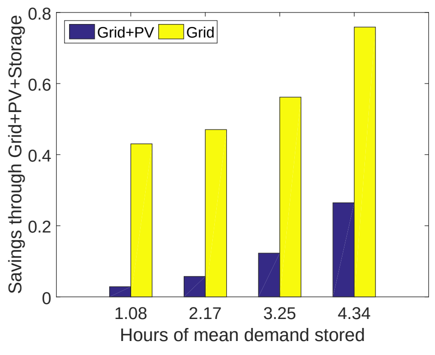


(a) 1-hour intervals

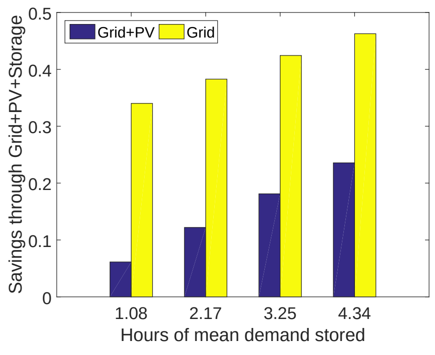


(b) 5-minute intervals

Figure 4.10: Value of storage/PV over varying storage capacity (via $K/\mathbb{E}[D_t]$) keeping c_{char} constant



(a) 1-hour intervals

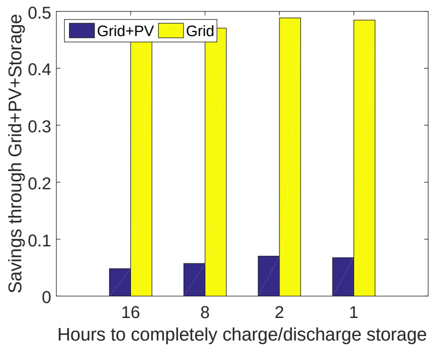


(b) 5-minute intervals

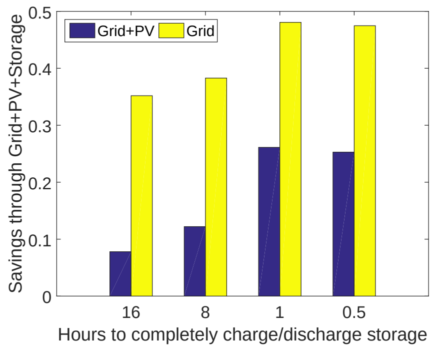
Figure 4.11: Value of storage/PV over varying storage capacity (via $K/\mathbb{E}[D_t]$) keeping $K/c_{char} = 8$

correspond to the (optimal) use of only PV (we note that without storage, the optimal operation of PV is trivial: simply use as much solar generation as possible to fulfill the current demand), and a in the right bars correspond to the use of neither PV nor storage.

The left bars present the fractional cost savings due the operation of storage, and can be therefore viewed as an illustrator on the value of storage. Similarly, the right bars illustrates the value of storage and PV. We observe from Fig. 4.10-4.13 that the value

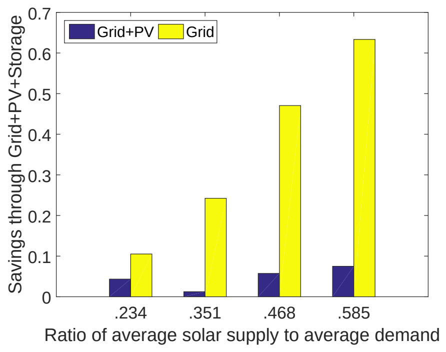


(a) 1-hour intervals

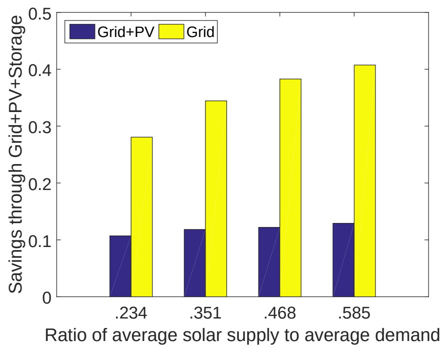


(b) 5-minute intervals

Figure 4.12: Value of storage/PV over varying hours to completely charge/discharge storage (via K/c_{char}) keeping K fixed



(a) 1-hour intervals



(b) 5-minute intervals

Figure 4.13: Value of storage/PV over varying ratio of average solar supply to average demand

of storage is much higher in 5-minute cases, due to the higher variability in costs under 5-minute real-time pricing than that under hourly pricing.

As shown in Fig. 4.10, the values of storage and PV do not increase appreciably with increase in storage size (without increasing rates of charging/discharging). We observe from Fig. 4.11 that the value of storage increases sharply with the storage capacity K, when the maximum charging rate c_{char} grows in proportional to K. We observe from Fig.

4.12 that the values of storage and PV increase with the maximum charging/discharging rate of the storage. Fig. 4.13 shows that the values of storage and PV increase with average solar PV generation.

4.6 Conclusion

Although deceptively easy to state, the problem of determining energy mix from the grid, renewable source and storage device is fairly complex to solve. We implemented six policies MDP, ORA, OLA, HWR, TBA and NOA, and compared their performance using real data of energy demand, renewable generation, and electricity prices.

The following were our findings.

1. ORA outperforms the other four heuristics in all cases, and at the same time, requires the most computing power among the five heuristics. For 1-hour cases, ORA results in $0.2-2\%$ more total cost than MDP, and OLA leads to $1-4\%$ more total cost than MDP. For many 5-minute cases, however, ORA significantly outperforms OLA; this is intuitive, since OLA always treats the next stage (the next five minutes in this case) as the terminal stage and completely ignores the system dynamics after the next five minutes. TBA performs well in some 5-minute cases. Tuning tolerance and coefficient parameters had virtually no effect on NOA.

2. HWR is an easily implementable algorithm that needs no training. Its performance to a large extent depends on the number of hours to fully charge storage (i.e. K/c_{char}). It achieves almost the same total discounted cost as ORA when K/c_{char} is large (e.g. > 8). On the other hand, the two ADP-based algorithms, ORA and OLA, significantly outperform HWR for the case with $K/c_{char} \leq 2$.

3. Under the one-step roll-out algorithm (ORA) and hourly pricing, value of storage is not too high with $K/c_{char} = 8$. Value of storage is much higher in 5-minute cases,

because the cost is much more fluctuating under 5-minute real-time pricing than that under hourly pricing. Value of storage would improve greatly if the storage size increases along with speed of charging and discharging. As solar penetration goes higher, storage has more value.

5. OPTIMAL DAY-AHEAD POWER PROCUREMENT WITH RENEWABLE ENERGY AND DEMAND RESPONSE*

5.1 Introduction

In recent years, many industries have witnessed a tremendous increase in energy consumption that has resulted in enormous expenses as well as carbon pollution. In 2013, U.S. data centers, one of the today's fastest-growing industries, consumed an estimated 91 billion kilowatt-hours of electricity, which is equivalent to the annual output of 34 large (500-megawatt) coal-fired power plants. Moreover, data centers energy consumption is projected to increase to roughly 140 billion kilowatt-hours annually by 2020, costing \$13 billion annually in electricity bills and emitting nearly 100 million metric tons of carbon pollution per year [81]. For this reason, many energy-intensive industries are striving to reduce energy cost and to have a positive impact on the environment. In this situation, renewable energy is considered as a promising solution for them to be energy-efficient. In other words, industries have an opportunity to utilize renewable energy to partially or fully serve their demand load to curtail expenses for procuring energy. In fact, U.S. renewable electricity has grown up to 13.5% of total electricity, and 7.4% of energy consumption in the industrial sector is currently met by renewable energy [82]. In addition, the amount of industrial energy consumption saved by renewable energy has been continuously increasing, and this trend is expected to continue in the future. In addition, from the energy-consumers perspective, there exists an opportunity for industries to adjust purchase and consumption of energy in response to time-varying price in the energy market. Traditionally, power consumers use electricity with a flat rate offered by utility companies or energy market for

*Reprinted with permission from "Optimal Day-Ahead Power Procurement with Renewable Energy and Demand Response" by Soongeol Kwon, Lewis Ntaimo, and Natarajan Gautam, *IEEE Transactions on Power Systems*, Accepted, Copyright © 2016, IEEE.

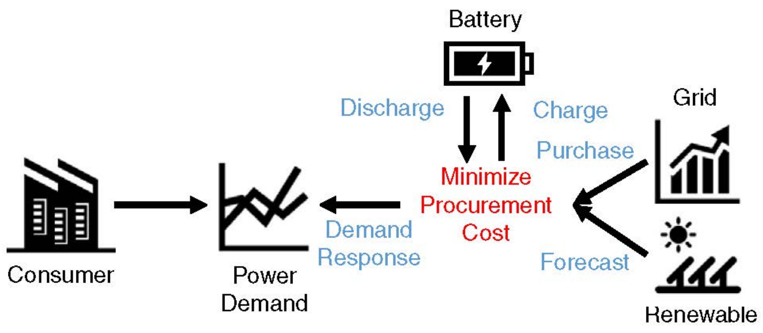


Figure 5.1: Demand-side power procurement

their usage. However, in recent years, it is becoming common for many utilities to offer day-ahead and real-time prices for smart pricing [83], and some independent system operators, such as ERCOT and California ISO, have recently allowed consumers to purchase electricity directly form the market while providing price information. Therefore, industries get a chance to procure energy by participating in the market while being fully aware of the time-varying price, and they may have an opportunity to determine the amount of their energy consumption depending on the electricity price. This opportunity is called *demand response*. Moreover, considering an opportunity to use renewable energy, demand response can also be successfully implemented to utilize renewable energy by consuming more renewable energy when it is available. In addition, by applying demand response to energy procurement, energy storage can be used to mitigate fluctuation of intermittent renewable supply and volatile electricity price. Data centers are one of promising application areas for *demand response*, since they have manageable and flexible workloads [84] and are currently using renewable energy to supply power demand by installing on-site renewable generation facility or make contracts with solar or wind farms [85]. Applying demand response in demand-side power system management is studied under the concept of “Virtual Power Plant" [86], [87], and [88].   To realize the aforementioned opportunities, practitioners are strongly encouraged to develop new technologies for planning, design, control, and operations of power systems against variability and uncertainty in renewable energy and electricity price. In other words, since the conventional systems and techniques have not been designed while considering integration of renewable energy and demand response into power system operations, intermittent renewable generation and volatile electricity price challenge power system engineers' decision making. In this context, current research in the power system has been focused on integrating optimization techniques to yield reliable and robust energy generation and procurement. It is anticipated that application of optimization techniques will have a significant impact on planning, design, control, and operations of power systems. For these reasons, this study focuses on developing a decision-making methodology for demand-side power procurement with renewable energy, storage, and demand response using a stochastic optimization technique. Specifically, this study considers a two-stage power procurement composed of day-ahead and real-time procurements. Note that there is a body of literature on demand-side power procurement based on Markov decision process, [65], [69], [71], [89], [4], and Lyapunov optimization [51], [66], [90], [91]. While all of the aforementioned literature focuses on modeling the sequential stochastic control problem and designing optimal policy tailored to real-time power procurement, this study proposes a two-stage stochastic optimization problem tailored to day-ahead power procurement and suggests a solution approach based on Benders decomposition. To the best of our knowledge, the two-stage stochastic optimization approach for day-ahead power procurement problem with renewable energy, storage, and demand response has not been addressed in the literature. Thus, this study would be a good starting point to study demand-side power procurement problem based on the framework of two-stage stochastic program. The rest of this paper is organized as follows: Section 5.2 gives a detailed description and assumption of the proposed two-stage power procurement problem and formulates the problem as a mathematical model. Section 5.3 introduces an algorithm based on Benders decomposition and suggests strategies designed to improve the algorithm. Section 5.4 analyzes the results obtained by numerical experiments, and Section 5.5 ends the paper with concluding remarks and future research directions.

5.2 Problem Description

5.2.1 Scenario and Assumption

Based on scenario considered in this study, consumer's power demand can be met by the following sources: (i) purchase from energy market, (ii) renewable energy, and (iii) discharge from energy storage as depicted in Figure 3.1. In practice, energy market includes day-ahead and real-time markets that work together as follows:

- *Day-ahead energy market* lets participants commit to buy electricity one day before the operating day to help avoid price volatility.

- *Real-time energy market* allows participants to buy electricity during the course of the operating day to balance mismatch between day-ahead purchase commitment actual demand load.

Considering the operations of energy market, we consider a two-stage framework that consists of day-ahead and real-time power procurement, and propose day-ahead procurement problem. Based on a two-stage stochastic program, the proposed day-ahead power procurement problem is designed so that the first-stage problem determines day-ahead purchase commitment (here-and-now decisions) based on the forecasted demand load and renewable supply, while the second-stage determines the real-time purchase (recourse decisions) to adjust the mismatch between purchase commitments and the actual power demand and renewable supply. We assume that day-ahead electricity price, forecasted power demand and renewable supply are known in the first-stage, but real-time electricity price, actual power demand and renewable supply are time-varying and stochastic. Note that forecasting power demand and renewable supply are out of the scope of this study. In addition, we consider energy storage operations with finite capacity, maximum charging and discharging rates, and inefficiency in charging and discharging. In fact, the frequent cycle of charging or discharging causes the degradation of the energy storage in terms of lifetime and efficiency. However, this study does not consider the degradation since it is assumed to be negligible within one-day operations. Moreover, we implement demand response into the proposed day-ahead power procurement so that consumer assigns time periods in day-ahead and allows demands to be shifted in real-time at assigned time periods, but should be met by the deadline in real time operation. According to the proposed two-stage power procurement framework, based on day-ahead purchase commitment, power loss (i.e. procured power that could not be used to neither serve power demand nor charge storage) might be occurred depending on actual demand load and renewable generations. In our study, we define a penalty cost charged for power loss to ensure that both day-ahead purchase commitment and renewable energy are fully used in real-time operations.

5.2.2 Nomenclature

For the description of mathematical formulation and solution approach, the set of indices, parameters and decision variables are summarized as follows:

5.2.2.1 Sets and Indices

- T: Index set of time periods $t \in T$
- ω: Index set of scenarios $\omega \in \Omega$

5.2.2.2 Deterministic Parameters

- $\overline{D}_t$: Forecasted power demand at time $t \in T$
- $\overline{R}_t$: Forecasted renewable supply at time $t \in T$

- C_t^{DA}: Day-ahead electricity price at time $t \in T$
- M^{char}, M^{dis}: Charging and discharging rate of storage
- S^{max}: Maximum level of energy storage
- η^{char}, η^{dis}: Charging and discharging inefficiency of storage
- P_t^{loss}: Penalty cost for power loss at time $t \in T$
- L^{max}: Allowed number of time periods for shifting demand
- TW: Time window to meet shifted power demand
- ϵ: Maximum fraction of amount of shifted load

5.2.2.3 *Stochastic Parameters (for Each Scenario $\omega \in \Omega$)*

- $D_t(\omega)$: Actual power demand at time $t \in T$
- $R_t(\omega)$: Actual renewable supply at time $t \in T$
- $C_t^{RT}(\omega)$: Real-time electricity price at time $t \in T$

5.2.2.4 *First-Stage Decision Variables (Day-Ahead Operations)*

- x_t: Day-ahead purchase commitment at time $t \in T$
- u_t: Binary variable indicates whether demand load at time $t \in T$ can be shifted by demand response
- zc_t^{DA}, zd_t^{DA}: Amount to be charged/discharged at time $t \in T$
- s_t^{DA}: Level of storage at the beginning of time $t \in T$

5.2.2.5 *Second-Stage Decision Variables (Real-Time Operations)*

- y_t: Real-time electricity purchase at period $t \in T$
- y_t^{loss}: Power loss at period $t \in T$
- $v_{t\ell}$: Amount of load shifted from time $t \in T$ will be satisfied at time $\ell \in T$ $(t < \ell)$
- w_t: Amount of shifted load at the beginning of time $t \in T$
- zc_t^{RT}, zd_t^{RT}: Amount to be charged/discharged at time $t \in T$
- s_t^{RT}: Level of storage at the beginning of time $t \in T$

5.2.3 Mathematical Model

We formulate the proposed day-ahead power procurement problem as a two-stage stochastic mixed-integer programming (SMIP) problem. The first-stage problem determines the purchase commitment and assign periods for shifting demand based on the day-ahead electricity price, forecasted demand and renewable supply considering storage operation to minimize day-ahead purchase cost and the expected recourse cost caused by the real-time procurement for each possible scenario. In the second-stage, the subproblem is defined to adjust mismatch caused by forecasting errors against actual power demand and renewable supply by purchasing electricity from a real-time market and shifting consumers demand based on operations of energy storage (charging/discharging). Our proposed day-ahead power procurement problem can be formulated as a two-stage SMIP as follows:

$$\text{Min} \sum_{t\in T} C_t^{DA} x_t + \mathbb{E}[f(x, u, \tilde{\omega})] \tag{5.1}$$

$$\text{s.t. } x_t + zd_t^{DA} - zc_t^{DA} = \overline{D}_t - \overline{R}_t \quad \forall t \in T \tag{5.2}$$

$$\sum_{t \in T} u_t \leq L^{max} \tag{5.3}$$

$$zc_t^{DA} \leq \min\{M^{char}, S^{max} - s_t^{DA}\} \quad \forall t \in T \tag{5.4}$$

$$zd_t^{DA} \leq \min\{M^{dis}, s_t^{DA}\} \quad \forall t \in T \tag{5.5}$$

$$s_{t+1}^{DA} - s_t^{DA} - \eta^{char} zc_t^{DA} + \frac{1}{\eta^{dis}} zd_t^{DA} = 0 \quad \forall t \in T \tag{5.6}$$

$$x_t, s_t^{DA}, zc_t^{DA}, zd_t^{DA} \geq 0 \quad \forall t \in T \tag{5.7}$$

$$u_t \in \{0, 1\} \quad \forall t \in T \tag{5.8}$$

where for each scenario $\omega \in \Omega$

$$f(x, u, \omega) = \sum_{t \in T} \left(C_t^{RT}(\omega) y_t + P_t^{loss} y_t^{loss} \right) \tag{5.9}$$

$$\text{s.t. } y_t - y_t^{loss} + zd_t^{RT} - zc_t^{RT} + \sum_{\ell=t+1}^{t+TW} v_{t\ell} - \sum_{\ell=t-TW}^{t-1} v_{\ell t} = D_t(\omega) - R_t(\omega) - x_t \quad \forall t \in T \tag{5.10}$$

$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} \leq D_t(\omega) u_t \quad \forall t \in T \tag{5.11}$$

$$w_{t+1} - w_t - \sum_{\ell=t+1}^{t+TW} v_{t\ell} + \sum_{\ell=t-TW}^{t-1} v_{\ell t} = 0 \quad \forall t \in T \tag{5.12}$$

$$w_t \leq \epsilon \sum_{\ell=1}^{t-1} D_\ell(\omega) \quad \forall t \in T \tag{5.13}$$

$$zc_t^{RT} \leq \min\{M^{char}, S^{max} - s_t^{RT}\} \quad \forall t \in T \tag{5.14}$$

$$zd_t^{RT} \leq \min\{M^{dis}, s_t^{RT}\} \quad \forall t \in T \tag{5.15}$$

$$s_{t+1}^{RT} - s_t^{RT} - \eta^{char} zc_t^{RT} + \frac{1}{\eta^{dis}} zd_t^{RT} = 0 \quad \forall t \in T \tag{5.16}$$

$$y_t, y_t^{loss}, v_{\ell t}, w_t, s_t^{RT}, zc_t^{RT}, zd_t^{RT} \geq 0 \quad \forall \ell, t \in T. \tag{5.17}$$

In the above formulation, the objective function (5.1) is composed of day-ahead power procurement costs and the expected recourse cost for real-time power procurement in the second-stage corresponding to the one-day operation cycle. Constraint (5.2) is the power balance equation for day-ahead power procurement plan ensuring that day-ahead purchase commitment is determined so that forecasted power demand is fully satisfied by considering forecasted renewable supply and energy storage operations. Constraint (5.3) assigns time periods for shifting demand in real time with a maximum allowed number of time periods. Constraints (5.4)-(5.6) are for day-ahead storage operations. Constraint (5.7) are the non-negativity restrictions and constraint (5.8) gives the binary restrictions on the first-stage decision variables. In the second-stage, the objective function of the subproblem for each scenario is formulated to minimize real-time operations cost, which is composed by real-time purchase cost and penalty cost for power loss as (5.9). Constraint (5.10) is the power balance equation for real-time power procurement operation including shifting and serving power demand (for demand response) corresponding to the actual power demand and wind power supply given day-ahead purchase commitment. Note that, power loss may happen when the amount of total power procurement is exceeding the actual power demand and the maximum charging amount. Constraints (5.11)-(5.13) are for demand response. Constraint (5.11) defines a condition that power demand can be shifted only at pre-assigned time periods, and constraint (5.12) is the balance equation for demand shifting under demand response. We define the quality of usage constraint as (5.13) so that the fraction of the amount of shifted demand (but not yet served) to the total amount of power demand cannot be exceeded a pre-agreed level. Constraints (5.14)-(5.16) are for real-time energy storage operations, and constraint (5.17) are the non-negativity restrictions on the second-stage decision variables.

Note that "$\min\{\}$" function used in constraints (5.4), (5.5), (5.14), and (5.15) can simply be linearized by two separate constraints. For example, constraint (5.4) is equivalent to $zc_t^{DT} \leq M^{char}$ and $zc_t^{DT} + s_t^{DT} \leq S^{max}$ $\forall t \in T$. We would like to emphasize that in the two-stage SMIP formulation, only the fist-stage problem includes integer variables and the subproblem is formulated without any integer variables, and thus, the proposed two-stage SMIP problem has continuous recourse. In addition, the two-stage SMIP has relatively complete recourse [92] such that every solution obtained by solving the master problem always results in a feasible subproblem.

5.3 Solution Approach

As described in Section 5.2.3, our proposed day-ahead procurement problem is formulated as a two-stage SMIP problem with continuous recourse where only the master problem includes binary decision variables. Note that the two-stage SMIP problem can be modelled as a deterministic equivalent problem (DEP) that is formulated as a large mixed integer programming problem with a finite number of scenarios. In general, solving a DEP of the two-stage SMIP problem is inefficient with a large number of scenarios, and in this case, decomposition techniques can be used to solve the problem efficiently. Specifically, for the continuous recourse, the L-shaped algorithm [93] and the multicut L-shaped algorithm [94] can be used to solve the two-stage stochastic programming problem based on Benders decomposition [95]. The main idea of the L-shaped algorithm and the multicut algorithm is to solve the decomposed master and subproblems separately by approximating a recourse function by adding Benders cuts within the course of solving the master problem. However, both algorithms based on Benders decomposition may lead the slow convergence to get an optimal solution depending on problem structure as well as scenario data. For these reasons, there has been a body of literature that the proposed techniques to generate stronger Benders cuts that accelerate the convergence of the algorithm [96], [97], [98], and [99].

In this study, we propose cut generation strategy (Section 5.3.1) that introduces valid inequalities to generate stronger Benders cuts and define valid optimality cuts that can be added to the master problem in addition to Benders cuts during the course of the multicut L-shaped algorithm. In addition to cut generation strategy, we suggest cut aggregation strategy (Section 5.3.2) based on the relative trade-off between the single cut and multicut methods [100], while investigating the optimal aggregation level of Benders cuts. Let us redefine decision variables used in formulation (5.1)-(5.17) as a set of vectors, $\mathbf{x}, \mathbf{u}$ and $\mathbf{y}$, such that $\mathbf{x}$ denotes vectors of continuous variables (i.e. $x_t, s_t^{DA}, zc_t^{DA}, zd_t^{DA}$ for all $t \in T$) and $\mathbf{u}$ denotes binary variables (i.e. u_t for all $t \in T$) in the first stage, and $\mathbf{y}$ denotes vectors of continuous variables in the second stage (i.e. $y_t, y_t^{loss}, v_{\ell t}, w_t, s_t^{RT}, zc_t^{RT}, zd_t^{RT}$ for all $\ell, t \in T$). Then, with suitable matrices, $\mathbf{A}, \mathbf{D}, \mathbf{W}, \mathbf{T}, \mathbf{H}(\omega)$, and vectors, $\mathbf{c}, \mathbf{b}, \mathbf{e}, \mathbf{q}(\omega), \mathbf{r}(\omega)$, our proposed two-stage day-ahead power procurement problem (5.1)-(5.17) can be defined as follows,

$$\text{Min } \mathbf{c}^\top \mathbf{x} + \mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega})] \tag{5.18}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{5.19}$$

$$\mathbf{D}\mathbf{u} \leq \mathbf{e} \tag{5.20}$$

$$\mathbf{x} \geq 0, \mathbf{u} \in \{0,1\}^n \tag{5.21}$$

where for each scenario $\omega \in \Omega$

$$f(\mathbf{x}, \mathbf{u}, \omega) = \text{Min } \mathbf{q}(\omega)^\top \mathbf{y} \tag{5.22}$$

$$\text{s.t. } \mathbf{W}\mathbf{y} \leq \mathbf{r}(\omega) - \mathbf{T}\mathbf{x} - \mathbf{H}(\omega)\mathbf{u} \tag{5.23}$$

$$\mathbf{y} \geq 0, \tag{5.24}$$

where $\tilde{\omega}$ is a multivariate random variable defined on a probability space with outcome scenarios $\omega \in \Omega$. Let s denote index of scenarios such that $s = 1, \ldots, S$ ($S = |\Omega| < \infty$) and p_s denote the probability of occurrence for each scenario, then based on the multicut

L-shaped algorithm, we solve the following master problem iteratively,

$$\text{Min } \mathbf{c}^\top \mathbf{x} + \sum_{s=1}^{S} p_s \eta_s \tag{5.25}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{5.26}$$

$$\mathbf{D}\mathbf{u} \leq \mathbf{e} \tag{5.27}$$

$$\beta_{\mathbf{t(s)}}^\top \mathbf{x} + \gamma_{\mathbf{t(s)}}^\top \mathbf{u} + \eta_s \geq \alpha_{t(s)} \quad t(s) = 1, ..., u(s), s = 1, ..., S \tag{5.28}$$

$$x \geq 0, u \in \{0,1\}^n, \eta_s \text{ free}, \quad s = 1, \dots, S, \tag{5.29}$$

where $t(s)$ is an index of Benders optimality cuts generated by solving the sub problem with scenarios $s \in S$ and $u(s)$ is the number of Benders optimality cuts added to the master problem during the course of algorithm. Note that Benders optimality cuts (5.28) are generated by solving the following dual subproblem for each possible scenario,

$$f_s(x) = \text{Max } \pi_{\mathbf{s}}^\top (\mathbf{r_s} - \mathbf{T}\mathbf{x} - \mathbf{H}\mathbf{u}) \tag{5.30}$$

$$\text{s.t. } \pi_{\mathbf{s}}^\top \mathbf{W} \leq \mathbf{q} \tag{5.31}$$

$$\pi_{\mathbf{s}} \leq 0, \tag{5.32}$$

with $\alpha_s = p_s(\pi_{\mathbf{s}}^*)\mathbf{r_s}$, $\beta_{\mathbf{s}}^\top = \mathbf{p_s}(\pi_{\mathbf{s}}^*)^\top \mathbf{T}$, and $\gamma_{\mathbf{s}}^\top = \mathbf{p_s}(\pi_{\mathbf{s}}^*)^\top \mathbf{H_s}$ with $\pi_{\mathbf{s}}^*(\mathbf{x})$ an optimal solution of the dual subproblem. We would like to emphasize that our proposed two-stage SMIP problem has relatively complete recourse, and thus, only optimality cuts (5.28) are generated and added to the master problem based on the multicut L-shaped algorithm.

In this study, we implement the Benders decomposition based on single search tree referred to as “Branch-and-Benders-cut" (B&BC) algorithm [101] by using the lazy constraints pool provided by CPLEX Concert Technology (IBM ILOG CPLEX [102]). The main advantage of B&BC is that Benders cuts can be added to the master problem during the course of branch-and-cut algorithm (i.e. single search tree) rather than re-solving the master problem as a new problem at each iteration when Benders cuts are generated and added by solving the subproblems. This can expedite solving the master program. However, there also might be disadvantages of using the lazy constraints pool due to the following reasons. During the course of branch-and-cut algorithm, Benders cuts are generated and added each time when the integer (and fractional) solutions are encountered, and the algorithm check the lazy constraint pool for the fractional solution. This might take longer computational time than the classical implementation of Benders decomposition. Therefore, we conducted preliminary experiments, and results showed that the B&BC algorithm using the lazy constraints pool outperforms the classical implementation for solving the proposed problem. Hence, we implement the multicut L-shaped algorithm by using the lazy constraints pool. The details of our proposed cut generation and aggregation strategies are described in the following Sections 5.3.1 and 5.3.2, respectively.

5.3.1 Cut Generation Strategy

5.3.1.1 Valid Inequalities

The key idea for improving performance of the multicut L-shaped algorithm is to generate stronger Benders cuts so that the solution space of the master problem can be significantly restricted. For the purpose of generating stronger Benders cuts, we propose the following valid inequalities (5.33) and (5.34). By adding valid inequalities (5.33) and (5.34), and projecting them into the solution space of the subproblem, the additional effects of the master problem's solution can be reflected in the subproblem's solution, and thus, stronger Benders cuts can be generated and added.

$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} \leq \epsilon\Big(\sum_{\ell=1}^{t-1} D_\ell(\omega)\Big) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} + \epsilon D_t(\omega) u_t \quad \forall t \in T \tag{5.33}$$

$$w_{t+1} \leq \epsilon \sum_{\ell=1}^{t-1} D_\ell(\omega) + \epsilon D_t(\omega) u_t \quad \forall t \in T. \tag{5.34}$$

For demand response, the two set of valid inequalities ensure that the amount of shifted demand at time period $t \in T$ does not exceed the actual allowable limit that is restricted by the quality of usage constraint (5.13). We have the following propositions and proofs to show the validity of the proposed inequalities (5.33).

Proposition 1. *The following inequality,*

$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} = \epsilon \left(\sum_{\ell=1}^{t-1} D_\ell(\omega) \right) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} + \epsilon D_t(\omega) \quad \forall t \in T, \tag{5.35}$$

is valid for problem (5.1)-(5.17).

Proof. By plugging (5.13) into (5.12), we can show that

$$\begin{aligned}
\sum_{\ell=t+1}^{t+TW} v_{t\ell} &= w_{t+1} - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} \\
&\leq \epsilon \left(\sum_{\ell=1}^{t} D_\ell(\omega) \right) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} \\
&= \epsilon \left(\sum_{\ell=1}^{t-1} D_\ell(\omega) \right) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} + \epsilon D_t(\omega) \quad \forall t \in T,
\end{aligned}$$

which proves the result. □

Proposition 2. *The inequality,*

$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} \leq \epsilon \Big(\sum_{\ell=1}^{t-1} D_\ell(\omega) \Big) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t} + \epsilon D_t(\omega) u_t \quad \forall t \in T, \tag{5.36}$$

is valid for problem (5.1)-(5.17).

Proof. Considering the value of decision variable u_t for all $t \in T$, we have the following two cases:

- Case 1: If $u_t = 0$, then
$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} = 0, \tag{5.37}$$
by constraint (11). Now, for $u_t = 0$, we have the inequality (5.36) as,
$$\sum_{\ell=t+1}^{t+TW} v_{t\ell} \leq \epsilon\Big(\sum_{\ell=1}^{t-1} D_\ell(\omega)\Big) - w_t + \sum_{\ell=t-TW}^{t-1} v_{\ell t}. \tag{5.38}$$
Note that the RHS of (5.38) would be positive by constraint (13), and $v_{\ell t} \geq 0$ for all $\ell, t \in T$. Hence, inequality (5.36) is valid for $u_t = 0$ for all $t \in T$.

- Case 2: If $u_t = 1$, then inequality (5.36) is equivalent to inequality (5.35) which is valid for the proposed problem (1)-(17). Hence, inequality (5.36) is valid for $u_t = 1$ for all $t \in T$.

□

In addition, the following proposition shows the validity of inequality (5.34).

Proposition 3. *The inequality,*
$$w_{t+1} \leq \epsilon \sum_{\ell=1}^{t-1} D_\ell(\omega) + \epsilon D_t(\omega) u_t \quad \forall t \in T, \tag{5.39}$$
is valid for problem (5.1)-(5.17).

Proof. By plugging (5.12) into valid inequality (5.36), we obtain inequality (5.39). □

Note that our proposed valid inequalities (5.36) and (5.39) are equivalent because of equation (5.12), however, their contribution to improve the performance of Benders decomposition might be different. In Section 5.4, we will compare performance improvement by applying each of valid inequalities (5.36) and (5.39).

5.3.1.2 Valid Optimality Cuts

In addition to Benders optimality cuts, we introduce a set of valid optimality cuts designed to be added to approximate the recourse function in the first-stage problem of the two-stage SMIP problem. Note that Laporte and Louveaux [103] developed the optimality cut for approximating the expected recourse function with the binary first-stage problem (i.e. the first-stage problem includes only binary decision variables). In this study, we extend their optimality cut so that it can be used to approximate the expected continuous recourse $F(\mathbf{x}, \mathbf{u}) = \mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega})]$ for the mixed-binary first-stage problem where x is continuous and u is binary decision variables. To introduce the proposed valid optimality cuts, we assume that a lower bound L on $\mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega})]$ is known, that is,

$$L \leq \min_{\mathbf{x},\mathbf{u}} \{\mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega})] | \mathbf{Ax} \leq \mathbf{b}, \mathbf{Du} \leq \mathbf{e}, \mathbf{x} \geq \mathbf{0}, \mathbf{u} \in \{0,1\}^n\}.$$

Let $\mathbf{x^k}$ and $\mathbf{u^k}$ denote the master problem's solution at kth iteration during the course of the multicut L-shaped algorithm, then we have recourse function for $\mathbf{x^k}$ and $\mathbf{u^k}$ as,

$$F(\mathbf{x^k}, \mathbf{u^k}) = \mathbb{E}[f(\mathbf{x^k}, \mathbf{u^k}, \tilde{\omega})],$$

and define the set S^k for kth binary decision variables as,

$$S^k = \{t \mid u_t^k = 1\}.$$

We summarize our proposed optimality cut in the following theorem.

Theorem 6. *The following cut is a valid cut for $F(x, u)$:*

$$\eta \geq (F(\mathbf{x^k}, \mathbf{u^k}) - L) \left(\sum_{t \in S^k} u_t - \sum_{t \notin S^k} u_t - |S^k| + 1 \right) + L - \mathbf{c}^\top (\mathbf{x} - \mathbf{x^k}). \quad (5.40)$$

Proof. 1. If $u = u^k$, $(\sum_{t\in S^k} u_t - \sum_{t\notin S^k} u_t - |S^k| + 1) = 1$.

- If $\mathbf{x} = \mathbf{x^k}$, then the cut $\eta \geq F(\mathbf{x^k}, \mathbf{u^k})$ is tight (i.e. active).
- If $\mathbf{x} \neq \mathbf{x^k}$, then the cut $\eta \geq F(\mathbf{x^k}, \mathbf{u^k}) - c^\top(\mathbf{x} - \mathbf{x^k})$ is valid for $\mathbf{x} \in \{\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^{\mathbf{n}}_+, \mathbf{x} \neq \mathbf{x^k}\}$. In addition, for incumbent solution $\mathbf{x^k}$ and $\mathbf{u^k}$ obtained during the course of branch-and-cut algorithm, the following inequality is valid,

$$\mathbf{c}^\top\mathbf{x} + F(\mathbf{x}, \mathbf{u^k}) \geq c^\top\mathbf{x^k} + F(\mathbf{x^k}, \mathbf{u^k}), \tag{5.41}$$

for all $\mathbf{x} \in \{\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. Note that $\mathbf{c}^\top\mathbf{x} + F(\mathbf{x}, \mathbf{u})$ represents objective function value of overall problem (i.e. including the first and second-stage objective function value) decision variable u is fixed as $u = u^k$ in both left-hand-side and right-hand-side of the above inequality (5.41). Now we have the following inequality:

$$\eta \geq F(\mathbf{x}, \mathbf{u^k}) \geq F(\mathbf{x^k}, \mathbf{u^k}) - c^\top(\mathbf{x} - \mathbf{x^k}). \tag{5.42}$$

This shows that the cut $\eta \geq F(\mathbf{x^k}, \mathbf{u^k}) - c^\top(\mathbf{x} - \mathbf{x^k})$ is valid.

2. If $\mathbf{u} \neq \mathbf{u^k}$, then $(\sum_{t\in S^k} u_t - \sum_{t\notin S^k} u_t - |S^k| + 1) \leq 0$. And let $M = (F(\mathbf{x^k}, \mathbf{u^k}) - L)\left(\sum_{t\in S^k} u_t - \sum_{t\notin S^k} u_t - |S^k| + 1\right)$, then $M \leq 0$ since $F(x^k, u^k) \geq L$.

- If $\mathbf{x} = \mathbf{x^k}$, then the cut is $\eta \geq M + L$ and it must be valid.
- If $\mathbf{x} \neq \mathbf{x^k}$, then the cut $\eta \geq L + M - \mathbf{c}^\top(\mathbf{x} - \mathbf{x^k})$ is valid since,

$$\begin{aligned}\eta \geq F(\mathbf{x}, \mathbf{u^k}) &\geq M + L - F(\mathbf{x^k}, \mathbf{u^k}) + F(\mathbf{x}, \mathbf{u^k}) \\ &\geq M + L - \mathbf{c}^\top(\mathbf{x} - \mathbf{x^k}),\end{aligned}$$

based on inequality (5.41).

$\square$

We would like to emphasize that optimality cut (5.40) is weak, therefore it should be used together with Benders cuts to improve the performance. In addition, optimality cut (5.40) can be implemented into the cut aggregation scheme based on the multicut L-shaped algorithm. Let $j \in J$ be the index of cut aggregate and define the expected recourse function for the subset of scenarios corresponding to each cut aggregate as,

$$F_j(\mathbf{x}, \mathbf{u}) = \mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega}_j)]. \tag{5.43}$$

Assuming that a lower bound L_j is known, that is,

$$L_j \leq \text{Min}_{x,u}\{\mathbb{E}[f(\mathbf{x}, \mathbf{u}, \tilde{\omega}_j)] | \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{D} \leq \mathbf{e}, \mathbf{x} \geq \mathbf{0}, \mathbf{u} \in \{0,1\}^n\}. \tag{5.44}$$

Then, the following cut is a valid optimality cut for $F_j(x, u)$:

$$\eta_j \geq (F(\mathbf{x}^{\mathbf{k}}, \mathbf{u}^{\mathbf{k}})_j - L_j) \left(\sum_{t \in S^k} u_t - \sum_{t \notin S^k} u_t - |S^k| + 1 \right) + L_j - \mathbf{c}^\top(\mathbf{x} - \mathbf{x}^{\mathbf{k}}). \tag{5.45}$$

Optimality cut (5.45) can be added to the master problem together with Benders optimality cuts for the approximated recourse function of each cut aggregate, η_j. To implement optimality cuts (5.45), lower bound L_j can be determined by solving the following relaxed problem:

$$\begin{aligned} L_j = \text{Min} & \sum_{\omega \in \Omega_j} p(\omega)\mathbf{q}(\omega)^\top \mathbf{y}(\omega) \\ \text{s.t. } & \mathbf{W}\mathbf{y}(\omega) \leq \mathbf{r}(\omega) - \mathbf{T}(\omega)\mathbf{x} - \mathbf{H}(\omega)\mathbf{u} \quad \forall \omega \in \Omega \end{aligned}$$

$$\mathbf{Ax} \leq \mathbf{b} \tag{5.46}$$

$$\mathbf{x} \geq 0, \mathbf{u} \in [0,1]^n, y(\omega) \geq 0 \quad \forall \omega \in \Omega,$$

where Ω_j represents subset of scenarios corresponding to cut aggregate $j \in J$. Note that problem (5.46) is relatively easy to solve with relaxed binary decision variables.

5.3.2 Cut Aggregation Strategy

The motivation of cut aggregation stems from the relative advantages of the L-shaped (single cut) algorithm and the multicut L-shaped algorithm. In general, the multicut L-shaped algorithm has less major iterations via passing more information by allowing for cuts up to the number of scenarios than the L-shaped algorithm, however, solving the master problem requires more computation time. On the other hand, when we aggregate cuts and less number of optimality cuts are added to the master problem, the algorithm may have more major iterations due to loss of information caused by aggregation. However, the master problem can be solved easier than when the multicut L-shaped algorithm is used. Based on the trade-off in terms of computational time, authors of [100] suggested an adaptive optimality multicut method that dynamically adjusts the level of aggregation of the optimality cuts in the master problem during the course of the algorithm. The numerical results of [100] show that the optimal computational time is achieved on some middle level of aggregation, but this level is not known a priori and depends on problem structure. In a similar fashion, we try to investigate an appropriate aggregation levels based on the trade-off of algorithm performance in terms of computational time.

In this study, we propose a cut aggregation strategy that assigns Benders optimality cuts to be aggregated for the given aggregation level during the course of the algorithm. The fundamental idea of our suggested strategy is to aggregate Benders cuts while minimizing loss of information caused by cut aggregation. This can be accomplished by aggregating

Benders optimality cuts obtained from the subproblem defined by "similar" scenario data. Each scenario consists of three-dimensional vectors, power demand, renewable supply, and electricity prices, respectively. These vectors show time-varying patterns across 24 hours periods corresponding to one-day time horizon. We would like to emphasize that relations among power demand, renewable supply, and electricity prices have a significant impact on the solution of the subproblem due to the problem structure. For example, if there exists a negative correlation between power demand and electricity price, then the optimal solution of subproblem is determined so that storage is charged and discharged more frequently as well as more power demand is shifted to minimize expense. In this context, we characterize the structure of each scenario data using pairwise correlations between power demand, renewable supply, and electricity prices and measure similarity of scenario data based on those correlations. For example, correlation between series of $D_t(\omega)$ and $C_t^{RT}(\omega)$ across time periods $t \in T$ for each scenario $\omega \in \Omega$, $\rho_{DC}(\omega)$, can be computed as follows:

$$\rho_{DC}(\omega) = \frac{\sum_{t=1}^{24}(D_t(\omega) - \overline{D}(\omega))(C_t(\omega) - \overline{C}(\omega))}{\sqrt{\sum_{t=1}^{24}(D_t(\omega) - \overline{D}(\omega))^2 \sum_{t=1}^{24}(C_t(\omega) - \overline{C}(\omega))^2}}, \tag{5.47}$$

where $\overline{D}(\omega)$ is the average power demand and $\overline{C}(\omega)$ is the average electricity prices for each scenario $\omega \in \Omega$. Likewise, we can determine pairwise correlation between power demand and renewable supply, $\rho_{DR}(\omega)$, and renewable supply and electricity price, $\rho_{RC}(\omega)$, for each scenario $\omega \in \Omega$.

To implement our idea for cut aggregation, we cluster scenarios using k-means clustering algorithm based on pairwise correlation values of each scenario. As mentioned above, three pairwise correlations are computed for each scenario, and thus we can cluster scenarios using k-means up to 3-dimensions based on selection of those pairwise correlations. For example, for 1-dimensional clustering, we can pick one of $\rho_{DC}(\omega)$, $\rho_{DR}(\omega)$,

Energy Storage	S^{max}	$S^{max} = \frac{E[D_t(\omega)]}{2}$
	M^{char}, M^{dis}	$M^{char} = M^{dis} = \frac{E[D_t(\omega)]}{4}$
	η^{char}, η^{dis}	$\eta^{char} = \eta^{dis} = 0.9$
Demand Response	L^{max}	$L^{max} = 4$
	TW	$TW = 4$
	ϵ	$\epsilon = 0.05$
Forecasted Demand	$\overline{D}_t$	$\overline{D}_t = E[D_t(\omega)]\ \forall t \in T$
Forcasted Renewable	$\overline{R}_t$	$\overline{R}_t = E[R_t(\omega)]\ \forall t \in T$
Penalty Cost	P_t^{loss}	$P_t^{loss} = C_t^{DA}$

Table 5.1: Parameter setting

and $\rho_{RC}(\omega)$), and for 2-dimensional clustering, we can choose combination of two correlations, $\rho_{DC}(\omega)$ and $\rho_{DR}(\omega)$, $\rho_{DR}(\omega)$ and $\rho_{RC}(\omega)$), $\rho_{DC}(\omega)$ and $\rho_{RC}(\omega)$). Note that original k-means clustering algorithm does not guarantee to generate equal-sized cluster, therefore we implemented k-means algorithm by using an open source data mining software [104] so that it yields equal-sized k clusters (i.e. each cluster consists of n/k where n is the number of scenarios) for balanced aggregation. Once the scenarios are clustered, Benders optimality cuts generated by solving the subproblem for scenarios in the same cluster will be aggregated and added to the master problem.

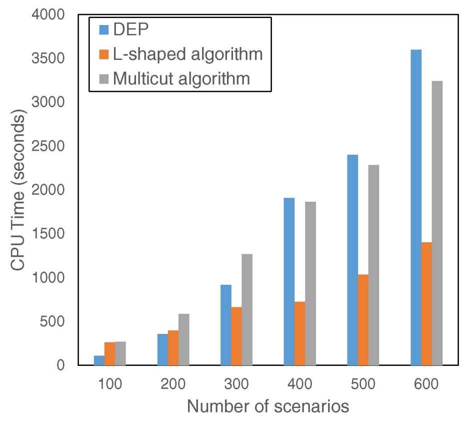
5.4 Numerical Experiments

For numerical experiments, we investigated the performance of the proposed algorithm using scenarios generated by the probabilistic model introduced by Kwon et al. [4]. Through analyzing real historical data, it is evident that power demand, wind generation, and electricity price are time-varying and stochastic, however, it is also reasonable to assume that there exist daily cyclic patterns in power demand and electricity price. In other words, there are deterministic and stochastic variabilities in power demand, renewable generation, and electricity price. Kwon et al. [4] proposed the probabilistic model using on Markov chain to adequately capture the both deterministic and stochastic variabilities.

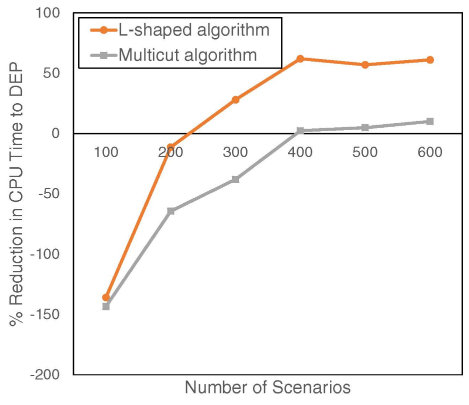
To generate a set of representative scenarios that adequately captures the both deterministic and stochastic variabilities, we train the probabilistic model by using real historical data obtained from Pennsylvania New Jersey Maryland (PJM) interconnection [105], and randomly generate scenarios using Monte Carlo simulation for replications. Note that the proposed probabilistic model using discrete time Markov chains on discrete state spaces, and we mapped random variables to 20 discretized states ($M = 20$) so that power demand, wind supply, and electricity price have 20 different values for each time period. Once we generate a pool of 100,000 scenarios, we obtain 10 replications for each sample size, 100, 200, 300, 400, 500, and 600 by selecting instances randomly from scenario pool. In addition, in terms of parameters in the proposed problem, we set parameters' value as described in Table 5.1. All the experiments were conducted on an Intel Core i7-3740 2.70GHz processor with 16GB memory. We summarize various numerical results for performance evaluation in the following Sections 5.4.2 and 5.4.3.

5.4.1 Value of Stochastic Solution

Before analyzing performance of our proposed approach, we would like to discuss the value of stochastic solution (VSS) [106]. In general, stochastic programs are computationally difficult to solve, and thus, practitioners may want to formulate the real-world problem as simpler versions, e.g. deterministic optimization problem by using nominal values as you mentioned. The solution obtained from the simpler versions of problems may provide nearly optimal solutions, however, sometimes yield totally inaccurate solution due to the lack of considering uncertainties. In this case, we can measure the value of the stochastic program by using VSS which is the possible cost reduction obtained by solving the stochastic optimization problem. When no further in formation about the future is available, VSS becomes more practically relevant [92]. We conducted preliminary experiments to analyze the quantity of VSS and we checked that about 10-15% of procure-



(a) Computational time



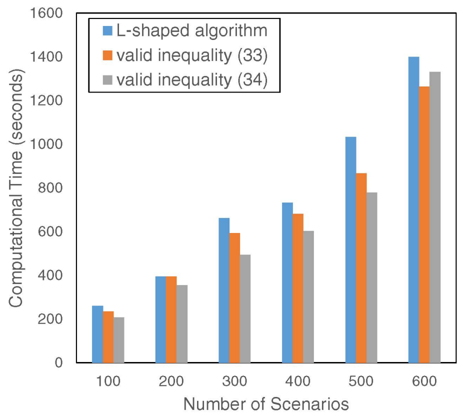
(b) % Reduction against DEP

Figure 5.2: Performance comparison: DEP vs L-shaped vs multicut L-shaped

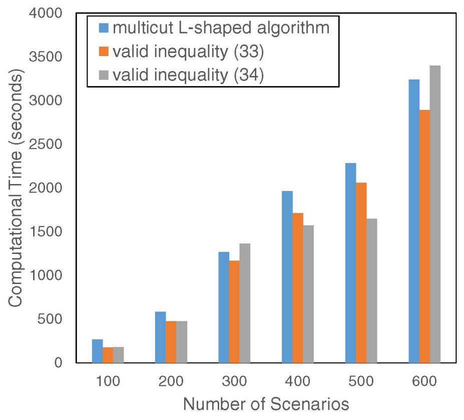
ment cost can be reduced by solving stochastic optimization problem instead of solving deterministic optimization problem using the expected value.

5.4.2 Performance Analysis of Cut Generation Strategy

We analyze the performance of the cut generation strategy introduced in Section 5.3.1. Based on the L-shaped and multicut L-shaped algorithms, we solve the problems for various sizes of scenarios by applying (i) the proposed valid inequalities (5.33) and (5.34), (ii) the proposed valid optimality cut (5.45), and combination of both (i) and (ii). We compare the performance of the L-shaped and the multicut L-shaped algorithm against the DEP. As depicted in Figure 5.2, as the size of scenarios increases, the L-shaped and the multicut L-shaped algorithms outperform the DEP. Moreover, we can find that the L-shaped algorithm shows better performance than the multicut L-shaped algorithm, and this indicates that the performance of the multicut L-shaped algorithm can be improved with cut aggregation strategy as we conjectured. We will investigate the algorithm performance for the different levels of aggregation in Section 5.4.3. Next, we investigate the performance improvement by the proposed valid inequalities for both the L-shaped and the multicut L-



(a) L-shaped algorithm



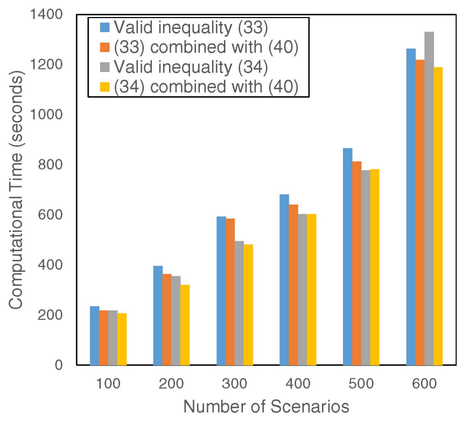
(b) Multicut algorithm

Figure 5.3: Performance analysis of the proposed valid inequalities (5.33) and (5.34)

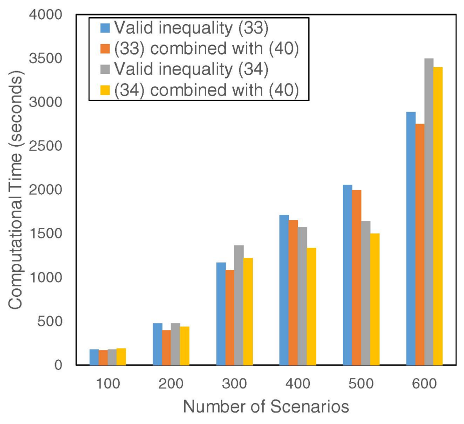
shaped algorithm in terms of computational time. As depicted in Figures 5.3 and 5.4, both the L-shaped and the multicut L-shaped algorithm are improved by applying our proposed valid inequalities. We found that valid inequality 5.34 performs better than valid inequality (5.33) in many instances, however, it does not dominate. In addition, we analyzed the performance improvement by applying the proposed valid optimality cut (5.45) combined with the valid inequalities (5.33) and (5.34). As depicted in Figure 5.4, we can see the most improved performance when applying both the proposed valid inequalities and valid optimality cut simultaneously during the course of the algorithm. Based on these findings, we use the proposed valid inequalities and valid optimality cuts when we investigate the effect of cut aggregation on the performance in Section 5.4.3.

5.4.3 Performance Analysis of Cut Aggregation Strategy

We conducted experiments aimed at studying the performance of the proposed cut aggregation strategy using k-means clustering algorithm introduced in Section 5.3.2. Specifically, we evaluated the performance of the proposed cut aggregation strategy comparing with the static multicut aggregation used by Trukhanov et al. [100]. Under the static mul-



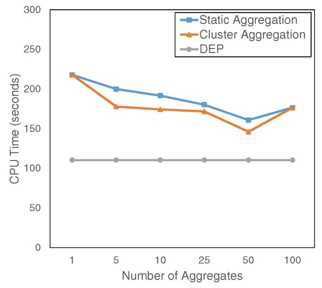
(a) L-shaped algorithm



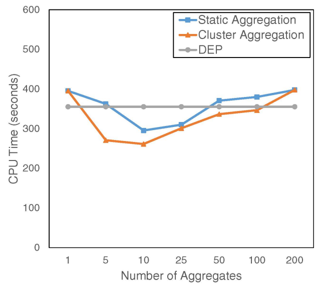
(b) Multicut algorithm

Figure 5.4: Performance analysis of the proposed valid inequalities (5.33) and (5.34) combined with optimality cut (5.40)

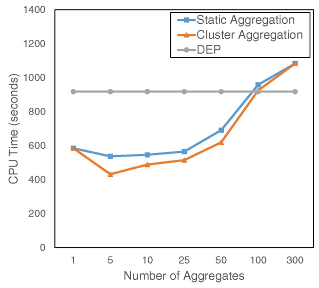
ticut aggregation, total n Benders optimality cuts that are generated from n scenarios were aggregated into k cuts so that each of k cuts is composed of n/k Benders cuts. For example, for 100 possible scenarios ($n = 100$), static cut aggregation with $k = 1$ corresponds to the L-shaped algorithm, $k = 100$ corresponds to the multicut algorithm, and $1 < k < 100$ corresponds to the partial aggregation that resigns between full aggregation (i.e. L-shaped algorithm) and full disaggregation (i.e. multicut algorithm). In addition, for implementation of the proposed cut aggregation, we use k as an input parameter (i.e. number of clusters) of k-means clustering algorithm, and Benders optimality cuts would be aggregated based on clustered scenarios. As described in Section 5.3.2, we have an option to choose dimensions of k-means clusters (up to 3-dimensions) for the combinations of three pairwise correlations, $\rho_{DC}(\omega)$, $\rho_{DR}(\omega)$, and $\rho_{RC}(\omega)$). Note that we use one-dimensional k-means clustering for the pairwise correlation between power demand and renewable supply, $\rho_{DR}(\omega)$, that shows the most improved performance for the scenarios used in this study. Figures 5.5 and 5.6 show the computational times to obtain an optimal solution using the multicut L-shaped algorithm with various aggregation level k where $1 \leq k \leq n$ for



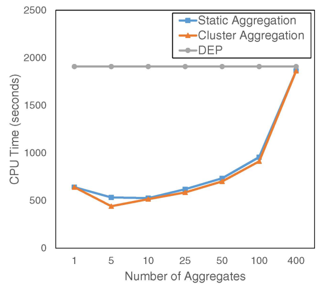
(a) Number of scenarios $n = 100$



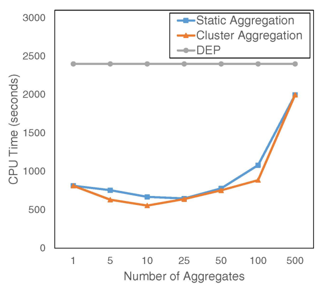
(b) Number of scenarios $n = 200$



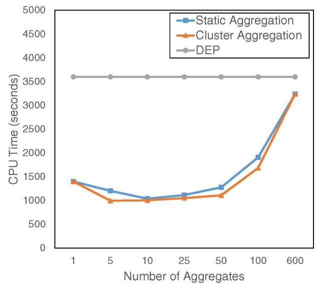
(c) Number of scenarios $n = 300$



(d) Number of scenarios $n = 400$



(e) Number of scenarios $n = 500$



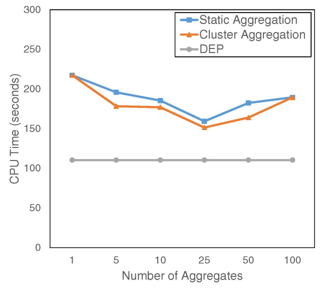
(f) Number of scenarios $n = 600$

Figure 5.5: % Reduction in CPU time: static versus cluster aggregations combined with valid inequality (5.33) and valid optimality cut (5.40)

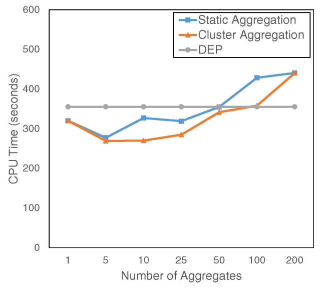
each size of scenarios $n = 100, 200, 300, 400, 500$ and 600. Through analyzing the results of numerical experiments, we find that (i) both the static and the proposed cut aggregation improve the performance of algorithm at certain level of aggregate k, where $1 \leq k \leq n$, (ii) the proposed cut aggregation strategy shows better performance improvement than the static aggregation, and (iii) the best k exists between both extreme cases. We would like to emphasize that the multicut L-shaped algorithm shows better performance at higher aggregation level for scenario data used in this study.

5.5 Concluding Remark and Future Work

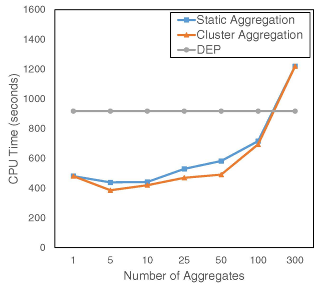
This study is motivated by an opportunity to reduce the energy cost and carbon pollution by utilizing renewable energy and adopting demand response from the demand-side



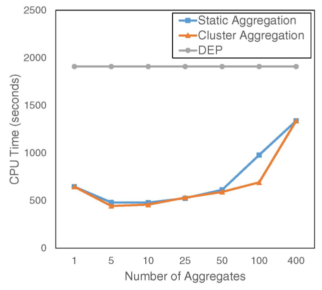
(a) Number of scenarios $n =$ 100



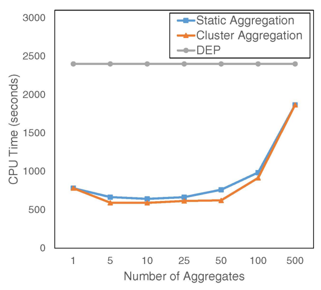
(b) Number of scenarios $n =$ 200



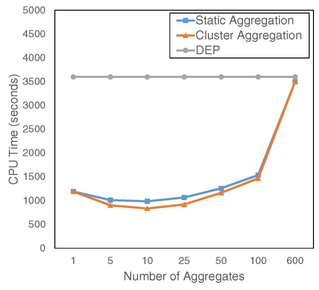
(c) Number of scenarios $n =$ 300



(d) Number of scenarios $n =$ 400



(e) Number of scenarios $n =$ 500



(f) Number of scenarios $n =$ 600

Figure 5.6: % Reduction in CPU time: static versus cluster aggregations combined with valid inequality (5.34) and valid optimality cut (5.40)

perspective. While utilizing renewable energy to meet power demand, consumers may be willing to adjust their demand load, which is called as demand response, to avoid peak electricity price as well as optimally utilize renewable energy to reduce procurement cost. In addition, energy storage can be used to mitigate fluctuations of intermittent renewable supply and volatile electricity price. Considering renewable energy, demand response, and energy storage, the main objective of this study is to propose decision-making models that enable energy consumers to procure energy in a cost-efficient manner in response to variability and uncertainty of renewable supply as well as electricity price. In summary, the main contributions of this paper are: (i) propose day-ahead power procurement problem and formulate it as a two-stage SMIP problem; (ii) introduce cut generation and cut aggregation strategies that can be integrated with the course of the multicut L-shaped algorithm to improve algorithm performance; and (iii) implement the proposed algorithm by using lazy constraints pool provided by CPLEX Concert Technology and investigate performance by conducting numerical experiments with various settings. The proposed day-ahead power procurement problem and solution approach can be applied to many industries (e.g. data centers and manufacturing) and also extended to grid-level power system operations (e.g. micro grid) to curtail expenses of procuring energy to meet demand load. We believe that this study would be a good starting point to study demand-side power procurement problem based on the framework of two-stage stochastic program and will have a significant impact on study for the utilization of renewable energy and implementation of demand response.

6. CONCLUSION

6.1 Summary of The Study

This study is motivated by pressing issues, such as a tremendous increase of energy consumption and cost in industrial sectors, and the main objective is to develop decision making methodologies to improve energy-efficiency and reduce energy cost in demands side, i.e. energy consumption side.

Firstly, this study focuses on improving energy-efficiency in data center operations and developed a server provisioning algorithm leveraging upon standard queueing analysis to simultaneously determine sizing (i.e. number of active servers), assignment and routing appropriately to ensure performance guarantees by enforcing time-stability for a time-varying and fast-changing system. By implementing the proposed server provisioning algorithm, the number of active servers can be adjusted in proportion to time-varying workloads to improve utilization of the servers. Specifically, this dissertation showed that the proposed provisioning algorithm provides performance bounds on both the mean queue length and the mean sojourn time, which can be derived by stationary analysis of queueing model for non-homogeneous and transient system. In addition, this study developed a discrete event simulation and conducted numerical experiments to evaluate the prosed server provisioning algorithm (*Section 2*). In addition, based on the proposed server provisioning algorithm, this dissertation suggests an approach that stabilizes (i) aggregate workload distribution based on moment matching approximation and (ii) arrival rates based on routing fractions to achieve time stable queue length distribution at each active server. Based on time-stability, time-homogeneity constraints can be derived so that they can be used to formulate a mixed-integer program to optimally determine various decisions of sizing, assignments, routing, and speed scaling to minimize energy costs while providing probabilistic performance guarantees on waiting times. Simulation results obtained by using the solution of the proposed mixed-integer program as input parameters show that the mean queue length is stabilized and the mean waiting time is bounded by the targeted value (*Section 3*).

Secondly, this study suggested a demand-side energy procurement model that enables energy consumers to procure energy to meet their power demand in a cost-efficient manner while considering usage of renewable energy based on demand response. Specifically, this dissertation developed a stochastic sequential decision making problem tailored to real-time energy procurement and formulated it as a Markov decision process with periodic cycles. As the dynamic program is computationally intensive for large-scale problems, this dissertation proposed algorithms based on approximate dynamic programming and compared performance of the proposed algorithms against the exact Markov decision process solutions and Lyapunov optimization-based algorithms under a variety of parameter settings on the energy storage capacity, the level of renewable generation, as well as the maximum charging/discharging rates (*Section 4*). Moreover, based on the real world practice in energy market operations, this dissertation formulated a day-ahead power procurement as a two-stage stochastic mixed integer program to minimize the expected energy procurement cost against possible scenarios. To efficiently solve the proposed problem, this dissertation suggests cut generation and cut aggregation strategies that can be integrated with the course of the multicut L-shaped algorithm based on Benders decomposition and conducted numerical experiments to analyze performance improvement by the suggested strategies (*Section 5*).

As a natural next step, integration of server provisioning and energy procurement will be proposed to minimize both energy consumption and cost for energy-efficient data center operations. In recent years, many companies that own and operate data centers (e.g. Google and Apple) have been tried to use renewable energy by installing on-site renewable

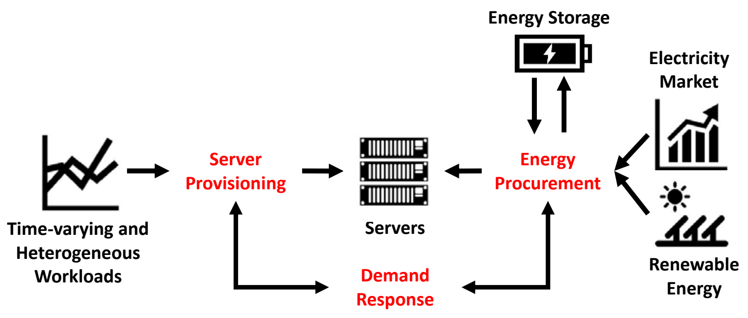


Figure 6.1: Integration server provisioning and power procurement

generation facilities (e.g. installing photovoltaic panels on rooftop) or make contracts with solar or wind farms. Moreover, data centers are regarded as one of the promising industries in which demand response can be successfully applied since they have manageable and flexible (i.e. delay-tolerant) demand load (Wierman et al. [84]). By adopting demand response, data centers are able to avoid peak electricity prices and use more renewable energy by adjusting purchase and consumption of energy. For the aforementioned opportunities, new decision-making methodologies tailored to determine server provisioning and power procurement in an integrated fashion with incorporation of renewable energy and demand response are urgently needed. Figure 6.1 depicts a concept of integration of server provisioning and energy procurement considered. To the best of the author's knowledge, the proposed model that integrates server provisioning and power procurement with renewable energy and demand response has not been well studied in the literature.

Based on the previous study described in Sections 2, 3, 4, and 5, the proposed model integrating server provisioning and energy procurement can be formulated as a probabilistic constrained two-stage stochastic program. Specifically, constraints on performance guarantees and violation for the predefined quality of service level can be defined as probabilistic constraints based on the server provisioning algorithm proposed in Sections 2 and 3, and it can be integrated with the proposed energy procurement model formulated as a two-stage stochastic program in Section 5. Specifically, the proposed integration model includes both probabilistic constrained and two-stage stochastic program features so that:

- The first stage determines proactive server provisioning and day-ahead purchase commitment (i.e. here-and-now decisions) for predicted workloads and forecasted renewable supply,
- the second stage determines reactive server provisioning and real-time purchase (i.e. recourse decisions) against the deviation of actual workload, renewable supply, and electricity price while satisfying probabilistic constraint with the given quality of service level.

The key idea behind the integration of server provisioning and energy procurement is to design a stochastic optimization problem so that (i) operational decisions on energy procurement are determined according to energy consumption derived by server provisioning and (ii) server provisioning can be adjusted by shifting delay-tolerant workloads to avoid purchasing electricity at peak price and utilize more renewable energy under a demand response scheme. The fundamental objective using a probabilistic constrained two-stage stochastic optimization is to obtain reliable and efficient solutions for demand-side management of energy-efficient data center operations considering both server provisioning and power procurement while satisfying quality of usage constraints, typically, in the presence of variability and uncertainty.

The proposed probabilistic constrained two-stage stochastic program can be reformulated as a deterministic mixed-integer program by introducing a big-M term for each inequality in the probabilistic constraint and a binary variable for each scenario (Luedtke et al. [107]). Note that the reformulated deterministic version of the problem is a large-scale mixed-binary program, and thus, it is not easy to solve. Moreover, the weakness of the linear program relaxation of a big-M formulation corresponding to probabilistic constraints makes it hard to solve using the conventional branch-and-cut algorithm provided by commercial tools, such as CPLEX (Liu et al. [108]). Thus, for the proposed probabilistic constrained two-stage stochastic programming problem, the plan is to develop strong valid inequalities to strengthen the linear program relaxation and propose a decomposition algorithm to solve the problem efficiently. Performance of the proposed algorithm and effectiveness of valid inequalities will be analyzed through numerical experiments by comparing it to the result of solving the deterministic mixed-integer program using CPLEX.

6.2 Contribution of The Study

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

- Developed server provisioning algorithms that provide provable performance bounds that can be simply derived based on stationary analysis for time-varying and transient system while considering energy efficiency. (*Section 2*)

- Proposed an integrated framework unifying sizing, assignment, routing, and speed scaling under heterogeneous conditions, which has seldom been implemented jointly. (*Section 3*)

- Introduced a mixed-integer program to reduce energy consumption while providing performance guarantees based on time-homogeneity constraints, which ensures time-stability based on moment matching approximation. (*Section 3*)

- Implemented and numerically compared approaches ranging from Markovian models to hybrid methods based on statistics and optimization to those that are based on Lyapunov optimization, under a variety of parameter settings on the storage size, the level of solar generation, as well as the maximum charging/discharging rates. (*Section 4*)

- Proposed day-ahead power procurement problem and formulated it as a two-stage stochastic mixed integer program and developed cut generation and cut aggregation strategies that can be integrated with the course of the multicut L-shaped algorithm to improve algorithm performance. (*Section 5*)
- Proposed decision making model that integrates server provisioning and power procurement with renewable energy and demand response, which has not been proposed in the literature. (*Section 6*)

6.3 Future Research Work

Based on the results described in the dissertation, there is possible future research work for each section as follows:

- Suggest real-time speed scaling control by varying server processing speed for time-stable performance. (*Section 2*)
- Develop an algorithm to efficiently solve the proposed mixed-integer program for the large-scale problem. (*Section 3*)
- Develop better algorithm to analyze the proposed Markov decision process and improve the developed approximated dynamic program. (*Section 4*)
- Integrate the proposed day-ahead power procurement model with data center operations and extend it to grid level energy system operations. (*Section 5*)

REFERENCES

[1] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, “Dynamic right-sizing for power-proportional data centers,” *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1378–1391, 2013.

[2] S. Kwon and N. Gautam, “Time-stable performance in parallel queues with non-homogeneous and multi-class workloads,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1322–1335, 2016.

[3] S. Kwon and N. Gautam, “Guaranteeing performance based on time-stability for energy-efficient data centers,” *IIE Transactions*, vol. 48, no. 9, pp. 812–825, 2016.

[4] S. Kwon, N. Gautam, and Y. Xu, “Meeting inelastic demand in systems with storage and renewable sources,” *IEEE Transactions on Smart Grid*. in press.

[5] S. Kwon, L. Ntaimo, and N. Gautam, “Optimal day-ahead power procurement with renewable energy and demand response,” *IEEE Transactions on Power Systems*. in press.

[6] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira Jr, “In search of invariants for e-business workloads,” in *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pp. 56–65, 2000.

[7] R. Brown, E. Masanet, B. Nordman, B. Tschudi, A. Shehabi, *et al.*, “Report to congress on server and data center energy efficiency: Public law 109-431,” Lawrence Berkeley National Laboratory, Berkeley, CA, 2008.

[8] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, “Energy-aware server provisioning and load dispatching for connection-intensive internet services,”

in *2008 USENIX Symposium on Networked Systems Design and Implementation*, vol. 8, pp. 337–350, 2008.

[9] J. Hamilton, “Cost of power in large-scale data centers,” 2008. [Online], Available: http://perspectives.mvdirona.com/2008/11/cost-of-power-in-large-scale-data-centers/.

[10] J. Koomey, “Growth in data center electricity use 2005 to 2010,” 2011.

[11] L. Barroso and U. Holzle, “The case for energy-proportional computing,” *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[12] W. Vogels, “Beyond server consolidation,” *Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[13] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, “Energy proportional datacenter networks,” in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 338–347, ACM, 2010.

[14] E. Feller, L. Rilling, and C. Morin, “Energy-aware ant colony based workload placement in clouds,” in *Proceedings of the 2011 IEEE/ACM International Conference on Grid Computing*, pp. 26–33, 2011.

[15] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, “Autoscale: Dynamic, robust capacity management for multi-tier data centers,” *ACM Transactions on Computer Systems*, vol. 30, no. 4, p. 14, 2012.

[16] Y. C. Lee and A. Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[17] K. Wang, M. Lin, F. Ciucu, A. Wierman, and C. Lin, “Characterizing the impact of the workload on the value of dynamic resizing in data centers,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, pp. 405–406, 2012.

[18] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy, "Autonomic mix-aware provisioning for non-stationary data center workloads," in *Proceedings of the 7th International Conference on Autonomic Computing*, pp. 21–30, 2010.

[19] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center sla violations and power consumption via hybrid resource provisioning," in *2011 International Green Computing Conference and Workshops (IGCC)*, pp. 1–8, 2011.

[20] J. A. Gallego Arrubla, Y. M. Ko, R. J. Polansky, E. Pérez, L. Ntaimo, and N. Gautam, "Integrating virtualization, speed scaling, and powering on/off servers in data centers for energy efficiency," *IIE Transactions*, vol. 45, no. 10, pp. 1114–1136, 2013.

[21] R. D. Foley, "Stationary poisson departure processes from non-stationary queues," *Journal of Applied Probability*, vol. 23, no. 01, pp. 256–260, 1986.

[22] J. A. Barnes and R. Meili, "A stationary poisson departure process from a minimally delayed infinite server queue with non-stationary poisson arrivals," *Journal of Applied Probability*, vol. 34, no. 03, pp. 767–772, 1997.

[23] Z. Feldman, A. Mandelbaum, W. A. Massey, and W. Whitt, "Staffing of time-varying queues to achieve time-stable performance," *Management Science*, vol. 54, no. 2, pp. 324–338, 2008.

[24] Y. Liu and W. Whitt, "Stabilizing customer abandonment in many-server queues with time-varying arrivals," *Operations Research*, vol. 60, no. 6, pp. 1551–1564, 2012.

[25] H.-L. Chen, J. R. Marden, and A. Wierman, "On the impact of heterogeneity and back-end scheduling in load balancing designs," in *2009 IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2267–2275, 2009.

[26] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, pp. 157–168, 2009.

[27] M. Harchol-Balter, A. Scheller-Wolf, and A. R. Young, "Surprising results on task assignment in server farms with high-variability workloads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 287–298, 2009.

[28] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *2007 IEEE International Symposium on Workload Characterization*, pp. 171–180, 2007.

[29] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *2012 International Green Computing Conference (IGCC)*, pp. 1–10, 2012.

[30] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013.

[31] M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 204–228, 1999.

[32] N. Gautam, *Analysis of queues: methods and applications*. CRC Press, 2012.

[33] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[34] Z. Zhu, J. Bi, H. Yuan, and Y. Chen, "SLA-based dynamic virtualized resources provisioning for shared cloud data centers," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pp. 630–637, 2011.

[35] K. Wang, M. Lin, F. Ciucu, A. Wierman, and C. Lin, “Characterizing the impact of the workload on the value of dynamic resizing in data centers,” *Performance Evaluation*, vol. 85, pp. 1–18, 2015.

[36] L. Bertini, J. Leite, and D. Mossé, “Dynamic configuration of web server clusters with qos control,” in *WIP Session of the 20th Euromicro Conference on Real-Time Systems*, vol. 4, pp. 8–11, 2008.

[37] W. Whitt, “Stabilizing performance in a single-server queue with time-varying arrival rate,” *Queueing Systems*, vol. 81, no. 4, pp. 341–378, 2015.

[38] D. Xu and X. Liu, “Geographic trough filling for internet datacenters,” in *2012 IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2881–2885, 2012.

[39] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, “Data centers power reduction: A two time scale approach for delay tolerant workloads,” in *2012 IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1431–1439, 2012.

[40] 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*, vol. 40, pp. 175–186, 2012.

[41] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, “Optimality analysis of energy-performance trade-off for server farm management,” *Performance Evaluation*, vol. 67, no. 11, pp. 1155–1171, 2010.

[42] B. Guenter, N. Jain, and C. Williams, “Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning,” in *2011 IEEE International Con-*

ference on Computer Communications (INFOCOM), pp. 1332–1340, 2011.

[43] A. Gandhi, M. Harchol-Balter, M. Kozuch, *et al.*, "Are sleep states effective in data centers?," in *2012 International Green Computing Conference (IGCC)*, pp. 1–10, 2012.

[44] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 303–314, 2005.

[45] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 103–116, 2001.

[46] P. Bodik, M. P. Armbrust, K. Canini, A. Fox, M. Jordan, and D. A. Patterson, "A case for adaptive datacenters to conserve energy and improve reliability," *University of California at Berkeley, Tech. Rep. UCB/EECS-2008-127*, 2008.

[47] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing*, vol. 12, no. 1, pp. 1–15, 2009.

[48] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: Coordinated multi-level power management for the data center," in *ACM SIGARCH Computer Architecture News*, vol. 36, pp. 48–59, 2008.

[49] H. N. Van, F. D. Tran, and J.-M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," in *2009 IEEE International Conference on Computer and Information Technology (CIT)*, vol. 1, pp. 357–362, 2009.

[50] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *2010*

IEEE International Conference on Computer Communications (INFOCOM), pp. 1–9, 2010.

[51] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 221–232, 2011.

[52] S. B. Peterson, J. Whitacre, and J. Apt, "The economics of using plug-in hybrid electric vehicle battery packs for grid storage," *Journal of Power Sources*, vol. 195, no. 8, pp. 2377–2384, 2010.

[53] R. Sioshansi, "Welfare impacts of electricity storage and the implications of ownership structure," *The Energy Journal*, vol. 31, no. 2, pp. 173–198, 2010.

[54] D. Noll, C. Dawes, and V. Rai, "Solar community organizations and active peer effects in the adoption of residential pv," *Energy Policy*, vol. 67, no. 2, pp. 330–343, 2014.

[55] J. Garcia-Gonzalez, R. M. R. de la Muela, L. M. Santos, and A. M. Gonzalez, "Stochastic joint optimization of wind generation and pumped-storage units in an electricity market," *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 460–468, 2008.

[56] L. Xie, Y. Gu, A. Eskandari, and M. Ehsani, "Fast mpc-based coordination of wind power and battery energy storage systems," *Journal of Energy Engineering*, vol. 138, no. 2, pp. 43–53, 2012.

[57] S. Bose and E. Bitar, "Variability and the locational marginal value of energy storage," in *2014 IEEE Annual Conference on Decision and Control (CDC), pages=3259–3265, year=2014*.

[58] H.-I. Su and A. El Gamal, "Modeling and analysis of the role of energy storage for renewable integration: Power balancing," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4109–4117, 2013.

[59] J. Qin and R. Rajagopal, "Dynamic programming solution to distributed storage operation and design," in *2013 IEEE Power and Energy Society General Meeting (PES)*, pp. 1–5, 2013.

[60] N. Gast, D.-C. Tomozei, and J.-Y. Le Boudec, "Optimal generation and storage scheduling in the presence of renewable forecast uncertainties," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1328–1339, 2014.

[61] F. Graves, T. Jenkin, and D. Murphy, "Opportunities for electricity storage in deregulating markets," *The Electricity Journal*, vol. 12, no. 8, pp. 46–56, 1999.

[62] P. Mokrian and M. Stephen, "A stochastic programming framework for the valuation of electricity storage," in *26th USAEE/IAEE North American Conference*, pp. 24–27, 2006.

[63] J. Qin, R. Sevlian, D. Varodayan, and R. Rajagopal, "Optimal electric energy storage operation," in *2012 IEEE Power and Energy Society General Meeting (PES)*, pp. 1–6, 2012.

[64] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa, "Demand-side management via distributed energy generation and storage optimization," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 866–876, 2013.

[65] I. Koutsopoulos, V. Hatzi, and L. Tassiulas, "Optimal energy storage control policies for the smart power grid," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 475–480, 2011.

[66] L. Huang, J. Walrand, and K. Ramchandran, “Optimal demand response with energy storage management,” in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 61–66, 2012.

[67] S. Lakshminarayana, T. Q. Quek, and H. V. Poor, “Cooperation and storage tradeoffs in power grids with renewable energy resources,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 7, pp. 1386–1397, 2014.

[68] J. Qin, Y. Chow, J. Yang, and R. Rajagopal, “Online modified greedy algorithm for storage control under uncertainty,” *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 1729–1743, 2016.

[69] P. M. Van de Ven, N. Hegde, L. Massoulié, and T. Salonidis, “Optimal control of end-user energy storage,” *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 789–797, 2013.

[70] P. Harsha and M. Dahleh, “Optimal management and sizing of energy storage under dynamic pricing for the efficient integration of renewable energy,” *IEEE Transactions on Power Systems*, vol. 30, no. 3, pp. 1164–1181, 2015.

[71] Y. Xu and L. Tong, “On the operation and value of storage in consumer demand response,” in *2014 IEEE Annual Conference on Decision and Control (CDC)*, pp. 205–210, 2014.

[72] R. Wilson, “Architecture of power markets,” *Econometrica*, vol. 70, no. 4, pp. 1299–1340, 2002.

[73] K. Divya and J. Østergaard, “Battery energy storage technology for power systemsâĂŤan overview,” *Electric Power Systems Research*, vol. 79, no. 4, pp. 511–520, 2009.

[74] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, “Carbon-aware energy capacity planning for datacenters,” in *2012 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 391–400, 2012.

[75] D. P. de Farias and B. Van Roy, “The linear programming approach to approximate dynamic programming,” *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[76] S. M. Ross, *Introduction to stochastic dynamic programming*. Academic press, 2014.

[77] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995.

[78] D. P. Joseph and T. J. Tou, “On linear control theory,” *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, vol. 80, no. 4, pp. 193–196, 1961.

[79] L. Xie and M. D. Ilic, “Model predictive economic/environmental dispatch of power systems with intermittent resources,” in *2009 IEEE Power & Energy Society General Meeting (PES)*, pp. 1–6, IEEE, 2009.

[80] M. Etezadi-Amoli, K. Choma, and J. Stefani, “Rapid-charge electric-vehicle stations,” *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1883–1887, 2010.

[81] J. Whitney and P. Delforge, “Data center efficiency assessment,” *Natural Resources Defense Council, New York City, New York*, 2014.

[82] US Department of Energy, “2014 renewable energy data book,” 2014.

[83] CNT Energy and Ameren Illinois Utilities, “Ameren power smart pricing,” 2012. [Online], Available: http://www.powersmartpricing.org/how-it-works/.

[84] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad, “Opportunities and challenges for data center demand response,” in *2014 International Green Computing Conference (IGCC)*, pp. 1–10, 2014.

[85] Google.com, “Google: Environment,” 2016. [Online], Available: https://environment.google/projects/announcement.

[86] P. Palensky and D. Dietrich, “Demand side management: Demand response, intelligent energy systems, and smart loads,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[87] J. Aghaei and M.-I. Alizadeh, “Demand response in smart electricity grids equipped with renewable energy sources: A review,” *Renewable and Sustainable Energy Reviews*, vol. 18, pp. 64–72, 2013.

[88] N. Ruiz, I. Cobelo, and J. Oyarzabal, “A direct load control model for virtual power plant management,” *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 959–966, 2009.

[89] N. Gautam, Y. Xu, and J. T. Bradley, “Meeting inelastic demand in systems with storage and renewable sources,” in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 97–102, 2014.

[90] Y. Guo and Y. Fang, “Electricity cost saving strategy in data centers by using energy storage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1149–1160, 2013.

[91] Y. Guo, Y. Gong, Y. Fang, P. P. Khargonekar, and X. Geng, “Energy and network aware workload management for sustainable data centers with thermal storage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2030–2042, 2014.

[92] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[93] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[94] J. R. Birge and F. V. Louveaux, "A multicut algorithm for two-stage stochastic linear programs," *European Journal of Operational Research*, vol. 34, no. 3, pp. 384–392, 1988.

[95] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[96] D. McDaniel and M. Devine, "A modified benders' partitioning algorithm for mixed integer programming," *Management Science*, vol. 24, no. 3, pp. 312–319, 1977.

[97] T. L. Magnanti and R. T. Wong, "Accelerating benders decomposition: Algorithmic enhancement and model selection criteria," *Operations Research*, vol. 29, no. 3, pp. 464–484, 1981.

[98] G. Zakeri, A. B. Philpott, and D. M. Ryan, "Inexact cuts in benders decomposition," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 643–657, 2000.

[99] G. K. Saharidis and M. G. Ierapetritou, "Speed-up benders decomposition using maximum density cut (mdc) generation," *Annals of Operations Research*, vol. 210, no. 1, pp. 101–123, 2013.

[100] S. Trukhanov, L. Ntaimo, and A. Schaefer, "Adaptive multicut aggregation for two-stage stochastic linear programs with recourse," *European Journal of Operational Research*, vol. 206, no. 2, pp. 395–406, 2010.

[101] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, “The benders decomposition algorithm: A literature review,” *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, 2017.

[102] IBM, “Ibm ilog cplex optimization studio v12.6.3 documentation,” *International Business Machines Corporation*, 2015.

[103] G. Laporte and F. V. Louveaux, “The integer l-shaped method for stochastic integer programs with complete recourse,” *Operations Research Letters*, vol. 13, no. 3, pp. 133–142, 1993.

[104] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek, “A framework for clustering uncertain data,” *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1976–1979, 2015.

[105] PJM Interconnection, “Energy market data,” 2016. [Online], Available: http://pjm.com/markets-and-operations/energy.aspx.

[106] J. R. Birge, “The value of the stochastic solution in stochastic linear programs with fixed recourse,” *Mathematical Programming*, vol. 24, no. 1, pp. 314–325, 1982.

[107] J. Luedtke, S. Ahmed, and G. L. Nemhauser, “An integer programming approach for linear programs with probabilistic constraints,” *Mathematical Programming*, vol. 122, no. 2, pp. 247–272, 2010.

[108] X. Liu, S. Küçükyavuz, and J. Luedtke, “Decomposition algorithms for two-stage chance-constrained programs,” *Mathematical Programming*, vol. 157, no. 1, pp. 219–243, 2016.