DISCRETE OPTIMIZATION FOR SUPPLY DEMAND MATCHING IN SMART GRIDS

by

Sanmukh Rao Kuppannagari

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER ENGINEERING)

August 2018

Dedication

To my parents for their sacrifices and support.

Acknowledgments

First and foremost, I would like to express my gratitude to my advisor Dr. Viktor Prasanna for his guidance and patience. The path to this dissertation had many turns with more than a few of them leading to dead ends. It is a testament to his mentor ship that I was able to complete my PhD with making enough mistakes to be able to learn from them but not enough to demoralize me. I would also like to thank Dr. Rajgopal Kannan whose mentor ship greatly enhanced by mathematical maturity and problem solving skills. The zest with which he approaches challenging computer science problems has inspired me greatly and I enjoyed our long discussions on such problems. Moreover, I would like to thank my defense committee members Dr. Mohammed Beshir, Dr. Shaddin Dughmi and Dr. Mihailo Jovanovic for taking an interest in my work and providing with useful feedback which will help improve my dissertation and my research in future. I would also like to thank my qualifier committee members Dr. Paul Bogadan, Dr. Muhammad Naveed and Dr. Insoon Yang for their useful feedback on my qualifier proposal.

I would like to thank Prof. Gandhi Puvvada for giving me a chance to teach his course for several semesters thereby improving my teaching skills. I would also like to thank Dr. Hemangee Kapoor and Dr. Krishnamachar Sreenivasan, who had the most impact on me during my under graduation and in my decision of pursuing a PhD. I would like to thank all the EE department staff members and student workers for making my

life easier by handling all the administrative tasks, especially Kathy Kassar who would help me even after putting the “emergencies only” board, Janice Thompson who helped me get settled in Prof. Prasanna’s group and Tim Boston who helped me get settled in the department. Tim you will be missed.

Additionally, I would like to thank my friends and colleagues for all the moral support. I would like to thank Hiteshi whom I turned to during times of emotional distress, Vishnu for his encouragement and support and Sananda for all the long discussions on green buildings. I would like to thank Geoffrey for the countless discussions we had over a bowl of ramen. I would like to thank Manpreet and Surbhi for always being there for me on phone if not in person. I would like to thank all the members of USC Vidushak especially Mallika, Bhamati, Kiran, Satya, Yamini and Albin. I would like to thank people such as Krishna, Sundar etc. who were my roommates at some point or the other during my PhD for the great times I had with them. I would like to thank the members of Prof. Prasanna’s group for all the things I learned from them and all the good times I had with them. I would like to thank my friends Vipul, Megha, Somya, Raghav, Amulya and countless others for their support. And most importantly, I would like to thank Om and Suparna who were like my family away from home.

The people I would like to thank the most are my parents: Shridhar and Sailaja. My PhD would not have been possible without all the sacrifices they made to make my life easy, allowing me to solely focus on my research. I would also like to thank all my family members especially my grandparents, my brother Vikrant, my sister Tanushree, my uncles and aunts for all their love and support.

Contents

List of Tables

List of Figures

Abstract

Traditional power grids consisted of large grid operator owned generators to supply the demand of the consumers. The inertia of the large generators was able to handle any sudden demand fluctuations followed by an eventual ramp up/down of generation. However, modern smart grids, characterized by the proliferation of small sized Distributed Energy Resources (DER) such as photovoltaics, energy storage etc. lack this ability of handling demand fluctuations due to their lack of inertia. Therefore, they require proactive supply demand matching by utilizing the available advanced sensing and remote control capability.

A plethora of work has been done assuming that the nodes (supply and demand) in the grid exhibit continuous operational values (generation/consumption values). Fine grained control is required for continuous operational values which might not be economically available. Thus, supply demand matching often needs to be performed over discrete sets of operational values of the nodes making this an NP-hard problem. While heuristics (with unbounded error) and Integer Program based algorithms (computationally expensive) have been studied extensively, the application of approximation algorithms to develop fast algorithms with bounded error guarantees has been limited.

In this work, we address these limitations by developing a suite of polynomial runtime complexity approximation algorithms to perform supply demand matching in smart grids. We first model the smart grid to capture the available discrete control strategies

for each node in the smart grid and the associated costs and uncertainties. We also model practical grid constraints such as the overhead associated with switching strategies and the smart grid network capacity constraints. We then develop approximation algorithms with a variety of objectives and constraints such as minimizing curtailment error in Demand Response, minimizing cost of supply demand matching, fairness etc. and theoretically provide the optimality guarantees. In addition to the theoretical analysis, we perform practical evaluations of the algorithms and show that our supply demand matching algorithms provide solutions which are close to optimal in a small amount of time.

Chapter 1

Introduction

1.1 Motivation

Electrical power grids have undergone a drastic transformation since the 1970s in terms of both scale and complexity [69]. Technological advances such as the use of bi-directional AMI meters, allowing real time remote monitoring and control, have transformed them into smart grids [48].

Continuous matching of supply and demand is the most critical grid operation. The electricity demand of the consumers fluctuates throughout the day. If the fluctuating demand is not matched using an equal amount of supply, it can lead to an increase (if supply is more than demand) or decrease (if demand is more than supply) in the grid frequency. If the variation in the frequency is not contained quickly, the stability of the grid is severely compromised with potential over-voltages or blackouts.

Traditional power grids are characterized by the presence of a centralized grid operator owning large generation units such as steam or water powered turbines to meet the demand. Any fluctuation in frequency due to demand variability is offset by the inertia of the turbines, with a loss in its kinetic energy. The grid operator can then increase or decrease the rotational speed of the turbines to match the demand. The larger the size of the generation units, the larger is the inertia. Larger inertia provides the ability to offset fluctuations for longer times. This allows the grid operators ample amount of time to ramp up/down the generation.

Technological advances in modern smart grids have enabled rapid proliferation of small sized consumer owned Distributed Energy Resources (DER) into the grid such as Photovoltaics (PV), Energy storage, fuel cells etc. For example, the adoption of distributed solar energy has increased dramatically due to the falling cost of solar PVs. The installed prices of U.S. residential and commercial PV systems declined 5-7% on average during 1998-2011 [6]. As per the DoE SunShot vision document, solar generated power is expected to grow to 14% of the total power supply in 2030 and 27% by 2050 [67]. As a result, the energy generation in the smart grids has decentralized.

The increase in DERs has significantly complicated the operation of supply demand matching due to the following reasons:

- Generation patterns of renewable energy based DERs are highly influenced by the weather conditions. Hence, in addition to the variability in demand, variability in supply needs to be considered.

- Grids where a major portion of the demand is met using small sized DERs, have low inertia to handle fluctuations. Thus, less time is available to ramp up/down the supply.

- The distributed nature of DERs increases the complexity of control as the grid operator needs to consider constraints such as cost optimality, fairness, network constraints etc. for DER scheduling.

- In addition to supply, the proliferation of controllable loads has enabled consumer participation using techniques such as Demand Response (DR), thereby providing a great opportunity as well as challenge for the operation of supply demand matching.

Hence, sophisticated algorithms which pro-actively perform supply demand matching in each interval - where an interval is a grid operator determined period of time -

by controlling both the demand and supply are required to address the challenges listed above.

1.2 Current Technology

A plethora of theoretically sound polynomial runtime complexity optimization algorithms have been developed by the smart grid community to perform supply demand matching in a cost effective manner [71]. However, these algorithms assume that the supply and demand nodes exhibit continuous operational values i.e. the generation value from a supply node or a consumption value from a demand node can take any value between its minimum and maximum. Using this assumption, the problem is formulated as linear or convex optimization problem [20] which admit fast polynomial runtime algorithms which provide optimal solutions using methods such as interior point [20]. However, the limitations of these works is that the assumption of continuous operational values requires fine grained control over the nodes which is not economical and might not even be feasible in certain scenarios.

1.3 Motivation for Discrete Operational States

To motivate the need for modeling and optimizing on discrete operational states, we consider a toy example. Consider a warehouse which consists only of 100 dimmable LED bulbs each consuming a maximum power of 1 W. We assume that using a single $O(1)$ time complexity operation, the consumption value of an LED can be set to any value between 0 and 1 W. The minimum consumption of the warehouse is 0 W, when all LEDs are set to 0 and the maximum is 100 W, when all LEDs are consuming maximum power (Figure 1.1). Hence, to obtain consumption values which are continuous in the

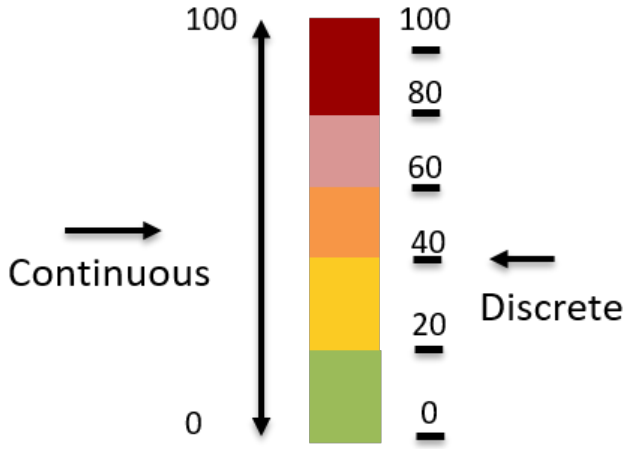


Figure 1.1: Continuous Operational Values: Node can take any operational value between 0 and 100. Discrete Operational Values: Node can only take the values 0, 20, 40, 60, 80 and 100.

range 0 to 100 for the warehouse, we need 100 different switches to control LEDs. Assuming identical LEDs, this requires 100 control signals.

Now consider the alternate scenario where a set of 20 LEDs are connected to a single switch. Consumption values of 0, 20, 40, 60, 80 and 100 can be obtained under such scenario using only 5 switches. Even if we assume each set to be unique, the number of control signals required are $2^5 = 32$ which is much less than 100.

In a real world grid, cost of installation is a major factor for a supply or demand node. Hence, even though it might be technically feasible to obtain continuous operational values, it might not be economically viable. Therefore, algorithms for supply demand matching need to operate efficiently over operational values for each node which form a set of discrete values.

1.4 Challenges in Optimizing over Discrete Operational States

The discrete nature of the operational values transforms the problem of supply demand matching into a NP hard problem (see Section 3.6 for a formal proof). Thus, no optimal polynomial runtime complexity algorithm can exist unless P = NP. In the worst case, a brute force search enumerating all possible solutions, which is exponential in the input size might be required. See Figure 1.2. This approach is followed by current techniques which use Integer Program based formulations. Thus, they provide computationally expensive optimal solutions. This is clearly not possible for grid operations where strict timing constraints need to be adhered.

To address the computational complexity, several works have focused on developing fast heuristics. Such heuristics do not provide any guarantee on optimality or on the worst case performance. They can incur unbounded errors in the objective value of the solutions they produce. Power grid operations require high reliability and hence such heuristics cannot be relied upon due to their uncertainty.

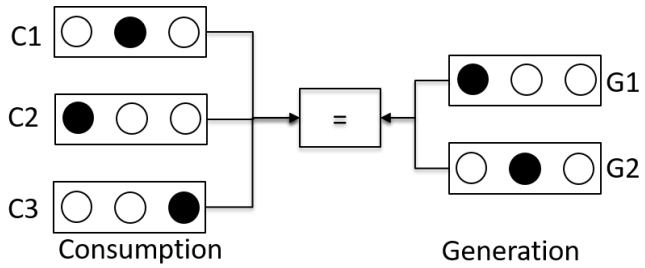


Figure 1.2: Supply demand matching in the above figure requires enumerating $3^5 = 243$ combinations in the worst case

1.5 Application of Approximation Algorithms

One of the most significant contribution of this work is to model the various supply demand matching problems as variants of packing problems. Several packing problems such as 0-1 knapsack, subset sum problem etc. admit dynamic programming based Fully Polynomial Time Approximation Schemes (FPTAS) or Polynomial Time Approximation Algorithms (PTAS) [72]. Given an accuracy parameter ϵ, an FPTAS is an algorithm with a polynomial runtime complexity in the input size and $\frac{1}{\epsilon}$ which outputs a solution with an objective value which is within $(1 \pm \epsilon)$ of the optimal value. PTAS provides similar guarantees, however, its runtime complexity is allowed to be exponential in $\frac{1}{\epsilon}$. Modeling the supply demand matching problems as variants of packing problems allows us to use the approximation techniques used for the latter problem.

Consider the following simple problem: We are given a set of supply nodes with node i exhibiting exactly two operational values, 0 and s_i with associated costs 0 and c_i^s. Similarly, we are given demand nodes with operational values, 0 and d_i with associated costs 0 and c_i^d. Now if we assume constant demand and control the supply to meet the demand and ignore the costs, it can be modeled as the subset sum problem. If we consider the costs, the problem can be modeled as the knapsack problem. If we consider costs and control both supply and demand, the problem becomes more complicated, but still admits a dynamic programming based FPTAS.

The problems that we consider in this work are higher in complexity as they optimize over several time periods, with each node equipped with more than two operational values and they incorporate several constraints and objective values. However, by carefully analyzing the structure of the problem, we are able to leverage several techniques such as rounding, LP relaxation etc. to develop approximation algorithms and provide guarantees both in terms of the achieved objective values and the constraint violations.

1.6 Thesis Statement

Rapid proliferation of Distributed Energy Resources (DER) in modern power grids, also known as "smart grids", enabled by advancement in sensing and monitoring technology has introduced unique challenges as well as opportunities for optimal grid operations. Proactive supply demand matching by controlling the DERs and loads is imperative to ensure reliable grid operations. Moreover, as fine grained control is not available economically, supply demand matching algorithms need to optimize on discrete sets of operational values. Our research focuses on modeling the modern smart grids consisting of controllable loads, DERs and infrastructure nodes with associated capacity to enable development of discrete supply demand matching algorithms. Our goal is to develop polynomial runtime complexity algorithms for supply demand matching with bounds on the worst case performance both in terms of the achieved objective values and the constraint violations. Our algorithms will enable the development of a smart grid control framework which will facilitate DER proliferation and reduce the cost of grid operations while ensuring reliability.

1.7 Research Contributions

In this work, we develop fast polynomial runtime complexity approximation algorithms for a variety of objective functions and constraints. We provide sound theoretical analysis of optimality guarantees supplemented with practical evaluations to show that our algorithms provide solutions which are close to optimal in a small amount of time. Specifically, our contributions are as follows:

1. We develop a smart grid model to be utilized by the supply demand matching algorithms to optimize over discrete set of operational values. Additionally, we

model the network of the smart grid and capture the capacity constraints of the infrastructure such as transformers and feeders.

2. We develop a suite of supply demand matching approximation algorithms for Demand Response. This is the scenario in which the supply is constant and the demand needs to be controlled (curtailed) such that its value is less than the available supply. We develop algorithms with several objectives such as coarse grained curtailment across the entire horizon of the Demand Response event (Traditional Demand Response) or equitable distribution across all the intervals in the event (Sustainable Demand Response). We also consider several practical constraints such as the overheads incurred by each node in switching strategies and fairness of curtailment across all the nodes.

3. We develop a suite of supply demand matching approximation algorithms under the scenario when both supply and demand are controllable with the objective of minimizing the cost. We again consider several practical constraints such as fairness and network capacity constraints.

4. We model storage to incorporate into the algorithms developed without significantly affecting the computational complexity.

5. We develop algorithms to perform supply demand matching under prediction uncertainties.

1.8 Dissertation Outline

The rest of the dissertation is organized as follows: We discuss the current works that focus on supply demand matching in Chapter 2. We then discuss our smart grid model for discrete supply demand matching optimizations in Chapter 3. This is followed by a

discussion of a suite of algorithms developed for performing Demand Response, where supply is constant and load is controllable in Chapter 4 and then the algorithms developed to perform cost optimal supply demand matching with both controllable loads and supply are discussed in Chapter 5. Discrete supply demand matching under prediction uncertainty is discussed in Chapter 6. Finally, we conclude the dissertation with a discussion of future works in Chapter 7.

Chapter 2

Related Work

As discussed in the previous chapter, several works have focused on developing theoretically sound algorithms to ensure reliability in smart grids by matching supply and demand. However, the focus of such works has mostly been on the nodes with continuous operational values. We refer the readers to the survey paper [71] which performs an exhaustive study of the DR techniques developed in the literature. As our focus in this work is on optimizing over discrete sets, we limit our literature survey to similar works. The works targeting discrete operational states either develop computationally expensive Integer program based solutions or develop fast heuristics with no error guarantees. In this chapter, we perform a literature review of such works targeting discrete operational states. We also discuss some works which applied approximation algorithm techniques to solve some specific problems related to supply demand matching in smart grids.

2.1 Earlier Works on Demand Response

Significant literature exists addressing the challenges, solutions, implementations and estimation methodology for calculating the energy savings for Demand Response (DR) [11, 51]. Early works focused on DR scheduling for individual residential cases [45] or household appliances [34]. These approaches are not scalable to smart grids.

Traditionally, DR algorithms have focused on targeting customers based on aggregate consumption data, relying on customer selection using billing data or surveys [43, 49], employing dynamic programming techniques for load management and minimizing peak load over a period [26], particle swarm optimization based techniques [61] and game theoretical solutions constrained by real time pricing [23] and customer comfort levels [17]. However, with data available from smart meters, work such as [64] show that such approaches are very inaccurate. The actual consumption data over a period differs significantly from the data obtained from surveys or billing cycles. Moreover, the selection is done oblivious to the distribution of load throughout the day. Therefore such approaches contribute little to reducing the peak energy consumption and distributing it over other periods.

2.2 Tractable Demand Response Solutions for Discrete Curtailment

In order to tractably solve the problem of achieving load curtailment using discrete curtailment values, stochastic optimization algorithms have been developed in works such as [38] and [24]. However, such works assume the availability of a large number of nodes in the grid. For example, the authors in [38] propose a stochastic knapsack based algorithm for selecting customers to maximize the probability the desired curtailment value is achieved over the period of the entire DR event while limiting the utility's cost. The algorithm relies on the central limit theorem to assume the joint customer response is normally distributed and thus is conditioned on the assumption that there are a large number of customers from whom a subset can be selected. For example, in order to achieve high reliability ($> 95\%$) for large curtailment targets, the algorithm requires around $N = 2000$ customers which increases the computation cost. Two heuristics are

proposed but the approximation bounds are dependent on the ratio of customer demand response variances which could be arbitrarily large. Although fine grained data is available, the selection algorithm focuses on aggregating the curtailment over a time interval. However, as mentioned before, such an approach can aggressively curtail load in some intervals and concentrate peak in other intervals. Moreover, by maximizing the probability of exceeding the desired curtailment, there might be cases where they overcompensate leading to reduced benefits to the utility.

Techniques such as [57], [41], [76] and [58] develop fast algorithms which can have arbitrarily large errors in the objective function (utility maximization, cost minimization etc.). Authors in [57] develop a genetic algorithm based heuristic while [76] presents a heuristic based on change making. The algorithm developed in [58] uses Linear Programming whose solutions need to be rounded to integral values and can have large errors (unbounded integrality gap). The notion of achieving sustainable DR over a peak period divided into subintervals was proposed in [76] using a change making heuristic to evenly distribute curtailment over intervals. However, a detailed analysis shows that it achieves consistency between intervals without reference to the target leading to unbounded errors which is also demonstrated by our experimental results. An automated framework employing this heuristic is developed in [77].

2.3 Optimal Demand Response using Integer Programming

The other approach is to provide computationally expensive exact solutions, for example, [28], [46], [18], [75], [25] and [16], where the authors use Integer/Mixed Integer Linear/Non-Linear Programming for their algorithm. The authors in [65] develop a quadratic programming formulation for Demand Response which is then reduced into

a distributed algorithm. The paper assumes the availability of continuous curtailment values. However, the practical DR implementation in USC's smartgrid allows only discrete curtailment values which forms the basis of modeling them as such in our work. This constraint makes the problem more complicated as the linear/quadratic program gets converted into an integer linear/quadratic program.

2.4 Solar Curtailment

Load Curtailment techniques are ineffective when supply due to solar PVs exceeds the demand. If this is left unmitigated, it causes over-voltages in the system leading to failures. Several works perform reactive solar curtailment in response to rising voltage. VVO [52] increases reactive power to lower the voltage due to real power while iPlug curtails the solar energy input to the grid by redirecting it to charge storage or coordinate with local demand ramp-up resources [56]. The authors in [40, 70, 63] achieve continuous curtailment from solar PVs by running them at voltages other than the Maximum Power Point (MPP). This requires fine grained control of the solar panels. For some scenarios, fine grained control might not be available due to limitations of inverter technology. Our work addresses such scenarios through a curtailment model that handles a discrete set of curtailment values and provides bounded polynomial time approximations for achieving discrete curtailment targets. As mentioned in [33], discrete solar curtailment can be performed by simply disconnecting individual PV modules using the micro-inverters installed at PV installations. As opposed to the technique developed in [33], which is reactive to over-voltages, we perform proactive solar curtailment. Works such as [47] develop a scheduling algorithm which determines the schedule of generation from various sources for the entire day. The sources include a mix of renewable and non-renewable sources and the objective is to minimize cost. The resulting

optimization problem is complex as it has to make a decision for each node for each interval of the entire day. This makes it difficult to scale this algorithm for large distribution grid or micro-grids with rooftop PVs installed.

2.5 Applications of Approximation Algorithms for Supply Demand Matching

As discussed before, the application of approximation algorithms for the problem of supply demand matching is limited. Constant factor approximation algorithms based on strip packing have been developed for peak demand reduction [68, 10, 73, 54]. The problem of strip packing is defined as follows: Given a set of axis aligned rectangles, pack them into fixed width strip while minimizing its height. In smart grid context, the rectangles denote the total energy consumed by each individual appliance with the height of the strip denoting the maximum power (energy) required at any given time. Essentially, these algorithms distribute a fixed amount of energy temporally to minimize the peak consumption. Peak reduction is one special case of the more generalized supply demand matching problem.

Chapter 3

Smart Grid Modeling for Discrete Supply Demand Matching Optimizations

In this work, we focus on a specific type of smart grid known as *discrete microgrids*. Discrete microgrids are characterized by the presence of a centralized controller which is responsible for the grid operations. The various supply and demand nodes in such smart grids exhibit discrete operational values i.e. at any given time step, the feasible region of the operational value the node can exhibit forms a discrete set. Moreover, due the presence of decentralized generation using small sized Distributed Energy Resources (DER), such grids have very low supply inertia, thereby making proactive supply demand matching imperative.

In this chapter, we discuss the smart grid model that we developed to model discrete microgrids. The smart grid model captures important grid characteristics such as the number of supply and demand nodes, the available controllable strategy space and their associated costs, the costs associated with switching strategies, the smart grid network and the capacity constraints of the various infrastructure nodes such as transformers, feeders etc. This model is then utilized by the algorithms developed in the subsequent chapters for supply demand matching.

3.1 Supply Demand Matching via Controllable Curtailment Strategies

3.1.1 Smart Grid

The Smart Grid that we consider in this work consists of several demand nodes: consumers of electricity, and supply nodes: electricity producers. The supply nodes that we consider in this work are the customers who have solar PVs installed. A single consumer can act both as a demand node and a supply node. In this case, we model two different nodes for the consumer: one for demand and one for supply. We assume that the smart grid has a high PV penetration i.e. the supply from solar PVs under normal weather conditions meet the demand of the consumers for most of the day. We assume that during night or during extremely unfavorable weather conditions, conventional sources of electricity are used to meet the demand.

3.1.2 Smart Grid Node

As discussed in the previous section, a smart grid node can be a demand node or a supply node. We assume that for each node at any given interval, the operational value when the node is not being controlled is the maximum output for that interval. Any control action is taken in terms of curtailment i.e. a reduction in the operational value. In Sections 3.1.4 and 3.1.5, we discuss how to obtain these output values. We use the term curtailment strategy to denote a curtailment control action.

Consider the node shown in Figure 3.1. At a given time t, if no control action is taken, or in other words, default strategy of 0 is followed, the operational value of the node is $O(t)$ with an associated curtailment value of 0. Following strategies 1, 2 or 3 results in an operational value of $\alpha_1 O(t)$, $\alpha_2 O(t)$ and $\alpha_3 O(t)$ respectively, where

$0 \leq \alpha_3 \leq \alpha_2 \leq \alpha_1 \leq 1$. The corresponding curtailment values are $(1-\alpha_1)O(t)$, $(1-\alpha_2)O(t)$ and $(1-\alpha_3)O(t)$ respectively. The shaded portion below show the uncontrollable operational state. For example, for a demand node, this could denotes the baseline demand which is imperative to be met. For some nodes, it is possible that the uncontrollable operational state area is empty. For example, for solar PVs it is possible to curtail the entire generation by disconnecting the PV from the grid.

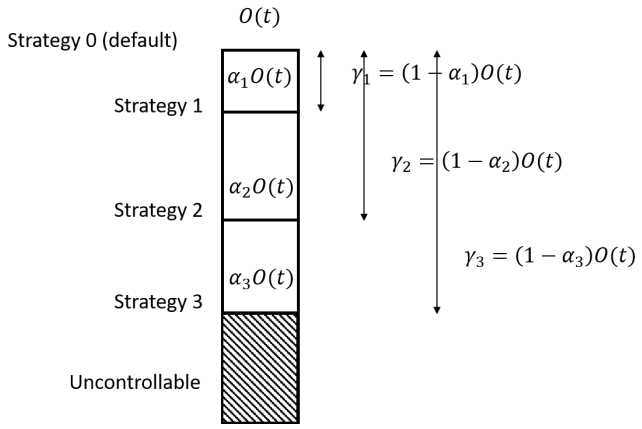


Figure 3.1: Smart Grid node with 3 + 1 (default) available curtailment strategies.

3.1.3 Smart Grid Controller

We assume there exists a centralized grid operator/controller with the capability of remotely switching a node into a curtailment strategy. One simple way in which this capability can be achieved is as follows: Each node can be equipped with a controller with the capability to receive and respond to some IP based smart grid specific communication protocol such as Active Network Management (ANM). The centralized grid operator/controller can then send signals for switching curtailment strategies using the same protocol.

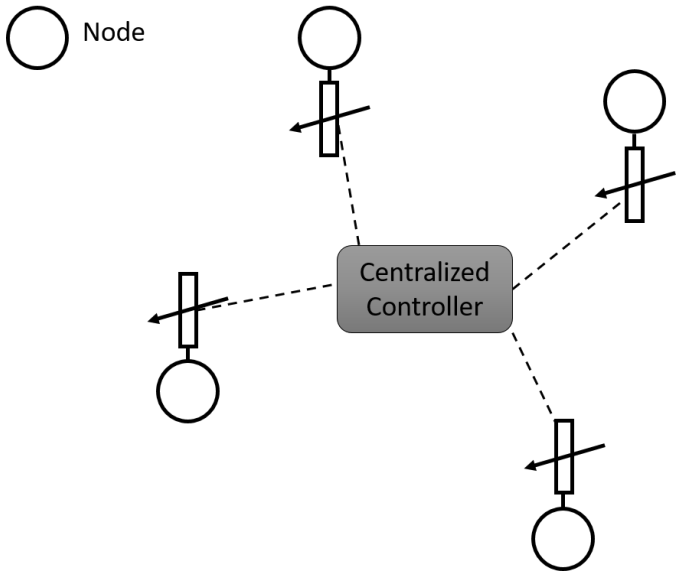


Figure 3.2: Smart Grid nodes with centralized controller communicating via protocol such as ANM

3.1.4 Demand Curtailment Strategies

The demand curtailment model considered in this work is based on a real world Demand Response implementation in USC's Campus Microgrid [36]. In the Smart Grid, each demand node is associated with several demand curtailment strategies. Examples of strategies include Global Zone Temperature Reset (GTR), Variable Frequency Drive Speed Reset (VFD), Equipment Duty Cycling (Duty) and their combinations [36]. Each curtailment strategy for each node in a given time interval exhibits a discrete curtailment value i.e. the reduction in the consumption from the maximum output due to the adoption of this curtailment strategy. This complicates the problem as Linear Programming based techniques, which are both fast and optimal can no longer be used. This value can be predicted using algorithms mentioned in [12]. Each node is also associated with a default curtailment strategy of curtailment value 0. Hence, if a node has no curtailment strategy available or does not participate in demand curtailment, we assume that it

follows the default strategy. Demand curtailment is known in the literature as Demand Response [11].

3.1.5 Solar Curtailment Strategies

Each supply node i.e., a node with PV installation in the Smart Grid consists of several solar panels (each solar panel is called a module). Traditionally, modules are connected in series to an inverter which in turn is connected to the grid. However, this topology affects the efficiency of the PV system as the inverter conditions the output according to the poorest performing module [33]. Therefore newer designs, in which each module is independently connected to a micro-inverter are becoming increasingly popular [27].

Technically, each micro-inverter of a PV installation is an independent grid connected generator, turning the PV installation into a segmented generator with discrete generation output. The maximum output of the PV installation will occur when all the solar panels are allowed to feed into the grid. However, at any given time, micro-inverters can be configured such that only a subset of PV modules are connected to the grid. Our objective is to exploit this capability by controlling the micro-inverter configuration and enabling discrete curtailment of supply. We refer the reader to [33] for more details on utilizing micro-inverters for solar curtailment. Note that the technique developed in [33] is a reactive technique which reacts to voltage increase and requires high frequency voltage sampling. Our technique is a proactive technique which avoids an increase in voltage by reducing supply in advance.

As discussed in [33], the curtailment can be achieved using module tripping by executing it as an inverter restart. Module tripping is a form of generator tripping where the loss of capacity is less than 100%. Generator tripping is recognized as the most effective way of resolving transient stability issues [33]. Thus, in this work we do not explicitly consider any stability issues related to solar curtailment.

3.1.6 Curtailment Cost

Each curtailment strategy for each node is associated with a cost value as curtailment leads to a loss in utility. These costs are determined by the grid operator to reflect the loss. Typically, the costs are some function of the curtailment value e.g., if a node, by following a strategy curtails γ, then the cost of this strategy will be $f(\gamma)$, where f is some function determined by the grid operator. Linear and quadratic functions are commonly used cost functions in grid operations. The objective of our framework is to minimize cost while performing supply demand matching.

3.1.7 Fairness in Curtailment

Curtailment performed by any nodes leads to a loss in utility. Hence, we should ensure that no single node should be disproportionately penalized while achieving a certain global curtailment target for supply demand matching. To ensure that certain nodes of the grid are not unfairly penalized, we assign a budget value B_b with each node b. We also assign a lower bound of $\alpha_b B_b$ on the curtailment value. A fairness constraint is added into the algorithms which ensures that no node curtails more than its budget value.

3.1.8 Incorporating Strategy Switching Overheads

In order to incorporate strategy switching overheads, for a node b, we define a function χ_b to model the cost of switching between allowable strategies. The purpose of the cost is to disallow frequent strategy switching when ramp time is high. If the node is allowed to switch from strategy j to k, then $\chi_b(j,k) < \infty$ denotes the cost of switching, else $\chi_b(j,k) = \infty$. Note that χ_b can be represented using a 2D square matrix whose i,j entry represents the cost of switching from strategy i to j. This ensures that a single call of

χ_b requires $O(1)$ amount of time. For a node b, we limit the cost incurred in switching strategies by τ_b.

For example, consider the state transition diagram shown in Figure 3.3. The following matrix represents the strategy switching overheads:

$$\begin{matrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & \infty \\ 2 & \infty & 0 & 1 \\ 3 & \infty & \infty & 0 \end{matrix} \tag{3.1}$$

$$\chi_b(j,k) = |k-j|$$

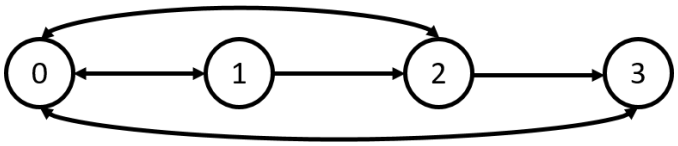


Figure 3.3: Strategy transition diagram for a node with 3 + 1 (default) strategy. A bi-directional arrow signifies that transition can occur in both directions.

3.2 Supply Demand Matching Framework

Mitigating supply-demand mismatch within tight timing constraints is critical for smooth operation of a smart grid. As shown in Figure 3.4, during several intervals of the day, such as regions 1 and 3, the demand of the consumers can exceed the solar supply. This can cause blackouts in the grid. Demand curtailment strategies need to be adopted during such intervals to avoid blackouts. The other extreme is shown using region 2 in Figure 3.4. These are the intervals in which the supply due to solar PVs exceeds the demand. This can cause over-voltages in the grid leading to the tripping

of fault prevention devices [33]. Under this scenario, supply curtailment strategies are required.

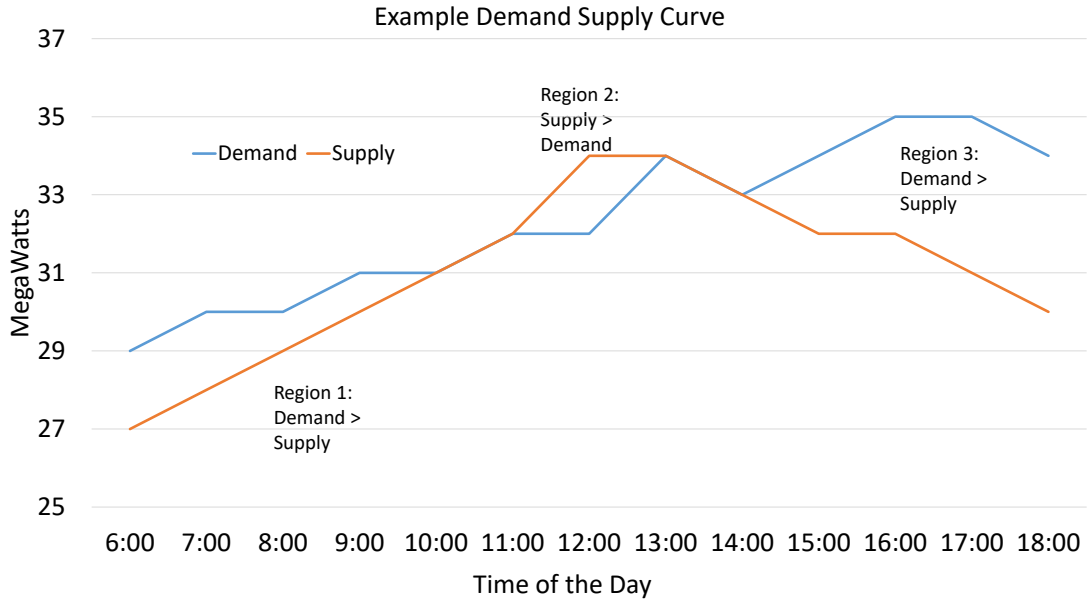


Figure 3.4: Example Supply Demand Curve

We develop a generalized framework which performs supply demand matching by selecting load or supply curtailment strategies. We define supply demand matching horizon as the time horizon during which the framework is used. Now, each supply node of the smart grid can be associated with a generation prediction model such as ARIMA+ANN ensemble [53] to output the generation value. Supply curtailment values corresponding to the strategies can be obtained as discussed in Section 3.1.5. Similarly, each demand node can be associated with a demand prediction model such as ARIMA [14] to output the demand value under default strategy. Each node can also be associated with a curtailment prediction model [12] which determines the curtailment obtained by following curtailment strategies. Determining the best prediction model for each node is a separate research topic and is out of the scope of this work.

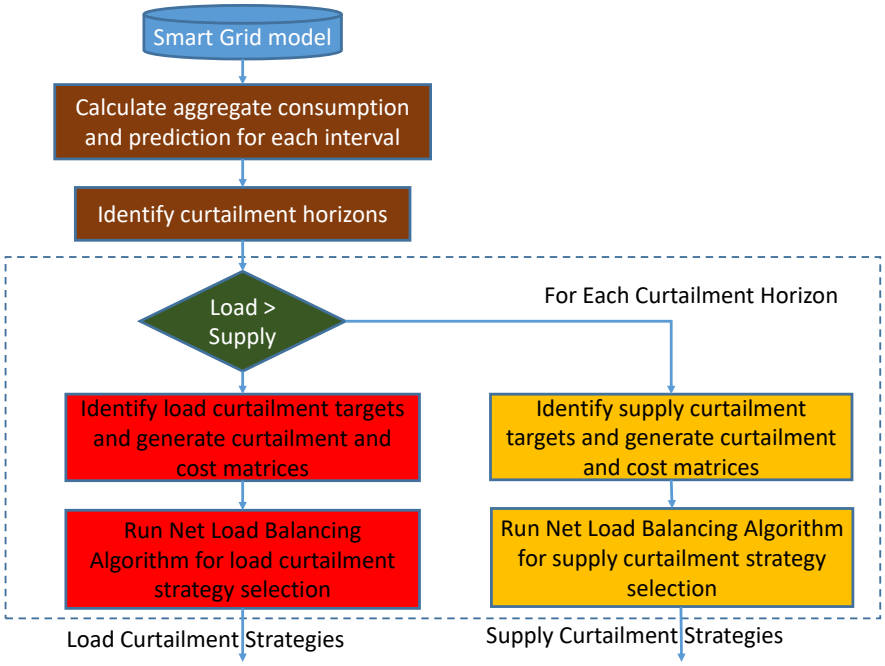


Figure 3.5: High Level Overview of Single Control Mode

Our supply demand framework runs in two different modes differentiated by the nodes that are being controlled in any given time interval of the supply demand matching horizon. The two modes are as follows:

1. **Single Control Mode:** In this mode, at any given time interval, either demand nodes are controlled or the supply nodes, but not both. A high level overview of single control mode is shown in Figure 3.5. The framework determines the aggregate load and supply for each interval in the supply demand matching horizon. It then identifies a list of load curtailment horizons and supply curtailment horizons and the respective curtailment targets. A curtailment horizon is defined as a period of time during which either demand is higher than the supply requiring a demand curtailment or vice-versa. For each interval of every curtailment horizon, the supply demand matching framework uses the curtailment prediction models to determine discrete curtailment values for each node. It also determines the cost

values associated with them. Then, for each curtailment horizon, it runs one of the supply demand matching algorithms developed in this work. The algorithm to run is pre-determined by the grid operator. The algorithm returns the curtailment strategies to be followed by each node in each time interval of each curtailment horizon.

2. **Dual Control Mode:** In this mode, at any given time interval, both the demand and supply nodes are controlled. In each interval, the framework determines the cost of achieving all possible supply demand matching values and chooses the one which satisfies all the constraints with minimum cost. See Figure 3.6.

The simplicity of single control mode allows us to optimize over several intervals, which is not possible in dual control mode. However, dual control mode can lead to more optimized solutions within a single interval by controlling both the supply side and the demand side.

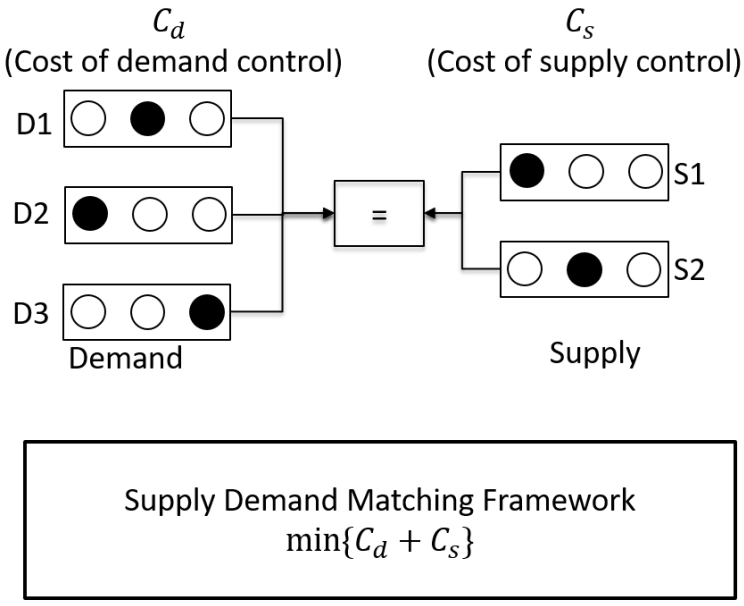


Figure 3.6: High Level Overview of Dual Control Mode

3.3 Modeling Smart Grid for Curtailment Selection

Now we mathematically define the smart grid model. As per our model, the Smart Grid consists of M nodes. For each node, there are N curtailment strategies available. The nodes are split into two disjoint sub-sets $\mathcal{S}$: supply nodes and $\mathcal{D}$: demand nodes. Let M_s be the number of supply nodes and M_d be the number of demand nodes. Let T be the number of time intervals in the curtailment horizon i.e., the intervals during which we schedule the curtailment. Each supply node $b_s \in \mathcal{S}$ is associated with a maximum supply value $S_{b_s}(t) \forall t \in \{1, \ldots, T\}$. Similarly, each demand node $b_d \in \mathcal{D}$ is associated with a maximum load value $L_{b_d}(t) \forall t \in \{1, \ldots, T\}$. We are given time varying curtailment matrices $\gamma_s(t) \imath R^{M_s \times N}$ and $\gamma_d(t) \in R^{M_d \times N}$ with elements $\gamma_{s,bj}(t)$ and $\gamma_{d,bj}(t)$ denoting the discrete supply and demand curtailment obtained by node b following curtailment strategy j at time $t \in \{1, \ldots, T\}$ respectively. For each time t, we are also given a cost matrices $C_s(t) \in R^{M_s \times N}$ and $C_d(t) \in R^{M_d \times N}$ where $c_{s,bj}(t)$ and $c_{d,bj}(t)$ denotes the cost associated with node b following supply and demand curtailment strategy j respectively. Let $X_s(t)$ and $X_d(t)$ be the decision matrices. Elements $x_{s,bj}(t) = 1$ if node b follows supply curtailment strategy j and 0 otherwise. Similarly, $x_{d,bj}(t)$ = 1 if node b follows demand curtailment strategy j in interval t and 0 otherwise. If the framework is being used in single control mode, for each interval t, we are given a curtailment target Γ_t calculated by taking the difference between the aggregate supply and demand. Γ_t represents the desirable curtailment target for each period, however, it might be exceeded. In order to limit wasteful curtailment, we are also given Γ, which denotes the upper bound on the achieved curtailment in the curtailment horizon. The notations used in the following sections are summarized in Table 3.1.

Table 3.1: List of Variables in the Models

Variable	Meaning
M_s	Number of supply nodes
M_d	Number of demand nodes
N	Number of curtailment strategies
T	Number of time intervals in curtailment horizon
$\gamma_{s,bj}(t)$	Curtailment achieved by supply node b following curtailment strategy j at time t
$\gamma_{d,bj}(t)$	Curtailment achieved by demand node b following curtailment strategy j at time t
$c_{s,bj}(t)$	Cost of supply node b following curtailment strategy j at time t. Essentially, cost associated with $\gamma_{s,bj}(t)$
$c_{d,bj}(t)$	Cost of demand node b following curtailment strategy j at time t. Essentially, cost associated with $\gamma_{d,bj}(t)$
$x_{s,bj}(t)$	0-1 decision variable which denotes whether supply node b should follow (1) strategy j at time t or not (0)
$x_{d,bj}(t)$	0-1 decision variable which denotes whether demand node b should follow (1) strategy j at time t or not (0)
Γ_t	Curtailment target for interval t
Γ	Upper bound on the curtailment achieved in the curtailment horizon
$\alpha_b B_b$, B_b	Lower bound and upper bound on the curtailment budget for node b
$\mathcal{S}, \mathcal{D}$	supply and demand node sub-sets
$\mathcal{F}$	Set of LV feeders
$\mathcal{T}$	Set of transformers
$\mathcal{T}(tx)$	Set of all nodes connected to transformer tx
$\mathcal{F}(f)$	Set of all nodes connected to feeder f
$\mathcal{F}^{tx}(f)$	Transformer connected to feeder f
Cap_{tx}	Maximum solar capacity of distributor connected to transformer tx
Cap_f	Maximum capacity of power flow from transformer to sub-station via feeder f
$S_b(t)$	Maximum supply for a supply node b in time interval t
$L_b(t)$	Maximum load (demand) for a demand node b in time interval t

3.4 Modeling Smart Grid Network

The network model of the Smart Grid that we consider in this work is adapted from the one used in [40] and is illustrated using Figure 3.7. Power from the high voltage transmission network is fed into a sub-station, where the voltage is stepped down to be

delivered to the Smart Grid. Several Low Voltage (LV) Feeders originate from the sub-station and terminate in transformers which further steps down the voltage to deliver to residential consumers using distributors.

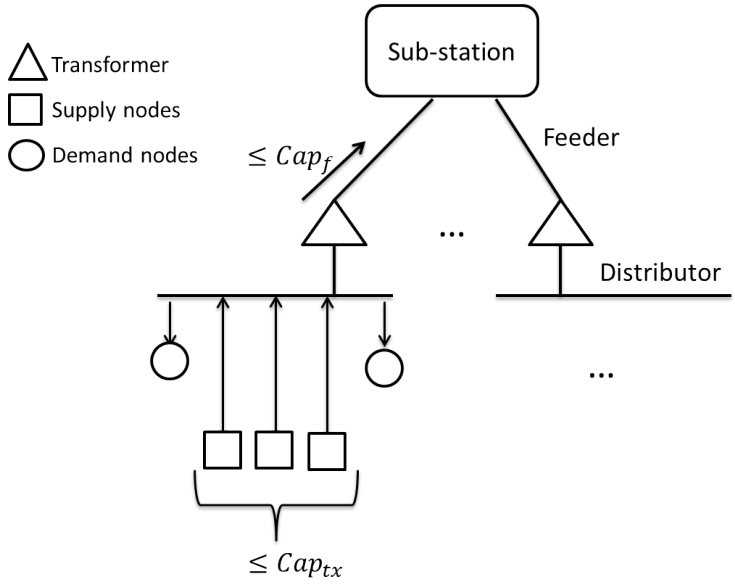


Figure 3.7: Smart Grid Network Model

In a smart grid with a high PV penetration, several consumers with PV installations can exhibit surplus supply which gets redistributed among the loads connected to the same transformer using the shared distributors. We also assume that any remaining surplus can be supplied back through the LV Feeders to the sub-station and eventually to the other transformers. We assume an LV Feeder capacity limit on the surplus that can be supplied back. We also assume that each distributor has an upper limit on the power that can be fed into it from the solar supply. This limit could be artificially applied by the utility to limit the generated power to ensure reliable grid operations as the current grids are not planned for bi-directional power flows. Typically, transformers have a capacity rating of the amount of power that can flow in each direction i.e. from LV feeder to distributor and vice-versa. We assume that this rating is more than the capacities of the LV feeder and the distributor and hence its modeling is redundant.

Note that we do not assume any limits on the amount of power that can flow from a sub-station to a transformer using the LV feeders and from a transformer to the loads using the distributors. This is a reasonable assumption as a smart grid infrastructure is typically planned to ensure uninterrupted supply to all the load consumers. The non-zero capacity constraint of LV-feeders and the solar input capacity constraint of the distributors are the additional constraints added to our network model when compared with the one used in [40].

Let $\mathcal{F}$ denote the set of the LV-feeders and let $\mathcal{T}$ denote the set of all the transformers in the Smart Grid. The capacity of a feeder f is Cap_f and that of a distributor connected to transformer tx is Cap_{tx}. The set of all the nodes connected to a transformer $tx \in \mathcal{T}$ is denoted using $\mathcal{T}(tx)$. Note that $\mathcal{T}(tx) \cap \mathcal{S}$ and $\mathcal{T}(tx) \cap \mathcal{D}$ denote the supply and demand nodes connected to the transformer tx respectively. The set of all the nodes fed by an LV-feeder f is denoted using $\mathcal{F}(f)$. This contains all the nodes connected to the transformer which is connected to the sub-station by the LV-feeder. $\mathcal{F}(f) \cap \mathcal{S}$ and $\mathcal{F}(f) \cap \mathcal{D}$ represent the supply and demand nodes fed by the feeder f. We also use the notation $\mathcal{F}^{tx}(f)$ to represent the transformer connected to a feeder f.

Now, the total solar supply input to the distributor connected to a transformer should be within its capacity. Hence,

$$Sup_{tx}(t) = \sum_{b_s \in \mathcal{T}(tx) \cap \mathcal{S}} (S_{b_s}(t) - \sum_{j=1}^{N} \gamma_{s,b_s j}(t) x_{s,b_s j}(t)) \; \forall tx \in \mathcal{T}, \forall t \in \{1, \ldots, T\} \quad (3.2)$$

$$Sup_{tx}(t) \leq Cap_{tx} \; \forall tx \in \mathcal{T}, \forall t \in \{1, \ldots, T\} \quad (3.3)$$

Moreover, the total surplus power injected from the transformer into the sub-station via a feeder should be within the capacity of the feeder. Therefore,

$$Sup_f(t) = \sum_{b_s \in \mathcal{F}(f) \cap \mathcal{S}} (S_{b_s}(t) - \sum_{j=1}^{N} \gamma_{s,b_s j}(t) x_{s,b_s j}(t)) \; \forall f \in \mathcal{F}, \forall t \in \{1, \ldots, T\} \quad (3.4)$$

$$Dem_f(t) = \sum_{b_d \in \mathcal{F}(f) \cap \mathcal{D}} (L_{b_d}(t) - \sum_{j=1}^{N} \gamma_{d,b_d j}(t) x_{d,b_d j}(t)) \; \forall f \in \mathcal{F}, \forall t \in \{1, \ldots, T\} \quad (3.5)$$

$$Sup_f(t) - Dem_f(t) \leq Cap_f \; \forall f \in \mathcal{F}, \forall t \in \{1, \ldots, T\} \quad (3.6)$$

Note that our network model considers only the capacity constraints of the infrastructure such as feeders and distributors. We do not consider the effects of the structure of the network on local grid operating state variables such as voltage fluctuations or phase imbalances.

3.5 Modeling Storage for Supply Demand Matching Algorithms

Energy storage systems are being increasingly adopted in cooperation with renewable energy sources such as solar PVs to provide uninterrupted supply to the consumers [30]. Storage systems can be used as a buffer to act as either supply or load when required, thus reducing the uncertainty of renewable generation. Therefore, modeling storage systems for supply demand matching algorithms will greatly improve the scope of the algorithms.

The energy storage model that we consider is developed in [66]. We briefly describe the significant parameters here:

- *Energy Storage Capacity R (kWh):* The maximum amount of energy that can be stored. This energy is with reference to a minimum storage capacity i.e. the

capacity below which the storage cannot be discharged. We assume the minimum storage capacity to be zero.

- *Storage Discharge Energy D (kWh):* The maximum output/discharging power of the storage multiplied by the interval duration. The storage output D_t for interval t satisfies $0 \leq D_t \leq D$.

- *Storage Charge Energy G (kWh):* The maximum input/charging power of the storage multiplied by the interval duration. The storage input G_t for interval t satisfies $0 \leq G_t \leq G$.

- *Charge (η_G) and Discharge (η_D) Efficiency:* Charge efficiency $\eta_G \in (0, 1)$ is the ratio of charged power to the input power and discharge efficiency $\eta_D \in (0, 1)$ is the ratio of the output power to the discharged power.

Using the parameters above, storage capacity at time intervals t and $t + 1$ satisfy $R_{t+1} = R_t + \eta_G G_t - \frac{1}{\eta_D} D_t$ with the following constraints: $0 \leq R_{t+1} \leq R, 0 \leq D_t \leq D$ and $0 \leq G_t \leq G$. A summary of the notations used in the energy storage model is listed in Table 3.2.

Table 3.2: List of Variables in the Storage Model

Variable	Meaning
R	Energy storage capacity
R_t	Energy storage available at time t
D	Maximum storage discharge energy in a single interval
D_t	storage discharged during interval t
G	Maximum storage charge energy in a single interval
G_t	storage charged during interval t
η_D	Discharging efficiency
η_G	Charging efficiency

The algorithms developed in this work focus on curtailment strategies. To incorporate storage, if the algorithm performs supply curtailment in an interval, storage can be

used in charging mode to be interpreted as a supply curtailment. Similarly, if the algorithm performs load curtailment, storage can be used in discharging mode. Our storage model can easily be extended to more sophisticated storage scheduling in which a global view of storage capacity across the intervals is considered and where storage can be used both for charging and discharging irrespective of whether the algorithm is performing supply curtailment or load curtailment.

3.6 NP-hardness of Supply Demand Matching Problem

In this section, we formally prove the NP hardness of the problem of supply demand matching with the nodes exhibiting discrete operational states. In order to prove that this problem is NP-hard, we will define a simpler version of the problem and reduce the well known knapsack problem, which is an NP-hard problem to it. Adding any additional constraints to this simpler version will only increase the complexity of the problem.

The simpler version of the problem Π is formulated as follows: We are given a set S of node-strategy pairs, where $s_{ij} \in S : i \in \{1, \ldots, M\}, j \in \{1, \ldots, N\}$ denotes the node-i-strategy-j pair, where M is the number of node and N is the number of strategies. Given a curtailment value Γ, we need to output a $S^* \subseteq S$ such that $\Gamma \leq \sum_{i,j:s_{ij} \in S^*} \gamma(s_{ij})$, where $\gamma(s_{ij})$ denotes the curtailment obtained by s_{ij} and $\sum_{j=1}^{N} I(s_{ij}) \leq 1 \forall i \in \{1, \ldots, M\}$, where $I(s_{ij}) = 1$ if $s_{ij} \in S^*$ and 0 otherwise and $\sum_{i,j:s_{ij} \in S^*} \mathcal{C}(s_{ij})$ is minimized, where $\mathcal{C}(s_{ij})$ denotes the cost of curtailment by s_{ij}.

Theorem 1. Π *is NP-hard.*

Proof. A 0-1 Knapsack problem [60] is defined as follows: Given M elements, with element i having value v_i and size d_i, find a sub-set of elements the sum of whose sizes is $\leq D$ and the value is maximized. To reduce this problem into Π, for each element i,

we add s_{i1} with $\gamma(s_{i1}) = -d_i$ and $\mathcal{C}(s_{i1}) = -v_i$. We set $\Gamma = -D$. Note that $j \in \{1\}$. One can easily observe that 0-1 knapsack problem has a solution if and only if Π has a solution. $\square$

Chapter 4

Optimal Curtailment Selection for Demand Response

In this chapter, we develop algorithms to perform selection of curtailment strategies for each consumer to perform Demand Response. In this scenario, the supply is fixed and the demand needs to be controlled to ensure that it remains less than supply. Demand curtailment avoids costly supply ramp up thus lowering the cost of grid operations. As supply is not controlled, all the algorithms discussed in this chapter run in single control mode with only demand being controlled.

4.1 Problem of Curtailment Strategy Selection for Demand Response

A Smart Grid is typically operated by a utility. The utility is responsible for providing power, controlling and monitoring the SmartGrid. In this chapter, we consider the scenario in which the utility provider has a fixed power generation capacity. Typically, this capacity is sufficient to fulfill the power requirements of the customers. However, when there is a surge in the demand from the customers, the utility needs to ensure that the demand is met by either adding generators or purchasing extra power from the spot market, both of which increase expenditure. Failure to do so compromises the system reliability and leads to blackouts.

Power consumption profile of a consumers varies throughout the day with periods of high demand interspersed with periods of low power consumption. Certain periods of the day observe an overlap between the high demands of several customers. We denote such periods with the power requirement of the grid substantially higher than the rest of the day as peak demand period. The demand in a peak period can exceed the power generation capacity.

To minimize or avoid the expenditure of purchasing extra power during peak demand periods, utilities adopt the technique of Demand Response. Customers are either incentivised to reduce their consumption during a Demand Response Event (DR-Event) or they are penalized by increasing the cost of power during these periods. This reduces the peak power consumption which is now expected to be met by the available generation capacity.

Using smart meters, utilities have the power consumption data of each customer. The granularity of the data can be as small as 15 minutes. The power consumption profile of a customer does not change rapidly from day to day, so it is straightforward to predict future pattern. By employing prediction techniques, utilities determine the peak demand periods. They also determine the targeted curtailment required for a DR-Event which should be scheduled during this period. Discussion on the prediction techniques is beyond the scope of this paper. Readers can refer to [13] for further knowledge on this topic.

Utilities roll out a program to implement Demand Reponse and enroll customers into it. A customer is provided with a list of strategies to be followed each of which leads to a certain amount of curtailment in power consumption. Strategies can include procedures such as increasing the temperature of the AC systems by 2 degrees or turning off every other light in the hallways, etc. which reduce power consumption. During a DR-Event, the utility signals each customer to follow a particular strategy. A consumer

may be penalized if it fails to comply. For instance, the University of Southern California SmartGrid consists of 50,000 sensors across the 170 buildings to monitor electricity usage. Each building can adopt any one of seven available strategies during DR events which occur on Weekdays 1-5 pm [3].

Careful selection of consumers is required to ensure that the targeted curtailment value is met. A good selection algorithm determines the subset of consumers along with the strategies they should follow during the DR-Event such that the achieved curtailment value is as close as possible to the target. The reasons are as follows:

1. Limiting the amount by which the achieved curtailment value *overshoots* the target ensures that the grid is not underutilized. This avoids any loss of revenues to the utility due to underutilization of grid by aggressive curtailment.

2. Limiting the amount by which the achieved curtailment value *undershoots* the target ensures that the utility can avoid purchasing power from external sources by bounding the peak demand of the customers.

In the following sections, we develop several formulations addressing various use cases to optimally perform Demand Response.

4.2 Traditional Demand Response

We formally define the problem of optimal customer selection for Demand Response (Traditional Demand Response) using the parameters defined in Table 3.1. We are given a list of M_d consumers and N strategies. Each consumer can adopt exactly one strategy in the DR event. The decision variable x_{bj} is 1 if consumer b adopts strategy j. We are also given the curtailment in power consumption $\gamma_{d,bj}$ obtained by consumer b adopting

strategy j. A default strategy with a curtailment value of 0 is also included in the curtailment matrix γ_d. A customer adopting a default strategy essentially means that it is not participating in the DR event.

A targeted curtailment value Γ for the DR event is provided. The objective it to achieve a curtailment value as close to Γ as possible. The ILP formulation for this problem is as follows:

$$Minimize: \mid \sum_{b=1}^{M_d} \sum_{j=1}^{N} \gamma_{bj} * x_{d,bj} - \Gamma \mid \tag{4.1}$$

$$Subject\ to: \sum_{j=1}^{N} x_{d,bj} = 1, \qquad b \in \{1, \ldots, M\} \tag{4.2}$$

$$x_{d,bj} \in \{0, 1\}, \qquad \forall b, j \tag{4.3}$$

Equation 4.1 minimizes the absolute curtailment error. Equation 4.2 ensures that a consumer cannot adopt more than one strategy in the DR event. Detailed experimental results for consumer selection using the above ILP is shown in Section 4.7.

4.3 Sustainable Demand Response

4.3.1 Motivation

The algorithm mentioned in the previous section might aggressively curtail the demand in some intervals while accumulating demands in other intervals. Such assignments have peaks in certain intervals, which can possibly exceed the generation capacity forcing the utility to pay for additional procurement of energy.

We define the notion of Sustainable Demand Response (SDR) to address such cases. SDR attempts to evenly smooth the curtailment over the entire period of the DR event.

Hence we define SDR as the customer-strategy assignment which minimizes the $\|l\|_1$ distance between achieved curtailment values and a smoothed target value per interval.

As before we are given a set S of M_d consumers, N strategies. The entire DR period is divided into discrete time intervals. Dynamic customer strategies are represented by a time varying curtailment matrix $\gamma_d(t) \in \mathbf{R}^{M_d \times N}$ with element $\gamma_{d,bj}(t)$ denoting the discrete curtailment value of consumer b adopting strategy j at time interval t where $t \in \{1, \ldots, T\}$. Let $X_d(t)$ be the decision matrix with element $x_{d,bj}(t)$ denoting the corresponding decision variable at time t with Γ denoting the achievable curtailment value across the entire DR event.

4.3.2 ILP Formulation for Sustainable DR

We use the following ILP to model a Sustainable DR event.

$$Minimize : \sum_{t=1}^{T} \epsilon_t \tag{4.4}$$

$$Subject\ to : |\sum_{b=1}^{M}\sum_{j=1}^{N} \gamma_{d,bj}(t) x_{d,bj}(t) - \frac{\Gamma}{T}| \leq \epsilon_t \qquad \forall t \tag{4.5}$$

$$\sum_{j=1}^{N} x_{d,bj}(t) = 1 \qquad \forall b, t \tag{4.6}$$

$$\forall x_{d,bj}(t) \in \{0, 1\} \qquad \forall b, j, t$$

The objective is the minimize the $\|L\|_1$ norm (Equation 4.4). As before, Equation 4.6 ensures that at any given interval, each consumer adopts exactly one strategy. The various intervals in the ILP above are independent. This makes it trivial to parallelize by solving each interval as a separate optimization problem on a single node in a high performance cluster.

4.4 Sustainable DR with Strategy Overheads

4.4.1 Motivation

Driven by our experience with existing DR implementations on the USC smartgrid, we observe that it is impractical for customers to switch between too many strategies during the DR event as this leads to additional overhead costs. We model this by associated a cost of 1 for each strategy transition and by limiting the number of transitions for each node by τ. An additional constraint is added to the ILP to incorporate the strategy switching overheads. Note that under this formulation, a consumer is likely to have contiguous strategies across intervals. In our experimental work section, we solve this ILP exactly for reasonable problem sizes (representing the USC microgrid) using the IBM CPLEX Solver. For very large problem sizes, the time required for an exact solution might be large. In such cases, one can use randomized rounding heuristics based on the LP-relaxation of the ILP to obtain approximate solution.

4.4.2 ILP formulation for Sustainable DR with Strategy Overheads

We use the following ILP to model a Sustainable DR event with strategy overheads.

$$Minimize : \sum_{t=1}^{T} \epsilon_t \tag{4.7}$$

$$Subject\ to : | \sum_{b=1}^{M_d} \sum_{j=1}^{N} \gamma_{d,bj}(t) x_{d,bj}(t) - \frac{\Gamma}{T} | \leq \epsilon_t \qquad \forall t \tag{4.8}$$

$$\sum_{j=1}^{N} x_{d,bj}(t) = 1 \qquad \forall b, t \tag{4.9}$$

$$x_{d,bj}(t) \in \{0, 1\} \qquad \forall b, j$$

$$S_{bj}(t) = | x_{d,bj}(t) - x_{d,bj}(t-1) | \qquad \forall b, j, t \in \{2, \ldots, T\} \tag{4.10}$$

$$\sum_{t=2}^{T} \sum_{j=1}^{N} S_{bj}(t) \leq 2\tau \qquad \forall b \tag{4.11}$$

The new constraints to limit the strategy switching are introduced using 4.10 and 4.11 where 4.10 calculates the number of times customer b switches a particular strategy. Equation 4.11 bounds the total number of times a customer can switch strategies. Since the state variable $S_{bj}(t)$ counts both switching into and switching out from strategy j, equation 4.11 uses 2τ as the bound. In our experiments, we fix the value of $\tau = 2$.

4.5 NO-LESS: Near OptimaL CurtailmEnt Strategy Selection for Supply Demand Matching in Micro Grids

4.5.1 Problem Definition

We are given a set of M_d nodes and N strategies. The entire curtailment horizon is divided into discrete time intervals. We are given a time varying curtailment matrix $\gamma_d(t) \in \mathbf{R}^{M_d \times N}$ with element $\gamma_{d,bj}(t)$ denoting the discrete curtailment value of node b adopting strategy j at time interval t where $t \in \{1, \ldots, T\}$. Let B_b be the maximum curtailment value for node b. Let $X_d(t)$ be the decision matrix with element $x_{d,bj}(t)$ denoting the corresponding decision variable at time t. The achievable curtailment value across the entire curtailment horizon is given by Γ. τ_b denotes the limit on the total cost of strategy switches for a node b. We assume that strategy 1 is the default strategy with a curtailment value of 0 i.e. if a node is not included in an interval of the curtailment horizon, it follows strategy 1.

Given the model above, the objective is to determine node-strategy pairs for each interval such that: (1) The curtailment target is achieved with minimum curtailment error: difference between the achieved and the targeted curtailment, (2) No node is curtailed more than its maximum allowable value, (3) only allowable strategy switches are performed by each node across each consecutive intervals, and (4) for each node, the cost incurred in switching strategies is within its allowable limit.

4.5.2 ILP Formulation for NO-LESS

NO-LESS can be formulated using the following ILP:

$$Minimize \ \ \epsilon \tag{4.12}$$

$$s.t. \ \ \sum_{t=1}^{T}\sum_{b=1}^{M}\sum_{j=1}^{N}\gamma_{d,bj}(t)x_{d,bj}(t) - \Gamma \leq \epsilon \tag{4.13}$$

$$\sum_{j=1}^{N} x_{d,bj}(t) = 1 \ \ \forall b, \ \ t = 1, 2, \ldots T \tag{4.14}$$

$$\sum_{t=1}^{T}\sum_{j=1}^{N}\gamma_{d,bj}(t)x_{d,bj}(t) \leq B_b \ \ \forall b \tag{4.15}$$

$$S_{bij}(t) = \mid x_{d,bi}(t-1) - x_{d,bj}(t) \mid$$

$$\forall b, \ \ \forall t, \ \ i, j \in \{1, \ldots, N\} \tag{4.16}$$

$$x_{d,b1}(0) = 1 \ \ \forall b \tag{4.17}$$

$$x_{d,bj}(0) = 0 \ \ \forall b, \ j \in \{2, \ldots, N\} \tag{4.18}$$

$$\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_{d,bi}(t-1) + x_{d,bj}(t) - S_{bij}(t))\frac{\chi_b(i,j)}{2}$$

$$\leq \tau_b, \ \ \forall b \tag{4.19}$$

$$\epsilon \geq 0 \tag{4.20}$$

$$x_{d,bj}(t) \in \{0, 1\} \ \forall b, \ \ \forall j, \ \ t = 1, 2, \ldots T \tag{4.21}$$

Equations 4.12, 4.13 ensure that the targeted curtailment is met with minimum error. Equation 4.14 ensures that a node follows only 1 strategy in any interval. Equations 4.15- 4.19 ensure that the curtailment and switching cost for each node is bounded. The ILP above returns the optimal solution, however due to its large execution time we develop an FPTAS for it.

4.6 Approximation Algorithms

In the previous sections, we formulated problems for various use cases using Integer Linear Programs. ILPs return optimal solution, however, they require large execution time and are not scalable. Hence, in this section we develop approximation algorithms for the ILP algorithms defined above.

4.6.1 Fast $\sqrt{2}$-factor Approximation for Sustainable DR

We now describe a fast algorithm for computing approximately optimal sustainable DR strategies. Our algorithm provides a $\sqrt{2}$-factor approximation to the optimal target during each curtailment period and therefore for the entire DR event.

Theorem 2. *Algorithm 1 is a $\sqrt{2}$-factor approximation to the optimal sustainable DR solution.*

Proof. Let Y_t denote the curtailment value for period t returned by Algorithm 1. We show that either $Y_t \in [\frac{\Gamma}{T\sqrt{2}}, \frac{\Gamma\sqrt{2}}{T}]$ or Y_t is the optimal curtailment value achievable for that period. If line 2 of the algorithm is satisfied, then this is trivially true. Assume line 2 is not satisfied. From line 8, we must have $\Gamma/(T\sqrt{2}) \leq Y_r = \sum_{b=1}^{r-1} \gamma_{d,bk_b} + \gamma_{d,rk_r} \leq 2 \cdot \Gamma/(T\sqrt{2}) = \Gamma\sqrt{2}/T$.

Finally, consider the case when when $r = M$ and $Y_M \leq \Gamma/(\sqrt{2}T)$. Since k_b represents the strategy with the largest curtailment value $\leq \Gamma/(\sqrt{2}T)$ for each customer b, by definition Y_r is the largest achievable curtailment value $< Y_p$ and so either Y_p or Y_r is the optimal strategy for this period. □

The following result follows from a straightforward analysis of the algorithm.

Theorem 3. *Algorithm 1 can be used to compute $\sqrt{2}$-approximate sustainable DR solutions in $O(TM_d \log N)$ time when strategies are preprocessed in advance for a given*

Algorithm 1: Fast $\sqrt{2}$-factor Sustainable DR Approximation

Preprocessing: Non-decreasing sorted lists $\{\gamma_{d,b}(t)\}$ of consumer-strategies for each consumer $b \in S$ and each interval t

for *intervals* $t = 1$ *to* T **do**
  if $\exists(k \in S) \wedge (j \in \gamma_{d,k}(t)) : \gamma_{d,kj}(t) \in [\frac{\Gamma}{T\sqrt{2}}, \frac{\sqrt{2}\Gamma}{T}]$ **then**
    $x_{d,kj}(t) \leftarrow 1$;
    // Select consumer k and curtailment strategy $\gamma_{d,kj}(t)$
  else
    $(p, q_p) \leftarrow \text{argmin}_{b \in S, j \in \gamma_{d,b}(t)} \{\gamma_{d,bj}(t) | \gamma_{d,bj}(t) \geq \Gamma\sqrt{2}/T\}$;
    // p and q_p represent the customer and strategy indices of the consumer with the smallest curtailment value $\geq \Gamma\sqrt{2}/T$
    $Y_p \leftarrow \gamma_{d,pq_p}(t)$;
    For each customer $b \in S : k_b \leftarrow \text{argmax}_k \{\gamma_{d,bk}(t) \leq \Gamma/(T\sqrt{2})\}$;
    Let r denote the smallest index such that $Y_r \leftarrow \sum_{i=1}^{r} \gamma_{d,bk_b}(t); Y_r \geq \Gamma/(T\sqrt{2})$;
    $r \leftarrow M$ if $\sum_{i=1}^{M} \gamma_{d,bk_b}(t) < \Gamma/(T\sqrt{2})$;
    // Select Strategies for Activation as follows
    if $Y_p - \Gamma\sqrt{2}/T \leq \Gamma/(T\sqrt{2}) - Y_r$ **then**
      Set $x_{d,pq_p}(t) = 1$;
    else
      for $b \leftarrow 1$ **to** r **do**
        Set $x_{d,bk_b}(t) = 1$;
      end
    end
  end
end

Output: Matrix $\{X_d(t)\}$ of selected consumer-strategies $\forall t$. Bounded curtailment values $\widehat{\epsilon}_t \in [\frac{\tilde{\epsilon}_t}{\sqrt{2}}, \sqrt{2}\tilde{\epsilon}_t]$, where $\tilde{\epsilon}_t = \max(\epsilon_t^*, \Gamma/T)$, ϵ_t^* is the optimal solution to ILP.

curtailment target. The one-time preprocessing cost assuming apriori knowledge of curtailment strategies is $O(TM_dN \log N)$.

By precomputing and storing results for a range of target values, we can speed up the retrieval of approximately optimal strategies even further to $O(1)$ time.

Note that the same algorithm can be used as an approximation algorithm for Traditional Demand Response.

4.6.2 A PTAS for Sustainable DR

While the approximation algorithm above can be used to very quickly compute sustainable DR solutions, the error due to the $\sqrt{2}$-factor approximation may be unacceptably large in some cases. Therefore, using ideas from the subset sum problem [35] we develop a Polynomial Time Approximation Scheme (PTAS) that approximates the optimal solution provided by the ILP in Equation 4.4 to within an arbitrarily small ϵ-factor in time polynomial in $M_d N/\epsilon$. Since each customer is restricted to using exactly one strategy, we cannot simply merge all the customer strategies and find the subset that comes closest to the target Γ/T.

Theorem 4. *Algorithm 2 is a PTAS for the ILP in Equation 4.4.*

Proof Sketch: The number of intervals $l \approx \log_{\frac{1}{1-\epsilon}}(\Gamma/T)$ is polynomial in $\frac{\ln(\Gamma/T)}{\epsilon}$. In line 9, if a curtailment value is already marked feasible (i.e $Q_s^{(k-1)}(t)$ is 0), then customer k does not contribute to the feasible solution. The total number of iterations is $O(lNM_d)$. From line 7, using induction, we can show $V_{j+1} \geq \sum_{\gamma_{d,qr} \in B_j^{(M)}(t)} \gamma_{d,qr} \geq (1-\delta)^M V_{j+1} \geq (1-\epsilon)V_{j+1} = V_j$. and hence $\sum_{\gamma_{d,qr} \in B_j^{(M)}(t)} \gamma_{d,qr} \in [V_j, V_{j+1}]$. Thus the algorithm outputs an ϵ approximation to the optimal achievable target and is therefore a PTAS. $\square$.

4.6.3 FPTAS for NO-LESS

We first develop a dynamic programming based FPTAS to determine the set of curtailment strategies followed by a single node b during the curtailment horizon to achieve a curtailment value of at most $(1+\epsilon)\Gamma$, where ϵ is a user determined accuracy parameter.

Algorithm 2: PTAS for Sustainable DR during each interval t

Preliminaries: $V_0 \leftarrow \min_{i,j} \gamma_{d,ij}(t)$; $Z \leftarrow \min_{i,j} \gamma_{d,ij}(t) \geq \Gamma/T$;
Divide $[V_0, Z]$ into l intervals $\{[V_i, V_{i+1}]\}$, $V_{i+1} = (1+\epsilon)V_i$, $0 \leq i \leq l-2$, $V_l = Z$;
Initialization: $\forall k \in S : Y_{kj} \leftarrow V_i$ if $\gamma_{d,kj} \in [V_i, V_{i+1}]$;
$B_i^{(0)}(t) \leftarrow \emptyset$; $Q_i^{(0)}(t) \leftarrow 0$; $i = 0, 1, \ldots, n-1$;
// $B_i^{(k)}(t)$ is a subset of the first k consumers, each with a non-zero strategy selection, that add up to a total curtailment value $\in [V_i, V_{i+1}]$.

```
for Consumers k = 1 to M_d do
    for all intervals i, all strategies r do
        Z_ir ← Q_i^(k-1)(t) + Y_kr ;
        Let Z_ir ∈ [V_s, V_{s+1}] ;
        if ¬Q_s^(k-1)(t) then
            Q_s^(k)(t) ← V_s // Making Q_s^(k)(t) a feasible
                curtailment value
            B_s^(k)(t) ← B_i^(k-1)(t) ⋃ γ_{d,kr}(t) ;
            // Adding consumer k strategy r pair to the
                feasible strategy set
        end
    end
end
```

Output: $B_j^{(M_d)}(t)$: Selection of consumer-strategy pairs, where j is the closest interval to DR target Γ/T with $Q_j^{(M)}(t) > 0$

The cost of strategy switches will be bounded by τ_b. We then combine the results of all the nodes to achieve the targeted curtailment value from all the nodes.

Achieving Curtailment Target for a Single Node

Let $\mu = \frac{\epsilon\Gamma}{T}$. For each $\gamma_{d,bj}(t)$, we define $\widehat{\gamma}_{d,bj}(t) = \lfloor \frac{\gamma_{d,bj}(t)}{\mu} \rfloor$. We also define $\widehat{\Gamma} = \lfloor \frac{\Gamma}{\mu} \rfloor$. We define a boolean function Θ which returns true if a curtailment value $\widehat{\gamma}_t$ can be

achieved using strategy S_k at time t incurring a cost $\leq q_t$ for strategy switches and False otherwise. Θ can be defined using the following dynamic programming formulation:

$$\begin{aligned}\Theta(\widehat{\gamma}_t, t, S_k, q_t) &= \text{FALSE if } \widehat{\gamma}_t - \widehat{\gamma}_{d,bk}(t) < 0 || q_t < 0 \\ &= ||_j \Theta(\widehat{\gamma}_t - \widehat{\gamma}_{d,bk}(t), t-1, S_j, q_t - \chi_b(j,k)) \\ &(\forall j \in \{1,\ldots,N\}) \;\; otherwise \end{aligned} \tag{4.22}$$

If any of $\Theta(\widehat{\Gamma}, T, S_k, \tau_b) \;\forall k \in \{1,\ldots,N\}$ is True, we can determine the required strategies by traversing the recursive formulation and determining the strategies at each time t as described in Algorithm 3. The dynamic program can be solved by creating a table of size $\widehat{\Gamma} \times T \times N \times \tau_b$. We will refer to the entire table as Θ to simplify the notations. We can initialize the table using the following equations:

$$\begin{aligned}\Theta(\gamma_l, 1, S_k, q) =& \text{TRUE} \;\; \text{if } \widehat{\gamma}_{d,bk}(1) == \gamma_l \forall k \in \{1,\ldots,N\} \\ &\&\& \; q >= \chi_b(1k) \;\; \forall l \\ &\text{FALSE otherwise} \end{aligned} \tag{4.23}$$

Lemma 1. *Algorithm 3 finds the strategies to be followed in each interval by node b to achieve the curtailment value Γ within a cost of τ_b for strategy switches with a time complexity which is polynomial in N, T and $\frac{1}{\epsilon}$.*

Proof. We omit the proof of correctness as it can easily by argued using the arguments used for dynamic programming algorithms. Now, filling a single entry of the table requires $O(N)$ time. Hence, the algorithm requires $O(\frac{T^2N^2\tau_b}{\epsilon})$ which is the dominating term. We assume $\tau_b = O(T)$ i.e. the ratio of the largest to smallest cost is bounded and the number of switches is $\leq T$, the algorithm is polynomial in N, T and $\frac{1}{\epsilon}$. The total number of entries in the table are $\widehat{\Gamma} \times T \times N \times \tau_b$. $N \times \widehat{\Gamma} \times T \times N \times \tau_b =$

Algorithm 3: A $(1+\epsilon)$ polynomial time approximation algorithm to achieve the curtailment value Γ by a single node b

Fill entries $\Theta(\widehat{\gamma_l}, t, S_k, q_t) \forall \widehat{\gamma_l} \in \{0, \ldots, \widehat{\Gamma}\}, \forall t \in \{1, \ldots, T\}, \forall S_k, k \in \{1, \ldots, N\}, \forall q_t \in \{1, \ldots, \tau_b\}$ using equations 4.22 and 4.23
$X_d(t) \leftarrow \phi \;\; \forall t \in \{1, \ldots, T\}$
$\gamma_{cur} \leftarrow \widehat{\Gamma}$
$q_{cur} \leftarrow \tau_b$
for *Time* $t = T$ *to* 1 **do**
  if $\exists k \in 1, \ldots, N$ *s.t* $\Theta(\gamma_{cur}, t, S_k, q_{cur})$ **then**
    if *!*$\Theta(\gamma_{cur} - c_{bk}(t), t-1, S_k, q_{cur})$ **then**
      $q_{cur} \leftarrow q_{cur} - 1$
    end
    $\gamma_{cur} \leftarrow \gamma_{cur} - \gamma_{d,bk}(t)$
    $X_d(t) \leftarrow S_k$
  end
end
Output: X_d, where $X_d(t)$ denotes the strategy to be followed at time t

$N \times \frac{T}{\epsilon} \times T \times N \times \tau_b = O(\frac{T^2 N^2 \tau_b}{\epsilon})$. If we assume that $\tau_b = O(T)$ the algorithm is polynomial in N, T and $\frac{1}{\epsilon}$. (For loop requires only $O(TN)$ amount of time after the table is filled.) □

Lemma 2. *The total curtailment value obtained by following the strategies output by Algorithm 3 is* $\leq (1+\epsilon)\Gamma^*$ *where* Γ^* *is the total curtailment value obtained by following the strategies output by an optimal algorithm.*

Proof. Let Γ_x be the curtailment value obtained by following the strategies output by Algorithm 3. Let Γ^* be the optimal curtailment value. Clearly, $\Gamma_x \leq \mu \sum_{t=1}^{T} \gamma_{d,bX(t)}(t)$. Also, $\Gamma^* \geq \mu \sum_{t=1}^{T} (\gamma_{d,bX(t)}(t) - 1)$. This implies $\Gamma_x \leq \Gamma^* + \mu T \leq \Gamma^* + \epsilon\Gamma$. Now, since $\Gamma \leq \Gamma^*$, $\Gamma_x \geq \Gamma^*(1+\epsilon)$. □

Using Lemmas 1 and 2, we get the following theorem.

Theorem 5. *Algorithm 3 is a FPTAS to determine the strategy to be followed in each interval* $1, \ldots, T$ *to achieve a curtailment value* Γ *by a single node* b *with the cost of strategy switches is bounded by* τ_b*.*

Achieving Total Curtailment by all the Nodes

We will run step 1 of Algorithm 3 with curtailment value $\widehat{B}_b$, the node-specific maximum curtailment value, $\epsilon' = \frac{\epsilon}{M_d}$, and limit on the cost on strategy switching τ_b for each node $b \in \{1, \ldots, M_d\}$ to create Θ_b. For a node b, define $\Phi_b = \cup\ \widehat{\gamma}_l \mid \exists\{S_k, q\}$ with $\Theta_b(\widehat{\gamma}_l, T, S_k, q) =$ TRUE. Assume Φ_b is in sorted order. Let $\mathcal{L}(\Phi_b)$ denote the number of entries in Φ_b

We will again use a dynamic programming based algorithm for this problem. We define a boolean function Ξ which returns True if a curtailment value $\widehat{\gamma_b}$ can be obtained by using nodes $\{1, \ldots, b\}$ and False otherwise. Ξ can be defined using the following formulation:

$$\begin{aligned}\Xi(\widehat{\gamma_b}, b) = \text{TRUE if } &\exists k \in \{1, \ldots, \mathcal{L}(\Phi_b)\} \\ &s.t. \Xi(\widehat{\gamma_b} - \Phi_b(j), b-1) \text{ is TRUE} \\ = \text{FALSE otherwise}\end{aligned} \tag{4.24}$$

Ξ can be initialized for all $\widehat{\gamma_b} \in \{0, \ldots, \widehat{\Gamma}\}$, where $\widehat{\Gamma} = \lfloor \frac{\Gamma}{\mu} \rfloor, \mu = \frac{\epsilon\Gamma}{M_d T}$ by the following equations:

$$\begin{aligned}\Xi(\widehat{\gamma_b}, 1) = \text{TRUE if } &\exists k \in \{1, \ldots, \mathcal{L}(\Phi_b)\}\ \ s.t. \widehat{\gamma_b} = \Phi_b(k) \\ = \text{FALSE otherwise}\end{aligned} \tag{4.25}$$

Given a total curtailment value Γ, Algorithm 4 can be used to combine the curtailment values that can be achieved from individual nodes to attain the total curtailment value.

Algorithm 4: A $(1+\epsilon)$ polynomial time approximation algorithm to achieve the curtailment value Γ

Run Step 1 of Algorithm 3 for each node b with $\widehat{B}_b = \lfloor \frac{\widehat{B}_b}{\mu} \rfloor$, where $\mu = \frac{\epsilon\Gamma}{M_d T}$, $\epsilon' = \frac{\epsilon}{M_d}$, and $\tau = \tau_b$ and produce Φ_b
Fill entries $\Xi(\widehat{\gamma}_b, b) \forall \widehat{\gamma}_b \in \{0, \ldots, \widehat{\Gamma}\}, \forall b \in \{1, \ldots, M_d\}$ using equations 4.24 and 4.25
$X_b(t) \leftarrow \phi \; \forall b \; \in \{1, \ldots, M_d\}, \forall t \in \{1, \ldots, T\}$
$\gamma_{cur} \leftarrow \widehat{\gamma}$
for $b = M$ *to* 2 **do**
    if $\exists k \in \{1, \ldots, \mathcal{L}(\Phi_b)\}$ *s.t.* $\Xi(\gamma_{cur} - \Phi_b(k), b-1) == 1$ **then**
        Call Algorithm 3 with inputs $\Phi_b(k), T, \tau_b$ to produce output X
        $X_b \leftarrow X$
        $\gamma_{cur} \leftarrow \gamma_{cur} - \Phi_b(k)$
    end
end
Output: X, where $X_b(t)$ denotes the strategy to be followed by node b at time t

Theorem 6. *Algorithm 4 is a FPTAS for NO-LESS*

Proof. The correctness can be argued similar to that of Algorithm 3.

Let Γ_x be the curtailment value obtained by following the strategies output by Algorithm 4. Let Γ^* be the optimal curtailment value. Clearly, $\Gamma_x \leq \mu \sum_{b=1}^{M} \sum_{t=1}^{T} \gamma_{d,bX(t)}(t)$. Also, $\Gamma^* \geq \mu \sum_{b=1}^{M} \sum_{t=1}^{T} (\gamma_{d,bX(t)}(t) - 1)$. This implies $\Gamma_x \leq \Gamma^* + \mu M_d T \leq \Gamma^* + \epsilon\Gamma$. Now, since $\Gamma \leq \Gamma^*$, $\Gamma_x \geq \Gamma^*(1+\epsilon)$.

Step 1 of Algorithm 4 requires $O(\frac{M_d T^3 N^2}{\epsilon})$ time. Step 2 requires $O(\widehat{\Gamma})$ time to fill each of the $\widehat{\Gamma} \times M_d$ entries hence, $O(\frac{M_d^3 T^2}{\epsilon^2})$ time. These being the dominating terms, the total complexity of Algorithm 4 is $O(\frac{M_d^3 T^2}{\epsilon^2} + \frac{M_d T^3 N^2}{\epsilon})$ which is polynomial in M_d, N, T and $\frac{1}{\epsilon}$. □

The For loop contains M_d calls to Algorithm 3 and hence requires $O(M_d \times \frac{T^2 N^2}{\epsilon})$ time. *Remark:* In a typical micro-grid such as the one used for our experiments, typically

the number of strategies is $N \leq 10$ and the number of time intervals of the DR $T \leq 16$ 15-minute intervals i.e. 4 hours. However, the number of nodes is typically large and keeps on increasing with new nodes being added to DR. Hence, our algorithm with a complexity of $O(\frac{M_d^3}{\epsilon^2})$ and an optimality guarantee of getting a solution within $(1+\epsilon)$ times the optimal for this NP-hard problem is a significant improvement over the state-of-the-art techniques.

4.7 Experimental Results

4.7.1 Experimental Setup for TDR and SDR

The USC SmartGrid has over 50,000 sensors to monitor electricity usage and equipment status in real time [3]. Demand Response Events occur on weekdays between 1 and 5 pm. We use the data collected by the software developed to support data-driven demand response optimization in USC smartgrid [62]. The software provides us the curtailment values for each strategy that can be adopted by any building (customer) in USC for the queried time interval. For our experiments we use the data from 27 buildings each of which can adopt one of seven strategies. The data is collected for the time intervals 1-3 pm and 3-5 pm for each day from Monday to Friday. We run our experiments for Targeted Curtailment values ranging from 100 kWh to 1400 kWh.

We use the Optimization Programming Language [2] to define the Integer Linear Programming formulation developed in this paper. IBM ILOG CPLEX optimization software [1] is used to solve the ILP and produce the set of customers and the strategies they should adopt.

We compare our results with the state-of-the-art heuristic [76]. Authors in [76] develop a change making problem based algorithm for customer selection. The change making problem determines how to make a given amount of money using the least amount of coins. The coins in the algorithm are the available customer-strategy pairs and their values the predicted curtailment values. The amount to be made is the targeted curtailment value. Customers are grouped into bins differentiated by their values. A greedy algorithm is used to pick customers from the bins with highest values. We choose this heuristic for comparison as it is used in practice by the USC SmartGrid to schedule customers and their strategies for the DR events.

4.7.2 Results and Analysis for TDR and SDR

The power consumption profile of a building is similar for the same day of a week across different weeks. So by running our experiments on data collected from DR events over a week, we are able to demonstrate our algorithm on a wide range of power consumption profiles. Moreover, a typical DR-event in the USC SmartGrid starts with the selection procedure at 1 pm and then another selection occurs typically around 3 pm. Therefore, we consider them as two separate DR events for our experiments.

Figure 4.1 and 4.2 show the errors incurred by our ILP based customer selection algorithm and the State-of-the-art heuristic [76] for every DR event from Monday to Friday for various curtailment target values. As shown in Figure 4.1, the highest error incurred by our ILP based algorithm is around 0.0002 kWh during the DR-events on Tuesday 1-3 pm for a targeted curtailment of 600 kWh and Friday 3-5 pm for a targeted curtailment of 800 kWh. For the State-of-the-art heuristic, the highest error incurred is around 8 kWh during the DR event on Tuesday 3-5 pm for the targeted curtailment of 400 kWh as shown in Figure 4.2.

One may note that the errors between the state-of-the-art heuristic and our approach differ by multiple orders of magnitude. so we take the ratio of the error for comparison. A higher value of ratio implies better performance by our approach. In Figures 4.3-4.12 we compare the errors incurred by the two algorithms for various targeted curtailment value for each DR event

The customer selection problem can be visualized as a packing problem. We are trying to pack the targeted curtailment with values obtained from the customer-strategy pairs. The ILP produces the best possible packing. Any error is due to the nature of the data. Similarly, the heuristic based approach tries to provide best packing in each of the bins. Error incurred in packing each bin accumulates throughout the algorithm and may lead to very large errors. Since we are using real world data, as seen in the

Figures 4.3-4.12 the peaks in the ratio of errors for various DR events varies with the targeted curtailment values with no discernible pattern. The highest ratio observed is around 3×10^7 which occurs during the DR event on Thursday 1-3 pm.

Although solving an ILP is computationally intensive, optimal customer selection for each target was obtained in less than 5 seconds on a standard workstation. This time can be significantly reduced by using sophisticated computational platforms. Note that in a typical DR Event, the utility determines the curtailment target well in advance. Thus even a 5 second delay in computing the optimal customer-strategy pairs and signaling the customers is tolerable.

4.7.3 Sustainable DR

In Figures 4.13-4.17, we compare the absolute errors in kWh incurred by the Sustainable Demand Response (SDR) ILP formulation and the state-of-the-art heuristic for Targeted curtailment values ranging from 50-1000 kWh. The vertical axis is limited to a maximum error of 3.0 kWh to ensure the readability of the graph. The actual error values are given in the table below the graphs.

Our SDR ILP is more restrictive as it ensures that the curtailment is distributed evenly across the intervals which is not a constraint in the state-of-the-art heuristic. Despite this restriction, our SDR ILP perform 4-2000 times better than the heuristic. The only time the heuristic performs better than our ILP is on Wednesday for a Target of 1000 kWh where the error of our SDR ILP is 0.048 kWh while that of the heuristic is 0.001 kWh.

To emphasize the significance of Sustainable DR over a DR targeting the entire interval, we compare the demand curtailment values achieved in each interval for the DR event for the targeted curtailment value of 1200 kWh. The DR ILP targeting the entire interval (TDR) incurs an error of 0.002 kWh which is far lower that the 0.031 kWh

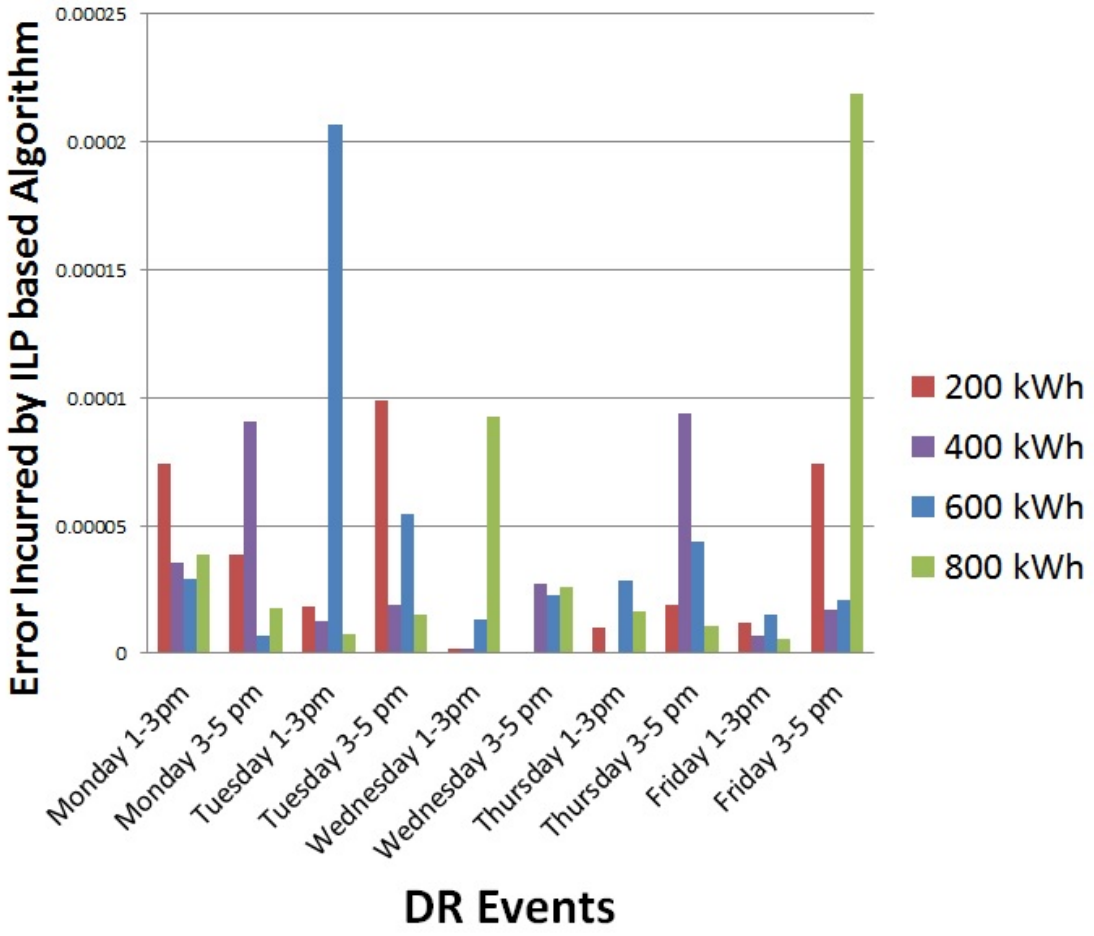


Figure 4.1: Error incurred by ILP based Algorithm for various targeted Curtailment Values for Different DR Event

error incurred by the Sustainable DR ILP. However, most of the curtailment is achieved in intervals 10 and 11 and the rest of the intervals have lower curtailment values. The Sustainable DR achieves a curtailment value of around 75 kWh in each interval.

4.7.4 Sustainable DR with Strategy Overheads

The motivation behind using Sustainable DR with strategy overheads is to reduce the number of times each building switches its strategy in the DR interval. We limit the number of times building i can switch strategies to $\tau = 2$. In Figures 4.13-4.17, we

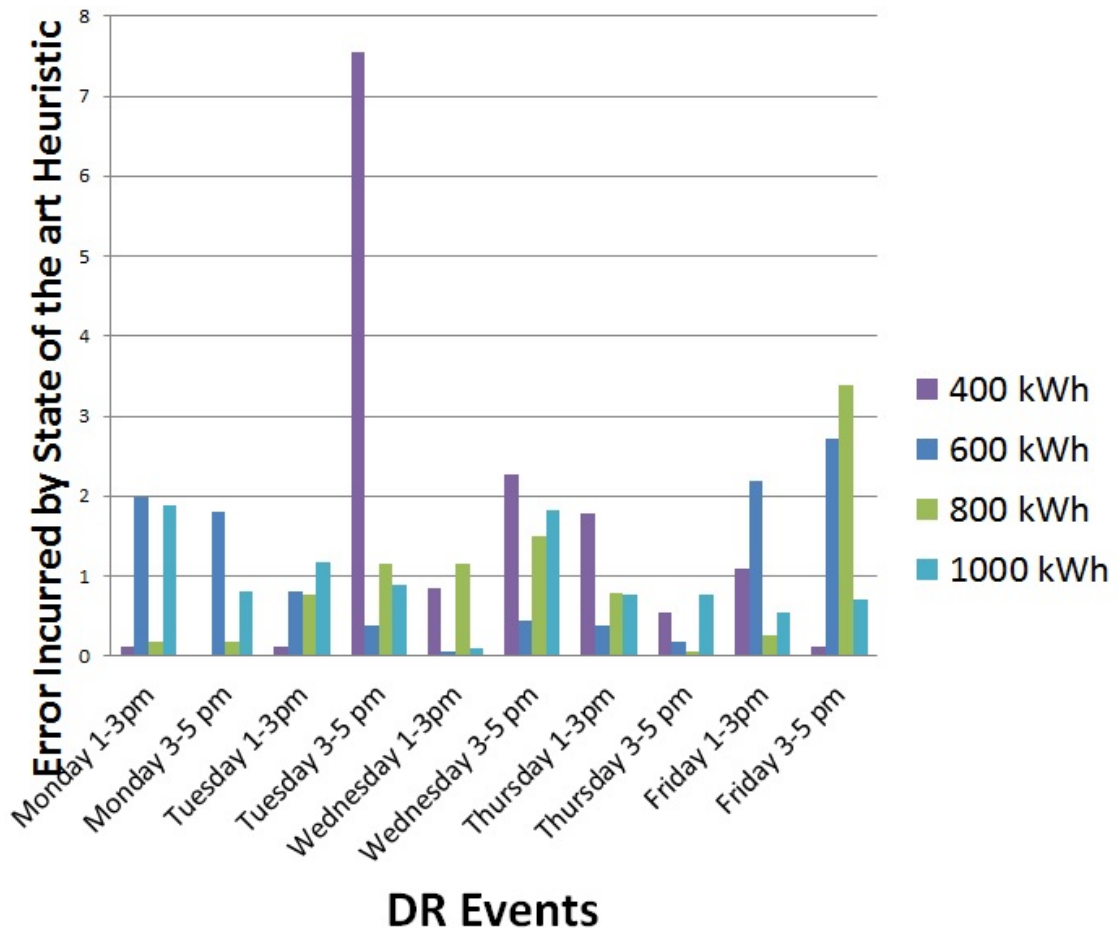


Figure 4.2: Error incurred by State-of-the-art Algorithm for various targeted Curtailment Values for Different DR Event

compare the absolute errors in kWh incurred by the Sustainable Demand Response with limits on strategy switching (SDR with switch limit) ILP formulation and the state-of-the-art heuristic for Targeted curtailment values ranging from 50-1000 kWh. The vertical axis is limited to a maximum error of 3.0 kWh to ensure the readability of the graph. The actual error values are given in the table below the graphs.

The SDR with strategy overheads is even more restrictive than the SDR ILP. This is reflected in the errors incurred which are an order higher than the SDR ILP. However, SDR with strategy overheads still performs 2-700 times better than the state-of-the-art

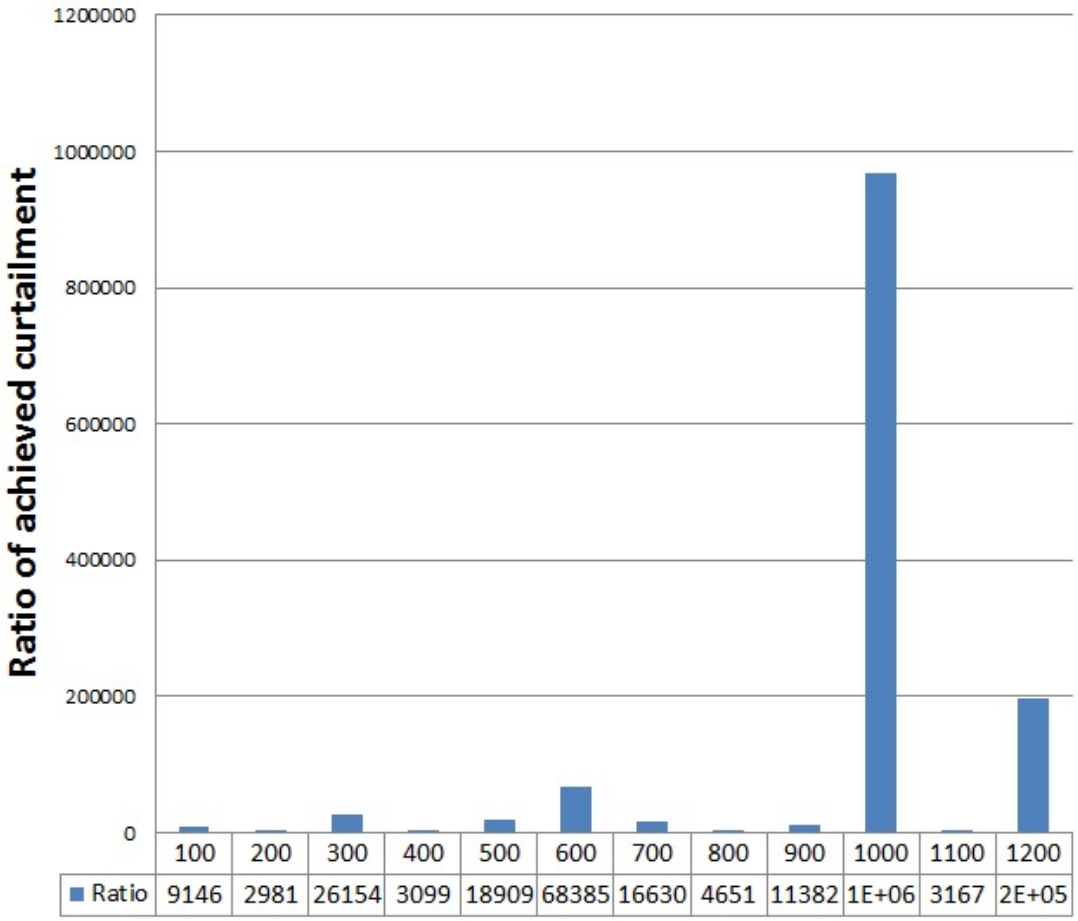


Figure 4.3: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Monday 1-3 pm

heuristic. The only times the heuristic performs better than our ILP is on Wednesday for a Target of 1000 kWh where the errors 0.083 kWh and 0.001 kWh respectively and on Tuesday for a Target of 400 kWh where the errors are 0.203 kWh and 0.172 kWh respectively for our ILP and the heuristic. The maximum relative error incurred by our ILPs is 1% whereas for the heuristic we observed relative errors as high as 12 %.

ILP is a computationally intensive process. To converge to the error rates shown in the results, the IBM CPLEX required 5-10 minutes of processing time. Since DR programs are based on predictive data [13] which are available in advance, the time required can be perceived as reasonable.

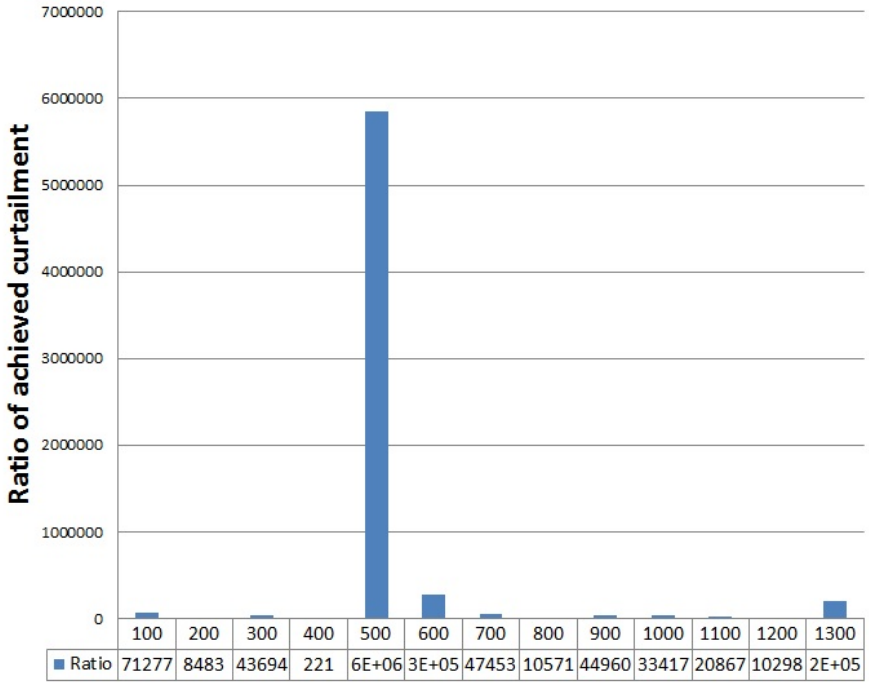


Figure 4.4: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Monday 3-5 pm

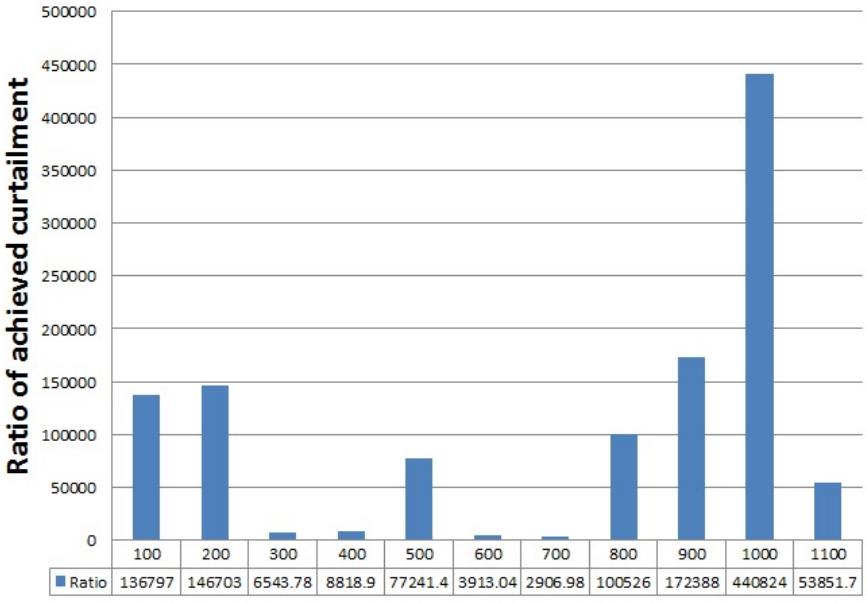


Figure 4.5: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Tuesday 1-3 pm

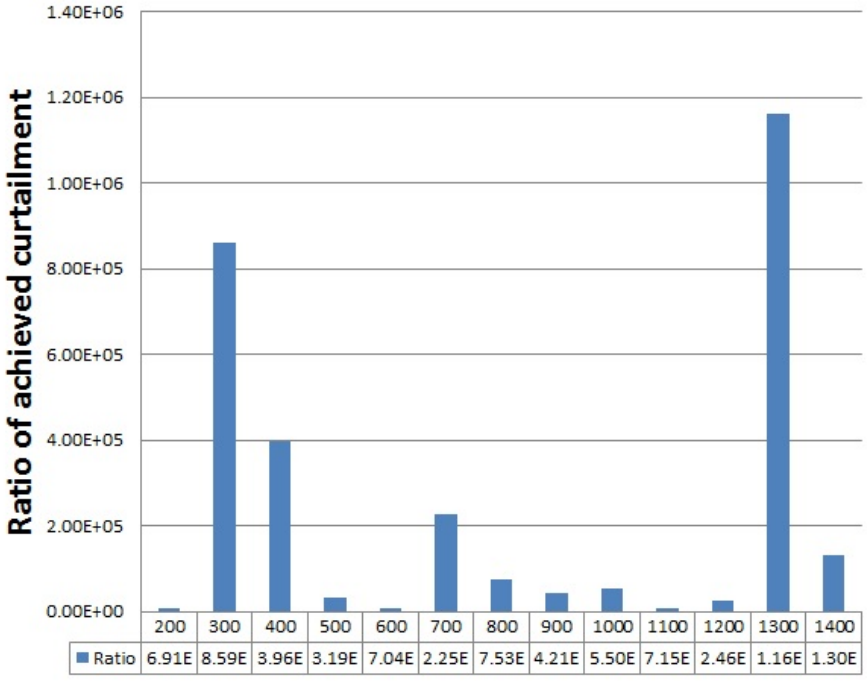


Figure 4.6: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Tuesday 3-5 pm

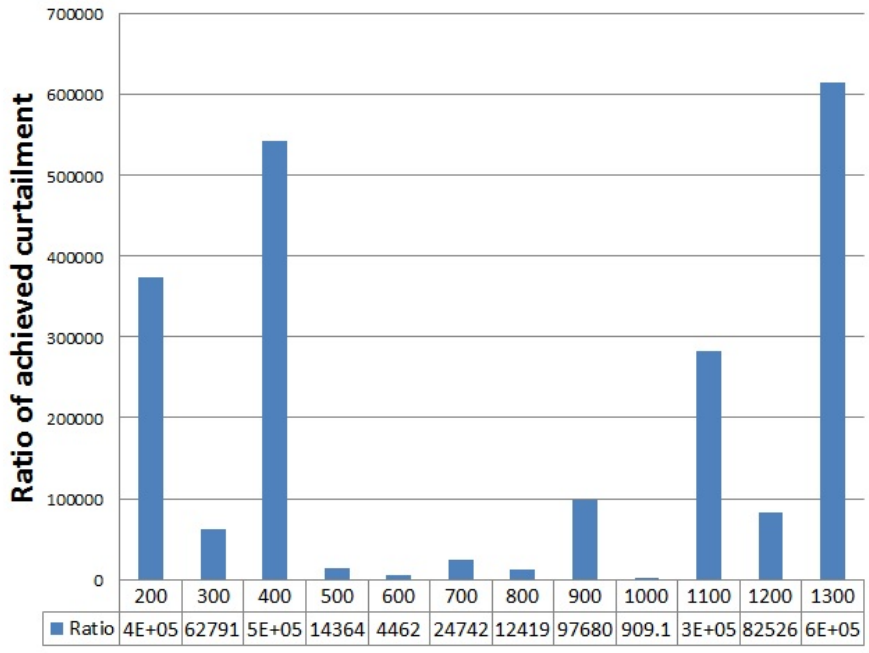


Figure 4.7: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Wednesday 1-3 pm

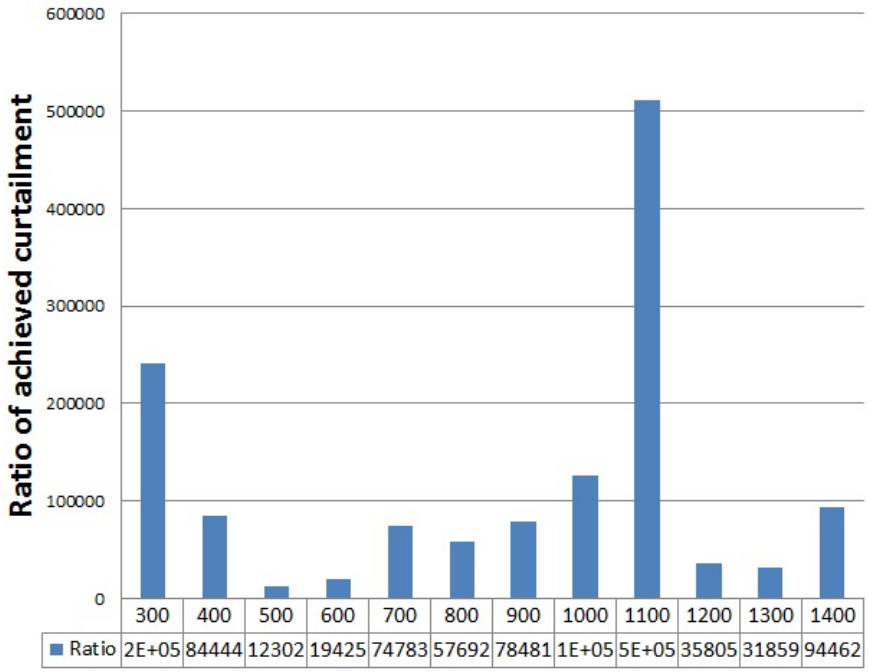


Figure 4.8: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Wednesday 3-5 pm

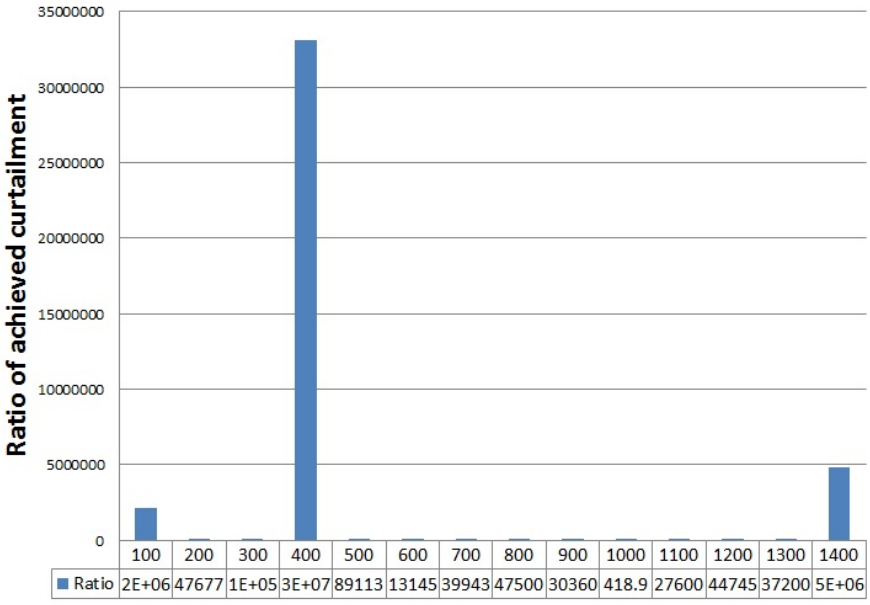


Figure 4.9: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Thursday 1-3 pm

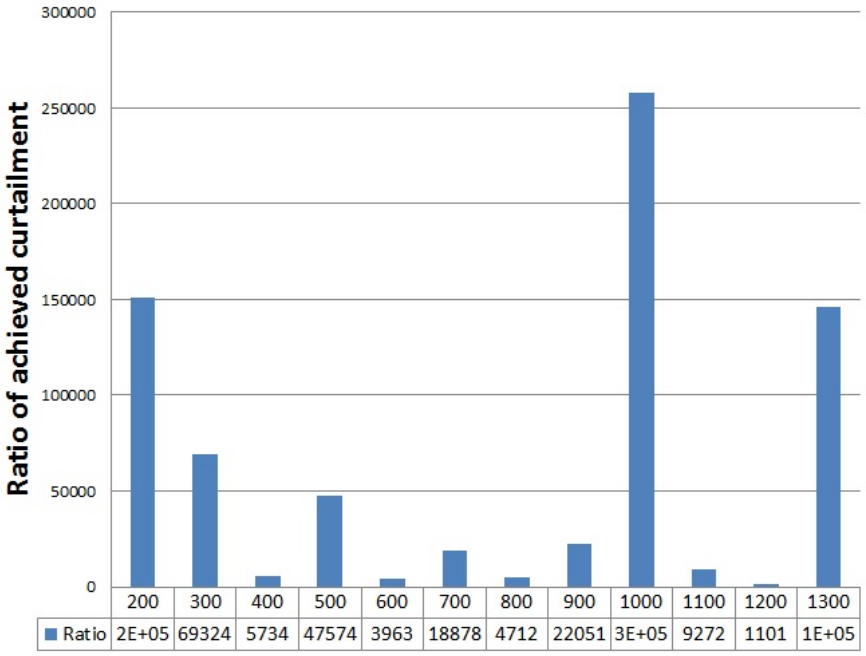


Figure 4.10: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Thursday 3-5 pm

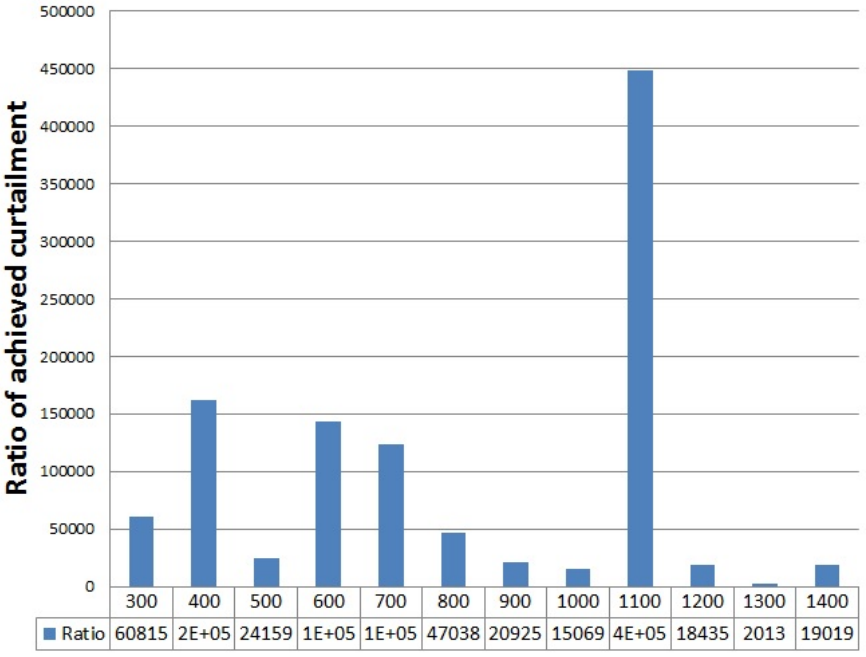


Figure 4.11: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Friday 1-3 pm

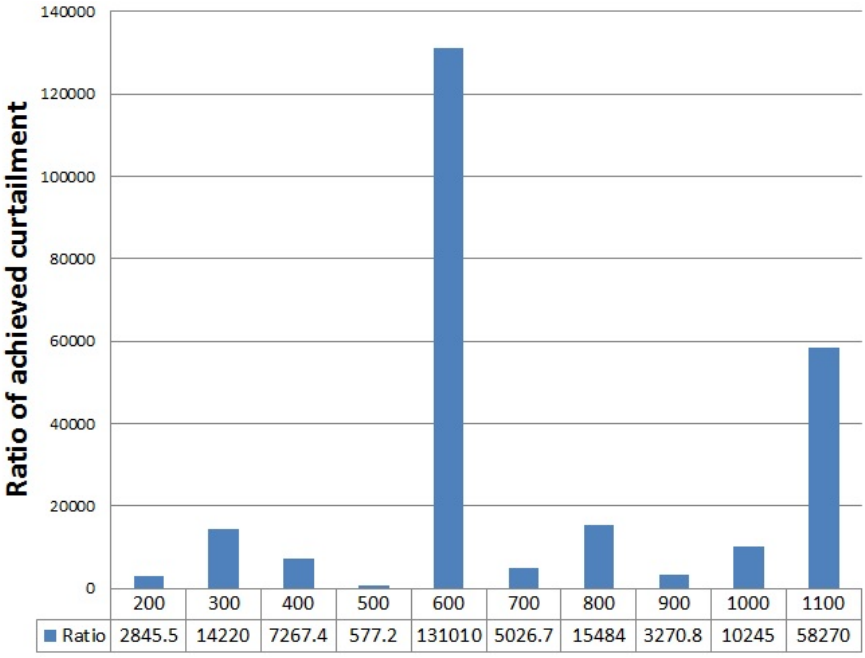


Figure 4.12: Ratio of error of State-of-the-art Heuristic and ILP algorithm for DR event Friday 3-5 pm

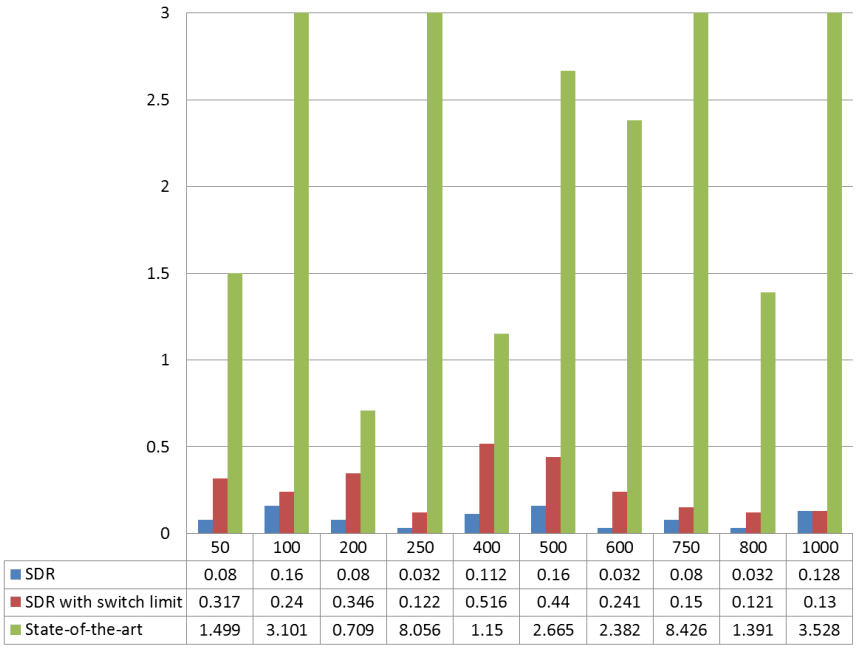
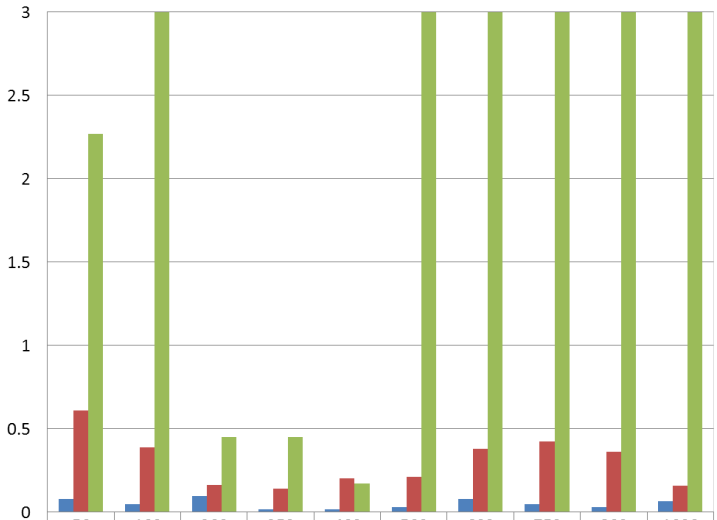
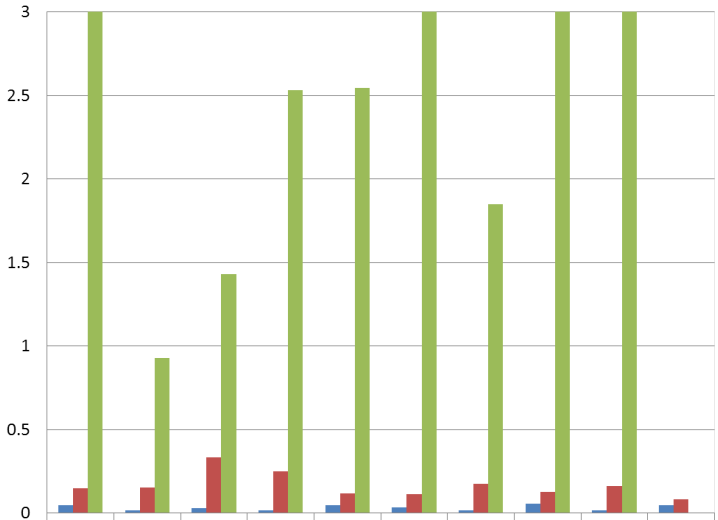


Figure 4.13: Absolute errors in kWh incurred for Sustainable DR event on Monday 1-5 pm



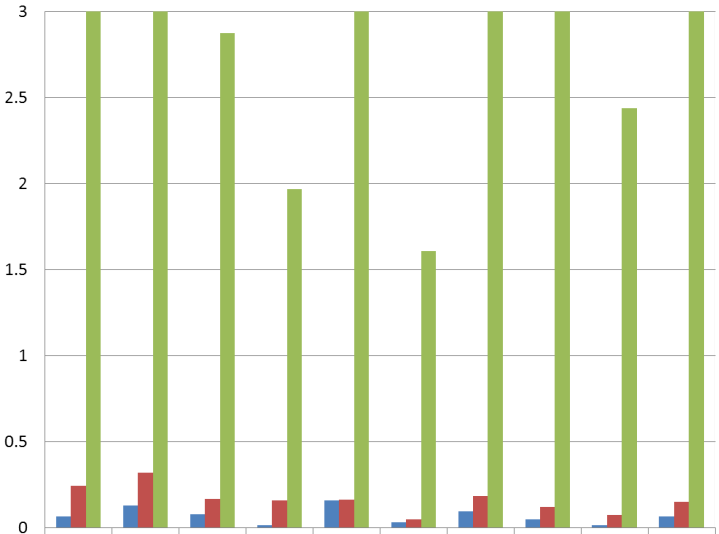
	50	100	200	250	400	500	600	750	800	1000
SDR	0.08	0.048	0.096	0.016	0.016	0.032	0.08	0.048	0.032	0.064
SDR with switch limit	0.608	0.389	0.164	0.142	0.203	0.211	0.381	0.423	0.364	0.159
State-of-the-art	2.27	8.896	0.45	0.452	0.172	7.859	6.342	3.189	6.649	5.241

Figure 4.14: Absolute errors in kWh incurred for Sustainable DR event on Tuesday 1-5 pm



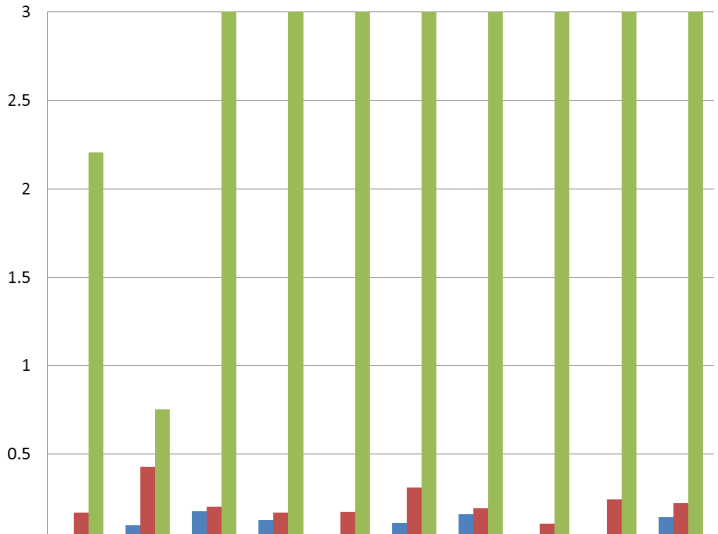
	50	100	200	250	400	500	600	750	800	1000
SDR	0.048	0.016	0.032	0.016	0.048	0.035	0.016	0.055	0.016	0.048
SDR with switch limit	0.148	0.153	0.333	0.249	0.119	0.116	0.175	0.128	0.161	0.083
State-of-the-art	4.388	0.929	1.431	2.533	2.547	4.302	1.85	9.225	3.673	0.001

Figure 4.15: Absolute errors in kWh incurred for Sustainable DR event on Wednesday 1-5 pm



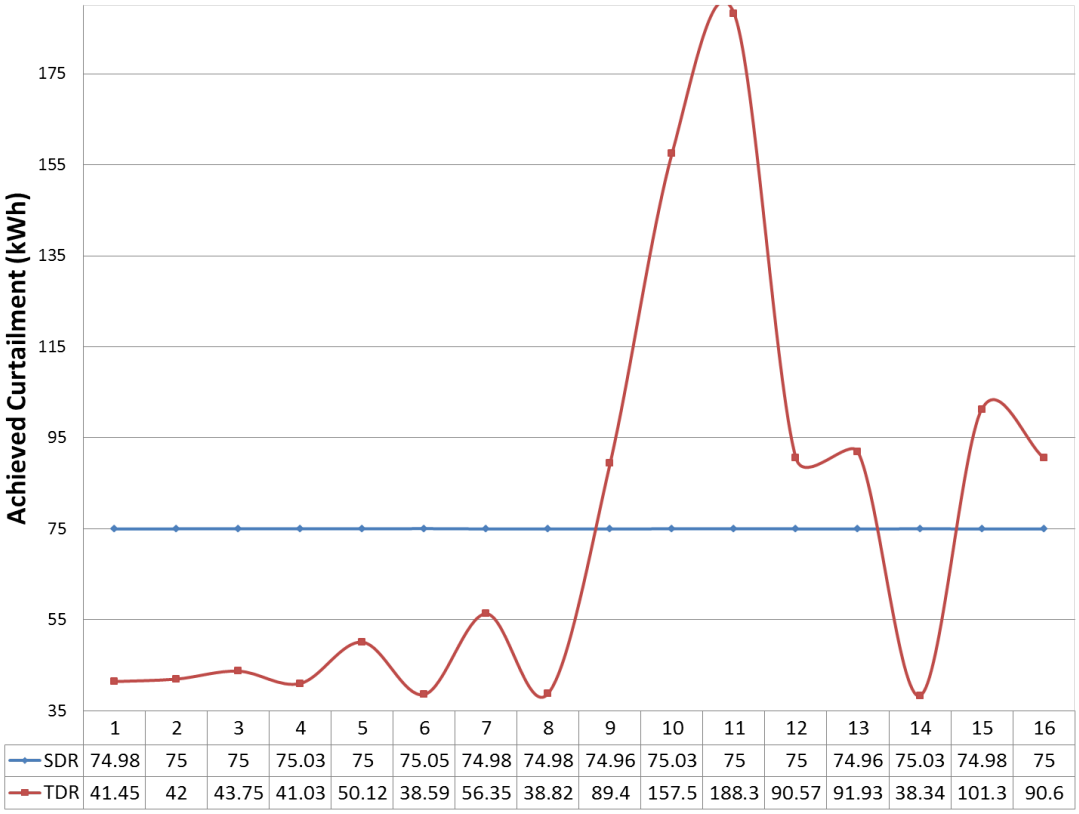
	50	100	200	250	400	500	600	750	800	1000
SDR	0.064	0.128	0.08	0.016	0.16	0.032	0.096	0.048	0.016	0.064
SDR with switch limit	0.242	0.32	0.166	0.157	0.164	0.049	0.186	0.12	0.073	0.15
State-of-the-art	4.325	3.191	2.874	1.969	4.298	1.607	31.068	3.576	2.441	4.117

Figure 4.16: Absolute errors in kWh incurred for Sustainable DR event on Thursday 1-5 pm



	50	100	200	250	400	500	600	750	800	1000
SDR	0.048	0.096	0.176	0.128	0.016	0.112	0.16	0.016	0.032	0.144
SDR with switch limit	0.168	0.426	0.204	0.17	0.175	0.309	0.195	0.106	0.242	0.223
State-of-the-art	2.206	0.755	7.168	9.007	10.426	6.443	6.002	7.031	5.465	5.894

Figure 4.17: Absolute errors in kWh incurred for Sustainable DR event on Friday 1-5 pm



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
SDR	74.98	75	75	75.03	75	75.05	74.98	74.98	74.96	75.03	75	75	74.96	75.03	74.98	75
TDR	41.45	42	43.75	41.03	50.12	38.59	56.35	38.82	89.4	157.5	188.3	90.57	91.93	38.34	101.3	90.6

Figure 4.18: Comparison of demand curtailment values achieved in each interval for SDR and TDR for the targeted curtailment value of 1200 kWh

4.7.5 NO-LESS

Experimental Setup for NO-LESS

We evaluate NO-LESS using the curtailment dataset obtained by the DR programs conducted in the University of Southern California (USC). The USC micro grid consists of more than 150 buildings equipped with smart meters to capture consumption data in 15-minute intervals. Each DR lasted for 4 hours during which each building switched to one out of 6 available strategies [36]. The curtailment data was obtained by using the algorithm developed in [22]. Althoug NO-LESS is developed for load curtailment, we

also evaluated its performance for strategy selection for solar curtailment. For solar curtailment, we use simulated data generated by using solar irradiance data for Los Angeles County [5] and varying the solar PV are from $10m^2$ to $20m^2$ and the yield from 5% to 15%. 6 different curtailment strategies: 0, $0.125 * O, 0.25 * O, 0.5 * O, 0.75 * O, O$, with O being the maximum output were generated. The cost of switching between strategies was fixed as 1. And the cost of strategy switching was limited to 10. The allowable strategy switches were $0 \leftrightarrow 1 \leftrightarrow 5$, $0 \leftrightarrow 2 \leftrightarrow 4 \leftrightarrow 5$ and $0 \leftrightarrow 3 \leftrightarrow 5$, where 0 is the default strategy with a curtailment value of 0. We implemented the NO-LESS in Java. The experiments were performed on Dell optiplex with 4-cores and 4 GB RAM.

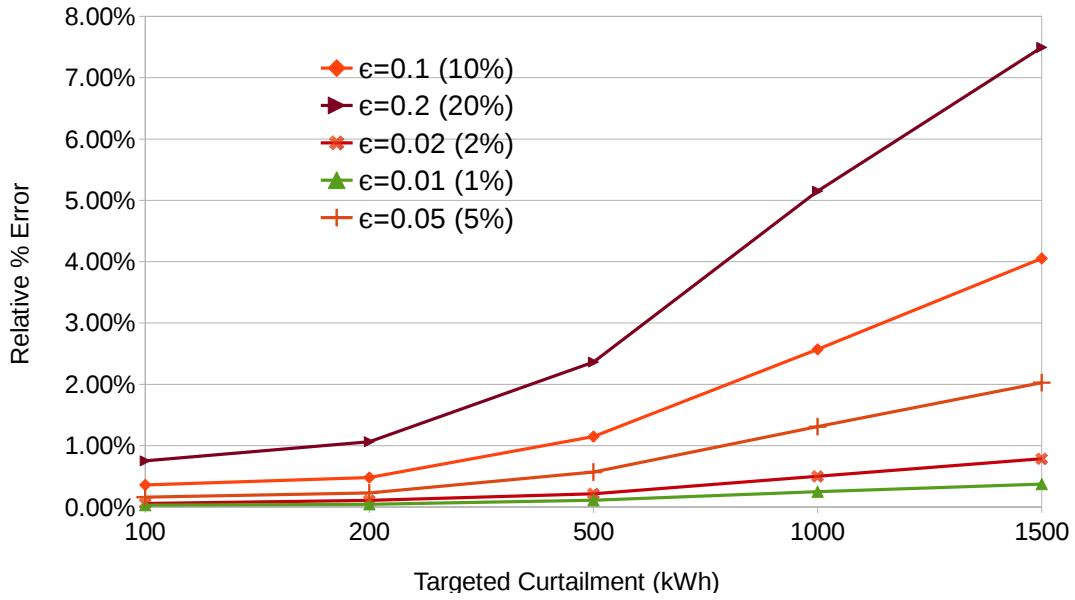


Figure 4.19: Near Optimality of NO-LESS

Near Optimality

We implemented an Integer Linear Program (ILP) for the problem objective to obtain optimal solutions. We varied the targeted curtailment values from 100 kWh to 2000 kWh. The number of nodes was fixed at 20. Figure 4.19 shows the relative percentage

error incurred by NO-LESS algorithm compared against the optimal solution. Relative percentage error is defined as: $\frac{(P-O)*100}{O}$, where P is the solution obtained by NO-LESS and O is the optimal solution obtained by the ILP. As shown in the figure, the relative percentage errors incurred by NO-LESS are much less than the worst case guarantee as determined by ϵ.

Notice that the relative percentage error increases as the curtailment target increases. This is because our algorithm is essentially a packing problem. For lower curtailment values, the algorithm has enough available node-strategy pairs to choose from to pack. However, as the curtailment value increases, the availability of such pairs becomes sparse. While the optimal algorithm tries all possible combination for optimizing the packing, our algorithm terminates as soon as it finds one which is within the worst case guarantee ϵ.

Scalability

NO-LESS algorithm has two critical parameters which affect scalability: the number of nodes M and the accuracy parameter ϵ. Note that our algorithm is independent of the targeted curtailment value.

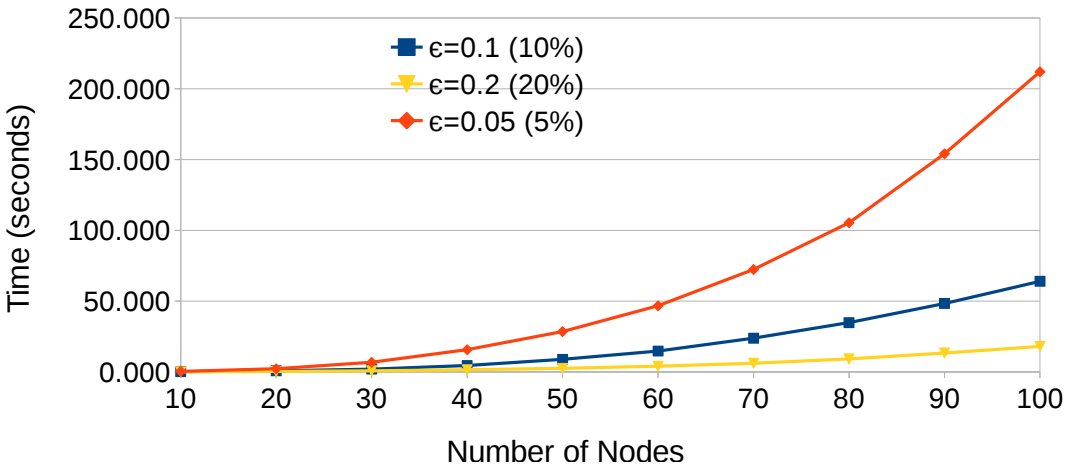


Figure 4.20: Execution Time vs Number of nodes for different values of epsilon

Figure 4.20 shows the effect on the execution time of the number of nodes for various values of ϵ. For each ϵ, the execution time increases polynomially with respect to the number of nodes. Even for an accuracy parameter of 5%, the runtime is less than 4 minutes.

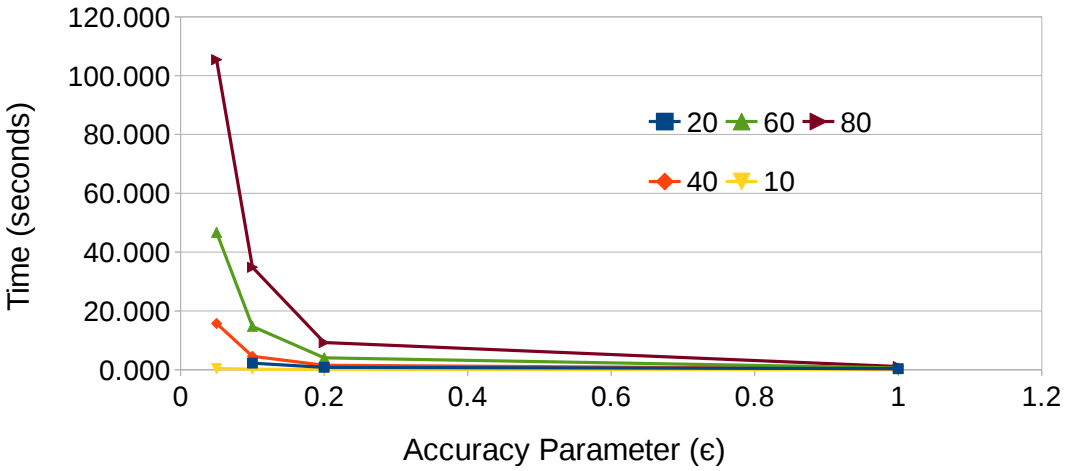


Figure 4.21: Execution Time vs epsilon for different number of nodes

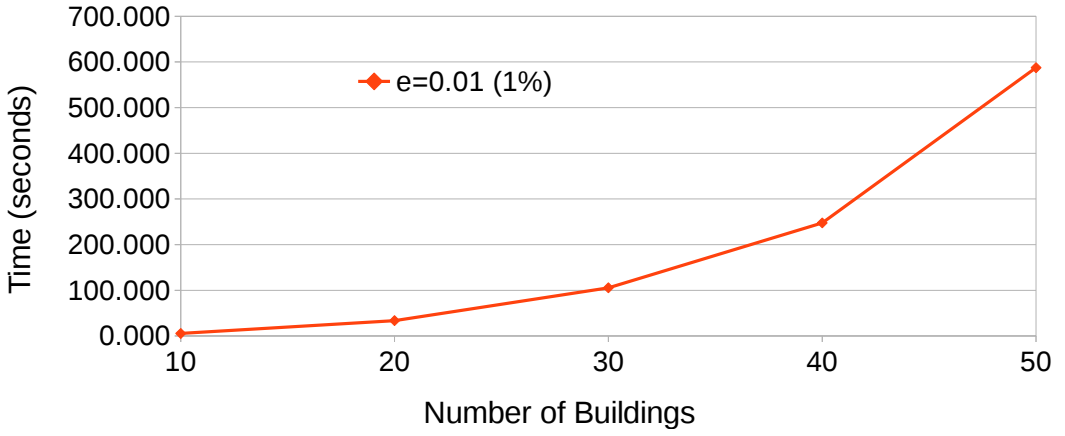


Figure 4.22: Execution Time vs Number of Buildings for $\epsilon = 0.01$

Figure 4.21 shows the trade-off between the accuracy parameter ϵ and the execution time in seconds for various number of nodes. The execution time increases quadratically with respect to $\frac{1}{\epsilon}$. For very small values of ϵ such as 0.01 (1%), this leads to very large

runtimes such as 600 seconds for 50 nodes. However, our algorithm fills the dynamic programming table in such a way that all entries in a row can be independently populated by looking at the entries of previous columns. This provides us with an opportunity to easily parallelize the algorithm to achieve dramatic improvement in the runtime. We do not perform such analysis in this work.

Comparison with State-of-the-art Techniques

We also compared our algorithm against a change making scheduler [76] based heuristic and an LP based algorithm [34] whose results are rounded to the nearest integers. The errors for the LP based algorithm range from 20% to 95% while that for the change maker heuristic range from 2% to 10%. Theoretically, both these heuristics can have unbounded errors. In contrast, the errors incurred by our algorithm are less than 1.5% for $\epsilon = 0.05$ and less than 0.25% for $\epsilon = 0.01$. Table 4.1 compares the theoretical and experimental errors and the computation time of our algorithm for various values of ϵ with state-of-the-art techniques. Our algorithm, with a small increase in runtime is able to provide solutions with extremely low errors as shown in the table.

Table 4.1: Error and Scalability (20 nodes) Comparison

Technique	Theoretical Error	Experimental Error	Time
Change Making [76]	None	2-10%	$\approx$1 second
LP based algorithm [34]	None	20-95%	$\approx$1 second
NO-LESS ($\epsilon = 0.05$)	5%	$<$1.5%	2.2 seconds
NO-LESS ($\epsilon = 0.01$)	1%	$<$0.25%	33 seconds

4.8 Automated Dynamic Demand Response Implementation on USC Campus

We implemented D^{2}R on USC campus micro-grid to demonstrate its large scale feasibility and identify and resolve the challenges associated with practical deployment. Our D^{2}R technique uses learning of occupant energy strategy preferences (at fine grained scales ranging from buildings to floor levels within buildings) to make accurate electricity consumption predictions and individual curtailment recommendations using the algorithms developed in this chapter using only a small subset of consumption data.

4.8.1 D^{2}R Implementation

Buildings on our campus are instrumented with smart metering and control that can be used to implement a large number of advanced energy curtailment strategies such as resetting temperature set points, reducing air flow, duty cycling. A simplified control and data flow diagram of our D^{2}R implementation is shown in Figure 4.23. The micro-grid utility initiates a DR event using OpenADR messages and provides a curtailment target γ to be achieved over a given DR interval T, typically 4 hours. The Policy Engine (PE) module provides campus wide curtailment strategy policy recommendations based on the analysis of historical consumption data and curtailment maximization customer (building) selection algorithms. Smart meter data is aggregated over 15 minute intervals into a consumption database. State-of-the-art data-driven models are then used by the PE module to predict energy consumption values over each 15 minute period comprising the entire DR interval T for each building across campus. This information is then provided to the optimization module in the form of a discrete time varying curtailment matrix. The outputs of the PE are sets of buildings-strategy pairs at 15 minute intervals that will achieve the required curtailment target γ over T.

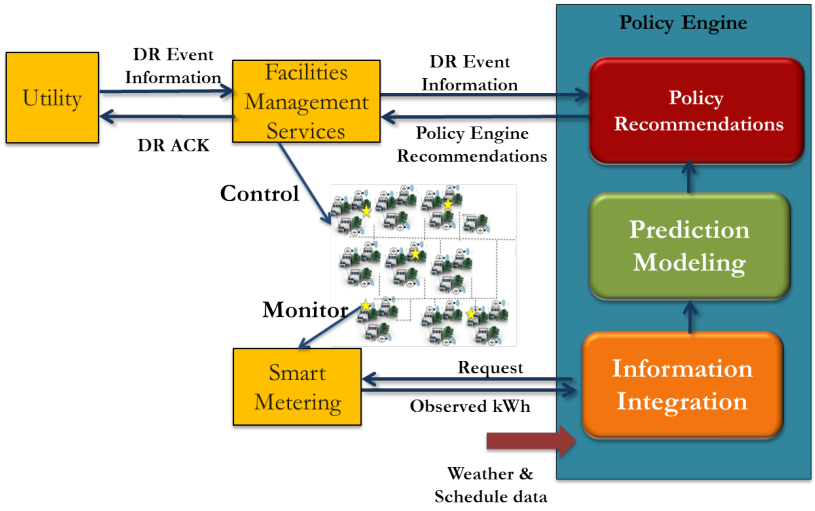


Figure 4.23: Control and Data Flow For D^{2}R Implementation

Our Policy Engine (PE) is composed of an influence model based demand prediction engine that feeds into a building-strategy selection optimization module as shown in Figure 4.24. Historical data from the energy consumption database before the DR day is used in conjuction with time series forecasting techniques such as ARIMA and Lasso-Granger (for learning temporal dependencies among multiple timeseries) to learn occupant energy strategy preferences and load profiles. A significant challenge is to ensure accurate prediction in the absence of high quality consumption data. To ensure quality data we have developed several techniques: interpolation methods for estimating intermittent missing data and sophisticated influence based learning models for estimating systematically unavailable data over larger periods. We have shown that only a small subset ($\approx$7%) of the meters are required in real-time to make predictions for buildings across the campus micro-grid [13].

The results of our adaptive customer prediction models are fed into our optimization module which consists of fast approximation algorithms for ILP based fair energy

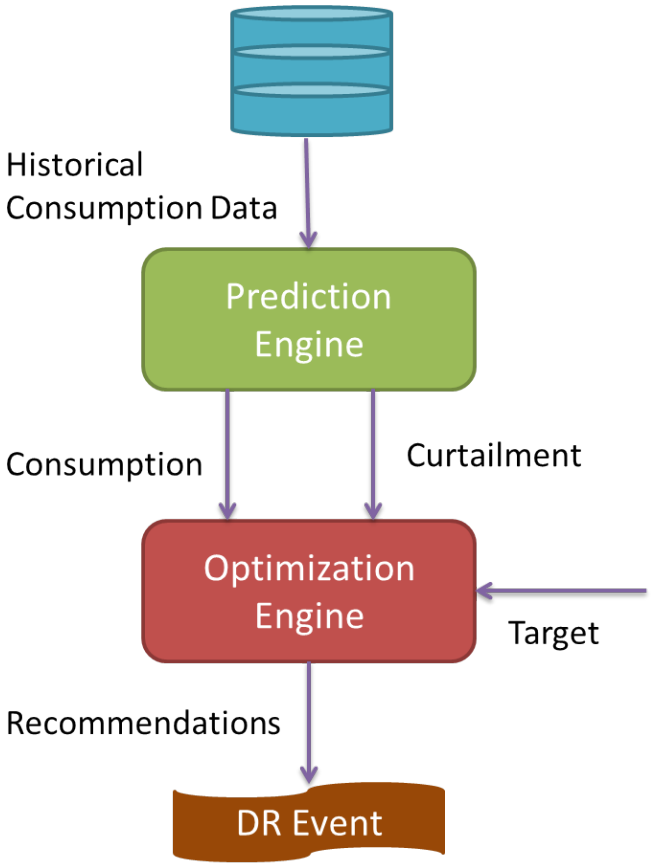


Figure 4.24: Policy Engine

curtailment maximization [37]. Given a targeted curtailment value γ over DR interval T, curtailment matrix M, the optimization module selects sets of building-strategy pairs for each 15 minute interval to achieve the curtailment value γ over T. Note that optimizing for γ over interval T without considering per-period curtailment values might lead to aggressive curtailment in some periods and low curtailment in others. This could be unsustainable to the utility as demands in periods with low curtailment may exceed the generation capacity. Therefore, we have defined and implemented the notion of Sustainable DR (SDR) in which the targeted curtailment value γ is distributed proportionally (as $\{\gamma_t\}$) across interval T [37]. Additionally, we have included the notion of fairness (via

building curtailment budgets) and strategy switching overhead as constraints in our optimization algorithm. We have shown that this problem is NP-hard and have developed fast polynomial time approximation schemes (PTAS) as well as bounded randomized rounding heuristics with provable error bounds i.e. deviation from the curtailment target γ. We have shown that our SDR algorithms achieve results with a very low absolute error of 0.001-0.05 kWh range [37].

4.8.2 Implementation Challenges

Real time quality data availability is a key challenge that we have addressed earlier (Section 4.8.1). Real world challenges such as changing environmental conditions may affect our prediction accuracy, buildings may have to be dropped out of DR due to unresponsiveness or thermal comfort violation. We address such cases by performing dynamic customer re-selection to offset deviations in the curtailment target. Another challenge is scalability while maintaining curtailment accuracy which we have addressed by developing fast polynomial time approximation algorithms with bounded errors. Finally, extending our implementation to a large city environment will face human behavioral challenges. Determining the right customer incentives to obtain a reasonable compliance rate is an open challenge that needs to be addressed.

4.8.3 Overall DR Evaluation

In this section, we discuss the performance of a DR event performed on the campus. 28 buildings were available for the 4 hour long event from 1 pm to 5 pm. Each building had anywhere between 1 to 7 strategies available to it. A targeted curtailment of 6400 kWh was set. the Policy Engine provided building-strategy pair recommendations to the FMS using the predicted curtailment values. It estimated that a curtailment of 2332 kWh can be obtained using the available building-strategy pairs.

Practical challenges such as increase in temperature above the comfort zone, resident complaints etc. lead to dropping out of certain buildings during the DR event. Sometimes buildings fail to respond to the signals for adopting the suggested strategies. Hence, the actual obtained curtailment value was 2103 kWh for the entire DR event as opposed to the estimated 2332 kWh.

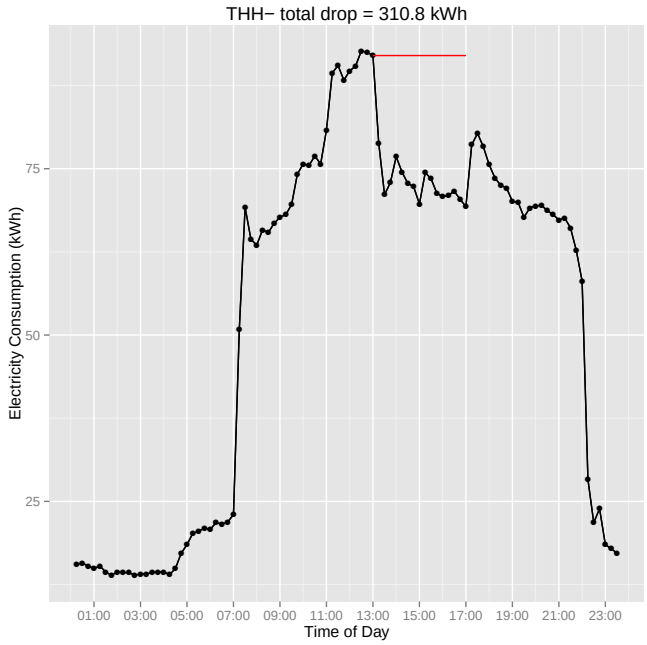


Figure 4.25: Consumption profile of building THH during DR

In Figures 4.25- 4.28, we show the energy consumption profile of a few buildings which achieved the highest curtailment values. The x-axis denotes the time of the day and the y-axis denotes the energy consumption value in kWh. For simplicity, we assume that the consumption in the absence of DR would be the observed value at the start of DR event (1pm). This is denoted by the red line in the figures. The buildings THH, MRF and WPH followed the technique of Variable Frequency Drive Speed Reset (VFD) [50] and obtained a curtailment value of 310, 145 and 98 kWh respectively. The building VKC

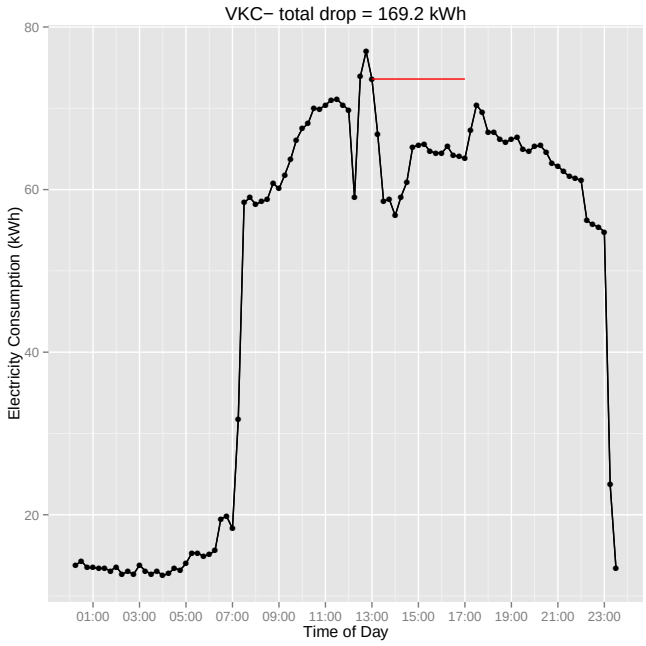


Figure 4.26: Consumption profile of building VKC during DR

adopted a combination of Variable Frequency Drive Speed Reset (VFD) and Equipment Duty Cycling (DUTY) [50] and obtained a curtailment value of 169 kWh.

However, not every building achieved the expected curtailment value. Figures 4.29-4.30 show the consumption profile of buildings BHE and DRB which achieved a negative curtailment value. Due to some practical limitations, these buildings failed to adopt the strategies the FMS signaled them to implement.

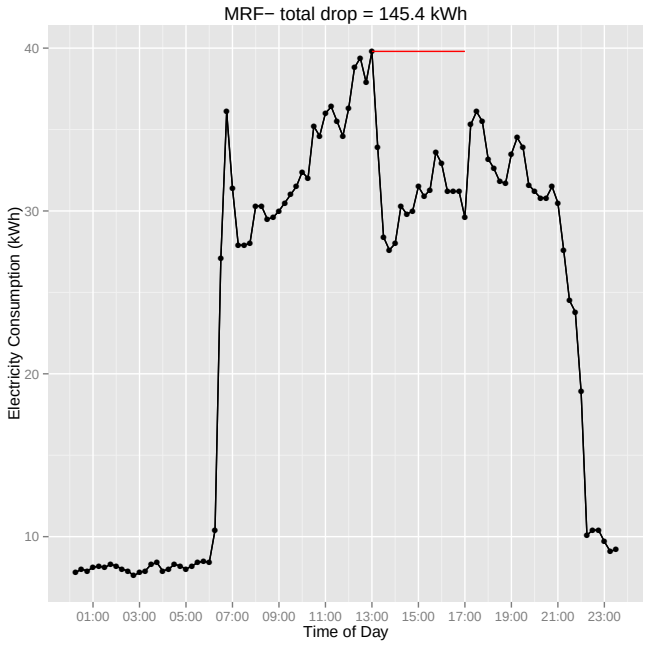


Figure 4.27: Consumption profile of building MRF during DR

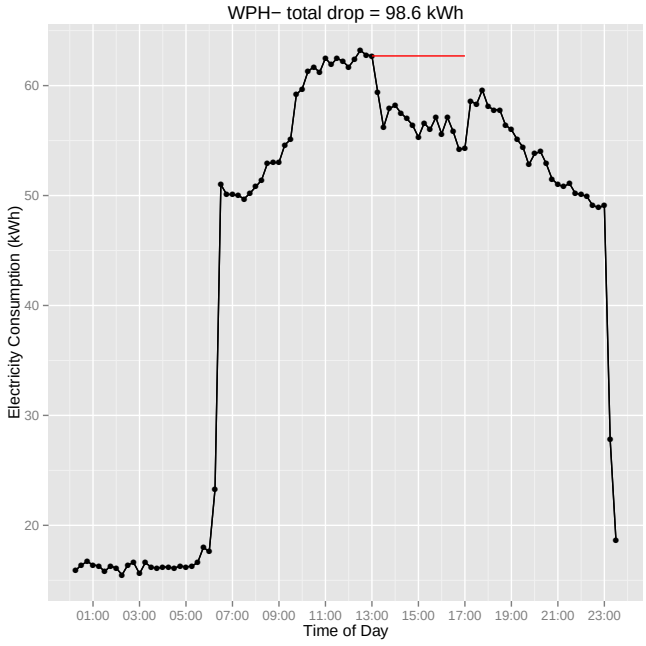


Figure 4.28: Consumption profile of building WPH during DR

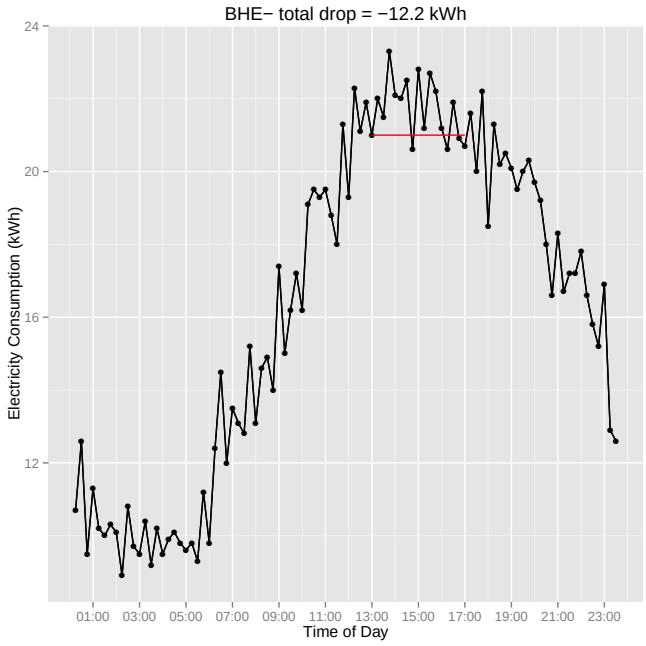


Figure 4.29: Consumption profile of building BHE during DR

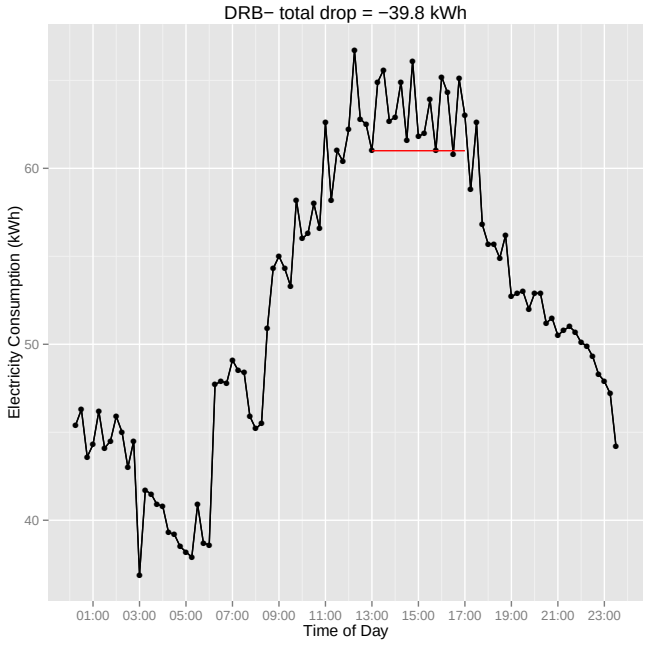


Figure 4.30: Consumption profile of building DRB during DR

Chapter 5

Cost Optimal Supply Demand Matching

In this section, we develop algorithms for discrete supply demand matching by controlling both supply and demand nodes. Algorithms developed in Sections 5.1, 5.2 and 5.4.1 assume a single control mode while the algorithm developed in Section 5.4.2 can be used in dual control mode.

5.1 Minimum Cost Supply Demand Matching

Given the Smart Grid Model above, the objective of the Minimum Cost Supply Demand Matching Algorithm is to determine node-curtailment strategy pairs for each interval of the curtailment horizon such that: (1) The curtailment target Γ_t for each interval t is achieved, (2) the cost is minimized for the entire curtailment horizon, and (3) the aggregate curtailment across the entire curtailment horizon is no more than Γ.

This problem, as we show in Section 3.6 is NP-hard. We first formulate the problem using an Integer Linear Program (ILP). However, the time complexity for solving ILPs is exponential. Hence, we develop a polynomial time approximation algorithm for the same.

More formally, we develop an algorithm with a runtime which is polynomial in the input size M, MN and T and $\frac{1}{\epsilon}$, where ϵ is an approximation guarantee (accuracy) parameter, which ensures that objective of the problem is minimized and in the worst

case the constraints are violated by a maximum factor of $(1 \pm \epsilon)$. Here $M = M_s$ or M_d based on whether supply curtailment is being performed or demand curtailment. To develop the approximation algorithm, we use a dynamic programming algorithm (Equation 5.6) which for each interval t, determines the cost of achieving various curtailment values, each of which is $\geq \Gamma_t$. We then use another dynamic programming algorithm (Equation 5.7) to combine the results of each interval to achieve an aggregated curtailment value of $\leq \Gamma$ with minimum cost. The sizes of the tables of both the dynamic programming algorithms are proportional to the maximum possible cost. This leads to very large runtime to solve the problem exactly. Hence, we scale and round the costs. The scaling and rounding causes several curtailment values to be indistinguishable, this introduces error into the value of our solution. Thus the resulting algorithm is an approximation algorithm (as opposed to exact algorithm) which produces an approximate solution in polynomial time which is independent of the maximum cost. We provide the worst case approximation guarantee for the algorithms.

In the following sections, to simply notations γ, c, x denote the variables corresponding to either supply or demand nodes based on whether supply curtailment is being performed or demand curtailment.

5.1.1 ILP Formulation

The ILP formulation for the Minimum Cost Supply Demand Matching problem is as follows:

$$Minimize: \sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} c_{bj}(t)x_{bj}(t) \tag{5.1}$$

$$s.t. \sum_{b=1}^{M}\sum_{j=1}^{N} \gamma_{bj}(t)x_{bj}(t) \geq \Gamma_t \quad \forall t \tag{5.2}$$

$$\sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} \gamma_{bj}(t)x_{bj}(t) \leq \Gamma \tag{5.3}$$

$$\sum_{j=1}^{N} x_{bj}(t) == 1 \quad \forall b, t \tag{5.4}$$

$$x_{bj}(t) \in \{0, 1\} \quad \forall b, j, t \tag{5.5}$$

Equation 5.2 ensures that the curtailment target for each time interval is achieved. Equation 5.3 ensures that the aggregate curtailment is less than the maximum limit Γ. Equation 5.4 ensures that each node in each time interval follows exactly one strategy (possibly the default strategy with 0 curtailment value).

5.1.2 Approximation Algorithm

Let $\Gamma_{min} = \min_t\{\Gamma_t\}$ be the smallest curtailment target among all the intervals. Define $\mu = \frac{\epsilon\Gamma_{min}}{M}$. For each $\gamma_{bj}(t)$, define $\widehat{\gamma}_{bj}(t) = \lceil\frac{\gamma_{bj}(t)}{\mu}\rceil$. Similarly, define $\widehat{\Gamma}$ and $\widehat{\Gamma}_t \forall t$. We refer to these values as rounded curtailment values.

For each interval t, we define a function $\Theta_t : \mathcal{Z}^+ \cup \{0\} \times \{1, \ldots, M\}$. $\Theta_t(\widehat{\gamma}, b)$ denotes the minimum cost required to achieve a curtailment value of $\widehat{\gamma}$ using nodes $1, \ldots, b$ where $b \in \{1, \ldots, M\}$. Θ_t can be defined recursively as:

$$\Theta_t(\widehat{\gamma}, b) = \begin{cases} \min_j\{c_{bj}(t)\} & \text{if } b = 1 \text{ and} \\ & \exists j \mid \widehat{\gamma} = \widehat{\gamma}_{bj}(t) \\ \infty & \text{if } b = 1 \text{ and} \\ & \widehat{\gamma}! = \widehat{\gamma}_{bj}(t)\ \forall j \\ \infty & \text{if } \widehat{\gamma} < 0 \\ \min_j\{\Theta_t(\widehat{\gamma} - \widehat{\gamma}_{bj}(t), b-1) + \widehat{\gamma}_{bj}(t)\} & \text{otherwise} \end{cases} \tag{5.6}$$

Algorithm 5: Determine node strategy pairs given $(c, \widehat{\gamma}) \in S_t$

Input: $(c, \widehat{\gamma})$, t

$x_{bj}(t) \in X(t) \leftarrow 0 \forall b, j$

$\gamma_{cur} \leftarrow \widehat{\gamma}$

for $b = M$ *to* 1 **do**

  if $b \neq 1$ **then**

    $j \leftarrow \text{argmin}_j\{\Theta(\gamma_{cur} - \widehat{\gamma}_{bj}(t), b-1) + c_{bj}(t)\}$

  else

    $j \leftarrow j \mid \widehat{\gamma}_{bj}(t) == \gamma_{cur}$

  end

  $x_{bj}(t) \leftarrow 1$

  $\gamma_{cur} \leftarrow \gamma_{cur} - \widehat{\gamma}_{bj}(t)$

end

Output: Output $X(t)$, the list of curtailment strategies to be followed by each node in interval t.

The dynamic program can be solved by creating a table of size $k \times M$, where $k = \widehat{\Gamma}$ for each interval. For notational simplicity, we refer to table using the same variable Θ_t as the recursive function. Once the table is filled, for each interval t, we define a set $S_t = \{(\Theta_t(\widehat{\gamma}, M), \widehat{\gamma}) | \widehat{\gamma} \geq \widehat{\Gamma}_t\}$. For any element $(\Theta_t(\widehat{\gamma}, M), \widehat{\gamma}) \in S_t$, Algorithm 5 can be used to determine the strategies to be followed by each node to achieve $\widehat{\gamma}$ with cost $\Theta_t(\widehat{\gamma}, M)$ in the interval t.

Algorithm 6: Minimum Cost Supply Demand Matching Algorithm

Input: $C(t), \gamma(t), \Gamma_t, \forall t, \Gamma$

Compute $\widehat{\gamma}_{bj}(t), \widehat{\Gamma}_t \forall t, \widehat{\Gamma}$

Fill the table $\Theta_t \forall t$ using equation 5.6

Compute $S_t = \{(\Theta_t(\widehat{\gamma}, M), \widehat{\gamma}) | \widehat{\gamma} \geq \widehat{\Gamma}_t\} \forall t$

Fill the table Φ using equation 5.7

$\widehat{\gamma}_{cur} \leftarrow \text{argmin}_{\widehat{\gamma}} \{\Phi(\widehat{\gamma}, T) | \widehat{\gamma} \leq \widehat{\Gamma}\}$

if $\widehat{\gamma}_{cur} == \phi$ **then**

    No curtailment strategies exist

    Exit Algorithm

end

for $t = T$ *to 1* **do**

    if $t \neq 1$ **then**

        $j \leftarrow \text{argmin}_{j \mid (c_j, \widehat{\gamma}_j) \in S_t} \{\Phi(\widehat{\gamma}_{cur} - \widehat{\gamma}_j, t-1) + c_j\}$

    else

        $j \leftarrow j \mid \widehat{\gamma} = \widehat{\gamma}_j, (c_j, \widehat{\gamma}_j) \in S_t$

    end

    Run Algorithm 1 with $(c_j, \widehat{\gamma}_j)$ to get $X(t)$

    $\widehat{\gamma}_{cur} \leftarrow \widehat{\gamma}_{cur} - \widehat{\gamma}_j$

end

Output: $X(t) \forall t$, the list of curtailment strategies to be followed by each node in each interval

Now, given $S_t \forall t \in \{1, \ldots, T\}$, we need to select exactly one element $e_t = (c_t, \widehat{\gamma}_t) \in S_t \forall t$ such that $\sum_{t=1}^{T} \widehat{\gamma}_t \leq \widehat{\Gamma}$ and $\sum_{t=1}^{T} c_t$ is minimized. We define a function $\Phi : \mathcal{Z}^+ \cup \{0\} \times \{1, \ldots, T\}$. $\Phi(\widehat{\gamma}, t)$ denotes the minimum cost required to achieve the curtailment value of $\widehat{\gamma}$ and time intervals $1, \ldots, t$. Φ can be defined recursively as:

$$\Phi(\widehat{\gamma},t)=\begin{cases}\min_j\{c_j\} & \text{if } t=1 \text{ and} \\ & \exists j \mid \widehat{\gamma}=\widehat{\gamma}_j,(c_j,\widehat{\gamma}_j)\in S_t \\ \infty & \text{if } t=1 \text{ and} \\ & \widehat{\gamma}!=\widehat{\gamma}_j\ \forall j \mid (c_j,\widehat{\gamma}_j)\in S_t \\ \infty & \text{if } \widehat{\gamma}<0 \\ \min_{j \mid (c_j,\widehat{\gamma}_j)\in S_t} \{\Phi(\widehat{\gamma}-\widehat{\gamma}_j,t-1) + \widehat{\gamma}_j\} & \text{otherwise}\end{cases} \tag{5.7}$$

This dynamic program requires a table of size $k \times T$, where $k=\widehat{\Gamma}$. Again, for notational simplicity, we refer to table using the same variable Φ as the recursive function. Algorithm 6 can be used to determine the curtailment achieved in each interval and the corresponding node strategy pairs.

Theorem 7. *Algorithm 6 is a polynomial time algorithm for minimum cost supply demand matching which in the worst case violates the maximum curtailment constraint (Equation 5.3) by at most* $(1+\epsilon)$ *factor and violates the per interval curtailment target constraint (Equation 5.2) by at most* $(1-\epsilon)$ *factor.*

Proof. For a curtailment value γ, we say that $\widehat{\gamma}$ is the rounded curtailment value. Also, we call γ as the unrounded curtailment value. *Correctness:* We define *Bucket* of $\widehat{\gamma}$ as the range $\mu\widehat{\gamma}-\mu<\gamma\leq\mu\widehat{\gamma}$ i.e. all the curtailment values which get rounded to $\widehat{\gamma}$. $C^{\Theta}_{\widehat{\gamma}}$ denotes the cost assigned to the bucket in table Θ. For each interval t, using induction on Equation 5.6, it is easy to show that $C^{\Theta_t}_{\widehat{\gamma}}\ \forall t, 0\leq\widehat{\gamma}\leq\widehat{\Gamma}$ will be the minimum cost required to achieve any curtailment value in the Bucket of $\widehat{\gamma}$ for table Θ_t. Hence, for each element $e_t=(c_t,\widehat{\gamma}_t)$, c_t is the minimum possible cost to achieve any curtailment value

in the Bucket of $\widehat{\gamma}_t$ corresponding to the table Θ_t. Now, if we can show that the range of curtailment values covered by the elements in $S_t \forall t$ contains the curtailment value chosen by any optimal solution, and that the cumulative curtailment value of any optimal solution at any time t is contained in the entries $\Phi(:, t)$ (i.e. entries corresponding to time interval t), then using induction on Φ table, we can show the aggregate cost of the solutions obtained by Algorithm 6 will be less than or equal to the optimal solution. Now, in each interval, the lowest rounded curtailment value $\widehat{\gamma}$ considered to create the set S_t is $\widehat{\gamma}_t$. Hence, the lowest unrounded curtailment γ satisfies: $\gamma_t - \mu < \gamma \leq \gamma_t$ i.e. $\gamma \leq \gamma_t$. Now, the maximum rounded curtailment value $\widehat{\gamma}$ considered by Algorithm 2 in the Φ table is $\widehat{\Gamma}$. So, the maximum curtailment value γ considered in the corresponding bucket is $\mu\widehat{\Gamma} \geq \Gamma$. Hence, the range of curtailment values considered by Algorithm 6 covers the range in which optimal solution can reside and so Algorithm 6 does not miss any possible solution of a lower cost.

Runtime: The Algorithm fills T Θ tables each of which is of size $M\widehat{\Gamma}$. We assume that $\Gamma = O(T\Gamma_{min})$. Each entry of Θ requires $O(N)$ time. Hence, the total runtime for line 2 and 3 of Algorithm 6 is $O(\frac{M^2NT^2}{\epsilon})$.

The number of entries in table Φ is $O(T\widehat{\Gamma})$. Each entry requires $O(\widehat{\Gamma})$ time. Hence, the total required time to fill Φ is $O(T\widehat{\Gamma}^2 = O(\frac{T^3M^2}{\epsilon^2}))$. Once the tables are filled, the for loop requires $O(TMN\widehat{\Gamma})$ to output the strategies. Hence, the algorithm is polynomial in the input size M, N, T and $\frac{1}{\epsilon}$.

Approximation Guarantee: Let $\widehat{\Gamma}_x = \sum_{i=1}^{M}\sum_{t=1}^{T}\widehat{\gamma}_{it}$ be the solution from our algorithm, where $\widehat{\gamma}_{it}$ denotes the rounded curtailment by node i in time t. From line 3 of the algorithm, for all t, we know that $\sum_{i=1}^{M}\widehat{\gamma}_{it} \geq \widehat{\Gamma}_t$. Also, $\frac{\gamma_{it}}{\mu} \leq \widehat{\gamma}_{it} \leq \frac{\gamma_{it}}{\mu} + 1$ by the definition of $\widehat{\gamma}_{it}$. This implies that $\frac{\Gamma_t}{\mu} \leq \sum_{i=1}^{M}(\frac{\gamma_{it}}{\mu} + 1) \leq \sum_{i=1}^{M}\frac{\gamma_{it}}{\mu} + M$. So, $\Gamma_t - \mu M \leq \sum_{i=1}^{M}\gamma_{it} \implies \sum_{i=1}^{M}\gamma_{it} \geq \Gamma_t - \epsilon\Gamma_{min} \geq (1-\epsilon)\Gamma_t$ as $\Gamma_t \geq \Gamma_{min}$. Hence, in

each interval, the curtailment target constraint (Equation 5.2) is violated by a maximum factor of $(1-\epsilon)$.

Now, from line 5 of the algorithm, $\widehat{\Gamma}_x \leq \widehat{\Gamma}$. So, $\frac{\Gamma_x}{\mu} \leq \frac{\Gamma}{\mu} + 1$. This means that $\Gamma_x \leq \Gamma + \epsilon\frac{\Gamma_{min}}{M} \leq \Gamma(1+\frac{\epsilon}{M}) \leq \Gamma(1+\epsilon)$ as $\Gamma_{min} \leq \Gamma$ and $M \geq 1$. Hence, the aggregate curtailment target constraint (Equation 5.3) is violated by a maximum factor of $(1+\epsilon)$. □

5.2 Minimum Cost Supply Demand Matching with Fairness

Curtailment from a node leads to a loss of utility for the node. Hence, it would be unfair to force some nodes to incur losses due to high curtailment while leaving others with minimal curtailment. The Minimum Cost Supply Demand Matching Algorithm discussed in the previous section does not take fairness into account and can lead to solutions with uneven curtailment values from the nodes. We address the issue of fairness in this section by assigning a curtailment budget range (which can be set by the grid operator) to each node. The algorithm, by ensuring that no node incurs a curtailment more or less than its budgeted range over each curtailment horizon, ensures that supply demand is done in a fair manner.

Similar to the previous problem, we first develop an ILP formulation for this problem. We then relax the ILP into a Linear Program (which can be solved in polynomial time) and round back the results to integers. This rounding, however, violates certain constraints and increases the objective value. We provide guarantees on the worst case violation of the constraints and the increase in the objective value in the worst case.

5.2.1 ILP Formulation

Let $[\alpha_b B_b, B_b]$ be the curtailment budget for node b with $\alpha_b \in [0,1]$. Both B_b and α_b are determined by the grid operator. The problem of minimum cost supply demand matching with fairness can be formulated using the following ILP:

$$Minimize: \sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} c_{bj}(t)x_{bj}(t) \tag{5.8}$$

$$s.t. \sum_{b=1}^{M}\sum_{j=1}^{N} \gamma_{bj}(t)x_{bj}(t) \geq \Gamma_t \quad \forall t \tag{5.9}$$

$$\sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} \gamma_{bj}(t)x_{bj}(t) \leq \Gamma \tag{5.10}$$

$$\alpha_b B_b \leq \sum_{j=1}^{N}\sum_{t=1}^{T} \gamma_{bj}(t)x_{bj}(t) \leq B_b \quad \forall b \tag{5.11}$$

$$\sum_{j=1}^{N} x_{bj}(t) == 1 \quad \forall b,t \tag{5.12}$$

$$x_{bj}(t) \in \{0,1\} \quad \forall b,j,t \tag{5.13}$$

Equation 5.11 is the additional constraint added that ensures that each node curtails an amount within its budgeted range.

5.2.2 Approximation Algorithm

The ILP formulated in Section 5.2.1, when relaxed to a Linear Program will lead to unbounded errors. Hence, we first redefine the ILP. We define $\gamma'_b(t) - \gamma'_{bj}(t) = \gamma_{bj}(t)$, where $\gamma'_b(t)$, is the maximum operational value for node b at time t and $\gamma'_{bj}(t)$ is the

reduced operational value of node b at time t by following strategy j with curtailment value $\gamma_{bj}(t)$. the redefined ILP is as follows:

$$Minimize: \sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} c_{bj}(t)x_{bj}(t) \tag{5.14}$$

$$s.t. \sum_{b=1}^{M}\sum_{j=1}^{N} \gamma'_{bj}(t)x_{bj}(t) \leq \sum_{b=1}^{M} \gamma'_{b}(t) - \Gamma_t \quad \forall t \tag{5.15}$$

$$\sum_{b=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T} \gamma_{bj}(t)x_{bj}(t) \leq \Gamma \tag{5.16}$$

$$\sum_{j=1}^{N}\sum_{t=1}^{T} \gamma_{bj}(t)x_{bj}(t) \leq B_b \quad \forall b \tag{5.17}$$

$$\sum_{j=1}^{N}\sum_{t=1}^{T} \gamma'_{bj}(t)x_{bj}(t) \leq \sum_{t=1}^{T} \gamma'_{b}(t) - \alpha_b B_b \quad \forall b \tag{5.18}$$

$$\sum_{j=1}^{N} x_{bj}(t) == 1 \quad \forall b, t \tag{5.19}$$

$$x_{bj}(t) \in \{0, 1\} \quad \forall b, j, t \tag{5.20}$$

We note that the above ILP is an instance of k column sparse Packing Integer Problem (PIP) i.e. a packing integer problem in which there in an upper limit k on the number of constraints a variable can occur in [15]. k column sparse PIPs have a $ek+O(k)$ factor randomized rounding based approximation algorithm for a fixed k [15].

This implies that even if we assume $\Gamma \geq \sum_{b=1}^{M} B_b$ i.e. the constraint (5.16) is redundant and that $\alpha_b = 0, \forall b$ each variable $x_{ij}(t)$ appears in three constraints: (1) Per interval curtailment target constraint (5.15), (2) Budget constraint (5.17), and (3) constraint (5.19). Hence, using the approximation algorithm developed in [15] we can get a worst case bound of > 8 which is unsatisfactory for reliable grid operations.

Hence, to develop an approximation algorithm with tighter theoretical worst case bounds, we first make the following assumption: The costs $c_{bj}(t)$ are a function of

$\gamma_{bj}(t)$ i.e., $c_{bj}(t) = f(\gamma_{bj}(t))$. We will derive approximation guarantees when the function f is linear and when it is quadratic. In order to develop a bounded approximation algorithm, we first relax the ILP to a linear program i.e., we replace Equation 5.20 with $0 \leq x_{bj}(t) \leq 1 \ \forall b, j, t$ and solve the Linear Program. Solving a linear program takes polynomial amount of time using methods such as inter-point or ellipsoid [20]. However, the solution will contain fractional values for the decision variables $x_{bj}(t), \ \forall b, j, t$ which need to be rounded to 0 or 1 for a valid solution. Now, naively rounding the decision variables leads to errors which are unbounded. Hence, we develop Algorithm 7 which is a novel rounding algorithm which guarantees that the constraints are violated by at most a factor of 2 in the worst case. For each b, t, the algorithm works by computing expected curtailment $\gamma' = \sum_{j=1}^{N} \gamma_{bj}(t) x_{bj}(t)$ and rounding it to the curtailment value $\gamma_{bj}(t)$ nearest to it. We have the following two results for this algorithm.

Theorem 8. *For a linear cost function f, Algorithm 7 is a (2,2)-factor Minimum Cost Supply Demand Matching with Fairness Algorithm. The cost of the solution achieved by Algorithm 7 is at most twice the optimal while the budget and targeted curtailment constraints (Eqs 5.16 and 5.17) are violated by at most a factor of two.*

Theorem 9. *For a quadratic cost function f, Algorithm 7 is (4,2)-factor algorithm.*

Proof. Let $\gamma'_{bt} = \sum_{j=1}^{N} \gamma_{bj}(t) x_{bj}(t)$ be the curtailment value obtained for node b in time interval t. Let γ^i_{bt} and γ^{i+1}_{bt} be the curtailment values of strategies between which γ'_{bt} falls i.e. $\gamma^i_{bt} \leq \gamma'_{bt} \leq \gamma^{i+1}_{bt}$. Now, if γ'_{bt} is rounded up to γ^{i+1}_{bt}, this implies $\gamma'_{bt} \geq (\gamma^i_{bt} + \gamma^{i+1}_{bt})/2 \geq \gamma^{i+1}_{bt}/2$. Summing over all values of b, t ensures that Equation 5.16 and the upper bound of Equation 5.17 are violated by at most a factor of 2. Similarly it can be shown that if the objective function is linear, it will be bounded by a factor of 2 and if it is quadratic, it will be bounded by a factor of 4.

Now, in order to provide a bound on the constraint violation of Equations 5.15 and 5.18, we note that if γ'_{bt} is rounded up, the bounds are trivially satisfied. However, if they are rounded down, then guarantee no longer holds. Hence, we make an assumption that $\gamma_{bt}^{i+1} \leq (2k-1)\gamma_{bt}^{i}$. Using this assumption, we can ensure that in the worst case $\gamma_{bt}^{i} \geq \gamma'_{bt}/k$ i.e. the constraint is violated at most by a factor of k. □

The analysis above can be used by the grid operator to improve the efficiency of supply demand matching. The grid operator can setup curtailment configurations (e.g. load curtailment strategies such as GTR, etc.) such that the difference between two consecutive curtailment values is not vary large. However, it is not always possible to control the curtailment configurations. Hence, the grid operator can develop techniques by noticing that a higher curtailment value can be expected to have less violations for two reasons: (1) very few values of γ'_{bt} will have $\gamma_{bj}^{i} = 0$ and (2) since a large number of non-zero curtailment strategies will be selected, several of them will be rounded up thus reducing the possibility that Equation 5.10 is violated by a large factor. Hence, if the required curtailment value is small for a curtailment horizon, the grid operator can reduce the number of nodes which participate in the curtailment horizon.

Note that the above guarantees are worst case guarantees. In practice, the performance is significantly better as shown using the experimental results.

5.3 Online Algorithm for Fair Supply Demand Matching

The algorithms discussed in the previous two sections require the availability of the curtailment prediction data for the entire horizon. However, certain scenarios require supply demand matching in an online manner. At the beginning of each interval, the

Algorithm 7: Minimum Cost Supply Demand Matching with Fairness

Input: $C(t), \gamma(t), \Gamma_t, \forall t, \Gamma, B_b, \alpha_b \forall b$

$x_{bj}(t) \in X(t) \leftarrow 0 \forall b, j, t$

Relax the ILP to an LP by replacing Equation 5.13 with $0 \leq x_{bj}(t) \leq 1 \ \forall b, j, t$

Solve LP to obtain solution $x^*_{bj}(t) \forall b, j, t$

foreach b, t **do**

  $\gamma' \leftarrow \sum_{j=1}^{N} \gamma_{bj}(t) x^*_{bj}(t)$

  Let $\gamma_{bi}(t) \leq \gamma' \leq \gamma_{bi+1}(t)$

  if $(\gamma' - \gamma_{bi}(t)) \geq (\gamma_{bi+1}(t) - \gamma')$ **then**

    $x_{bi+1}(t) \leftarrow 1$

  else

    $x_{bi}(t) \leftarrow 1$

  end

end

Output: $X(t) \forall t$, the list of curtailment strategies to be followed by each node in each interval

data is made available and supply demand matching needs to be performed in a myopic way.

We develop a greedy online heuristic algorithm (Algorithm 8) for this problem. The algorithm finds a minimum cost way to achieve the curtailment target $\Gamma^o_l = \Gamma_t$ for the current interval while ensuring that no node curtails more than the budget for the current interval. The upper limit of the budget for the current interval B^o_b for each node b is determined by multiplying the ratio of $\frac{B_b}{\sum_{t=1}^{T} \Gamma_t}$ with Γ^o_l. The lower bound is simply $\alpha_b B_b$. The upper bound on curtailment Γ^o_u is determined by multiplying the ratio of $\frac{\Gamma}{\sum_{t=1}^{T} \Gamma_t}$ with Γ^o_l. Let γ^o denote the curtailment matrix and C^o denote the cost matrix with γ^o_{bj} being the curtailment for node b following strategy j and c^o_{bj} being its cost. We remove any values γ^o_{bj} which are outside the curtailment budget range from the curtailment matrix γ^o. Note that the values $\Gamma, \Gamma_t, B_b \forall b$ are unknown for the current supply demand matching horizon. They can be obtained from some past horizon.

Algorithm 8: Online Algorithm for Fair Supply Demand Matching

Input: $\Gamma_l^o, \gamma^o, C^o, \Gamma, \Gamma_t \forall t, B_b, \alpha_b \forall b$

1 $\Gamma_u^o \leftarrow \frac{\Gamma}{\sum_{t=1}^{T} \Gamma_t} \Gamma_l^o$

2 $B_b^o \leftarrow \frac{B_b}{\sum_{t=1}^{T} \Gamma_t} \Gamma_l^o \ \forall b$

3 Compute $C_{min}^o, C_{max}^o, \widehat{c}_{bj}^o, \widehat{C}_{max}^o$ similar to Algorithm 6 using γ^o in which curtailment values outside curtailment budget range are removed.

4 Fill table Θ using equation 6

5 $c^* \leftarrow min_{\widehat{c}}\{\widehat{c} | \Gamma_l^o \leq \Theta(\widehat{c}, M) \leq \Gamma_u^o\}$

6 Run Algorithm 5 with $(c^*, \Theta(c^*, M))$ to get X^o

Output: X^o, the list of curtailment strategies

5.4 Supply Demand Matching with Network Constraints

The Supply Demand Matching algorithms discussed in Section 5.1 and 5.2 are agnostic to the underlying network model of the smart grid. This can lead the framework to output solutions which violate network capacity constraints. In this section, we will develop a Supply Demand Matching Framework which explicitly models the Smart Grid network (Section 3.4) and ensures that the solutions output satisfy the constraints imposed by the capacity limitations of the various components of the Smart Grid.

5.4.1 Transformer Level Supply Demand Matching

Traditional smart grids are not equipped with bi-directional LV Feeders i.e. the feeders can carry power from the sub-station to the transformers but not in the other direction. This implies $Cap_f = 0 \forall f \in \mathcal{F}$. Hence, Supply Demand Matching needs to be performed at each transformer of the Smart Grid. We develop Algorithm 9 for the same.

Algorithm 9 takes as input the time varying cost and curtailment matrices for both supply and demand nodes, the distributor capacities and the per interval maximum supply (demand) for each supply (demand) node. In lines 1 and 2, it initializes the decision

Algorithm 9: Transformer Level Supply Demand Matching Algorithm

Input: $C_s(t), C_l(t), \gamma_s(t), \gamma_l(t)\ \forall t; Cap_{tx}\ \forall tx; S_{b_s}(t)\ \forall b_s \in \mathcal{S}\ \forall t; L_{b_d}(t)\ \forall b_d \in \mathcal{D},\ \forall t$

1 $X_s(t) \leftarrow 0$
2 $X_l(t) \leftarrow 0$
3 **foreach** $tx \in \mathcal{T}$ **do**
4 **foreach** $t \in \{1, \ldots, T\}$ **do**
5 $Sup_{tx}(t) = \sum_{b_s \in \mathcal{T}(tx) \cap \mathcal{S}} (S_{b_s}(t))$
6 $Dem_{tx}(t) = \sum_{b_d \in \mathcal{T}(tx) \cap \mathcal{D}} (L_{b_d}(t))$
7 $\Gamma_s(t) \leftarrow \max\{Sup_{tx}(t) - Cap_{tx}, 0\}$
8 $\Gamma_l(t) \leftarrow Dem_{tx}(t) - Sup_{tx}(t) + \Gamma_s(t)$
9 **if** $\Gamma_l(t) \leq 0$ **then**
10 $\Gamma_s(t) \leftarrow -\Gamma_l(t)$
11 $\Gamma_l(t) \leftarrow 0$
12 **end**
13 **end**
14 $\Gamma_s \leftarrow$ grid operator determined upper bound on solar curtailment.
15 $\Gamma_l \leftarrow$ grid operator determined upper bound on load curtailment.
16 $X_s^{tx}(t) \leftarrow$ output of Algorithm 6 with inputs $C_s(t), \gamma_s(t), \Gamma_s(t) \forall t, \Gamma_s$
17 $X_l^{tx}(t) \leftarrow$ output of Algorithm 6 with inputs $C_l(t), \gamma_l(t), \Gamma_l(t) \forall t, \Gamma_l$
18 $X_s(t) \leftarrow X_s(t) + X_s^{tx}(t) \forall t$
19 $X_l(t) \leftarrow X_l(t) + X_l^{tx}(t) \forall t$
20 **end**

Output: $X_s(t), X_l(t)$ the list of solar and load curtailment strategies for each node in each interval

matrices to 0. Now, for each transformer tx, in each time interval t (foreach blocks at lines 3 and 4), it calculates the total solar supply (line 5) and the total demand (line 6). If the supply is more than the capacity of the distributor, the curtailment target $\Gamma_s(t)$ is set to be the excess supply (line 7). Now, if demand is greater than the total supply (minus the supply curtailment required to satisfy the distributor capacity constraint), a demand curtailment target $\Gamma_l(t)$ is identified (line 8). The If block starting at line 9 handles the cases where supply is more than the demand. In such a scenario, the supply should be curtailed for supply demand matching. Algorithm 6 is then used to determine the per interval load and supply curtailment strategies for each node (lines 16 and 17).

The upper bounds as required by Algorithm 6 can be determined by the grid operator (lines 14 and 15). Note that in the interest of simplicity, we make an assumption that only the required amount of curtailment is performed and any unnecessary curtailment is avoided. For example, for a transformer, with solar surplus, if a lower cost solution can be found by performing both load and solar curtailment, it is not considered. Our algorithm can easily be extended to handle the cases where the assumption is not true.

Theorem 10. *Algorithm 9 is a polynomial time algorithm for transformer level minimum cost supply demand matching with network constraints which in the worst case violates the distributor constraints (Equation 3.3) by at most* $(1 - \epsilon)$ *factor.*

Proof. In each transformer, Algorithm 6 is being called with Cap_{tx} as the curtailment target. As Algorithm 6 is a polynomial time algorithm which produces a solution of minimum cost while violating the curtailment target by at most $(1 - \epsilon)$, so does Algorithm 9. □

5.4.2 Smart Grid Level Supply Demand Matching

The increased distributed generation in future smart grids will lead to a more active participation by the consumers and will required bi-directional power exchanges [29] using LV-feeders. This implies that the maximum power that can be injected from the transformers i.e. $Cap_f \geq 0$. This will allow us to perform more cost efficient supply demand matching by optimally distributing the supply from surplus transformers to the deficit ones.

We develop Algorithm 10 to perform smart grid level supply demand matching at any given time interval t. The brief description of Algorithm 10 is as follows: For each transformer, the algorithm first checks whether the solar generation from the nodes attached is within its capacity or not. If not, then the nodes are marked for the required

curtailment. Now, in the resulting grid (the grid in which the nodes have performed the required curtailment), the algorithm partitions the feeders into two sets: one with supply surplus ($\mathcal{S}_{sup}$) and one with load surplus ($\mathcal{S}_{dem}$). For each set, the minimum costs of performing all possible curtailments are identified. Finally, the algorithm chooses supply and load curtailments such that the cost is minimized and the total supply after solar curtailment equals the total demand after load curtailment. Unlike the transformer level supply demand matching in Section 5.4.1, which distributes curtailment temporally to minimize costs, this algorithm performs spatial curtailment distribution in each interval separately.

We define additional notations $\gamma_{sf}(t), \gamma_{df}(t), C_{sf}(t), C_{df}(t), X_{sf}(t), X_{df}(t)$ denoting the sub-matrices of the original matrices defined in Section 3.4 corresponding to supply (s) or demand (d) nodes attached to feeder f for the given time interval t. Let $\Gamma_{min} = \min_{s,d,f}\{\Gamma_{sf}^{min}(t), \Gamma_{df}(t)\}$, where $\Gamma_{sf}^{min}(t)$ and $\Gamma_{df}(t)$ are defined in lines 8 and 11 of Algorithm 10. Also let $F_{max} = \max_f\{|\mathcal{F}(f)|\}$ Define $\mu = \frac{\epsilon\Gamma_{min}}{F_{max}}$. We define $\widehat{\gamma}_{sf}(t) = \lceil\gamma_{sf}(t)\rceil$, $\widehat{\gamma}_{df}(t) = \lceil\gamma_{df}(t)\rceil$. We refer to these values as rounded curtailment values. Similarly, we define $\widehat{\gamma}_{sf}^{min}(t), \widehat{\gamma}_{sf}^{max}(t)$. For each feeder f, we define a function $\Theta_f : \mathcal{Z}^+ \cup \{0\} \times \{1, \ldots, |\mathcal{F}(f)|\}$, where $|A|$ outputs the size of the set A. $\Theta_f(\widehat{\gamma}, b)$ denotes the minimum cost required to achieve a curtailment value of $\widehat{\gamma}$ using nodes $1, \ldots, b$ where $b \in \{1, \ldots, |\mathcal{F}(f)|\}$. To avoid redundancy in notations, if Θ_f is used to denote supply curtailment, any load consuming node b is assumed to have 0 supply curtailment strategies. Θ_f can be defined recursively as:

$$\Theta_f(\widehat{\gamma}, b) = \begin{cases} \min_j c_{bj}(t) & \text{if } b = 1 \text{ and } \exists j \mid \widehat{\gamma} = \widehat{\gamma}_{bj}(t) \\ \infty & \text{if } b = 1 \text{ and } \widehat{\gamma}! = \widehat{\gamma}_{bj}(t)\ \forall j \\ \infty & \text{if } \widehat{\gamma} < 0 \\ \min_j \{\Theta_f(\widehat{\gamma} - \widehat{\gamma}_{bj}(t), b-1) + c_{bj}(t)\} & \text{otherwise} \end{cases} \tag{5.21}$$

Note that this is similar to the dynamic program of Equation 5.6 and hence Algorithm 5 can be used to determine the strategies to be followed by each node connected to the feeder f. For each f, if $f \in \mathcal{S}_{sup}$, we define $S_f = \{(\Theta_f(\widehat{\gamma}, M), \widehat{\gamma}) | \widehat{\Gamma}_{sf}^{min}(t) \leq \widehat{\gamma} \leq \widehat{\Gamma}_{sf}^{max}(t)\}$, where $\widehat{\Gamma}_{sf}^{max}(t), \widehat{\Gamma}_{sf}^{min}(t)$ are defined in line 11. The minimum curtailment requirement ensures that the feeder capacity constraints are met. If $f \in \mathcal{S}_{dem}$, then $S_f = \{(\Theta_f(\widehat{\gamma}, M), \widehat{\gamma}) | \widehat{\Gamma}_{df}(t) \geq \widehat{\gamma}\}$.

Now, we create two dynamic programming tables Φ_{sup} and Φ_{dem} using the sets $S_f \forall f \in \mathcal{S}_{sup}$ and $S_f \forall f \in \mathcal{S}_{dem}$ respectively. The tables are built using functions $\Phi_{sup} : \mathcal{Z}^+ \cup \{0\} \times \{1, \ldots, |S_{sup}|\}$ and $\Phi_{dem} : \mathcal{Z}^+ \cup \{0\} \times \{1, \ldots, |S_{dem}|\}$. The functions $\Phi_{sup}(\widehat{\gamma}, f)$ denotes the minimum cost required to achieve the curtailment value of $\widehat{\gamma}$ using supply nodes from feeders $\{1, \ldots, f\}$ and is defined using the recursive function:

$$\Phi_{sup}(\widehat{\gamma}, f) = \begin{cases} \min_j c_j & \text{if } f = 1 \text{ and } \exists j \mid \widehat{\gamma} = \widehat{\gamma}_j, (c_j, \widehat{\gamma}_j) \in S_f \\ \infty & \text{if } f = 1 \text{ and } \widehat{\gamma}! = \widehat{\gamma}_j\ \forall j \mid (c_j, \widehat{\gamma}_j) \in S_f \\ \infty & \text{if } \widehat{\gamma} < 0 \\ \min_{j \mid (c_j, \widehat{\gamma}_j) \in S_f} \{\Phi(\widehat{\gamma} - \widehat{\gamma}_j, f-1)\ + c_j\} & \text{otherwise} \end{cases} \tag{5.22}$$

$\Phi_{dem}(\widehat{\gamma}, f)$ is defined similarly. Leveraging the tables built using Equations 5.21 and 5.22, Algorithm 10 performs minimum cost smart grid level supply demand matching. $\widehat{S}$ and $\widehat{D}$ used in equation 19 are the rounded aggregated solar and load surplus in the Smart Grid respectively. A detailed line by line description of Algorithm 10 is provided later in Section 20. We have the following theorem.

Theorem 11. *Algorithm 10 is a polynomial time algorithm for smart grid level minimum cost supply demand matching with network constraints which in the worst case violates the feeder and distributor constraints (Equations 3.3 and 3.6) by at most* $(1-\epsilon)$ *factor. Moreover, the curtailment achieved by Algorithm 10 is within* $(1 \pm 2\epsilon)$ *factor of the optimal curtailment target which can mitigate the supply demand mismatch.*

Proof. Correctness of the algorithm can be proved using induction, hence we exclude the details.

Runtime: The dominating term in the runtime complexity will be the creation of the Φ_{dem} and Φ_{sup} tables. Each of these tables will require creation of Θ tables requiring $O(M^2 N\widehat{\Gamma})$, by assuming that $|\mathcal{F}| = O(M)$ i.e. the number of nodes and $\max_f\{|\mathcal{F}(f)|\} = O(M)$ and $\Gamma = \max_{s,d,f}\{\Gamma_{sf}^{max}, \gamma_{df}(t) \times M\}$. This is equal to $O(\frac{M^4 N}{\epsilon})$ with the assumptions that $F_{max} = O(M)$ and $\Gamma = O(M\Gamma_{min})$. The creation of Φ_{dem} and Φ_{sup} tables after creating the Θ tables will require $O(M\widehat{\Gamma}^2) = O(\frac{M^5}{\epsilon^2})$. Hence, the algorithm is polynomial in the input size and $\frac{1}{\epsilon}$.

The assumption $\Gamma = O(M\Gamma_{min})$ implicitly implies that no transformer level load/solar curtailment value is "too small". If this is not true, we can isolate the transformer with the smallest curtailment target and solve it independently. We can repeat the process until the condition is satisfied. This will lead to a possibly higher cost solution but will significantly reduce the computation time required.

Approximation Guarantee: Let $\widehat{\Gamma}_x = \sum_i \widehat{\gamma}_i$ be the rounded curtailment values selected from the nodes connected to a transformer. Now, we know that $\widehat{\Gamma}_{cap} \leq \widehat{\Gamma}_x$,

where $\widehat{\Gamma}_{cap}$ is the minimum curtailment required to satisfy capacity constraints (as determined in Equations 8 and 11). So, $\frac{\Gamma_{cap}}{\mu} \leq \sum_i (\frac{\gamma_i}{\mu} + 1) \leq \sum_i \frac{\gamma_i}{\mu} + F_{max}$, where Γ_{cap} and γ_i are the unrounded curtailment values. This implies $\Gamma_{cap} - \epsilon\Gamma_{min} \leq \Gamma_x \implies \Gamma_x \geq (1-\epsilon)\Gamma_{cap}$, where again Γ_x is the unrounded curtailment value. Hence, the capacity constraints are violated by a maximum factor of $(1-\epsilon)$.

Now, in order to prove the second part of the theorem, without loss of generality, let supply be surplus in the Smart Grid. Let $\widehat{S}$ be the aggregate supply and $\widehat{D}$. Let $\widehat{\Gamma}_{sx} = \sum_i \widehat{\gamma}_{si}$ be the solar curtailment values chosen and let $\widehat{\Gamma}_{dx} = \sum_i \widehat{\gamma}_{di}$ be the chosen load curtailment values. Let $\Gamma = S - D$ be the required unrounded curtailment value and let $\Gamma_x = \gamma_{sx} - \gamma_{dx}$ be the achieved unrounded curtailment value. Now, we know $\widehat{S} - \widehat{\gamma}_{sx} = \widehat{D} - \widehat{\gamma}_{dx}$. By rearranging and unrounding, we can get 1: $S + \gamma_{dx} \leq D + \gamma_{sx} + \mu + \mu M$ and 2: $D + \gamma_{sx} \leq S + \gamma_{dx} + \mu + \mu M$. Using 1, by substitution, we get $\Gamma - \epsilon\Gamma_{min}(1 + \frac{1}{M}) \leq \Gamma_x \implies \Gamma(1-2\epsilon) \leq \Gamma_x$ as $M \geq 1$. Using 2, by substitution, we get $\Gamma_x \leq \Gamma + \epsilon(1 + \frac{1}{M})\Gamma_{min} \implies \Gamma(1+2\epsilon)$ as $M \geq 1$. So, the curtailment achieved by Algorithm 10 is within $(1 \pm 2\epsilon)$ factor of the optimal curtailment. □

Algorithm 10 Description

Algorithm 10 takes as input the cost C_s, C_l and curtailment γ_s, γ_l matrices for all the supply and demand nodes, the capacity of each distributor Cap_{tx} and each feeder Cap_f and the maximum supply S_{b_s} and maximum demand L_{b_d} for each supply or demand node respectively. It then initializes two empty sets (line 1). S_{sup} denotes the feeders where the maximum possible supply is greater than the maximum possible demand and S_{dem} are the feeders with maximum possible demand greater than supply. The decision variables are all set to 0. For each feeder (for each block starting at line 2), the algorithm calculates the maximum possible supply and demand (lines 3 and 4). In line 5, the algorithm identifies the minimum supply curtailment required to meet the distributor

Algorithm 10: Smart Grid Level Supply Demand Matching Algorithm for time interval t

Input: $C_s(t), C_l(t), \gamma_s(t), \gamma_l(t); Cap_{tx}\ \forall tx; Cap_f\ \forall f; S_{b_s}(t)\ \forall b_s \in \mathcal{S}; L_{b_d}(t)\ \forall b_d \in \mathcal{D}$

1 $\mathcal{S}_{sup} \leftarrow \phi$; $\mathcal{S}_{dem} \leftarrow \phi$; $X_{sf}(t) \leftarrow 0; \forall s \in \mathcal{F}(f) \cap \mathcal{S}, \forall f$; $X_{df}(t) \leftarrow 0; \forall d \in \mathcal{F}(f) \cap \mathcal{D}, \forall f$

2 **foreach** $f \in \mathcal{F}$ **do**

3 $\quad Sup_f(t) = \sum_{b_s \in \mathcal{F}(f) \cap \mathcal{S}}(S_{b_s}(t))$

4 $\quad Dem_f(t) = \sum_{b_d \in \mathcal{F}(f) \cap \mathcal{D}}(L_{b_d}(t))$

5 $\quad \Gamma_{sf}(t) \leftarrow \max\{Sup_f(t) - Cap_{\mathcal{F}^{tx}(f)}, 0\}$

6 $\quad$ **if** $Dem_f(t) \geq Sup_f(t) - \Gamma_{sf}(t)$ **then**

7 $\quad\quad \mathcal{S}_{dem} \leftarrow f$

8 $\quad\quad \Gamma_{df}(t) \leftarrow Dem_f(t) - Sup_f(t) + \Gamma_{sf}(t)$

9 $\quad$ **else**

10 $\quad\quad \mathcal{S}_{sup} \leftarrow f$

11 $\quad\quad \Gamma^{max}_{sf}(t) \leftarrow Sup_f(t) - Dem_f(t)$; $\Gamma^{min}_{sf}(t) \leftarrow \max\{Sup_f(t) - Cap_f, \Gamma_{sf}(t), 0\}$

12 $\quad$ **end**

13 **end**

14 **foreach** $f \in \mathcal{S}_{dem}$ **do**

15 $\quad X_{sf}(t) \leftarrow$ output of MinCB with inputs $C_{sf}(t), \gamma_{sf}(t), \Gamma_{sf}(t), \Gamma_{sf}(t)$

16 $\quad$ Create Φ_{dem} using Equations 5.21 and 5.22, Let S_{demf} be the sets created after Equation 5.21

17 **end**

18 **foreach** $f \in \mathcal{S}_{sup}$ **do**

19 $\quad$ Create Φ_{sup} using Equations 5.21 and 5.22, Let S_{supf} be the sets created after Equation 5.21

20 **end**

21 $\Phi(\widehat{\gamma}) \leftarrow \Phi_{sup}(\widehat{S} - \widehat{D} + \widehat{\gamma}, |\mathcal{S}_{sup}|) + \Phi_{dem}(\widehat{\gamma}, |\mathcal{S}_{dem}|)\ \forall \widehat{\gamma} \geq 0$

22 Call Algorithm 11 to get $X_{sf}(t), X_{df}(t)$

Output: $X_{sf}(t), X_{df}(t)$ the list of curtailment strategies for each node in the interval t.

constraints. Then, in the if-then-else block starting at line 6, if the maximum demand in a feeder is greater than the supply (after curtailment required for satisfying distributor constraints), the feeder is inserted into S_{dem} and a maximum demand curtailment target is identified to ensure a feasible supply demand matching configuration. Otherwise, the feeder is inserted into S_{sup} and a minimum and maximum supply curtailment

Algorithm 11: Finding node strategy pairs from Φ

Input: Φ

1 $\widehat{\gamma}_{cur} \leftarrow \text{argmin}_{\widehat{\gamma}}\{\Phi(\widehat{\gamma})\}$
2 **for** $f = |\mathcal{S}_{sup}|$ *to 1* **do**
3 **if** $f \neq 1$ **then**
4 $j \leftarrow \text{argmin}_{\{j \mid (c_j,\widehat{\gamma}_j) \in S_{supf}\}}\{\Phi_{sup}(\widehat{\gamma}_{cur} - \widehat{\gamma}_j, f-1) + c_j\}$
5 **else**
6 $j \leftarrow j \mid \widehat{\gamma} = \widehat{\gamma}_j, (c_j, \widehat{\gamma}_j) \in S_{supf}$
7 **end**
8 Run Algorithm 5 with $(c_j, \widehat{\gamma}_j)$ over Θ_f to get $X_{sf}(t)$
9 $\widehat{\gamma}_{cur} \leftarrow \widehat{\gamma}_{cur} - \widehat{\gamma}_j$
10 **end**
11 $\widehat{\gamma}_{cur} \leftarrow \text{argmin}_{\widehat{\gamma}}\{\Phi(\widehat{\gamma})\}$
12 **for** $f = |\mathcal{S}_{dem}|$ *to 1* **do**
13 **if** $f \neq 1$ **then**
14 $j \leftarrow \text{argmin}_{\{j \mid (c_j,\widehat{\gamma}_j) \in S_{demf}\}}\{\Phi_{dem}(\widehat{\gamma}_{cur} - \widehat{\gamma}_j, f-1) + c_j\}$
15 **else**
16 $j \leftarrow j \mid \widehat{\gamma} = \widehat{\gamma}_j, (c_j, \widehat{\gamma}_j) \in S_{demf}$
17 **end**
18 Run Algorithm 5 with $(c_j, \widehat{\gamma}_j)$ over Θ_f to get $X_{df}(t)$
19 $\widehat{\gamma}_{cur} \leftarrow \widehat{\gamma}_{cur} - \widehat{\gamma}_j$
20 **end**

Output: $X_{sf}(t), X_{df}(t)$ the list of curtailment strategies for each node in the interval t.

is identified. The minimum supply curtailment is required for satisfying the feeder and distributor constraints while the maximum denotes the curtailment after which supply demand matching is not possible due to less supply. In the foreach blocks (lines 14 and 18), sets S_{demf} and S_{supf} are created by filling dynamic programming tables Φ_{dem} and Φ_{sup} respectively. Additionally, for the demand surplus feeders, any supply curtailment required to meet the distributor constraints is performed using Algorithm 5 (with time horizon of just 1 interval). Now, in line 21, for each valid rounded demand curtailment $\widehat{\gamma}$, the cost of operation is calculated. Cost of operation is the sum of $\widehat{\gamma}$ demand curtailment and $\widehat{S} - (\widehat{D} - \widehat{\gamma})$ supply curtailment. The latter curtailment is required to match supply with demand. Here, $\widehat{S}$ and $\widehat{D}$ are the rounded aggregate maximum

solar and demand surplus in the smartgrid respectively. These are the values which are available to perform smart grid level supply demand matching while ensuring that within each feeder, the distributor and feeder constraints are satisfied and feasible supply demand matching configurations are maintained. Now it calls Algorithm 11 which in lines 1 to 20, using the value of $\widehat{\gamma}$ identified to ensure minimum cost, the dynamic programming tables are traversed to find suitable strategies for each node in each feeder and the corresponding decision variables are set. As the tables Θ_f are similar to the Θ table filled in Algorithm 6, Algorithm 5 is used to traverse the same.

5.5 Modeling Storage for Supply Demand Matching Algorithms

Operational states of energy storage systems are continuous in nature as opposed to the discrete operational states of supply and demand nodes considered in the work. It is trivial to incorporate storage in the Integer Linear Program based algorithm of Section 5.2 by adding continuous variables denoting storage charge/discharge and converting it into a Mixed Integer Linear Program (MILP). However, incorporating storage into algorithms developed in Section 5.1, Section 5.4.1 and Section 5.4.2 requires us to discretize the charge/discharge parameter in such a way that the runtime is not affected. In order to incorporate storage in dynamic programming based approximation algorithms, we define $\widehat{R} = \lceil \frac{R}{\mu} \rceil, \widehat{G} = \lceil \frac{G}{\mu} \rceil$ and $\widehat{D} = \lceil \frac{D}{\mu} \rceil$. For each interval t, we define the charging strategies of storage as $\widehat{G}_t = \{0, 1, \ldots, \widehat{G}\}$ and discharging strategies as $\widehat{D}_t = \{0, 1, \ldots, \widehat{D}\}$. Using these variables, the storage operation can be defined as

$$\widehat{R}_{t+1} = \widehat{R}_t + \eta_G \widehat{G}_t - \frac{1}{\eta_D} \widehat{D}_t \tag{5.23}$$

and the constraints are defined as follows:

$$\mu T \frac{1}{\eta_D} \leq \widehat{R}_{t+1} \qquad \text{Discharge Mode} \tag{5.24}$$

$$\widehat{R}_{t+1} \leq \widehat{R} - \mu \eta_G T \quad \text{Charge Mode} \tag{5.25}$$

$$0 \leq \widehat{D}_t \leq \widehat{D} \tag{5.26}$$

$$0 \leq \widehat{G}_t \leq \widehat{G} \tag{5.27}$$

Now, if the algorithm chooses $\widehat{D}_t$ as discharge, $\min\{\mu\widehat{D}_t, D\}$ is discharged. Similarly, if the algorithm chooses $\widehat{C}_t$ as charge, $\min\{\mu\widehat{C}_t, C\}$ is charged. This ensures that the maximum charging/discharging constraints are met. Equations 5.24 and 5.25 ensure that the storage operation is always within storage capacity. This is because the maximum difference between the charge values obtained by our algorithm: μG_x and actual charge: G_a is bounded by $\mu\eta_G T$ and the discharge range is bounded by $\mu T\frac{1}{\eta_D}$. Moreover, when $\widehat{R}_T \leq \mu T\frac{1}{\eta_D}$, the storage cannot be used in discharge mode or when $\widehat{R}_T \geq \widehat{R} - \mu\eta_G T$, it cannot be used in charge mode.

It is evident that this storage model leads to a $\eta\mu T$ amount of storage being unused, where $\eta = (\eta_G + \frac{1}{\eta_D})$. The value Γ_{min} used to define μ can be modified to provide an upper bound on the unused storage. For example, if the storage capacity $R \geq \frac{1}{k}\frac{\Gamma_l}{M}$, where Γ_l is the aggregate of minimum curtailment target across all the intervals, then modifying Γ_{min} to $\frac{1}{\eta k}\Gamma_{min}$ will bound the unused storage by $\mu T = \frac{\eta\epsilon\Gamma_{min}T}{\eta kM} \leq \frac{\eta\epsilon\Gamma_l}{\eta kM} \leq \epsilon R$. The condition $R \geq \frac{1}{k}\frac{\Gamma_l}{M}$ implies that the storage capacity is at least $\frac{1}{k}$ times the average curtailment produced by a node in an interval to achieve the target Γ_l. However, the downside here is that this leads to a $(\eta k)^2$ factor increase in the runtime. The typical charge/discharge efficiency for Li-ion batteries is 80-90% [4], so $\eta \approx 2.15$.

Now, when the algorithms fill the dynamic programming table in Equations 5.6 and 5.21, instead of iterating through N strategies to find the minimum, the algorithm

iterates through $\widehat{D}+1$ strategies if the storage is used for discharging or $\widehat{G}+1$ strategies if it is used for charging. This increases the runtime of filling Θ by a factor of $O(\frac{\widehat{\Gamma}}{N})$. However, the runtime of Algorithm 6 and Algorithm 10 still remain $O(\frac{T^3M^2}{\epsilon^2})$ and $O(\frac{M^5}{\epsilon^2})$ respectively.

5.6 Results and Analysis

In addition to providing theoretical guarantees, we perform practical evaluations of the algorithms developed. We implemented the algorithms using MATLAB [8]. The LP and ILP algorithms were implemented using IBM ILOG Cplex Optimization Studio [1] which is a toolkit for mathematical and constraint programming. The experiments were performed on Dell optiplex with 4-cores and 4 GB RAM. A supply demand matching horizon of 32 15-min intervals was considered with 16 intervals of load curtailment and 16 intervals of solar curtailment.

We evaluated the algorithms by varying the (L,U) pair where, for notational simplicity, L=$\sum_{t=1}^{T}\Gamma_t$ and U=Γ (Section 5.2.1). Note that L and U are the lower and upper bound on the curtailment to be achieved in the curtailment horizon and hence represents the feasible curtailment range. The costs of the strategies were evaluated using the function $f(\gamma)=2\gamma^2$, where γ is the curtailment value of the strategy. Section 5.6.1 describes the input dataset generation. As the dataset was generated from historical data, perfect knowledge of the future was assumed with no prediction errors.

5.6.1 Dataset

We obtained the load curtailment data from the demand response implementation on our University Campus. Our campus consists of 150 DR enabled nodes (buildings) each of which can follow 6 load curtailment strategies. The load curtailment values for each

node-strategy pair was generated using algorithms mentioned in [12]. We varied the load curtailment target from 500 to 1500 kWh.

Unlike load curtailment data for which we had a real world dataset, we had to simulate solar curtailment data. The output of a solar PV is determined mainly by the solar radiance at the PV installation, the PV area and the PV yield [7]. We used the hourly solar radiance data available at [5] for the Los Angeles County. We then used the PV-output calculator available at [7] to calculate the solar generation data by varying the PV area from $10m^2$ to $20m^2$. We also varied the solar panel yield from 5% to 15%. Hence, a fixed PV area and yield represents a node in our dataset. To obtain solar curtailment values, if the PV output for a given hour for a node was O, we generated 6 curtailment values: 0, $0.125 * O, 0.25 * O, 0.5 * O, 0.75 * O, O$. Hence, each curtailment value represents a PV connection/disconnection setting. All the 4 15-min intervals for a given hour were assigned the same curtailment values. For clarity, given an (L, U) pair, we report the error of either solar or load curtailment, whichever one performs worse. We used a precision of 3 decimal places for both load and solar curtailment values.

5.6.2 Minimum Cost Supply Demand Matching

We evaluate our minimum cost supply demand matching (MinCB) Algorithm by varying the (L,U) pairs as discussed above. For each (L,U) we study the amount by which the curtailment target constraint (Equation 5.2) is violated. Moreover, we also compare the cost of the solution obtained by our algorithm to the optimal cost obtained by the ILP. We also perform a scalability analysis of our algorithm.

Accuracy Analysis

Figures 5.1 and 5.2 show the percentage error of the curtailment target constraint (Equation 5.2) violation for various values of the theoretical guarantee ϵ. For example, if

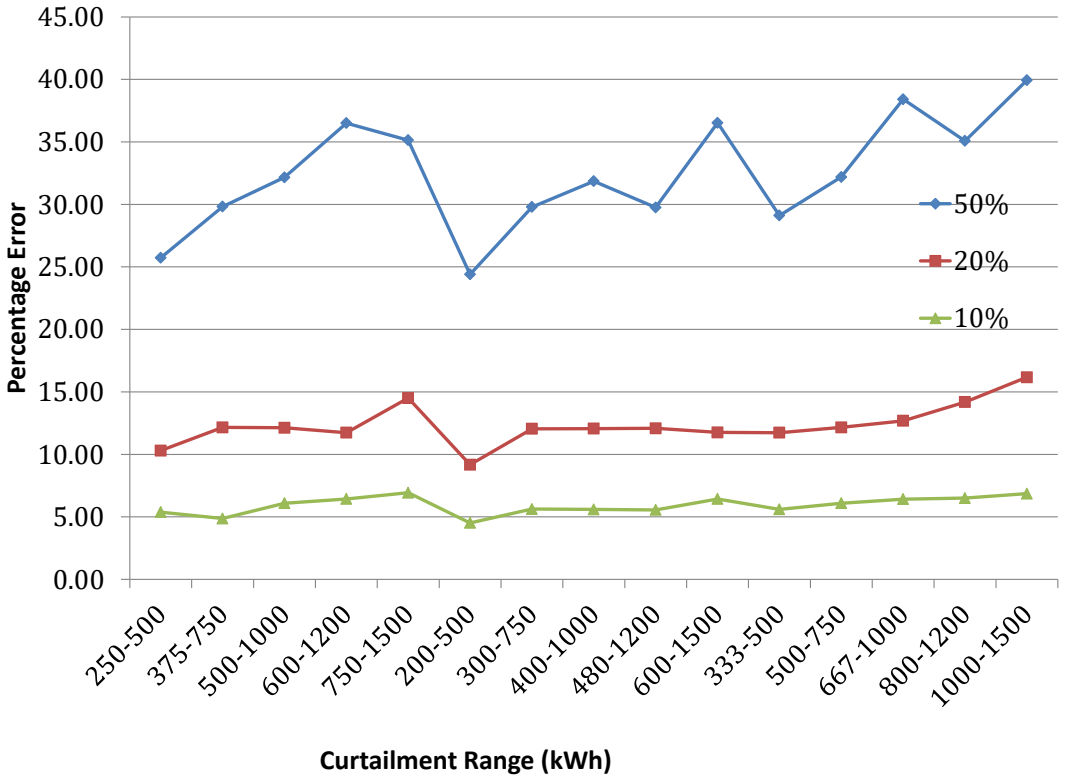


Figure 5.1: Percentage Error of the curtailment target constraints for ϵ = 0.5-0.1 (50-10% Error Guarantee)

$\epsilon = 0.05$, the algorithm will incur an error of 5% in the worst case. As we can note from the figures, the errors incurred by our solution are within the theoretical guarantees provided by the number ϵ. In practice, the errors are much lower. For $\epsilon = 0.5$ (50%), the highest error incurred is 40%. Similarly, for $\epsilon = 0.2$ (20%), barring a few cases, the errors are less than 15%.

As evident from Theorem 1, the cost of the solutions obtained from MinCB are less than or equal to the optimal solution. This is possible because instead of tightly satisfying the constraint, as the optimal solution does, MinCB tries to find a solution with a lower cost which possibly violates the constraints by a maximum of ϵ. Figure 5.3 shows the ratio of the cost of solution obtained by MinCB and the optimal solution

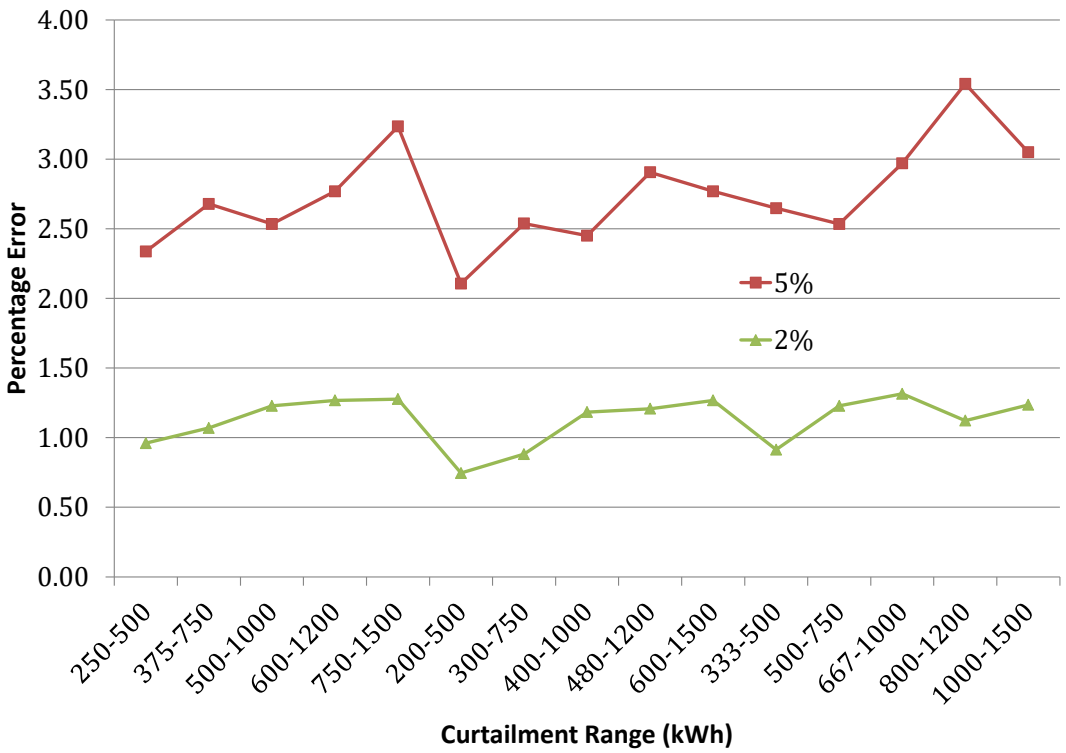
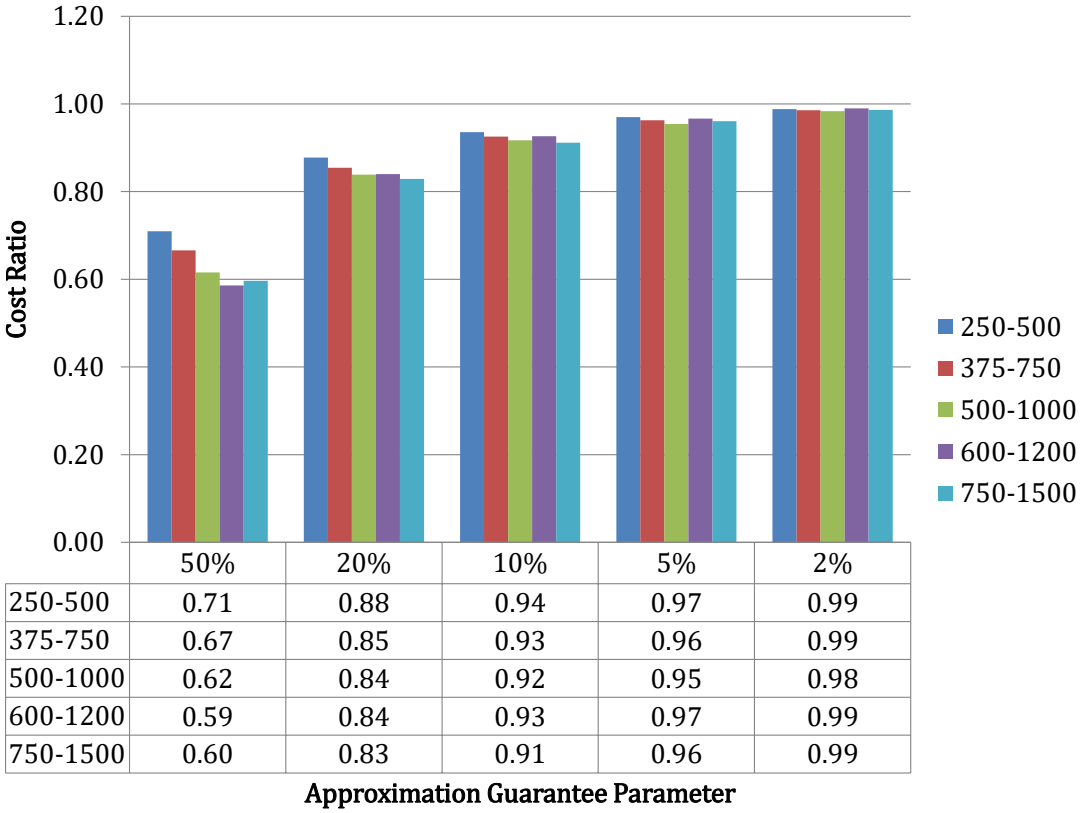


Figure 5.2: Percentage Error of the curtailment target constraints for $\epsilon = 0.05$-0.02 (5-2% Error Guarantee)

versus the approximation guarantee (ϵ). As we can notice from the figure, for each curtailment range, the ratio increases as the approximation guarantee is tightened i.e. reduced. For lower values of ϵ such as 0.02 (2%), the ratio is close to 1. The ratio is never greater than 1 implying that the cost of the solution obtained by MinCB is always less than the optimal cost.

We can also note that for a fixed approximation guarantee (ϵ), the ratio typically decreases with an increase in the upper bound U of the curtailment range. This trend is more pronounced in higher values of ϵ such as 0.5 (50%) and 0.2 (20%). This is because a higher value of U provides a larger error range in which to search for minimum cost



	50%	20%	10%	5%	2%
250-500	0.71	0.88	0.94	0.97	0.99
375-750	0.67	0.85	0.93	0.96	0.99
500-1000	0.62	0.84	0.92	0.95	0.98
600-1200	0.59	0.84	0.93	0.97	0.99
750-1500	0.60	0.83	0.91	0.96	0.99

Figure 5.3: Ratio of the cost of solution obtained by MinCB and the optimal solution versus the approximation guarantee (ϵ)

solutions. For example, for 50% error guarantee, the error range which is 750 kWh for U = 1500, is three times larger than the error range of 250 kWh for U = 500.

Scalability Analysis

In order to perform scalability analysis, we fix the values of T: the number of time intervals and N: the number of curtailment strategies per node. We vary M: the number of nodes for various values of ϵ. As one can see from Figure 5.4, the algorithm exhibits a near quadratic increase in the runtime with respect to the number of nodes. This is consistent with the runtime complexity analysis.

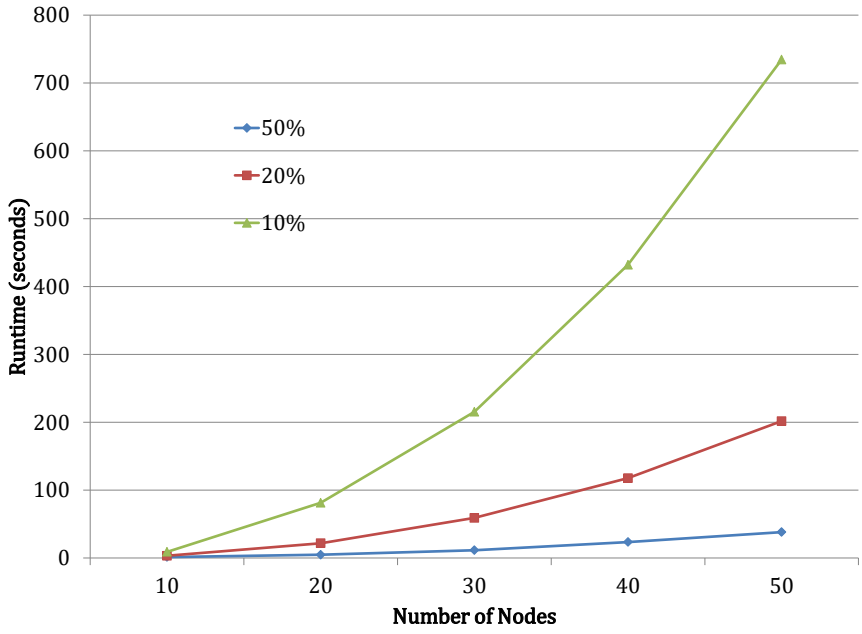


Figure 5.4: Runtime of MinCB versus: (a) Number of nodes for various values of epsilon

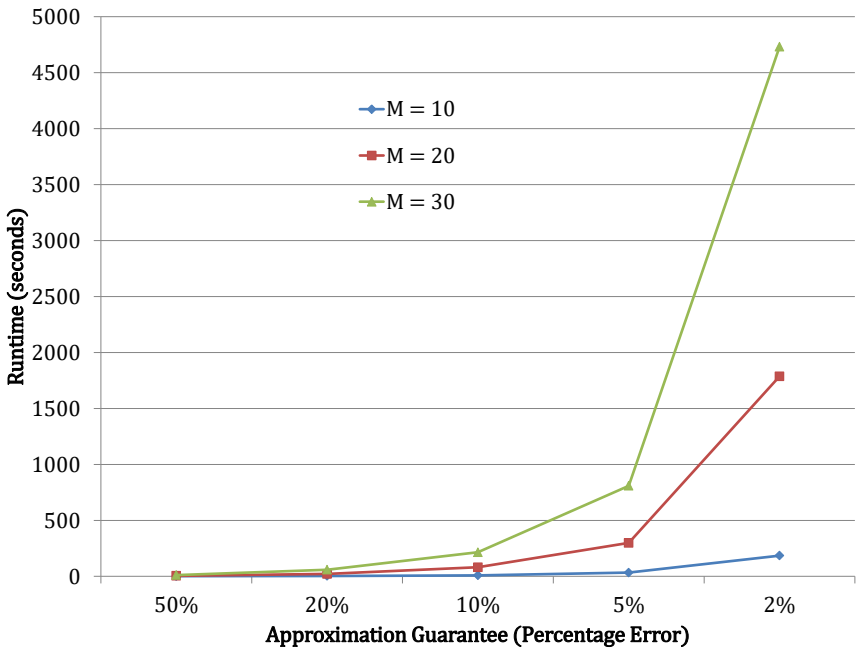


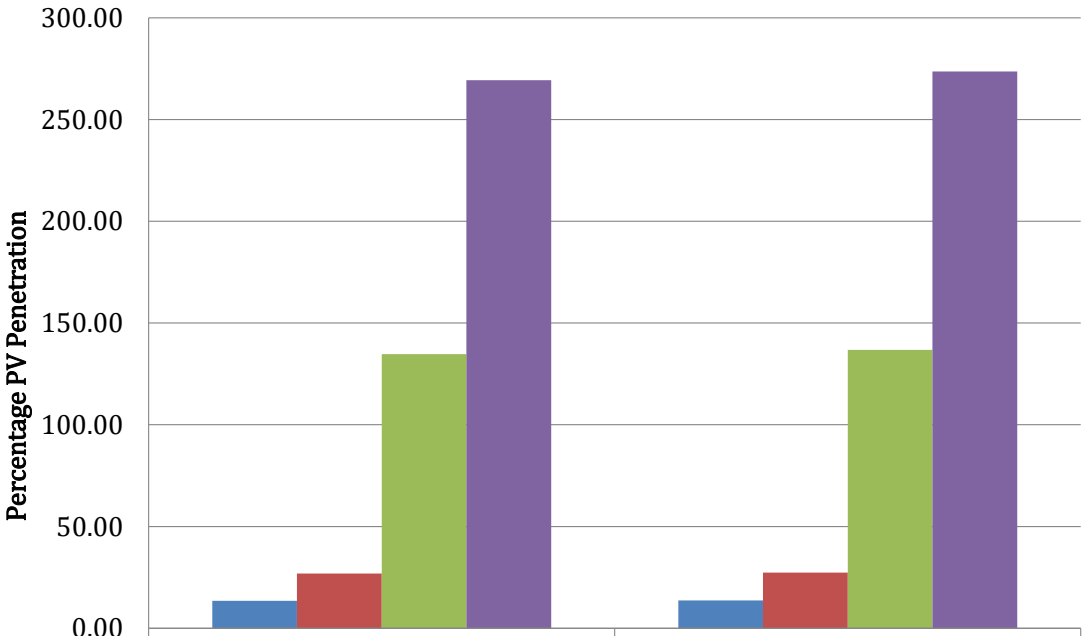
Figure 5.5: Runtime of MinCB versus: (b) ϵ (denoted as percentage error) for various values of the number of nodes

We also analyze the scalability with respect to ϵ by varying the value of M while keeping T and N fixed. As shown in Figure 5.5, decreasing ϵ (increasing accuracy) has a significant impact on runtime. Hence, ϵ is a parameter that can be used to trade-off accuracy versus computational complexity.

A reader might comment that the runtimes observed, especially for smaller values of ϵ are very high. This can be justified as the problem of supply demand matching is NP-hard and hence, high computation capacity is required to increase the accuracy. Our objective in this work is to show that a polynomial time approximation algorithm exists for this NP-hard problem. We did not focus on finding the best optimal solution for the same. Moreover, our experiments are performed on MATLAB. For a real world deployment of this software, using faster programming languages such as C++ will significantly improve the run times (as high as 10-20 times as per the experience of the authors). In the context of real world scenarios, the California ISO's MRTU applications determine the desired generation changes 5-min ahead of the beginning of the interval and the system needs to start moving towards the set point 2.5 minutes ahead of the interval [44]. Given the typical inverter control delays of the order of milliseconds [33], even our naive MATLAB implementation meets the California ISO constraints for 40 nodes with $\epsilon = 0.2$ (20% error) and 25 nodes for $\epsilon = 0.1$ (10% error). Similar implementation in C++, assuming a conservative estimate of 10x improvement will meet the constraints for 70 nodes with $\epsilon = 0.1$ (10% error) and 40 nodes with $\epsilon = 0.05$ (5% error).

To further improve the runtime, parallelization techniques can be used as the algorithm is amenable to parallelization trivially. The T Θ_t tables can be filled in parallel. Moreover, all the entries corresponding to a node b in Θ_t and time t in Φ can be filled in parallel leading to dramatic improvement in the performance.

We also compare our algorithm against demand curtailment selection techniques such as those developed in [76] and [34]. We observed that these techniques typically incur errors (constraint violations) of around 5-20% and in the worst case can go as high as 95%. We exclude the details of this analysis as comparison against the optimal solutions obtained by solving the ILP makes this analysis redundant.



	Feb-May	July-October
No NLB	13.47	13.68
50%	26.94	27.36
10%	134.68	136.78
5%	269.36	273.56

Figure 5.6: Improvement in PV Penetration

5.6.3 Improvement in PV Penetration

We also analyze the improvement that can be achieved in the percentage of PV penetration using our minimum cost supply demand matching (MinCB) algorithm using a simple analysis. We consider a smart grid with a ratio of average load to minimum load

as 10 during the day light hours of 10 am to 5 pm. We analyze the solar irradiance time-series data obtained for the Los Angeles County [5] for the periods of February-May 2010 and July-October 2010 to calculate the normalized mean and standard deviation. We define *PV Penetration Percentage* as $\frac{PV_{avg} \times 100}{Load_{avg}}$, where PV_{avg} is the average supply from solar and $Load_{avg}$ is the average load. To calculate *PV Penetration Percentage*, we assume that the maximum supply i.e. average supply plus one standard deviation, minus any curtailment should be less than the minimum load. Moreover, we assume that the maximum curtailment that can be performed is $1 - \epsilon$, where ϵ is the accuracy parameter chosen. For example, if $\epsilon = 0.5$ i.e. 50% then maximum curtailment that can be performed is 50% of the maximum supply. This is because if the curtailment target is the entire maximum supply, the algorithm can guarantee only 50% curtailment for $\epsilon = 0.5$. As shown in Figure 5.6, in the absence of supply demand matching (net load balancing) algorithm (labeled No NLB), the maximum PV penetration is around 13%. This value dramatically increases with the use of supply demand matching framework and can be as high as $\approx 270\%$.

The analysis above suggests that by enabling PV curtailment with high accuracy, we can achieve the following objectives:

- We can ensure that the maximum supply in any time step, assumed to be average supply plus one standard deviation minus supply curtailment, does not exceed the minimum possible load value.

- A high PV penetration value can be achieved, thereby, reducing the amount of supply from external markets during peak demand periods. For example, a PV penetration value of 270% implies that at any given time step, if the PV is supplying its average value and even if the demand is 2.7 times the average load value, the PV supply will be enough to meet the demand and no external supply from the market will be required.

5.6.4 Minimum Cost Supply Demand Matching with Fairness

In Section 5.6.4, we introduced the notion of fairness by defining curtailment budget ranges for nodes. Here we evaluate the empirical performance of our minimum cost supply demand matching with fairness (MinCBF) algorithm. Note that the budget ranges for each node can be set appropriately by user/grid operator. In our experiments, we set the budget B_b for each node b to $\gamma^b_{max} / \sum_{b=1}^{B} \gamma^b_{max} \Gamma$ where γ^b_{max} denotes the sum of the maximum curtailment values across all the intervals. We also set $\alpha_b = \alpha$ and vary the value α.

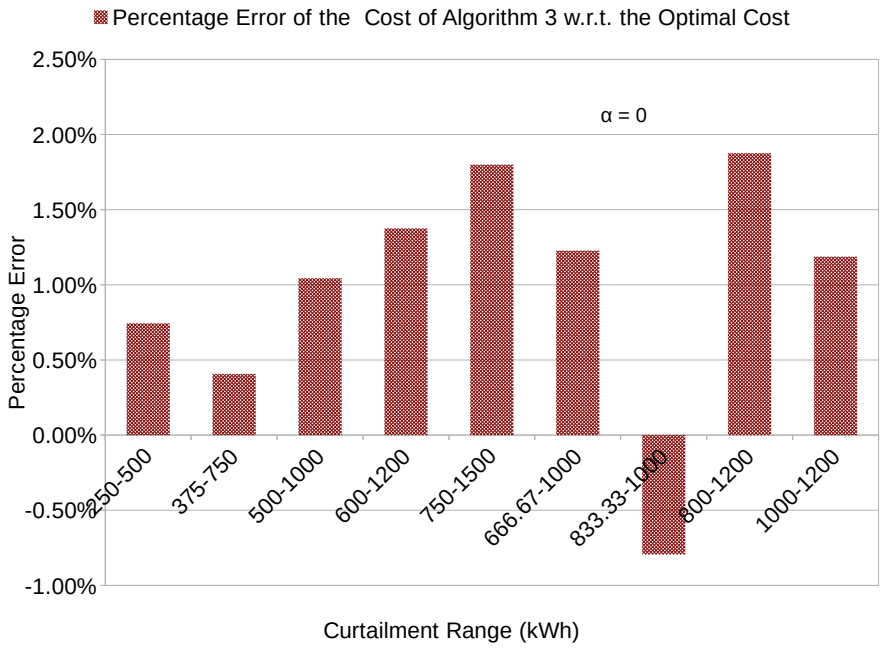


Figure 5.7: Percentage Error of the Cost of MinCBF w.r.t. the Optimal Cost

In order to evaluate the accuracy of our algorithm, we compare against the optimal solution obtained by solving the ILP defined in Section 5.2.1. Figure 5.7 shows the percentage error of the cost of the solution produced by MinCBF as compared against the optimal cost. Although, the worst case theoretical guarantee is a factor of 4 for quadratic cost function as provided by Theorem 5.3, in practice the algorithm performs

much better with errors varying from -0.79% to 1.88%. The negative percentage error implies that the cost of the solution from our algorithm was less than the cost of the optimal solution. Note that this is possible because our solution violates certain constraints which the optimal solution does not.

However, the complexity of the ILP prevented CPLEX studio to generate results in a reasonable amount of time. Hence, we compared the objective value of the rounded results obtained from our algorithm against the objective value of the LP with fractional solutions. Since, an LP relaxation of an ILP always produces better objective value (albeit with infeasible integral solutions), the results shown will only improve if compared against the optimal ILP solution.

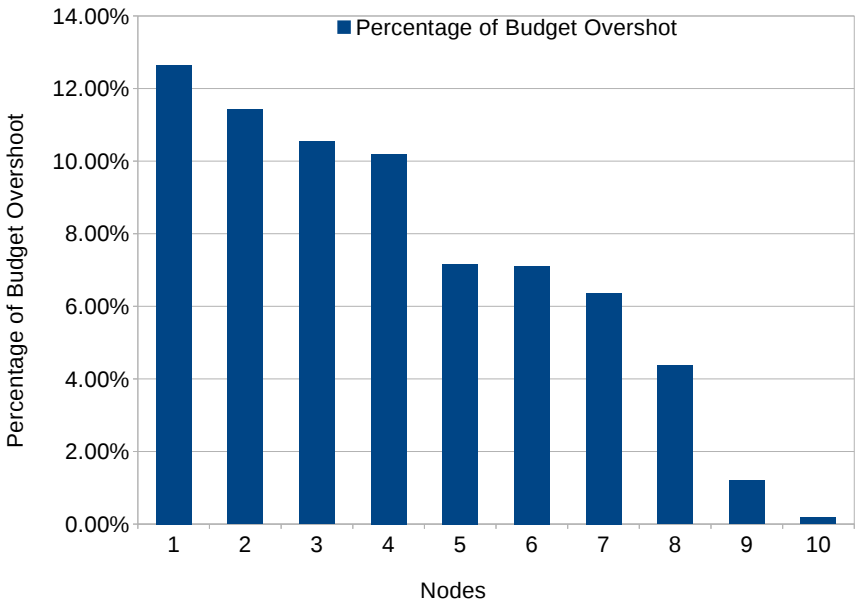


Figure 5.8: Percentage of Budget Overshoot for 10 Worst Affected Nodes

Figure 5.8 shows the percentage of the budget overshot for the top 10 worst affected nodes across all the values of L and U. The theoretical guarantee of factor 2 (100%)

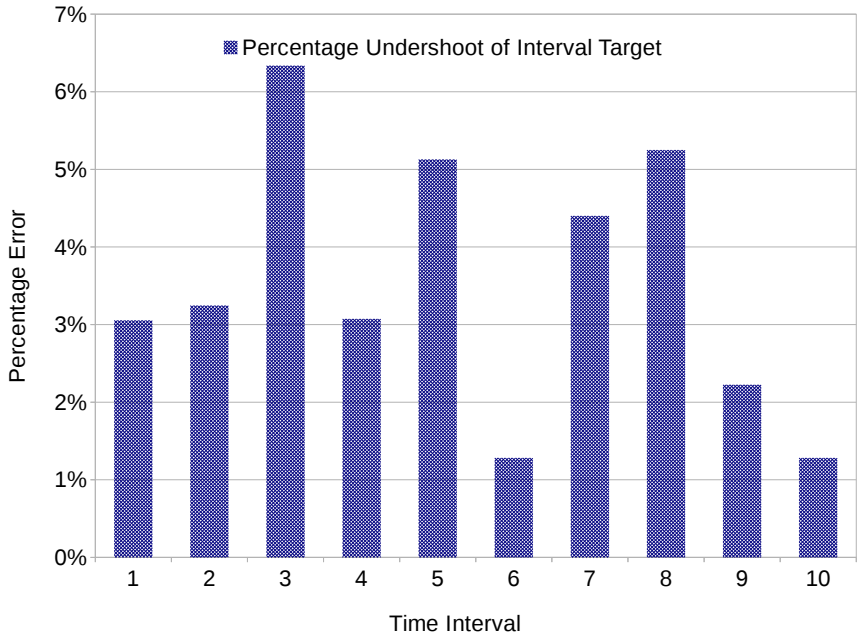


Figure 5.9: Percentage Undershoot of Interval Curtailment Target for various intervals (note: the labels of x axis do not denote the actual interval number)

provided by Theorem 5.2 is honored in all the cases. Moreover, in practice the error is less than 13% (1.13 factor) for the worst performing node.

Figure 5.9 shows the percentage of the interval curtailment target Γ_t undershot for the top 10 worst affected intervals across all the values of L and U. The theoretical guarantee of factor 2 (100%) provided by Theorem 5.2 is honored in all the cases. As we can see from the figure, in practice the error is less than 7% (1.07 factor) for the worst performing interval. This implies that the curtailment target for the worst performing interval could not be met and was deficit by 7%.

We also calculated the gini coefficient – which is the most commonly used measure of inequality in economics – of the curtailment achieved by each node as a proportion of its budget to measure the fairness of curtailment. Figure 5.10 shows the results for various values of α for a curtailment range of 500-1000 with 20 nodes. The value of

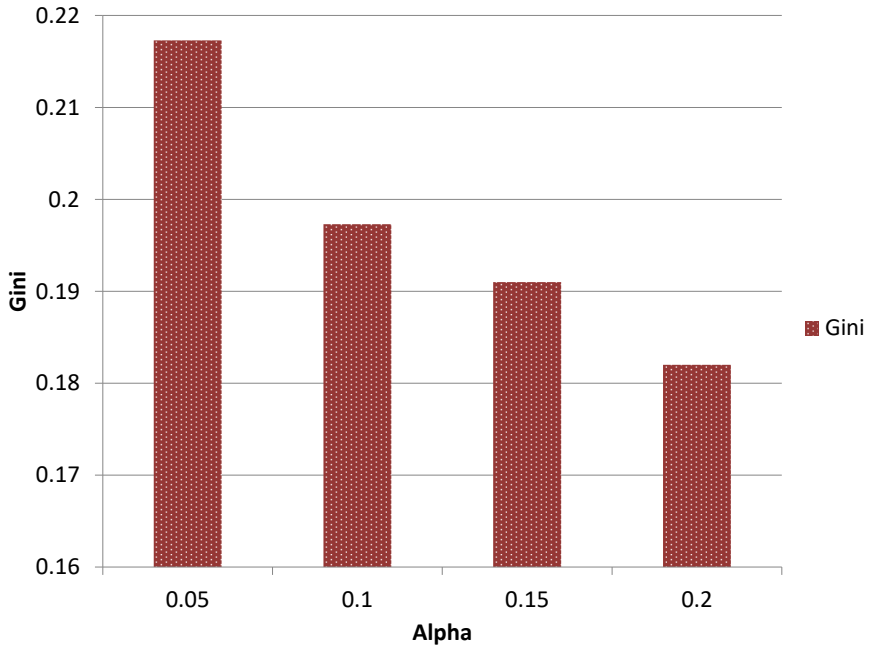


Figure 5.10: Gini Coefficient for various values of Alpha (α)

gini decreases with increasing α as the dispersion of the curtailment decreases. Above $\alpha = 0.2$, no feasible solution could be found.

Mathematically,

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n \sum_{i=1}^{n} x_i} \tag{5.28}$$

5.6.5 Online Algorithm for Minimum Cost Supply Demand Matching with Fairness

In order to evaluate our online algorithm for minimum cost supply demand matching with fairness (Online MinCBF), we compare the cost of the solutions obtained against the optimal solutions of the ILP defined in Section 5.2.1 as generated in Section 5.6.4. For various values of L and U pairs, we calculate the percentage error in the cost

obtained by the online algorithm (Algorithm 8) with respect to the optimal solutions. The budget values B_b input to Algorithm 8 are same as the ones used in Section 5.6.4.

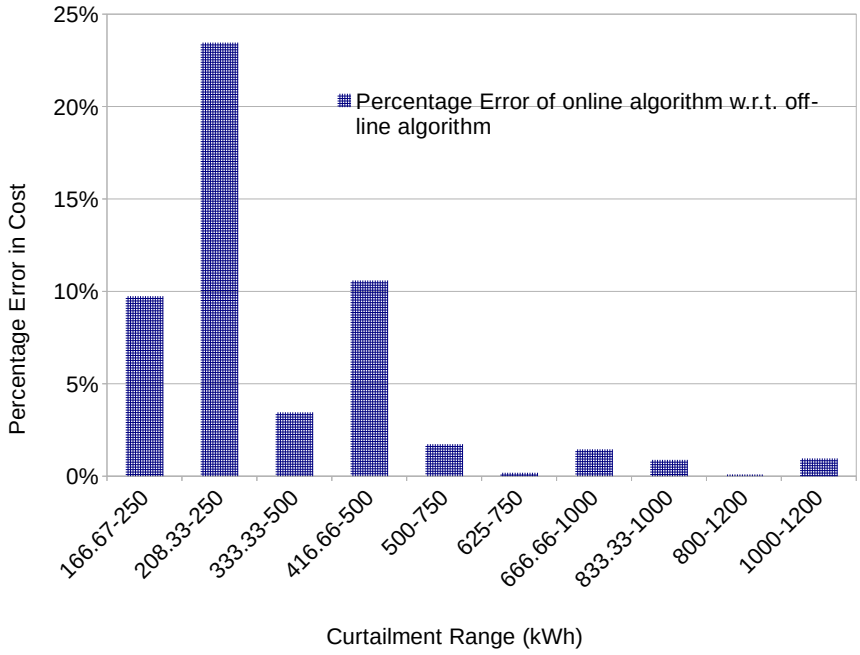


Figure 5.11: Percentage Error of the Cost of Online MinCBF w.r.t the Optimal Cost

Figure 5.11 shows the results obtained. Even though we do not provide any guarantee on the worst case bounds for the online algorithm, in practice the error incurred is low. The highest error incurred is around 23% (factor 1.23). This makes the online algorithm a good candidate for supply demand matching when the predictions for the entire horizon are not known in advance.

5.6.6 Supply Demand Matching with Network Constraints

We evaluate Algorithms 9 and 10 by considering a smart grid with 10 transformers each connected to 10 nodes. We consider a single time interval in which 5 of the transformers have a load surplus and 5 have solar surplus. For each transformer with solar surplus, we assume that the aggregate load connected is 0.25 times its total solar generation

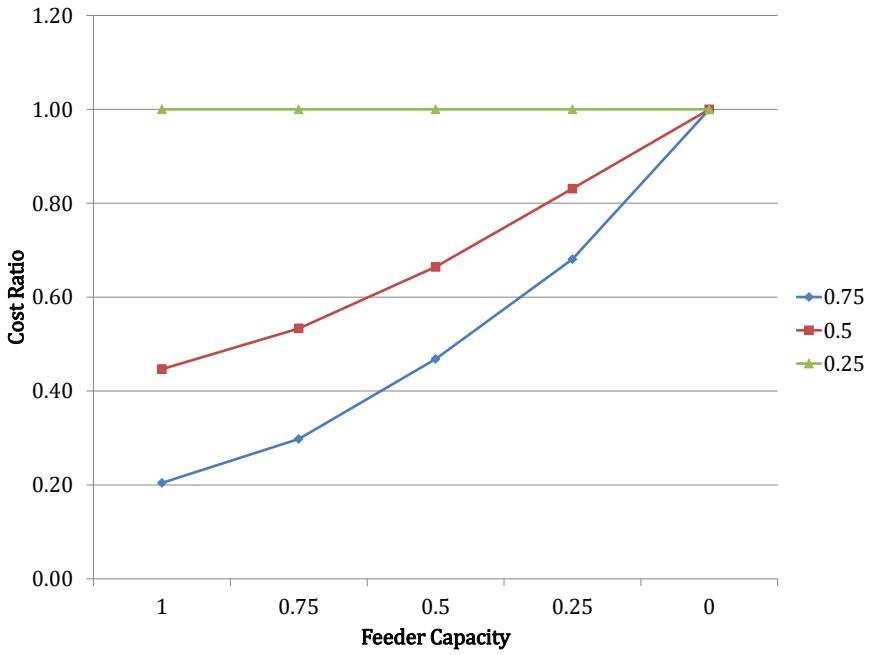


Figure 5.12: Ratio of Cost of Algorithm 10 to Algorithm 9 versus Feeder Capacity for various values of Distributor capacity.

S_{tx}. We set the capacity of distributor for each transformer Cap_{tx} equal to $f_1 S_{tx}$, where $f_1 \in [0, 1]$ and vary f_1. We set the capacity of the corresponding feeder Cap_f equal to $f_2 \times f_1 S_{tx}$, where $f_2 \in [0, 1]$ and vary f_2. Figure 5.12 shows the ratio of the cost of solution obtained by Algorithm 10 to that of Algorithm 9 versus the feeder capacity, which is determined by varying f_2. Each line in the figure corresponds to a single distributor capacity as determined by f_1.

As evident from the figure, for a fixed distributor capacity, a reduction in the feeder capacity leads to a reduction in the cost ratio i.e. cost of Algorithm 10 and Algorithm 9 converge with reduction in the feeder capacity until they are equal for a feeder capacity of 0. Moreover, for a fixed feeder capacity, the gap between the costs reduces with a reduction in the distributor capacity. This is because reduced distributor capacity results in lower solar surplus and hence less surplus redistribution. Note that when the distributor capacity is $f_1 = 0.25$, the costs are equal. This is because whatever solar power

is being produced in a distributor, it is being consumed by the load (which is equal to 0.25 times the solar generation). Hence, the cost incurred by both the algorithms is for curtailing loads in the load surplus transformers.

5.6.7 Storage Modeling

We also evaluate the percentage increase in runtime due to incorporation of storage nodes into our algorithm. We fix the curtailment range as (667,1000) kWh and the number of total nodes as 20. We assume an infinitely sized storage with charge/discharge capacity of 1000 kWh and charge/discharge efficiency of 1 in each time interval. We then vary the number of storage nodes for various values of percentage errors as denoted by ϵ and calculate the percentage increase in runtime compared against a setup in which no storage nodes are used. Figure 5.13 shows the detailed results. Even for 15 storage nodes and $\epsilon = 0.02$ i.e. 2%, the increase in runtime is around 160% i.e. the increased runtime is less than 2.6 times the runtime of the algorithm without any storage nodes.

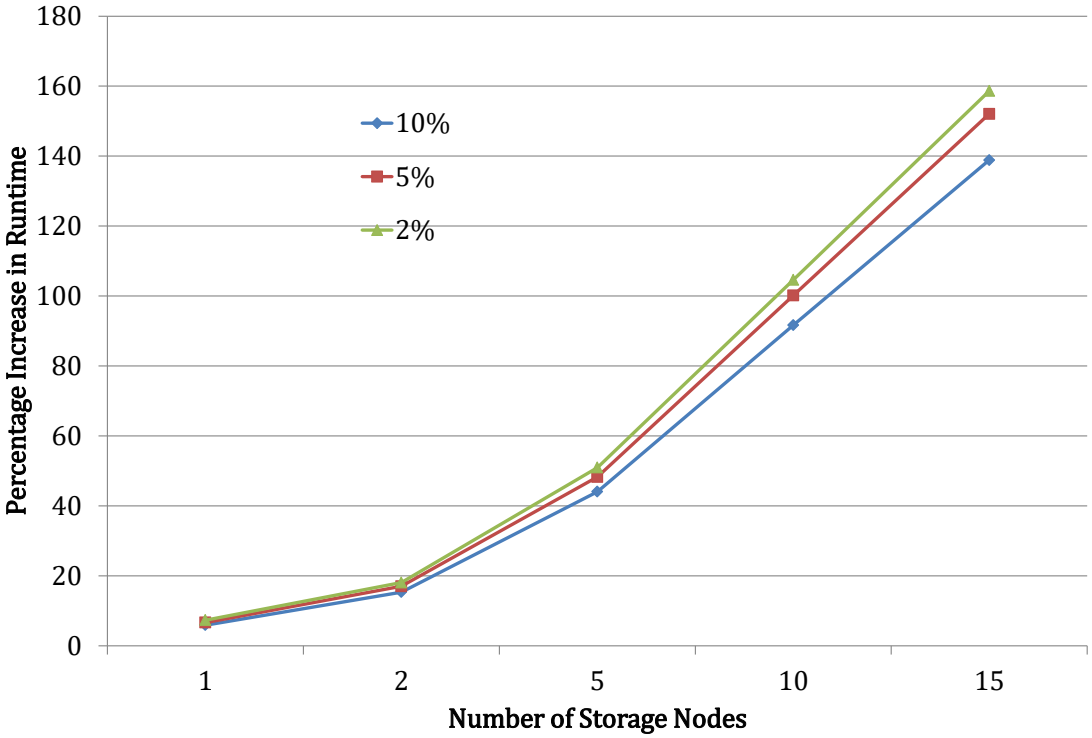


Figure 5.13: Percentage Increase in Runtime For various number of storage nodes for varying epsilon.

Chapter 6

Discrete Supply Demand Matching Under Prediction Uncertainty

Previous chapters discussed algorithms for performing discrete supply demand matching assuming no errors in prediction data of the operational values of each nodes. However, in practice predictions are uncertain which can lead to errors in the solutions. In this chapter, we explicitly model prediction uncertainty and develop approximation algorithm to perform discrete supply demand matching accounting for the uncertainty.

6.1 Two Stage Stochastic Recourse Model for Discrete Supply Demand Matching

We consider a model similar to two stage stochastic recourse model [55] to perform discrete supply demand matching while minimizing the expected uncertainty due to errors in the prediction models. We first define a few notations to supplement our smart grid model and then delve into the problem formulation and the solution using an approximation algorithm .

6.1.1 Smart Grid Model

As before, our smart grid consists of M_s supply nodes and M_d demand nodes each of them with N strategies. The horizon of planning consists of T intervals. In each interval

$t \in [T]$ where, for notational simplicity, $[a]$ denotes the set $\{1, 2, \ldots, a\}$, the supply node i_s (demand node i_d) can be in exactly one operational state j with a predicted discrete supply (load) value of $\gamma^s_{i_sj}(t)(\gamma^d_{i_dj}(t))$. In this chapter, we consider optimization in a single interval and hence we drop (t) to simplify notations. Moreover, $\gamma^s_{i_sj}(\gamma^d_{i_dj})$ denotes the actual generation or consumption for state j instead of the curtailment value as done in the previous chapters. The corresponding utility of the operational state is $u^s_{i_sj}(u^d_{i_dj})$. We assume that D is the total uncontrollable load in the system. This the minimum load which needs to be matched and hence we assign no utility to it. We use the 0-1 matrix $X^s(X^d)$ to represent the decision variables in which $x^s_{i_sj}(x^d_{i_dj})$ is set to 1 if node $i_s(i_d)$ is in operational state j and 0 otherwise.

6.1.2 Supply Demand Matching without Prediction Uncertainty

The grid operator needs to ensure that in any given interval, the total supply in the grid is equal to the total demand while maximizing the utility. This problem can be formulated using the following Integer Program (IP):

$$\max_{X^s, X^d} U(X^s, X^d) = \sum_{i_s=1}^{M_s} \sum_{j \in [N]} u^s_{i_sj} x^s_{i_sj} + \sum_{i_d=1}^{M_d} \sum_{j \in \mathcal{O}_{i_d}} u^d_{i_dj} x^d_{i_dj} \tag{6.1}$$

$$s.t. \sum_{i_s=1}^{M_s} \sum_{j \in \mathcal{O}_{i_s}} \gamma^s_{i_sj} x^s_{i_sj} = \sum_{i_d=1}^{M_d} \sum_{j \in \mathcal{O}_{i_d}} \gamma^d_{i_dj} x^d_{i_dj} + D \tag{6.2}$$

$$\sum_{j \in [N]} x^s_{i_sj} = 1 \ \ \forall i_s \tag{6.3}$$

$$\sum_{j \in [N]} x^d_{i_dj} = 1 \ \ \forall i_d \tag{6.4}$$

In the above formulation, Equation 6.2 ensures that supply and demand are matched. Equations 6.3 and 6.4 ensure that a node is in exactly one operational state. We assume that the IP is always feasible.

6.1.3 Incorporating Prediction Uncertainty in Supply Demand Matching

To model the prediction uncertainty, each discrete supply (load) value is associated with a random variable $\delta^s_{i_sj}(\delta^d_{i_dj})$, which denotes the error in prediction i.e. the actual real-time supply (load) value is $\gamma^s_{i_sj} + \delta^s_{i_sj}(\gamma^d_{i_dj} + \delta^d_{i_dj})$.

The incorporation of the random parameters into our problem coverts the deterministic problem of Section 6.1.2 into a stochastic one. We employ two state recourse stochastic programming model to make decisions under uncertainty [55]. In a two stage recourse model, a first stage decision is made and a (higher cost) recourse action is taken in second stage after the realizations of the random variables based on the first state decisions. Hence, the first stage decision needs to be made with two objectives: 1) Maximize the utility of implementing first stage decisions, and 2) minimize the cost associated with any recourse action taken in the second stage for all possible realizations of the random variables.

We define a scenario as a possible realization of all the random variables. Let $\mathcal{S}$ denote the set of all possible realizations. Let $K = |\mathcal{S}|$ denote the number of scenarios. For a random variable $\delta^s_{i_sj}(\delta^d_{i_dj})$, let $\delta^{si}_{i_sj}(\delta^{di}_{i_dj})$ denote the realized value under scenario $i \in \mathcal{S}$. Let p^i be the probability of the realization of scenario i. For a first stage decision X^{*s}, X^{*d} obtained by solving the optimization problem (6.1), the realization of a scenario i can lead to violation of the constraint 6.2 i.e. a imbalance between supply and demand can occur. The aggregate net load imbalance (load minus supply) is given using:

$$v^i(X^{*s}, X^{*d}) = \sum_{i_d=1}^{M_d} \sum_{j \in \mathcal{O}_{i_d}} \delta_{i_d j}^{di} x_{i_d j}^{*d} - \sum_{i_s=1}^{M_s} \sum_{j \in \mathcal{O}_{i_s}} \delta_{i_g j}^{gi} x_{i_g j}^{*g} \tag{6.5}$$

Hence, in the second stage $v^i(X^{*s}, X^{*d})$ amount of supply needs to injected into the grid (or load increased if negative). We assume this is achieved using battery storage. We assume the cost of the recourse action is given using $q(v^i(X^{*s}, X^{*d}))$ or simply $q(v^i)$.

We can formulate the two stage recourse model for supply demand matching as the following bi-criteria optimization:

2-Stage Recourse (2-SR):

$$\max_{X^s, X^d} \{U(X^s, X^d)\}, \min_{X^s, X^d} \{\sum_{i=1}^{K} p^i q(|v^i|)\} \tag{6.6}$$

$$s.t. \;\; v^i = \sum_{i_d=1}^{M_d} \sum_{j \in [N]} \delta_{i_d j}^{di} x_{i_d j}^{d} - \sum_{i_s=1}^{M_s} \sum_{j \in [N]} \delta_{i_s j}^{si} x_{i_s j}^{s} \;\; \forall i = 1, \ldots, K \tag{6.7}$$

$$(6.2) - (6.4)$$

The objective of the two stage recourse model is to maximize the utility of the decisions taken in stage one and minimize the expected value of the cost incurred due to recourse action.

There are two significant challenges in efficiency solving *2-SR*: 1) The decision variables X_s, X_d are discrete binary variables. As this problem is an NP-hard problem, no algorithm can exist which has a polynomial complexity runtime in the input parameters and which provides the optimal solution unless P == NP, and 2) the value K can be very large making the algorithm definition itself vary large. The algorithms developed in the next section address both the challenges. The first challenge is addressed by developing

an approximation algorithm which in polynomial time outputs a solution whose value, even in the worst case scenario, is "close" to the optimal solution. The second challenge is addressed by making certain assumptions on the randomness of the scenarios.

6.2 Approximation Algorithms for Supply Demand Matching Under Prediction Uncertainty

We consider two models for the realization of the random variables and develop approximation algorithms for the same.

6.2.1 Latent State Model

This model assumes the existence of a set of latent states $\mathcal{H}$. Each latent state $h_i \in \mathcal{H}$ is realized with a probability p_i and is responsible for the realization of scenario $i \in \mathcal{S}$. In other words, all the random load and generation variables are observations corresponding to these latent states.

Now, the number of latent states can be infinite. However, using the expectation value $E[\delta^s_{i_sj}](E[\delta^d_{i_dj}])$ for each random variable $\delta^s_{i_sj}(\delta^d_{i_dj})$, we can reformulate the 2-SR optimization problem. Assuming a linear cost function q, Equation 6.6 can be written as:

$$\max_{X^s,X^d}\{U(X^s,X^d)\},\ \min_{X^s,X^d}\{\sum_{i=1}^{K} p^i(|\sum_{i_d=1}^{M_d}\sum_{j\in[N]} q(\delta^{di}_{i_dj})x^d_{i_dj} - \sum_{i_s=1}^{N_s}\sum_{j\in[N]} q(\delta^{si}_{i_sj})x^s_{i_sj}|)\} \quad (6.8)$$

switching the summation order and moving the probability p inside the cost function we get:

$$\max_{X^s,X^d}\{U(X^g,X^d)\},\ \min_{X^s,X^d}\{|\sum_{i_d=1}^{N_d}\sum_{j\in[N]}q(\sum_{i=1}^{K}p^i\delta_{i_dj}^{di})x_{i_dj}^d-\sum_{i_s=1}^{N_s}\sum_{j\in[N]}q(\sum_{i=1}^{K}p^i\delta_{i_sj}^{si})x_{i_sj}^s|\} \tag{6.9}$$

Replacing the terms $\sum p^i\delta^i$ with the expected value and ignoring K, we have the following formulation:

2-Stage Recourse for Latent state model (2-SRL):

$$\max_{X^s,X^d}\{U(X^s,X^d)\},\ \min_{X^s,X^d}\{Q(X^s,X^d)\} \tag{6.10}$$
$$s.t.(6.2)-(6.4)$$

where $Q(X^s,X^d)=|\sum_{i_d=1}^{M_d}\sum_{j\in[N]}q(E[\delta_{i_dj}^d])x_{i_dj}^d-\sum_{i_s=1}^{M_g}\sum_{j\in[N]}q(E[\delta_{i_sj}^s])x_{i_sj}^s|$. To simplify notations, the expected cost for node i_s (i_d) with operational state j: $q(E[\delta_{i_sj}^s])$ ($q(E[\delta_{i_dj}^d])$) will be denoted using $\tilde{q}_{i_sj}^s$ ($\tilde{q}_{i_dj}^d$).

6.2.2 Approximation Algorithm for 2-SRL

In order to develop an approximation algorithm for 2-SRL, *2-SRL-Approx*, we will create two dynamic programs, one for the generation nodes and one for the demand nodes. For each possible generation (demand) value, utility value and uncertainty value triplet, the dynamic program will return true or false denoting whether it is possible to achieve the triplet using all the generation (demand) nodes or not. Using the approximation technique of rounding, we will ensure that the size and the runtime complexity of the dynamic program (and hence the algorithm) is polynomial in the input size and $\frac{1}{\epsilon}$, where

ϵ is the accuracy parameter. However, this will lead to a loss in accuracy which we will prove that in the worst case is bounded by ϵ.

We will discuss the dynamic program for the demand nodes. The dynamic program for the generation nodes can be developed analogously. Let t be number such that $\tilde{q'}^{d}_{i_d j} = \tilde{q}^{d}_{i_d j} + t > 0 \ \forall i_d, j$. We refer to $\tilde{q'}$ as translated uncertainty values. As this translation translates the objective function by a constant amount, it does not affect the optimization problem. Let Γ_{max} a bound on the maximum possible demand or supply value, u_{max} be a bound on the maximum possible utility value and $\tilde{q'}_{max}$ be a bound on the maximum possible translated uncertainty value that can be achieved using all the demand nodes. Also, let $\Gamma_{min}, u_{min}, \tilde{q'}_{min}$ be the minimum possible values greater than zero. We assume the quantities $\frac{\Gamma_{max}}{\Gamma_{min}}$, $\frac{u_{max}}{u_{min}}$ and $\frac{\tilde{q'}_{max}}{\tilde{q'}_{min}}$ to be bounded by polynomial in the input size and $\frac{1}{\epsilon}$. Let $M = \max\{M_s, M_d\}$ We create the following partitions:

$$[0], [\Gamma_{min}, \Gamma_{min}(1+\epsilon)^{\frac{1}{M}}), [\Gamma_{min}(1+\epsilon)^{\frac{1}{M}}, \Gamma_{min}(1+\epsilon)^{\frac{2}{M}}), \ldots, [\Gamma_{min}(1+\epsilon)^{\frac{k_d-1}{M}}, \Gamma_{min}(1+\epsilon)^{\frac{k_d}{M}})$$

$$[0], [u_{min}, u_{min}(1+\epsilon)^{\frac{1}{M}}), [u_{min}(1+\epsilon)^{\frac{1}{M}}, u_{min}(1+\epsilon)^{\frac{2}{M}}), \ldots, [u_{min}(1+\epsilon)^{\frac{k_d-1}{M}}, u_{min}(1+\epsilon)^{\frac{k_u}{M}})$$

$$[0], [\tilde{q'}_{min}, \tilde{q'}_{min}(1+\epsilon)^{\frac{1}{M}}), [\tilde{q'}_{min}(1+\epsilon)^{\frac{1}{M}}, \tilde{q'}_{min}(1+\epsilon)^{\frac{2}{M}}), \ldots, [\tilde{q'}_{min}(1+\epsilon)^{\frac{k_d-1}{M}}, \tilde{q'}_{min}(1+\epsilon)^{\frac{k_q}{M}})$$

where $k_d = \lceil M \log_{1+\epsilon}(\frac{\Gamma_{max}}{\Gamma_{min}}) \rceil$, $k_u = \lceil M \log_{1+\epsilon}(\frac{u_{max}}{u_{min}}) \rceil$ and $k_q = \lceil M \log_{1+\epsilon}(\frac{\tilde{q'}_{max}}{\tilde{q'}_{min}}) \rceil$. The intervals in the partitions are indexed using $\widehat{\gamma} \in \{0, \Gamma_{min}, \ldots, \Gamma_{min}(1+\epsilon)^{\frac{k_d-1}{M}}\}$, $\widehat{u} \in \{0, u_{min}, \ldots, u_{min}(1+\epsilon)^{\frac{k_u-1}{M}}\}$ and $0, \tilde{q'}_{min}, \ldots, \tilde{q'}_{min}(1+\epsilon)^{\frac{k_q-1}{M}}$ respectively, i.e. the lowest value of the intervals. Without loss of generality, we assume that the demand nodes are ordered as $1, 2, \ldots, M_d$. We now define a dynamic programming function $\Phi_d(\widehat{\gamma}, \widehat{u}, \widehat{q'}, i_d)$ which returns true (1) if there exists a triplet γ, u, q' such that each element γ, u and q' of the triplet belongs to

the intervals indexed by $\widehat{\gamma}$, $\widehat{u}$ and $\widehat{q'}$, respectively and the triplet values can be achieved using demand nodes $1, \ldots, i_d$. The function can be defined recursively as follows:

$$\Phi_d(\widehat{\gamma}, \widehat{u}, \widehat{q'}, i_d) = \begin{cases} 1 & \text{if } i_d = 1 \text{ and } \exists j \in \mathcal{O}_{i_d} \mid \gamma_{i_d j} \in \widehat{\gamma}, u_{i_d j} \in \widehat{u}, \tilde{q}_{i_d j} \in \widehat{q'} \\ 1 & \text{if } \exists \widehat{\gamma^-}, \widehat{u^-}, \widehat{q'^-}, j \in \mathcal{O}_{i_d} \mid \Phi_d(\widehat{\gamma^-}, \widehat{u^-}, \widehat{q'^-}, i_d - 1) = 1 \text{ and} \\ & \gamma_{i_d j} + \widehat{\gamma^-} \in \widehat{\gamma},\ u_{i_d j} + \widehat{u^-} \in \widehat{u},\ \tilde{q}_{i_d j} + \widehat{q'^-} \in \widehat{q'} \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

Now, all the 1 entries in $\Phi_d(\widehat{\gamma}, \widehat{u}, \widehat{q'}, M_d)$ denote the feasible $\widehat{\gamma}, \widehat{u}, \widehat{q'}$ triplet for demand nodes. We can obtain similar triplet for the generation nodes. To distinguish them, we use the notations $\widehat{\gamma}_d, \widehat{u}_d, \widehat{q'}_d$ and $\widehat{\gamma}_g, \widehat{u}_g, \widehat{q'}_g$ for demand and generation nodes respectively. We define a function $\Theta(\widehat{\gamma})$ which outputs a list of the utility and the associated uncertainty for each value of $\widehat{\gamma}$. Θ can be populated using Algorithm 12 . Θ is used to determine the dominating solutions using Algorithm 13. A simple dynamic programming table traversal algorithm can then be used to determine the operational states to be followed by each node for any given dominating solution from the pareto frontier.

We have the following theorem for 2-SRL-Approx.

Theorem 12. *2-SRL-Approx is an algorithm with runtime complexity polynomial in the input size and $\frac{1}{\epsilon}$, where ϵ is an accuracy parameter, that produces a pareto frontier for discrete supply demand matching with uncertainty such that for each pareto solution, the utility is within $(1 - \epsilon)$ of the corresponding solution, and the maximum supply demand mismatch is within ϵ factor of both actual supply and demand. Additionally, the uncertainty associated with either supply or demand are within $(1 + \epsilon)$ of the optimal supply or demand uncertainty.*

Algorithm 12: Populating Θ using Φ_d and Φ_s

Input: $\Phi_d, \Phi_s, \widehat{\gamma}$
foreach $\widehat{\gamma}$ **do**
  $L_d \leftarrow$ all $\widehat{u}_d, \widehat{q}'_d$ s.t. $\Phi_d(\widehat{\gamma}, \widehat{u}_d, \widehat{q}'_d, M_d) = 1$
  $L_s \leftarrow$ all $\widehat{u}_s, \widehat{q}'_s$ s.t. $\Phi_s(\widehat{\gamma}, \widehat{u}_s, \widehat{q}'_s, M_s) = 1$
  $L_{\widehat{\gamma}} \leftarrow \phi$
  foreach $\widehat{u}_d, \widehat{q}'_d \in L_d$ **do**
    foreach $\widehat{u}_s, \widehat{q}'_s \in L_s$ **do**
      $L_{\widehat{\gamma}} \leftarrow L_{\widehat{\gamma}} \cup (\widehat{u}_d + \widehat{u}_s, |\widehat{q}'_d - \widehat{q}'_s|)$
    end
  end
  $\Theta(\widehat{\gamma}) \leftarrow L_{\widehat{\gamma}}$
end
Output: Θ

Proof. Correctness: We will prove the correctness of the algorithm in two steps. In step 1, we will prove that the dynamic program runs correctly to fill Φ (Φ_d and Φ_s) and step 2, we will prove that if the dynamic program runs on actual values instead of the partition indices and an actual value can be achieved, then this information is not missed in dynamic program run on partition indices.

We will prove step 1 using induction. Essentially, we need to prove that $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', i) = 1$ iff it is possible to achieve $\widehat{\gamma}, \widehat{u}, \widehat{q}'$ using first i nodes. For $i = 1$, it is true trivially, as we will set $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', 1) = 1$ iff there exists a j in $[N]$ such that $\gamma_{i_dj} \in \widehat{\gamma}, u_{i_dj} \in \widehat{u}, \tilde{q}_{i_dj} \in \widehat{q'}$. Let the inductive hypothesis be true for $i = k$. For $k+1$, $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', k+1) = 1$ if and only if $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', k) = 1$ and $\gamma_{ij} + \widehat{\gamma}^- \in \widehat{\gamma},\ u_{ij} + \widehat{u}^- \in \widehat{u},\ \tilde{q}_{ij} + \widehat{q}'^- \in \widehat{q}'$. So, if $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', k+1) = 1$, we can achieve $\widehat{\gamma}, \widehat{u}, \widehat{q}'$ by achieving $\widehat{\gamma}^-, \widehat{u}^-, \widehat{q}'$ using k nodes and $\gamma_{ij}, u_{ij}, \tilde{q}_{ij}$ by following j^{th} operational state of $(k+1)^{st}$ node. If no such j exists or $\Phi(\widehat{\gamma}, \widehat{u}, \widehat{q}', k) = 0$, it cannot be achieved.

To proves step 2, we show that if using k nodes, an actual value of $\gamma_a = \sum_{i=1}^{k} \gamma_i$ (proof is analogous for u and $\tilde{q}$) can be achieved, then $\widehat{\gamma} \leq \sum_{i=1}^{k} \gamma_i < (1+\epsilon)^{\frac{k}{M}} \widehat{\gamma}$

Algorithm 13: Algorithm to generate dominating solutions using Θ

Input: Θ

$L \leftarrow$ all $\widehat{u}, \widehat{q}'$ s.t. $\Theta(\widehat{\gamma}) \neq \phi \ \ \forall \widehat{\gamma}$

Sort L in descending order using $\widehat{u}$ as key, if multiple entries have $\widehat{u}$, keep the one with lowest $\widehat{q}'$ and remove all other entries

Let L_0 be the entry with largest $\widehat{u}$ and L_K be the entry with smallest $\widehat{u}$

$D \leftarrow L_0$

for $i = 1$ *to* K **do**

  $\widehat{u}, \widehat{q}' \leftarrow L_i$

  foreach $\widehat{u}_l, \widehat{q}'_l \leftarrow l \in D$ **do**

    if $\widehat{q}' \geq \widehat{q}'_l$ **then**

      break from for each loop

    end

  end

  $D \leftarrow D \cup L_i$

end

Output: D: set of dominating solutions

and some entries $\Phi(\widehat{\gamma} : (1+\epsilon)^{\frac{k}{M}}\widehat{\gamma}, :, :, k)$ will be set to 1. We again use induction. For $k = 1$, if $\gamma \in \widehat{\gamma}$ can be achieved then some entries of $\Phi(\widehat{\gamma}, :, :, 1)$ are true and $\widehat{\gamma} \leq \gamma < (1+\epsilon)^{\frac{1}{M}}\widehat{\gamma}$. Let the inductive hypothesis be true for k' i.e. if a value of $\gamma'_k = \sum_{i=1}^{k'} \gamma_i$ can be achieved by node i exhibiting an operational value of γ_i, then some entries of $\Phi(\widehat{\gamma}'_k : (1+\epsilon)^{\frac{k'}{M}}\widehat{\gamma}'_k, :, :, k')$ are true and $\widehat{\gamma}'_k \leq \sum_{i=1}^{k'} \gamma_i < (1+\epsilon)^{\frac{k'}{M}}\widehat{\gamma}'_k$. For $k'+1$, as per the definition of the dynamic program, the entry where $\widehat{\gamma}'_k + \gamma_{k'+1}$ falls will be true. So, we just need to show that $\widehat{\gamma}_{k'+1} \leq \gamma_{k'+1} = \sum_{i=1}^{k'+1} \gamma_i < (1+\epsilon)^{\frac{k'}{M}}\widehat{\gamma}_{k'+1}$.

We know the following: (1) $\widehat{\gamma}_{k'+1} \leq \widehat{\gamma}_k + \gamma_{k'+1} < (1+\epsilon)^{\frac{1}{M}}\widehat{\gamma}_{k'+1}$, and (2) $\sum_{i=1}^{k'} \gamma_i < (1+\epsilon)^{\frac{k'}{M}}\widehat{\gamma}'_k$. So, $\sum_{i=1}^{k'+1} \gamma_i < (1+\epsilon)^{\frac{k'}{M}}\widehat{\gamma}'_k + \gamma_{k'+1} < (1+\epsilon)^{\frac{k'}{M}}(\widehat{\gamma}'_k + \gamma_{k'+1}) < (1+\epsilon)^{\frac{k'}{M}}(1+\epsilon)^{\frac{1}{M}}\widehat{\gamma}_{k'+1} < (1+\epsilon)^{\frac{k'+1}{M}}\widehat{\gamma}_{k'+1}$ as $(1+\epsilon)^{\frac{k'}{M}} > 1$, when $\epsilon, k' > 0$.

Runtime: The dynamic programming table has $k \times k_q \times k_u \times M$ entries. Each entry requires $k \times k_q \times k_u \times N$ time to fill. Hence, the total runtime is $k^2 \times k_q^2 \times k_u^2 \times M \times N$. Since, $k_d = \lceil M \log_{1+\epsilon}(\frac{\Gamma_{max}}{\Gamma_{min}}) \rceil$, $k_u = \lceil M \log_{1+\epsilon}(\frac{u_{max}}{u_{min}}) \rceil$ and $k_q = \lceil M \log_{1+\epsilon}(\frac{\tilde{q}'_{max}}{\tilde{q}'_{min}}) \rceil$, the runtime is polynomial in the input size and $\frac{1}{\epsilon}$.

Approximation Guarantee: Consider a dominating solution $\widehat{\gamma}, \widehat{u}, \widehat{q}'$. Let u be actual reward value that can be achieved and belongs to the partition $\widehat{u}$. It is evident from the correctness proof above that such a u will always exist. Now, as shown in step 2 of the correctness proof, $\widehat{u} \leq u < (1+\epsilon)^{\frac{k}{M}}\widehat{u}$ for k nodes. So, for $M_s, M_d \leq M$ nodes this implies, $\widehat{u} > \frac{1}{1+\epsilon}u = \frac{1-\epsilon}{1-\epsilon^2}u$. Hence, $\widehat{u} \geq (1-\epsilon)u$. Similarly, for uncertainty we can prove that $\widehat{q}' \leq (1+\epsilon)q'$.

The above proof establishes that Φ_d and Φ_s are correct, can be filled in polynomial time and the solution values are within $(1+\epsilon)$ of the optimal. Φ_d and Φ_s are used to fill Θ. Since utilities are a simple summation and uncertainty is simple absolute different, correctness of Θ follows. Algorithm 12 has polynomial time complexity. Now, $\widehat{u}_d \geq (1-\epsilon)u_d$ and $\widehat{u}_s \geq (1-\epsilon)u_s$ implies $\widehat{u}_d + \widehat{u}_s \geq (1-\epsilon)(u_d + u_s)$. As the final uncertainty is obtained using subtraction, we cannot provide a guarantee for the same.

Now, consider the supply and demand partition indices $\widehat{\gamma}_s = \widehat{\gamma}_d = \widehat{\gamma}$. Let γ_s and γ_d be the actual generation and demand values. $\widehat{\gamma} \leq \gamma_s, \gamma_d < (1+\epsilon)\widehat{\gamma}$. So, $\gamma_s - \gamma_d < (1+\epsilon)\widehat{\gamma} - \widehat{\gamma} \leq \epsilon\widehat{\gamma} \leq \epsilon\gamma_s, \epsilon\gamma_s$. Hence, the mismatch in generation and demand is within ϵ factor of both actual generation and consumption.

□

Discussion on translation of uncertainty and upper bounds Γ_{max}, u_{max} and $\tilde{q'}_{max}$

The uncertainty value $\tilde{q}_{ij}$ associated with a node i and operational state j can have a negative value. Thus it cannot be directly used for indexing the dynamic programming table Φ. Hence, we add a value $t = |\min\{0, \min_{ij}\{\tilde{q}_{ij}\}\}|$ to $\tilde{q}_{ij}$ for each i, j to produce $\tilde{q'}_{ij}$ which are non-negative. The following relationship holds: $\sum_{i=1}^{M}\sum_{j\in[N]}\tilde{q'}_{ij} = \sum_{i=1}^{M}\sum_{j\in[N]}\tilde{q}_{ij} + Mt$, where M_d is the number of generation nodes M_s or demand nodes M_d.

The values Γ_{max}, u_{max} and $\tilde{q'}_{max}$ denote the maximum possible value of supply (or demand) γ, utility u and translated uncertainty $\tilde{q'}$ respectively. These values have an impact on the number of entries in the dynamic programming table Φ. In this work, we set the values as follows: $\Gamma_{max} = \sum_{i=1}^{M} \max_{j \in [N]} \gamma_{ij}$, $u_{max} = \sum_{i=1}^{M} \max_{j \in [N]} u_{ij}$ and $\tilde{q'}_{max} = \sum_{i=1}^{M} \max_{j \in [N]} \tilde{q'}_{ij}$ i.e. we take the sum of maximum possible value that each node can exhibit.

We further make the following assumptions: $\Gamma_{max} = poly(M, \frac{1}{\epsilon}, \Gamma_{min})$, $u_{max} = poly(M, \frac{1}{\epsilon}, u_{min})$ and $\tilde{q'}_{max} = poly(M, \frac{1}{\epsilon}, \tilde{q'}_{min})$. This assumption implies the the minimum possible non zero value exhibited by any node is not "too small" compared to the maximum achievable value from all the nodes. For the nodes in a smart grid, this is reasonable assumption to make.

6.2.3 Black Box Model

We now relax the assumption that all the random load and generation variables are observations corresponding to a set of latent states. The implication here is that each random variable potentially has its own state governing its realization and hence the number of possible scenarios in the system explodes combinatorially. In this case, it is not even possible to define the problem in polynomial time (due to exponential number of constraints of the form (6.7)).

Therefore, we consider a black box model to study such a problem. The analysis here is a minor modification of the analysis performed in [21]. Under this model, we are given a black box, which can be used to sample scenarios, where each scenario corresponds to a particular realization of all the random variables. Sampling of a single scenario is assumed to be an $O(1)$ time complexity operation. We show that by sampling only polynomial number of scenarios, we can obtain a solution which is within $1 + O(\epsilon)$ of the optimal solution. This technique is also known as Sample Average Approximation

(SAA) [59]. The techniques simply draws K number of scenarios and assumes that each scenario occurs with a probability of $\frac{1}{K}$ to populate the expected values for defining the problem.

We first define some notations and assumptions. Let the optimal 2-SR formulation, i.e. the 2-SR formulation which considers the exponential number of scenarios be defined as:

$$\max\{U(x)\}, \min\{E[v(x)]\}$$
$$s.t. x \in X \tag{6.12}$$

where X denotes the feasible range of decision variables X^s, X^d, $U(x)$ denotes the utility for a feasible solution x and $E[v(x)]$ denotes the expected value of the uncertainty for x. Now, let the 2-SR model under blackbox model i.e. the formulation obtained by sampling K scenarios, where K is a polynomial in the input size be defined as:

$$\max\{\widehat{U}(x)\}, \min\{\frac{1}{K}\sum_{i=1}^{K} v^i(x)\}$$
$$s.t. x \in X \tag{6.13}$$

We make the following assumptions:

1. $X^d = \mathbf{0}$ i.e. load is uncontrollable.

2. The feasible range of $x \in X$ is such that the predicted supply meets the demand in the first stage i.e. $\sum_{i_s=1}^{M_s} \sum_{j \in \mathcal{O}_{i_s}} \gamma_{i_s j}^s x_{i_s j}^s = D$

3. Let Z^* be the minimum value of $\{E[v(x)]\} \forall x \in X$. Let x^* be the minimizer. We assume: $0 \leq v^i(x) \leq \frac{\lambda Z^*}{\epsilon} \forall x, i$, where $\lambda, \epsilon > 0$. This assumption implies

that the minimum (non zero) amount of storage required in second stage under any first stage decision with any possible scenario realization is not "too small" compared to the maximum amount of storage required in the second stage. Note that $0 \leq v^i(x)$ can be achieved by translating all the uncertainty values by a fixed amount. Also note that $Z^* > 0$ as it is a value corresponding to expectation value of non-negative terms.

Now, we want to study the error introduced due the use of sample mean instead the expected value: $|U(x) - \widehat{U}(x)|$ and $|E[v(x)] - \frac{1}{K}\sum_{i=1}^{K} v^i(x)|$ to compare the solutions (6.12) and (6.13). As $U(x)$ is independent of the realized scenario, $U(x) = \widehat{U}(x)$. So, we just focus on $|E[v(x)] - \frac{1}{K}\sum_{i=1}^{K} v^i(x)| \forall x \in X$.

For $x \in X$, we define $Y_i = \frac{\epsilon v^i(x)}{\lambda Z^*} \forall i$. It follows from assumption (3) that $Y_i \in [0, 1]$. For notational simplicity we drop x and write the terms as $E[v]$ and $\frac{1}{K}\sum_{i=1}^{K} v^i$.

6.2.4 Approximation Algorithm for 2-SRB

2-SRB-Approx: In order to develop an approximation algorithm for 2-SRB, we will sample $K = \Theta(\lambda^2 \epsilon^{-4} \log |X| \log \frac{1}{\delta})$ scenarios and calculate the expected values $\frac{1}{K}\sum_{i=1}^{K} \delta$ for all the random variables. We will then use *2-SRL-Approx* to generate the dominating solutions.

Now, as per Chernoff's Bound [21], Given independent random variables $Y_1, \ldots, Y_K$ with $Y_i \in [0, 1]\ \forall i$, and $Y = \sum_{i=1}^{K} Y_i$ for every $\epsilon > 0$:

$$Pr\{|Y - E[Y]| > \epsilon K\} \leq 2e^{-\epsilon^2 K} \tag{6.14}$$

Now, note that $E[Y] = E[\sum_{i=1}^{K} Y_i] = \frac{\epsilon K}{\lambda Z^*} E[\frac{1}{K}\sum_{i=1}^{K} v^i] = \frac{\epsilon K}{\lambda Z^*} E[v]$. So, $Pr\{|E[v] - \frac{1}{K}\sum_{i=1}^{K} v^i| > \epsilon Z^*\}$

$$= Pr\{|\frac{\lambda Z*}{\epsilon K} E[Y] - \frac{\lambda Z^*}{\epsilon K} \sum_{i=1}^{K} Y_i| > \epsilon Z^*\} \tag{6.15}$$

$$= Pr\{|E[\sum_{i=1}^{K} Y_i] - \sum_{i=1}^{K} Y_i| > \frac{\epsilon^2 K}{\lambda}\} \tag{6.16}$$

By equation 6.14 and substituting the value of $K = \Theta(\lambda^2 \epsilon^{-4} \log |X| \log \frac{1}{\delta})$, this probability is less than $\frac{\delta}{|X|}$. Taking union bound over all $x \in X$, with probability $1 - \delta$, $\forall x \in X$:

$$|E[v(x)] - \frac{1}{K} \sum_{i=1}^{K} v^i(x)| \leq \epsilon Z^* \tag{6.17}$$

Now, let $E'[v(x)] = \frac{1}{K}\sum_{i=1}^{K} v^i(x)$. Let x' be the minimizer of $E'[v(x)]$. Note that x' is not an exact minimizer and the approximation algorithm 2-SRL is used to obtain x'. Using Equation 6.17 with x^*, we get $E'[v(x^*)] - Z^* \leq \epsilon Z^*$. Similarly, with x', we get $E[v(x')] \leq \epsilon Z^* + E'[v(x')]$. This implies, $E[v(x')] \leq \epsilon Z^* + (1+\epsilon)E'[v(x^*)] \leq \epsilon Z^* + (1+\epsilon)^2 Z^*$. So $E[v(x')] \leq (1+\epsilon)(2+\epsilon)Z^*$. As each sampling is assumed to be an $O(1)$ operation, we have the following theorem

Theorem 13. *2-SRB-Approx, with probability $(1-\delta)$, gives a $(1+\epsilon)(2+\epsilon)$-factor bound on uncertainty for 2-SRB with runtime polynomial in λ, $\frac{1}{\delta}$, $\frac{1}{\epsilon}$ and the input size.*

6.2.5 Practical Implementation

2-SRL-Approx and *2-SRB-Approx* algorithms, while providing both worst case error and polynomial runtime guarantee might not scale very well for large problem sizes due to the large dimensionality of the dynamic program table of the order $k \times k_q \times k_u \times M$.

In this section, we discuss a few techniques to improve the runtime performance of the algorithm. Note that the filling of the dynamic program table is highly parallelizable, as for a node, each entry can be filled independently just using the values of the previous node. However, the large size of the table can quickly fill up the available memory on the system. This can be addressed using the following techniques:

1. When the entries corresponding to node i is being filled, only the entries corresponding to node $i-1$ are needed. Entries for any prior node can be deleted thereby reducing the size of the table by a factor of $\frac{M}{2}$.

2. The algorithm can be used in single mode similar to algorithms developed in Section 5.1. This will allow us to use the following memory optimizations:

   (a) The dimension corresponding to uncertainty can be removed. Instead, each entry of the table $\Phi(\widehat{\gamma}, \widehat{u}, i)$can now store the minimum possible uncertainty value that can be achieved by nodes $1, \ldots, i$ at utility $\widehat{u}$ with operational value $\widehat{\gamma}$. The subtraction of demand and supply uncertainties to produce the final uncertainty value does not allow this optimization in dual mode.

   (b) The dimension corresponding to operational values can be removed if we assume that the constraints governing the aggregate operational value from all the nodes are of one of the following two forms (but not both): (i) the aggregate operational value is greater than a threshold, (ii) the aggregate operational value is below some threshold.

More sophisticated techniques for runtime performance improvement is beyond the scope of this work as our objective is to prove the existence of polynomial runtime complexity approximation algorithm for discrete supply demand matching with prediction uncertainty and not necessarily the development of a runtime optimized framework.

6.3 Risk Aware Supply Demand Matching in Smart Grids with High DER Penetration

We now consider an alternative approach to incorporate prediction uncertainty into decision making. In this approach, a storage schedule is developed to minimize the expected values of the cost of the decisions over a horizon. Moreover, the loads and local generation using DERs such as PVs are assumed to be uncontrollable and storage is assumed to be the only controllable DER. Moreover, this approach explicitly considers the presence of an external electricity market which was not considered in the previous models.

6.3.1 Smart Grid Model for Risk Aware Supply Demand Matching

The smart grid that we consider in this approach is defined as a micro grid that best represents a University or Industrial campus. It is characterized by the presence of several load consuming nodes such as buildings, and several producers of non-commercial scale (cheap) electricity. In this work, we assume that the producers supply electricity to the micro grid using PV installations over the buildings. The micro grid consists of distributed storage with an aggregate capacity R. The storage has a cumulative charge/discharge capacity of c. We do not model storage specific parameters such as Depth of Discharge (DoD), efficiency as we assume the presence of a number of distributed storage systems rather than a single battery system. A centralized controller is responsible for ensuring smooth grid operations over a decision making horizon defined over a set of daytime intervals $t \in \{1, \ldots, T\}$. Note that the horizon is defined over daytime since supply within the micro grid using PVs is only available during that time. We assume that during the night time, all demand needs to be met using the storage accumulated during the day. In our framework, the controller responds to per interval

supply demand mismatch by either procuring/selling electricity from the external market or by charging/discharging the storage. For each interval t, a buy action b_t needs to be performed, where b_t is the amount of electricity bought from the market (in kWh). Moreover, a discharge action sr_t is performed, where sr_t is the amount of the electricity discharged from the storage (in kWh). If b_t and sr_t are negative, opposite actions of selling the electricity or charging the storage, respectively are performed. The action performed at t is represented using $a_t =< b_t, sr_t >$. The cost of buying (selling) electricity at interval t is C_t^b. We do not associate any cost with storage charging/discharging. We also assume that the set of costs $C_t^b,\ 0 \leq t \leq T$ is known a priori.

6.3.2 Input Data Model

The inputs to our model are the load and generation predictions. As the predictions are prone to errors, they need to be modeled carefully so that the framework can make decision accounting for the errors. We assume that at the beginning of each interval t of the decision making horizon, the controller receives an input data vector $\bar{Y_t} =< D_t, D_{t+1}, \ldots, D_T >$. Here $D_{t'} = L_{t'} - P_{t'},\ t' \in \{t, \ldots, T\}$ denotes the per interval supply demand mismatch; $L_{t'}$ and $P_{t'}$ denote the per interval aggregate consumption and PV generation respectively.

At time t, D_t denotes the actual supply demand mismatch (net load imbalance) as observed in real time whereas the $D_{t'},\ t' \in \{t+1, \ldots, T\}$ are random variables denoting the predicted imbalances. The relation between action a_t and D_t is $D_t = b_t + sr_t$. Thus, if we want to buy excess storage for the future at time t, we can choose $b_t > D_t$.

6.3.3 Tail End Risk

At the end of the decision making horizon, we assume that the minimum amount of storage required for night time operations is R'. Hence, the tail end event whose risk

needs to be minimized is the unavailability of storage capacity of more than R' by the end of the decision making horizon. Depending upon the severity of the implications of failing to avoid the risk, the grid operator can associate it with a cost function as follows:

$$C(R_T, R') = \begin{cases} 0 & R_T \geq R' \\ C_{risk} & \text{otherwise} \end{cases} \tag{6.18}$$

where R_T denotes the storage available after the interval T and C_{risk} denotes the cost of not avoiding the risk. C_{risk} needs to be calculated appropriately to provide the operator with a choice to meet or not to meet R' for a given input data. In this work, we assume that the risk needs to be avoided at all cost and hence $C_{risk} = \infty$. We will extend our framework to handle partial risk tolerance in our future works.

6.3.4 Supply Demand Framework

A high level overview of our supply demand matching framework is shown in Figure 6.1. A storage capacity of R_0 is available at the beginning of the decision making horizon. In each time interval t, using the most updated input data $\bar{Y}_t$, the framework makes a decision and outputs action a_t for buying (selling) from the external market and discharging (charging) the storage. The action a_t for each interval is determined by formulating and solving a Markov Decision Process (MDP) [19] as described in the next section. MDP is a widely used mathematical framework for solving sequential decision making problems. At the end of the horizon, the amount of storage available R_T should be $\geq R'$.

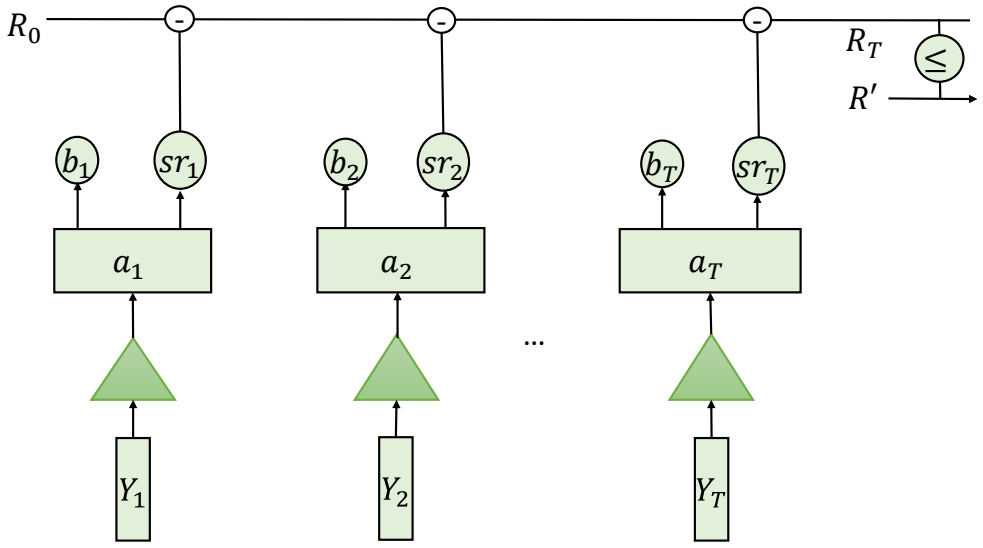


Figure 6.1: Supply Demand Matching Framework

6.3.5 Risk Aware Supply Demand Matching Framework

In this section, we discuss the details of the Risk Aware Supply Demand Matching Framework that we developed to perform minimum cost per interval supply demand matching using storage and electricity market while ensuring that the tail end risk of the shortfall of storage is avoided.

Objective

Given the supply demand matching model defined in Section 6.3.1, the objective of our framework is to minimize the cost of grid operations while ensuring that in each interval, the power ingress is equal to the power egress and at the end of the horizon, the minimum storage requirement is met.

Solution

MDP Formulation For each interval $t \in \{1, \ldots, T\}$, with information $\bar{Y}_t$, we define the MDP to determine action a_t using the following parameters:

Decision Epoch The decision epochs – the time intervals during which the MDP makes decisions – are $\{t, \ldots, T\}$ with $T < \infty$.

States MDP uses the information regarding the current state to make a decision. For each interval $t' \in \{t, \ldots, T\}$, we define state $s_{t'} =< d_{t'}, R_{t'} >$, where $d_{t'} \backsim D_{t'}$ denotes the predicted net load imbalances. $R_{t'},\ 0 \leq R_{t'} \leq R$ denotes the available storage capacity and is defined as $R_{t'} = R_{t'-1} - sr_{t'}$.

Initial and Terminal States The initial state is denoted using a single state $s_{t-1} =< 0, R_{t-1} >$. The terminal states are denoted using $s_T =< 0, R_T >$ where $R_T \geq R$. Note that the net load imbalance for initial and terminal states are zero. This is because the initial state imbalance is addressed before invoking the current MDP and the terminal state is outside the decision epoch.

Actions MDP outputs an action from all available actions in the action space for each time interval after making a decision. The action ensures that the load imbalance for the interval is mitigated while making sure that the storage charging/discharging constraints are satisfied. For a state $s_{t'} =< d_{t'}, R_{t'} >$, action space contains actions $a_{t'} =< b_{t'}, sr_{t'} >, t' \in \{t, \ldots, T\}$ such that $d_{t'} = b_{t'} + sr_{t'}$, $\max\{R_{t'} - R, -c\} \leq sr_{t'} \leq \min\{-R_{t'}, c\}$ where c denotes the per interval charge/discharge capacity of the storage system. Only the action a'_t for $t' = t$ i.e., corresponding to the current interval is executed. The remaining actions are part of the computation framework while solving the objective function.

State Transition Probabilities Given state $s_{t'} =< d_{t'}, R_{t'} >$ at time t', the transition probability for $s_{t'+1} =< d_{t'+1}, R_{t'+1} >$ is as follows:

$$p(s_{t'+1}|s_{t'}, a_{t'}) = \begin{cases} 0 \text{ if } R_{t'+1} \quad \neq R_{t'} - sr_{t'} \\ P[d_{t'+1}|\bar{Y_t}] \text{ otherwise} \end{cases}$$

where $P[d_{t'+1}|\bar{Y_t}]$ denotes the probability distribution of $d_{t'+1} \backsim D_{t'+1}$ under input data $\bar{Y_t}$.

Objective The MDP solves the following objective problem:

$$\min E[\sum_{t'=t}^{T} C^b_{t'}.b_{t'} + C(R_T, R')] \tag{6.19}$$

subject to the conditions defined above.

MDP Solution

When input data $\bar{Y_t}$ is available at the beginning of interval t,the MDP is used to determine the action $a_t =< b_t, sr_t >$ which are executed immediately. If the storage capacity at the beginning of the interval was R_{t-1}, then the storage capacity at the end of the interval $R_t = R_{t-1} - sr_t$. The MDP is solved as detailed below.

Solving MDP We define the cost to go function at time $t' \in \{t, \ldots, T\}$ as follows:

$$J_{t'}(R_{t'}, d_{t'}) = C^b_{t'}.(d_{t'} - sr_{t'}) + E[C(R_T, R') + \sum_{i=t'+1}^{T} C^b_i \times (D_i - sr_i)|\bar{Y_t}] \tag{6.20}$$

If $J^*_{t'}$ is the optimal value of the function, then it can be written recursively as:

$$J^*_{t'}(R_{t'}, d_{t'}) = \inf_{sr \in \mathcal{SR}} \{C^b_{t'}.(d_{t'} - sr_{t'}) + \sum_{d \backsim D_{t'+1}} P[d|\bar{Y_t}] \times J^*_{t'+1}(R_{t'} - sr_{t'}, d)\} \tag{6.21}$$

where $\mathcal{SR} = \{sr : \max\{R_t - R, -c\} \leq sr_t \leq \min\{R_t, c\}\}$. The recurrence relation above implies that at time t' the best course of action is the one which minimizes the sum of the cost of operation in the current interval and the expected value of the future cost of operations. The above recurrence can be solved using standard dynamic programming techniques. The initial condition for the recurrence relation is as follows:

$$J^*_{T+1}(R_T, 0) = \begin{cases} 0 & \text{if } R_T \geq R' \\ \infty & \text{otherwise} \end{cases} \tag{6.22}$$

The optimal action at time t can be determined by solving $J_t(R_{t-1}, 0)$ after setting up the dynamic program for the MDP. The variables need to be discretized in order to solve the dynamic program.

Runtime Let $[a, b]_\delta$ denote the number of discrete entries when the range $[a, b]$ is discretized using δ units i.e. $[a, b]_\delta = \lceil \frac{b-a}{\delta} \rceil$. The total number of entries to

be filled are: $T \times [0, R]_\delta \times \max_t\{[d_t^{min}, d_t^{max}]_\delta\}$, where $d_t^{min/max}$ denote the minimum/maximum value attained by the random variable D_t. Each entry requires $O(|\mathcal{SR}_\delta| \times \max_t\{[d_t^{min}, d_t^{max}]_\delta\})$ time. Hence, the total time complexity is $O(T \times [0, R]_\delta \times |\mathcal{SR}_\delta| \times \max_t\{[d_t^{min}, d_t^{max}]_\delta\}^2)$.

6.4 Results and Analysis

6.4.1 2-SR Algorithms

In order to perform practical evaluations, we implemented the algorithms using python. The experiments were performed on Dell optiplex with 4-cores and 4 GB RAM.

Dataset

We obtained the load curtailment data from the demand response implementation on our University Campus. We also used the solar dataset as generated in Section 5.6.1. The costs of the strategies were evaluated using the function $f(\gamma) = 2\gamma^2$, where γ is the curtailment value of the strategy. The uncertainty values were evaluated using the function $f(\gamma) = 2\gamma$.

Scalability

In order to perform scalability analysis, we calculate the runtime of the algorithm by varying the number of nodes and the value of ϵ (denoted as Epsilon in the figures).

We first fixed the value of $\epsilon = 0.2$ and varied the number of nodes. As shown in Figure 6.2, the increase in runtime is polynomial (cubic) with an increase in the number of nodes.

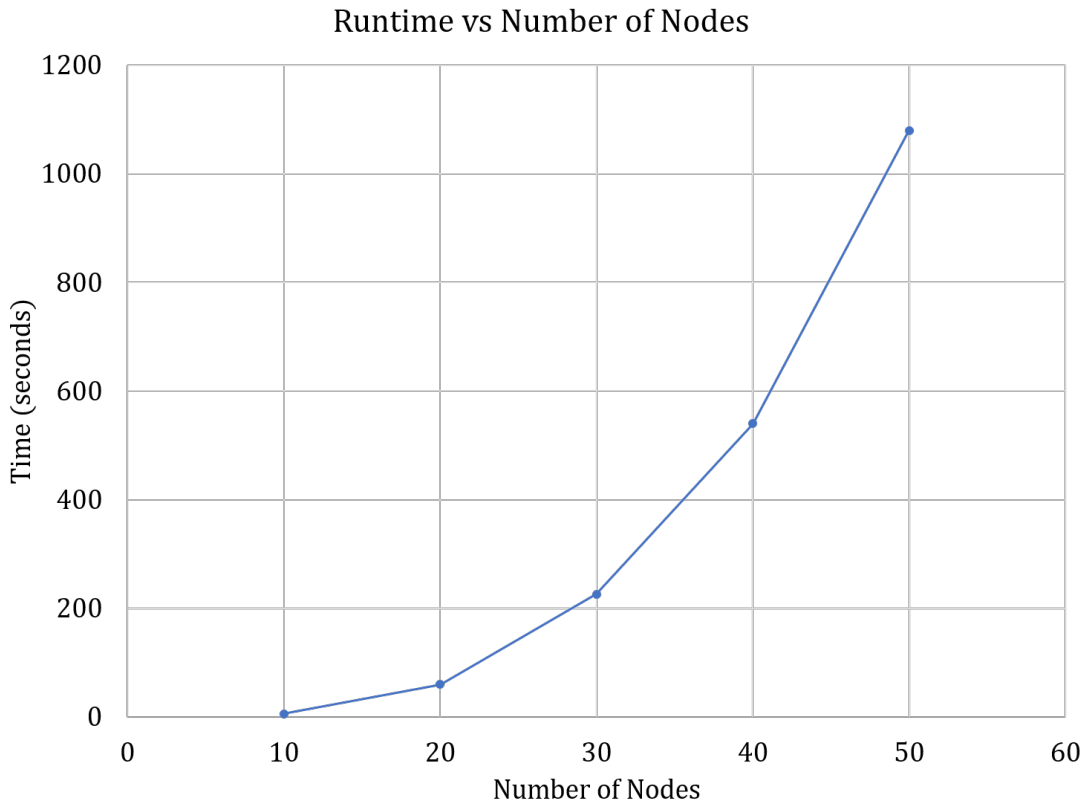


Figure 6.2: Runtime of 2-SRL-Approx for varying values of nodes for fixed $\epsilon = 0.2$

We then fix the number of nodes to 20 and vary ϵ. As shown in Figure 6.3, the runtime increases significantly with the decrease in ϵ. However, the increase is still polynomial in $\frac{1}{\epsilon}$.

The runtimes of the algorithms might seem large at the first look. However, note that these are calculated using a naive python implementation. For real world scenarios, an implementation in faster programming languages such as C/C++ can lead to significantly better runtimes. Moreover, several trivial parallelization techniques can be adopted to get drastic improvement in the runtime.

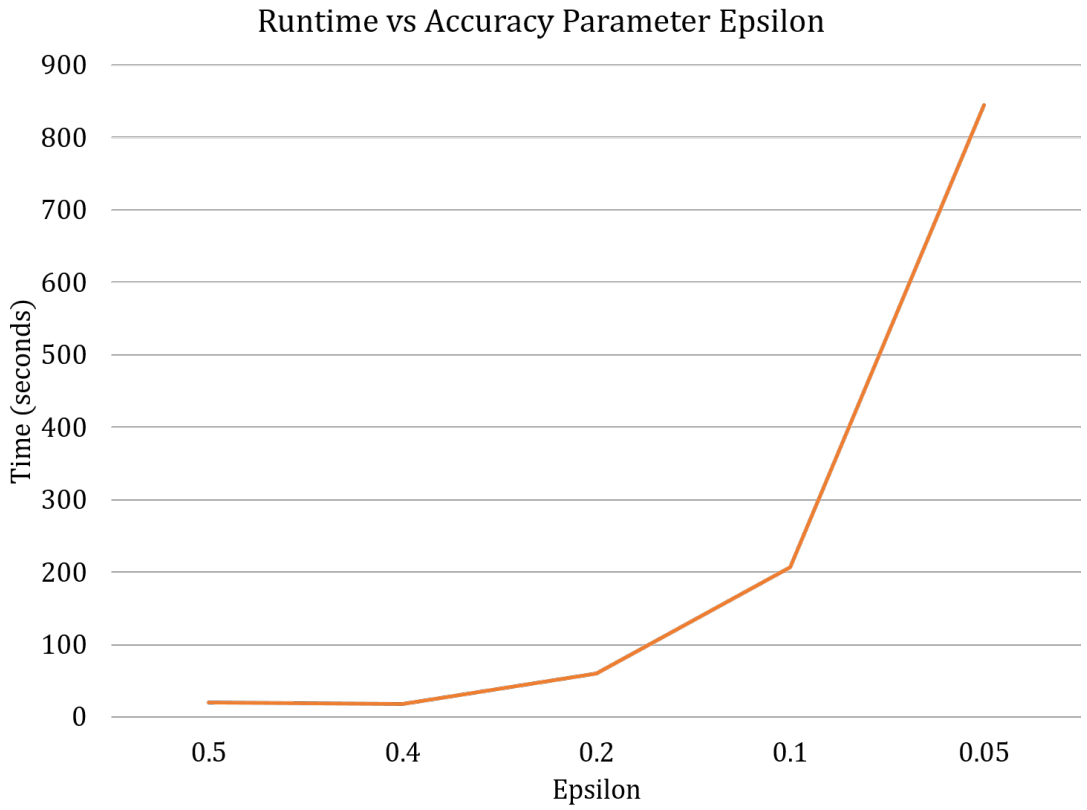


Figure 6.3: Runtime of 2-SRL-Approx for varying values of ϵ for fixed number of nodes = 20

6.4.2 Risk Aware Supply Demand Matching

We evaluated the framework developed in Section 6.3.5 to qualitatively assess its performance in terms of cost minimization and scalability. The framework was implemented using C++ on Dell Optiplex with 4 cores running at 2 Ghz clock frequency. The framework was run over a decision making horizon of 20 15-intervals (11 am - 4 pm).

Dataset

We used electricity consumption (load) time series data available from USC's microgrid [36]. We selected a subset of buildings such that the average load during peak periods (1-4 pm) was 90 kWh and during off peak periods was 70 kWh. We assumed

a storage capacity of 1500 kWh with a charge/discharge rate of 140 kW. The storage requirement at the end of the horizon was set as 1400 kWh, which is sufficient to supply off peak average load for 19 hours i.e. outside the decision making horizon. We used solar irradiance data available at [5] for Los Angeles USC Downtown area to calculate the solar generation data. We varied the area of solar panels and used PV output calculator [7] to generate three time series solar generation data which were then used to create three load imbalance time series. The three time series differed in the fraction of times either load or solar generation was surplus during the decision making horizon. Figure 6.4 shows the time series data of load and solar generation used to create the load imbalance data. The three solar time series are as follows: S1) Both load and solar surplus almost equal number of times, S2) load is surplus most of the time, and S3) solar generation is surplus most of the time. We assumed additive errors using gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 5, 10, 20$ i.e., $Pr[e_{t'}|\bar{Y}_t] \backsim \mathcal{N}(0, (t'-t)\times\sigma^2)$, such that $D_{t'} = \hat{d}_{t'} + e_{t'}$, with $\hat{d}_{t'}$ denoting the actual value of load imbalance.

Optimality

We compare the cost of operation across the entire decision making horizon, i.e. 11 am - 4pm, of our framework against an optimal framework which has the correct data for the entire horizon in advance. The optimal framework solves a single MDP for decision epochs $\{1, \ldots, T\}$ using the available correct data. The cost of buying was set at \$0.75/kWh for 11am - 1pm and \$0.95/kWh for the remaining horizon. We evaluated our framework using 6 timeseries data generated i.e. three load imbalance profiles with two different variance of 10 and 20. We set the initial value of storage to 0 and 750 kWh.

Figure 6.5 shows the relative percentage error of the cost of solutions obtained from our framework with respect to the optimal algorithm. As we can see from the figure,

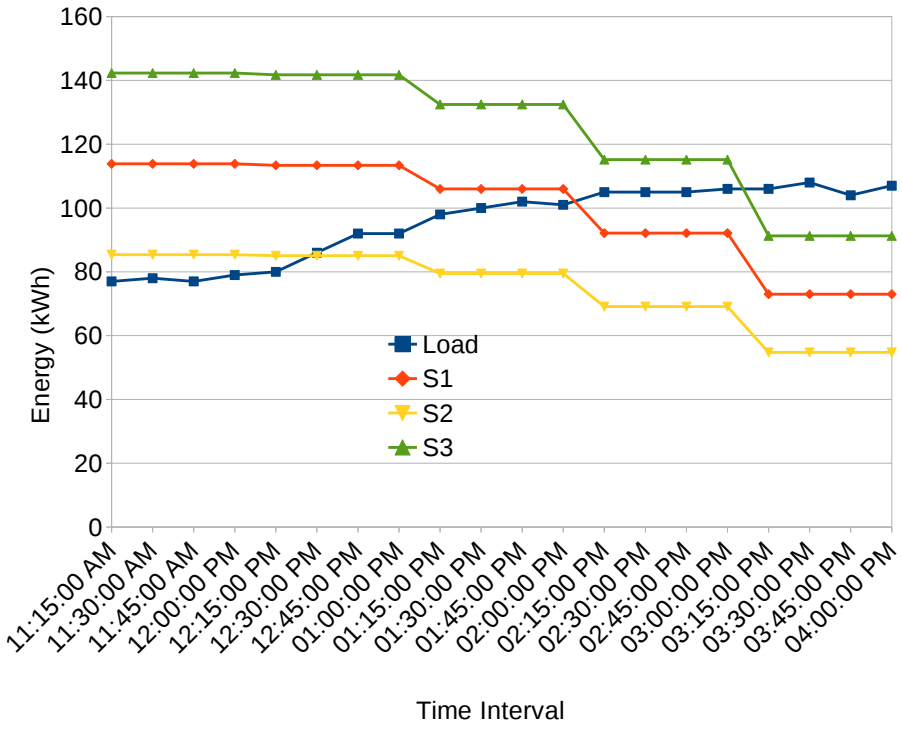


Figure 6.4: Load and Solar Timeseries Data Used for Evaluation

for each pair of initial storage-load imbalance curve, a higher variance leads to slightly higher errors. This is expected as increasing uncertainty in the input data increases the errors in decision making. However, note that the error difference is not significant implying that the framework is able to compensate for the increased uncertainty.

Also, note that the errors increase with the change in the initial storage value. This is due to the fact that when the initial storage is 0 kWh, the major focus of the framework is to just charge the storage to its final value (1400 kWh) and fulfill the deficit. However, when the initial storage is 750 kWh, charging requirements are reduced and the framework tries to focus on participating in the external market operations by selling the excess energy. The additional actions open up new avenues for the uncertainty in the input data to introduce errors as manifested in the higher relative percentage errors.

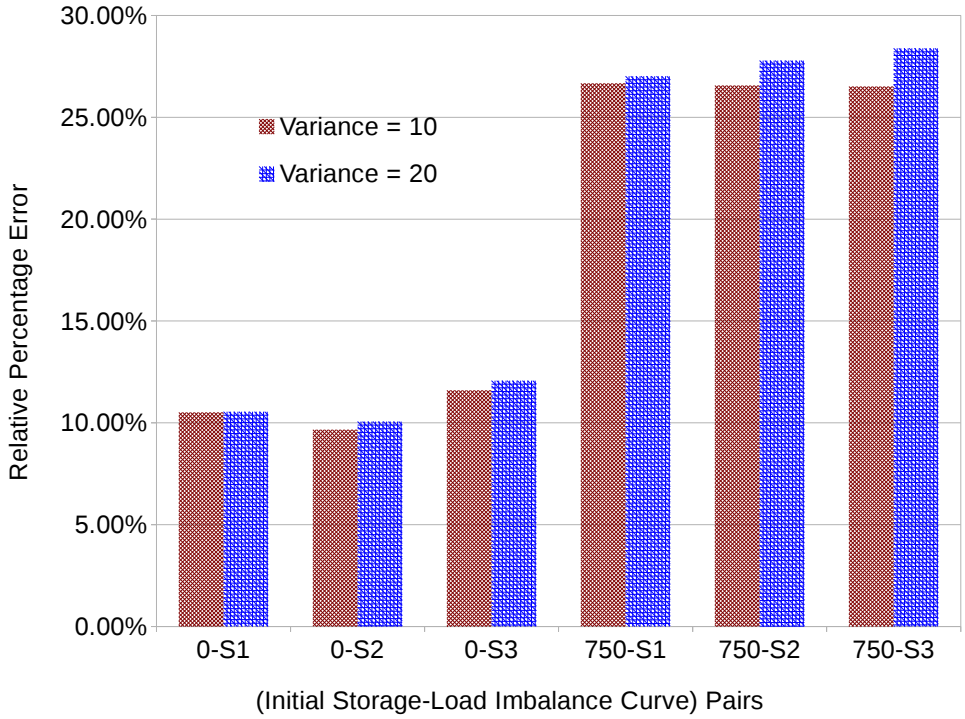


Figure 6.5: Relative Percentage Error of Framework w.r.t. Optimal Cost For Varying Initial Storage and Variance

In our future work, we will focus on reducing the errors furthers by using reinforcement learning based techniques which learn from the past experience to improve the accuracy for the future operations.

Scalability

We perform scalability analysis of the framework by varying the storage capacity and the variance of the errors. We calculate the time required to make decision at time period 0 for the entire horizon. For each storage capacity value R, we fix the charge discharge rate to $0.1 \times R$. Figure 6.6 shows the time required to make decisions with respect to the storage capacity R for various values of variance. The increase in runtime is quadratic in R. This is in agreement with the runtime which is $O(R \times |\mathcal{SR}|)$, when other variables are fixed and $|\mathcal{SR}| = O(R)$. The increase in runtime with variance is

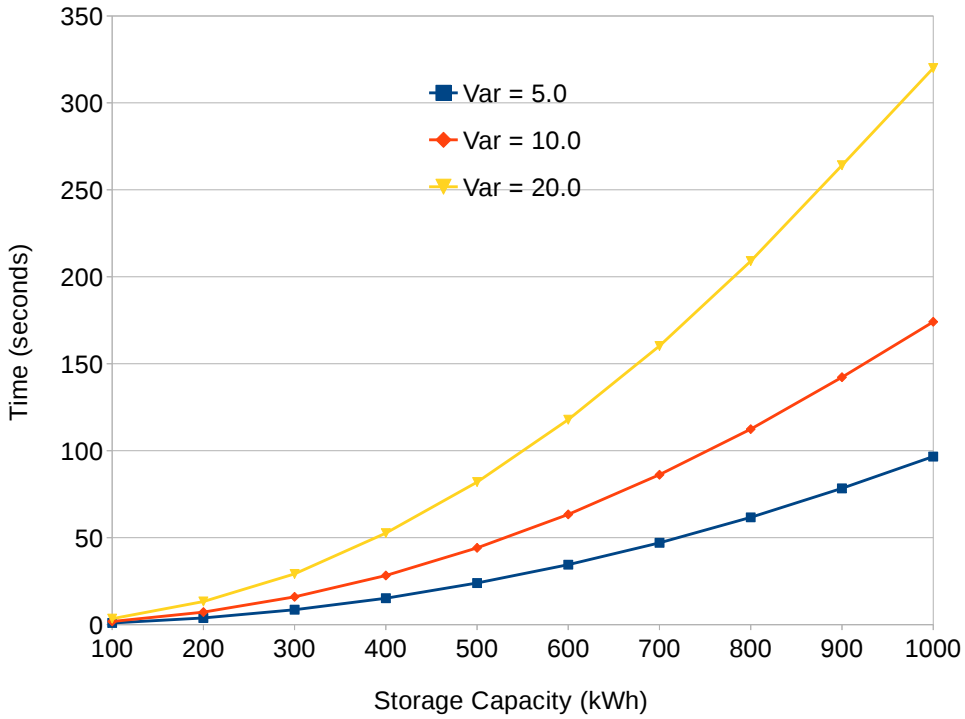


Figure 6.6: Time Required for Decision Making w.r.t Storage Capacity for Varying Variance

significant. However, even for a storage capacity of 1000 kWh and variance of 20, the runtime is just around 340 seconds.

A few techniques can be followed to further improve the scalability. The entries of the dynamic table have low interdependencies. Hence, they are easy to parallelize. Moreover, as the runtime is significantly affected by the storage capacity, the smart grid can be partitioned into several smaller grids with smaller storage capacity. This improvement in runtime would be quadratic in nature. As the focus of this paper is to develop a new supply demand matching framework with reasonable performance, a detailed analysis of such techniques is omitted.

Chapter 7

Conclusion and Future Work

In this chapter, we will provide concluding remarks to the dissertation with a discussion on the broader impacts of the work and the future directions it can lead into.

7.1 Broader Impacts

The rapid increase in the penetration of Distributed Energy Resources such as PVs and batteries in the grid has decentralized energy generation [31]. This has enabled the development of microgrids wherein a localized group of electricity sources and loads constitute a subsystem which can be islanded i.e. operated without any import or export of electricity from outside the subsystem [39]. We envision future smart grids consisting of several microgrids operating as separate monolithic systems which interact with the larger distribution system as a single entity. A holistic microgrid controller which consists of an integrated modeling, prediction and optimization framework is imperative for reliable, secure and economic operation of such microgrids [9]. Optimization frameworks along the line developed in this work will be a significant component of such microgrid controllers.

Moreover, in the absence of fine grained control, several components of the grid exhibit discrete operational values. As evident from Table V in the survey paper [71], while a plethora of optimization algorithms have been developed for grids with continuous operational values (for example, see column titled ”convex optimization problem” in the table), similar algorithms and analysis techniques are lacking for the discrete

case. The research community has focused only on Integer/Mixed Integer Programming based algorithms which are computationally intractable or heuristics which have no theoretical optimality guarantees. The fast and optimal algorithms developed in this work and accompanying theoretical analyses will enable further development of sophisticated optimization frameworks for controlling discrete smart grids.

7.2 Future Directions

Approximation algorithms are powerful tools to develop provably optimal polynomial runtime algorithms for a wide range of NP hard problems. In this section, we discuss a couple of open problems in the domain of smart grid and the larger domain of smart cities/infrastructure.

7.2.1 Discrete Multi Phase Supply Demand Matching

The algorithms developed in this work focus on supply demand matching of active single phase power assuming no power losses due to resistance and reactance of the distributors and feeders. The assumption here is that any fluctuations in reactive power that occurs by modifying the active power can be addressed by reactive power control techniques such as Volt-Var Optimization [52]. In order to create a holistic smart grid

control framework, this assumption needs to be addressed. Consider the relaxed branch flow model and the optimal power flow (OPF) problem defined over it [32]:

$$\min_{p,q,P,Q,l,v} f \tag{7.1}$$

subject to

$$p_j = \sum_{k:j\to k} P_{jk} - \sum_{i:i\to j} (P_{ij} - r_{ij} l_{ij}) + g_j v_j, \qquad \forall j \tag{7.2}$$

$$q_j = \sum_{k:j\to k} Q_{jk} - \sum_{i:i\to j} (Q_{ij} - x_{ij} l_{ij}) + b_j v_j, \qquad \forall j \tag{7.3}$$

$$v_j = v_i - 2(r_{ij} P_{ij} + x_{ij} Q_{ij}) + (r_{ij}^2 + x_{ij}^2) l_{ij} \qquad \forall (i,j) \in \mathcal{E} \tag{7.4}$$

$$l_{ij} \geq \frac{P_{ij}^2 + Q_{ij}^2}{v_i} \qquad \forall (i,j) \in \mathcal{E} \tag{7.5}$$

The above OPF is defined on a smart grid modeled using a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edges denoted using (i, j) or $i \to j$, and i and j being the vertices. p_j and q_j denote the active and reactive power injected at vertex j. v_j denotes the voltage magnitude. P_{ij}, Q_{ij} and l_{ij} denote the active power, reactive power and the magnitude of current flowing over edge $i \to j$ respectively.

Assuming all the variables as continuous entities, the above OPF is a convex relaxation. It is an exact relaxation for radial network under certain assumptions with respect to the objective function and the maximum value of p_j and q_j.

Now, in order to solve the discrete version of the above OPF, the variables p_j and q_j will belong to a discrete set instead of being continuous variables. Sophisticated approximation algorithms to solve this formulation is an open research problem.

7.2.2 Joint Optimization over Multiple Infrastructure for Smart Cities

The proliferation of Internet-of-Things (IoT) based technologies is able to incorporate seamlessly a large number of different and heterogeneous systems such as smart grids, smart transportation [74] thereby evolving the current cities into smart cities. Several components of such smart cities will exhibit discrete operational values and thus sophisticated approximation algorithms will be required to jointly optimize over multiple infrastructures. For example, the charging rates of Electric vehicles form a discrete set of values. Similarly, the problem of smart parking needs to optimize over binary variables denoting the occupancy or reservation status of parking spots [42].

Consider the following problem. A transportation company has a fleet of Electric Vehicles (trucks or cars). From a starting location, each EV needs to visit a number of pre-defined locations (for example to deliver goods) and return to the original location. Given a graphical model of the transportation network, and assuming the costs are in metric space, this problem is the classical traveling salesman problem (TSP) with an approximation factor of 1.5.

In the context of smart cities, the problem can be extended as follows. In addition to visiting the locations for delivery, the EVs can also visit charging station. In the charging station, the EVs can charge themselves at some cost or provide the grid with a portion of the energy stored in their batteries, thereby receiving a monetary compensation. Hence, the problem that needs to solved is to minimize the sum of the charging costs, the traveling costs minus the discharging compensation while ensuring all the locations are visited within a fixed deadline.

7.3 Concluding Remarks

Approximation algorithms are strong techniques to solve several intractable real world problems in a reasonable amount of time while providing theoretical worst case guarantees. However, their application in the domain of smart grids and the more general domain of smart cities has been limited. In this work, we show that using these techniques for solving the problem of discrete supply demand matching in smart grids can lead to significant improvements over the state of the art.

The work done in this dissertation are first steps towards wider applications of approximation algorithms in these domains. We hope that this dissertation can spark the interest of research community in the applications of approximation algorithms for smart grids and smart cities. This will lead to the development of scalable and reliable solutions for various challenging problems in these domains. Moreover, new theoretical insights developed as a result will broaden the understanding of several core problems of interest to the theoretical computer science community.

Reference List

[1] Ilog cplex optimization studio welcome page. Online: http://www-01.ibm.com/support/knowledgecenter/SSSA5P/welcome.

[2] Opl language reference manual.

[3] Usc smart grid. Online.

[4] Car battery efficiencies. Online: http://large.stanford.edu/courses/2010/ph240/sun1/, 2010.

[5] National solar radiation data base, 2010.

[6] Photovoltaic (pv) pricing trends: Historical, recent, and near-term projections, 2013.

[7] How to calculate the annual solar energy output of a photovoltaic system?, 2013-2016.

[8] Matlab-mathworks. https://www.mathworks.com/products/matlab.html, 2017.

[9] Deep solar. Online: http://deepsolar.usc.edu/, 2018.

[10] S. Alamdari, T. Biedl, T. M. Chan, E. Grant, K. R. Jampani, S. Keshav, A. Lubiw, and V. Pathak. Smart-grid electricity allocation via strip packing with slicing. In *Workshop on Algorithms and Data Structures*, pages 25–36. Springer, 2013.

[11] M. H. Albadi and E. El-Saadany. Demand response in electricity markets: An overview. In *IEEE power engineering society general meeting*, volume 2007, pages 1–5, 2007.

[12] S. Aman, C. Chelmis, and V. Prasanna. Learning to reduce: A reduced electricity consumption prediction ensemble. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] S. Aman, C. Chelmis, and V. K. Prasanna. Influence-driven model for time series prediction from partial observations. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[14] S. Aman, M. Frincu, C. Chelmis, M. Noor, Y. Simmhan, and V. K. Prasanna. Prediction models for dynamic demand response. In *IEEE International Conference on Smart Grid Communications (SmartGridComm): Data Management, Grid Analytics, and Dynamic Pricing*, 2015.

[15] N. Bansal, N. Korula, V. Nagarajan, and A. Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1):533–565, 2012.

[16] A. Barbato, C. Bolchini, M. Delfanti, A. Geronazzo, G. Accetta, A. Dede, G. Massa, and M. Trioni. An energy management framework for optimal demand response in a smart campus. *ICGREEN*, page 11, 2015.

[17] A. Barbato, A. Capone, L. Chen, F. Martignon, and S. Paris. A power scheduling game for reducing the peak demand of residential users. In *Online Conference on Green Communications (GreenCom), 2013 IEEE*, pages 137–142. IEEE, 2013.

[18] M. Behrangrad, H. Sugihara, and T. Funaki. Analyzing the system effects of optimal demand response utilization for reserve procurement and peak clipping. In *Power and Energy Society General Meeting, 2010 IEEE*, pages 1–7. IEEE, 2010.

[19] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[20] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] M. Charikar, C. Chekuri, and M. Pál. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 257–269. Springer, 2005.

[22] C. Chelmis, S. Aman, M. R. Saeed, M. Frincu, and V. K. Prasanna. Estimating reduced consumption for dynamic demand response. In *Proceedings of the Twenty-Ninght AAAI Conference on Artificial Intelligence. AAAI Press*, 2015.

[23] J. Chen, B. Yang, and X. Guan. Optimal demand response scheduling with stackelberg game approach under load uncertainty for smart grid. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 546–551. IEEE, 2012.

[24] Z. Chen, L. Wu, and Y. Fu. Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization. *IEEE Transactions on Smart Grid*, 3(4):1822–1831, 2012.

[25] S. Choi, S. Park, D.-J. Kang, S.-j. Han, and H.-M. Kim. A microgrid energy management system for inducing optimal demand response. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 19–24. IEEE, 2011.

[26] A. I. Cohen and C. C. Wang. An optimization method for load management scheduling. *IEEE Trans. Power Syst.;(United States)*, 3(2), 1988.

[27] C. Deline. Partially shaded operation of multi-string photovoltaic systems. In *Photovoltaic Specialists Conference (PVSC), 2010 35th IEEE*, pages 000394–000399. IEEE, 2010.

[28] Y. M. Ding, S. H. Hong, and X. H. Li. A demand response energy management scheme for industrial facilities in smart grid. *IEEE Transactions on Industrial Informatics*, 10(4):2257–2269, 2014.

[29] J. Driesen and F. Katiraei. Design for distributed energy resources. *IEEE Power and Energy Magazine*, 6(3), 2008.

[30] A. T. Elsayed, C. R. Lashway, and O. A. Mohammed. Advanced battery management and diagnostic system for smart grid infrastructure. *IEEE Transactions on Smart Grid*, 7(2):897–905, 2016.

[31] H. Farhangi. The path of the smart grid. *IEEE power and energy magazine*, 8(1), 2010.

[32] M. Farivar and S. H. Low. Branch flow model: Relaxations and convexificationpart i. *IEEE Transactions on Power Systems*, 28(3):2554–2564, 2013.

[33] O. Gagrica, P. H. Nguyen, W. L. Kling, and T. Uhl. Microinverter curtailment strategy for increasing photovoltaic penetration in low-voltage networks. *IEEE Transactions on Sustainable Energy*, 6(2):369–379, 2015.

[34] N. Gatsis and G. B. Giannakis. Cooperative multi-residence demand response scheduling. In *Information Sciences and Systems (CISS), 2011 45th Annual Conference on*, pages 1–6. IEEE, 2011.

[35] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[36] S. R. Kuppannagari, R. Kannan, C. Chelmis, and V. K. Prasanna. Implementation of learning-based dynamic demand response on a campus micro-grid. In *The 25th International Joint Conference on Artificial Intelligence*, 2016.

[37] S. R. Kuppannagari, R. Kannan, C. Chelmis, A. S. Tehrani, and V. K. Prasanna. Optimal customer targeting for sustainable demand response in smart grids. *Procedia Computer Science*, 80:324–334, 2016.

[38] J. Kwac and R. Rajagopal. Demand response targeting using big data analytics. In *Big Data, 2013 IEEE International Conference on*, pages 683–690. IEEE, 2013.

[39] R. H. Lasseter and P. Paigi. Microgrid: A conceptual solution. In *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, volume 6, pages 4285–4290. IEEE, 2004.

[40] S. Lee, S. Iyengar, D. Irwin, and P. Shenoy. Distributed rate control for smart solar arrays. In *Proceedings of the Eighth International Conference on Future Energy Systems*, pages 34–44. ACM, 2017.

[41] T. Logenthiran, D. Srinivasan, and T. Z. Shun. Demand side management in smart grid using heuristic optimization. *Smart Grid, IEEE Transactions on*, 3(3):1244–1252, 2012.

[42] R. Lu, X. Lin, H. Zhu, and X. Shen. Spark: a new vanet-based smart parking scheme for large parking lots. In *INFOCOM 2009, IEEE*, pages 1413–1421. IEEE, 2009.

[43] L. Lutzenhiser, L. Cesafsky, H. Chappells, M. Gossard, M. Moezzi, D. Moran, J. Peters, M. Spahic, P. Stern, E. Simmons, et al. Behavioral assumptions underlying california residential sector energy efficiency programs. *Prepared for the California Institute for Energy and Environment Behavior and Energy Program*, 2009.

[44] Y. V. Makarov, C. Loutan, J. Ma, and P. De Mello. Operational impacts of wind generation on california power systems. *IEEE Transactions on Power Systems*, 24(2):1039–1050, 2009.

[45] F. Mangiatordi, E. Pallotti, P. D. Vecchio, and F. Leccese. Power consumption scheduling for residential buildings. In *Environment and Electrical Engineering (EEEIC), 2012 11th International Conference on*, pages 926–930. IEEE, 2012.

[46] A. Molderink, V. Bakker, M. G. Bosman, J. L. Hurink, and G. J. Smit. Domestic energy management methodology for optimizing efficiency in smart grids. In *PowerTech, 2009 IEEE Bucharest*, pages 1–7. IEEE, 2009.

[47] H. Moradi, D. Groff, and A. Abtahi. Optimal energy schadualing of a stand-alone multi-sourced microgid considering environmental aspects. In *2017 IEEE Innovative Smart Grid Technologies Conference (ISGT)*, 2017.

[48] K. Moslehi and R. Kumar. A reliability perspective of the smart grid. *Smart Grid, IEEE Transactions on*, 1(1):57–64, 2010.

[49] S. J. Moss, M. Cubed, and K. Fleisher. Market segmentation and energy efficiency program design. *Berkeley, California Institute for Energy and Environment*, 2008.

[50] N. Motegi, M. A. Piette, D. S. Watson, S. Kiliccote, and P. Xu. Introduction to commercial building control strategies and techniques for demand response. *Lawrence Berkeley National Laboratory LBNL-59975*, 2007.

[51] F. Rahimi and A. Ipakchi. Demand response as a market resource under the smart grid paradigm. *Smart Grid, IEEE Transactions on*, 1(1):82–88, 2010.

[52] S. Rahimi, M. Marinelli, and F. Silvestro. Evaluation of requirements for volt/var control and optimization function in distribution management systems. In *Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International*, pages 331–336. IEEE, 2012.

[53] P. Ramsami and V. Oree. A hybrid method for forecasting the energy output of photovoltaic systems. *Energy Conversion and Management*, 95:406–413, 2015.

[54] A. Ranjan, P. Khargonekar, and S. Sahni. Offline preemptive scheduling of power demands to minimize peak power in smart grids. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6. IEEE, 2014.

[55] W. Romeijnders, L. Stougie, and M. H. van der Vlerk. Approximation in two-stage stochastic integer programming. *Surveys in Operations Research and Management Science*, 19(1):17–33, 2014.

[56] S. Rongali, T. Ganuy, M. Padmanabhan, V. Arya, S. Kalyanaraman, and M. I. Petra. iplug: Decentralised dispatch of distributed generation. In *Communication Systems and Networks (COMSNETS), 2016 8th International Conference on*, pages 1–8. IEEE, 2016.

[57] T. Roy, A. Das, and Z. Ni. Optimization in load scheduling of a residential community using dynamic pricing. In *2017 IEEE Innovative Smart Grid Technologies Conference (ISGT)*, 2017.

[58] N. Ruiz, I. Cobelo, and J. Oyarzabal. A direct load control model for virtual power plant management. *IEEE Transactions on Power Systems*, 24(2):959–966, 2009.

[59] A. Ruszczyński and A. Shapiro. Stochastic programming models. *Handbooks in operations research and management science*, 10:1–64, 2003.

[60] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM (JACM)*, 22(1):115–124, 1975.

[61] A. Sepulveda, L. Paull, W. G. Morsi, H. Li, C. Diduch, and L. Chang. A novel demand side management program using water heaters and particle swarm optimization. In *Electric Power and Energy Conference (EPEC), 2010 IEEE*, pages 1–5. IEEE, 2010.

[62] Y. Simmhan, V. Prasanna, S. Aman, S. Natarajan, W. Yin, and Q. Zhou. Toward data-driven demand-response optimization in a campus microgrid. In *Proceedings of the Third ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 41–42. ACM, 2011.

[63] A. Singh, S. Lee, D. Irwin, and P. Shenoy. Sunshade: enabling software-defined solar-powered systems. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pages 61–70. ACM, 2017.

[64] B. A. Smith, J. Wong, and R. Rajagopal. A simple way to use interval data to segment residential customers for energy efficiency and demand response program targeting. In *ACEEE Summer Study on Energy Efficiency in Buildings*, 2012.

[65] P. Srikantha and D. Kundur. Distributed demand curtailment via water-filling. In *IEEE International Conference on Smart Grid Communications*, 2015.

[66] H.-I. Su and A. El Gamal. Modeling and analysis of the role of fast-response energy storage in the smart grid. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 719–726. IEEE, 2011.

[67] SunShot. Sunshot vision study a comprehensive analysis of the potential for u.s. solar electricity generation, 2012.

[68] S. Tang, Q. Huang, X.-Y. Li, and D. Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *INFOCOM, 2013 Proceedings IEEE*, pages 1133–1141. IEEE, 2013.

[69] C.-W. Ten, C.-C. Liu, and G. Manimaran. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, 2008.

[70] R. Tonkoski, L. A. Lopes, and T. H. El-Fouly. Coordinated active power curtailment of grid connected pv inverters for overvoltage prevention. *IEEE Transactions on Sustainable Energy*, 2(2):139–147, 2011.

[71] J. S. Vardakas, N. Zorba, and C. V. Verikoukis. A survey on demand response programs in smart grids: Pricing methods and optimization algorithms. *IEEE Communications Surveys & Tutorials*, 17(1):152–178, 2015.

[72] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[73] S. Yaw, B. Mumey, E. McDonald, and J. Lemke. Peak demand scheduling in the smart grid. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 770–775. IEEE, 2014.

[74] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[75] Z. Zhu, J. Tang, S. Lambotharan, W. H. Chin, and Z. Fan. An integer linear programming based optimization for home demand-side management in smart grid. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–5. IEEE, 2012.

[76] V. Zois, M. Frincu, C. Chelmis, M. R. Saeed, and V. Prasanna. Efficient customer selection for sustainable demand response in smart grids. In *IGCC*, pages 1–6. IEEE, 2014.

[77] V. Zois, M. Frincu, and V. Prasanna. Integrated platform for automated sustainable demand response in smart grids. In *Intelligent Energy Systems (IWIES), 2014 IEEE International Workshop on*, pages 64–69. IEEE, 2014.