Software-Defined Infrastructure for IoT-based Energy Systems

Stephen Lee

SOFTWARE-DEFINED INFRASTRUCTURE FOR IOT-BASED ENERGY SYSTEMS

A Dissertation Presented

by

STEPHEN LEE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2019

College of Information and Computer Sciences

SOFTWARE-DEFINED INFRASTRUCTURE FOR IOT-BASED ENERGY SYSTEMS

A Dissertation Presented

by

STEPHEN LEE

Approved as to style and content by:

ACKNOWLEDGMENTS

This thesis has been an immense learning experience, and I am sincerely grateful to many who supported me on this journey. Foremost, I would like to express my deepest gratitude to my advisor, Prof. Prashant Shenoy, for, without his continuous guidance, this thesis would not have been possible. His enthusiasm, encouragement, and constant support have helped me throughout my study. I have learned so much from him on various aspects of research: focussing on the right problems, prioritizing my research efforts, and the importance of presenting my ideas to others. I could not have asked for a better advisor and mentor for my Ph.D. study.

I would also like to thank my collaborators for their participation and feedback, which greatly improved the quality of this thesis. I am particularly grateful to Prof. David Irwin, who has, on several occasions, provided his mentorship and support. I have been fortunate to work with Prof. Ramesh Sitaraman, who taught me the importance of clarity in research. I have also collaborated with researchers outside of UMass. I want to thank Prof. Krithi Ramamritham for his valuable guidance and dedication to research. I am also grateful to Prof. Srinivasan Keshav for mentoring me during my time at the University of Waterloo and for providing me with a broader perspective on doing Systems research. I would also like to thank my committee member, Prof. Deepak Ganesan, for his valuable feedback and interest in my research.

I wouldn't have started my journey had it not been for the support and encouragement of some key people. I am deeply grateful to Dr. Harrick Vin, Dr. Raman Srinivasan, Dr. Usha Rani, Dr. Sriram Nambiar, Rahul Kelkar, Dr. Madhavan Mukund, Dr. Shirish Karande, Dr. Dilys Thomas, Dr. Rajeev Raman — each of whom played an important part and provided me the necessary guidance to succeed. I would especially like to thank Dr. Harrick Vin

and Rahul Kelkar for giving me the opportunity to work at Tata Research Development and Design Center (TRDDC), which helped in developing my interest in research.

My time in Amherst wouldn't have been so enjoyable and memorable without my friends and colleagues. I would especially like to thank Srinivasan Iyengar, who has been a great friend, and I really enjoyed working with him in all of our research collaborations. His constant encouragement and support brought the best in me, and his feedback helped in shaping many of my ideas. I am also thankful to Prateek Sharma and Tian Guo. Both have been great friends, and with whom I have collaborated, played frisbees, cooked food, and helped me take a break from research. I am also grateful to Navin Sharma, who advised me on numerous things, and also, taught me how to drive a car. I would also like to thank my friends and colleagues, especially Amee Trivedi, Aditya Mishra, Jeremy Gummeson, Xin He, Lucas Chaufournier, Vani Gupta, Christian Gorenflo, John Wamburu, Phuthipong Bovornkeeratiroj, and many others — for their company and making the lab a fun place to work.

I have also developed an incredible group of friends in Amherst, which made life even more enjoyable. I want to thank Abhyuday Jagannatha for the stimulating conversations, his advice, and all the fun we had together hiking, climbing, and biking. I would also like to thank all my friends: Abhishek Roy, Paul Datta, Mohit Sharma, Saurabh Sarkar, Abhishek Agarwal, and many others for their joyful company and conversations during my Ph.D. I am thankful to Leeanne Leclerc and Eileen Hamel for their help in navigating the graduate paperwork, and to Karren Sacco for assisting many of the administrative tasks.

Finally, I want to thank my family — my parents and brothers — for their endless love and support. If not for their continuous encouragement, I might not have been where I am today. And last but not least, I owe a very special thanks to Caroline Lo, who has wholeheartedly supported me in every way, and I look forward to continuing our journey together.

ABSTRACT

SOFTWARE-DEFINED INFRASTRUCTURE FOR IOT-BASED ENERGY SYSTEMS

SEPTEMBER 2019

STEPHEN LEE

B.Sc., ST. STEPHEN'S COLLEGE, DELHI

M.Sc., CHENNAI MATHEMATICAL INSTITUTE, CHENNAI

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Prashant Shenoy

Internet of Things (IoT) devices are becoming an essential part of our everyday lives. These physical devices are connected to the internet and can measure or control the environment around us. Further, IoT devices are increasingly being used to monitor buildings, farms, health, and transportation. As these connected devices become more pervasive, these devices will generate vast amounts of data that can be used to gain insights and build intelligence into the system. At the same time, large-scale deployment of these devices will raise new challenges in efficiently managing and controlling them.

In this thesis, I argue that the IoT devices need programmability and need to provide software controls in order to manage them efficiently. Further, it will need data-driven modeling techniques to process and analyze a vast amount of data from heterogeneous devices to derive actionable insights. My thesis explores the problems posed by software-defined IoT energy infrastructure. I present four techniques that use systems and machine learning principles to design, analyze and deploy the next generation of smart IoT energy systems.

First, I discuss how current state-of-the-art LIDAR-based approaches in identifying ideal locations on rooftops for deploying energy systems such as solar do not scale to many regions of the world. To address the challenges, I propose DeepRoof, a data-driven approach that uses deep learning to estimate the solar potential of roofs using satellite imagery and identify ideal locations for installation. We evaluate our approach on different types of roof and show that our technique is comparable to LIDAR-based methods.

Second, I study how excessive solar can cause problems in the grid and examine how programmatic control of the solar output can prevent congestion in the electric grid. Further, I present a decentralized approach that can control the solar arrays in a grid-friendly manner. Also, my approach provides flexible control of solar output, and I show that such mechanisms allow for higher solar penetration in the grid.

Third, I discuss the challenges in community-owned (and shared) distributed energy resources that do not provide independent control to users. To do so, I propose vSolar, an approach to virtualize the solar arrays and energy storage that allows independent control. Further, I show how using vSolar users can exercise independent control, implement their custom energy sharing policies, and reduce energy costs through energy trading.

Finally, I present the challenges, and the high throughput needs to enable a peer-to-peer energy trading platform using permissioned blockchains. I propose FabricPlus, an enhanced Hyperledger Fabric blockchain, that contains a series of optimizations to enable high throughput transactions. FabricPlus increases the transaction throughput many folds, without requiring any changes to its external interfaces. I also show considerable performance improvement over the baseline Fabric.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Advances in hardware and wireless technologies have made it feasible to deploy pervasive sensing and computing that integrate into every aspect of our physical world. This has led to the emergence of Internet of Things (IoT), a network of physical devices, that are cheap, easily deployable, and connected to the Internet. Consequently, large-scale deployments of distributed IoT is becoming commonplace that can sense, monitor, and actuate in new and exciting ways. As such, this has enabled many use cases in multiple domains such as transportation, healthcare, and smart cities. It is estimated that 50 billion IoT devices will be connected to the Internet by 2020 [106]. This thesis discusses the challenges and opportunities in designing software-defined infrastructure for managing and coordinating these large-scale distributed devices in the context of IoT energy systems by using principled approaches from both systems and machine learning.

1.1 Motivation

Internet of Things (IoT) devices comprises a vast number of diverse and heterogenous embedded sensors and actuators that interact with the physical environment. These devices consist of embedded software and electronics that are engineered to enable a host of functions such as sensing, connectivity, and actuation, thereby enhancing the capabilities of everyday devices. As they continuously monitor our environment, they produce vast amounts of data [63, 64], which can be used to gather insights and model the behavior of the system, necessary for any data-driven response system.

Unfortunately, these distributed IoT devices tend to operate within silos, defined by manufacturers and associated technology stacks. That is, IoT services tend to be closed within the manufacturers' ecosystem, where the interactions and data exchanges are limited to only specific devices and services. They often use non-standard technology and software, which prevents integration with devices from other domains or manufacturers. This limits the degree of control and interoperability across IoT devices, of various capabilities, from other manufacturers and service providers. We already see these limitations in today's (so-called) IoT devices. For instance, many smart switches are not accessible or incompatible with smart home assistants, making it challenging to coordinate and create richer applications.

Connecting heterogeneous devices in flexible ways raise a myriad of challenges since the interactions are pre-defined, and any communication outside of the application domain is discouraged. Such pre-negotiated controls, both within and outside of the system, prevent wide-scale interoperability, but more importantly, cause difficulty in management and control when multiple devices are involved. This lack of flexibility is reminiscent of legacy networks built on rigid and inflexible components, such as switches and routers, that did not support *programmability*. But modern networking now employs a flexible *software-defined network* (SDN) architecture that emphasizes programmability, and flexible abstractions for easy management and control. Such flexible abstractions allow efficient management of network devices and create a more dynamic and agile system.

Similar to how SDN provides programmability and flexible abstractions, we argue that the next generation of IoT devices will need software-defined infrastructures to enable users to exercise software control and create new use cases. However, unlike SDN, a software-defined infrastructure raises new challenges in designing systems that can work seamlessly with a myriad of IoT devices. First, it will need data-driven analytics that can process and analyze data from heterogeneous IoT devices to gather actionable insights. Second, it will need to provide flexible actuation and control across the IoT devices that support interaction among devices. Third, it will need efficient resource management techniques that can manage large-scale IoT deployments. Finally, it should scale to the demands of IoT devices and handle data transactions from a large number of sensors.

In this thesis, I explore the problems posed by designing software-defined infrastructures in the context of IoT-based energy systems. As the current power grid becomes increasingly integrated with IoT-based energy systems (e.g., solar arrays, energy storage, electric vehicles), I argue that there is an opportunity to develop software-defined infrastructures that can improve the responsiveness and effectiveness of such systems.

In particular, my thesis seeks to address the following problems:

- How to use large-scale data to design data-driven approaches for planning and deployment of IoT energy systems?
- How can distributed IoT-based energy systems communicate and self-regulate its output in the power grid?
- How to provide flexible software-defined abstractions in IoT-based energy systems to support custom user policies?
- How to scale and improve the performance of existing software systems to enable high throughput IoT-based data transactions?

The questions above describe some of the challenges we need to overcome to design software-defined infrastructures for energy systems. By addressing these questions, I seek to draw upon design principles from both systems and machine learning to build this software-defined infrastructure. In doing so, I develop novel techniques for managing IoT-based energy systems.

1.2 Thesis Contributions

The thesis proposes novel techniques that provide new algorithms and mechanisms for managing IoT-based energy systems and data-driven methods for analysis and control of such systems. Specifically, this thesis considers solar and battery-based energy systems and enables flexible control and intelligence. To this end, this thesis makes the following key contributions:

1. *Planning and placement*: A data-driven system that uses deep-learning based approach to identify ideal locations for installing solar arrays on rooftops by estimating solar potential using satellite imagery.

2. *Decentralized Control*: A decentralized approach to dynamically rate control solar arrays, thereby preventing congestion collapse and allowing high renewable penetration in the grid.

3. *Resource Management*: A virtualization abstraction mechanism that enable independent control and management of energy systems across multiple users, in effect providing a 'virtual system' to implement user-defined policies.

4. *Decentralized Architecture*: An architecture that allows a blockchain system to achieve high throughput IoT energy transactions without necessitating any change in its external interfaces.

Each component is described in more detail below.

1.2.1 Planning and Placement

It is estimated that the annual energy generation potential of small building rooftop PV is 9.26 terawatt-hours (TWh) — one-fourth the total electricity sales in the US [44]. Since rooftop solar has tremendous potential in generating output, solar potential estimation of a roof and identifying ideal locations for installation can substantially benefit homeowners deciding to adopt solar. Until recently, to estimate the solar potential of a roof requires homeowners to consult contractors, who manually evaluate the site. Even state-of-the-art methods for estimating the solar potential of a roof works only for places where LIDAR data is available, thereby limiting their reach to just a few places in the world. I propose

DeepRoof, a data-driven system that uses satellite images and other third-party sources for assessing the solar potential of a roof. This allows for better scalability as these data are readily available through public APIs. Further, DeepRoof can provide a pixel-level estimate of solar potential that helps in identifying ideal spots on a roof for installing solar panels.

1.2.2 Decentralized Control

Net metering allows consumers to feed the surplus electricity back to the grid, effectively selling electricity to the utility. However, net metering large amounts of solar power to the electric grid is problematic as grid operators must continuously balance supply and demand. If the total net-metered output from intermittent solar arrays fluctuates too rapidly, it can cause supply and demand mismatches. Limiting the solar capacity reduces this stochasticity seen from these distributed sources, which makes supply and demand more manageable. To regulate the solar output, in this thesis, I propose a decentralized rate control technique that can self-regulate in a grid-friendly manner. Similar to rate control of network flows in TCP, generation sources back off when there is congestion in the network and increase the rate when network capacity is available. We show that this provides flexible control of solar output and allows for higher penetration of solar in the grid. Further, solar rate control also provides grid operators with an additional control knob when continuously matching supply and demand.

1.2.3 Distributed Resource Management

Penetration of residential solar installations has grown rapidly in recent years. Unfortunately, due to space constraints, many residential locations are unsuitable for solar deployments, and community-owned solar arrays with energy storage that are collectively shared by a group of owners have emerged as a solution. However, such a group-owned system does not allow individual control over how the electricity generated from these energy systems is used for optimizing a home's electricity bill. To overcome this limitation, I propose vSolar, a technique that virtualizes a large-deployment of solar and battery arrays and provides a flexible abstraction of a virtual solar and battery array to each owner. Importantly, virtualization enables each owner to independently manage their solar generation and stored energy as if it were a dedicated system. A second key benefit of virtualization of a community-owned system is that it enables the sharing of electricity generated or stored in batteries by each virtual system. Such energy sharing, which is not possible in dedicated independently deployed systems, allows a resident to temporarily borrow electricity from one or more neighbor's shares to provide capital and operational savings.

1.2.4 Decentralized Architecture

With an increasing number of small-scale rooftop solar installations, the centralized grid is slowly transitioning into a decentralized grid having distributed generation sources. This has led to a new paradigm, where individuals have surplus solar energy and can supply this electricity to others. To realize such a trading system, where individuals can buy or sell electricity, requires a ledger — an accounting system that can track energy transactions between two parties. Recently, the use of blockchain technology has gained traction to enable such peer-to-peer energy trading. Unfortunately, current blockchain technology does not support high throughput transactions necessary to facilitate energy transactions at scale. I propose FabricPlus, a permissioned blockchain system that can support such high throughput transactions necessary for energy trading. To do so, I re-architect an existing opensource blockchain system, namely Hyperledger Fabric. I propose a series of optimizations that increases the transaction throughput many folds, without requiring any changes to its external interfaces.

1.3 Thesis Outline

The remainder of the thesis is structured as follows. **Chapter 2** provides background on IoT-based energy systems and discusses prior work. **Chapter 3** presents DeepRoof, a data-driven approach to estimating the solar potential of roof.**Chapter 4** describes an optimization approach to regulating solar output. **Chapter 5** discusses vSolar, an approach to virtualize solar arrays and energy storage, and provide a platform for implementing user-specific energy policies. In **Chapter 6**, proposes an architecture that achieves high throughput energy transactions on a blockchain-based platform. Finally, **Chapter 7** presents the thesis summary and future work.

CHAPTER 2
BACKGROUND AND RELATED WORK

This chapter provides background and related work on energy systems, specifically, solar arrays and energy storage.

2.1 Solar Arrays

Advances in technology and declining cost continue to stimulate solar adoption among homeowners. This has given rise to highly distributed solar sites that can supply clean energy but has also opened new research challenges in deploying, managing, and controlling these systems. Below, we provide background on how solar arrays are sized and deployed in residential homes, implications of rising solar adoption on the power grid, and emerging challenges.

2.1.1 Sizing and Deployment

Solar potential analysis of a roof help understand the recommended areas for installing solar panels as well as estimating the potential benefits in energy cost savings if any. Solar potential of a location can be defined as the amount of available solar energy over a given period. A standard measure for estimating and analyzing the availability of solar is *peak sun hours*, which captures the amount of solar insolation a location receives on a typical day (see Figure 2.1(a)). Specifically, a peak sun hour is an hour during which the solar intensity is $1kW/m^2$. Thus, peak sun hours provide a rough estimate of the potential of an area, as it accounts for the various factors that affect available sunlight.

The amount of available sunlight depends on various factors such as the sun's position in the sky, geography, and local climate conditions such as clouds. Figure 2.1 illustrates

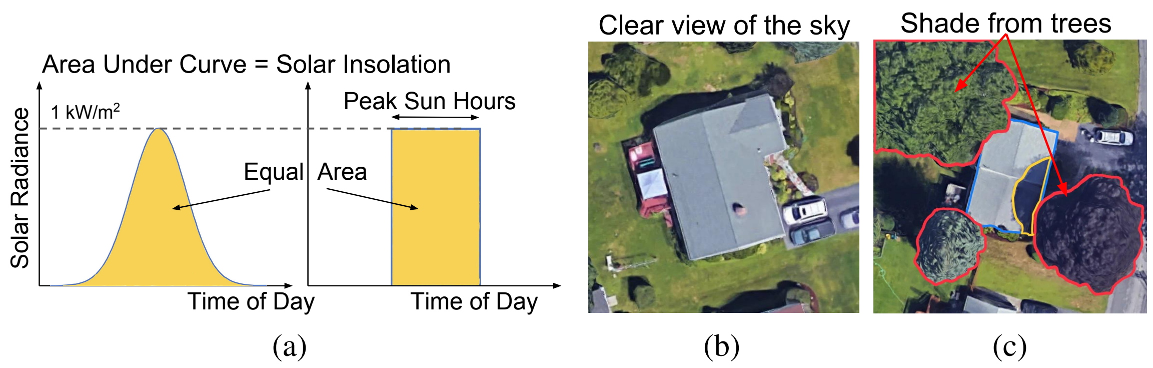


Figure 2.1. (a) Relationship between peak sun hours and cumulative solar irradiation. (b) Roof with clear view of the sky (c) Roof with shade from nearby structures.

the amount of solar irradiation a surface receives varies over the day and usually peaks at solar noon. Similarly, cloudy conditions can reduce the amount of irradiance a surface gets. Geographical location also plays an important role, especially in places where days are longer during the summer season. For instance, in higher latitudes, cities have more daylight hours in summer than winter.

A common approach to estimating the solar potential of a location involves pyranometers that measure the solar irradiance falling on a surface. These pyranometers are usually placed on flat surfaces with a clear view of the sky and record the solar insolation a location receives under "ideal" conditions. These solar insolation values are accessible online for several locations around the world [45].

2.1.1.1 Challenges

Assessing the *solar potential of a roof* is challenging as several local factors are involved. One such factor is a roof's geometry. A roof's geometry is defined by its (i) orientation — the direction the roof is facing and (ii) pitch — the slope of the roof. Intuitively, the amount of energy generated is proportional to the sunlight incident on the roof's surface. In the northern hemisphere, a south-facing roof has more direct sunlight exposure than roofs that face in other directions. Thus, the orientation angle of the roof (i.e., the

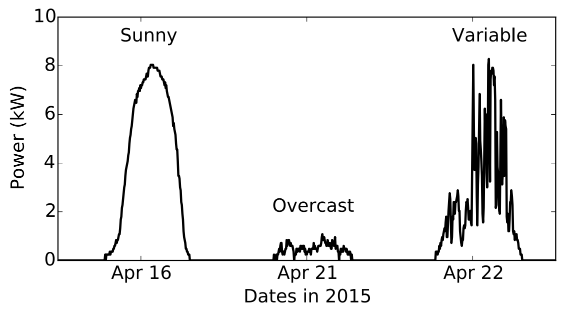


Figure 2.2. Solar power output varies based on the time of day and weather conditions.

horizontal angle measured from the north) determines the actual solar generation output. Similarly, the pitch of the roof also governs the amount of sunlight it receives. A surface that is perpendicular to the incident sunlight will receive more sunlight as more surface area is exposed, whereas a surface parallel to the incident sunlight will receive no sunlight. Thus, solar installers position the PV panels to the latitude of a location to maximize the area exposed to sunlight.

Another factor that affects the solar potential of a rooftop is their *local terrain* (see Figure 2.1(b) and (c)). While roofs with a clear view of the sky receive maximum sunlight, buildings with shade from nearby structures such as trees can significantly reduce the amount of solar irradiance incident on its roof. Since available sunlight will be minimal, such roofs may not merit an investment in a PV installation. Thus, these local factors need to be considered to assess a roof's solar potential.

2.1.2 Grid-tied Solar Arrays

Solar arrays installed on residential buildings can be connected to the grid through *net metering*. Net metering is a billing mechanism that allows these grid-tied solar arrays to feed the surplus power into the grid, effectively selling electricity back to the utility, thereby reducing month energy bills. However, solar energy generation is intermittent and highly weather dependent (see Figure 2.2). For example, on sunny days, the amount of solar-generated by a panel is at its maximum, but on overcast days, the amount of solar generation may be relatively low. Thus, the amount of solar power "net-metered" to the grid depends on (i) local demand from loads, and (ii) the solar radiation incident on the panel, which is weather dependent.

2.1.2.1 Challenges

As solar penetration grows, the impact of intermittent solar *complicates* balancing the supply and demand. If the net-metered output from solar arrays fluctuates rapidly due to changing weather patterns and local demands, it can increase grid dynamics and cause supply-demand mismatches. Since most conventional power sources have limited generation capacity and take time to ramp up its power to full capacity, managing such large fluctuations in output may become increasingly challenging for grid operators.

To avoid using "excessive" amount of solar power from being injected into the grid, many governments strictly regulate grid solar connections [12]. Many states in the US set hard limits by passing laws to regulate the number of solar installations. Limiting the solar capacity limits the stochasticity seen from these distributed sources, which in turn makes matching supply and demand a more manageable problem despite intermittency. For example, while the state of Virginia has a cap of 1%, a similar law exists in Massachusetts that caps the solar at 2% of the total power generation. Importantly, these caps are generally based on the rated maximum capacity of a solar installation, regardless of what it actually generates. That is, the caps assume a solar array outputs maximum capacity all the time.

2.2 Energy Storage

A key challenge arising from the increased penetration of renewable sources is their intermittent nature [60, 63, 128, 129]. Many factors affect solar output including the sun's position, shade from trees, and cloud cover in the sky. Such fluctuations in production, as well as lack of solar output during the night, complicates the management of the grid where supply and demand must be continuously matched. Energy storage in the form of batteries has been proposed as a potential solution for dealing with intermittency from a renewable source [50, 78, 99]. Battery-based energy storage can smooth out fluctuations in output from solar arrays by absorbing surplus energy from solar arrays and feed the excess power back to the grid when there is a deficit.

Energy storage also provides other benefits other than smoothing out fluctuations. For instance, energy storage can exploit electricity price differentials to reduce costs. Since electricity prices may fluctuate over a given period, storing electricity when prices are low and discharging during peak-periods has the potential to cut costs [69, 99]. Similarly, other benefits of energy storage include supporting demand-response [31, 80], providing back-ups [105] and reducing peak usage [109]. Hence, energy storage is seen as a potential solution for solving some of the problems in the existing grid.

Until recently, the high cost of batteries has been a barrier to large-scale energy storage deployments. However, this is beginning to change with the development of new battery technologies and falling prices. Today, battery-based systems such as Tesla Powerwall and others are increasingly deployed in conjunction with solar array deployments.

2.2.1 Virtualization of Energy Systems

Modern distributed systems and computer networks have employed virtualization as a fundamental building block to flexibly multiplex a set of physical resources across users, where virtualized resources resemble the physical counterparts [125, 126]. A virtual representation of energy systems can provide the illusion of a dedicated physical array and battery that can be utilized and managed independently of other virtual arrays and batteries. This allows each owner to make the “optimal” decision of how to utilize their virtual array and batteries independently of what other owners decide at each instant.

A virtualization abstraction also gives each owner an illusion of ownership of the unit, even though the physical resources are collectively owned by the group. In some sense, a virtual solar and battery arrays can act like N independent smaller array and battery

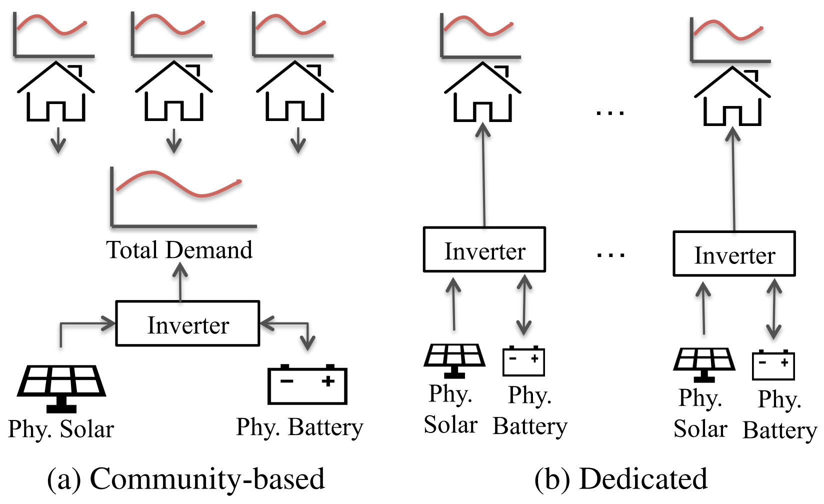


Figure 2.3. Energy use in a non-virtualized community-based and a dedicated solar and battery system.

installations — while being cheaper to install due to economies of scale of installing a single large solar and battery array over N smaller ones. In addition, many components like inverters can be shared rather than duplicated.

Further, a virtual solar array and battery system can be utilized for many differing energy optimizations. In scenarios with time-of-use pricing with different pricing slabs, surplus solar production during off-peak or mid-peak price periods can be stored in the battery for later use, such as peak price periods to maximize cost savings. During peak periods, the system prioritizes the use of solar production and stored energy in the virtual battery overdrawing power from the grid. Finally, if there is surplus solar production after serving local loads and charging the battery, this excess energy can be net metered to the grid to earn revenues (which offset changes in the monthly electricity bill).

2.2.1.1 Challenges

Today physical IoT-based energy systems do not have such virtualization capabilities that allow users to control their share of resources independently. As shown in Figure 2.3(a)), community-owned energy systems, shared across a large number of users, do not provide dedicated control to individual users. Energy output from solar and battery arrays is used to meet the aggregate energy demand across all homes, which prohibits individual owners from deciding on how to use their share of energy. A solution is to use dedicated systems that allow independent control. But, they are expensive and maybe infeasible in many residential locations where space is a constraint (see Figure 2.3(b)).

Virtualization mechanisms can also enable sharing of energy. The virtualization layer can expose control primitives for individuals to determine whom to share energy. If an individual has excess energy from their share, they can sell it to others. Similarly, if they have deficit energy, they can buy energy from others, thereby reducing the reliance on the grid. To enable energy trading among individuals will require an accounting system that tracks the various energy transaction between two parties. The amount of electricity borrowed or lent must be tracked, recorded, and periodically settled using billing infrastructures. Since such transactions may be frequent, such an accounting system will need to support high throughput transactions — especially if there are many individuals trading energy. Further, in a decentralized setting where transactions occur between untrusted parties, the system should record these transactions in a transparent, secure, and auditable manner to maintain trust.

CHAPTER 3

SOLAR POTENTIAL ESTIMATION OF ROOFS

Rooftop solar deployments are an excellent source for generating clean energy. As a result, their popularity among homeowners has grown significantly over the years. Unfortunately, estimating the solar potential of a roof requires homeowners to consult solar consultants, who manually evaluate the site. Recently there have been efforts to automatically estimate the solar potential for any roof within a city. However, current methods work only for places where LIDAR data is available, thereby limiting their reach to just a few places in the world. In this chapter, we present DeepRoof, a data-driven approach that uses widely available satellite images to assess the solar potential of a roof.

3.1 Motivation

Solar deployments vary in size, ranging from large solar farms that are deployed by utilities to small-scale installations by individuals [63,65]. More than half of the installed solar capacity continue to come from small-scale solar deployments, i.e., arrays with 10kW of capacity of less [79]. Most of these installations are residential in nature with deployments on rooftops of homes.

However not all roofs are suitable for solar array deployments. A clear view of the sky with no surrounding obstructions and proper orientation (e.g., south or southwest facing roofs in the northern hemisphere) are key to maximizing solar energy generation of rooftop deployments. In contrast, residential buildings surrounded by trees or other buildings that cast shadows or roofs that do not face south are considered unsuitable for rooftop deployments. The task of determining whether a particular building is well suited for rooftop solar deployment has largely been a manual process—a professional solar energy installer measures the roof area, their orientation, and uses a pyranometer and shade measurement tools[1] to assess the amount of sunlight received on the roof for different times of the day. These measurements are then used to find the ideal locations for installing solar arrays on the roof, if any, and to compute the solar generation potential of the roof. Such a process is laborious and time-consuming and certainly does not scale to large number of buildings in a city.

There have been a few recent efforts that have attempted to automate this laborious process using data-driven algorithm [67, 142]. For instance, Mapdwell [67] and Google's Project Sunroof [114] have both used LIDAR data to assess the solar potential of building rooftops in a city. LIDAR is a laser-based aerial mapping technology that uses airborne LIDAR sensors to extract the 3D surface structure to create a Digital Elevation Model (DEM), which can then be used to determine the geometry of the roof as well as shade from nearby objects [142]. Unfortunately LIDAR data is expensive to collect and involve flying airplanes or drones with aerial LIDAR mapping sensor, and thus, such data is not widely available for many regions in the US and the world. Consequently, current state-of-the-art techniques only offer solar potential data for select cities where LIDAR data is available, leaving large parts of the world without any coverage.

At the same time, satellite images showing rooftops of buildings are widely available for most countries through mapping services such as Google, Bing Maps or commercial ones such as DigitalGlobe. Our key hypothesis is that advances in computer vision techniques make it feasible to use 2D rooftop satellite imagery to automate the estimation of solar generation potential for any building rooftop. In our case, key research questions include whether it is feasible to (i) recognize a rooftop from its surroundings, and (ii) infer the 3D shape of the roof, and specifically the plane of each roof surface and their orientation, and

[1]Solar Path Finder, SunEye 210 `http://www.solmetric.com/`

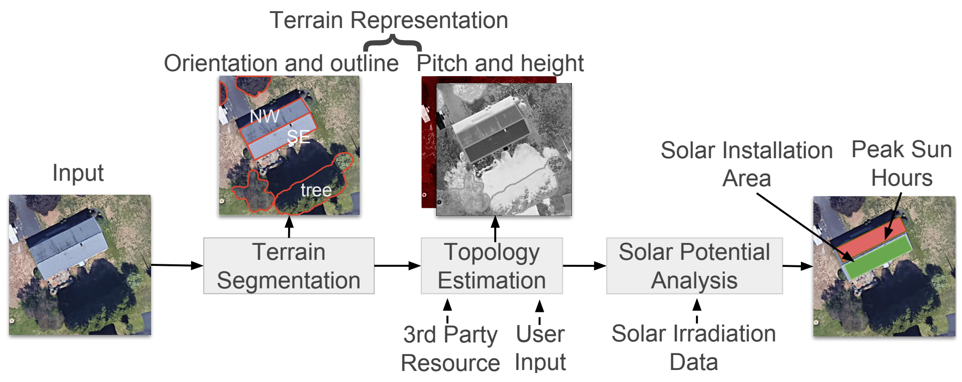


Figure 3.1. An overview of our DeepRoof architecture.

(iii) estimate the solar generation potential of the roof based on its location, weather, and potential trees or other visible occlusions. In addition to widely available satellite imagery, historical solar irradiance data for various locations around the world is available from the US National Renewable Energy Lab (NREL) and public tax records in many countries provide information about the number of floors and height of a building. Consequently, we hypothesize that it is feasible to develop an automated data-driven algorithm that utilizes 2D satellite images of building roofs for solar potential estimation and that such an approach has broader applicability than LIDAR-based methods, including vast swaths of rural areas and smaller cities that are unlikely to be mapped by LIDAR in the near future.

3.2 DeepRoof Design

In this section, we describe our data-driven approach to assess the solar potential of a roof. Our approach, DeepRoof, relies on the key observation that size and structure of roofs are observable in a satellite image, essential for estimating the solar system a building can support. Satellite images also indicate if there are nearby structures such as trees or buildings that can obstruct a roof segment partially or completely. These structures can be identified and used to estimate its overall impact on available solar irradiance incident on the roof. DeepRoof, illustrated in Figure 4.2, uses this insight to compute the solar potential of planar roof segments in a building, and identify suitable locations for installing solar panels. DeepRoof's approach, shown in Figure 4.2, has three key steps:

- **Terrain Segmentation** uses deep vision techniques to create a terrain outline of the input image by identifying all the planar roof segments and trees in the image.
- **Topology Estimation** creates a representation of the topology using the terrain outline from the previous step. We approximate the height of the building and nearby structures using publicly available datasets that may cast shadows on the roof.
- **Solar Potential Analysis** estimates the per-pixel solar potential of the roof using the output from the previous steps and historical solar irradiance data.

Moreover, our algorithm identifies roof locations where panels will receive maximum sunlight, accounting for shade from nearby structures. Below, we describe each step in detail.

3.2.1 Terrain Segmentation

The first step in our pipeline is to determine all the planar roof segments, the orientation of each planar roofs and nearby structures in a satellite image. Extracting the roof segments is useful for determining the rooftop locations where solar panels can be installed. Further, trees and nearby buildings provide locations where these objects may cast shadows on the rooftop, thereby rendering them unsuitable for solar panels. Let $\mathcal{I}$ be the input satellite image of size $w \times h$, in this step, DeepRoof constructs the terrain matrix $\mathcal{T}_A$ of size $w \times h$, where the pixels correspond either to the orientation of the planar roof segments or trees in the image.

Identifying objects in an image at a pixel-level, referred to as semantic segmentation, is a well-researched computer vision problem [19, 83, 90]. Since recent deep learning approaches have outperformed previous vision techniques on segmentation tasks [52, 101], we leverage these methods in our work. In our approach, we use the Feature Pyramid Net-

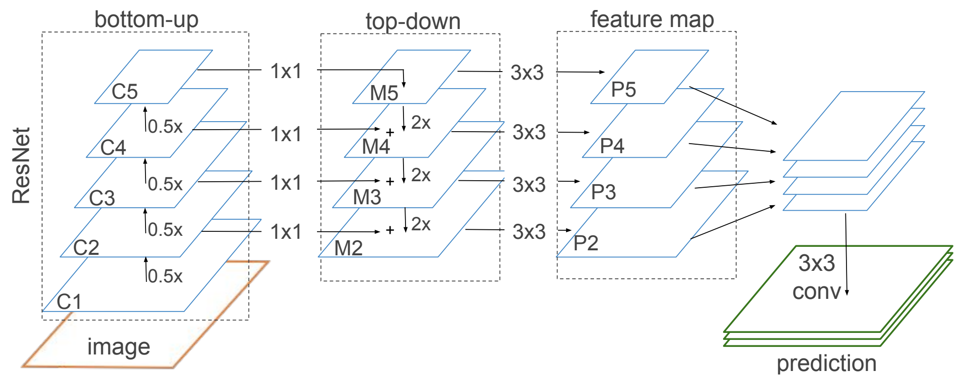


Figure 3.2. Overview of the FPN framework.

work (FPN) to identify planar roof segments and nearby structures. Below we describe the key aspects of FPN, and refer the readers to the original paper [83] for more details.

Figure 3.2 illustrates the architecture of a feature pyramid network for the segmentation task. FPN takes as input an image and extracts features using a convolutional neural network (CNN) architecture (e.g. ResNet [53]) augmented with a pyramid-like structure. As shown in the figure, the bottom-up pathway is augmented with a top-down pathway and lateral connections to build a multi-scale feature pyramid of the input image. Since FPN uses a standard CNN architecture for feature extraction, the network can be initialized with pre-trained weights on ImageNet [74] dataset, which allows our technique to work on relatively small datasets.

In a CNN architecture (e.g., ResNet), the bottom layers learn the low-level features such as edges, and as we move higher up, the top layers learn higher-level semantics of a real-world object such as trees, cars etc. In DeepRoof, the CNN architecture learns the planar roof segments, which are the building blocks to construct the geometry of a roof. In FPN, the ResNet layers are grouped into different network stages $\{C_1, C_2, C_3, C_4, C_5\}$, and the output map from the last layer of each stage is selected as a reference set to create the feature pyramid.

As shown, the lateral connections in the top-down pathway combines the low-resolution and the high-resolution from the convolutional network to create a multi-scale feature

$\{M_2, M_3, M_4, M_5\}$ by applying a 1x1 convolution filter. A 3x3 convolution filter is applied to the output to obtain the final feature maps $\{P_2, P_3, P_4, P_5\}$. Note that the image resolution of each P_i is one-fourth the input image and has 128 channels each. Finally, $\{P_2, P_3, P_4, P_5\}$ feature maps are concatenated to create a layer with 512 channel. We then use two successive 3x3 convolution filters and batch normalization to create a feature map with channels equivalent to the number of output classes for prediction. The output is then up sampled to its original image size using bilinear interpolation and a softmax activation layer is applied to predict the final output.

We now discuss how our approach creates the roof orientation matrix $\mathcal{T}_A$. Our approach views each planar roof segment as an object with azimuth as its label. For instance, a planar roof segment facing north-west is labeled as NW. Similarly, horizontal roof surfaces are labeled as *flat* and we also label tree crowns. The model is trained using this labeled set of images. After our model is trained, the final output contains a per-pixel prediction such that each pixel is labeled with the class of the object. We then use the final output to create the roof orientation matrix $\mathcal{T}_A$, where each pixel label corresponds to roof orientation, trees or background.

3.2.2 Topology Estimation

In this step, we determine the outline matrix $\mathcal{T}_O$ that contains all the planar roof segments of the candidate building. We also describe how we estimate the height and the pitch of the candidate roof. We assume that the outline of the candidate building is available. This is used to determine the roof segments of a candidate building from neighboring rooftops. We note that outline of a building property for a location can be easily obtained from public maps [10]. For example, in a given geographical area, OpenStreetMap provides the outline of all the buildings within a specified area, as well as their addresses [10]. Further, an outline of the candidate building can also be easily obtained as an input through an user interface. In our approach, we use the OpenStreetMap API to obtain the outline of the candidate building in our input image.

In order to recognize the planar roof segments in the orientation matrix $\mathcal{T}_A$, we run the *marching squares algorithm* [91] that identifies all the contours in an image. The marching squares algorithm approximates the line along the edges where the orientation value changes. The contours correspond to a planar roof segment as we expect the orientation to be similar for a given roof segment. Next, for all the contours predicted by our algorithm, we associate a contour with the candidate building if it intersects with the building's outline. This creates an outline matrix $\mathcal{T}_O$, which contains all the planar roof segments of the candidate building.

We then approximate the height and the pitch of the contours identified in the candidate building as well as height of nearby structures. Currently, we rely on third-party sources to create the roof pitch matrix $\mathcal{T}_P$ and the height matrix $\mathcal{T}_H$. We observe that number of floors available in real-estate dataset and Federal Emergency Management Agency (FEMA) guidelines [59] provide reasonable estimate about the height and pitch of the roof, respectively. As part of our future work, this step can be further improved by obtaining these inputs through an interactive interface from users, which can be used to fine-tune the solar potential estimation.

3.2.3 Solar Potential Analysis

We now discuss how we compute the solar potential of a roof and the available area for installing solar panels using the terrain matrix.

3.2.3.1 Solar irradiation on a roof

We note that the solar potential of a roof is the combined potential of all its planar roof segments. We determine the amount of solar irradiance for each planar roof surface in a candidate building for different time of the day in a year, accounting for shade from nearby objects. We now describe how the solar irradiance is computed for a tilted roof surface. The power output of a solar panel depends on the angle sunlight is incident on the PV module, which is maximum when the PV surface is perpendicular to the sun. Thus, the solar irradiance of a roof plane having an orientation $\psi \in \mathcal{T}_A$ and roof pitch $\beta \in \mathcal{T}_P$ are dependent on two components — beam and diffused irradiance. While the beam irradiance S_B is the direct radiation received from the sun, the diffused radiation S_D is received from radiations scattered by particles in the atmosphere. Assuming an isotropic model for diffused irradiance [84], the total solar irradiance of a tilted roof surface is given by:

$$S(\beta, \psi) = \underbrace{S_B \cdot R_B(\beta, \psi)}_{\text{beam irradiance}} + \underbrace{S_D \cdot R_D(\beta, \psi)}_{\text{diffused irradiance}}$$

$$\begin{aligned} R_B(\beta, \psi) &= \cos\alpha \sin\beta \cos(\psi - \theta) + \sin\alpha \cos\beta \\ R_D(\beta) &= \frac{1 + \cos(\beta)}{2} \end{aligned}$$

where, α is the solar elevation angle and varies with the time of the day and θ is the solar azimuth angle and dependent on the latitude of the location. Past values of S_B and S_D are publicly available from various sources [45], and can then be used to compute the total irradiance S for different time of the day in a year.

We consider objects that are roughly within 100 meters from the building for analyzing shadows. We compute the periods when shadows are cast from nearby objects, and subtract the direct radiation S_B from our calculation, i.e., direct sunlight. Note that the diffused irradiance is still received through scattering, and hence it is not ignored. We then compute the annual peak sun hours at a pixel-level, i.e. number of hours with $1kW/m^2$.

3.2.3.2 Solar installation size

Finally, we estimate the number of solar panels that can be installed on the roof. The planar roof segments are already available in the terrain matrices. The general procedure is to pack as many solar panels on each of the planar roof segments in a candidate roof. Our problem is similar to the 2D bin packing, where the objective is to maximize the number of 2D shapes that can fit into a rectangular bin. Here, the planar roof segments represent an irregular shaped bin and the 2D object is the solar panel. Since the computational complexity of 2D bin packing is known to be NP-hard, we use a greedy algorithm to determine the number of panels that can fit on the roof. The greedy algorithm outputs the overall number of panel that fits the roof, and we determine the install capacity by multiplying the total number of panels with the rated power output per panel.

3.3 DeepRoof Implementation

We have implemented DeepRoof as a system to automate the process of solar potential estimation. DeepRoof can operate in two modes: batch and interactive. In the batch mode, our system takes a list of addresses, or GPS coordinates, as input and computes the solar generation potential for each building in the specified list. The batch mode is useful when computing the solar generation potential of all homes in a neighborhood or an entire city. Our system takes the batch of addresses and first computes the GPS coordinates of each address. It then queries a mapping service, currently set to Google Maps in our implementation, to download the satellite rooftop imagery for each location in the list. The batch of roof images is then provided as input to our DeepRoof model, which outputs the planar roof segments. Our system then uses the approach outlined in Section 5.2 to output the per-pixel solar potential as well as available roof area. These results can then be viewed by clicking on each address in the list. Figure 3.3(a) shows the process for computing the solar potential for a batch of buildings.

Our system can also operate in interactive mode via a web interface. In this case, the user specifies an address or the GPS coordinates of a location. Our system then invokes the backend of DeepRoof, which are the same for both the batch and interactive modes. After computing the results, the per-pixel solar potential is overlayed on the satellite imagery. The interactive interface displays the overall potential of the rooftop at a pixel-level. Figure 3.3(b) shows the overall energy potential as well as the available installation area as shown in interactive mode. The bright region indicates the location which receives maximum sunlight.

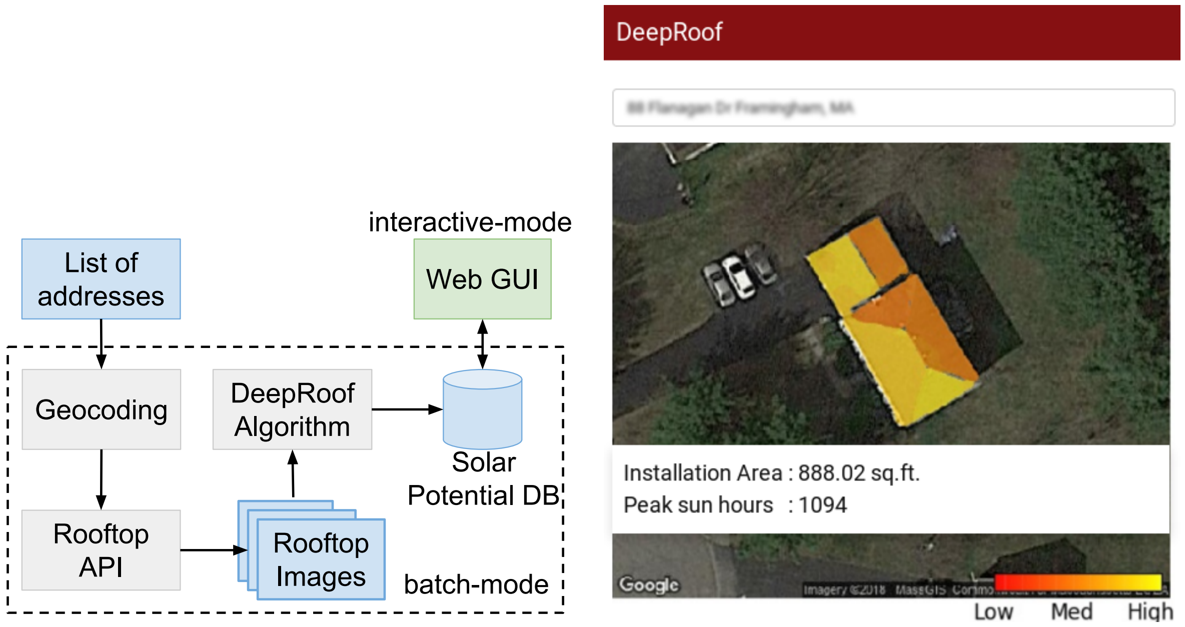
Overall, our system implementation consists of three components: (i) an interface that allows a user to input an address of a building and visualize its overall solar potential (ii) our deep learning model that identifies the planar roof segments and nearby structures in a rooftop imagery and (iii) a set of APIs that implements our approach in Section 5.2 to query Google Maps for rooftop imagery and compute the solar potential. DeepRoof's user interface is implemented using `flask`, a light-weight web framework in `python`. Our DeepRoof's CNN model is implemented using the `keras` library, which invokes Google's Tensorflow in the backend. Finally, our system has the ability to parallelize its TensorFlow computations on a cluster of nodes when processing a batch of buildings—in order to scale the computations to a larger number of homes in a region or city.

3.4 Evaluation Methodology

Below we describe our dataset, experimental setup and metrics used to evaluate our approach.

3.4.1 Dataset

- *Dataset 1:* We collected satellite images from six different cities using Google Maps API (Table 3.1). We labeled the images using a modified VIA annotator tool [41]. Each roof plane in the image was annotated (including adjacent buildings) and assigned an orientation angle from the north, or labeled as a flat roof. However, we did not label some of the small roof segments, where solar panels cannot be installed. We also labeled nearby trees with visible tree crowns. Apart from the satellite images, we also downloaded the outline and height of the building from



Installation Area : 888.02 sq.ft.
Peak sun hours : 1094

(a) (b)

Figure 3.3. (a) Key components in DeepRoof's implementation. Our approach supports two modes: batch and interactive mode (using a Web GUI) (b) Screenshot of the web interface that help visualize the solar potential of a building.

OpenStreetMap API [10]. We augmented the height of the building with third-party real-estate datasets in cases where the height was not available.

- *Dataset 2:* For our city-scale case study, we selected a total of 1982 buildings from the city of Framingham, MA (see Table 3.2). The dataset contains real-estate information such as number of floors, roof type (e.g., hip and gable). We downloaded the satellite images and building outline from Google Maps and OpenStreetMaps. Further, we also collected the solar installation area and available sun hours from Google's Project Sunroof to compare our approach with a LIDAR-based approach. For estimating the peak sun hour, we used the solar irradiation data from National Solar Radiation Database (NSRDB) [45]. The dataset contains the diffused and direct solar irradiation as well as the azimuth and elevation of the sun for a given location at

Table 3.1. Dataset 1: Summary of the labeled roof dataset.

City	#images	#buildings	#roof segments
Framingham, MA	279	1161	1722
Pinellas Park, FL	122	944	2121
Fresno, CA	43	69	171
Seattle, FL	8	46	158
Denver, CO	7	44	90
Indianapolis, IN	5	10	50
Total	464	2274	4312

Table 3.2. Dataset 2: Key characteristics of the unlabelled roof dataset used in our city-scale case study.

roof types	#buildings	#floors	land area(acres)
Gable, Flat Hip, Complex hip	1982	1 - 6	0.031 - 2.92

a granularity of 1 hour. Our dataset is available for download at UMass Trace Repository (`http://traces.cs.umass.edu/index.php/Smart/Smart`).

3.4.2 Experimental Setup

We augmented our dataset by rotating the images at different angles. Further, we categorized the orientation directions (0°to 360°) to one of the 16 orientation (i.e., N, NNE, NE, etc.), assigning each roof segment to its closest orientation. In addition to using FPN in DeepRoof, we used other baseline segmentation models — namely UNet [120] and MaskRCNN [52]. Further, the segmentation models were trained using two different CNN architectures namely ResNet 50 and ResNet101, resulting in a total of six models. Since FPN, UNet, and MaskRCNN can use ResNet architecture for feature extraction, pre-trained weights from ImageNet were used as per the literature [52, 74].

3.4.2.1 Training and model selection

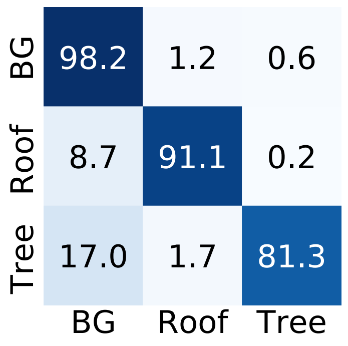
We split our dataset into three disjoint sets: train (60%), validation (20%) and test (20%). The datasets are split before augmenting the dataset to prevent the model from seeing the hold-out set. To prevent overfitting, we trained our models until their performance doesn't improve further on the validation dataset. Further, we used stochastic gradient descent optimizer, with a learning rate of 0.001 and a momentum of 0.9. For training the model, we ran our neural network model for 240k iterations and reduced the learning rate by a factor of 10 at the 100k and 160k iteration. We report our result on the unseen test dataset.

3.4.3 Metrics

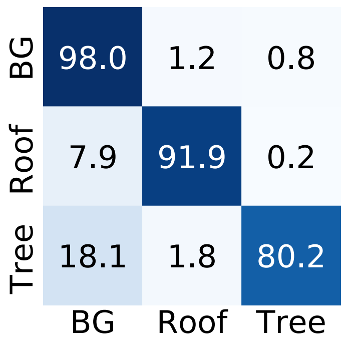
We note that standard error metrics such as the mean absolute error are not an ideal evaluation metric for capturing the performance of the model in predicting orientation. For instance, if the predicted orientation is NNW (337.5°) and the ground truth orientation is N (0°), the error in prediction is 22.5°. However, mean absolute error will report an error of 337.5°. Thus, we introduce *mean orientation error* (MOE) as a metric to capture the per-pixel error between the predicted and the actual azimuth angle.

$$mean\_orientation\_err = \frac{1}{M}\sum_{i}\frac{\sum_j p_{ij} * degree\_separation(o_i, o_j)}{t_i}$$

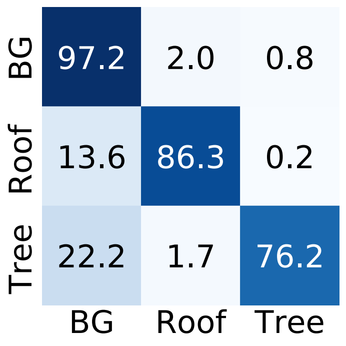
where, M is the number of classes (i.e., azimuths), o_i is the azimuth angle, p_{ij} denotes the number of pixels of azimuth j classified as azimuth i, and t_i is the total number of pixels in class i. Finally, the $degree\_separation$ is a function that returns the azimuth angle difference between the two azimuths. The value of MOE is between 0°(perfect prediction) and 180°(opposite prediction).



(a) DeepRoof (101)



(b) UNet (101)



(c) MaskRCNN (101)

Figure 3.4. Normalized confusion matrix of roof classifcation.

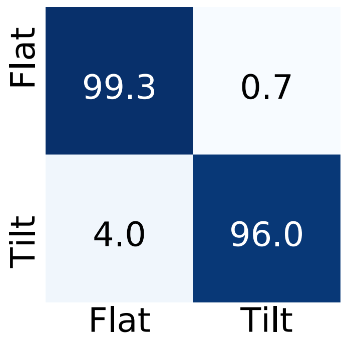
3.5 Experimental Results

In this section, we validate our results with the ground truth including a LIDAR-based approach. We also validate our output with solar experts and show that our data-driven approach can be used to analyze a city-scale dataset.

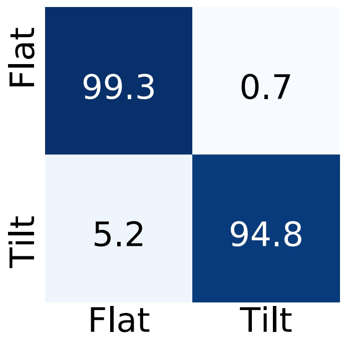
3.5.1 Roof Classification

We first evaluate the performance of DeepRoof in identifying roofs since these locations are potential sites where solar panels can be installed. To do so, we classify a pixel as a roof if the segmentation model predicts an orientation for the pixel or has labeled the roof as flat. We present our results for all the segmentation models used in our *terrain segmentation* step. Figure 3.4 shows the normalized confusion matrix of roof classification across all models using ResNet101 architecture on Dataset 1. The values in the confusion matrix are normalized by the number of pixels in each class. We observe that the ResNet50 architecture yields a lower true positive rate[2] compared to ResNet101 (not shown in the figure). This is because the deeper network enables it to learn the feature subspaces better. In particular, we observe that the true positive rate of ResNet101 reaches 91.1%, 91.9% and 86.3% for DeepRoof, UNet and MaskRCNN respectively. Further, the results of DeepRoof

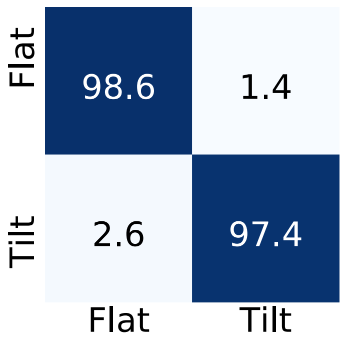
[2]True positive rate is the ratio of true positives identified from all the positive cases.



(a) DeepRoof (101)



(b) UNet (101)



(c) MaskRCNN (101)

Figure 3.5. Normalized confusion matrix of slope type.

and UNet are comparable with the difference in the true positive rate within $\pm 1\%$ of each other in classifying roofs and nearby structures.

Next, we evaluate the performance of DeepRoof in differentiating between a flat roof and a pitched roof. We classify the slope of a pixel as tilted if the model predicts an orientation for the pixel else we classify it as a flat roof segment. Figure 3.5 shows the normalized confusion matrix of slope type prediction on Dataset 1. Both ResNet50 and ResNet101 shows high accuracy in differentiating between the types of slope, i.e., flat or tilted, with accuracy $> 98\%$ and $> 93\%$ for flat and pitched roofs respectively. As again, ResNet101 architecture yields better result compared to ResNet50 owing to the deep architecture. Although MaskRCNN performs relatively poor in classifying roofs, among the pixels classified as roofs, it has a high true positive rate of $> 97.4\%$ in differentiating between flat and pitched roof. We also note that DeepRoof has a higher true positive rate compared to UNet and yields better results in identifying slope types. In particular, DeepRoof has a true positive rate of 96% compared to UNet's 94.8%.

3.5.2 Roof Orientation

Figure 3.6 shows several examples of the segmentation output of different roof types for all models. Each color in the figure depicts a roof plane and the orientation of the

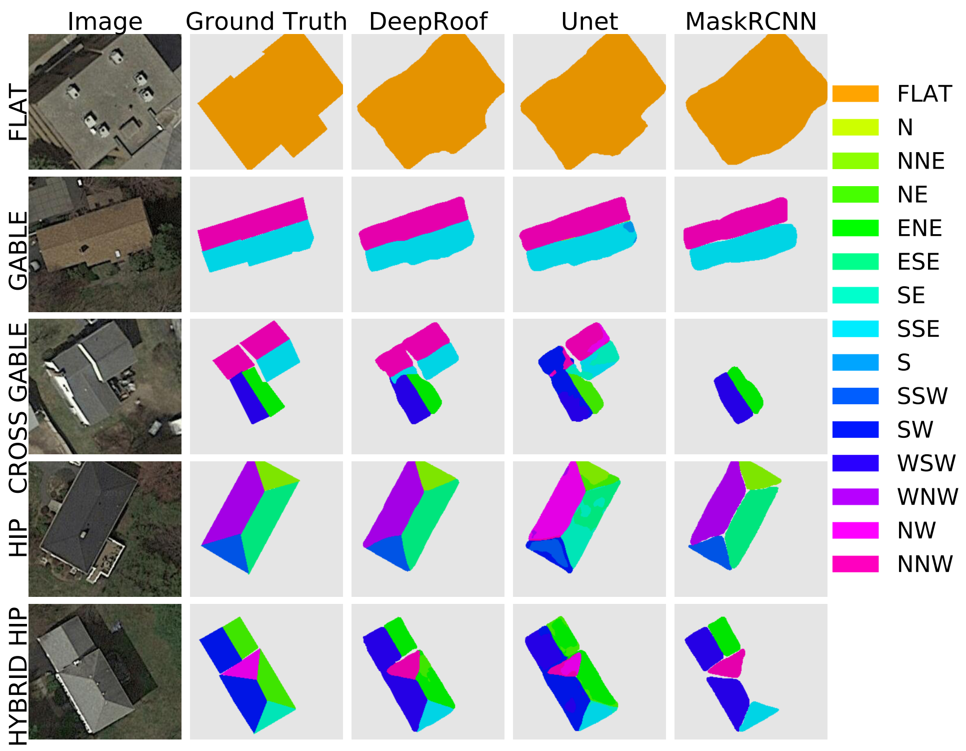


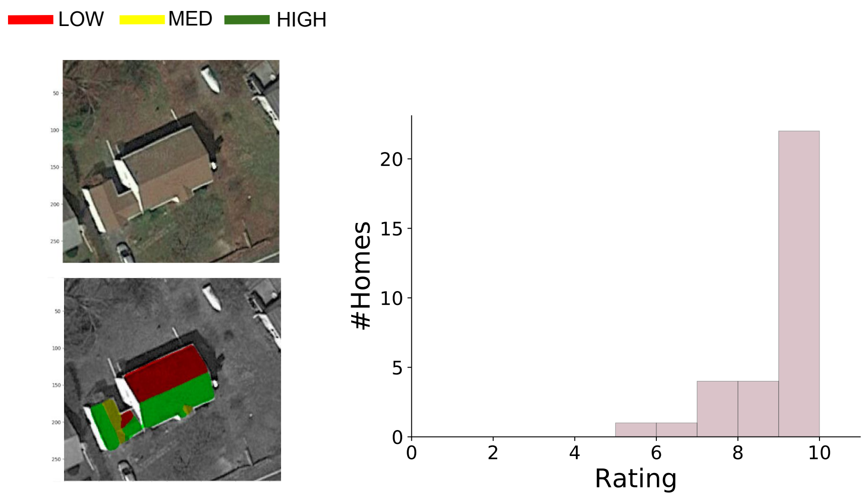
Figure 3.6. Segmentation output on different roof types.

roof plane. The figures illustrate that the models can identify the planar roof segments and also determine their orientation that are visually close to the ground truth. ResNet101 shows better visual improvements in segmentation output compared to ResNet50. Also, we observe that MaskRCNN fails to identify some of the roof segments, which corroborates the lower true positive rate discussed above. As seen in the figure, both DeepRoof and UNet shows a close resemblance to the ground truth, even when the buildings have different geometry.

Next, we evaluate the performance of predicting the orientation of the rooftop. Table 3.3 summarizes the mean orientation error on Dataset 1. A lower mean orientation error is better, where a zero value indicates that the predicted pixel orientation matches the ground truth. Similarly, a mean orientation error of 180° indicates that the pixels were predicted opposite to the ground truth orientation. Overall, we observe that all the models yield an MOE of less than 12°. Since the orientation labels are 22.5° apart, it indicates that

Table 3.3. Mean orientation error of the predicted roofs for different architectures on Dataset 1.

Backbone	DeepRoof	UNet	MaskRCNN
ResNet50	10.63	10.94	11.43
ResNet101	**9.3**	11.94	9.37



(a) (b)

Figure 3.7. (a) Sample image provided to experts for validation (b) Average rating distribution of the response on a scale of 1 to 10 (10 being the highest).

the models are able to predict a rooftop's orientation correctly in most cases. It is interesting to note that the models are able to learn the spatial relationship between the planar roof segments and predict their orientation. As shown, DeepRoof yields the lowest MOE compared to other segmentation models with MOE of 10.6 and 9.3 using ResNet50 and ResNet101 respectively.

3.5.3 Expert Validation

For this experiment, we asked two independent solar experts with experience in installing PV panels to rate our solar estimation output from DeepRoof. Our objective with this study was to address the following questions (i) How well does DeepRoof estimate the solar potential of each planar roof segments? (ii) Are there locations on the roof that our approach fails to identify as possible locations where experts would consider installing solar panels?

To answer the above questions, we selected only buildings with pitched roofs from the test dataset and omitted flat roofs that are relatively easier to estimate solar potential. We highlighted all the areas on the roof where solar panels can be installed. Further, we discretized the sunlight received in each planar roof segment into high, medium and low, and presented the image to the experts for analysis. We asked them to rate DeepRoof's result on a scale of 1 to 10 (10 being the highest rating). We also asked them to consider nearby structures such as trees that may affect the solar potential while rating DeepRoof's output. In total, we randomly selected 30 images from the test Dataset 1 for our evaluation. Figure 3.7 (a) shows a sample image from the study that was provided to the experts for validation. The image on the top shows the ground truth, and the image on the bottom shows the output estimated by our algorithm.

Figure 3.7 (b) shows the average rating distribution of the expert's response to DeepRoof's output. The graph shows that DeepRoof estimates the solar potential with high accuracy, as seen from the high rating received for most homes. Overall, we observe that both the experts gave a rating of 8 and above to 22 of the 30 homes. For these homes, DeepRoof not only predicted the orientation correctly but also considered shade from nearby trees to estimate the solar potential. For homes that received a rating lower than 8, in most cases, DeepRoof failed to identify the surrounding trees. We note that these images had fall trees without leaves and, thus, was classified as background pixels. On an average, the experts gave a rating of 8.8 for a typical home. We also received responses on whether DeepRoof identified all the roof segments where solar panels can be installed. Both the experts validated that DeepRoof didn't miss out any candidate roof segment suitable for solar installation.

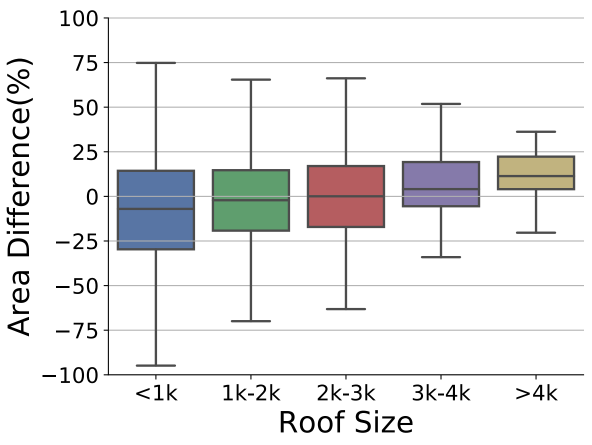


Figure 3.8. Difference in installation area estimated by Sunroof and DeepRoof for different roof sizes.

3.5.4 Comparison with a LIDAR-based Approach

Google's Sunroof project, a LIDAR based approach, estimates the solar potential of a roof as follows. For a given address, the Sunroof provides the total solar installation area and a pixel-level sunlight available on the roof. These estimates are calculated using LIDAR and NREL's solar irradiance data. To compute the available solar installation area, Sunroof uses a greedy algorithm that maximizes the number of solar panels that can fit on a planar roof segment [114]. Since the pixel-level solar potential is not accessible via Sunroof, we cannot meaningfully compare the results, and hence we only compare the solar installation area of the roof.

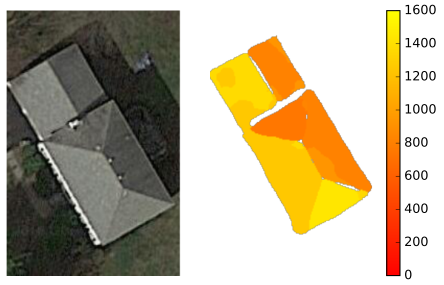
In our approach, we select a fixed solar panel size of 250W[3], and align it based on the orientation and pitch of the roof segments. Our greedy algorithm then uses the panel dimensions as input to analyze the number of panels that can fit on each of the predicted planar roof segments. We ran our algorithm on the test dataset in Dataset 1 and all the buildings in Dataset 2. We had a total of 2073 buildings after combining the datasets.

[3]The standard dimensions of a 250W PV panel is 1m$\times$1.65m. Google Sunroof project also uses a 250W panel to analyze the potential [114].

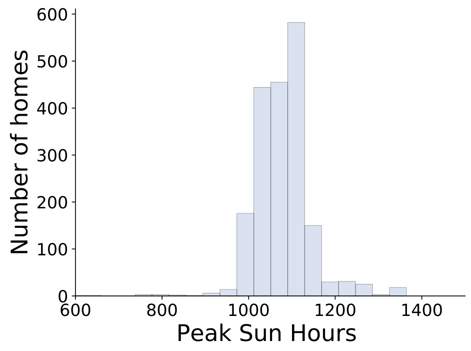
Figure 3.8 plots the distribution of percentage difference in solar installation area predicted by Sunroof and DeepRoof for different roof sizes. A negative percentage difference indicates under-prediction and a positive difference indicates over-prediction. As seen in the figure, the median percentage difference between Sunroof and DeepRoof for different roof sizes varies from -7% to 11%. The median percentage difference is 0.5% for roof size between 2000 to 3000 sq.ft. This indicates that on an average DeepRoof's estimation tends to be close to Sunroof's estimated area. We also note that the variance decreases with an increase in roof sizes. This is because planar roof segments in small rooftops are comparatively difficult to identify. However, we note that for the first and the third quartiles are within 25% of Google Sunroof's estimate. Thus, DeepRoof estimates the solar installation area using rooftop images that are close to LIDAR based approaches.

3.5.5 City-scale Solar Estimation

We first provide a breakdown of the time it takes for DeepRoof to estimate the potential of a rooftop image. The computation-heavy tasks in DeepRoof involve semantic segmentation of the image, estimating the solar potential and the solar installation area. We note that the DeepRoof model achieves an inference time of 169 ms on an NVIDIA M40 GPU per rooftop imagery. Separately, depending on the number of planar segments identified and nearby structures, calculating the solar potential and the installation area takes on an average 3 to 5 seconds to complete on a single machine. Assuming 5 seconds of processing time per image, a single machine running DeepRoof can process 10,000 buildings in approximately 14 hours. Since processing rooftop images is an embarrassingly parallel, a server cluster can be deployed to speed up the process further. Thus, DeepRoof can easily scale to millions of homes. We now analyze the output of DeepRoof on the citywide buildings in Dataset 2 and present our results below.



(a)



(b)

Figure 3.9. (a) Per-pixel peak sun hours of a building. The bright colored region indicates higher solar potential. (b) Average peak sun hours distribution in Dataset 2.

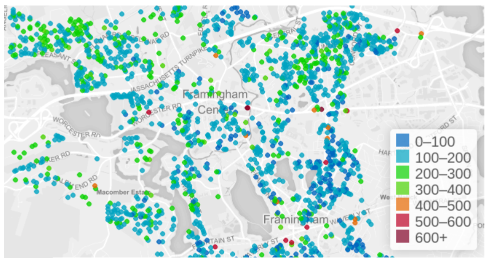
3.5.5.1 Peak sun hours

Figure 3.9(a) shows the rooftop image and the peak sun hours received by each pixel in a sample building. As seen in the image, our technique can identify south-facing and south-west facing roofs with higher solar potential (depicted by the brighter yellow color). Similarly, north and northeast facing roofs have lower energy yield. Thus, DeepRoof's output can provide custom insights to homeowners on the optimal placement of panels.

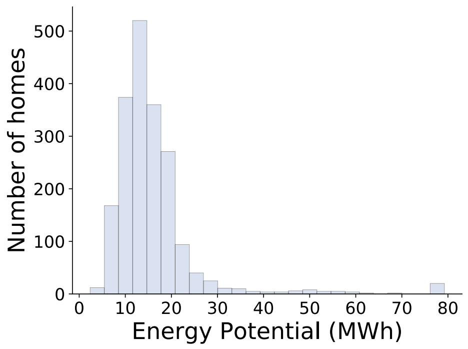
Figure 3.9(b) shows the distribution of the average peak sun hours of all the building in Dataset 2. We observe that the peak sun hours range from 607 to 1471.7 hours. The variation in the average peak sun hours is due to the different orientation and pitch of the roof segments along with the shadows caused by nearby structures. Further, the median peak sun hours available is 1077.95 hours. We observe that all but 5 locations receive a minimum of 800 hours of peak sunlight — indicating significant solar potential among these buildings.

3.5.5.2 Energy generation potential

We compute the energy generation potential assuming the entire roof can generate electricity [115]. Figure 3.10 (a) shows the spatial representation of the overall solar potential in MWh of the buildings and Figure 3.10 (b) shows the solar energy distribution of all the



(a)



(b)

Figure 3.10. (a) Spatial representation of the annual solar energy generation potential (b) Energy potential distribution.

buildings. As seen in the figure, the median energy potential of a home is approximately 14.03 MWh. Assuming the national US average energy consumption of 10.9MWh for a typical residential customer [8], our results indicate that most households can become completely energy self-sufficient using rooftop solar. We also find that on an average, except for homes that have significant foliage or obstruction, the total annual solar production of all the buildings to be 31248.3 MWh, which is at least 1.44 times the annual energy needs of a typical home.

3.6 Related Work

There has been significant work on estimating the global irradiance at ground level [26, 28, 34, 48]. Previous studies have used satellite data on the earth-atmosphere system and ground pyranometer to measure the variability in solar irradiance for a location [34]. These provide reasonable estimates on how much sunlight is available for a location over a given period [45]. Prior work has also studied the sunlight available on tilted surface [48]. We use these estimation models in our work to study the solar potential of a roof.

Automatically estimating a roof's solar potential requires identifying buildings and trees. Various methods have been proposed to automatically identify buildings in satellite and aerial images [57,107,122,142]. Most techniques rely on LIDAR based approaches for detecting and modeling buildings [142]. Some of these studies also detect roof planes of a building to reconstruct a 3D model [68, 121, 134]. Separately, there have been studies that combine street and aerial images to detect street trees and identify its species [140]. However, most of the existing approaches use LIDAR data for modeling roofs and extracting its geometry. In contrast, our approach provides an alternative to LIDAR-based approaches and uses satellite images for solar potential estimation.

Recently there has been significant interest in estimating the potential of roofs for installing solar panels [18,67,75,108,114]. While some studies have proposed manual methods [75], others have proposed automated methods for estimating potentials [38,114]. Manual estimation requires expensive instruments [116] and professionals to reasonably assess a roof's suitability. On the other hand, automated approaches require LIDAR data, which are not readily available for all cities or remote locations [94]. Unlike prior work, we use satellite images that are readily available from mapping services. Recent advances in deep vision techniques make detection of objects in aerial images feasible [93, 101]. Our work leverages the state-of-the-art vision techniques to approximate both the orientation and the roof's geometry using only rooftop images and publicly available irradiance datasets. Thus, our approach provides a scalable approach for estimating solar potential in locations where LIDAR is not available.

3.7 DeepRoof Summary

Solar potential estimation of a roof can substantially benefit homeowners deciding to adopt solar. In this chapter, we proposed DeepRoof, a data-driven approach to estimate the solar potential of a roof using satellite images. We extensively evaluated our approach using available ground truth roof dataset having diverse roof shapes and sizes. We also validated our results with solar experts and compared DeepRoof's output to a LIDAR-based approach. Our results showed that DeepRoof can accurately extract the roof geometry such as the planar roof segments and their orientation, and achieved a true positive rate of 91.1% in identifying roofs and a low mean orientation error of 9.3°. Further, we also analyzed the solar potential of a city-scale dataset and showed that installing solar panels can lead to energy self-sufficiency in these homes.

CHAPTER 4

DISTRIBUTED RATE CONTROL FOR ENERGY SYSTEMS

In this chapter, we introduce the notion of *solar rate control* to regulate the surplus power being fed into the electric grid. We propose a decentralized algorithm that enables smart solar arrays to self-regulate its power output in a grid-friendly fashion to prevent congestion in the grid. We show how solar rate control enables higher solar penetration and can reduce the variability from solar in the grid.

4.1 Motivation

Conventional wisdom holds that there is a limit to the amount of the solar penetration in the grid, i.e., the maximum fraction of demand satisfied by solar power that the grid can handle. Since solar generation is intermittent, utilities must offset any large increases or decreases in solar output by decreasing or increasing output from other sources to compensate. However, with high penetration and variable weather conditions, fluctuations in aggregate solar output may occur too quickly to be offset from mechanical generators to offset, resulting in supply-demand mismatches. Consequently, current regulations strictly limit the number and size of grid-connected solar deployments that use net metering.

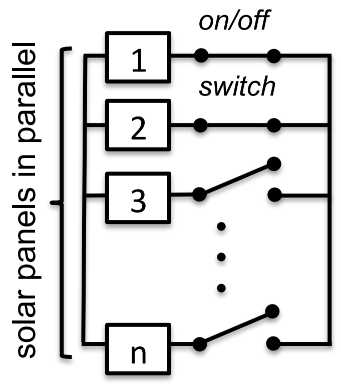
The problem faced by the grid is reminiscent of problems faced by the early Internet. Early transport protocols for network data transmissions did not include congestion control and allowed users to inject data into the Internet at arbitrarily high rates. Since network capacity was fixed, too many users sending data at excessively high rates drove the network to near congestion collapse. The imminent threat of congestion collapse led the design of TCP, a transport protocol that uses congestion and rate control to gracefully adapt sending rates upon detecting congestion to maximize aggregate goodput, prevent congestion collapse, and fairly share the Internet's available bandwidth among active flows [66].

Today's "dumb" electric grid and solar arrays are akin to the early Internet—it permits grid-tied solar systems to generate and transmit large amounts of power into the grid without regard for its current state and available excess transmission capacity. For example, on a sunny day, the cumulative output of solar deployments throughout the grid could cause a supply-side surplus that exceeds demand and causes grid "congestion". In contrast, on a cloudy day, the grid may be able to accept additional power from many solar systems that are currently forced off-grid due to strict caps.

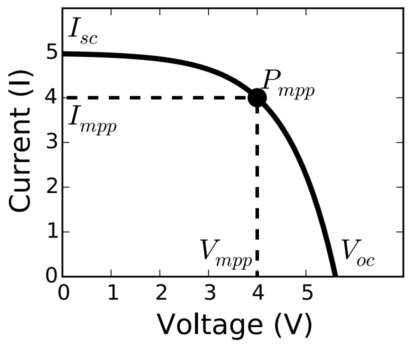
Instead of introducing strict caps to avoid imbalance, we propose an alternate approach wherein smart solar arrays are capable of self-regulating their output in a grid-friendly fashion. Our smart solar arrays can control their generation rate by backing off when supply exceeds demand (more precisely, the aggregate solar output is greater than some threshold), and increasing the rate when needed. The idea is similar to rate control of network flows in TCP, where sources back off when there is congestion in the network and increase the rate when network capacity is available. We argue that solar rate control has the potential to permit a much larger solar capacity to be installed, thereby increasing solar penetration. Solar rate control also provides grid operators with an additional control "knob" when continuously matching supply and demand.

4.1.1 Why is Solar Rate Control Feasible?

Interestingly, practically every solar panel today, as well as solar arrays, have the ability to control their solar output. At an array scale, this can be trivially done in discrete steps by dynamically connecting and disconnecting individual panels. Figure 4.1(a) shows an array



(a) Discrete Rate Control



(b) Continuous Rate Control

Figure 4.1. Rate control approaches in solar panels.

where panels are connected in parallel and a program switch can be used to dynamically disconnect k out of n panels, thereby providing *discrete* control[1].

Even at the granularity of a single panel, it is possible to control the output of the panel. The output of photovoltaic solar is given by its *I-V curve* depicted in Figure 4.1(b). Given a certain amount of solar irradiance, the I-V curve shows all possible operating points of the panel for that solar irradiation. Specifically, any voltage on the curve can be chosen and the panel will then produce the corresponding current. Since power is defined as the product of current and voltage i.e. $P = I \cdot V$, the panel actually can provide a different power output based on the choice of voltage. In general, panels operate at a voltage V at the knee of the curve, which yields the maximum output. The point where the panel generates the maximum power is called the maximum power point (MPP).

However, there is no particular reason to operate a solar panel at its maximum power point. It is possible to pick other values of V [55, 127], using a buck-boost converter, which are akin to "backing off" and producing an output less than the output at MPP. Thus, any solar panel's output can be altered by changing its operating voltage. Our smart solar panels are built on this idea. We assume the presence of software controls that enable the

[1]Typical rooftop solar installation is 5kW (20 panels) [77]. Thus, we can control the power output in 5% (250W) increments.

output to be lowered below the MPPT, and thus control the power output of the panel. This mechanism enables *continuous* rate control to limit the power injected to the grid.

Modern inverters are beginning to offer more configurability and in the long run, we expect them to expose rate control mechanisms [77, 127]. Both the discrete control above, or the continuous control can be used to regulate the rate. Given smart solar panels connected to the grid, our goal is to control the solar output in order to provide higher control over distributed solar-powered systems.

4.2 Solar Rate Control

The problem of controlling solar power is similar to the rate control problem in communication networks [71, 92]. This body of work proposes an optimization framework for determining the rates allocated to different network flows given network capacity constraints. These ideas from network rate control were first applied to the power grid scenario by Ardakanian et. al. albeit in a different context—controlling the rate of electric vehicle charging [17]. In our case, we use these principles from networking [71, 92] to address the problem of *solar rate control*. Next, we present the problem of solar rate control. We then outline our design objectives and assumptions.

4.2.1 Centralized Problem

We first formulate our solar rate control problem as a centralized optimization problem. The centralized problem requires knowledge of the load at the feeders/transformers level and the current generation rate of individual solar installations in order to compute the solar allocation rate while adhering to certain grid constraints. The allocation rate should not only maximize the individual user's output but also maximize the overall grid utilization.

Intuitively, we want to limit the aggregated distributed solar generation to a certain *capacity*. This leads to the problem of apportioning the capacity among different solar arrays to determine the generation rate for each array. Note that the grid demand and solar generation output is time varying, and may change over the day. Thus, at each time t, the optimization problem needs to recompute the capacity and the allocation rate for each solar array . For simplicity, we describe the optimization formulation for a single time step.

We consider a distributed grid transmission network with a set of transmission feeders F, transformers K and smart solar arrays S. Electric power is transmitted from the power station to substations at high voltages. At the distribution substation i.e. low voltage (LV) feeder, voltage is stepped down and distributed to transformers, wherein it is further stepped down before it is transmitted to residential users. Thus, the smart solar arrays are connected to the LV feeder via a transformer. Formally, we say that the smart solar array s is connected to a LV feeder f, if $s \in S(k)$ and $f = F(k)$, where k is the transformer located in between s and f. We model the key characteristics of our problem as follows:

- *Transformer constraint:* Power flow at the transformer level can be bi-directional and the maximum power flow at the transformer is dependent on the transformers rating $\mathcal{C}$. The transformer rating is between $-\mathcal{C}$ to $\mathcal{C}$ kVA, where the negative sign indicates reverse power flow from the transformers to the feeders. Usually, the transformers are right-sized to ensure that the load at the transformers does not exceed its rating. However, high solar penetration in residential homes may cause reverse power flow and the following constraint must be satisfied to maintain grid stability.

$$\sum x_s \leq load_k + \mathcal{C}_k \quad \forall s \in S(k) \text{ and } k \in K \tag{4.1}$$

  where x_s is the solar generation rate of the smart solar array $s \in S(k)$ and $load_k$ is the aggregate load from the residential homes in transformer k.

- *Feeder constraint:* Most residential LV feeders are not equipped with infrastructure to allow reverse power flow i.e. electricity does not flow from an LV feeder to a medium voltage transmission line and thus obeys the following constraint

$$\sum x_s \leq load_f, \quad \forall s \in S(f) \text{ and } f \in F \tag{4.2}$$

where $load_f$ is the load at the feeder f, and $S(f)$ are the smart solar arrays in feeder f.

- *Grid capacity constraint:* The grid utility may cap solar output to reduce variability in grid or due to legislative reasons [12]. The aggregate solar generation output may be capped at a fraction of the aggregate grid demand

$$\sum x_s \leq capacity, \qquad \forall s \in S \tag{4.3}$$

where $capacity$ is defined as a fraction of the total power demand at the grid level.

- *Solar PV constraint:* The maximum power generated by a solar panel lies in the interval $[0, x_s^{mppt}]$, where x_s^{mppt} is the MPPT rate of the solar PV and is defined as

$$0 \leq x_s \leq x_s^{mppt} \quad \forall s \in S \tag{4.4}$$

Note that (4.1), (4.2) and (4.3) can be combined and represented as a single inequality

$$R\mathbf{x} \preceq \mathbf{c} \tag{4.5}$$

where $R \in \mathbb{R}^{m \times n}$ matrix, with m combined constraints from (4.1), (4.2) and (4.3) and n smart solar arrays; $\mathbf{x} \in \mathbb{R}^{n \times 1}$ vector is the set of smart solar arrays; $\mathbf{c} \in \mathbb{R}^{m \times 1}$ vector captures the capacity constraints; and finally, $\preceq$ represents the generalized inequality of vectors. R can be represented as:

$$R_{is} = \begin{cases} 1 & \text{if } s \in S \text{ is present in the } i^{th} \text{ constraint} \\ 0 & \text{otherwise} \end{cases}$$

Remember our goal is to take some aggregate capacity and apportion it among individual solar installations. Thus, our objective is to maximize the total utility of the individual smart solar arrays $U_s(x_s)$; subject to constraints (4.4) and (4.5). To summarize, our optimization problem can be defined as:

$$\begin{aligned} \max_{x_s} \quad & \sum_{s \in S} U_s(x_s) & \\ \text{subject to:} \quad & R\mathbf{x} \preceq \mathbf{c} & \text{and,} \\ & 0 \le x_s \le x_s^{mppt} & \forall s \in S \end{aligned}$$

We refer to the above problem as the *primal* problem. We assume that the utility function is strictly concave, increasing and twice differentiable. Since each constraint is convex, a unique maximizer exists and solving the optimization problem generates a solar allocation that is optimal.

The centralized optimization problem discussed earlier is mathematically tractable. However, solving the optimization necessitates a prohibitively high communication overhead, as it requires two-way communication infrastructure between the smart solar arrays and the control center. Moreover, an increase in solar array deployments will increase the coordination overhead between the control center and smart solar arrays to compute the solar allocation rate. Hence, we formulate a distributed approach to mitigate some of the issues in the centralized approach.

4.2.2 Design Objectives

4.2.2.1 Maximize utility of end-users and the grid

Solar panels are *net-metered* and the amount of electricity supplied to the grid earns residential customers billing credits. To model the benefit of net metering, we attribute a utility function $U_s(x_s)$ to the user for generating solar output at rate x_s. From the user's perspective, each user would like to maximize its own utility. However, from the grid perspective, the utility function should also maximize the overall utilization of the network.

We explore two utility functions, *non-weighted* and *weighted*, described in Kelly et. al. [72], which maximizes both the grid and the user's utility function. The non-weighted utility function, $U_s(x_s) = \log(x_s)$, provides equal utility regardless of the size of the solar panel. Since, $\log(x_s)$ is a strictly increasing function, an increase in solar output x_s denotes an increase in the *utility*. On the other hand, the weighted utility, $U_s(x_s) = w_s \log(x_s)$, provides additional benefit to users for installing larger solar panel, where weight w_s represents the weight corresponding the size of the solar panel. Both the utility functions are increasing, strictly concave and continuously differentiable.

4.2.2.2 Fairness in solar rate allocation

We are interested in an allocation that is fair to the user. Here, we use a utility function that provides *proportional fairness* and *weighted proportional fairness*. Any feasible allocation vector $\mathbf{x}$ is proportionally fair, if for any other feasible rate vector $\mathbf{y}$, the aggregate of proportional change is non-positive i.e.

$$\sum_{s \in S} \frac{y_s - x_s}{x_s} \leq 0 \tag{4.6}$$

Similarly, any feasible allocation vector $\mathbf{x}$ is *weighted proportionally fair*, if for any other feasible vector $\mathbf{y}$ the following holds.

$$\sum_{s \in S} w_s \frac{y_s - x_s}{x_s} \leq 0 \tag{4.7}$$

As shown in [71], the logarithmic utility function discussed above achieves proportional fairness and the allocation vector obeys the fairness property (4.6). In addition, it is shown that proportional fairness is *pareto optimal*, since increasing a user's allocation will decrease allocation of another user.

4.3 Distributed Rate Control

The centralized problem discussed in the previous section has three key drawbacks in practice. First, it requires full knowledge of the maximum generation output (MPP) of all grid-connected smart solar arrays. Second, the control center requires knowledge of the grid's network topology in order to compute the solar rate. Third, a two-way communication needs to be established between the control center and smart solar arrays for controlling the solar rate. Hence, we reformulate the centralized optimization problem to an equivalent distributed optimization problem, which can then be solved locally by smart solar arrays and eliminate some of the disadvantages of the centralized approach. In contrast to the centralized approach, the distributed algorithm does not require knowledge of the grid's network topology and eliminates the need to share local information.

4.3.1 Dual Decomposition

We use the *dual decomposition* approach to divide the centralized optimization problem into smaller subproblems. Note that the optimization problem has a coupling constraint (4.5), which prevents solving each subproblem independently. Clearly, without the coupling constraint, each user can maximize its utility independent of each other, thus maximizing the aggregate objective function. Below, we present the Lagrangian dual problem, which relaxes the coupling constraint using *control prices* (Lagrangian multipliers) and thus allows solving the problem as independent subproblem.

We define the Lagrangian of our optimization problem and consider *control* prices λ to relax the coupling constraint

$$
\begin{aligned}
\mathcal{L}(x,\lambda) &= \sum_{s\in S} U_s(x_s) - \sum_{l\in L} \lambda_l (y_l - c_l) \\
&= \sum_{s\in S} (U_s(x_s) - x_s q_s) + \sum_{l\in L} \lambda_l c_l
\end{aligned}
$$

where l denotes the row number and L is the total number of constraints in matrix R; and

$$y_l = \sum_{s \in S} R_{ls} x_s \qquad \forall l \in L$$

$$q_s = \sum_{l \in L} R_{ls} \lambda_l \qquad \forall s \in S$$

Thus, the *Lagrangian dual* problem can be formulated as:

$$\mathbf{D}(\lambda) : \min_{\lambda \geq 0} \quad \sum_{s \in S} V_s(x_s, \lambda_s) + \sum_{l \in L} \lambda_l c_l \tag{4.8}$$

$$\text{subject to:} \quad \lambda_l \geq 0 \qquad \forall l \in L \tag{4.9}$$

where,

$$V_s(x_s, \lambda_s) = \max_{0 \leq x_s \leq m_s} (U_s(x_s) - x_s q_s) \qquad \forall s \in S \tag{4.10}$$

As discussed earlier, the utility function (U_s) is strictly concave. Since the sum of concave function U_s is concave and the linear constraints are concave, strong duality holds i.e. the primal and the dual solutions are equal. Hence, solving the dual problem solves our original primal problem.

We solve the dual problem using the gradient projection method. Note that for a fixed λ, the dual problem is completely *separable* in x_s and each subproblem in x_s can be maximized independently by each smart solar array using (4.10). In particular, for a given price λ, a unique maximizer exists that maximizes (4.10). Since the utility function U_s is continuously differentiable, using the Karush-Kuhn-Tucker (KKT) theorem[2], the unique

[2]KKT conditions are first order necessary conditions for a nonlinear program to yield a solution that is optimal

maximum x_s^* is given by

$$x_s^* = \min\{\max\{1/U_s'(x_s), 0\}, x_s^{mppt}\} \tag{4.11}$$

where U_s' is the derivative of the utility function U_s.

The control prices (λ) manage the subproblems and are computed by the master algorithm that solves the dual problem. The master algorithm computes the prices by determining λ that minimizes the objective function in (4.8). This is done by updating λ using the gradient $\nabla D(\lambda)$ given by

$$g_l = \frac{\partial}{\partial \lambda_l} D(\lambda) = c_l - y_l \tag{4.12}$$

The gradient projection algorithm solves the dual problem iteratively. At each iteration, each subproblem is solved parallely, and the master algorithm updates the control prices in opposite direction of the gradient such that

$$\lambda_l(t+1) = \max\Big\{\lambda_l(t) - \gamma(c_l - y_l), 0\Big\}, \quad \forall l \in L \tag{4.13}$$

where, $\gamma > 0$ is an appropriate step size.

4.3.2 Choosing a Step Size

Our algorithm is similar to the distributed algorithm described in [17] and guarantees to converge as ∇D is Lipschitz continuous[3] and bounded, provided the step size is appropriately selected. In other words, the convergence of the distributed algorithm is sensitive to the step size used for updating the control prices. While a big step size may cause the

[3]Lipschitz continuous guarantees existence and uniqueness of a solution

algorithm to oscillate around the optimal solution, a small step size may increase the number of iterations required to converge to the solution. Here, we discuss two approaches we used to select a step size to solve the dual problem.

4.3.2.1 Fixed gradient

At each iteration, the master algorithm updates the control prices using the gradient controlled by a fixed step size parameter using (4.13). As shown in [17], the solution generated by the distributed algorithm converges to the primal-dual optimal when the step size satisfies the following condition

$$0 < \gamma < 2/\bar{\alpha}\bar{L}\bar{S} \tag{4.14}$$

where $\bar{\alpha} = max_s\{-1/U_s''(x_s)\}$; $\bar{L} = max_s\{\sum_{l\in L} R_{ls}\}$ and $\bar{S} = max_l\{\sum_{s\in S} R_{ls}\}$.

4.3.2.2 Adaptive gradient (AdaGrad)

In contrast to the fixed gradient, the adaptive gradient modifies the step size as a function of time and updates the control prices $\forall l \in L$ as follows

$$\lambda_l(t+1) = \max\left\{\lambda_l(t) - \frac{\gamma}{\sqrt{G_l(t)+\epsilon}} \cdot g_l(t), 0\right\} \tag{4.15}$$

where $G_l(t) = \sum_{i=1}^{t} g_l^2(i)$ is the sum of the squares of the gradients w.r.t. λ_l up to iteration t; and $\epsilon = 1e^{-8}$ is a smoothing term to avoid division by zero error. Note that the accumulated sum $G_l(t)$ grows with the number of iterations, which in turn causes the step size to shrink. The benefit of Adagrad is it is not very sensitive to the initial step size, and any appropriate step size converges in reasonable amount of time. The convergence guarantees of Adagrad is well studied and the algorithm converges to the optimal solution [40]. Empirically, Adagrad converges faster than the fixed gradient approach and we evaluate both of them in our distributed algorithm.

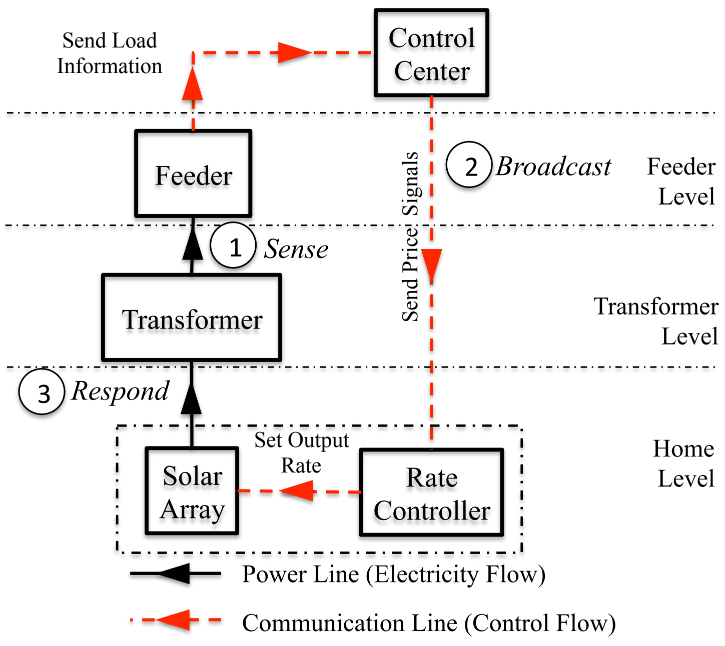


Figure 4.2. *Sense* - *Broadcast* - *Respond* protocol communication among the feeder/transformer level sensors, the control center and the smart solar arrays.

4.3.3 System Design

We now describe our assumptions and the *Sense-Broadcast-Respond* protocol — a round-based protocol. We assume that power flows unidirectionally from the power station to the feeders. However, below the feeder power flow is bi-directional in transformers. Further, we assume the solar arrays have the capability to receive control signals and adjust its rate accordingly.

In our proposed protocol, each round maps to the iterations the distributed algorithm takes to converge to the optimal solution. In each round, prices are computed using (4.13) and sent to individual smart solar arrays to modulate their power outputs. To better illustrate our *Sense-Broadcast-Respond* protocol, we describe the steps on how the control center communicates with the smart solar arrays to rate control its power output (see Figure 4.2).

4.3.3.1 Sense

Sensors at the feeder and transformer capture the load at each time interval. The feeder then communicates the captured information to the grid's control center using Algorithm 1. Note that the aggregate load sensed at the feeder is the combination of the uncontrolled load from buildings and the regulated power from solar panels and is equivalent to the gradient $(c_l - y_l)$ presented in (4.12).

Algorithm 1 Feeder/Transformer's algorithm

```
1: while True do
2:     sense $load_f$
3:     send $load_f$ information to the control center
4:     wait for the next clock tick
5: end
```

4.3.3.2 Broadcast

The utility's control center receives the load from the feeder or transformer and computes the control prices using Algorithm 2. The control prices is adjusted using (4.13) or (4.15). Next, the computed control prices are broadcasted to all smart solar arrays.

Algorithm 2 Utility's control algorithm

Input: γ

```
1: while True do
2:     receive load from feeders/transformers $\forall f, k$
3:     compute gradient $g_l$ based on the load
4:     $\lambda_l := \max\{(\lambda_l - \gamma * g_l), 0\}$          ▷ update control prices
5:     broadcast prices to solar $s \in S(l)$, in constraint $l$
6:     wait for the next clock tick
7: end
```

4.3.3.3 Respond

The smart solar array consists of an identifier pair that associates the array with its parent feeder/transformer. When a smart solar array receives the broadcasted control prices, it computes the rate using (4.11). The identifier aids in associating the prices relevant to the smart solar array. After the rate is computed, the smart solar array sets its generation rate.

Algorithm 3 Smart solar array's algorithm

```
1: while True do
2:     receive control price vector $\lambda$
3:     $q_s := \sum_{l \in L} R_{ls} \lambda_l$                    ▷ aggregate price in $l$
4:     $x_s :=$ argmax$_{0 \le x_s \le m_s}(U_s(x_s) - x_s q_s)$
5:     set solar generation rate for $x_s$
6:     wait for the next clock tick
7: end
```

4.4 Evaluation

We describe the dataset and experimental setup for evaluating our distributed algorithm with different utility functions.

4.4.1 Dataset

For evaluation, we use the smart meter data gathered from a small city in the New England region of the United States. The dataset consists of energy consumption data from 11,186 residential homes. Apart from electricity consumption, we also have the electric grid distribution network information — consisting of the feeders-to-transformers-to-meters connections. Table 4.4.1 shows a brief description of the dataset characteristics and was obtained from the authors of [61].

The dataset also contains solar power generated from a single residential home. To generate solar power dataset for multiple homes, we first normalize the solar power output using its maximum output for the year. Second, we assume the solar installation sizes to be in the range of 4 to 10 kW. Next, we scale the normalized solar output with the uniformly generated points for all homes from this range.

Table 4.1. Key characteristics of the energy demand dataset.

Characteristics	**Value**
Number of electric meters	11,186
Electric meter granularity	5 minutes
Number of feeders	29
Number of transformers	1108
Transformers rating(kVA)	5 to 750
Duration	12 months

4.4.2 Experimental Setup

We run our evaluation for three days (sunny, overcast and output on a variable day) in the month of April that consists of different solar profiles unless otherwise stated. (see Figure 2.2) These solar profile patterns are representative of the different fluctuations observed over a year. Along with the solar profiles, we use the load profile from the corresponding dates as an input to our distributed algorithm.

Our distributed approach takes step size γ as an input to the parameter. For the fixed gradient approach, we use $\gamma = 2/\bar{\alpha}\bar{L}\bar{S} - \epsilon$, as this is the maximum step size to guarantee convergence (4.14). As discussed earlier, the adaptive gradient (Adagrad) is insensitive to the initial step size. We use $\gamma = 0.5$ as the step size for the Adagrad approach. For our experiments, we limit the solar capacity to 15% of the aggregate demand observed at grid level. The time step size is 5 minutes (granularity of the dataset). In addition, instead of reinitializing the control prices at every time step, we use the control prices of the previous time step as an input for the next time step. Further, we use the *cvxpy* library — a python based convex optimization library — to solve the centralized formulation. Internally, the *cvxpy* solver uses *cvxopt* solver to find the optimal solution. Separately, for the distributed scenario we use python to simulate the environment.

4.4.3 Metrics

4.4.3.1 Fairness metric

To assess the fairness of our algorithms, we use the *Gini coefficient* to measure the inequality in allocation distribution. The Gini coefficient is a widely used metric in economics to show the distribution (inequality) of income among the residents of a country. The value for the coefficient is between 0 (perfect equality) and 1 (perfect inequality). Mathematically, it is given by (G),

$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2\sum_{i=1}^{n}\sum_{j=1}^{n} x_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2 \cdot n \sum_{i=1}^{n} x_i} \tag{4.16}$$

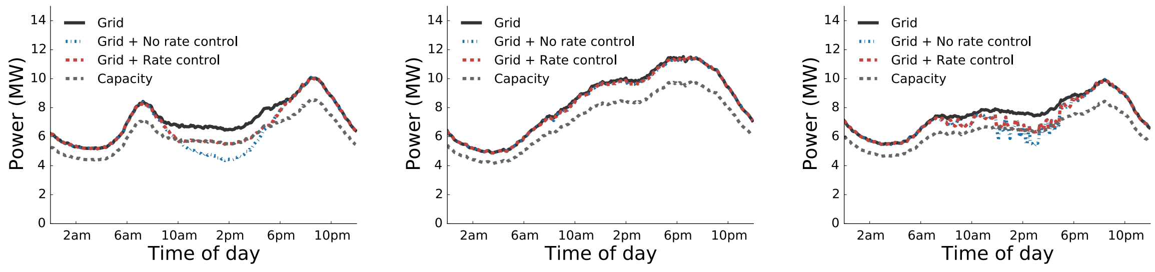
where, in our case, x_i is the rate allocated to user i and n is the total number of grid-tied solar installations.

4.4.3.2 Variability metric

Due to solar intermittency, volatility of the load profile observed at the grid level increases with the introduction of solar energy. This increased volatility makes grid operation of matching the demand with supply more challenging, thereby reducing power quality (i.e. more voltage fluctuations). This volatility can be reduced by controlling the solar output. We use *variability* metric ($\mathcal{V}$) to determine the impact of controlling the rate of solar output and is measured by taking the standard deviation of the successive difference of the power values

$$\mathcal{V} = \sigma(\Delta P) \tag{4.17}$$

where, P is a vector representing the power generated during the day; ΔP represents the difference between successive values in P; and σ represents the standard deviation function. A higher value indicates more variability.



(a) Sunny day (Apr 16) (b) Overcast day (Apr 21) (c) Variable day (Apr 22)

Figure 4.3. Impact of rate control on the aggregate grid demand for different days. With 5% solar penetration and no regulation the solar output exceeds the solar capacity. Our distributed rate control algorithm caps the power output to the desired level.

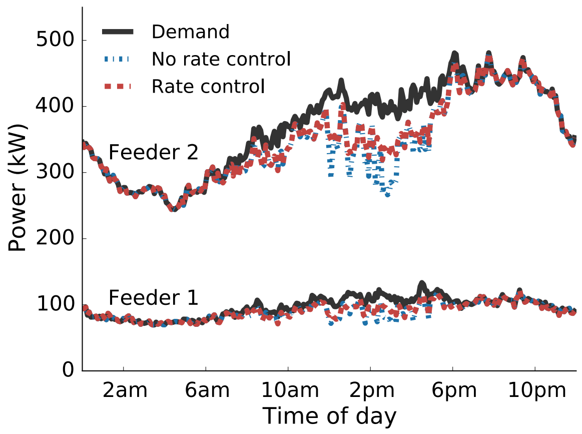


Figure 4.4. Impact of rate control in two feeders with 5% solar penetration. Feeder 1 has fewer homes comparatively.

4.4.4 Experimental Results

4.4.5 Impact on Grid Demand

We assume 5% solar penetration at each feeder i.e. 5% of residential homes have solar panel installations. We compare our approach against no rate control scenario i.e. each solar panel generates power at its maximum value (MPP).

Figure 4.3 shows the impact of our distributed rate control on the aggregate grid demand. The aggregate grid demand profiles usually have two peaks — one in the morning and the other in the evening (Figure 4.3 (a)). The aggregate grid demand with increased solar penetration with no rate control resembles a sitting duck — also known as the *duck* *curve* — and causes *ramp up* and *ramp down* problems [1]. Our algorithm ensures that the net demand with solar power never crosses the solar cap set by the grid. The solar cap alleviates the ramp down and ramp up problems in power generation due to high solar penetration, thereby reducing the need for expensive peaking power plants. This is clearly seen in Figure 4.3(a), where the ramping up/down need is cut in half.

Usually, solar generation on an overcast day is low. Hence, the overall solar energy generated never exceeds the capacity mandated at the grid level (Figure 4.3(b)). In contrast, Figure 4.3(c) depicts a demand profile with variable solar generation, with generation greater or less than the capacity during the different times of the day. Our distributed algorithm adjusts the rate such that it doesn't exceed the solar capacity or the solar array's maximum generation rate .

We observe a similar behavior at the feeder level (see Figure 4.4). Apart from the results shown here, we also ran our simulation for solar penetrations higher than 5%. Even when the maximum solar generation capacity exceeds the local demand, our algorithm limits the rate such that local feeder constraints are met.

4.4.6 Impact on Grid Variability

Next, we show the impact on variability with and without rate control mechanisms. We compute the variability in the demand curve using (4.17). We observe that the net demand seen by the grid with rate control is less variable compared to no rate control mechanisms. Table 4.4.6 shows the variability metric for three representative solar profiles Figure 4.3. Note that introduction of solar energy (regulated or unregulated) increases the variability — as shown by the increased values of the variability metric. However, the variability is much lower with rate control than without it. Moreover, with rate control the load profile at the grid level is either less or equally variable compared to no solar scenario.

Result: Our distributed approach limits the aggregate solar generation output to available solar capacity. Moreover, it decreases the variability in the aggregate grid demand

Table 4.2. Variability metric for different days in 2015.

Load Profile	**Apr 16**	**Apr 21**	**Apr 22**
Grid	0.079	0.076	0.069
Grid + No rate control	0.09	0.079	0.226
Grid + Rate control	**0.084**	**0.079**	**0.145**

4.4.6.1 Impact of Utility function on Solar Rate

We analyze the behavior of weighted and non-weighted utility functions of our rate control algorithm on different panel sizes. Clearly, at the grid level, the output of both the utility functions remain similar as it maximizes both the grid's and user's utility simultaneously. However, the rate allocation generated by the utility functions for individual solar panels would differ based on the size of the solar panel. This is trivially true for the weighted scenario as the allocation is proportional to the size of the panel. In the non-weighted scenario, a smaller sized panel might have reached its maximum generation capacity, thereby allowing larger panels to generate more power. We plot the rate allocation observed on a sunny day for different sized panels (see Figure 4.5). As expected, in the weighted scenario, we observe each panel *backs off* its generation rate proportional to the panel size. Whereas, in the non-weighted scenario, each panel generate power at a similar rate (unless its maximum rate is reached for smaller panels).

Result: Small sized panels benefit more with non-weighted utility, while weighted utility is favorable to bigger panels

4.4.7 Impact of Solar Power Control Policies

Several states in the US have enforced hard limits on the amount of solar energy net metered into the grid. However, these hard caps are quite conservative and do not exploit the available solar potential. Moreover, these policies limit the adoption of solar by residential homeowners. Here, we analyze the change in the number of homes adopting solar installations and the amount of solar energy generated with different rate control policies.

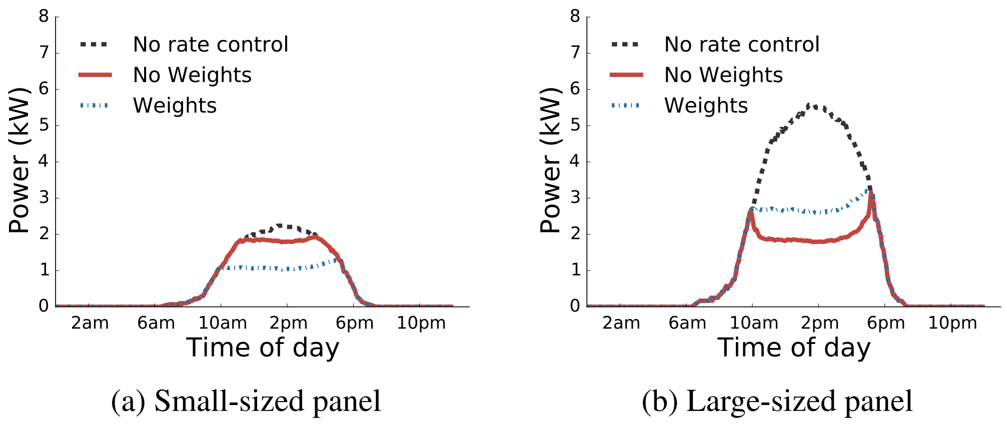
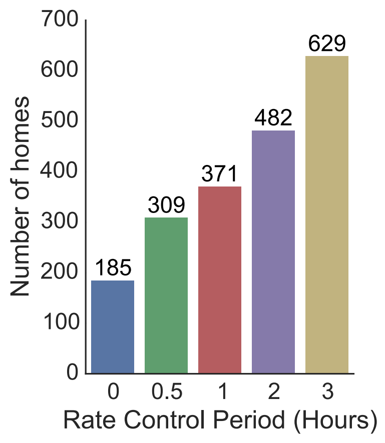


Figure 4.5. Rate allocation for different sized panel. In weighted allocation, each panel backs off its rate proportional to their panel size. However, a non-weighted allocation treats each panel equally and a small-sized panel may generate power equal to a larger sized panel.

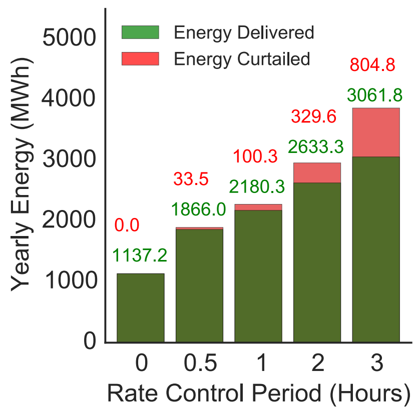
Unlike other experiments, we also assume all panels to be of equal size (5 kW) and evaluate for the entire year 2015. We define the rate control policy as the average hourly curtailment of solar energy per day. For this experiment, we choose rate control policies between 0 to 3 hours.

Figure 4.6 (a) shows the number of homes that can install solar panel systems with different rate curtailment policies. With no daily curtailment, a maximum of 185 homes may be permitted to install solar panels of size 5 kW. However, if we allow just 30 minutes of average daily curtailment, the number can be increased to 309 homes. As we increase the rate curtailment to an hour, we can double the number of homes adopting solar panel systems. Furthermore, with 2 and 3 hours of average daily curtailment we can have 2.6× and 3.4× increase in the number of homes having solar panel systems respectively.

Figure 4.6(b) quantifies the amount of energy delivered to and curtailed by the grid with different rate curtailment policies. As discussed earlier, a maximum of 185 homes can install solar panel system when the total installation size is limited to the minimum load observed for the entire year. The total solar energy supplied to the grid from these distributed sources is around 1137 MWh. However, increasing the average daily curtailment



(a) Number of homes

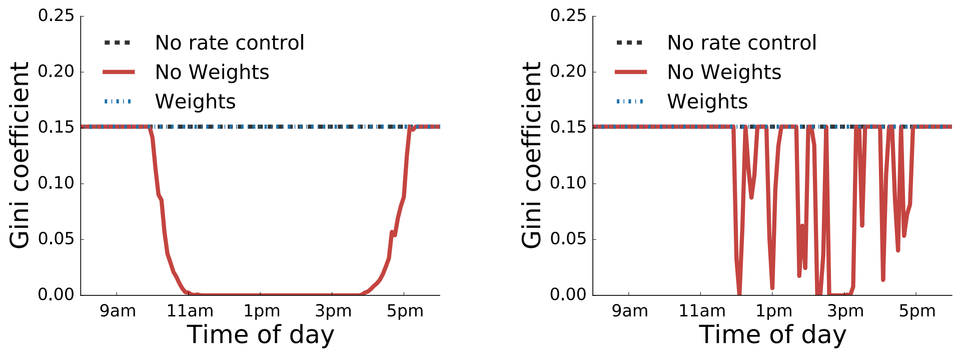


(b) Solar energy

Figure 4.6. Impact of average daily rate control period.

period to 30 minutes, the solar energy delivered to the grid increases by 64%, with solar energy curtailment of just over 1.8%. Furthermore, increasing the curtailment period to an hour, the installed panels can contribute almost doubles the amount of energy to the grid with solar energy curtailment of 4.6%. Similarly, with 2 and 3 hours of average curtailment period, installed solar panels contributes around 2.3× to 2.7× to the grid, with energy curtailment of around 12.5% to 26.2% respectively. Clearly, increasing the rate control period increases the solar energy utilization in the grid provided a small fraction of curtailment is allowed. Intuitively, a solar panel only reaches its peak generation capacity around noon on a clear sunny day. For most periods, the power output is a fraction of the total installation size. Thus, increasing the aggregate installation size increases the amount of solar energy utilized by the grid.

Result: Increasing the rate control period, increases the overall solar utilization in the grid. In particular, an average curtailment of 2 hours enables 2.6× more solar penetration, while causing smart arrays to reduce their output by as little as 12.4%.



(a) Sunny day (Apr 16) (b) Variable day (Apr 22)

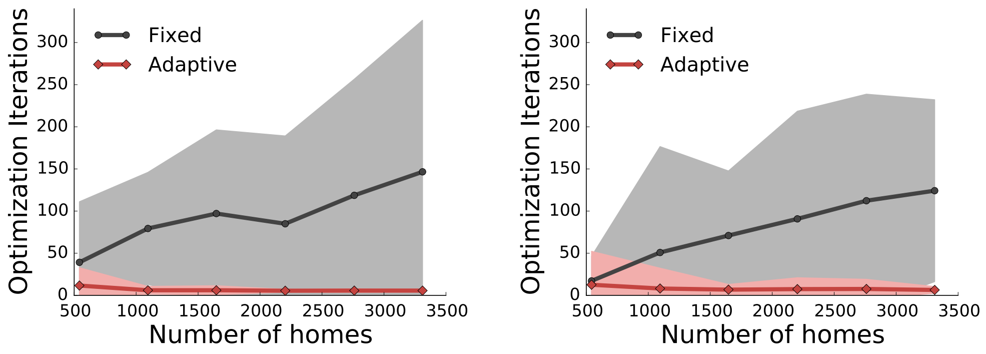
Figure 4.7. Fairness comparison between no rate control mechanism, weighted and non-weighted utility functions.

4.4.7.1 Fairness in Solar Rate Allocation

Our allocation scheme ensures that generation rates of all net-metered solar arrays are assigned in a fair manner — even when solar generation and grid's capacity vary. We use Gini coefficient, a metric for statistical dispersion, to measure the fairness of our proposed approach.

We compare the two utility functions — weighted and non-weighted — with a solar panel generating power at its maximum capacity (MPP) i.e no rate control. We evaluate for three days with 5% solar penetration at each feeder level (see Figure 4.7). With no rate control, all panels will generate power at its maximum rate, wherein the rate is proportional to its installation size. Thus, the Gini coefficient is a constant value, that indicates the inequality in the distribution of the panel sizes. Similarly, in the weighted scenario, the rate allocated would be proportional to the size of the panel. Thus, the Gini coefficient does not change with time and is similar to the MPP scenario.

In contrast, the Gini coefficient will not be constant in the non-weighted scenario as depicted in Figure 4.7. As shown in Figure 4.7 (a), until 10 am, the Gini coefficient is equivalent to the weighted scenario. This is because even when all the panel generates power at its maximum rate it is not able to meet the total available solar capacity. However, as the day progresses, the total generation exceeds the maximum solar capacity and all



(a) Sunny day (Apr 16) (b) Variable day (Apr 22)

Figure 4.8. Convergence plots of fixed and adaptive gradient using our distributed algorithm.

the panels are allocated equal rate, which causes the Gini coefficient to reach zero. On an overcast day (not shown in figure), the maximum available capacity is never reached as all the panels operate at MPP. Hence, Gini coefficient is constant. Separately, on a variable day(see Figure 4.7 (b)), the Gini coefficient varies as it depends on the amount of available capacity met by the generated solar discussed earlier.

Result: Both weighted and non-weighted utilities can be used to achieve fairness in rate allocation.

4.4.7.2 Convergence of our Distributed Approach

As discussed earlier, the convergence of the distributed algorithm is dependent on the step size. Theoretically, a large step size will oscillate and not converge to the optimal solution, while a small step size will take a long time to reach the optimal solution. Here, we empirically, compare the performance of two step-size selection methods — i) Fixed gradient, and ii) Adaptive gradient (AdaGrad). We select step sizes and evaluate for all three days as described in the experimental setup section. Moreover, we assume that the distributed approach has converged if the objective function's output is within two consecutive iteration is less than $1e^{-5}$.

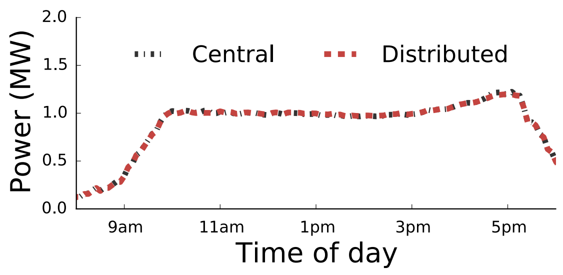


Figure 4.9. Aggregated solar power comparison between the centralized and distributed algorithms.

Figure 4.8 shows the convergence results of the distributed algorithm using different step size methods. Note that the distributed algorithm is run for each time instance of a day. The shaded area highlights the range of iteration counts executed by the algorithm to converge over the day. In the fixed gradient method, the mean and the standard deviation of the number of iterations increases linearly with the number of homes with solar panels.

In contrast, the adaptive gradient takes smaller number of iterations — almost 3$\times$ to 30$\times$ fewer — to converge compared to the fixed gradient approach. Moreover, the adaptive gradient is more reliable, as the standard deviation of the iterations over the day is small. Further, compared to fixed gradient, the number of iterations doesn't grow linearly in the number of homes with solar panels. However, we notice that on an overcast day, the number of iterations required for both the fixed and the adaptive gradient is almost identical. Due to overcast conditions, the maximum solar generation rate is small, which results in faster convergence.

We also compare the performance of our distributed approach with the centralized approach. Note that the centralized approach has full knowledge about the routing topology and the maximum power point (MPP) of each solar installation. As seen in Figure 4.9, the performance of both the centralized and the distributed approach is similar. On an average, we observe that the distributed solution converges to 98.3% of the centralized solution. Moreover, the maximum absolute difference between the distributed and centralized

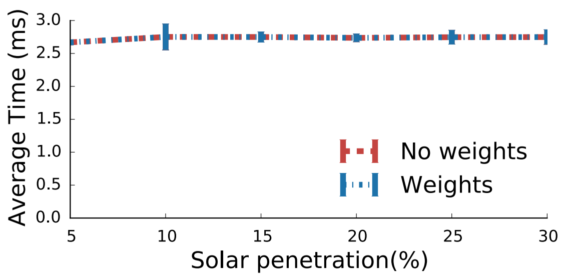


Figure 4.10. Time taken to compute solar rate using control prices in a Raspberry Pi 3.

is 0.029 MW, while the average difference is 0.005 MW.

Result: In comparison to fixed gradient, Adagrad requires 3× to 30× fewer iterations to converge. Moreover, our approach performs similar to the centralized approach i.e. within 98.3% on average.

4.4.7.3 Distributed Solar Rate Computation

We assume that a smart solar-powered arrays will have a Raspberry Pi class processor to receive control prices and control its solar rate at every iteration. Thus, we analyze the average time Raspberry Pi takes to complete a single iteration of the distributed algorithm on un-optimized python code. Note that the solar rate computed depends on the size of the control prices which varies based on the size of the distribution network (number of feeders and transformers). However, the number of feeders and transformers change infrequently for a given grid network (once in every few months or years). Thus, the time taken to compute the rate should theoretically remain the same. Figure 4.10 shows the empirical average time taken to execute the algorithm on Raspberry Pi 3. We observe that the execution time per iteration varies between 2.5 to 3 ms. If we assume the average communication time between the control center and the smart solar arrays to be 10 ms, with 20 iterations (AdaGrad) per convergence (for 5% solar penetration), the distributed algorithm should take less than 0.3 seconds to converge.

Result: With 5% penetration, our distributed approach takes less than 0.3 sec to find the optimal rate allocation.

4.5 Related Work

A detailed assessment of distributed solar impact on the grid highlights the need for generation flexibility in managing solar variability [24]. Solar over-generation causes large ramp up of power generation, which has been shown to pose operational challenges and put a tremendous amount of stress on the grid [1]. Prior work on controlling distributed solar generation include *demand side management* using storage or load matching [123] and *solar regulation* through curtailment or cutoff. Separately, other research work has focused on distributed generation control [81] and shown distributed and centralized voltage control have similar potential in increasing capacities of distributed generation [135].

Numerous studies on solar regulation through curtailment exist [21, 81, 89, 119, 133, 135]. Tonkoski et. al. presents an active power curtailment technique to increase the overall distributed solar capacity at the low-voltage feeder [133]. Rongali et. al. describes a voltage-based curtailment where the solar rate is reduced if the sensed voltage is higher than normal [119]. Lo et. al. presents a discrete curtailment approach by completely disconnecting the solar units through control signals from the utility's command center [89]. In contrast, we present a distributed algorithm that apportions the available solar capacity to individual smart solar arrays through a proportional fairness scheme.

In demand side management, user's demand and solar generation profile is either scheduled intelligently or shifted using energy storage —- to avoid the risk of excess solar supply. Zhao et. al. presents control algorithms for electric vehicle charging to mitigate the impact of renewable energy integration to the grid [147]. Palensky et. al. discusses different approaches to control demand side load [110]. Energy storage absorbs excess energy generated from solar and acts as a buffer for large variations in the output [23, 35]. However, energy storage costs are high and when energy storages are full, excess solar may still need to be curtailed. Our distributed approach is complementary to the energy storage and provides more control over distributed solar generation.

Distributed approach for rate control has been widely studied in the networking literature [71,92]. However, these approaches are now being studied in the context of rate control of electric vehicles [17, 30, 118]. Carvalho et. al. discusses different fairness protocols to mitigate congestion in the grid caused by electric vehicles [30]. Our distributed formulation is similar to the approach proposed in [17]. However, unlike [17], which explores rate control for electric vehicles — we explore rate control in the context of distributed solar and explicitly model electricity distribution network constraints. Moreover, we explore different approaches for faster convergence of our distributed algorithm.

4.6 Distributed Rate Control Summary

In this thesis, we addressed the problem of growth in solar deployments that could cause supply-demand imbalance due to intermittency in power generation. We designed a decentralized rate control algorithm to allocate a generation rate of individual smart solar arrays and apportion the aggregate grid solar capacity. Our proposed decentralized algorithm made decisions local to a solar deployment to compute its solar rate without any need for explicit communication with the utility. We evaluated our rate control algorithm on a city-scale electric distribution network and showed that a dynamic rate control achieves significantly higher solar penetration with negligible energy curtailment compared to the current hard caps placed on solar deployments. We also presented convergence results that exhibit the tractability of our algorithm. Further, we assessed the feasibility of our approach on a Raspberry Pi-class processor and showed that it executes in 0.3 seconds for a solar penetration level of 5%.

CHAPTER 5

VIRTUALIZING DISTRIBUTED ENERGY SYSTEMS

Community solar and storage systems generate energy for all owners as a single aggregated system and do not permit individual control to the users. To overcome this limitation, we argue that community solar and storage should be *virtualized*, wherein a virtual instance of the physical system is created, to maximize its efficiency. In this chapter, we present vSolar — a mechanism to virtualize community solar and energy storage to enable flexible energy sharing algorithms.

5.1 Motivation

While large-scale solar array deployments continue to grow rapidly, the majority of solar installations in North America and Europe continue to be small-scale rooftop systems, primarily in residential homes. However, not every type of residential building is suitable for rooftop installations. In such scenarios, *community solar arrays (CSA)* have emerged as a solution to these challenges. CSA is an array that is collectively owned by a group of individuals and is deployed in a shared location. Each owner leases or purchases a share of the array and is allocated a certain fraction of the solar array in proportion to their share.

As the power output of solar arrays is intermittent, community storage systems are often used in conjunction with CSA to smooth out fluctuations. A community storage system consists of an array of energy storage batteries that are collectively owned by a group, with a fraction of the storage capacity allocated to each owner. While both community solar and storage are nascent technologies, the combination of the two opens up new opportunities for increasing solar penetration and performing various energy optimization.

Community solar and storage systems generate energy for all owners as a single aggregated system and do not permit individual control to the users. To overcome this limitation, we argue that community solar and storage should be *virtualized* to maximize its effectiveness. There are various benefits to virtualizing community solar and storage. First, virtualization enables each owner to independently manage their solar generation and stored energy as if it were a dedicated system. Similar to how virtual machines provide an individual server abstraction and are multiplexed onto a physical machine, a virtual solar and battery arrays provide an abstraction of individually-owned solar and battery arrays that are multiplexed onto the community-owned physical solar and battery array. A second key benefit of virtualization of a community-owned system is that it enables *sharing* of electricity generated or stored in batteries by each virtual system. Such energy sharing, which is not possible in dedicated independently deployed systems, allows a resident to temporarily borrow electricity from one or more neighbor's shares to provide capital and operational savings. Prior works have discussed the benefits of a shared pool of energy storage [85,100] and energy sharing [29, 56, 145]. Here, we present mechanisms to virtualize community solar and storage and also to enable flexible energy sharing algorithms in such systems.

5.1.1 Assumptions

Our work assumes a community solar and storage array that is collectively owned by a group of residents. Each resident is assumed to own a certain fraction of the community solar and storage array. Consequently, the corresponding share of solar output and the stored energy is assigned to each owner. We assume that each community owner can use their portion of the solar array and battery in any manner to perform energy optimization or reduce energy bills. We also assume that each owner can share electricity from their virtual solar or battery array with their neighbors. For example, rather than net-metering surplus electricity to the grid, it is also possible to sell (or lend) this surplus to a neighbor who has high current demand (and is drawing electricity from the grid). Such sharing in the form of lending and borrowing provides both capital and operational cost benefits — it allows a community owner to provision a smaller virtual solar and battery system than the dedicated setup and borrows from others during peak periods. It also provides additional operational benefits by increasing cost savings from a solar and battery system.

Our work also assumes an energy sharing pricing model. Currently, utilities purchase any surplus solar energy at retail prices from users (via net metering). However, when the wholesale price (the cost to sell power to the utility) is same as the retail price (the cost to purchase power from the utility), selling energy to the grid or the neighbor does not provide any additional cost benefits to a user. Cost benefits from sharing energy arise when the wholesale price is less than the retail price. Instead of selling electricity at wholesale price to the utility, a user can earn a profit by selling energy to its neighbors at a rate higher than the wholesale price. Similarly, in this scenario, borrowing energy at a rate lower than the retail price yields cost benefits to both the borrower and the lender. Increased solar penetration has impacted the grid (i.e., wholesale prices turned negative [2]), and with more solar adoption, utility companies will have to rethink how they purchase electricity from distributed sources. We assume that, in the future, utilities will purchase power at a lower rate than retail price, which will enable borrowing and lending of energy.

5.2 vSolar Design

In this section, we first present the key primitives to virtualize a solar and battery array system and then show how these primitives can be used to implement algorithms to control the energy generated and stored in each virtual array.

5.2.1 vSolar Virtualization Mechanisms

Consider a community solar array consisting of P panels with a capacity C_{solar}. The system also consists of a community battery array of B battery cells with a total capacity of C_{batt}. We assume the community-owned system is collectively-owned by N residents, and

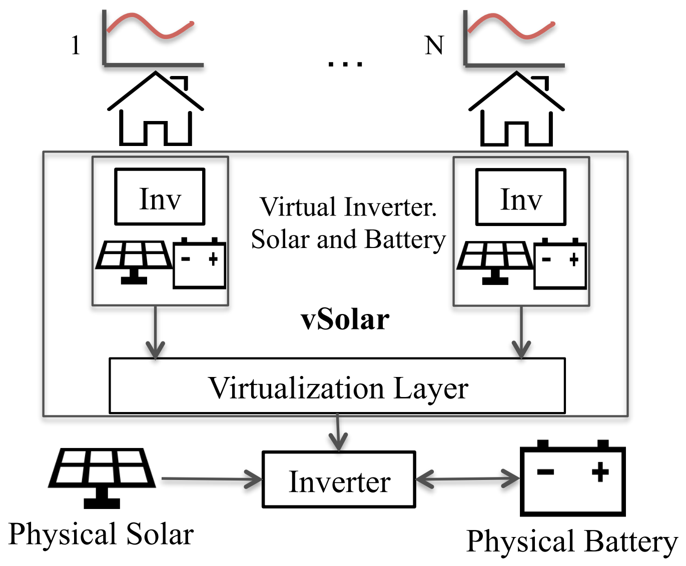


Figure 5.1. vSolar architecture diagram.

allocate each user a virtual share of the solar and battery array. Suppose that the i^{th} owner is allocated a fraction S_i of the solar array and a fraction B_i of the battery array, where $\sum_{i=1}^{N} S_i = 1$ and $\sum_{i=1}^{N} B_i = 1$, $0 < S_i < 1$ and $0 < B_i < 1$. This implies that $S_i \cdot C_{solar}$ capacity of the aggregate solar array and $B_i \cdot C_{batt}$ capacity of the aggregate battery array is allocated to owner i.

From a virtualization standpoint, the system presents the illusion of N smaller solar and battery arrays of the corresponding size, each of which appears as a dedicated system to its owner (see Figure 5.1). That is, owner i sees a virtual solar array of size $S_i \cdot C_{solar}$, a virtual battery of capacity $B_i \cdot C_{batt}$ and a virtual controller (e.g., a virtual inverter) to determine how the solar and battery array output is used at each instant. Virtualization allows each owner to make independent decisions on how to use the system, regardless of how others use their system. The N virtualized systems are "multiplexed" onto the underlying physical solar and battery array, and the overall behavior of the system at any instant represents the aggregate decisions made by each individual virtualized system.

To implement this abstraction, vSolar exposes a set of virtualization primitives that can be controlled by software algorithms in each virtual controller. Let us assume that the array uses a virtual or physical sensor to monitor the solar output, the energy stored in the battery, and the electricity demand of each owner. Let $solar_i(t)$, $battery_i(t)$, and $demand_i(t)$ represent the electricity output of virtual array i, energy stored in virtual battery i and electricity demand of home i at time instant t. To enable an owner to control their virtual system based on these monitored values independently, the physical controller exposes these following software-defined primitives to each virtual controller:

- $charge_i(t)$, which specifies the rate at which the virtual battery should be charged using the output of the virtual solar array at time t
- $discharge_i(t)$, which specifies the rate at which the virtual battery should be discharged to meet a portion of $demand_i(t)$
- $send\_to\_grid_i(t)$ which specifies the rate at which surplus solar electricity should be transmitted (net metered) to the electric grid at time t
- $draw\_from\_grid_i(t)$ which specifies the rate at which electricity should be drawn from the electric grid to meet a portion of $demand_i(t)$

Together these primitives enable each virtual controller to implement flexible software algorithms to control how the solar output and energy storage in the virtual solar and battery array should be used. Each virtual controller can implement its own decisions regardless of how other owners behave.

5.2.2 vSolar Virtualization Algorithm

We now present the vSolar virtualization algorithm that uses the above primitives to implement software control of the virtual solar and battery system within the virtual controller. For home i, let us assume the solar output of virtual array is $solar_i(t)$ and demand is $demand_i(t)$; the vSolar algorithm can determine if the current solar output is adequate to satisfy the demand. If so, the net surplus is computed as:

$$surplus_i(t) = \max\left(solar_i(t) - demand_i(t), 0\right) \tag{5.1}$$

If not, the net deficit is computed as:

$$deficit_i(t) = \max\left(demand_i(t) - solar_i(t), 0\right) \tag{5.2}$$

In the event of a surplus, after first using the solar output to satisfy the entire demand, the controller needs to determine how to utilize the remaining surplus. In this case, if the virtual battery is not fully charged, the surplus is first used to charge the battery at the max charging rate as follows:

$$charge_i(t) = \min(max\_charge\_rate, surplus_i(t)) \tag{5.3}$$

If the battery is full, $charge_i(t)$ is set to zero. If there is additional solar output left after charging the battery at max rate, the rest is net metered to the grid as follows:

$$send\_to\_grid_i(t) = solar_i(t) - demand_i(t) - charge_i(t) \tag{5.4}$$

Conversely, in the event of a deficit, the controller must determine how to satisfy the portion of the demand not met by the virtual solar array. In this case, the decision will depend on the current electricity prices. If off-peak pricing is in effect at time t, then it is better to conserve battery energy for peak periods and satisfy the current deficit from the electric grid:

$$draw\_from\_grid_i(t) = deficit_i(t) \tag{5.5}$$

If peak prices are in effect and the battery is not empty, the controller first draws power from the battery i.e.

$$discharge_i(t) = \min(deficit_i(t), max\_discharge\_rate)$$

so long as $battery_i(t) > low\_threshold$. If the stored energy in the battery is below the $low\_threshold$, then $discharge_i(t)$ is set to zero. Any unsatisfied demand beyond the maximum discharge rate from the virtual battery is met from the grid.

$$\begin{aligned} draw\_from\_grid_i(t) = \max(&demand_i(t) - solar_i(t) \\ &- discharge_i(t), 0) \end{aligned} \tag{5.6}$$

Thus, the vSolar algorithm within each virtual controller can make independent decisions based on the solar output, battery level, and demand of each home. Further, using the virtualization primitives, vSolar enables software-driven algorithmic control of the virtual system.

5.2.2.1 Mapping virtual controller decisions to a physical system

The physical solar and battery controller aggregates all of the decisions made by individual virtual controllers to implement physical control as follows. If the total charge rate of all virtual batteries is greater than the total discharge rate, then the physical battery is charged at a rate

$$charge(t) = \sum_{i=1}^{N} charge_i(t) - \sum_{i=1}^{N} discharge_i(t) \tag{5.7}$$

In contrast, if the total discharge rate across all virtual batteries is greater than the total charge rate, then the physical battery is discharged at the rate of

$$discharge(t) = \sum_{i=1}^{N} discharge_i(t) - \sum_{i=1}^{N} charge_i(t) \tag{5.8}$$

Similarly, if the total power transmitted to the grid by all virtual solar arrays is greater than the total power drawn from the grid, the physical solar array will perform overall net-metering at the following rate:

$$\begin{aligned} send\_to\_grid(t) = \sum_{i=1}^{N} (send\_to\_grid_i(t) \\ - draw\_from\_grid_i(t)) \end{aligned} \tag{5.9}$$

If the opposite is true, no power is net-metered, since all of the solar output is used to satisfy the local demands of all homes and to store energy in the battery.

5.3 Energy Sharing in vSolar

We now discuss how vSolar's virtualization mechanisms can be employed to permit flexible energy sharing. The energy sharing algorithm aims to maximize the energy cost savings across homes while incentivizing borrowers and lenders.

5.3.1 vSolar Energy Sharing Algorithm

vSolar's virtualization algorithm allows each owner to operate their virtual solar and battery array independently of others. In this case, each virtual system is isolated from others, and there is no direct interaction between them. However since all virtual arrays are multiplexed onto a common physical solar array, there are opportunities for the virtual systems to *collaborate* with one another. One form of collaboration is *energy sharing* where virtual systems with a surplus solar generation or surplus stored energy shares it with virtual systems that have a deficit. Such sharing further reduces reliance on the grid, since some or all of the demand of a home is met from other neighboring virtual systems with surplus capacity. In practice, opportunities for energy sharing arise since different homes have different demand profiles. Some homes with daytime occupants will see higher peak usage during day hours, while homes with working occupants will see low usage during

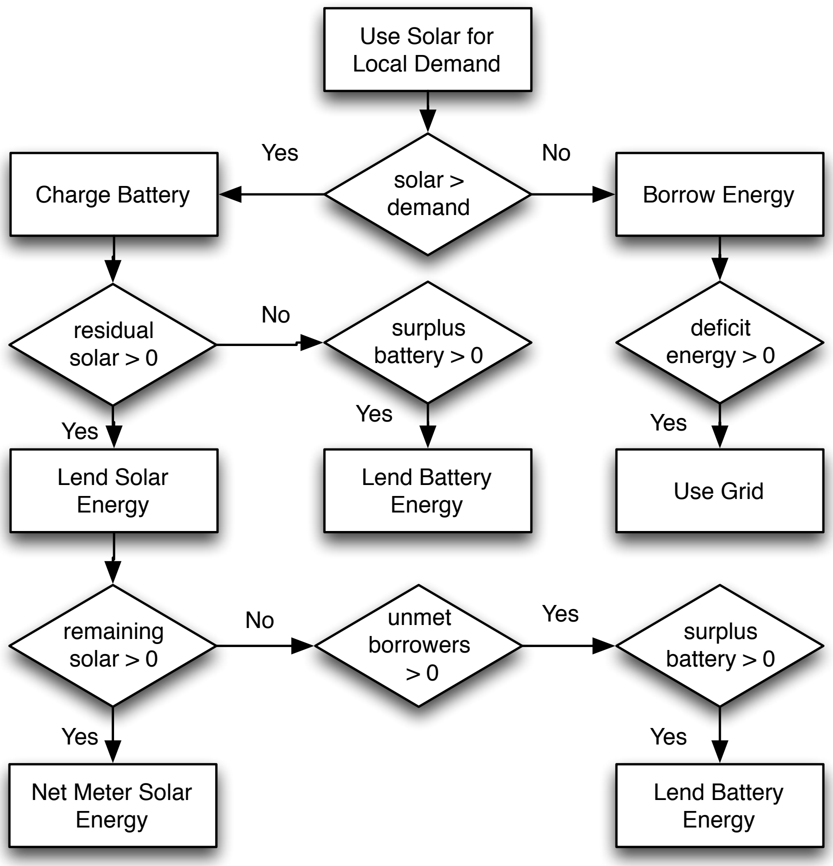


Figure 5.2. A flow chart of vSolar's energy sharing algorithm.

day hours with peak solar generation. The latter homes can lend surplus electricity that would otherwise be net-metered to the grid to the former homes. Similarly, during evening peak periods, demand from homes may peak at different times (e.g., homes with evening peak versus those with late-night usage). In such cases, virtual batteries with surplus stored energy can lend it to others if it is not being used locally for any reason.

Energy sharing makes economic sense only under certain types of electricity pricing schemes. In scenarios where the cost at which grid purchases electricity is the same as the retail cost of buying electricity from the grid, energy sharing provides no monetary benefit. A virtual system can then sell any surplus to the grid via net metering and neighbors with deficit can buy it back from the grid at the same price, requiring no direct cooperation between virtualized systems. However, in scenarios where the grid purchases net-metered electricity at wholesale generation prices and sell it to homes at retail prices, direct lending without grid involvement provides monetary benefits. In this case, rather than purchasing electricity from the grid at a retail price, a virtual system can procure this electricity from a neighboring virtual system with surplus electricity and do so at a price that is *higher* than the wholesale price but *lower* than the retail prices. This incentivizes systems with a surplus since they can sell the surplus at a rate higher than the grid's wholesale prices, while homes with a deficit can purchase this surplus at a price that is lower than the grid's retail rate.

From a virtualization standpoint, energy sharing relaxes the assumption of strict isolation between virtualized systems. It allows a virtual solar array or a virtual battery to increase its capacity by borrowing from surplus homes temporarily. This is analogous to virtual machines that temporarily use unused physical CPU capacity that is allocated to other virtual machines but not currently used.

To implement such energy sharing, vSolar virtual inverters need two additional virtualization primitives.

- $borrow_i(t)$ which specifies the amount of power that home i needs to borrow from any other virtual solar or battery system at time t
- $lend_i(source, t)$ which specifies the amount of surplus power that home i will lend from the specified source at time t. The source can be $solar$, in which case surplus power is lent from the virtual solar array, or $battery$, in which case power is drawn for energy stored in the virtual battery.

These primitives enable a virtual controller to implement any energy sharing algorithm that is best suited to its needs. For our current work, we design a vSolar energy sharing algorithm that is directly based on the vSolar virtualization algorithm presented in the previous section. Our energy sharing algorithm is an enhancement to the basic virtualization algorithm as follows. First, the algorithm determines if the current home should become a borrower, a lender, or neither, at time t. A home is a candidate for lending electricity if its virtual solar array has surplus power that it would have net-metered to the grid. In this case, all of this surplus power becomes available for lending to other homes rather than being net-metered. A home is also a candidate for lending electricity if its virtual battery has a high charge level (above a high watermark threshold) and is willing to share some of the stored energy with others. Specifically, if $solar_i(t) - demand_i(t) - charge_i(t) > 0$ then the home has surplus power it would have previously net-metered and the virtual controller indicates it is willing to lend this power:

$$lend_i(solar, t) = solar_i(t) - demand_i(t) - charge_i(t) \tag{5.10}$$

Further, if the battery has a high charge level indicated by $battery_i(t) > high\_threshold$ and the battery power is not being consumed at the maximum discharge rate, the surplus can be drawn as follows:

$$\begin{aligned} lend_i(battery, t) = \min(&max\_discharge\_rate \\ &- discharge_i(t), 0) \end{aligned} \tag{5.11}$$

Conversely, a home becomes a candidate for borrowing electricity if it has a deficit that would normally require drawing power from the grid. In this case, the home can first request surplus power from other virtual systems, and only request grid power if its deficit cannot be fully met by other lenders. That is, if $demand_i(t) - solar_i(t) - discharge_i(t) > 0$ the home has a unmet deficit and it can make a borrow request as follows:

$$borrow_i(t) = demand_i(t) - solar_i(t) - discharge_i(t) \tag{5.12}$$

In all other cases, $lend_i$ and $borrow_i$ are set to zero. Note that it is possible for a home to neither be a lender not a borrower at time t, a scenario that occurs if it has zero deficit (i.e., has no need to borrow) but can not lend either since all solar electricity is being directed to the virtual battery, which itself has a low charge level (and thus has no solar or battery capacity to lend).

5.3.1.1 Mapping virtual sharing requests onto the physical system

Both $lend_i$ and $borrow_i$ indicate the maximum amount of power that each virtual controller i wishes to lend or borrow based on its current generation and demand. The actual amount of power that is lent or borrowed must then be computed by the physical controller by matching borrowers and lenders. To do so, the physical controller first computes the total borrowing needs as;

$$borrow(t) = \sum_{i=1}^{N} borrow_i(t) \tag{5.13}$$

The solar capacity available for lending is the:

$$lend(solar, t) = \sum_{i=1}^{N} lend_i(solar, t) \tag{5.14}$$

If the solar lending capacity $lend(solar, t)$ exceeds the borrowing demand $borrow(t)$, then all of the borrowing needs can be met from the surplus solar capacity that is available. Each lender can lend an equal amount to meet the total borrowing need or lend in proportion to its solar share S_i. If the total borrowing demand exceeds the total solar capacity, any unmet borrowing need can be lent from stored battery energy that can be lent. The maximum battery power that can be lent is:

$$lend(battery, t) = \sum_{i=1}^{N} lend_i(battery, t) \tag{5.15}$$

Finally, if the borrowing need is still not satisfied by the lending solar and battery capacity (that is, $borrow(t) > lend(solar, t) + lend(battery, t)$), the rest must be drawn from the grid.

$$draw\_from\_grid_i(t) = borrow_i(t) - borrowed\_power_i(t) \tag{5.16}$$

Conversely, if all of the borrowing needs are met by surplus solar energy, any remaining solar can be net metered to the grid as follows

$$send\_to\_grid_i(t) = lend_i(solar, t) - lent\_solar_i(t) \tag{5.17}$$

where $send\_to\_grid_i(t)$ is zero if there are no surplus solar energy, and $borrowed\_power_i(t)$ and $lent\_solar_i(t)$ are amount of power borrowed or lent by home respectively,

5.3.2 Machine Learning based Peak Demand Prediction

The above algorithm assumes a simple threshold-based approach to determine when stored energy can be shared with others. A fixed threshold-based approach to determine the amount of energy to lend does not consider future electricity demands of the owner. In practice, the threshold should not be a fixed value. In scenarios where the owners are away from home, the threshold should be set lower to lend surplus energy and gain maximum benefit from sharing. Similarly, if there is a higher demand the following day, the threshold should be set higher to minimize its reliance on the grid. To handle such dynamic mechanisms, we design a machine learning-based approach to predict demands during peak periods and estimate the surplus energy that can be lent from the battery.

For simplicity, we assume that each owner uses the same algorithm to determine how much surplus energy can be lent from the battery. To determine the surplus energy to lend, we first build a demand model of the home to learn the energy usage behavior during peak pricing periods. The model is then used to predict the electricity demand of the home for the following day. We explore two techniques: Autoregressive Integrated Moving Average (ARIMA) and Support Vector Machines (SVM) with different kernel functions. ARIMA is a popular model used for time series prediction. It uses the AutoRegressive (AR), Integrated (I) and Moving Average (MA) components to predict future points. The autoregressive part is a linear combination of their own lagged values; the moving average part

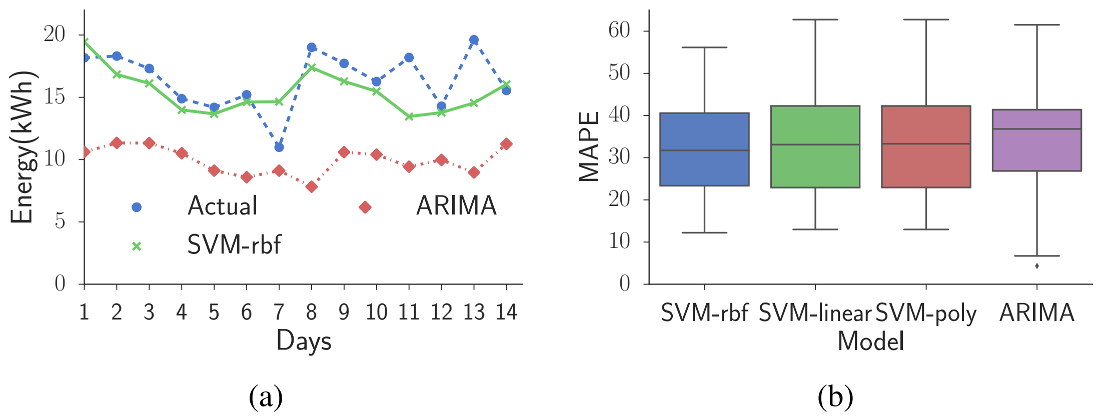


Figure 5.3. (a) Comparison between the actual energy versus predicted output of a home. (b) MAPE values for predicting the electricity usage during peak periods.

captures the regression error as a linear combination of the past error term. And finally, the integrated part indicates whether the data is differenced to make data stationary. Together, it forms a linear equation that uses past values and errors to determine future points.

In contrast, SVM is a supervised regression technique that can use exogenous information such as weather, along with past energy usage to predict energy usage for the following day. We use multiple features as inputs to the SVM model: day of the week, day of the year, outside temperature, humidity, heating and cooling degree days, past power consumption, weekday or weekend information, and holidays. Since the day of the week is cyclical, i.e., repeats every seven days, we encode it by transforming the day into an angle (in steps of $2\pi/7$) and use the sine and cosine values as input feature vectors. We use a similar approach to encode the day of the year. Further, we use one-hot encoding, a standard machine learning technique for encoding categorical labels, to encode weekday or weekend data, before using it as input features. For other data points, we use raw values as inputs.

We use the mean absolute percentage error (MAPE) as the metric to measure the prediction accuracy of the model. We create a prediction model for each home. To train the SVM model, we use one year of the dataset in 2014 and use the 2015 dataset for testing the model. To train the ARIMA model, we perform a grid search on the parameters and select the parameters with the lowest prediction error (MAPE). Figure 5.3(a) shows energy demand prediction during peak periods of a home for several models. The model predicts the energy usage for the following day using past power consumption and future weather information. As seen in the figure, the SVM model is closer to the actual prediction than ARIMA model and is able to predict the dips and spikes in energy consumption. In general, we observe that the ARIMA model captures the general trend of the actual demand but may under or over-predict. Figure 5.3 shows the MAPE value of different models across all homes. We observe that the SVM-RBF model has a median MAPE value of 31%, and is the lowest among all the models used. Since SVR-RBF performs better than other techniques, we use it for evaluating our algorithm.

Determining the surplus energy to lend also requires predicting the solar output for the following day. We use the technique discussed in [62] to predict the solar output for the next day. The method takes into account irradiance and other weather parameters as features to accurately predict future solar production. Combining the future solar output, electricity demand generated from the model, and using the current battery capacity, a candidate home can estimate the amount of energy to lend from the virtual battery. Specifically, the surplus energy is defined as

$$\begin{aligned} surplus\_energy_i(t) = \max(&future\_solar_i(t) + charge_i(t) \\ &- future\_demand_i(t), 0) \end{aligned} \tag{5.18}$$

where $future\_solar_i(t)$ and $future\_demand_i(t)$ are the future predictions for the next 24 hours. Amount of energy to lend, when $surplus\_energy_i(t) > 0$, is defined as

$$\begin{aligned} lend_i(battery, t) = \min(&surplus\_energy_i(t), \\ &max\_discharge\_rate_i - discharge_i(t)) \end{aligned} \tag{5.19}$$

5.4 Discussion

There are several design considerations for realizing vSolar virtualization in practice. First, our approach assumes homes have smart electric meters to monitor local demand

Table 5.1. Summary of the virtualization API primitives vSolar exposes to a virtual controller.

Name	**Description**
$charge(t)$	charge rate of the virtual battery
$discharge(t)$	discharge rate of the virtual battery
$send\_to\_grid(t)$	amount of energy virtual inverter sends to the grid
$draw\_from\_grid(t)$	amount of energy virtual inverter draws from the grid
$borrow(t)$	amount of energy to borrow from others
$lend(source, t)$	amount of energy to lend to others from $solar$ or $battery$

($demand_i$) at an appropriate time granularity (e.g., every few minutes). Second, our approach assumes that in addition to paying monthly electricity bills for consuming electricity from the grid, homes will make or receive micro-payments for borrowing or lending electricity to or from other community homeowners. The information on the amount of electricity borrowed or lent must be tracked by the physical controller (i.e., inverter) and periodically "settled" via actual payments. Third, modern inverters for non-virtualized community solar and battery arrays (e.g., Schneider inverters [4]) provide configuration controls on how much to net-meter and how much or when to charge the battery. Such an inverter can be easily enhanced to support vSolar's virtualization. vSolar can be implemented as a software layer on top of these configuration controls. Specifically, the software layer would expose a virtual controller with vSolar's virtualization primitives to each homeowner. This enables independent control of each virtualized system. The virtualized software layer then takes the decisions from the virtual controllers, aggregates them, and directly exercises the aggregate decision on the physical configurations exposed by the inverters. Thus, it appears as though each user owns a dedicated system making independent judgments. Although, in practice, the decisions are "multiplexed" onto the physical solar and battery. As explained in the previous section, the virtualization layer exposes the API described in Table5.4 to each virtual controller to support virtualization.

5.4.1 Billing and Reconciliation

We also assume that we can net meter and share energy with others. In practice, borrowing and lending of energy require a bill reconciliation infrastructure to account for the micro-payments between homes. Smart meters available today are capable of reporting the local energy consumption of a home. A billing infrastructure can be implemented using smart meters and vSolar primitives, wherein energy consumption information of each home can be logged along with energy borrowed or lent between homes. Separately, we require a billing agreement among homes to determine the costs of energy borrowed or lent.

5.4.2 Other Benefits

There are two key benefits of our approach over multiple dedicated individual-owned or a non-virtualized community-owned system. A dedicated solar and battery system may be infeasible in most cases. Even if the installation was feasible, sharing is not possible as each installation will be on separate circuits. On the other hand, a non-virtualized community-owned installation allows sharing but does not provide individual control of the system. So the total bill across all homes is reduced, but it does not minimize the bill of each house. In contrast, vSolar allows each home to minimize their local bills rather than the overall bill across all households. Another key benefit of vSolar is users can design their own optimization policies. vSolar enables virtualization primitives that are powerful enough to implement other algorithms. For example, in our energy sharing algorithm, users first use surplus energy locally (i.e., satisfy local demand and charge batteries) and then share remaining energy or net meter. However, a range of algorithms is possible. A user can first share surplus energy, and then use the remaining power to store in batteries or net meter. Separately, instead of minimizing energy bills, other policy objectives can also be implemented. A user may have shiftable loads that they can directly control, and thus might define a policy for their virtual battery in conjunction with their controllable background loads.

5.5 Evaluation

We focus on evaluating the potential benefit of vSolar using trace-driven simulations. To do so, we use real electricity load dataset from 50 homes over two years between 2014 to 2015. Electricity dataset was gathered from the New England region of the United States and consists of energy consumption information at a resolution of 30 minutes [63]. We construct different demand profile mixes of 20 homes each from these 50 homes to generate a diversity of homes in a building. To construct the demand profile mixes, we separate the homes into *day* and *night demand profiles*. We define *day profile* homes as homes that have most of their energy demand during the peak pricing hours, i.e., peak to off-peak energy usage is higher than one. In contrast, *night profile* homes use energy mostly during the off-peak pricing period. Next, we randomly select houses belonging to either of the demand profiles proportionately and ran our experiment multiple times to report the overall savings.

We use Wisconsin electric's time-of-use (TOU) prices as a representative pricing model [7]. However, we also use other pricing models, which we present in our results. Wisconsin's peak pricing periods are between 7 a.m. to 7 p.m., while the off-peak periods are from 7 p.m. to 7 a.m. Typically, wholesale prices are 30% to 50% of the retail price [6]. We assume the wholesale electricity rates to be 40% of the retail price and the apartments share energy to others at prices between wholesale and retail price. Note that the residents share only their portion of the community solar or battery energy. The share of solar and battery for each home is determined based on their energy consumption in the previous year, i.e., we assign a solar and battery proportionate to their overall yearly load.

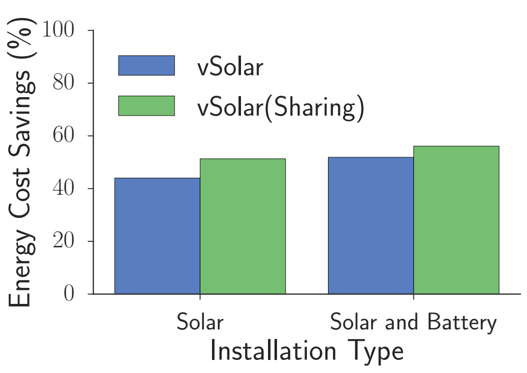
We also use weather data for predicting future electricity usage of a home and its solar generation output. The weather data is available at a one-hour granularity, collected from the Weather Underground website [5]. We resample the weather dataset to 30-minute resolution before running the prediction algorithms. The decision on when and the amount of energy to share is determined using the prediction output, as described in section 5.3.

5.6 Experimental Results

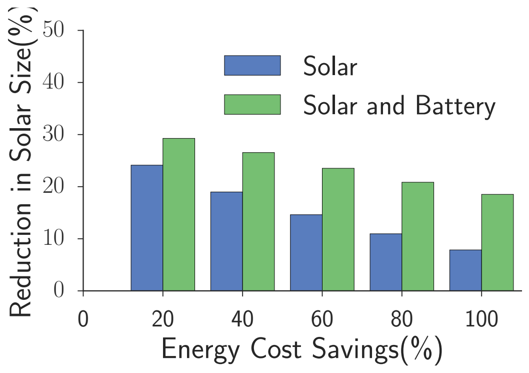
5.6.1 vSolar Benefits

We first analyze the cost benefits of using vSolar and compare it to the energy sharing scenario. Figure 5.4(a) shows the median energy cost savings of a home for both vSolar and the energy sharing scenario. With 40 kW of solar array, vSolar achieves 43% of energy cost savings and yields 8.8% higher savings when coupled with an 80 kWh battery. The energy cost savings further increases when energy is shared. Compared to vSolar, energy sharing provides an additional 8% increase in cost savings. This is because, rather than net metering to the grid at wholesale prices, users can sell surplus electricity at a higher rate to its neighbors, benefiting both the borrower and the lender. We also compared the overall cost savings of vSolar to non-virtualized community solar and battery system, not shown in the figure. We observed that vSolar with sharing achieves similar savings that are close to non-virtualized community solar and battery. This is because, even though vSolar does local optimization (compared to community solar and battery that does global optimization), the difference in energy usage between homes allows enough opportunity to share energy, and thus achieves similar savings.

We now examine the capital expenditure (CapEx) savings achieved through energy sharing. Clearly, dedicated solar arrays and batteries for each home cost more, assuming it is feasible to install one in a community area, simply because we cannot get *economies of scale*. On the other hand, virtualized community-owned solar and battery arrays do not require additional inverters, separate wirings, and thus cost less. Figure 5.4(b) shows the reduction in solar array size (CapEx) of a home with a dedicated system can achieve when the system is virtualized, and energy is shared with others. We observe that as energy cost



(a)



(b)

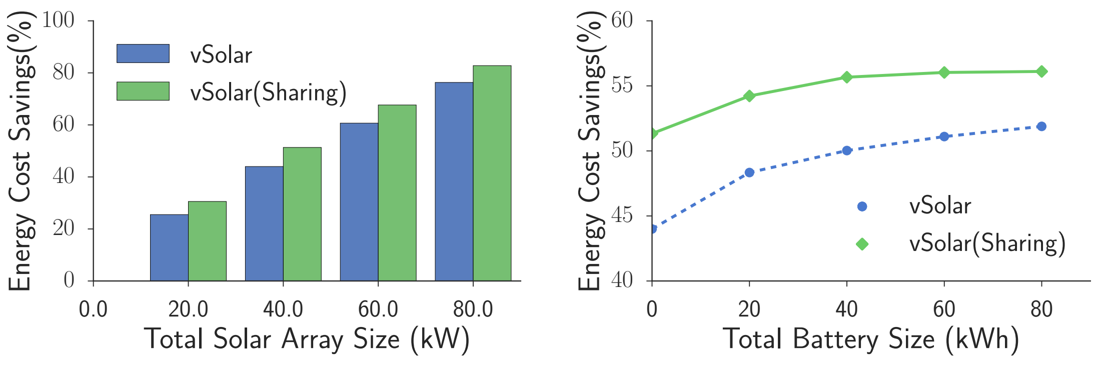
Figure 5.4. (a) Median energy cost savings from vSolar using 40 kW and 80 kWh battery. (b) Reduction in solar capacity through sharing energy to achieve similar cost benefits compared to a dedicated solar and battery system.

savings increase, the percentage reduction in solar array size decreases. With larger solar installations, most homes will have surplus energy to lend, which will reduce the need to borrow energy from others. We note that a dedicated system can reduce its solar array size by 14.6%, through virtualization and energy sharing, to achieve 60% energy cost savings, which is 8.6 kW reduction in absolute values. We observe that a higher reduction in solar arrays can be achieved when batteries are installed. Using a battery capacity of 80 kWh, which is roughly 4 kWh per home, we note a dedicated system can achieve 23.5% reduction in solar array size, which is 13.8 kW reduction in absolute values.

Result: vSolar achieves 43% energy cost savings while providing each user independent control over the virtual system. Moreover, vSolar can reduce the size by 14.6% of a dedicated system through virtualization and energy sharing.

5.6.2 Impact of Solar Arrays

We analyze the impact of different solar array sizes on energy cost savings using vSolar. Intuitively, larger solar arrays generate more solar energy, which in turn reduces the reliance on the grid and minimizes energy costs. However, it is not evident *a priori* how much cost benefits sharing energy provides. Figure 5.5(a) shows the median energy cost savings



(a) Savings with Solar (b) Savings with Battery

Figure 5.5. Median energy cost savings with varying sizes.

across homes using vSolar's virtualization of a 40kW solar array. As expected, the graph shows that the median energy cost savings of a home increase with an increase in solar array size. Moreover, the energy cost savings is higher with energy sharing than the no sharing scenario. Since some occupants during the day may not use their share of solar energy, instead of net metering, the surplus solar energy can be lent to other homes to achieve higher cost savings. Thus, the variations in demand profile among homes allow sharing of energy. In particular, we observe that the median energy cost savings for a home is 43% with a 40 kW solar array size and increases to 51% when energy is shared. Moreover, an 80 kW solar panel can yield energy cost savings of up to 85% using vSolar.

Result: Sharing energy increases the energy cost savings from 43% to 51% for a 40 kW solar installation.

5.6.3 Impact of Energy Storage

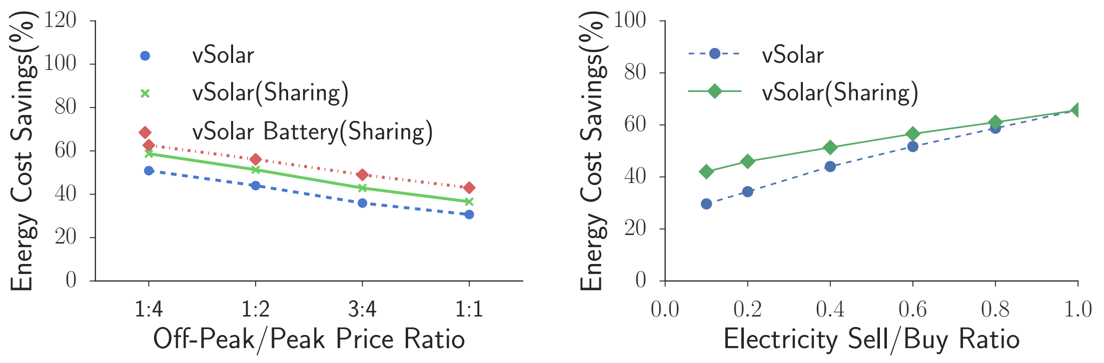
We now study the benefits of employing energy storage. The cost savings from batteries arise due to two primary reasons. First, batteries reduce the amount of energy net metered by storing surplus energy. The energy stored can then be used during peak pricing hours to increase cost savings. Second, sharing any surplus stored energy with others, especially during peak pricing periods, also increases cost savings. Figure 5.5(b) shows the median energy cost savings for different battery sizes and a solar array size of 40 kW. The graph shows that as battery capacity increases the energy cost savings increase. With a small battery size of 40 kWh, which is roughly equivalent to 2 kWh of battery per home, vSolar can increases energy cost savings from 43% to 50%. The energy savings further increase by an additional 5.6% when energy stored in the battery is shared with others. This is because the discharge rate of any battery is limited. Even if the battery has sufficient energy to meet local demands, it may not be possible for a battery to fulfill all of the local energy needs as its maximum discharge rate may limit how much local demand it can satisfy. Since homes may not need to draw energy from batteries at all times, owners can allow energy discharge from their share to fulfill part or all of the local demand, thereby reducing energy costs. Unsurprisingly, we observe diminishing returns with an increase in battery capacity. Since the solar output is finite, there are fewer price differentials to exploit.

We observe that sharing energy also provides battery CapEx savings, i.e., a dedicated system will require a smaller battery size to achieve similar savings compared to the sharing scenario. As seen in Figure 5.5(b), a battery capacity of 80 kWh is required to achieve 51.8% of cost savings. On the other hand, when energy stored in the battery is shared, vSolar achieves 54.2% energy cost savings with a battery capacity of 20 kWh — a 75% reduction in battery capacity yielding significant CapEx savings. Since vSolar allows multiplexing on the same physical unit, it achieves higher cost savings from a solar and battery system compared to a dedicated system for each home.

Result: Virtualization can increase energy cost savings by 8% with a 40 kWh battery size. Cost savings increase an additional 5.6% with energy sharing. Moreover, sharing provides CapEx benefits and can reduce the battery capacity of a dedicated system by 75%.

5.6.4 Effect of Energy Pricing Models

As mentioned earlier, energy cost savings are sensitive to the pricing model. Pricing models such as TOU pricing allow smart algorithms to exploit the price differential between peak and off-peak periods to achieve higher savings. So far, we used Wisconsin's time-of-

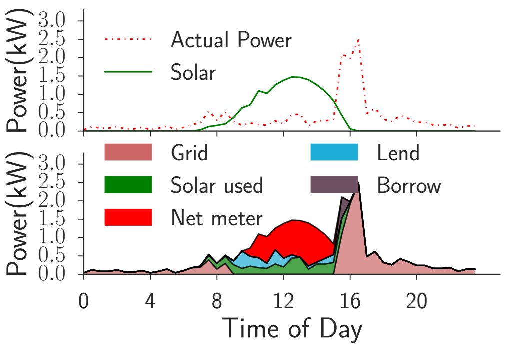


(a) (b)

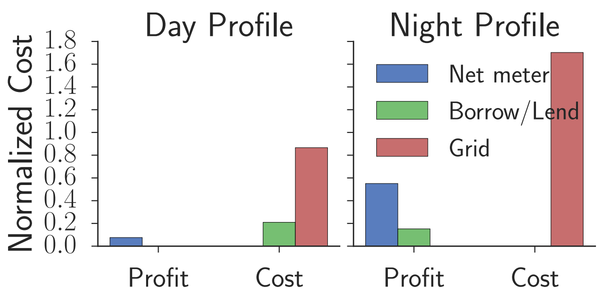
Figure 5.6. (a) As off-peak prices approach peak prices, smaller price differential reduces energy cost savings. (b) As wholesale electricity approaches retail rate, the incentive to share energy reduces.

use pricing model, wherein the off-peak to peak ratio is roughly 1:2. We now analyze how a change in different off-peak to peak ratio impact energy cost savings. Figure 5.6(a) shows the impact on energy cost savings for varying off-peak to peak ratio from a 40 kW solar array size. Clearly, when peak and off-peak prices are similar, there will be no price differential to exploit, and the energy savings will be small. With higher off-peak to peak ratio, we will see more cost savings. As expected, the graph shows that as the difference between peak and off-peak price increases, so does the cost savings. While an off-peak to peak ratio of 1:2 gives 43% in energy cost savings. Further, as off-peak to peak ratio decreases, the energy cost savings increase to 50.8%. Similarly, in the sharing scenario, the energy cost savings increase from 51% to 58.7%. Separately, using 80 kWh battery, the energy cost savings increase from 56.1% to 62.6% with energy sharing. Since batteries play the role of shifting the solar energy to generate cost savings, it has more potential to exploit the price differential in the TOU pricing model.

Next, we examine the impact of the wholesale price (i.e., the price at which electricity is sold to the grid) on energy cost savings. Intuitively, if the wholesale price is the same as the retail price, it is more beneficial to net meter the surplus energy to the grid than share. This is because storing energy for later use may result in loss of power due to battery



(a)



(b)

Figure 5.7. (a) Energy breakdown of a home's demand profile with and without energy sharing. (b) Example cost distribution of homes for day and night demand profiles.

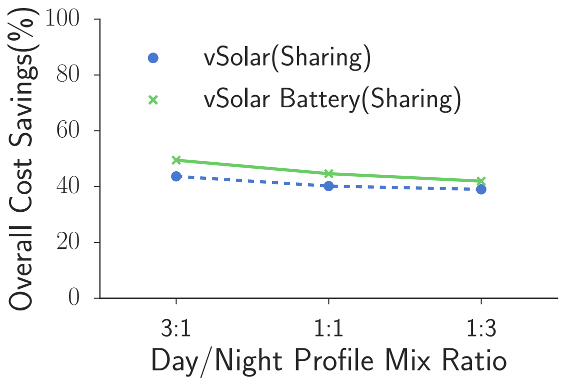


Figure 5.8. Impact of demand profile mixes on cost savings using a 40kW solar and 80kWh battery arrays.

inefficiency, thereby impacting cost savings. However, we observe that sharing is beneficial even when there is battery inefficiency as long as there is some price differential to exploit between wholesale and retail price. Figure 5.6(b) illustrates the energy cost saving with different wholesale to retail price ratio from a solar array size of 40 kW. As wholesale price increase and approaches the retail price, the energy cost savings increase. This is because of the increase in profit by selling electricity to the grid at a higher rate. However, in the sharing scenario, this increase in profit is marginal as the surplus energy is already is sold at a higher rate than wholesale price to others. As expected, the energy cost savings from vSolar equals the sharing scenario as the wholesale price approaches retail price.

Result: As off-peak to peak ratio decreases, energy cost savings in the sharing scenario increases from 51% to 58%.

5.6.5 Effect on Demand Profiles

We now examine the effect of energy sharing algorithm on different demand profiles. Figure 5.7(a) illustrates the energy distribution of a home for a day and compares energy sharing to a non-virtualized setup. The figure shows that without virtualization, any surplus energy is net-metered to the grid. However, with energy sharing, surplus energy is first lent, and the remaining energy is net metered. Further, when energy demand rises late afternoon, and local solar output is insufficient, energy is borrowed from others. Figure 5.7(b) depicts the normalized cost distribution of two homes with day and night demand profiles, with median energy cost savings in their respective cohort. We normalize the cost with its final cost. Note that the energy costs from the grid are higher for night profiles than day profiles. Since solar energy is only available during the day, most of the solar output is either net-metered or lent, resulting in a higher profit than day profile homes, and higher grid costs. In contrast, day profile homes tend to net meter less energy to the grid, with lower net meter cost, but also have to borrow more power during the daytime, resulting in higher borrow costs. It is important to note that while we have implemented a simple energy sharing policy, vSolar's abstractions are flexible, and allows custom different optimization policies for each user.

Finally, we examine the different mix of demand profiles in a building and their impact on the overall cost savings. The overall cost savings are computed by summing the final energy cost of all homes and the original cost. As discussed in Section 5.5, the day profile homes have high energy demands during peak hour periods, while night profile homes have high energy demands during off-peak periods. Figure 5.8 illustrates the impact on cost savings as demand profile mix varies for a 40 kW solar array size. We compare the overall cost savings with solar only versus having both solar and battery. The graph shows that the

overall cost savings are higher for profile mixes where the number of day profile homes is more than the night profile homes. Since peak pricing period occurs during the daytime, most day homes can exploit the price differential and benefit from it. In contrast, homes with night demand profile, do not have high energy usage during the day to exploit the price difference. In particular, overall cost savings vary from 43.6% to 39% as the number of day demand profile decreases compared to night demand profiles. As expected, an addition of an 80 kWh battery increases the overall cost savings as it shifts the surplus solar energy to other periods, and provides an additional potential to exploit price differential.
Result: A demand profile mix, where the day profile homes are more than night profile homes, sees higher overall energy cost savings.

5.7 Related Work

Our work is related to these areas: energy systems, economics, and energy sharing in smart grids.

5.7.1 Energy Systems

Recent work has looked at providing abstractions to physical resources for controlling and improving energy efficiency in buildings [13,33,64,70,73,76,112,141]. BOSS [33] and Microsoft HomeOS [37] provides service abstractions of shared physical resources. These abstractions connect different physical systems and allow the development of applications that enable energy optimization such as HVAC or electrical lighting control. Also, there are studies on developing systems to control HVAC systems in buildings [13, 70, 143]. However, prior works do not propose abstractions for solar or battery arrays for designing energy-efficient systems.

5.7.2 Energy Economics

Numerous studies have focused on the use of renewable energy and its impact on the grid [29, 98, 103, 111, 138]. Since electricity from renewable sources such as solar is highly intermittent and unreliable, intelligent use of batteries can minimize reliance on grid electricity without impacting daily usage patterns [49, 50, 98]. Also, energy cost savings achieved with batteries may be sensitive to certain pricing schemes [29], batteries are profitable when combined with demand response programs [43]. A significant amount of work has also focused on designing algorithms to reduce energy consumption in buildings [39, 98]. There are also studies on designing grid network using solar and storage to manage its local power needs [54]. Our work is built on previous work, where we leverage the variation in demand profiles of homes to optimize the use of solar and batteries.

5.7.3 Energy Sharing

There has been work in pricing and incentivizing energy trading in a microgrid scenario [47, 56, 86, 87, 132, 139, 148]. For example, [148] proposes a novel pricing model to incentivize energy sharing. Further, energy trading between microgrids is shown in [47], while energy sharing model with price-based demand response is proposed in [87]. Other incentives include energy sharing to mitigate privacy leakage [56]. In contrast, our work is complementary and focuses on the systems issues of virtualizing solar and battery sharing. Some of the pricing incentives discussed in prior work can be used in conjunction with our approach. Recent work has also studied bill reconciliation in energy trading [9, 11, 100]. PowerLedger, an energy startup company, uses a distributed ledger to enable a bill reconciliation platform for energy trading [11]. Again, their work is complementary to ours as they provide a platform for tracking energy usage, and does not provide mechanisms for controlling solar or battery arrays. Previous approaches have also studied energy sharing in microgrids [51, 145, 149]. Zhu et al. designed an energy matching algorithm to minimize energy loss via energy sharing [149]. However, our work is complementary as we look into the systems aspects and focuses on providing control to a shared community solar and storage system.

5.8 vSolar Summary

We proposed vSolar — a mechanism to virtualize community-owned solar and battery, wherein each virtual solar and battery can be assigned to an owner and controlled independently, regardless of others. Further, vSolar provides virtualization abstractions that allow each owner to implement their custom energy optimization policy. To show vSolar's potential, we implemented an energy sharing algorithm that enables energy sharing to minimize their local electricity bill. We compared vSolar to a non-virtualized community-owned system and showed similar cost savings while providing local control of the virtualized system. Further, we showed that energy sharing using vSolar achieves energy cost savings over non-virtualized community-owned and dedicated solar and battery systems.

CHAPTER 6

ARCHITECTURE FOR HIGH-THROUGHPUT ENERGY TRANSACTIONS

The previous chapter presented techniques to virtualize a shared energy infrastructure, such as solar arrays and energy storage, to effectively multiplex these resources across multiple users. Such a virtualized system can enable energy trading — allow homeowners to sell or buy energy from their neighbors. However, to realize such an energy trading platform also requires a *ledger*, i.e., an accounting system, that can track energy transactions between two parties.

In this chapter, we present a technique that achieves high throughout IoT energy transactions on a blockchain system. Blockchain technology provides a *distributed ledger* that can enable peer-to-peer energy trading, especially when an infrastructure is shared and energy transactions need to be recorded in a transparent, secure, and verifiable manner. We rearchitect a blockchain system called FabricPlus and show how our approach can scale to the demands of tracking energy transactions.

6.1 Motivation and Overview

In recent years, residential owners are increasingly installing solar arrays with storage. These energy systems are connected to the grid, which enables energy consumers to sell their surplus electricity to others. Since the power grid mostly relies on centralized power plants and operators, so far, electricity is *centrally* managed and traded in the electricity market. Thus, electricity consumers who also produce energy, also known as *prosumers*, can only trade energy with a central entity, which mediates all the transactions [88]. However, the rise of distributed generation sources is driving power companies to rethink energy transactions from a decentralized approach to better manage distributed energy sources and enable more participation from small-scale prosumers. The electric power sector is now experimenting with decentralized systems such as blockchains to facilitate energy transactions and create a self-sufficient, resilient microgrid to make energy markets more efficient [95, 96].

Distributed ledger technology such as blockchains is primarily designed to enable peer-to-peer transactions that bypass a central entity. It provides a distributed ledger, shared by everyone in the network, where transactions between buyers and sellers can be recorded in a verifiable and tamper-proof manner, *necessary* for any energy trading platform. In other words, it makes it easier for energy buyers to directly connect to sellers, eliminating the need for a trusted centralized third-party to verify transactions, which speeds up the verification process. As such, it is widely believed that blockchains will significantly impact the energy sector, reduce costs, improve efficiency, and simplify management of several aspects of electric power systems including trading energy [42, 88].

An essential step before implementing an energy trading platform on blockchains is to understand how energy transactions are done. Since electricity has to be available on-demand, today, energy transactions between two entities do not change the physical flow of electricity. Energy transactions are recorded outside of the electricity flow and used for accounting when and how much energy was delivered and consumed. The frequency of energy trading depends on the cost of electricity on wholesale spot markets, which changes on an hourly or half-hour basis. Thus, hundreds of thousands of buildings will transact every hour or so, and as such, will require a scalable processing system.

While blockchain remains one of the popular decentralized solutions in the energy sector [82, 97], one of the key challenges in blockchains is scalability. Since there is no central entity, participating nodes must process and verify all transactions to guarantee consistency. In a permissionless blockchain, which does not restrict network membership, all participating nodes, and thus, they have limited throughput. From a technical standpoint, the trustless nature of permissionless blockchains requires either proof of work or stake, or expensive global-scale Byzantine-fault-tolerant consensus mechanisms [136], which makes them slow (around 20 transactions per second) [102, 144]. Much recent work has focused on making them more efficient [130, 136]. On the other hand, consortium blockchains or permissioned blockchains, where identities of all participating nodes are known, are known to support faster transactions in comparison to public blockchains, since they delegate consensus and transaction validation to a selected group of nodes, reducing the burden on consensus algorithms. [14, 137]. While even in this setting consensus remains a bottleneck, it is being addressed in recent work [130, 146], motivating us to look beyond consensus to identify further performance improvements in a blockchain.

Our work is based on Hyperledger Fabric v1.2 since it is reported to be the fastest available open-source permissioned blockchain [36]. While there has been some work on optimizing Hyperledger Fabric, e.g., using aggressive caching [131], we are not aware of any prior work on re-architecting the system as a whole. Importantly, our optimizations do not violate any APIs or modularity boundaries of Fabric, and therefore they can be incorporated into the planned release of Fabric version 2.0 [58]. In doing so, we make the following contributions:

- We separate the metadata from data in Fabric's consensus layer, where transaction order is decided. Although it receives whole transactions as input, the consensus layer only requires transaction IDs to decide the transaction order. We redesign Fabric's transaction ordering service to work with only the transaction IDs, resulting in greatly increased throughput.

- We exploit *parallelism* to improve orderer client's throughput and aggressively *cache* unmarshaled blocks.

- Fabric's key-value store in the ordering service that maintains the world state is replaced with a light-weight in-memory data structure whose lack of durability guaran-

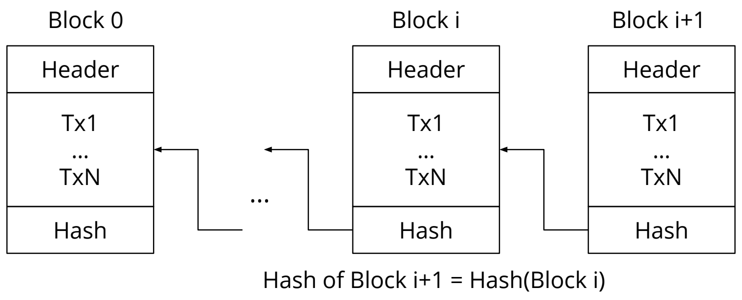


Figure 6.1. Blockchain structure: The blocks are linked together using the cryptographic hash of the previous block.

tees can be compensated by the blockchain. We redesign Fabric's data management layer around a light-weight hash table that provides faster access to the data.

- Our architectural optimizations based on common system design techniques improve the end-to-end transaction throughput by a factor of almost 7, from 3,000 to 20,000 transactions per second, while decreasing block latency.

6.2 Blockchain Basics

A blockchain is a distributed ledger, typically deployed in a peer-to-peer network, where the network participants can transact over the internet without a central entity processing and verifying each transaction. Since each participant has access to the ledger, and no single party controls the data, every member can verify the transactions. The transactions itself is recorded in the distributed ledger in an immutable fashion, by grouping one or more transactions as blocks and protecting them from tampering through cryptographic hashes and a consensus mechanism.

The structure of the blockchain is as follows. New transactions are bundled into a block, and the consensus mechanism is used to decide which blocks are appended to the distributed ledger. Furthermore, the blocks are linked together using the cryptographic hash of the previous block (see Figure 6.1). Thus, the ledger consists of a sequence of interdependent blocks, such that, altering a transaction in a block would require altering the cryptographic hash of every block that is linked.

6.3 Hyperledger Fabric

Hyperledger Fabric is a component of the open-source Hyperledger project hosted by the Linux Foundation and is one of the most actively developed permissioned blockchain systems [27]. In this section, we provide only a short synopsis of Hyperledger Fabric, its architecture, and how transactions are committed to the blockchain. A more detailed version of the transaction flow can be found in [14].

To avoid pitfalls with smart-contract determinism and to allow plug-and-play replacement of system components, Fabric is structured differently than other common blockchain systems. Transactions follow an *execute-order-commit* flow pattern instead of the common *order-execute-commit* pattern. Client transactions are first executed in a sandbox to determine their *read-write sets*, i.e., the set of keys read by and written to by the transaction. Transactions are then ordered by an ordering service, and finally validated and committed to the blockchain. This workflow is implemented by nodes that are assigned specific roles, as discussed next.

6.3.1 Node Types

Clients originate transactions, reads and writes to the blockchain, that are sent to Fabric *nodes*[1]. Nodes are either *peers* or *orderers*; some peers are also *endorsers*. All peers commit blocks to a local copy of the blockchain and apply the corresponding changes to a *state database* that maintains a snapshot of the current *world state*. Endorser peers are permitted to certify that a transaction is valid according to business rules captured in

[1]Because of its permissioned nature, all nodes must be known and registered with a membership service provider (MSP), otherwise, they will be ignored by other nodes.

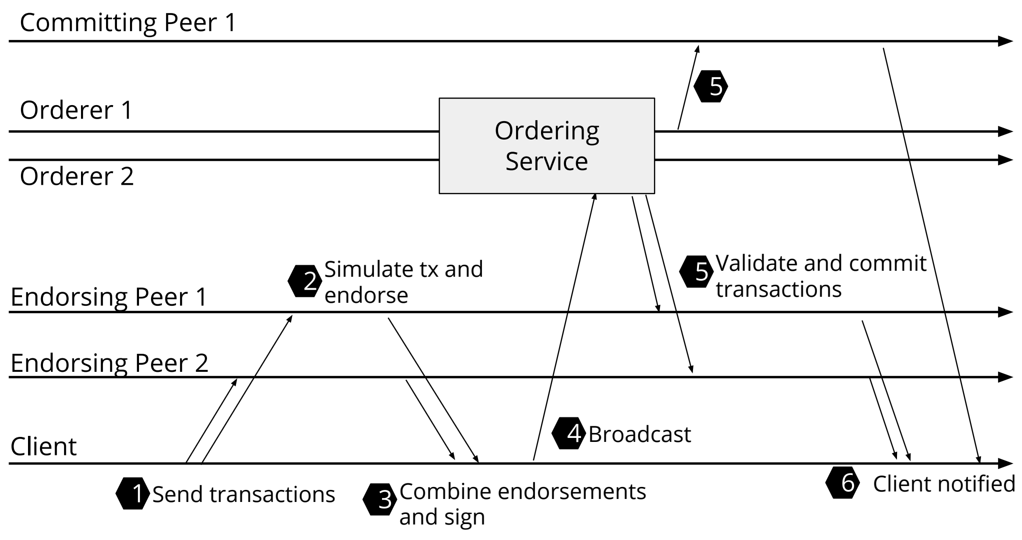


Figure 6.2. The transaction flow of Hyperledger Fabric.

chaincode, Fabric's version of smart contracts. Orderers are responsible solely for deciding transaction order, not correctness or validity.

6.3.2 Transaction Flow Protocol

Figure 6.2 shows the transaction flow of Hyperledger Fabric and works as follows.

1. Client applications sign the transaction proposal with its identity and send it to endorsing peers. Since Fabric is a permissioned blockchain, the identity of the client is important to access the chaincode and the ledger.

2. Each endorser verifies the transaction and executes them in a sandbox. The transaction inputs are used to compute the corresponding read-write set along with the version number of the state that was accessed. Note that the ledger is not updated. Each endorser also uses business rules to validate the correctness of the transaction. The endorsing peers construct a transaction response containing the simulation results and sign it with its identity.

Table 6.1. Transaction throughput per second required for peer-to-peer energy transactions, assuming 48 trades per day for each participant [3]. Note that these transactions do not include trading instructions and other intermediary settlements, likely to increase the overall transactions per day per participant.

Type	Participants	Throughput (txs/sec)	Can Fabric Support?
Small city	10K to 100K	< 56	yes
City	10K to 1M	< 555	yes
Regional	1M to 5M	< 2.7K	yes
ISO-scale	> 5M	> 2.7K	no

3. The client waits for a sufficient number of endorsements and constructs a transaction proposal based on the endorsement policy and signs the proposal.

4. The client *broadcasts* the transaction proposals to the orderer, which implement the ordering service. The orderers are responsible for ordering the transactions and constructing the signed chain blocks. To do so, the orderers first come to a consensus about the order of incoming transactions and then segment the message queue into blocks.

5. Blocks are delivered to peers, who then validate and commit them. The peers verify the identity of the orderers delivering the block, validate the endorsement policy of the transaction, and checks the read set versions to ensure that executing the transactions does not result in an invalid world state.

6. Finally, the blocks are appended to the peer's ledger. Furthermore, only the write sets of the valid transactions are committed to the peer, and the clients are notified through an event.

6.3.3 Problem Statement

The primary goal is to improve the throughput of permissioned blockchain to support the demands of a peer-to-peer energy platform and whether existing blockchain platform supports such throughput. Table 6.1 shows the number of participants and the corresponding throughput required at various-scale. We observe that Hyperledger Fabric cannot support transactions required to support a large city's throughput requirements. Thus, we look into improving Fabric's performance, and our objectives are defined as follows:

- *Performance Benchmarking* - To study the performance of Hyperledger Fabric, and specifically, identify the bottlenecks in the ordering service. Our main goal is to derive insights into improving the overall throughput of the system.

- *Architecture Optimizations* - We aim to improve the overall throughput by identifying all the bottlenecks in the system, and use common system design techniques to improve the end-to-end throughput.

6.4 FabricPlus Design

This section presents our changes to the architecture and implementation of Fabric version 1.2. This version was released in July 2018, followed by the release of version 1.3 in September 2018 and 1.4 in January 2019. However, the changes introduced in the recent releases do not interfere with our proposed techniques, so we foresee no difficulties in integrating our mechanisms with newer versions. Importantly, our improvements leave the interfaces and responsibilities of the individual modules intact, meaning that our changes are compatible with existing peer or ordering service implementations. Furthermore, our improvements are mutually orthogonal and hence can be implemented individually. Before we describe our proposal for performance optimizations, we first take a closer look at the ordering service architecture.

6.4.1 Ordering Service Architecture

To set the stage for a discussion of improvements in Section 6.4, we now take a closer look at the ordering service architecture. Since Fabric is designed to be modular, it supports plug-and-play capability and allows any pluggable consensus protocol for ordering the transactions. The two goals of the ordering service are (a) to achieve consensus on the transaction order and (b) to deliver blocks containing the ordered transactions to the committer peers. The current implementation of Fabric uses an Apache Kafka cluster and Zookeeper, which provides a replicated ordering service that is crash fault-tolerant [16].

After receiving responses from endorsing peers, a client creates a *transaction proposal* containing a header and a payload. The header includes the Transaction ID, and Channel ID[2]. The payload includes the read-write sets and the corresponding version numbers, and endorsing peers' signatures. The transaction proposal is signed using the client's credentials and sent to the ordering service.

When an orderer receives a transaction proposal, it checks if the client is authorized to submit the transaction. If so, the orderer publishes the transaction proposal to a Kafka cluster, where each Fabric channel is mapped to a Kafka topic to create a corresponding immutable serial order of transactions. Each orderer then assembles the transactions received from Kafka into blocks, based either on the maximum number of transactions allowed per block or a block timeout period. Blocks are signed using the orderer's credentials and delivered to peers using gRPC for final validation and committment [32].

6.4.2 Preliminaries

Using a Byzantine-Fault-Tolerant (BFT) consensus algorithm is a critical performance bottleneck in HyperLedger [136]. This is because BFT consensus algorithms do not scale well with the number of participants. In our work, we chose to look beyond this obvious bottleneck for three reasons:

- Arguably, the use of BFT protocols in permissioned blockchains is not as important as in permissionless systems because all participants are known and incentivized to keep the system running in an honest manner.

[2]Fabric is virtualized into multiple *channels*, identified by the channel ID.

- BFT consensus is being extensively studied [22] and we expect a higher-throughput approach may emerge, as researchers address this challenge.
- In practice, Fabric v1.2 does not use a BFT consensus protocol, but relies, instead, on Kafka for transaction ordering, as discussed earlier.

For these reasons, the goal of our work is not to improve orderer performance using better BFT consensus algorithms, but to mitigate new issues that arise when the consensus is no longer the bottleneck. We now present the improvements to the ordering service.

6.4.3 Orderer Improvement I: Separate transaction header from payload

In Fabric v1.2, orderers using Apache Kafka send the entire transaction to Kafka for ordering. Transactions can be several kilobytes in length, resulting in high communication overhead, which impacts overall performance. However, obtaining consensus on the transaction order only requires transaction IDs, so we can obtain a significant improvement in orderer throughput by sending only transaction IDs to the Kafka cluster.

Specifically, on receiving a transaction from a client, our orderer extracts the transaction ID from the header and publishes this ID to the Kafka cluster. The corresponding payload is stored separately in a local data structure by the orderer, and the transaction is reassembled when the ID is received back from Kafka. Subsequently, as in Fabric, the orderer segments sets of transactions into blocks and delivers them to peers. Notably, our approach works with any consensus implementation and does not require any modification to the existing ordering interface, allowing us to leverage existing Fabric clients and peer code.

6.4.4 Orderer Improvement II: Message pipelining

In Fabric v1.2, the ordering service handles incoming transactions from any given client one by one. When a transaction arrives, its corresponding channel is identified, its validity checked against a set of rules and finally it is forwarded to the consensus system, e.g. Kafka; only then can the next transaction be processed. Instead, we implement a pipelined

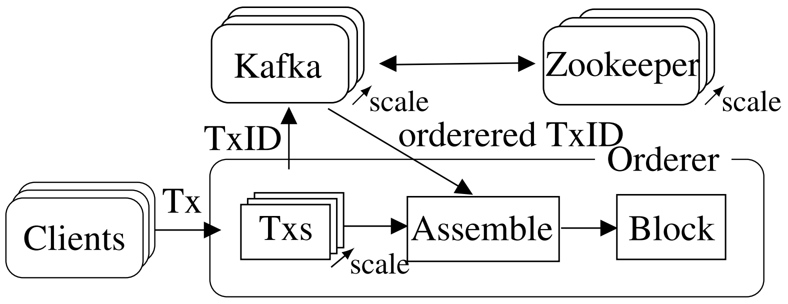


Figure 6.3. New orderer architecture. Incoming transactions are processed concurrently. Their TransactionID is sent to the Kafka cluster for ordering. When receiving ordered TransactionIDs back, the orderer reassembles them with their payload and collects them into blocks.

mechanism that can process multiple incoming transactions concurrently, even if they originated from the same client using the same gRPC connection. To do so, we maintain a pool of threads that process incoming requests in parallel, with one thread per incoming request. A thread calls the Kafka API to publish the transaction ID and sends a response to the client when successful. The remainder of the processing done by an orderer is identical to Fabric v1.2.

Fig. 6.3 summarizes the new orderer design, including the separation of transaction IDs from payloads and the scale out due to parallel message processing.

6.4.5 Orderer Improvement III: Cache unmarshaled blocks

Fabric uses gRPC for communication between nodes in the network. To prepare data for transmission, *Protocol Buffers* are used for serialization. To be able to deal with application and software upgrades over time, Fabric's block structure is highly layered, where each layer is marshaled and unmarshaled separately. This leads to a vast amount of memory allocated to convert byte arrays into data structures. Moreover, Fabric v1.2 does not store previously unmarshaled data in a cache, so this work has to be redone whenever the data is needed.

To mitigate this problem, we propose a temporary cache of unmarshaled data. Blocks are stored in the cache while in the validation pipeline and retrieved by block number whenever needed. Once any part of the block becomes unmarshaled, it is stored with the block for reuse. We implement this as a cyclic buffer that is as large as the validation pipeline. Whenever a block is committed, a new block can be admitted to the pipeline and automatically overwrites the existing cache location of the committed block. As the cache is not needed after commitment and it is guaranteed that a new block only arrives after an old block leaves the pipeline, this is a safe operation. Note that unmarshaling only adds data to the cache, it never mutates it. Therefore, lock-free access can be given to all go-routines in the validation pipeline. In a worst-case scenario, multiple go-routines try to access the same (but not yet unmarshaled) data and all proceed to execute the unmarshaling in parallel. Then the last write to the cache wins, which is not a problem because the result would be the same in either case.

Call graph analysis shows that, even with these optimizations, memory (de)allocation due to unmarshaling is still responsible for the biggest share of the overall execution time. This is followed by gRPC call management and cryptographic computations. The last two problems, however, are beyond the scope of this work.

6.5 Experimental Methodology

This section presents an experimental performance evaluation of our architectural improvements. We used fifteen local servers connected by a 1 Gbit/s switch. Each server is equipped with two Intel® Xeon® CPU E5-2620 v2 processors at 2.10 GHz, for a total of 24 hardware threads and 64 GB of RAM. We use Fabric v1.2 as the base case and add our improvements step by step for comparison. By default, Fabric is configured to use LevelDB as its state database and the orderer stores completed blocks on LevelDB. Instead, we use the in-memory database to store the blocks. Furthermore, we run the entire system without using docker containers to avoid additional overhead.

While we ensured that our implementation did not change the validation behavior of Fabric, all tests were run with non-conflicting and valid transactions. This is because valid transactions must go through every validation check and their write sets will be applied to the state database during commitment. In contrast, invalid transactions can be dropped. Thus, our results evaluate the worst case performance.

For our experiments which focus specifically on the orderer, we isolate the respective system part. In the orderer experiments, we send pre-loaded endorsed transactions from a client to the orderer and have a mock committer simply discard created blocks.

Then, for the end-to-end setup, we implement the full system: Endorsers endorse transaction proposals from a client based on the replicated world state from validated blocks of the committer; the orderer creates blocks from endorsed transactions and sends them to the committer. We do not, however, implement the optimizations in the committer peers, which is beyond the scope of this thesis, and a detailed description of the committer optimizations can be found in [46].

For a fair comparison, we used the same transaction chaincode for all experiments: Each transaction simulates a money transfer from one account to another, reading and making changes to two keys in the state database. These transactions carry a payload of 2.9 KB, which is typical [130]. Furthermore, we use the default endorsement policy of accepting a single endorser signature.

6.6 Results

In this section we first evaluate the performance of Hyperledger Fabric and the Kafka cluster. Next, we use the insights from the evaluation to optimize Hyperledger Fabric and show improvement in the overall throughput.

6.6.1 Impact of Memory-based Store

Note that the orderer maintains the world ledger in LevelDB, and this must be updated sequentially each time. Thus, it is critical that updates to the data store happen at the highest possible rate. In the common scenarios, such as for tracking of wallets, the world

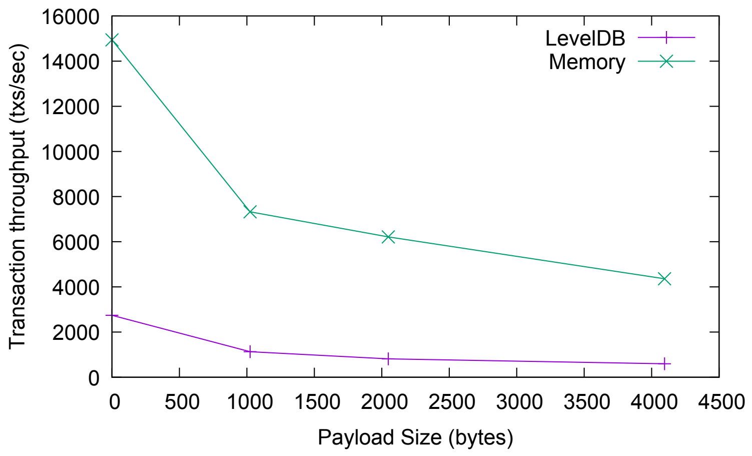


Figure 6.4. Effect of type of world state on throughput.

state is likely to be relatively small. Even if billions of keys need to be stored, this can be kept in the memory to reduce hard disk access, which is relatively slow in comparison. Clearly, volatile memory are susceptible to node failures but since peers already store a copy of the blockchain, it is redundant for orderers to have one. Besides, the blocks can be reconstructed using Kafka cluster, as they persist these transactions. In this experiment, we use the memory key-value store provided by Fabric to evaluate the throughput.

Figure 6.4 shows the overall throughput with different key-value store type. As shown in the figure, the throughput decreases with increase in transaction size. This is because it takes a little longer for the orderer to send/receive transactions to/from the Kafka cluster. We also observe that the memory-based key-value store has a much higher throughput than LevelDB. This is because LevelDB requires disk access, which slows down the overall throughput. In particular, we observe that for a transaction size of 970 bytes (i.e., no payload), we can increase the throughput from 2743 to 11624 transactions per second.

To ensure that our Kafka cluster was not the bottleneck, we also benchmarked our Kafka cluster. Figure 6.5 shows the overall throughput as we vary the payload size. As depicted in the figure, the throughput decreases with increase in payload size. But, more

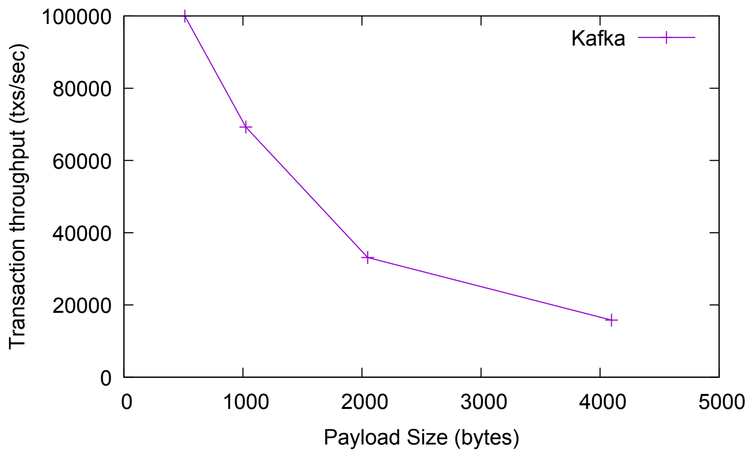


Figure 6.5. Effect of payload size on the overall throughput in Kafka.

importantly, we observe that Kafka can support throughput of 100K TPS for a payload size of 512 bytes, which is much higher than Fabric's 2,743 TPS.

- *Observation 1:* Smaller transaction sizes can support higher throughput. Limiting the size of the transaction for ordering can improve the overall throughput of the system. The throughput increases from 4362 TPS to 7328 TPS when payload size is reduced from 3KB to 1KB.
- *Observation 2:* A memory-based key-value store has higher performance compared to LevelDB. We can switch to memory-based key value for the ordering service, as the primary goal for orderers is to order the transactions.
- *Observation 3:* Since there is a large gap between Fabric and Kafka's performance, there may still be room for improving the overall throughput of the orderer.

6.6.2 Impact of Metadata Separation

In this experiment, we set up multiple clients that send transactions to the orderer and monitor the time it takes to send 100,000 transactions. We evaluate the rate at which an orderer can order transactions in Fabric v1.2. Since we know that *memory-based* key-value

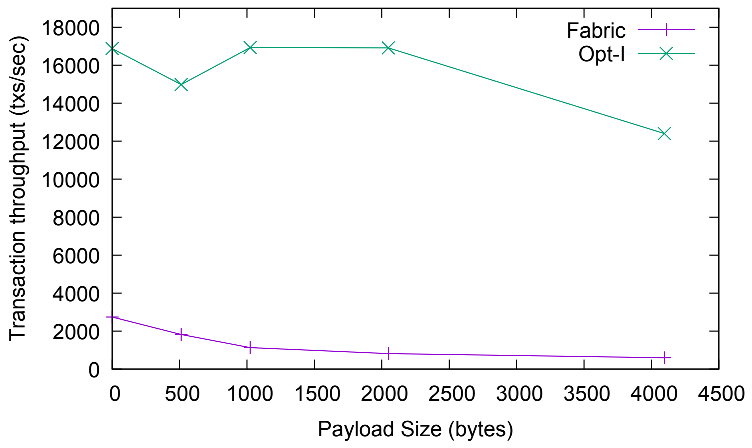


Figure 6.6. Effect of separating the metadata and using Transaction ID for ordering.

store performs better, in the rest of the experiment, we use the memory-based optimization setup in our evaluation.

Based on our previous observation, we know that keeping the transaction size small (both in Kafka and Fabric) and improve the overall throughput. Thus, we only use the Transaction ID for ordering and publish it to Kafka. Figure 6.6 shows the transaction throughput for different payload sizes in comparison to memory-based Fabric v1.2. In Fabric v1.2, transaction throughput decreases as payload size increases due to the overhead of sending large messages to Kafka. By sending only the transaction ID to Kafka (Opt I), we can almost triple the average throughput (2.8×) for a payload size of 4096 KB.

6.6.3 End-to-end Throughput

We now discuss the end-to-end throughput achieved by combining all of our optimizations, i.e., Opt-I combined with Opt-II and -III, compared to our measurements of unmodified Fabric v1.2. We set up a single orderer that uses a cluster of three ZooKeeper servers and three Kafka servers, with the default topic replication factor of three, and connect it to a peer. Blocks from this peer are sent to a single data storage server that stores world state in LevelDB and blocks in the file system. For scale-out, five endorsers replicate the peer state and provide sufficient throughput to deal with client endorsement load. Finally, a client is

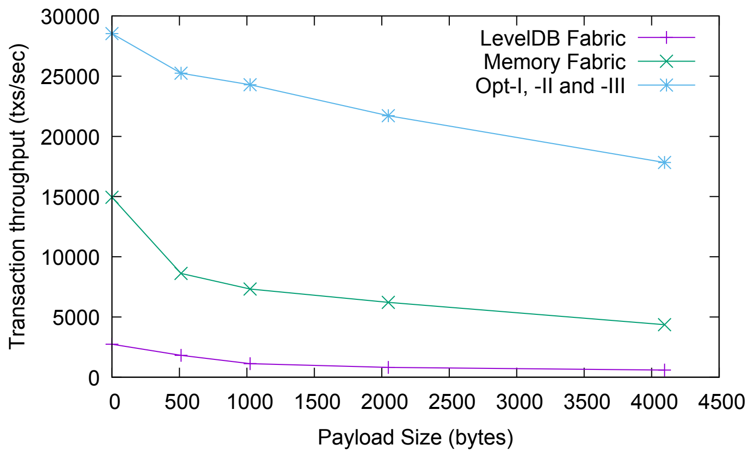


Figure 6.7. End-to-end throughput improvement with orderer optimizations.

installed on its own server; this client requests endorsements from the five endorser servers and sends endorsed transactions to the ordering service. This uses a total of fifteen servers connected to the same 1 Gbit/s switch in our local data center.

We send a total of 100,000 endorsed transactions from the client to the orderer, which batches them to blocks of size 100 and delivers them to the peer. To estimate throughput, we measure the time between committed blocks on the peer and take the mean over a single run. These runs are repeated 100 times. Figure 6.6 shows the overall improvement when we combine all the optimization. We observe that our optimizations can We observe that for a payload size of 2KB, we can achieve a throughput of 24289 transactions per sec, an increase of $3.5\times$ compared to the memory-based Fabric. In comparison to our baseline LevelDB Fabric v1.2 benchmark, for a payload size of 2KB, the improvement is roughly $26\times$.

6.7 Related Work

Hyperledger Fabric is an open-source blockchain system that is still undergoing rapid development and significant changes in its architecture. Hence, there is relatively little work on the performance analysis of the system or suggestions for architectural improvements. Here, we survey recent work on techniques to improve the performance of Fabric.

The work closest to ours is by Thakkar *et al* [131] who study the impact of various configuration parameters on the performance of Fabric. They studied the performance on Hyperledger Fabric v1.0. They find that the major bottlenecks are repeated validation of X.509 certificates during policy verification, sequential policy validation of transactions in a block, and state validation during the commit phase. They introduce aggressive caching of verified endorsement certificates, parallel verification of endorsement policies, and batched state validation and commitment. These improvements increased the overall throughput by a factor of 16, and the optimizations were incorporated in Fabric v1.1. This paper studies the performance of Fabric v1.2 and improves over the currently available version.

In recent work, Sharma *et al* [124] study the use of database techniques, i.e., transaction reordering and early abort, to improve the performance of Fabric. Some of their ideas related to early identification of conflicting transactions are orthogonal to ours and can be incorporated into our solution. Androulaki *et al* [15] study the use of channels for scaling Fabric. However, this work does not present a performance evaluation to establish the benefits from their approach quantitatively. Raman *et al* [117] study the use of lossy compression to reduce the communication cost of sharing state between Fabric endorsers and validators when a blockchain is used for storing intermediate results arising from the analysis of large datasets. However, their approach is only applicable to scenarios which are insensitive to lossy compression, which is not the general case for blockchain-based applications.

Some studies have examined the performance of Fabric without suggesting internal architectural changes to the underlying system. For example, Dinh *et al* use BlockBench [36], a tool to study the performance of private blockchains, to study the performance of Fabric, comparing it with that of Ethereum and Parity. They found that the version of Fabric they studied did not scale beyond 16 nodes due to congestion in the message channel. Nasir *et al* [104] compare the performance of Fabric v0.6 and v1.0, finding, unsurprisingly, that the 1.0 version outperforms the 0.6 version. Baliga *et al* [20] showed that application-level parameters such as the read-write set size of the transaction and chaincode and event payload sizes significantly impact transaction latency. Similarly, Pongnumkul *et al* [113] compare the performance of Fabric and Ethereum for a cryptocurrency workload, finding that Fabric outperforms Ethereum in all metrics. Bergman [25] compares the performance of Fabric to Apache Cassandra in similar environments and finds that, for a small number of peer nodes, Fabric has a lower latency for linearizable transactions in read-heavy workloads than Cassandra. On the other hand, with a larger number of nodes, or write-heavy workloads, Cassandra has better performance. However, their approaches do not propose any optimizations to improve the overall throughput of Fabric's orderer component.

6.8 FabricPlus Summary

We show that existing blockchains are insufficient to support peer-to-peer energy transactions of a large city. The main contribution of this work is to show how we can re-engineer a permissioned blockchain framework such as Hyperledger Fabric to support $> 20,000$ transactions per sec, a throughput sufficient to support a large city-scale peer-to-peer energy platform. We accomplished this goal by implementing a series of independent optimizations focusing on I/O, caching, parallelism, and efficient data access. In our design, orderers only receive transaction IDs instead of full transactions, and we aggressive use caching and memory-based storage for fast data access on the critical path. Our results show that we can achieve a throughput of 24,289 transactions per sec, an increase of $26\times$ compared to the baseline LevelDB Fabric.

CHAPTER 7
SUMMARY AND FUTURE WORK

7.1 Thesis Summary

This thesis has explored the opportunities and challenges in designing IoT-based smart energy systems and provides insights into building the next-generation smart grid. In this dissertation, I have demonstrated how systems and machine learning principles can be used to build smart IoT energy systems. We developed several new techniques and proposed new mechanisms for a wide range of areas in IoT-based energy systems. In doing so, I have made the following contributions:

1. *Planning and placement*: First, I discussed how the current state-of-the-art LIDAR approaches in estimating the solar potential of a roof do not scale. To address the challenges, I proposed DeepRoof, a data-driven system that uses deep-learning to identify ideal locations for installing solar arrays on rooftops by estimating solar potential using satellite imagery. I evaluate our approach on different types of roof and show that our technique is comparable to LIDAR-based approaches.

2. *Decentralized control*: Second, I defined the problem of solar rate control to prevent congestion in the grid. Further, I proposed a decentralized rate control technique that can self-regulate in a grid-friendly manner. Additionally, the proposed approach provides flexible control of solar output, and I showed that such mechanisms allow for higher solar penetration in the grid.

3. *Resource management*: Third, I discussed the challenges in community-owned distributed energy resources that do not provide independent control to users. I proposed

vSolar, an approach to virtualize the solar arrays and energy storage that enables independent control. Further, vSolar can be used to implement custom energy sharing policies and reduce the energy costs of individual homeowners through energy trading.

4. *Decentralized architecture*: Finally, I presented the challenges in energy trading using permissioned blockchains. I proposed FabricPlus, a series of optimizations in an open-source Hyperledger Fabric, that allows a blockchain system to achieve high throughput IoT energy transactions, without necessitating any change to its external interfaces. I also show considerable performance improvement over the baseline Fabric.

7.2 Future Work

This dissertation covers a broad range of areas in the area of IoT energy systems and gives rise to several promising directions to enable the next generation of smart grid. Next, I will outline some directions for future work that has emerged from this thesis.

1. *Rooftop modeling and analytics:* The deep learning-based approach in DeepRoof can be extended to either improve the system and derive additional insights for city-scale planning and development. While DeepRoof uses only satellite imagery of a rooftop, images taken from other angles, such as street view, can be combined with satellite images to construct a more accurate 3D model of the roof. Additionally, the material of roofs and any cracks or damage to a roof are readily available. These sources of information can be combined in predicting the type and damages to the rooftop and help city planners assess the post-seismic damage to a city's buildings in a scalable manner.

2. *Virtualization and virtual power plant:* The virtual abstractions described in Chapter 5 are designed to control a vast deployment of batteries and solar and enable indepen-

dent control across users. Alternatively, the same abstractions can be used to enable a *software layer* to control a host of virtual energy systems. The software layer can be a set of mechanisms that manage the energy output of solar arrays and batteries from distributed energy systems. This can help utility companies to manage renewable solar energy and feed this energy to the grid as needed.

3. *Energy trading platform:* While this thesis has focussed on re-architecting the blockchains to support high throughput energy transactions, implementing a peer-to-peer energy trading platform may involve challenges other than scalability. For starters, we need to determine the kind of information being recorded on the shared ledger, that does not leak private information. We also need an efficient mechanism to match the buyers and the suppliers and to establish a marketplace. Both the buyers and the sellers need to respond to the market dynamics, which reflect the grid networks needs at each moment to load balance. It would be interesting to explore the different scenarios and conditions and how various entities establish communication to enable trading.

BIBLIOGRAPHY

[1] What the Duck Curve Tells us about Managing a Green Grid. `https://goo.gl/YvI2uc`, Feb 2016.

[2] *California is getting so much power from solar that wholesale electricity prices are turning negative*, 2017. `http://bit.ly/2pdJKg2` (Accessed Jan, 2018).

[3] Peer-to-Peer Distributed Ledger Technology Assessment - Virtual Peer-to-Peer Energy Trading Using Distributed Ledger Technology: Comprehensive Project Assessment Report. In *AGL Energy Limited* (2017).

[4] *Schneider Solar Hybrid Inverter Systems*, 2017. `https://solar.schneider-electric.com/`.

[5] *Weather dataset API*, 2017. `https://www.wunderground.com/`.

[6] *Wholesale Electricity Price in the US*, 2017. `https://www.eia.gov/electricity/wholesale/`.

[7] *Wisconsin Electric Rates*, 2017. `http://bit.ly/1lRsroB`.

[8] How much electricity on average do homes in your state use? (ranked by state). `https://goo.gl/j49Wnp`, May 21st 2018.

[9] *LO3 Energy: Brooklyn micro-grid*, 2018. `https://lo3energy.com/` (Accessed Jan, 2018).

[10] *OpenStreetMap Overpass API*, 2018. `https://wiki.openstreetmap.org/wiki/Overpass_API`.

[11] *Power Ledger: Peer to peer energy trading.*, 2018. `https://powerledger.io/` (Accessed Jan, 2018).

[12] The 50 States of Solar: A Quarterly Look at America's Fast-Evolving Distributed Solar Policy Conversation. `https://goo.gl/bXRy3p`, November 2015.

[13] Agarwal, Y, Balaji, B, Dutta, S, Gupta, R K, and Weng, T. Duty-cycling buildings aggressively: The next frontier in HVAC control. In *10th International Conference on Information Processing in Sensor Networks (IPSN)* (2011).

[14] Androulaki, Elli, Barger, Artem, Bortnikov, Vita, Cachin, Christian, Christidis, Konstantinos, De Caro, Angelo, Enyeart, David, Ferris, Christopher, Laventman, Gennady, Manevich, Yacov, Muralidharan, Srinivasan, Murthy, Chet, Nguyen, Binh, Sethi, Manish, Singh, Gari, Smith, Keith, Sorniotti, Alessandro, Stathakopoulou, Chrysoula, Vukolić, Marko, Cocco, Sharon Weed, and Yellick, Jason. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18* (2018), 1–15.

[15] Androulaki, Elli, Cachin, Christian, De Caro, Angelo, and Kokoris-Kogias, Eleftherios. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In *European Symposium on Research in Computer Security* (2018), Springer, pp. 111–131.

[16] Apache Foundation. Apache Kafka, A Distributed Streaming Platform, 2018.

[17] Ardakanian, O., Rosenberg, C., and Keshav, S. Distributed control of electric vehicle charging. In *Proceedings of the fourth international conference on Future energy systems* (2013).

[18] Assouline, Dan, Mohajeri, Nahid, and Scartezzini, Jean-Louis. Building rooftop classification using random forests for large-scale pv deployment. In *Earth Resources and Environmental Remote Sensing/GIS Applications VIII* (2017), International Society for Optics and Photonics.

[19] Badrinarayanan, V, Kendall, A, and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561* (2015).

[20] Baliga, Arati, Solanki, Nitesh, Verekar, Shubham, Pednekar, Amol, Kamat, Pandurang, and Chatterjee, Siddhartha. Performance Characterization of Hyperledger Fabric. In *Crypto Valley Conference on Blockchain Technology, CVCBT 2018* (2018).

[21] Bandyopadhyay, S, Kumar, P, and Arya, V. Planning curtailment of renewable generation in power grids. In *Proccedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling* (2016).

[22] Bano, Shehar, Sonnino, Alberto, Al-Bassam, Mustafa, Azouvi, Sarah, McCorry, Patrick, Meiklejohn, Sarah, and Danezis, George. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936* (2017).

[23] Barton, J., and Infield, D. Energy storage and its use with intermittent renewable energy. *IEEE transactions on energy conversion* (2004).

[24] Bebic, J. *Power system planning: emerging practices suitable for evaluating the impact of high-penetration photovoltaics*. NREL, 2008.

[25] Bergman, Sara. Permissioned blockchains and distributed databases: A performance study. Master's thesis, Linkoping University, 2018.

[26] Bishop, J KB, and Rossow, W B. Spatial and temporal variability of global surface solar irradiance. *Journal of Geophysical Research: Oceans* (1991).

[27] Cachin, Christian. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers* (2016), vol. 310.

[28] Cano, Daniel, Monget, Jean-Marie, Albuisson, Michel, Guillard, Hervé, Regas, Nathalie, and Wald, Lucien. A method for the determination of the global solar radiation from meteorological satellites data. *Solar energy* (1986).

[29] Carpenter, T, Singla, S, Azimzadeh, P, and Keshav, S. The impact of electricity pricing schemes on storage adoption in Ontario. In *Proceedings of the 3rd International*

Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (2012).

[30] Carvalho, R., Buzna, L., Gibbens, R., and Kelly, F. Critical behaviour in charging of electric vehicles. *New Journal of Physics 17*, 9 (2015).

[31] Chen, Zhi, Wu, Lei, and Fu, Yong. Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization. *IEEE Transactions on Smart Grid* (2012).

[32] Cloud Native Computing Foundation. gRPC: A high performance, open-source universal RPC framework, 2018.

[33] Dawson-Haggerty, Stephen, Krioukov, Andrew, Taneja, Jay, Karandikar, Sagar, Fierro, Gabe, Kitaev, Nikita, and Culler, David E. BOSS: Building Operating System Services. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation(NSDI)* (2013).

[34] Demain, Colienne, Journée, Michel, and Bertrand, Cédric. Evaluation of different models to estimate the global solar radiation on inclined surfaces. *Renewable Energy* (2013).

[35] Denholm, P., Ela, E., Kirby, B., and Milligan, M. The role of energy storage with renewable electricity generation.

[36] Dinh, Tien Tuan Anh, Wang, Ji, Chen, Gang, Liu, Rui, Ooi, Beng Chin, and Tan, Kian-Lee. BLOCKBENCH: A Framework for Analyzing Private Blockchains. *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17* (2017), 1085–1100.

[37] Dixon, C, Mahajan, R, Agarwal, S, Brush, AJ, Lee, B, Saroiu, S, and Bahl, P. An operating system for the home. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation(NSDI)* (2012).

[38] Dornaika, Fadi, Moujahid, Abdelmalik, El Merabet, Youssef, and Ruichek, Yassine. Building detection from orthophotos using a machine learning approach: An empirical study on image segmentation and descriptors. *Expert Systems with Applications* (2016).

[39] Doukas, H, Patlitzianas, K D, Iatropoulos, K, and Psarras, J. Intelligent building energy management system using rule sets. *Building and environment* (2007).

[40] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011).

[41] Dutta, A., Gupta, A., and Zissermann, A. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016.

[42] Espinel, Victoria, O'Halloran, D, Brynjolfsson, E, and O'Sullivan, D. Deep shift, technology tipping points and societal impact. In *New York: World Economic Forum–Global Agenda Council on the Future of Software & Society (REF 310815)* (2015).

[43] Exarchakos, L, Leach, M, and Exarchakos, G. Modelling electricity storage systems management under the influence of demand-side management programmes. *International Journal of Energy Research* (2009).

[44] Gagnon, Pieter, Margolis, Robert, Melius, Jennifer, Phillips, Caleb, and Elmore, Ryan. Rooftop solar photovoltaic technical potential in the united states. a detailed assessment. Tech. rep., NREL (National Renewable Energy Laboratory (NREL), Golden, CO (United States)), 2016.

[45] George, Ray, Wilcox, Steve, Anderberg, Mary, and Perez, Richard. National solar radiation database (nsrdb)–10 km gridded hourly solar database. Tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO., 2007.

[46] Gorenflo, Christian, Lee, Stephen, Golab, Lukasz, and Keshav, Srinivasan. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. *arXiv preprint arXiv:1901.00910* (2019).

[47] Gregoratti, D, and Matamoros, J. Distributed energy trading: The multiple-microgrid case. *IEEE Transactions on Industrial Electronics* (2015).

[48] Gulin, Marko, Vašak, Mario, and Baotic, Mato. Estimation of the global solar irradiance on tilted surfaces. In *17th International Conference on Electrical Drives and Power Electronics (EDPE 2013)* (2013).

[49] Gupta, Vani, Lee, Stephen, Shenoy, Prashant, Sitaraman, Ramesh, and Urgaonkar, Rahul. Towards cooling internet-scale distributed networks on the cheap. In *ACM SIGMETRICS Performance Evaluation Review* (2015), vol. 43, ACM, pp. 469–470.

[50] Gupta, Vani, Lee, Stephen, Shenoy, Prashant, Sitaraman, Ramesh K, and Urgaonkar, Rahul. How to cool internet-scale distributed networks on the cheap. In *Proceedings of the Seventh International Conference on Future Energy Systems* (2016), ACM, p. 9.

[51] Han, J, Choi, M, Lee, I, and Kim, S. Photovoltaic energy sharing system in a multi-family residential house to reduce total energy costs. In *IEEE International Conference on Consumer Electronics (ICCE)* (2016).

[52] He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, and Girshick, Ross. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)* (2017), IEEE.

[53] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016).

[54] He, M, Reutzel, E, Jiang, X, Katz, R, Sanders, S, Culler, D, and Lutz, K. An architecture for local energy generation, distribution, and sharing. In *Energy 2030 Conference* (2008).

[55] Hohm, DP, and Ropp, ME. Comparative study of maximum power point tracking algorithms using an experimental, programmable, maximum power point tracking test bed. In *Photovoltaic Specialists Conference, 2000. Conference Record of the Twenty-Eighth IEEE* (2000).

[56] Huang, Z, Zhu, T, Gu, Y, and Li, Y. Shepherd: sharing energy for privacy preserving in hybrid AC-DC microgrids. In *Proceedings of the Seventh International Conference on Future Energy Systems* (2016), ACM.

[57] Huertas, Andres, and Nevatia, Ramakant. Detecting buildings in aerial images. *Computer vision, graphics, and image processing* (1988).

[58] Hyperledger Fabric. [FAB-12221] Validator/Committer refactor - Hyperledger JIRA.

[59] IBC, ICC. International building code. *International Code Council, Inc.(formerly BOCA, ICBO and SBCCI)* (2006).

[60] Irwin, David, Iyengar, Srinivasan, Lee, Stephen, Mishra, Aditya, Shenoy, Prashant, and Xu, Ye. Enabling distributed energy storage by incentivizing small load shifts. vol. 1, ACM, p. 10.

[61] Iyengar, S., Lee, S., Irwin, D., and Shenoy, P. Analyzing energy usage on a city-scale using utility smart meters. In *Proceedings of the ACM International Conference on Systems for Energy Efficient Build Environments* (November 2016).

[62] Iyengar, S, Sharma, N, Irwin, D, Shenoy, P, and Ramamritham, K. SolarCast: a cloud-based black box solar predictor for smart homes. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings* (2014).

[63] Iyengar, Srinivasan, Lee, Stephen, Irwin, David, and Shenoy, Prashant. Analyzing energy usage on a city-scale using utility smart meters. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments* (2016), ACM, pp. 51–60.

[64] Iyengar, Srinivasan, Lee, Stephen, Irwin, David, Shenoy, Prashant, and Weil, Benjamin. Watthome: A data-driven approach for energy efficiency analytics at city-scale. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), ACM, pp. 396–405.

[65] Iyengar, Srinivasan, Lee, Stephen, Sheldon, Daniel, and Shenoy, Prashant. Solarclique: Detecting anomalies in residential solar arrays. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies* (2018), ACM, p. 38.

[66] Jacobson, V., and Karels, M. Congestion Avoidance and Control. In *SIGCOMM* (August 1998).

[67] Jakubiec, J A, and Reinhart, C F. Towards validated urban photovoltaic potential and solar radiation maps based on lidar measurements, gis data, and hourly daysim simulations. *IBPSA-USA Journal* (2012).

[68] Jochem, A, Höfle, B, Rutzinger, M, and Pfeifer, N. Automatic roof plane detection and analysis in airborne lidar point clouds for solar potential assessment. *Sensors* (2009).

[69] Kanoria, Y, Montanari, A, Tse, S, and Zhang, B. Distributed storage for intermittent energy sources: Control design and performance limits. In *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2011), IEEE.

[70] Karmakar, G, Kabra, A, and Ramamritham, K. Maintaining thermal comfort in buildings: feasibility, algorithms, implementation, evaluation. *Real-Time Systems* (2015).

[71] Kelly, F., Maulloo, A., and Tan, D. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* (1998).

[72] Kelly, F., and Yudovina, E. *Stochastic networks*, vol. 2. Cambridge University Press, 2014.

[73] Keshav, S. Technical Perspective The Chemistry of Software-Defined Batteries, 2016.

[74] Krizhevsky, A, Sutskever, I, and Hinton, G. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012).

[75] Kuthanazhi, Vivek, Jois, Santhosh, Jadhav, Prachi, Kumar, Kamlesh, Magal, Akhilesh, Pimpalkhare, Ameya, Vasi, Juzer, Kottantharayil, Anil, Ramamritham, Krithi, Narayanan, NC, et al. Estimating mumbai's rooftop pv potential through mobilization of ieee student community. In *Photovoltaic Specialists Conference (PVSC)* (2016), IEEE.

[76] Lam, A.H, Yuan, Y, and Wang, D. An occupant-participatory approach for thermal comfort enhancement and energy conservation in buildings. In *Proceedings of the 5th international conference on Future energy systems* (2014).

[77] Lee, Stephen, Iyengar, Srinivasan, Feng, Menghong, Shenoy, Prashant, and Maji, Subhransu. DeepRoof: A Data-driven Approach For Solar Potential Estimation Using Rooftop Imagery.

[78] Lee, Stephen, Iyengar, Srinivasan, Irwin, David, and Shenoy, Prashant. Shared solar-powered ev charging stations: Feasibility and benefits. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)* (2016), IEEE, pp. 1–8.

[79] Lee, Stephen, Iyengar, Srinivasan, Irwin, David, and Shenoy, Prashant. Distributed rate control for smart solar arrays. In *Proceedings of the Eighth International Conference on Future Energy Systems* (2017), ACM, pp. 34–44.

[80] Li, Na, Chen, Lijun, and Low, Steven H. Optimal demand response based on utility maximization in power networks. In *Power and Energy Society General Meeting* (2011), IEEE.

[81] Li, Na, Zhao, Changhong, and Chen, Lijun. Connecting automatic generation control and economic dispatch from an optimization view. *IEEE Transactions on Control of Network Systems* (2016).

[82] Li, Zhetao, Kang, Jiawen, Yu, Rong, Ye, Dongdong, Deng, Qingyong, and Zhang, Yan. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE transactions on industrial informatics 14*, 8 (2017), 3690–3700.

[83] Lin, Tsung-Yi, Dollár, Piotr, Girshick, Ross, He, Kaiming, Hariharan, Bharath, and Belongie, Serge. Feature pyramid networks for object detection. In *CVPR* (2017).

[84] Liu, B, and Jordan, R. Daily insolation on surfaces tilted towards equator. *ASHRAE Journal* (1961).

[85] Liu, J, Zhang, N, Kang, C, Kirschen, D S, and Xia, Q. Decision-making models for the participants in cloud energy storage. *IEEE Transactions on Smart Grid* (2017).

[86] Liu, Jingkun, Zhang, Ning, Kang, Chongqing, Kirschen, Daniel, and Xia, Qing. Cloud energy storage for residential and small commercial consumers: A business case study. *Applied energy* (2017).

[87] Liu, N, Yu, X, Wang, C, Li, C, Ma, L, and Lei, J. An energy sharing model with price-based demand response for microgrids of peer-to-peer prosumers. *IEEE Transactions on Power Systems* (2017).

[88] Livingston, David, Sivaram, Varun, Freeman, Madison, and Fiege, Maximilian. Applying block chain technology to electric power systems. Tech. rep., Discussion Paper. Council on Foreign Relations, 2018.

[89] Lo, C., and Ansari, N. Alleviating solar energy congestion in the distribution grid via smart metering communications. *IEEE Transactions on Parallel and Distributed Systems* (2012).

[90] Long, J, Shelhamer, E, and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2015).

[91] Lorensen, William E, and Cline, Harvey E. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics* (1987), vol. 21, ACM, pp. 163–169.

[92] Low, S., and Lapsley, D. Optimization flow control-I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking 7*, 6 (1999).

[93] Malof, J M, Hou, R, Collins, L M, Bradbury, K, and Newell, R. Automatic solar photovoltaic panel detection in satellite imagery. In *International Conference on Renewable Energy Research and Applications* (2015), IEEE.

[94] Margolis, Robert, Gagnon, Pieter, Melius, Jennifer, Phillips, Caleb, and Elmore, Ryan. Using gis-based methods and lidar data to estimate rooftop solar technical potential in us cities. *Environmental Research Letters* (2017).

[95] Meeuw, Arne, Schopfer, Sandro, Ryder, Benjamin, and Wortmann, Felix. Lokalpower: Enabling local energy markets with user-driven engagement. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), ACM, p. LBW613.

[96] Mengelkamp, Esther, Gärttner, Johannes, Rock, Kerstin, Kessler, Scott, Orsini, Lawrence, and Weinhardt, Christof. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy 210* (2018), 870–880.

[97] Mengelkamp, Esther, Notheisen, Benedikt, Beer, Carolin, Dauer, David, and Weinhardt, Christof. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science-Research and Development 33*, 1-2 (2018), 207–214.

[98] Mishra, A, Irwin, D, Shenoy, P, Kurose, J, and Zhu, T. Greencharge: Managing renewable energy in smart buildings. *IEEE Journal on Selected Areas in Communications* (2013).

[99] Mishra, Aditya, Sitaraman, Ramesh, Irwin, David, Zhu, Ting, Shenoy, Prashant, Dalvi, Bhavana, and Lee, Stephen. Integrating energy storage in electricity distribution networks. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems* (2015), ACM, pp. 37–46.

[100] Mnatsakanyan, A, and Kennedy, S W. A novel demand response model with an application for a virtual power plant. *IEEE Transactions on Smart Grid* (2015).

[101] Mnih, Volodymyr, and Hinton, Geoffrey E. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)* (2012).

[102] Nakamoto, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org* (2008), 9.

[103] Narayanaswamy, B, Garg, V K, and Jayram, TS. Online optimization for the smart (micro) grid. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where energy, computing and communication meet* (2012).

[104] Nasir, Qassim, Qasse, Ilham A, Abu Talib, Manar, and Nassif, Ali Bou. Performance analysis of hyperledger fabric platforms. *Security and Communication Networks 2018* (2018).

[105] Nayar, Chemmangot V, Ashari, Mochamad, and Keerthipala, WWL. A grid-interactive photovoltaic uninterruptible power supply system using battery storage and a back up diesel generator. *IEEE Transactions on Energy Conversion* (2000).

[106] Nordrum, Amy. Popular internet of things forecast of 50 billion devices by 2020 is outdated. *IEEE Spectrum 18* (2016).

[107] Noronha, Sanjay, and Nevatia, Ramakant. Detection and modeling of buildings from multiple aerial images. *IEEE Transactions on pattern analysis and machine intelligence* (2001).

[108] NREL PVWatts Calculator. `http://pvwatts.nrel.gov/`, Accessed March 2017.

[109] Oudalov, Alexandre, Cherkaoui, Rachid, and Beguin, Antoine. Sizing and optimal operation of battery energy storage system for peak shaving application. In *Power Tech, 2007 IEEE Lausanne* (2007), IEEE.

[110] Palensky, P., and Dietrich, D. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE transactions on industrial informatics* (2011).

[111] Pan, D, Wang, D, Cao, J, Peng, Y, and Peng, X. Minimizing building electricity costs in a dynamic power market: Algorithms and impact on energy conservation. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th* (2013), IEEE.

[112] Pan, J, Jain, R, Paul, S, Vu, T, Saifullah, A, and Sha, M. An internet of things framework for smart energy in buildings: designs, prototype, and experiments. *IEEE Internet of Things Journal* (2015).

[113] Pongnumkul, Suporn, Siripanpornchana, Chaiyaphum, and Thajchayapong, Suttipong. Performance analysis of private blockchain platforms in varying workloads. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on* (2017), IEEE, pp. 1–6.

[114] Project Sunroof. https://www.google.com/get/sunroof/data-explorer/data-explorer-methodology.pdf, Accessed March 2017.

[115] Solar roof. https://www.tesla.com/solarroof, Accessed May 2018.

[116] Suneye 210 shade tool. http://www.solmetric.com/, Accessed March 2018.

[117] Raman, Ravi Kiran, Vaculin, Roman, Hind, Michael, Remy, Sekou L, Pissadaki, Eleftheria K, Bore, Nelson Kibichii, Daneshvar, Roozbeh, Srivastava, Biplav, and Varshney, Kush R. Trusted multi-party computation and verifiable simulations: A scalable blockchain approach. *arXiv preprint arXiv:1809.08438* (2018).

[118] Rigas, E S, Ramchurn, S D, Bassiliades, N, and Koutitas, G. Congestion management for urban ev charging systems. In *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on* (2013), IEEE.

[119] Rongali, S., Ganuy, T., Padmanabhan, M., Arya, V., Kalyanaraman, S., and Petra, M. iPlug: Decentralised dispatch of distributed generation. In *COMSNETS* (January 2016).

[120] Ronneberger, O, Fischer, P, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[121] Rottensteiner, Franz. Automatic generation of high-quality building models from lidar data. *IEEE Computer Graphics and Applications* (2003).

[122] Saeedi, Parvaneh, and Zwick, Harold. Automatic building detection in aerial and satellite images. In *10th International Conference on Control, Automation, Robotics and Vision. ICARCV* (2008), IEEE.

[123] Samadi, P, Wong, V WS, and Schober, R. Load scheduling and power trading in systems with high penetration of renewable energy resources.

[124] Sharma, Ankur, Schuhknecht, Felix Martin, Agrawal, Divya, and Dittrich, Jens. How to databasify a blockchain: the case of hyperledger fabric. *arXiv preprint arXiv:1810.13177* (2018).

[125] Sharma, Prateek, Lee, Stephen, Guo, Tian, Irwin, David, and Shenoy, Prashant. Spotcheck: Designing a derivative iaas cloud on the spot market. In *Proceedings of the Tenth European Conference on Computer Systems* (2015), ACM, p. 16.

[126] Sharma, Prateek, Lee, Stephen, Guo, Tian, Irwin, David, and Shenoy, Prashant. Managing risk in a derivative IaaS cloud. *IEEE Transactions on Parallel and Distributed Systems 29*, 8 (2017), 1750–1765.

[127] Singh, A., Lee, S., Irwin, D., and Shenoy, P. SunShade: Software-defined Solar Power. In *Proceedings of the 8th ACM/IEEE International Conference on Cyber-Physical Systems* (2017).

[128] Singh, Akansha, Lee, Stephen, Irwin, David, and Shenoy, Prashant. Sunshade: software-defined solar systems. In *Proceedings of the Seventh International Conference on Future Energy Systems Poster Sessions* (2016), ACM, p. 8.

[129] Singh, Akansha, Lee, Stephen, Irwin, David, and Shenoy, Prashant. Sunshade: enabling software-defined solar-powered systems. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)* (2017), IEEE, pp. 61–70.

[130] Sousa, Joao, Bessani, Alysson, and Vukolic, Marko. A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2018), IEEE, pp. 51–58.

[131] Thakkar, Parth, Nathan, Senthil, and Vishwanathan, Balaji. Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. *arXiv* (2018).

[132] Thakur, S, Saha, M, Singh, A, and Agarwal, Y. Wattshare: Detailed energy apportionment in shared living spaces within commercial buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings* (2014).

[133] Tonkoski, R., Lopes, L., and El-Fouly, T. Coordinated active power curtailment of grid connected pv inverters for overvoltage prevention. *IEEE Transactions on Sustainable Energy* (April 2011).

[134] Verma, Vivek, Kumar, Rakesh, and Hsu, Stephen. 3d building detection and modeling from aerial lidar data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2006), IEEE.

[135] Vovos, P N, Kiprakis, A E, Wallace, A R, and Harrison, G P. Centralized and distributed voltage control: Impact on distributed generation penetration. *IEEE Transactions on Power Systems* (2007).

[136] Vukolić, Marko. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security* (2015), Springer, pp. 112–125.

[137] Vukolić, Marko. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts* (2017), ACM, pp. 3–7.

[138] Wamburu, John, Lee, Stephen, Shenoy, Prashant, and Irwin, David. Analyzing distribution transformers at city scale and the impact of evs and storage. In *Proceedings of the Ninth International Conference on Future Energy Systems* (2018), ACM, pp. 157–167.

[139] Wang, H, Zhang, J X, and Li, F. Incentive mechanisms to enable fair renewable energy trade in smart grids. In *Sixth International Green Computing Conference and Sustainable Computing Conference (IGSC)* (2015).

[140] Wegner, Jan D, Branson, Steven, Hall, David, Schindler, Konrad, and Perona, Pietro. Cataloging public objects using aerial and street-level images-urban trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[141] Wei, P, Chen, X, Chandrasekaran, R, Song, F, and Jiang, X. Adaptive and Personalized Energy Saving Suggestions for Occupants in Smart Buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments* (2016).

[142] Wei, Yanfeng, Zhao, Zhongming, and Song, Jianghong. Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection. In *IEEE International Geoscience and Remote Sensing Symposium* (2004), Ieee.

[143] Weng, T, and Agarwal, Y. From buildings to smart buildings-sensing and actuation to improve energy efficiency. *IEEE Design & Test of Computers* (2012).

[144] Wood, Gavin. Ethereum: a Secure Decentralised Generalised Transaction Ledger. *Yellow Paper* (2014).

[145] Wu, C, Kalathil, D, Poolla, K, and Varaiya, P. Sharing electricity storage. In *55th Conference on Decision and Control (CDC)* (2016), IEEE.

[146] Yin, Maofan, Malkhi, Dahlia, Reiter, Michael K, Golan Gueta, Guy, and Abraham, Ittai. HotStuff: BFT Consensus in the Lens of Blockchain. *arXiv preprint arXiv:1803.05069* (2018).

[147] Zhao, S., Lin, X., and Chen, M. Robust online algorithms for peak-minimizing EV charging under multi-stage uncertainty. Tech. rep., Available online: https://goo.gl/vtNHq0.

[148] Zhong, W, Huang, Z, Zhu, T, Gu, Y, Zhang, Q, Yi, P, Jiang, D, and Xiao, S. ides: Incentive-driven distributed energy sharing in sustainable microgrids. In *Green Computing Conference (IGCC), 2014 International* (2014).

[149] Zhu, T, Huang, Z, Sharma, A, Su, J, Irwin, D, Mishra, A, Menasche, D, and Shenoy, P. Sharing renewable energy in smart microgrids. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)* (2013).