UNIVERSITY OF CALIFORNIA, SAN DIEGO

A Context-Aware Approach for Automation of End-User Elements in the Smart Grid

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Jagannathan Venkatesh

Committee in charge:

Professor Tajana Šimunić Rosing, Chair
Professor Scott Baden
Professor Carlos Coimbra
Professor Rajesh Gupta
Professor Dean Tullsen

2016

The dissertation of Jagannathan Venkatesh is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2016

DEDICATION

To Sarah for her patience, to Callie for her support, and to Mohan for the motivation to finish.

To Patti.

EPIGRAPH

Whatever it is you seek, you have to put in the time, the practice, the effort. You must give up a lot to get it. It has to be very important to you. And once you have attained it, it is your power. It can't be given away: it resides in you. It is literally the result of your discipline.

Raptors are smart. Very smart.

—Michael Crichton, *Jurassic Park*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to take this opportunity to thank everyone who supported me during my PhD process.

First, thanks to my advisor Prof. Tajana Šimunić Rosing for her guidance over the past six years. She gave me the opportunity to pursue my goals, made me a better student, teacher, and researcher, and guided my career. I also want to thank the rest of my doctoral committee: Prof. Scott Baden, Prof. Carlos Coimbra, Prof. Rajesh Gupta, and Prof. Dean Tullsen for their feedback and support.

My research was made possible by funding from National Science Foundation (NSF) Project GreenLight Grant 1036931, NSF ERC CIAN Grant 812072, the Multiscale Systems Center (MuSyC), NSF CitiSense Grant 0932403, the Terraswarm Research Center, UCSD Center for Networked Systems (CNS), and NSF MetaSense Grant 1446912, and corporate support from Oracle, Panasonic, Qualcomm, and Google. I thank them for their generous support.

None of my research was done in isolation, and my colleagues and co-authors are also my friends. They have provided guidance, discussion, reinforcement, and feedback throughput my PhD, and made me a better writer, presenter, and researcher. I owe special thanks to Bariş Akşanli, Alper Sinan Akyürek, and Christine Chan.

My parents, Venkatesh and Viji Chari and Paula and Joseph Peeden, and my brothers and sisters Joe, Preston, Alex, and Vaishnavi, have been a source of unconditional love and support. I cannot express enough how much I am indebted to them.

Finally, I want to thank Sarah. She helped make me the person I am today, and she is my motivation to be the best person I can be.

Chapters 1 and 2 contain material from "Using datacenter simulation to evaluate green energy integration", by Bariş Akşanli, Jagannathan Venkatesh and Tajana Šimunić Rosing, which appears in IEEE Computer 45, September 2012 [25]. The dissertation author was one of the primary investigators and the second

author of this paper.

Chapters 1 and 2 contain material from "Renewable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks", by Bariş Akşanli, Jagannathan Venkatesh, Tajana Šimunić Rosing, and Inder Monga, which appears in Computational Sustainability, Springer, 2015 [26]. The dissertation author was author was one of the primary investigators and second author of this paper.

Chapters 1 and 3 contain material from Jagannathan Venkatesh, Bariş Akşanli, and Tajana Šimunić Rosing, "Residential Energy Simulation and Scheduling: A Case Study Approach", which appeared in Proceedings of the International Symposium on Computers and Communications (ISCC), 2013 [93]. The dissertation author was the primary investigator and author of this paper.

Chapters 1 and 3 contain material from Jagannathan Venkatesh, Bariş Akşanli, Jean-Claude Junqua, Philippe Morin, and Tajana Šimunić Rosing, "HomeSim: Comprehensive, Smart, Residential Electrical Energy Simulation and Scheduling", which appeared in Proceedings of the International Green Computing Conference (IGCC), 2013 [92]. The dissertation author was the primary investigator and author of this paper.

Chapters 1 and 3 contain material from Jagannathan Venkatesh, Shengbo Chen, Peerapol Tinnakornsrisuphap, and Tajana Šimunić Rosing, "Lifetime-dependent Battery Usage Optimization for Grid-Connected Residential Systems", which appeared in Proceedings of 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES) 2015 [95]. The dissertation author was the primary investigator and author of this paper.

Chapters 1 and 4 contain material from Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "A Modular Approach to Context-Aware IoT Applications", which appeared in Proceedings of the 1st Conference on Internet of Things Data and Implementation (IoTDI), 2016 [94]. The dissertation author was the primary investigator and author of this paper.

Chapters 4 and 5 contain material from Jagannathan Venkatesh, Bariş Akşanli, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "Scal-

able IoT Application Design for Automated Learning", which was submitted for consideration in IEEE Software, Special Issue on Software Engineering for the Internet of Things, 2016. The dissertation author was the primary investigator and author of this paper.

VITA

2008	B. S. in Electrical Engineering, Computer Science, University of Virginia
2008-2010	Software Development Engineer, Microsoft Corporation.
2010-2016	Graduate Research Assistant, University of California, San Diego
2010-2015	Software Engineering Intern, Google, Inc.
2013	Teaching Assistant, University of California, San Diego
2013	M. S. in Computer Science, University of California, San Diego
2014	Research Intern, Qualcomm Technologies, Inc.
2016	Ph. D. in Computer Science, University of California, San Diego

PUBLICATIONS

Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyürek and Tajana Šimunić Rosing. "A Modular Approach to Context-Aware IoT Applications", In *Proceedings of the 1st Conference on Internet of Things Data and Implementation (IoTDI)*. April 2016.

Bariş Akşanli, Jagannathan Venkatesh, Tajana Šimunić Rosing, and Inder Monga. "Renewable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks." Book Chapter, *Computational Sustainability*, Springer. 2015.

Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "A Context-Driven IoT Middleware Architecture", *SRC TechCon*. September 2015

Jagannathan Venkatesh, Shengbo Chen, Peerapol Tinnakornsrisuphap, Tajana Šimunić Rosing, "Lifetime-dependent Battery Usage Optimization for Grid-Connected Residential Systems", In *Proceedings of 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pp. 1-6. April 2015

Bariş Akşanli, Alper Sinan Akyürek, Madhur Behl, Meghan Clark, Alexandre Donze, Prabal Dutta, Patrick Lazik, Mehdi Maasoumy, Rahul Mangharam, Truong X. Nghiem, Vasumathi Raman, Anthony Rowe, Alberto Sangiovanni-Vincentelli, Sanjit Seshia, Tajana Šimunić Rosing, and Jagannathan Venkatesh. "Distributed control of a swarm of buildings connected to a smart grid: demo abstract." In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys 14)*. pp. 172-173. November 2014.

Bariş Akşanli, Jagannathan Venkatesh, Tajana Šimunić Rosing, and Inder Monga, "A comprehensive approach to reduce the energy cost of network of datacenters." In *Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC)*, pp.275-280, July 2013.

Jagannathan Venkatesh, Bariş Akşanli, and Tajana Šimunić Rosing, "Residential Energy Simulation and Scheduling: A Case Study Approach", In *Proceedings of the 2013 International Symposium on Computers and Communications (ISCC)*, pp. 161-166. July 2013

Jagannathan Venkatesh, Bariş Akşanli, Jean-Claude Junqua, Philippe Morin, and Tajana Šimunić Rosing, "HomeSim: Comprehensive, Smart, Residential Electrical Energy Simulation and Scheduling", In *Proceedings of the International Green Computing Conference (IGCC)*, pp. 1-8. June 2013

Mohammad Moghimi, Jagannathan Venkatesh, Piero Zappi and Tajana Šimunić Rosing, "Context-Aware Mobile Power Management Using Fuzzy Inference as a Service", In *Proceedings of the 4th International Conference on Mobile Computing, Applications, and Services (MobiCASE)*, pp. 314-327. October 2012

Bariş Akşanli, Jagannathan Venkatesh, and Tajana Šimunić Rosing, "Using Datacenter Simulation to Evaluate Green Energy Integration." *IEEE Computer*, vol.45, no.9, pp.56-64. September 2012.

Bariş Akşanli, Jagannathan Venkatesh, Liuyi Zhang, and Tajana Šimunić Rosing. "Utilizing green energy prediction to schedule mixed batch and service jobs in data centers." *ACM SIGOPS Operating Systems Review 45*, no.3 pp. 53-57. January 2012.

ABSTRACT OF THE DISSERTATION

A Context-Aware Approach for Automation of End-User Elements in the Smart Grid

by

Jagannathan Venkatesh

Doctor of Philosophy in Computer Science

University of California, San Diego, 2016

Professor Tajana Šimunić Rosing, Chair

The smart grid is driven by distributed, heterogeneous energy sources and controllable loads matched to sources to optimize energy usage. The introduction of distributed, heterogeneous sources provides a level of instability compared to the previously monolithic, centralized energy producers. This drives the need for accurate modeling of the energy grid, and distributed control of available loads to gracefully react to changes and maintain overall stability. Currently, this is accomplished by ubiquitous data collection of energy generation and consumption is used for prediction of sources and loads, which in turn is used for net metering, and controlling retail energy prices and loads. This thesis proposes using context — additional high-level information — about various elements of the smart grid

(sources, loads, and storage) to improve the efficiency of its operation.

We first investigate the integration of local renewables in data centers, some of the largest consumers in the grid. We mitigate the variability of renewables with short-term prediction of solar and wind energy using additional environmental data. This allows us to use renewable energy for computing loads, the largest segment of energy consumption while maintaining quality of service requirements of the data center. We extend this work to a networked set of data centers, allocating and migrating jobs to improve the efficiency of use of available green energy to over 90%.

While individual large consumers are of particular importance to the grid, there is relatively little focus on automated regulation of smaller consumers like individual houses, despite the residential sector consuming over 1/3 of national energy consumption. We propose a residential electrical energy simulation platform that enables investigating the impact of technologies such as renewable energy, different battery types, centralized vs. distributed in-home energy storage, and smart appliances. We further develop a formulation for battery usage based on more realistic battery models, demonstrating an improvement in modeling accuracy by over 2x. We can solve our new battery model to optimize the benefit of discharging the battery and minimizing the time of return on investment. We simulate the new optimization's results using HomeSim, demonstrating a recoupment of battery costs in as few as 5 years.

Finally, residential energy is directly driven by the activities and behavior of the users. The growing advent of sensing in the IoT presents a unique opportunity: using general-purpose modular data processing to generate grid-related context: energy prediction and flexibility. This contextual information can be used to drive individualized automated actuation that scales to the size of the grid. We demonstrate that user context can provide up to 86% accuracy in energy flexibility prediction, a 14% improvement over the current state of the art of not having user data.

Chapter 1

Introduction

The smart grid consists of distributed energy resources; a hierarchical, scalable infrastructure; and instrumented and controllable end-use elements (loads, sources, and hybrid elements like batteries). Analysis of the available data provide prediction of energy generation and consumption and enable use of the heterogeneous elements in more complex ways while maintaining energy efficiency and avoiding critical and expensive scenarios. Figure 1.1 below illustrates the growing complexity of smart grid elements and infrastructure. New source-side elements include distributed renewable energy sources, which alleviate concerns over the growing costs, increasing emissions output, and the limited nature of non-renewable energy sources [62]. Consequently, renewables such as solar-electric and wind have demonstrated over 25% growth in capacity annually [99] [39]. On the infrastructure and load side, new technology includes backfeeding energy generated by end users, distributed energy storage resources like batteries, and complex controllable loads such as smart appliances. Grid efficiency is achieved through new control scenarios exploiting characteristics of the new elements to maintain the stability and efficiency of the smart grid: much tighter source-load matching through metering and grid automation, efficiency of distributed renewable energy sources, more appropriate real-time pricing based on usage, etc.

Renewable sources, in particular solar and wind, are unlike traditional sources in that they are directly at the mercy of the elements: e.g. solar irradiance, wind speed, wind direction, and various other environmental factors. This

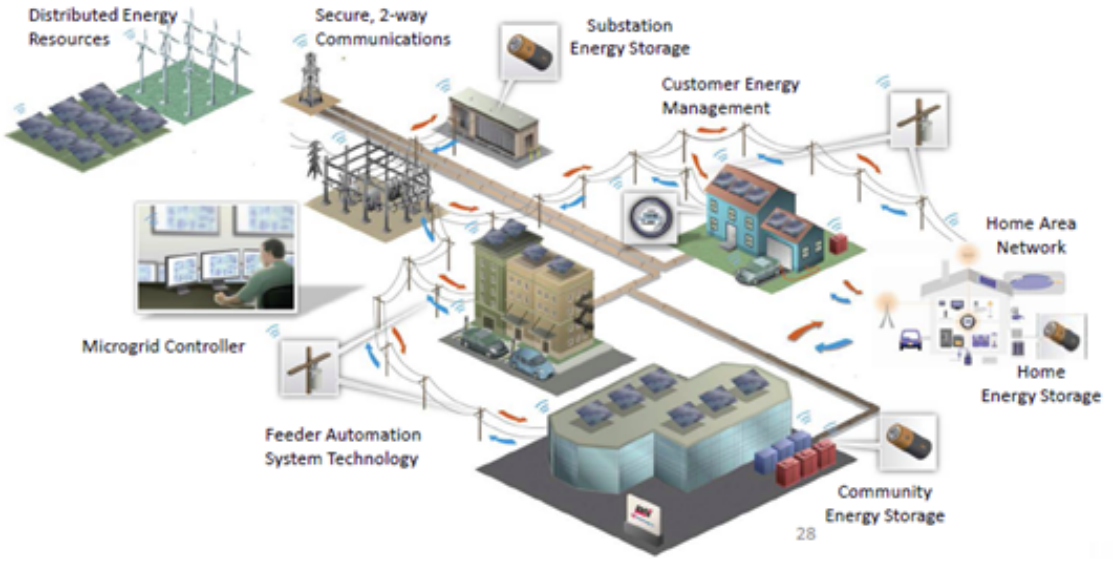

Figure 1.1: Growing complexity of smart grid elements (sources, loads, and hybrid nodes), infrastructure, and scenarios. [2].

precludes their use as primary means of energy generation without taking additional precautions: overprovisioning them to overcome low-output periods [20], supplementing or switching to non-renewable sources [72], adding energy storage to the system[73], or shifting loads to meet availability [61]. Large loads such as data centers can have significant impact on grid stability, but may incorporate on-site renewables and storage to mitigate expensive peak-power conditions. Utilities and retail energy providers also contract automated net metering with these consumers: control over some non-essential loads that can be limited or shut off in order to reduce expensive peak-power conditions [48].

Granular load control and net metering is not integrated to the same extent in residential nodes despite them composing 38% of national energy consumption [9]. This is in part due to individual houses posing an insignificant contribution towards grid stability, and the relative complexity of individual residential energy usage and preferences among users and households. Additionally, the lack of integration of energy storage and automated loads at the residential level has precluded more complex interactions like those found in data centers and other large consumers. In the current state of the art, energy prediction is done primarily at the

macro scale: of neighborhoods and residential districts using smart meter data to determine when to turn on and off grid-level energy sources and control wholesale and retail pricing. However, the advent of the Internet of Things has resulted in more useful residential elements: from smart, connected appliances to home energy storage [93], the residential sector is gaining the level of automation previously only seen in large consumers such as data centers. However, unlike data centers, residential energy use is more strongly driven by individual users' behavior.

In this thesis, we propose the use of different types of context — high-level, abstracted information as opposed to raw data — to handle issues with two common types of smart grid end users: data centers and residential nodes. We first identify issues in both sectors that are not addressed by the current state of the art, and select supplementary context to help mitigate these issues. For data centers with on-site renewables, source output variability creates difficulty in driving their primary energy loads: compute workloads with tight timing constraints. We identify additional data to better predict renewable, or "green" energy availability and exploit characteristics of the workloads to improve the efficiency of green energy use. For residences, we explore the new scenarios enabled by the growth in technology in the residential sector: smart appliances with flexible schedules, distributed energy resources, and cost-aware rescheduling. We design a modular residential energy simulation platform — HomeSim — that is capable of modeling these scenarios. We then identify the additional context required to realize them and create event-driven schedulers to use with HomeSim that demonstrate residential behavior improvements from each. Finally, we explore how the residential smart grid is representative of a context-aware Internet of Things (IoT) platform: a combination of environmental and user context used to drive output actuation. We identify that data-specific implementations of grid actuation do not scale to the level of the smart grid, and instead propose a modular approach to context-aware IoT applications, facilitating general-purpose machine learning and demonstrating latency and scalability improvements with limited impact on output prediction accuracy. Our final case study connects our approach to the grid, predicting residential energy usage, individual and aggregate energy flexibility in the residential

sector. We demonstrate that user context has over 21% correlation to energy and flexibility prediction, and can improve prediction accuracy by over 14%.

1.1 Renewable Energy and Context in Data Centers

Data centers are one of the largest individual energy consumers in the smart grid, using over 3% of U.S. electricity consumption [7]. The expense and unsustainability of non-renewable sources has pushed data centers owners, including Facebook, Google, Amazon, and Apple to actively integrate green energy sources into their data center installations [55]. While data centers have the spatial and infrastructure requirements to support on-site renewables such as solar and wind, these sources do not have the consistent output of non-renewables due to their dependence on environmental conditions. This compromises the ability for data centers to complete their latency-constrained service workloads. These workloads cannot be halted for lack of energy availability of green sources without affecting a data center's quality of service (QoS) guarantees.

Existing data centers have used different approaches to circumvent availability: overprovisioning renewable sources [20], supplementing green energy with non-renewable sources [55], and using renewables for secondary or less crucial loads such as lighting or airflow [71]. Other research has provided complementary operational solutions: server power capping [48], minimizing the cost of supplemental non-renewables through price-aware scheduling [37], or forcing loads to follow energy availability [61]. These techniques come with associated costs: either physical and financial costs for additional infrastructure and purchased energy, or at the cost of not meeting the data center's service-level agreement (SLA) for the short-running service workloads of data centers. Furthermore, they are difficult to use for the longer-running but non-SLA batch workloads precisely because energy availability is variable. One approach that has been largely overlooked by data centers is green energy prediction, which is used in other domains for developing hybrid energy systems (embedded platforms) and energy pricing (utility grid) [27]. Short-

term prediction of energy availability can provide better matching of batch workloads to the available energy. Appropriately distributing batch workloads through a data center while accounting for their impact on running service workloads can ensure that green energy use is improved while the SLA is not impacted.

1.2 Modeling Residential Energy Management in the Smart Grid

Building energy consumption has been well-researched, but the focus has been on commercial and industrial domains, which constitute a larger fraction of global energy consumption. However, the residential domain contributes over a third of the total energy consumption in the United States [9]. Moreover, the residential sector is important because of the significantly higher number of end users impacted: in the United States alone, residential energy consumption impacts hundreds of millions of homes and other residences. As such, recent research has turned to improving residential energy use.

Related research focuses on reducing a single aspect of consumption [98], supplementing a residence with energy storage [73], or providing more granular energy information to end-users in order to facilitate user-driven improvements [60]. These use cases in these works signify the growing number of complex scenarios that are now possible in residential energy consumption, and highlight the importance of being able to test their benefits, including combining different cases to determine compounded energy usage improvements. These scenarios have been tested using independent data collection and modeling, and the results cannot be quantitatively compared.

Home energy simulators and testbeds have been developed as part of other research endeavors, but they are either specific to a single residence or scenario [23] [9], or particular to one aspect of energy consumption (e.g. heating, ventilation and cooling systems). There is a lack of a general-purpose residential modeling platform that can represent different types of houses. The different types of elements in a residence would need to be accurately represented and extensible to handle

new technology, such as local energy storage, rooftop solar, and direct-current (DC) circuits. Furthermore, technological advances open up the possibility of more complex usage patterns, including adjusting to grid-provided time-of-use pricing, peak power and energy restrictions, and user flexibility. A comprehensive residential energy simulator should be able to simulate these scenarios both individually and in combinations.

One of the scenarios we explore using complex end-use elements in residences is distributed energy storage. Batteries are an important element for residences in grid-connected systems that have on-site renewable energy generation, as they can help overcome the variability of renewables [73]. However, batteries have nonlinear charging, discharging, and degradation properties that make it difficult to optimize their usage [58]. While the nonlinear models of chemical batteries have been extensively studied, related work on optimizing residential battery use opt for less accurate linear models [73], which leads to overestimating the benefits of batteries and inaccurate quantification of their benefits.

1.3 Improving Residential Energy Modeling with User Context

Unlike the industrial sector, residential energy consumption is directly driven by individual user behavior [26]. Furthermore, the complex scenarios enabled by technology improvements and the smart grid depend on user context such as activity, location, usage preferences, and interaction with appliances. The residential smart grid embodies the Internet of Things (IoT): an infrastructure of sensing and actuation devices connected to a distributed backend storage and processing infrastructure [76]. Furthermore, the advent of the IoT provides a readily available source of user data from wearables and other web-connected user devices. In addition to user tracking, the IoT has also stimulated the growth of connected appliances and home monitoring products such as standalone devices [16] or integrated solutions [18].

The combination of user tracking and smart residential elements provides

the foundation for a prototypical context-aware IoT application: integrating high-level information about a space and the users within it to provide high-level context as output. In our case, we use activity and other user-derived context with energy traces to more accurately predict residential energy and flexibility, which in turn is used to improve residential energy management. However, the current state of the art of IoT applications is complete end-to-end applications that are dependent on the sensors and actuators for which they were designed [76]. As the number and heterogeneity of sensing devices for each application changes, these tightly-coupled applications do not promote adaptation to the changing amount and sources of data or available compute nodes. For the residential grid, this means that every house with a different combination of grid elements and user context requires a custom application. Approaching context-aware IoT applications as a set of smaller, simpler functional units that provide intermediate steps towards the overall application can alleviate compute redundancy and scalability issues. While this approach may not match the efficiency of a customized black-box application, it provides an advantage in the dynamic IoT space for general implementations of context applications. Instead of placing the burden of processing on black-box applications [69], general-purpose machine learning can drive automation while customizing actuation to user behavior. Furthermore, the same overall implementation can be used with different sets of house, energy, and user context sources to improve the accuracy of complex residential energy scenarios such as local renewables, time-of-use pricing, and user-customized flexibility of smart appliances [92].

1.4 Thesis Contributions

This thesis focuses on using supplemental high-level data — context — to improve the efficiency of various end users of the smart utility grid. We investigate consumers that provide significant impact on the stability of the smart grid, from large industrial consumers with on-site renewables (data centers) and the user-behavior-driven residential sector, with negligible individual loads but significant

aggregate consumption. The following breakdown highlights the contributions in the rest of the thesis:

- We develop novel, low-overhead, flexible predictors for solar and wind energy prediction, specifically targeted for effective integration of green energy into data centers. We leverage readily available context for each predictor, including historical energy output and local wind speed and direction. We outline an experiment using the predictors in conjunction with output data from real solar and wind sources for our own data center simulator, demonstrating up to 3x green energy efficiency improvement. We extend this work by using green energy prediction in geographically distributed sites and varying brown energy prices to investigate job migration algorithm among data centers to reduce the overall cost of energy, obtaining 27% better batch job completion time compared to no migration with only a 6-12% increase in total energy cost. This study is presented in Chapter 2.

- We propose HomeSim, a residential electrical energy simulation platform that enables investigating the impact of technologies such as renewable energy and different battery types. HomeSim allows us to simulate different scenarios including centralized vs. distributed in-home energy storage, intelligent appliance rescheduling, and outage management. Using measured residential data, HomeSim quantifies different benefits for different technologies and scenarios, including up to 50% reduction in grid energy through a combination of distributed batteries and reschedulable appliances. We then identify an issue with current residential battery-based optimization: the simplified, linear approach is inaccurate compared to the true, nonlinear behavior of the battery. We verify this inaccuracy using HomeSim and develop a formulation for battery usage based on more realistic battery models, optimizing the benefit of discharging the battery. We design the scheme based on this updated model, optimizing the trading of battery capacity on the energy market to minimize the expected return on investment (RoI). HomeSim, the case studies, and residential battery management is presented as part of Chapter 3.

- We investigate the design of context-aware Internet of Things applications, identifying user-driven residential energy consumption as a characteristic example. We propose a modular approach to these context-aware applications, breaking down monolithic applications into an equivalent set of functional units, or context engines. By exploiting the characteristics of context-aware applications, context engines can reduce compute redundancy, computational complexity and scalability with a nominal impact on accuracy. We provide an example large-scale application residential smart grid control as a case study, improving energy prediction accuracy by 14%, and system scaling speedup by 30%. The design and implementation of the context engine and the user activity and grid flexibility case studies in Chapter 5.

Chapter 1 contains material from "Using datacenter simulation to evaluate green energy integration", by Bariş Akşanli, Jagannathan Venkatesh and Tajana Šimunić Rosing, which appears in IEEE Computer 45, September 2012 [25]. The dissertation author was one of the primary investigators and the second author of this paper.

Chapter 1 contains material from "Renewable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks", by Bariş Akşanli, Jagannathan Venkatesh, Tajana Šimunić Rosing, and Inder Monga, which appears in Computational Sustainability, Springer, 2015 [26]. The dissertation author was author was one of the primary investigators and second author of this paper.

Chapter 1 contains material from Jagannathan Venkatesh, Bariş Akşanli, and Tajana Šimunić Rosing, "Residential Energy Simulation and Scheduling: A Case Study Approach", which appeared in Proceedings of the International Symposium on Computers and Communications (ISCC), 2013 [93]. The dissertation author was the primary investigator and author of this paper.

Chapter 1 contains material from Jagannathan Venkatesh, Bariş Akşanli, Jean-Claude Junqua, Philippe Morin, and Tajana Šimunić Rosing, "HomeSim: Comprehensive, Smart, Residential Electrical Energy Simulation and Scheduling", which appeared in Proceedings of the International Green Computing Conference

(IGCC), 2013 [92]. The dissertation author was the primary investigator and author of this paper.

Chapter 1 contains material from Jagannathan Venkatesh, Shengbo Chen, Peerapol Tinnakornsrisuphap, and Tajana Šimunić Rosing, "Lifetime-dependent Battery Usage Optimization for Grid-Connected Residential Systems", which appeared in Proceedings of 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES) 2015 [95]. The dissertation author was the primary investigator and author of this paper.

Chapter 1 contains material from Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "A Modular Approach to Context-Aware IoT Applications", which appeared in Proceedings of the 1st Conference on Internet of Things Data and Implementation (IoTDI), 2016 [94]. The dissertation author was the primary investigator and author of this paper.

Chapter 2

Renewable Energy and Context in Data Centers

Data centers utilize context to schedule workloads and manage efficiency: job arrival rates, type, and overhead analysis help determine job allocation, distribution, etc. Similarly, context can help improve the efficiency of renewabale energy efficiency within data centers. With concerns over growing costs, increasing emissions output, and the limited nature of non-renewable energy sources, there has been a movement towards utilizing green energy for power generation. In particular, data centers, which are among the largest individual consumers in the electrical grid and facing an annual electricity growth rate of 15% [68], have a strong motivation to incorporate renewables. Several corporations, including Google, Emerson Network Power, AISO.net, and i/o Datacenters, have built solar-powered or solar-assisted data centers [70], while others (e.g. OWI [20], Facebook [45], Green House Data [21] are utilizing on-site or local wind power.

The main obstacle to widespread utilization of renewable energy sources is their output variability (Figure 2.1). Unlike traditional means of power generation, which produce consistent output, renewable energy output is very much dependent on environmental factors such as solar irradiance or wind speed. For data centers, this variability is compounded by the tight timing constraints of the primary jobs executed in data centers: service workloads. Currently, the integration of green energy sources is accompanied by large integration costs to offset the impact of

variability. Other works explore the use of renewable sources in conjunction with various attempts at mitigating its variability, such as by using battery storage [48]. A possible solution is the use of green energy prediction, which has been previously used to compensate for source variability in developing hybrid energy systems, the smart grid [79], and wireless sensor networks [43].

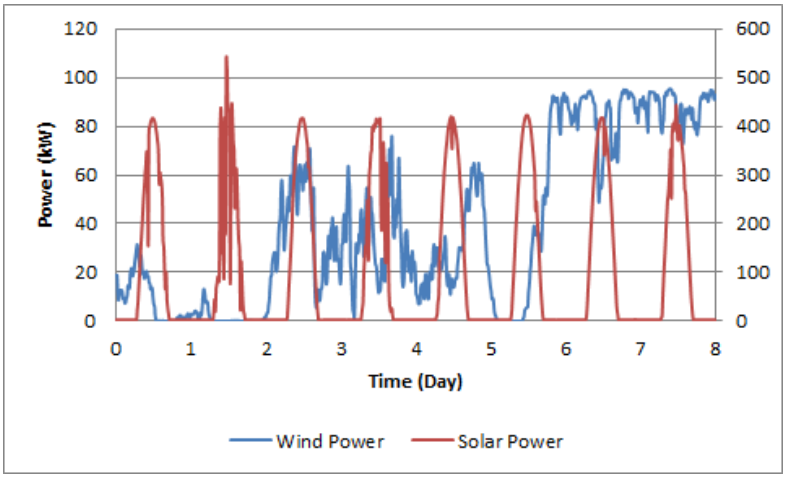

Figure 2.1: Variability of solar and wind energy sources from the UCSD grid and a Lake Benton, MN wind farm, respectively, over the course of a week. [91].

In this chapter, we outline an approach to integrating local renewables with data centers, improving the efficiency of green energy. We identify and develop short-term predictors for solar and wind energy, using readily-available context to improve prediction accuracy. With a baseline of utility-provided nominal energy, we ensure that service jobs complete within their service level agreement (SLA). We aim to maximize the utilization of available green energy by using them to run the other major workload in data centers: batch jobs. As batch jobs are longer-running, our renewable energy predictors allow us to appropriately allocate batch workloads for the energy available. Using prediction, we demonstrate over 90% green energy efficiency, up to 3x more than without prediction, and extend this work to a networked set of data centers, improving the average completion time of batch jobs by 27%.

2.1 Related Work

In addition to industrial implementations of green data centers described above, academic research in renewable energy integration focuses on investigating data center workload and power management in order to improve energy efficiency and mitigate green energy variability.

Gmach et al. [48] aim to reduce the peak power of a data center, a significant impact on data center runtime costs, by incorporating renewable energy from local solar and wind sources in conjunction with power-management algorithms. They investigate power capping, both of individual servers using dynamic frequency scaling, and of server pools by reducing the number of machines utilized in each pool. They then incorporate both energy storage and renewable energy sources to their management system, observing their impact on the power capping schemes. They note significant quality-of-service violations when limiting peak power, with over two orders of magnitude increase in violations per hour. Additionally, although using a local solar source helped improve green energy efficiency by reducing the need for grid power, the periodic unavailability of solar energy at night limited the reduction of peak power by 20kW compared with non-renewable sources alone. The use of a local wind installation showed negligible reduction of peak power because of the higher variability of wind sources. Only by utilizing battery storage could the variability effects be mitigated.

Le et al. [65] investigates the impact of capping non-renewable, or brown, energy needs in data centers. Cities such as Kyoto impose strict restrictions on brown energy use to limit the carbon impact of large energy consumers. The authors leverage the fact that internet services are distributed across multiple data centers, enabling them to schedule workloads based on local electricity prices or green energy availability. They define the workload distribution to each data center as a local optimization problem. By utilizing traces of brown energy consumption and electricity prices and predicting both for the upcoming 1-hour optimization interval using an autoregressive moving average (ARMA) model, the authors demonstrated 35% lower brown energy consumption with a nominal (10%) hit on service level agreement (SLA) violations.

Similarly, Buchbinder et al. [37] develop a solution for energy management in a cluster of data centers by optimizing for energy prices. Instead of just reducing brown consumption, they aim to reduce overall energy costs by distributing workloads to data centers with the lowest energy prices. An additional insight is that renewable energy sources have different cost traces than utility sources, as they are driven by environmental factors rather than peak load. Consequently, renewable sources such as solar energy are actually cheapest during the day, when workloads are at the highest, as opposed to utility sources, which are most expensive at those times. By assuming that energy costs remain fixed over short time intervals and associating a job capacity for each data center, the job migration problem can be reduced to an optimization problem. The algorithm identifies a local minimum energy cost among the available data centers that still meets bandwidth and latency characteristics for migrating and executing the job, and maintains the integrity of the power capacity of the selected data center. Although the proposed algorithm assumes knowledge of future energy prices, a slightly unreasonable expectation, the assumption can be justified by the use of prediction. The paper concludes by showing demonstrating on real data center cluster traces that their algorithm performs within 5.7% of the optimum distribution, as opposed to 10.4% and 68.5% of the optimum for established greedy algorithms.

Krioukov et al. [61] analyze the opportunities and difficulty of using supply-following loads to match renewable energy availability. They focus on utilizing recent green energy data to provision an appropriate amount of work within a data center. They describe two options for stopping data center workloads when instantaneous availability is insufficient: either by terminating or suspending the current workloads, then restarting or resuming them, respectively, when availability returns. However, due to the high variability of wind at any given time, and the constantly changing solar irradiance before and after peak availability, simulations of instantaneous usage in a supply-following capacity show very low green energy efficiency. In addition, supply-following workloads do not provide service-level guarantees that most data centers are required to provide.

The above data center examples demonstrate the high cost and necessary

precautions that must be taken in order to successfully use green energy, all of which reduce efficient utilization due to the high variability of renewable sources. While there are no published reports on efficiency of green energy in the implemented solar and wind installations above, our data center simulator demonstrates that the instantaneous use of renewable sources provides, on average, only 58% green energy efficiency. While the above examples implement different ways to help reduce the variability of green sources, an important means of reducing such variability has been unexplored in data centers: utilizing prediction. By forecasting the availability of renewables in the intervals ahead, data centers can better allocate their workloads to the available energy. Since data center workloads are relatively short-lived, we focus on predicting short-term availability of solar and wind sources. In particular, we focus on predicting up to 30 minutes ahead, as this is a typical time it takes for the data center's longer-running batch jobs such as MapReduce to complete [27].

2.2 Short-term Solar Energy Prediction

This section investigates the solar energy prediction algorithms among the related work, identified as applicable to the data center domain. Data centers use localized arrays of photovoltaic cells to obtain solar energy. This precludes the use of spatial and Numerical Weather Prediction (NWP) models, whose strengths are in expansive, low-granularity prediction. Time series models, however, use sequential output power readings to provide predictions that are as granular as the input data. Furthermore, solar energy has a periodic output due to its dependence on solar irradiance, which is subject to daily cycling. As such, the following subsections examine and quantitatively compare the state of the art in solar prediction using time-series models. In particular, we aim to predict for a time horizon appropriate to data center batch workloads, and as such, target short-term prediction.

2.2.1 Exponential Weighted Moving Average (EWMA)

The Exponential Weighted Moving Average (EWMA) algorithm has long been used for solar and weather forecasting [53], and represents the base case for our testing. The basic EWMA algorithm is described below:

$$X(i+1) = \alpha * X(i) + (1-\alpha) * x(i) \tag{2.1}$$

where $X(i)$ defines the prediction value for slot i, $x(i)$ denotes the measured value for slot i, and α represents the fixed weighting factor.

2.2.2 Extended EWMA (eEWMA)

The extended EWMA algorithm [43] goes one step further, incorporating data from slot $x(i-1)$ as well as compensating for the error of the previous and current slot:

$$X(i+1) = \alpha x(i) * (1-\epsilon_1) + (1-\alpha) * x(i-1) * (1-\epsilon_2) \tag{2.2}$$

with the error coefficients defined as follows:

$$\epsilon_1 = \frac{x(i) - X(i)}{x(i)}, \quad \epsilon_2 = \frac{x(i-1) - X(i-1)}{x(i-1)} \tag{2.3}$$

The algorithm can be similarly extended to encompass values from multiple previous days.

2.2.3 Weather-Conditioned Moving Average (WCMA)

Finally, the WCMA algorithm developed in [78] takes into account D number of days with N number of slots per day, allowing quicker calculation of both a baseline and adaptation to seasonal changes. The algorithm is defined as follows:

$$E(d, n+1) = \alpha * E(d, n) + GAP_k * (1-\alpha) * M_D(d, n+1) \tag{2.4}$$

Here again, α is the weighting factor. However, unlike the previous algorithms, which predict the value of an interval in a future day, the WCMA algorithm predicts a future interval $(n+1)$ of the current day (d). However, the algorithm introduces

several new factors to more accurately calculate $E(d, n+1)$, utilizing the mean of the previous D days' values for the slot to be predicted:

$$M_D(d,n) = \frac{\sum_{i=d-1}^{d-D} E(i,n)}{D} \tag{2.5}$$

as well as weighting the mean values by the distance to the point to be predicted (the GAP factor):

$$v_k = \frac{E(d,n-K+k-1)}{M_D(d,n-K+k-1)}, \quad \mathbf{V} = [v_1, v_2, ..., v_K]$$

$$p_k = \frac{k}{K}, \quad \mathbf{P} = [p_1, p_2, ..., p_K] \tag{2.6}$$

$$GAP_k = \frac{\mathbf{V}*\mathbf{P}}{\sum \mathbf{P}}$$

It is important to note the distinction between the WCMA algorithm and the previous EWMA implementations. Solar irradiance, in addition to a daily pattern, experiences seasonal variations, which are handled by the adaptive nature of all the EWMA algorithms. However, an individual day's weather conditions can significantly impact the daily irradiance. Piorno et al. address this issue with the GAP factor, which biases predictions by scaling the current day's observed energy over the previous D days. Thus, daily irradiance fluctuations are accounted for, and predictions are scaled appropriately.

2.2.4 Results

In order to select an effective green energy prediction algorithm, the above algorithms are tested under three conditions: consistent levels of solar irradiance, where the variation in output energy between days is below 25%; moderately inconsistent conditions, where the variation is between 25-50%; and severely inconsistent conditions, where the variation between days is greater than 50%. The input power traces for each of the above conditions are obtained from solar arrays at the UCSD microgrid, and the results of each algorithm are compared against the actual data for mean error.

Table 2.1: Solar Energy Prediction Algorithm Comparison

Algorithm	**Mean Absolute Error (%)**		
	Consistent	**Inconsistent**	**Severely Inconsistent**
EWMA	12.7	32.5	46.8
eEWMA	4.9	23.4	58.7
WCMA	3.99	9.6	18.3

Table 2.1 shows the results for all three cases. The α values are selected empirically, testing against values at a granularity of 0.05 until a value that minimizes error is found. For the EWMA and WCMA cases, the α value of 0.25 is found to minimize error, while for eEWMA, the α value of 0.3 is selected. The proximity of α to 0 demonstrates that recently observed values are the dominant factor in the final prediction.

The results demonstrate the marked improvement of the WCMA prediction over the other algorithms, even in severely inconsistent conditions. The primary reason is the sensitive nature of the algorithm to daily variations of solar irradiance. The WCMA algorithm is uniquely suited to solar energy prediction, as the GAP factor inherently adjusts for the change in irradiance in the current day compared day compared to previous days.

2.3 Short-term Wind Energy Prediction

We now explore wind energy prediction algorithms, identifying several that are applicable to the energy sources and timing constraint restrictions in the data center domain. Wind energy prediction is typically divided into two approaches: physical modeling and statistical modeling. Physical modeling utilizes meteorological forecasts represented in space to develop relationships to wind energy. At each location represented by wind turbines, corresponding power models, such as manufacturer-provided power curves, convert physical characteristics to power output [47]. Statistical modeling, on the other hand, uses collected meteorological data, either through direct measurement or numerical weather prediction data, in conjunction with measured wind turbine power. A variety of algorithms are then used to create a relationship between the meteorological variables and output

power, which can be used to predict future wind power [63].

Physical modeling is typically used when predicting wind energy for large farms, which can take advantage of the location-specific results provided by such algorithms. They depend heavily on Numerical Weather Prediction (NWP) models, which involve high-overhead local data gathering for meteorological variables and complex processing to obtain prediction of the variables over different points in space. Though highly useful for large wind farms, where the meteorological data may be different at each turbine's location, they trade local accuracy for a larger coverage area. However, the data centers represent a much smaller spatial system than the comparatively large wind farms, and benefit from the specificity of statistical models. Statistical models also have a reduced dependence on NWP forecasts, which can vary in accuracy and number of variables based on location.

In contrast, statistical models are more widely applicable, and can leverage a varied number of inputs. Additionally, statistical models better predict the behaviors for individual turbines or small clusters, which suffer less distortion from other meteorological variables. Therefore, in this section, we investigate the state of the art in wind power prediction algorithms, investigating the statistical approaches that lend themselves to the wind installations more commonly found in data centers.

2.3.1 Persistence Prediction

Wind predictors are typically measured against a persistence model [47], which simply predicts the output of a future interval to be identical to the output of the current interval. This is the first of the algorithms we use. Although persistence predictors typically perform better for shorter-term prediction, the relatively sudden changes in output power evident in Figure 2.1 demonstrate very high error rates.

2.3.2 Data Mining Prediction

The case study in [64] uses an input based on 120 meteorological variables, retrieved from data acquisition units (DAUs) situated at each turbine and collecting at high frequency. It then uses several data-mining models to develop relationships between the input variables: the support-vector machine regression model (SVMReg), Multilayer Perceptron network (MLP), Radial Basis Function network (RBF), the Classification and Regression tree (C&R), and the random forest algorithm. Each algorithm uniquely interrelates the variables to predict future wind speed. The results, shown below in Table 2.2, demonstrate the range of prediction errors and the advantage in accuracy the SVMReg algorithm provides over the others.

Table 2.2: Data Mining Wind Speed Prediction Algorithm Comparison

Algorithm	**Mean Absolute Error (%)**	**Std Dev (%)**
SVMReg	16.89	17.92
MLP	10.94	9.99
RBF	20.32	19.68
C&R	25.43	22.57
Random Forest	22.19	19.89

After determining the effectiveness of the support-vector machine regression algorithm to predict wind speed, the authors retained the characterizing functions. They proceed to predict wind farm power output by applying the same derived functions used wind speed prediction, with the justification is that wind speed has a very close relationship to wind power. The mean error for wind farm power, however, is compounded by the error for wind speeds shown above in Table 3. While 10-minute-ahead prediction results in 19.8% mean error, longer intervals demonstrate a consistently growing error, up to 73.3% mean error for 60-minute-ahead prediction. While other data-mining algorithms demonstrate better accuracy ($<$ 20% mean error) [47], they depend heavily on data gathered and processed through local Numerical Weather Prediction (NWP), and ultimately provide inconsistent results for different locations.

2.3.3 Autoregressive Moving Average (ARMA) Prediction

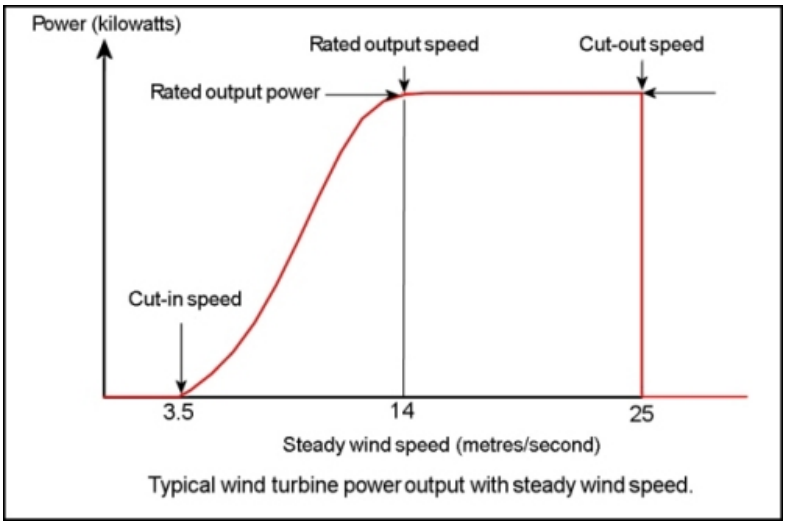

Figure 2.2: Manufacturer-provided wind turbine power curve, showing cut-in point, dynamic range, and maximum rated output power and cut-out [22].

Another approach to wind prediction explores the errors inherent in manufacturer power curves. The work presented in [84] analyzes the theoretical power curves against real data, demonstrating marked inaccuracy. They then provide the solution of generating power curves. Power curves provided by manufacturers represent wind power measured against wind speed, as shown in Figure 2.2. However, in plotting actual wind speed/wind power data, [84] demonstrates that the highly nonlinear relationship between wind speed and power, combined with the impact of other meteorological factors such as air density and wind direction, cause manufacturer power models to be inaccurate as a universal measure. The alternative proposed is a combination of several statistical predictors, most prominently the Auto-Regressive Moving Average (ARMA) model. The ARMA model is expanded with wind speed and direction as follows:

$$p_{t+1} = a_{0,t} + P_t(k, c) + W_{t+1|t}(q) + D_{t+1|t}(q) + e_{t+1|t} \tag{2.7}$$

where P_t represents the weighted moving average of the past k values and $W_{t+1|t}$ and $D_{t+1|t}$ are the parameterized models for wind speed and direction, respectively, and $e_{t+1|t}$ represents the error from the previous interval. The results show 50%

reduction in the average error present in power prediction. The combination of dynamic power models shows maintenance of the reduced error levels between prediction horizons of 2 hours to 45 hours.

2.3.4 Proposed Nearest-Neighbor Prediction

The diversity and requirements of the wind energy prediction algorithms pose a difficulty for testing. While the time-series solar prediction algorithms require only past output data, the algorithms described above require various types of input, from access to NWP forecasts to DAUs recording real-time data at each turbine. Additionally, the targeted horizons for each predictor are different, causing applicability in other prediction intervals to be different. In contrast, we develop a novel wind energy predictor with the goal of wide applicability and low input overhead. Instead of data that needs to be specially collected or provided, we utilize readily available data that has been shown to strongly correlate with wind energy prediction [84]: wind speed and wind direction.

Utilizing the effectiveness of dynamic power curves, our algorithm creates weighted nearest-neighbor (NN) tables to generate wind power curves using available wind speed and direction data at each 30-minute interval. Weighted tables allow the algorithm to adapt to seasonal changes by weighting recent results highly, while the power curves offer flexibility, allowing the algorithm to be used with different wind farms. The appropriate power curve table is updated using the current interval's observed wind velocity, direction, and output power as follows:

$$P_{new}(v,d) = \alpha * P_{obs}(v,d,t) + (1-\alpha) * P_{old}(v,d) \tag{2.8}$$

where $P_{new}(v,d)$ is the new power curve table entry for a given wind velocity v and direction d, $P_{old}(v,d)$ is the existing value for the same velocity and direction, and $P_{obs}(v,d,t)$ is the observed value at time t. While α can vary from 0 to 1, we found most accurate results with $\alpha = 0.75$, which weights the model more heavily towards currently observed data. Future interval prediction uses a table lookup based on the predicted wind velocity and direction:

$$P_{pred}(v,d,t+k) = P\left(v(t+k), d(t+k)\right) \tag{2.9}$$

The table is initially populated with a training set. When a prediction value for a particular $v(t), d(t)$ is not available, a prediction is obtained by the average of the N nearest neighbors to the needed data point:

$$P_{pred}(v, d, t) = \frac{\sum_{i=0}^{N} neighbor_i(v(t), d(t))}{N} \tag{2.10}$$

2.3.5 Results

The necessary power data for our testing was provided by a wind farm in Lake Benton, MN, in one-minute intervals, and the meteorological data was provided by published reports from the National Renewable Energy Laboratory (NREL). The prediction interval is determined by the least-granular of our input data, wind direction, which is available every 30 minutes. The availability of input data, and consequently, the widespread applicability of prediction, is the primary goal of our algorithm. So, to provide an appropriate comparison, we subject all the prediction algorithms to this limitation of the input variables. Persistence prediction, our baseline for prediction, depends only on the output data from previous intervals, and requires no adjustment for comparison. The other algorithms tested are:

- *Data mining*: we limit the data acquisition to the two variables of wind speed and wind direction and use the Multilayer Perceptron network (MLP) algorithm, as it performed the best in the corresponding work (Table 2.2). The only input variables required are the forecasted wind speed and the history of wind power.

- *Dynamic Power Modeling*: we develop parameterized models for each of the input variables and create extended ARMA models. The α_i, β_i, and γ_i values from Equation 2.7 are derived by analyzing previous interval data and developing a trend for the lowest error for 3 past intervals[84]. While the ARMA model is usable on its own, the dynamic switching among algorithms requires additional input variables. As such, we are limited to Equation 2.7, which uses only wind speed and wind direction, our established input variables.

- *Wind speed Nearest-Neighbor Predictor*: we use forecasted data for wind speed and 100 values for a training set [64]. We tested the algorithm against values for k between 1 and 10 finding the minimum error to occur when k=7.

- *Custom Nearest Neighbor Predictor*: our predictor uses both the inputs specified: wind speed and wind direction. Random sequential sampling of available data showed that the table was, on average, over 80% populated when the training set reached over 1000 samples. Iterative testing demonstrates that prediction error becomes negligible as the number of nearest neighbors increases past 4.

The results, shown in Table 2.3, reflect both the applicability and the versatility of each algorithm. Persistence, as expected, has a very high error at 137.4%, compounded by the relatively high variability of the wind farm power output. The data-mining algorithm also produces a relatively high error, at 83.91%, despite using the two most-correlated variables. We attribute this to the relative inaccuracy of the model when only provided with two variables. The ARMA model performs better, at 63.24% error, and has the best standard deviation, but the accuracy is hampered by the limited input data available, and the consequent lack of additional models. The lack of additional models further precludes the ability to dynamically switch among predictors. Finally, the wind-speed-based nearest-neighbor predictor, the kNN algorithm, performed the best of the comparison predictors, with 48.21% error. While this error is the best of the comparison predictors, it still performs with over 20% more average error than the previously published results. However, the accuracy of the NREL input data is unknown, and the published wind farm data exhibited less variability than the Lake Benton wind farm used in our experiment. Finally, the custom nearest-neighbor predictor, which uses both wind speed and wind direction to generate a dynamic power curve for wind farm output, performs the most accurately. The model, which is more adaptive to recent conditions and performs well under short-term prediction, performs over 25% better than the next-best algorithm.

The advantage of the custom NN predictor centers on the fact that it is developed for both the target prediction horizon and the available input data.

Table 2.3: Wind Power Prediction Algorithm Comparison

Algorithm	**Mean Absolute Error (%)**	**Std Dev. (%)**
Persistence	137.4	340.2
Data Mining	83.91	101.0
Dynamic Power Modeling (ARMA)	63.24	11.67
Wind Speed NN Predictor	48.21	32.11
Custom NN Predictor	21.23	17.40

However, it also incorporates the advantages of the other predictors. Specifically, it addresses the inaccuracy of static power curves by deriving adaptive power curves. It accounts for the impact of wind direction on turbine output by adapting direction into wind power predictions. Using a weighted moving average to update the table reflects the seasonal patterns handled by all the comparison algorithms. Finally, the use of widely-available meteorological variables leverages the accuracy of NWP forecasts without incurring the complexity and measurement needs of weather data. In short, the inclusion of several important characteristics of wind energy prediction, and the design of the algorithm around the input data and prediction horizon, result in a widely-applicable, accurate predictor.

2.4 Case Study: Predictive Allocation of Batch Workloads in Data Centers

In this case study, we focus on both intra- and inter-data center renewable energy integration. We first envision a data center whose nominal operational power requirements are met with non-renewable energy. The data center's power supply is supplemented by local wind and solar energy sources, which can be used to run additional workloads. The data center scheduler leverages renewable energy prediction algorithms to schedule additional workloads as appropriate within the prediction horizon. In order to evaluate the benefits of short-term prediction, we compare the experiment to scheduling workloads with only instantaneous renewable energy information. We then expand this model to a networked set of

data centers. Leveraging the differences in green energy prediction and brown energy pricing at geographically distributed locations, we formulate an optimization problem where online job migration among data centers is used to improve the performance of batch workloads. Furthermore, we incorporate network constraints such as availability, link capacity and transfer delay.

2.4.1 Data Center Workloads

In order to ensure accuracy of the simulations, we leverage traces from real data center workloads and benchmarks. Data centers typically run two different types of workloads: latency-sensitive service workloads that are measured by completion time, and compute-intensive batch workloads, which are longer-running and measured by throughput.

Data centers set strict completion deadlines for service workloads in order to maintain quality of service. As such, service jobs do not lend themselves to being run under green energy, as lack of energy availability would prevent jobs from executing and compromise the quality of service. Batch jobs, however, are longer-running and throughput-dependent. As such, they are more applicable to green energy, as they do not have tight timing constraints and are more tolerant of green energy variability.

The service workloads are derived from traces of the Rubis benchmark, which models bidding systems such as eBay [17]. The batch jobs are based on Hadoop's MapReduce implementation, which are throughput-oriented workloads divided into subtasks and distributed throughout the system, then recombined after all the subtasks complete. In addition to being an actual data center batch workload, MapReduce is uniquely suited to our experiment: at our specified parameters, the workloads show 90% completion rate in 30 minutes, which is an ideal threshold for our short-term prediction algorithms. Additionally, the innate distributed approach provides the ability for a job to be resumed, as individual subtasks can be canceled without the entire job needing to be restarted.

2.4.2 Data Center Simulator

The experiment is run on our datacenter simulator [27], which models the execution of mixed workloads in a datacenter, simulates the corresponding power consumption, and incorporates the available green energy. While implementation specifics are detailed in [27] [25], we illustrate the relevant characteristics. The simulator is event-driven, progressing with each workload event, such as a job entering or exiting the system, and with each system event, such as power consumption or throughput monitors. A global scheduler handles the allocation of each new job to an individual server, and a local scheduler handles the execution of a job within the server. The execution of jobs is controlled by trigger events including the monitoring of overall power consumption, green energy availability, and service time requirements. As service jobs determine the quality of service provided by the data center, they are given priority over batch jobs. After all service jobs are scheduled, batch jobs are scheduled to fill the immediately remaining power capacity. In addition, when green energy prediction data is available, batch jobs are scheduled to fill the projected capacity for the prediction horizon. We validate the data center simulator's power consumption against actual measurements run on the representative jobs on Nehalem servers. The quality of service ratio (QoS) for service jobs is measured by the 90th percentile response time over the expected response time, with values below 1 representing acceptable service. As this metric dictates whether or not the data center is meeting its requirements, it is important to ensure that our data center simulator accurately reflects it. Table 2.4 shows less than 3% mean error in average power consumption, and 6% mean error for QoS, comparable to existing datacenter simulators [25].

Table 2.4: Data Center Simulator Validation

System	Measured	Simulated	Mean Absolute Error
Avg. power	246 W	251 W	3%
Avg. service job QoS ratio	0.08	0.085	6%
Avg. batch job completion time	21.2 min	22.8 min	8%

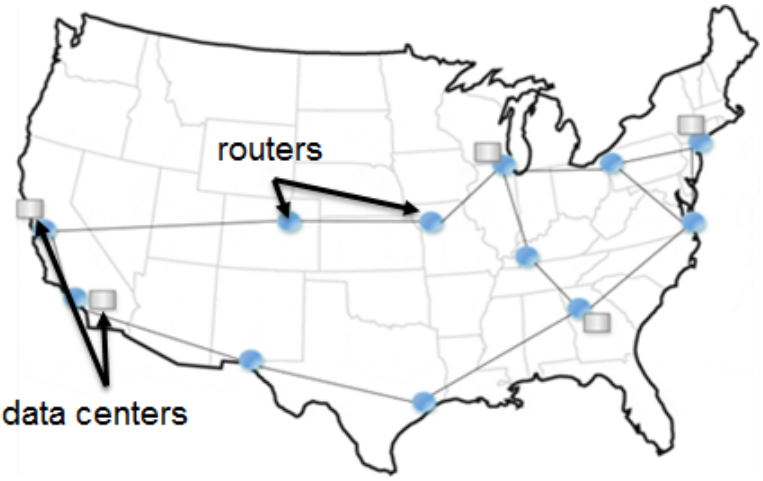

Figure 2.3: ESNet network topology. [26].

In addition to modeling a single data center, we also simulated a networked set of data centers connected by a wide area network. This allows us to investigate scenarios such as local pricing, job migration, and regional renewable energy availability. We simulate a network that is a subset of the ESNet topology (Figure 2.3). It includes 5 distributed data centers and 12 routers distributed across the USA, where each connection link has a fixed maximum capacity from 10Gbps to 100Gbps. 10% of this capacity is allocated for background traffic in our experiments. We compute the total network energy need with the router power and a fixed offset for the link power consumption. We estimate the router power consumption with a linear model based on bandwidth utilization. The model details are specified in [26]. Table 2.5 outlines the configuration of the network at each of the different locations, including the available hardware and on-site renewable energy, if applicable.

2.4.3 Green Energy and Prediction

The experiment aims to utilize local sources of both solar and wind energy, using power traces from a solar array from the UCSD Microgrid and a wind farm from Lake Benton, MN. We scale the input to provide on average 35% of the energy input to the data center, based on the analysis of real data centers and

Table 2.5: Data center network config. and available on-site renewable energy

Location	Node Hardware	Onsite Renewables
Chicago	Data Center + Router	Wind
Atlanta	Data Center + Router	Solar
Kansas	Router	—
Nashville	Router	Wind
San Francisco	Data Center + Router	Wind + Solar
Denver	Router	—
New York	Data Center + Router	Wind
San Diego	Data Center + Router	Solar
El Paso	Router	Solar
Cleveland	Router	Wind
Houston	Router	Solar
Washington, DC	Router	—

their capacities [27]. We leverage prediction numbers for batch workloads, whose expected durations are 30 minutes. The prediction algorithms need to correspond to the same horizon in order to provide useful data for scheduling.

Of the solar predictors analyzed in the prior section, the WCMA predictor performed with the highest accuracy and consistency, and predicted the above traces with an error of 9.6%. As the WCMA algorithm is intended for short-term prediction, and the granularity of the input data corresponds to the prediction horizon, it was selected.

The wind predictors analyzed above explored several with good short-term prediction capabilities, but when identifying a good predictor for our purposes, the applicability of the algorithm to the available input data limits the options. Numerical Weather Prediction models are simply unusable for the Lake Benton wind facility, as none are available for the relatively remote locale. The data-mining predictor [63] requires 250 variables provided by data-acquisition units (DAU), which again are unavailable. Instead, we opted to use our custom nearest-neighbor predictor. The NREL publishes wind speed and direction information for most of the United States, with sufficient granularity for the custom NN predictor. This, along with the power output information provided by the Lake Benton wind farm, fulfilled the necessary requirements. Although the mean error was relatively high at 21.2%, it was nevertheless favorable to an attempted adaptation of the

data mining approach utilizing the same NREL data in lieu of collected data, which resulted in a mean error of 48.2%.

The impact of prediction errors are realized in two ways. The first case is overprediction, which causes the scheduler to allocate more jobs than can be handled in an interval. The excess workload will be terminated as soon as the quota of actual energy for an interval is exceeded, resulting in wasted work. The second case is under-prediction, when the scheduler will only allocate as much work as dictated by the prediction, leaving the difference in the actual energy unused. Our metrics, defined in the following section, reflect the penalty in green energy efficiency caused by both sides of misprediction.

2.4.4 Results

As we need to demonstrate the impact of short-term prediction over instantaneous use of green energy, we must select appropriate metrics. The first of these, the *Green Energy Efficiency (GEE)* metric, is defined as follows:

$$GEE = \frac{GE\ used\ for\ job\ completion}{Total\ GE\ available} \tag{2.11}$$

This metric penalizes any wasted green energy. For the instantaneous case, a decrease in green energy variability can cause batch jobs to be canceled. In the short-term prediction case, both over- and under-prediction are penalized: the former because the excess green energy is wasted, and the latter because scheduled jobs that exceed the available green energy will be canceled. A second useful metric is the average *Batch Job Completion Time*. As the throughput of batch jobs reflects the effectiveness of the data center, the average completion time over the simulation interval quantifies the improvement attributed to green energy. The final metric quantifies the percentage of the total work that is performed with green energy:

$$GE\ Job\% = \frac{Jobs\ completed\ with\ GE}{Jobs\ completed} \tag{2.12}$$

This provides an overview of the pervasiveness of green energy use within the data center.

Intra-Data Center

The control for the simulations is instantaneous green energy use, which reflects how renewables are currently integrated into data centers. The comparison is the experiment we devise, leveraging short-term prediction of green energy output to schedule additional batch workloads in a power-proportional manner. The experiment is also performed on three separate renewable-energy cases: solar energy, wind energy, and a combination of both, scaled to achieve the 35% energy contribution.

The Green Energy Efficiency Metric reflects the percentage of available green energy used to perform useful work. The results in Figure 2.4 show, on average, 1.75X improvement of short-term prediction over instantaneous green energy use, and in the best case, 3X for wind energy alone.

An analysis of the simulator data reflects the initial observations: the fluctuations in green energy input, particularly in the more highly-variable wind energy case, prevent the batch workloads scheduled based on instantaneous energy data, from completing. However, the improved knowledge of energy availability from prediction reduces these errors and enables a more efficient use of green energy. The Batch Job Completion Time also shows improvement when using green energy

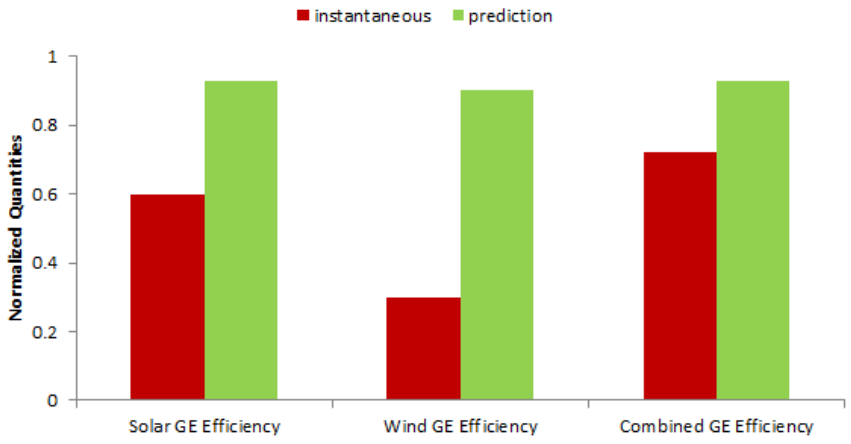

Figure 2.4: Green energy efficiency metric for instantaneous and short-term prediction. Prediction shows over 90% energy efficiency across the board, in the best case (wind-only), over 3x improvement over instantaneous use.

.

prediction. Reflected in Figure 2.5, job completion time is reduced by 12.5% using prediction. These results show the benefit of better power proportioning per interval, as batch workloads are more appropriately scheduled, and fewer jobs must be terminated prematurely. In the instantaneous case, the inherent fluctuation of wind energy plays a major role in forcing early termination of batch jobs, as does the steady decline of available solar energy in the latter half of each 24-hour day. The GE job % reflects the overall impact of green energy within the data center.

Figure 2.5: Normalized Batch Job Completion Time for no green energy, instantaneous green energy, and short-term prediction. , showing 12.5% reduction with prediction.

.

Utilizing prediction over instantaneous use results in, on average, 15% more jobs completed with green energy, as demonstrated in Figure 2.6.

Multiple Data Centers with Job Migration

Finally, we investigate the case of a networked set of data centers using the heuristic scheduling algorithm described in [26]. We modify this iterative part of the algorithm to maximize the performance of the workloads. Batch jobs that

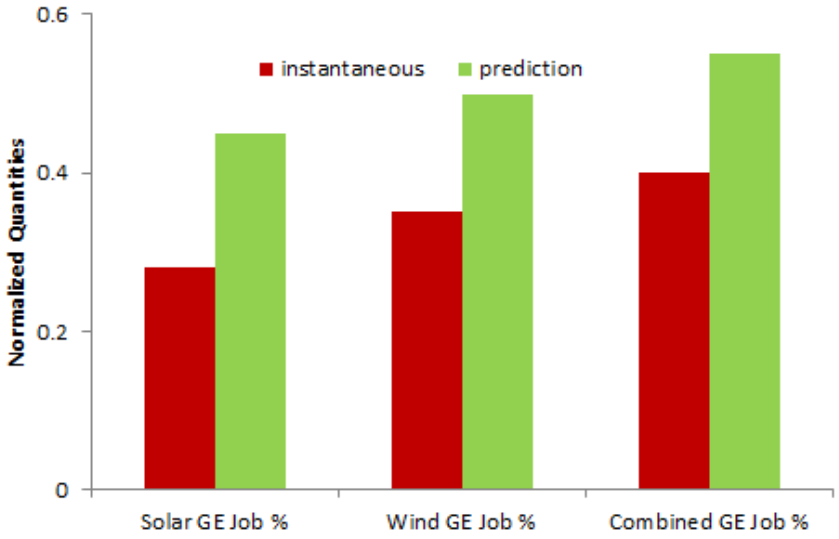

Figure 2.6: Green energy job % for instantaneous and short-term prediction. Prediction shows an average of 15% more jobs completed with green energy.

.

are actively waiting in execution queues are migrated to data centers with excess green energy availability, where the green energy is predicted as above and the availability and configuration of each data center is described in Table 2.5. The key to this policy is that waiting tasks are migrated, resulting in more jobs executed overall. The scheduler produces a matrix representing workload transfers among data centers. This transfer matrix is then provided to the networking algorithm, found in [29], which calculates the paths to be used and the amount of bandwidth that needed by each selected path.

Our simulation results show that the average completion time of MapReduce jobs is 16.8 min, 27% faster than the baseline, with no performance hit for service requests. Furthermore, since we are leveraging all predicted available green energy for extra workloads, the Green Energy Efficiency (GEE) is 85%, significantly higher than the baseline of no migration, which has a GEE of 59%.

2.5 Conclusion

In this chapter we provide a comprehensive background of green energy use in data centers today. We introduce the inherent difficulties in green energy use, namely: unpredictability of the output, and demonstrate how current datacenter implementations attempt to compensate to overcome this cost. We then explore methods of short-term renewable energy prediction, analyzing difficulties, and provide solutions: the WCMA prediction algorithm for solar energy, and a predictor of our own design that aims for wide applicability for wind energy. Finally, we define and evaluate a case study of single and interconnected data centers leveraging instantaneously available local green energy, and compare it to using green energy prediction and job migration. The results show that prediction leads to 3x higher green energy efficiency and reduces the number of terminated batch tasks due to insufficient green energy availability during task execution by up to 7.7x as compared to instantaneous green energy usage. For networked data centers, we demonstrate a 27% reduction in the completion time of batch workloads through job migration based on energy prediction at connected sites.

Chapter 2 contains material from "Using data center simulation to evaluate green energy integration", by Baris Aksanli, Jagannathan Venkatesh and Tajana Simunic Rosing, which appears in IEEE Computer 45, September 2012 [25]. The dissertation author was one of the primary investigator and the second author of this paper.

Chapter 2 contains material from "Renewable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks", by Baris Aksanli, Jagannathan Venkatesh, Tajana Rosing, and Inder Monga, which appears in Computational Sustainability, Springer, 2015 [26]. The dissertation author was a primary investigator and second author of this paper.

Chapter 3

Modeling Residential Energy Management in the Smart Grid

Building energy consumption has been well-researched, but the focus has been on commercial and industrial domains, which constitute a majority of global energy consumption. However, the residential domain contributes 38% of the total energy costs in the United States (Figure 3.1). Moreover, the residential domain is important because of the significantly higher number of end users impacted: in the United States alone, residential energy consumption impacts hundreds of millions of homes and other residences. In addition to the significant impact, the breakdown of energy use is shifting towards appliances and electronics (Figure 3.2) — elements that are becoming smarter and remotely actionable by entities such as the smart grid[23]. Some related research focuses on reducing a single aspect of consumption [98], while others seek to provide more granular energy information to end-users in order to facilitate user-driven improvements [60]. While these approaches are indeed beneficial, they require considerable overhead in data collection and testing, and results cannot be quantitatively compared.

Although a few building simulators have been proposed, there are no residential energy simulators capable of modeling more complex scenarios and exploring the tradeoffs in home energy management. In this chapter, we outline a residential energy simulation platform (HomeSim) that makes it possible to investigate the impact of technologies such as renewable energy, smart appliances, and

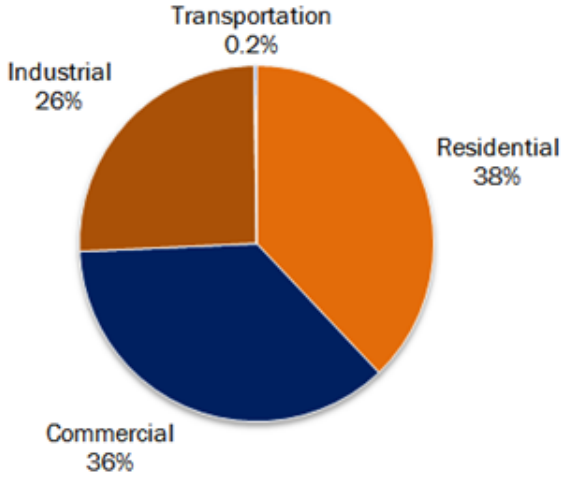

Figure 3.1: Fig. 1. 2009 Retail Electricity Sales in the United States, Total by End-Use Sector. [8].

different battery types. Additionally, HomeSim allows us to simulate a number of different scenarios, including centralized vs. distributed in-home energy storage, intelligent appliance rescheduling, and outage management. Using measured residential data, HomeSim quantifies the benefits of different technologies and scenarios, including up to 50% reduction in grid energy through a combination of distributed batteries and reschedulable appliances.

3.1 Related Work

3.1.1 Residential Energy Simulation

Building simulation involves modeling loads and sources, with a schedule to aggregate each element's provision or consumption [41]. It can provide accurate results orders of magnitude faster than real-time implementation. However, few home energy simulation platforms have been developed, and those that have require very stringent assumptions. This prevents them from being applicable for general studies such as the ones presented in our work.

The work in [87] provides a comprehensive review of residential energy, detailing both top-down and bottom-up simulation, which involves the modeling

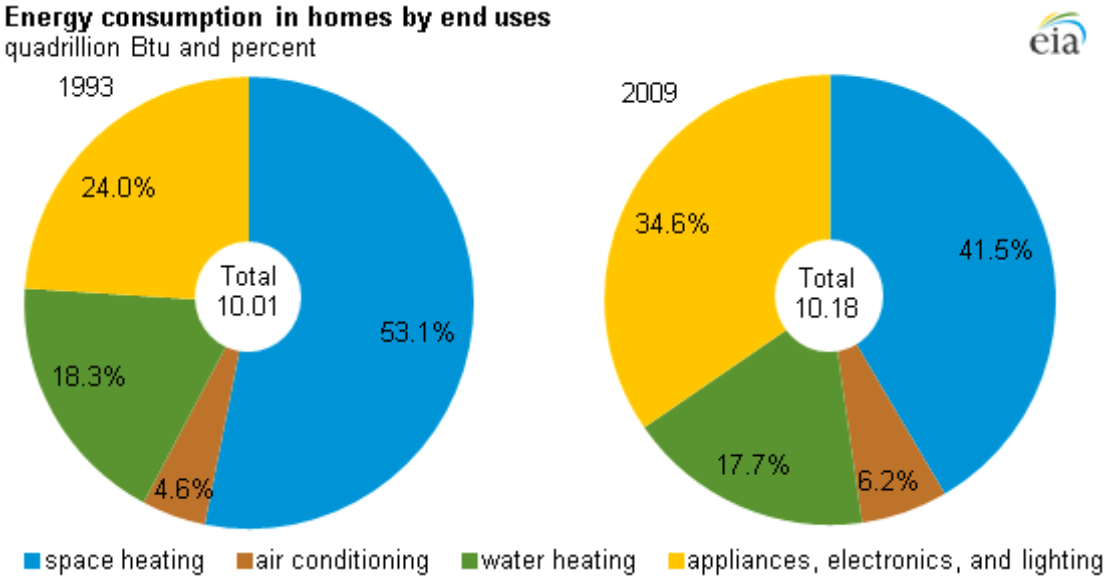

Figure 3.2: Residential energy consumption breakdown in 1993 and 2009. [12].

individual buildings, similar to our goals, and leveraging the models into classifications (single-family residences, apartments, etc), as appropriate to a region, and simulating the result.

The simulators in [33] & [98] study the tradeoffs between renewable generation and grid pricing, but only for the homes and neighborhoods where their study was deployed. This precludes the possibility of verifying their results with different usage and residence configurations. The Department of Energy's NZERTF home simulator [23] provides an open interface for user models, allowing comparison of energy consumption based on user behavior patterns. However, usage patterns are tested on a single, instrumented house, with no ability to specify a different home configuration, thus limiting its scope.

Commercial and open-source energy simulators provide the complex interactions found between elements in the energy grid. However, the granularity of the interactions is not extended to the residential domain. GridLAB-D, a comprehensive grid simulation platform, provides nominal residence and appliance modeling [38]. However, the simple probabilistic load model is not able to convey local renewable generation, distributed storage within a home, and the various scenarios introduced by smart grid automation. While some attempts have been made to-

wards extending GridLAB-D in this direction [31], a majority of the work focuses on simply modeling elements within the house rather than the different means of interaction. Additionally, the ability to handle distributed batteries is not covered under these extensions. Similarly, OpenDSS [19] provides the ability to model complex distribution networks at different levels of the grid. However, this does not extend to the level of individual end-use elements, which can be specified as generic loads or sources, but without the fine granularity needed to model complex end-use scenarios.

3.1.2 Residential Energy Load Monitoring and Modeling

Residential energy research has been motivated in part by the number of people it affects [87]. A majority of work has focused on characterizing the previously dominant HVAC consumption in the home, but as Figure 3.2 demonstrates, appliance energy use is growing compared to other residential loads. The ability to correctly model and control appliances in a home is important for scheduling and determining load use, and is helpful for implementing the complex scheduling behavior explored in our case studies.

The work in [60] focuses on non-intrusive load monitoring, with the ultimate goal of presenting very granular power consumption for each appliance to enable users to make smarter decisions. The significant contribution is the ability to isolate appliance power data non-intrusively using a learning model, with 82% accuracy. However, the work does not strive to automate the process, choosing instead to leave power management decisions in the hands of users. [33] presents automated energy efficiency improvement in homes that are partially powered by green energy with storage. The work proposes an energy management system that provides early warnings, and suggests task rescheduling for maximizing energy. However, again, decisions are left to the user, with automation addressed as a feasibility study. In the wake of smart appliance and home automation research, we choose to investigate the impact of automation in reducing grid consumption.

Another approach [73] automates battery charging and discharging in a green home based on predicted solar energy and battery state-of-charge. The work

reduces grid energy by an impressive 3.9x. However, the paper only improves one aspect of energy consumption—battery charge—while making several assumptions that limit the widespread benefit, including a reliance of time-of-use (TOU) pricing; depleting batteries completely instead of a specific depth-of-discharge; and dependence on a variety of forecast data.

A second work [31] compares the use of different learning techniques such as Bayesian Networks and Artifical Neural Networks (ANNs) to predict residential water use, and develops an integrated ANN to predict demand with average relative error of 30%. Similarly, [52] uses Bayesian Networks to predict user behavior, which in turn is used to determine appliance usage, and ultimately, energy needs for 24 hours ahead. Using time, energy, and duration as the input random variables, the paper predicts appliance usage based on a real dataset. Finally, MIT's REDD project [60] utilizes the Factorial Hidden Markov Model to disaggregate overall energy data from residences, and predict which appliances are active over a given timeframe in a non-intrusive manner. They train their predictor using supervised approaches, providing as granular circuit-level information as possible, and are able to determine the appliance with 82% accuracy.

3.1.3 Takeaways

The technology of residential energy management has been evolving. The emergence of distributed energy storage, smart appliances, and automated control [73][98] has blurred the distinction between loads and sources. Storage elements such as batteries can consume grid or renewable power for charging or be used as energy sources. Consequently, scheduling evolves from a mapping of consumers and producers to a distributed system of interactions among elements. The related work demonstrates that existing residential energy simulators do not provide a flexible platform for comparing residential energy configurations. They lack the sophistication to handle emerging technologies and the ability to quantitatively compare the impact of different home energy management policies. The research in energy management performs one-off comparisons for specific energy scenarios, but cannot compare different technologies or implementations side-by-side.

To address these issues, we develop HomeSim, a simulator for evaluating residential electrical energy usage, storage, and generation. It is capable of modeling energy consumption of typical sources and loads, including utility power, generators, and household appliances, as well as energy storage, renewables, fuel cells, and "smart" appliances. HomeSim's model enables many configurations of end-use elements. Similarly, while the majority of existing simulators use a monolithic event-driven scheduler, HomeSim provides a highly extensible scheduling algorithm that can simulate more complex interactions among nodes and subsets of nodes, uniquely providing the ability to test new scenarios.

With the added capabilities of HomeSim, we are able to explore the impact of home energy scenarios that were previously impossible without actual implementation and instrumentation. We consider lithium-iron phosphate batteries, an emerging technology considered to be an alternative to traditional lead-acid batteries due to better performance characteristics. Through HomeSim, we can quantify their benefit within different house configurations and compare to their theoretical benefit. We also implement two energy-saving improvements suggested in the related work: distributed, appliance-specific batteries [75] and dynamic rescheduling of appliances [33]. By modifying the scheduling algorithm and the configuration of energy elements in the home, we can test both scenarios in HomeSim, demonstrating 36% reduction in grid energy using distributed batteries, and 25% reduction using distributed batteries. In addition, we investigate the impact of outage management, taking into account the appliance limitations/restrictions that must be imposed and the scheduling policies that would improve battery consumption, demonstrating high green energy efficiency. The ability to quantitatively validate residential energy management policies exemplifies the benefit of our simulator vs. the previous work.

3.2 HomeSim Design

A residential energy simulation platform can be broken down into two key components: the end-use elements (i.e. loads, sources) and the scheduler. Figure

3.3 depicts the general model of HomeSim, including the event-driven *scheduler* and the constituent *nodes* which represent energy sources, loads, storage, and hybrid components of a home energy management system. The scheduler determines which nodes are active at a particular iteration, and executes the consumption computation for them. While the core of the scheduler is straightforward and representative of a bottom-up home energy model [87], the actual computation at each step is configurable. This is particularly useful when modeling complex interactions between loads and sources. One example is the growing popularity of "smart" appliances, which can modify their own power profiles or schedules as a function of the control input. HomeSim's modular computation step facilitates the adaptive behavior of such emerging technologies in a way that previous simulation platforms simply cannot. The *nodes* provide an easy way to define all the relevant

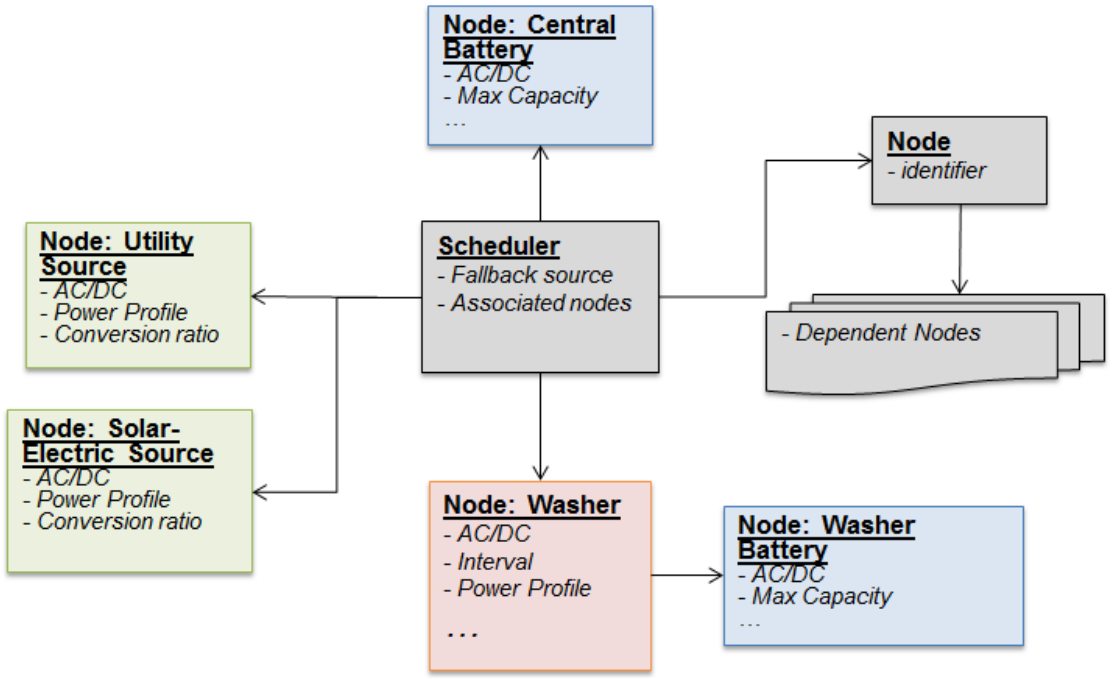

Figure 3.3: HomeSim System Design Model.

parameters for energy sources, loads, storage, and hybrid elements of a residential system. They also contain *power profile* functions, which provide a time-indexed mapping of the node's power data. This provides the flexibility to model complex devices using a single data structure, such as different battery technologies

and renewable energy sources, by changing the behavior of power profile function. Additionally, each node can be associated with a list of *dependent nodes*. This list creates a direct connection between nodes that affect each other. The result is a tree of dependent nodes under the scheduler that inherently captures interactions among elements that is lacking in other simulators. The dependency lists can model circuits within a home or hierarchical loads. They provide the possibilities of load, source, and storage interaction previously unavailable in home energy simulation. The next subsections explore these components and the scenarios they enable in greater detail.

3.2.1 Nodes

A *node* is the abstract data structure that encompasses the end-use elements of a home. This data structure has an identifier, which classifies the device as a *source*, *load*, or *hybrid*. Complex elements such as batteries fall into the latter category, as they are both producers (when discharging) and consumers (when charging). Each node has a list of *dependent nodes*, which informs the scheduler of node to node interactions. In Figure 3.3, the "Node: Washer" illustrates this idea, with "Washer Battery" as a linked node. Here, the washer and the battery behave differently, with the washer leveraging the additional energy stored in its battery. This approach allows modeling more complex scenarios, like a backup generator, or virtually partitioning the hierarchy of nodes by the physical circuits in the home. Section 3.2.2 describes interactions between nodes and scheduler. Nodes act as loads, sources or a combination of both.

Loads

Loads represent the sinks in our energy model, consuming power whenever active. Based on the data provided by energy traces [60] [42], we can further classify loads with the binary variable $Always\ On = periodic, continuous$, where periodic loads (i.e. dishwashers, dryers) have fixed intervals and frequencies, and continuous appliances (e.g. HVAC) have functional usage patterns over time. Examples of periodic (lighting) and continuous (refrigerator) loads are found in the

sample dataset in Figure 3.4. The actual model for each load is defined by a

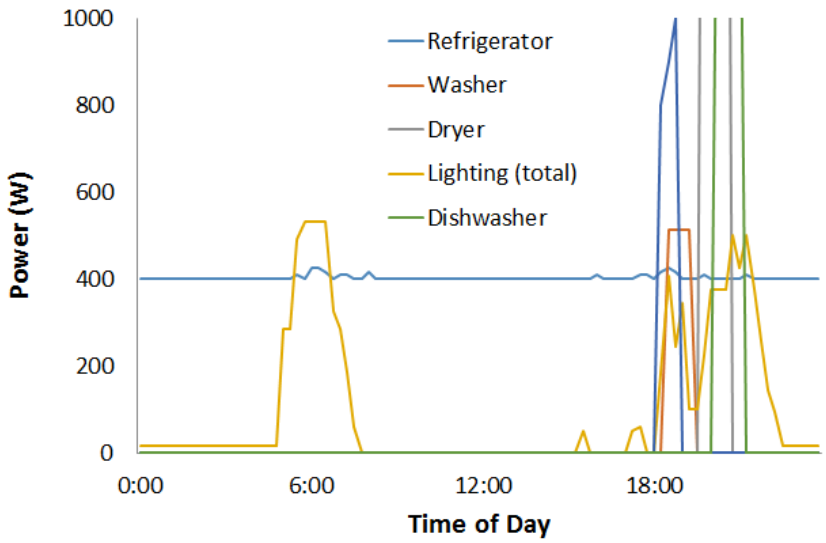

Figure 3.4: Appliance load profiles from REMODECE [42].

dynamic *power profile*, mapped against a fixed interval (for periodic appliances) or a time-dependent function (for continuous appliances). An appliance can also be characterized as operating on AC or DC power. Depending on the state of the incoming energy, appropriate conversion efficiency losses are used. Table 3.1 summarizes the parameters used for loads.

Table 3.1: Load parameters

Parameter	Description
AC/DC	AC or DC power
Interval	The time until the next event instance
Offset	The daily time offset to begin the first event
Power Profile	Power consumption profile function
AC/DC Duration	The length of an event interval (for periodic loads)
Continuous	Continuous or periodic load
Conversion factor	Factors to calculate transmission/conversion loss

Sources

Sources refer to nodes that are purely generators. The typical residential energy source is the utility power. Others may include solar-electric, wind, and

fuel cells. Consequently, HomeSim maintains a completely open model, with a binary *AC/DC* specification, a *power profile* function over time, and appropriate power *conversion factors*. This allows for very fine-grained modification of a source's energy data. For example, utility power can be modeled as constant power function with very high magnitude, since utility production far exceeds residential consumption. The parameters for each source are provided in Table 3.2. We also provide a *cost function*, which returns the energy cost for a particular power value. Like the power profile, this can be extended to handle different scenarios. For example, time-of-use (TOU) can be expressed as a function of power and time.

Table 3.2: Source parameters

Parameter	Description
AC/DC	AC or DC power
Power Profile	Power generation profile function
Cost Function	Operational expense of a source (cost/watt-hr)
Conversion factor	Factors to calculate transmission/conversion loss

Hybrid Elements

Hybrid elements such as batteries, flywheels, and plug-in electric vehicles (PEVs), are becoming more prominent in the residential domain [73]. Their ability to both supply and consume energy requires a separate interface. In addition to having a fixed capacity, these sources can also have different charge and discharge rates. Their characteristics are outlined in Table 3.3. They can either be associated directly with the HomeSim scheduler, for centralized behavior, or with a specific node, for distributed behavior.

Batteries

Batteries are a special case of hybrid nodes, and require a more sophisticated model to capture additional parameters. Accurate battery modelling has been extensively studied. The overview in [58] outlines the different approaches: electrochemical models simulate the changes in the chemical compositions of batteries over time; electrical-circuit models create an equivalent capacitive circuit;

Table 3.3: Hybrid node parameters

Parameter	Description
AC/DC	AC or DC power
Max Capacity	The maximum stored capacity in Ah
Current capacity	The current capacity of the node, in Ah
Nominal Voltage	The device line voltage
Upper charge/ discharge voltage	The maximum charge/discharge voltage
Lower current limit	The minimum operational current of the device
Upper charge/ discharge current	The maximum charge/discharge current limits
Conversion factor	Factors to calculate transmission/conversion loss

Table 3.4: Battery parameters

Parameter	Description
Peukert Exponent	The storage efficiency of a battery
Depth of Discharge (DoD)	The maximum fractional depth of discharge
State of Health (SoH)	The fractional available battery capacity
State of Charge (SoH)	The current fractional battery capacity

and analytical and stochastic models create a mathematical representation of the change in various battery characteristics over time.

Chemical batteries have non-linear charge/discharge characteristics, both during each cycle and over the lifetime of the battery. These properties increase the complexity of accurate battery models, which in turn make simple optimizations more difficult. Additionally, the behavior of batteries over each cycle throughout their lifetime is not consistent [35] [80].

Depth of Discharge (DoD)

The depth of discharge (DoD) of the battery identifies how low batteries are drained [77]. Changing the maximum DoD reduces the cycle life of the battery in a nonlinear fashion, and as seen in Figure 3.5, higher depths of discharge result in dramatically reduced cycle life.

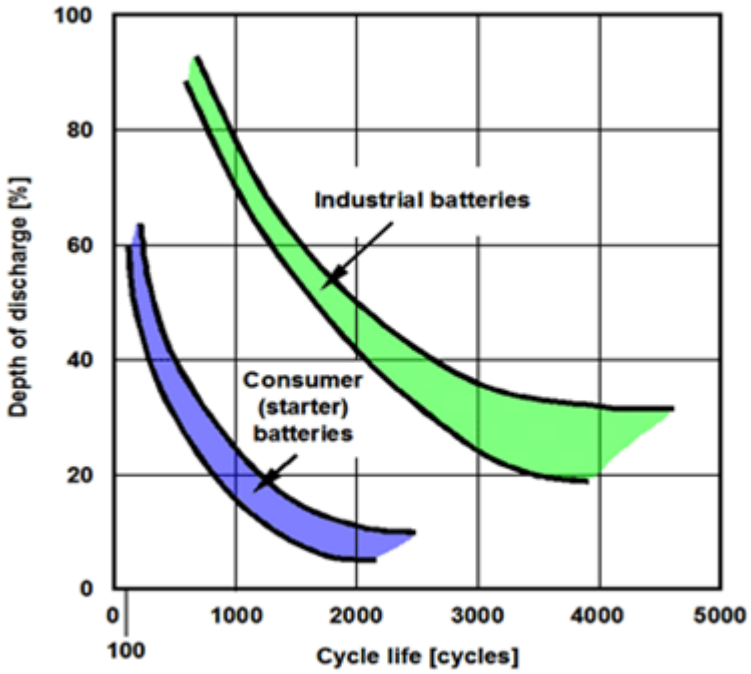

Figure 3.5: The cycle life of lead-acid batteries mapped against the depth of discharge [14].

Effective Capacity

Another significant impact on battery characteristics is the degradation of the capacity of the battery over time. Every discharge-charge cycle incurs a degradation of the remaining maximum, or *effective*, capacity of the battery. In addition, different discharge rates (i.e. discharge current and voltage) result in both different degradation rates and different cycle life. For example, Figure 3.6 illustrates the impact of changing discharge voltage for a Li-ion battery.

State of Health (SoH)

The state of health (SoH) of the battery is the ultimate metric for the lifetime of the battery. SoH is the percentage of the effectiveness of the battery in comparison to the rated condition of the battery. Different metrics are used to determine battery SoH, and of these, we use the remaining effective capacity as our metric. The terminating condition of the battery is when the effective capacity reaches a manufacturer-specified lower bound (e.g. 80% [28]).

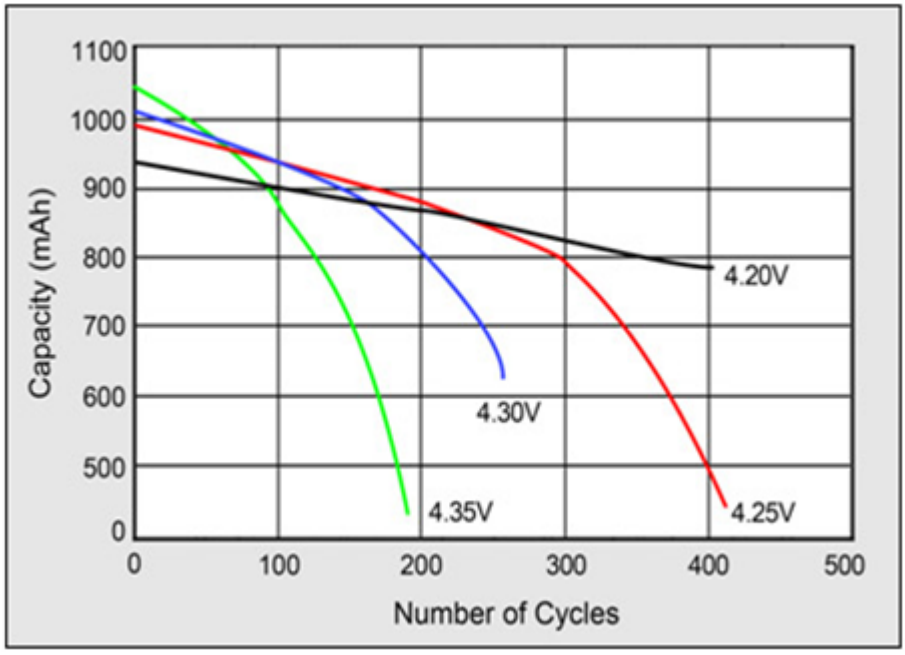

Figure 3.6: Change in cycle life and degradation in capacity for different discharge voltages [7].

Modeling Batteries

To account for battery nonlinearity, four additional parameters are used to store battery information: the *Peukert exponent*, *depth of discharge (DoD)*, *state of health (SoH)*, and *state of charge (SoC)*. The parameters are outlined in Table 3.4. The *state of charge (SoC)* tracks the current level of discharge. *Depth of discharge (DoD)* is a function of battery technology, and specifies the minimum level of charge that should remain in the battery for correct operation. Battery lifetime is dependent on the SoH, which decreases with the number of charge/discharge cycles. HomeSim uses the Coulomb Counting method to estimate these parameters [85], whose main benefit is simplicity, as it only needs measurements of voltage and current. The battery's discharge current, $I_{discharge}$, is:

$$C_{released} = \Delta t * I_{discharge} \tag{3.1}$$

The current depth of discharge can be calculated using:

$$DOD_{curr} = \frac{C_{released}}{C_{maximum}} * 100\% \tag{3.2}$$

where $C_{maximum}$ refers to the maximum capacity of the battery. We can then express the *effective capacity* of the battery:

$$C_{eff} = C_{maximum} * \left(\frac{C_{released}}{I_{discharge*H}} \right)^{k-1} * \frac{SoH_{old}}{100} \tag{3.3}$$

where H is the rated discharge time, k is the Peukert's exponent, and SoH_{old} is the previous SoH (initialized at 100). Using this value and a manufacturer-provided Depth of Discharge (DoD), we can calculate the current SoC and SoH:

$$SoC = DoD - DoD_{curr} \tag{3.4}$$

$$SoH_{new} = SoH_{old} - (100 - SoH_{dead}) * \frac{C_{maximum}}{Cycles_{DoD_{final}} * C_{eff}} \tag{3.5}$$

where DOD_{final} is the final discharge depth after the current cycle, using Equation 3.2, extended through the last SoH update; SoH_{dead} is the point at which the battery is effectively dead (technology-dependent); and $Cycles_{DoD_{final}}$ is the number of cycles over the lifetime of the system where the final discharge point is DoD_{final}.

3.2.2 Scheduling Algorithms

The standard scheduler for HomeSim is an event-driven scheduler over a time-ordered list of nodes, similar to other simulation platforms. It is the computation at each step, called the *execute* step, that distinguishes HomeSim from the previous home simulations. The *execute* step operates on the list of active nodes, allocating energy to each as necessary and determining the net consumption or generation. The implementation of this step can vary to handle different configurations of nodes and scheduling goals. In essence, the execute step provides an open scheduling platform. The following subsections investigate the different scheduling algorithms implemented within the *execute* step.

Default Scheduler

The simplest implementation is representative of the state of the art in renewable-enabled homes today, shown in Figure 3.7. At each step, each node uses

the green energy greedily, and then reverts to battery energy when there is not enough green energy remaining. Finally, if the battery capacity is also exhausted, the node uses grid energy. This is a relatively simple scenario, so in the next two subsections we illustrate two more sophisticated cases of scheduling that are made possible by the flexibility of HomeSim infrastructure.

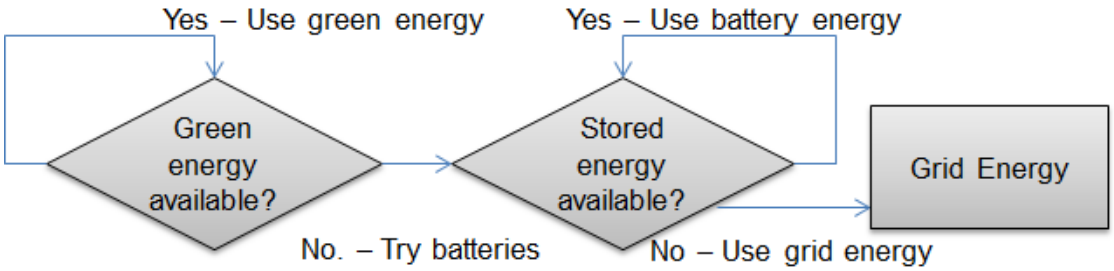

Figure 3.7: Default *execute* phase block diagram.

Distributed Battery Scheduler

The incorporation of green energy has made energy storage a necessity. The state of the art is a large, centralized battery, although the concept of appliance-specific batteries has been mentioned in [75]. Such distributed batteries offer the flexibility to allocate energy for the largest consumers, mitigating the stress on a centralized battery and preventing interference from other appliances. Figure 3.8 illustrates how a distributed battery model can be run by the scheduler's *execute* step. In this case the dependent node lists for each appliance establish a storage element as a distributed, appliance-specific battery (see "Node:Washer" and "Node:Washer Battery" in Figure 3.3). When the appliance is active, it will greedily seek out its own battery before attempting to use the central battery. Conversely, other appliances will not be able to access the node-specific battery.

Smart Appliance Scheduling with Green Energy Prediction

Smart appliances refer to the ability develop learned or automated behavior in appliances. A popular example is NEST thermostat [16], which learns temperature patterns in the home and automatically sets appropriate temperatures.

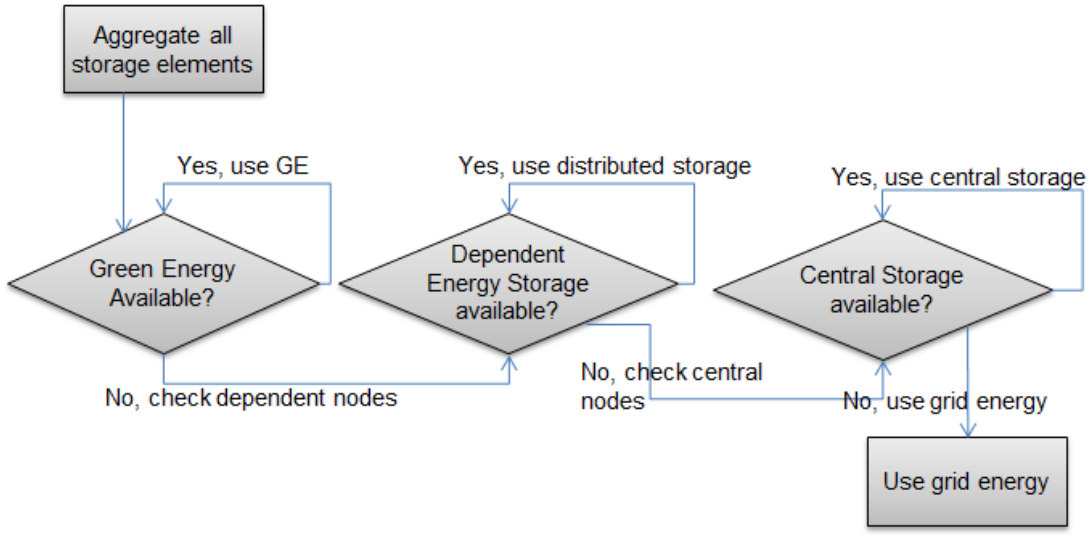

Figure 3.8: Scheduler to prioritize local distributed batteries.

Similarly, we envision adaptable appliances, a concept presented in [33], which set flexible deadlines for loads such as dishwashers, as their execution is typically open to rescheduling. While the *execute* phase in this case remains the same as in the previous section, the ability to reschedule appliances requires modifications of the event queue. The general approach is to predict the appliance usage and either use instantaneous or predicted green energy data to determine the best schedule for flexible appliances. Based on precedent from previous work [87] [60] [52], in this

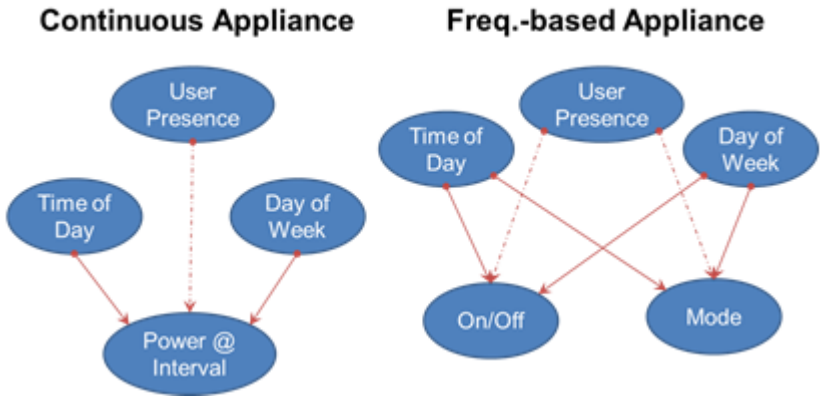

Figure 3.9: Continuous and discrete Bayesian networks for appliance prediction.

example we choose to perform appliance prediction using a learning model. With

a few input variables, techniques such as Support Vector Machines (SVM) or Artificial Neural Networks (ANN) are error-prone, while Bayesian networks and their derivative Hidden Markov Models are more appropriate. We use the more versatile Bayesian network, as in [52], where the random variables can easily be adapted to match the training set. Based on input data from the MIT REDD database [60], we develop the Bayesian networks shown in Figure 3.9 for each home appliance. Bayesian network probability variables can be obtained by counting. Equation 3.6 represents all the instances that a random variable matched the expected outcome over all training samples. The scheduler utilizes the data provided by the predictor to schedule appliances in an energy-efficient manner. The procedure of the algorithm is provided in Table 3.5.

$$p(X_i = x_j) = T^{-1} * count(x(t) = x_j) \; for \; all \; t = 1...T \tag{3.6}$$

Table 3.5: Smart appliance scheduling algorithm

```
power[] = predicted_renewable_schedule[]
for each inflex_appl[] ia s.t.  (ia.prediction >threshold):
  for each timeslot t:
    power[t] -= ia.power[t]

Sort flexible_intervals[] by max power consumed

for each flexible_slot fs s.t.  (fi.prediction >threshold):
  flexible_interval.scheduledInterval =,argmax(power[])
Recalculate power[]

Return flexible_intervals[]
```

The following summarizes the operation of the algorithm:

1. The expected available energy at each timeslot, *power[]*, is determined by the predicted green energy availability, *predicted_renewable_schedule[]*.

2. At this point, the energy consumption of all non-reschedulable appliances, *inflex_app[]*, is deducted from the potential energy available, in order to determine how much energy is actually free for use.

3. Based on the results in step 2, we can now schedule all reschedulable appliances, represented by the array *flexible_intervals[]*. It is important to sort them by the highest maximum energy consumption first, to match the largest consumers to the highest green energy slots.

4. Iteratively, the algorithm schedules each successive appliance and recalculates the new free energy for each timeslot.

5. The ultimate result is the scheduled slots for each reschedulable appliance in the variable *flexible_intervals[]*. This is then provided to the scheduler for execution.

In general, the algorithm determines the energy available at each interval based on the predicted solar energy. Depending on the prediction horizon, different predictors can be used. The predicted energy is reduced by the predicted schedule of appliances which do not have flexible deadlines, resulting in the expected unused solar energy at each interval. The scheduler then allocates each flexible-deadline appliance based on the highest green energy available in the 24-hour period. The scheduler iterates this process until all flexible appliances are allocated, and provides this schedule to the simulator.

3.3 Case Studies

HomeSim provides a versatile, configurable residential energy simulation platform capable of quantifying the impact of current and future technology improvements. In the following section, we validate its simulation accuracy against real traces, then we test our simulator using prevalent and proposed residential energy management scenarios and technologies:

1. Test and quantify the theoretical benefits of lithium-iron phosphate batteries (e.g. 5x cycle life) compared to the more traditional lead-acid batteries

2. Smart appliances rescheduled for better energy efficiency [33]

3. Replacing centralized batteries with distributed, appliance-specific batteries

4. Cost savings using each of the above technologies

5. Price-aware scheduling using time-of-use retail prices.

.

3.3.1 Input Data

All of our case studies use measured data from MIT's REDD project [60] for residential energy consumption. The dataset contains low-frequency (1Hz) readings of power consumption from the major appliances in 6 houses over two weeks. We use several datasets, representative of a typical home, with data for major appliances: stove, microwave, washer/dryer, refrigerator, dishwasher, and HVAC; and other, more minor energy loads: kitchen outlets, lighting, electronics, etc. These readings are composed into a schedule for each load, and are also used for appliance prediction.

Table 3.6: Experimental battery specifications

Specification	**LFP battery**	**LA battery**
Capacity (kWh)	18.6	18.6
Nominal voltage (V)	12	12
Charge/discharge cutoff (V)	14/10	14/10
Depth of Discharge limit	0.6	0.6
Lower/upper current limits (A)	300/400	150/250
Peukert ratio	1.05	1.15

Our renewable energy data is obtained from the UCSD Microgrid photovoltaics at 15-minute intervals, and normalized to match 35% of the residence's average consumption for a more appropriate rooftop solar capacity [4]. We incorporate lead-acid (LA) and lithium-iron phosphate (LFP) batteries into our analysis. Battery characteristics are described in Table 3.6. Battery pricing is obtained from [1] [15]. For cost models, utility pricing is obtained similarly to our data center approach in the previous chapter: wholesale energy prices from the California ISO, normalized to match retail SDGE pricing [93]. The pricing averages used in our experiments are shown in Table 3.10.

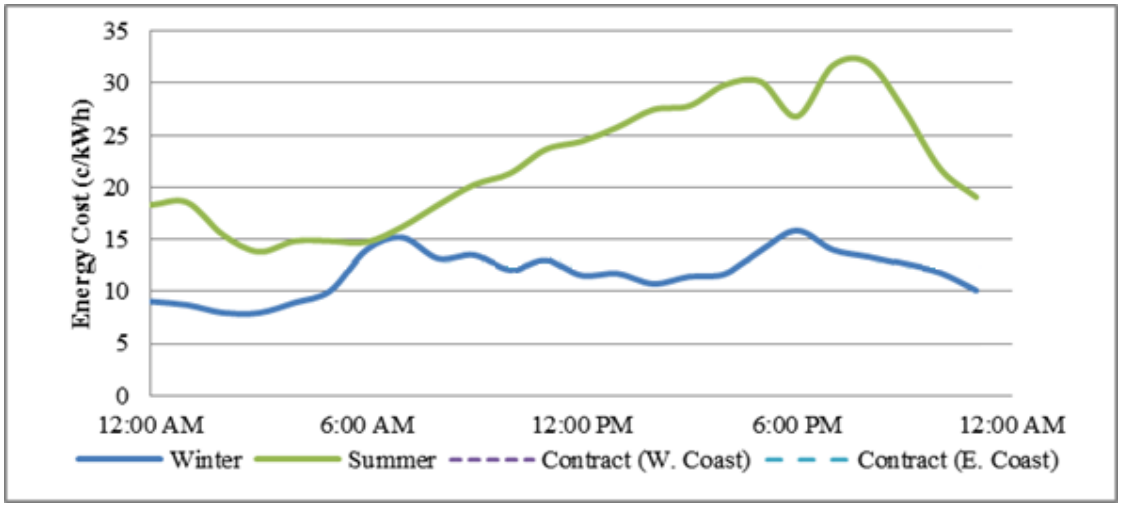

Figure 3.10: Wholesale time-of-use energy costs from CalISO scaled to residential retail levels.

3.3.2 Smart Appliances

The smart appliance scheduling algorithm requires a set of appliances that can be rescheduled and predicted green energy in order to correctly estimate when appliances should run. We use the washer, dryer and dishwasher as smart appliances with flexible schedules, with the flexibility parameters in Table 3.7. The flexibility range is defined statically for each of the flexible appliances using threshold categories specified in [56]. Appliance execution is determined by the Bayesian predictor specified in Figure 3.9. We set a threshold for the confidence level at which appliance prediction is considered valid, so that the predicted appliance can be scheduled. We derive this value empirically, varying the threshold over our training data in intervals of 0.1, and selected the minimum error (0.7, with mean error of 0.31).

Table 3.7: Flexible appliance scheduling

Appliance	**Flexible Schedule [56]**
Washer	Up to 12 hr before predicted deadline
Dryer	Up to 12 hr before deadline, within 2 hr of washer
Dishwasher	Within 6 hr after predicted deadline

3.3.3 Renewable Energy Prediction

Smart appliance scheduling can also leverage predicted green energy, which, for our experimental setup, is solar. Referencing a quantitative comparison of several time-series prediction algorithms from the previous chapter, we take advantage of the reasonable accuracy (<10% error) and low overhead of the Weather Conditioned Moving Average (WCMA) prediction algorithm, which predicts 24 hours in advance. We empirically determine the weighting factor α with lowest prediction error at $\alpha = 0.45$ for the UCSD photovoltaic dataset. This predictor is used to estimate solar energy availability for the 18h window we used for appliances listed in Table 3.7.

3.3.4 Simulation Engine Validation

We validate HomeSim's simulated energy output against the actual measured REDD energy values and appliance instances. We also validate the battery model above against real battery traces [24]. Our metric is the mean absolute error (MAE) of each individual model, as well as the MAE of the overall simulation. The results provided in Table 3.3 show that our simulator very accurately estimates the appliance power consumption at the appropriate granularity, though the error increases to 10% when simulated with input models that are discretized to 1 min. This is not unexpected, as at such granularity simulation of very short-duration appliances loses accuracy. The higher error in HomeSim compared to the total energy provided by the grid is caused by the non-ideality of transmission and conversion. While HomeSim models these losses (see *Conversion Factor* in Table 3.1), their nonlinearity inevitably introduces some noise in total simulation accuracy.

Table 3.8: Model Validation Error

Model	**Mean Absolute Error (%)**
Battery State of Health (SoH)	8
Avg. appliance power per interval	0.2
HomeSim total energy consumption	7

3.3.5 Case 1: Battery Technologies

Related research discusses the use of batteries and the advantages they provide, whether to improve the efficiency of renewable sources [33] or to reduce the energy costs via TOU pricing. These works assume lead-acid (LA) batteries, the most popular option. However, recent work motivates the use of lithium-iron-phosphate (LFP) batteries over LA for residences, citing 2.7x increase in energy density and 5x improvement in cycle life [15]. Neither work can compare or quantify the difference between the two scenarios. The simulator in [33] does not take into account a sophisticated battery model, opting to use a linear model instead, while work presented in [73] requires physical measurements to test the results, severely limiting its applicability. In contrast, HomeSim can easily model and test both types of batteries and quantify the differences.

We simulated and compared a lead-acid (LA) battery with an equal-volume lithium iron-phosphate (LFP) battery. We show the differences in the various power characteristics in Table 3.9, where Green Energy Efficiency (GEE) refers to the fraction of green energy used for useful work (running loads or charging batteries) compared to the total available renewable energy in the system. By taking into account the state of health, we can estimate the lifetime of each battery in the given scenario.

The results do not match the theoretical 5x improvement in battery life due to the complexity of interaction between the batteries, the solar source, and the different loads. While we see both a reduction in total grid energy and improvement in green energy efficiency, the latter is due to additional energy spent charging the larger LFP battery. Due to the larger capacity and slower overall recharge time, the LFP battery spends a majority of time in low SoC, accrues a higher number of total charge/discharge cycles, and reduces the LFP lifetime from a theoretical 5x to under 3x. So, while there is an improvement with the LFP battery, HomeSim demonstrates that it is tempered by the particular configuration of the system.

Table 3.9: Battery technology results

Model	LA Battery	LFP Battery
Total Grid Energy (kWh)	78.5	63.8
Green Energy Efficiency (%)	20.4	23.1
Average SoC	0.47	0.49
Estimated Lifetime (yrs)	2.3	6.08

3.3.6 Case 2: Reschedulable Appliances

With HomeSim, we can explore the possibility of rescheduling appliances with flexible deadlines. We used the flexibility ranges per appliance, outlined in [33] [56] to obtain hours-of-operation ranges for each appliance, as listed in Table 3.7.

Table 3.10: Prediction model validation

Prediction	Mean Absolute Error (%)
Solar Energy Prediction	9
Appliance Prediction	31
Appliance Prediction (+/- 2 timeslots)	14

Table 3.10 summarizes the accuracy of the predictors that we use in the *Reschedulable Appliances* case. The error for appliance prediction compared to the actual appliance usage traces from REDD is due to the discretization of a more continuous sample. These errors are aggregated when energy consumption is calculated. The appliance prediction incurs 31% mean error. However, as the last row demonstrates, appliance prediction is significantly improved when verifying appliance prediction within +/- 2 timeslots. In the reduced granularity of the predictor, appliances that execute across a timeslot boundary may be predicted to execute in either slot. Qualitatively, however, predicting an interval early or late does not make a significant impact on the efficacy of rescheduling, as solar energy values are comparable for adjacent intervals.

The results in Table 3.11 demonstrate that rescheduling appliances has a positive impact on the total energy drawn from the grid and green energy efficiency (GEE), with a reduction in total grid energy by nearly 25%. Green energy efficiency was improved, as seen in Figure 3.11 on the right, where a rescheduled

Table 3.11: Reschedulable appliance results

Model	LA Battery	LFP Battery
Total Grid Energy (kWh)	83.0	61.6
Green Energy Efficiency (%)	41.5	47.7
Green Energy Sold to Grid (kWh)	53.7	48.0
Grid Energy Cost ($)	21.1	15.65

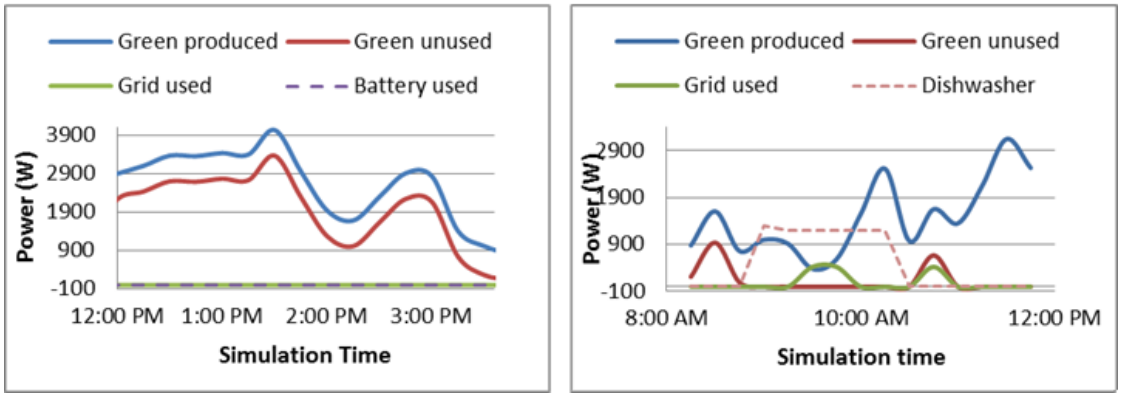

Figure 3.11: Low GEE intervals (left) vs. high GEE intervals (right).

dishwasher at 9:00 AM consumes all available green energy. However, this efficiency improvement is limited by the fact that battery usage was slightly reduced, resulting in more intervals where there was surplus green energy and all batteries were fully charged. This case is displayed in Figure 3.11 on the left, where only a constant 600W is needed to run an appliance, with the rest of the green energy unused. In a residence that can backfeed energy, the unused solar output can be sold back to the grid, as shown in Table 3.11. The cost of grid consumption is also presented, with 26% in cost savings when using LFP batteries.

3.3.7 Case 3: Distributed Batteries

The distributed battery example stems from recent industrial research and development into associating batteries with appliances [75]. An appliance first uses its own battery before reverting to other sources. In the case of a centralized battery, the battery experiences a sustained drain on its energy from the combination of loads, forcing a more frequent fallback to grid energy. Load-proportioned distributed batteries, however, are sized to meet the capacity of the associated

appliance, and can better sustain the appliance without reverting to grid energy. In testing distributed batteries, we apportioned distributed batteries to the large appliances based on a ratio of their power consumption, normalized against the total capacity (18.6kWh) of the single centralized battery.

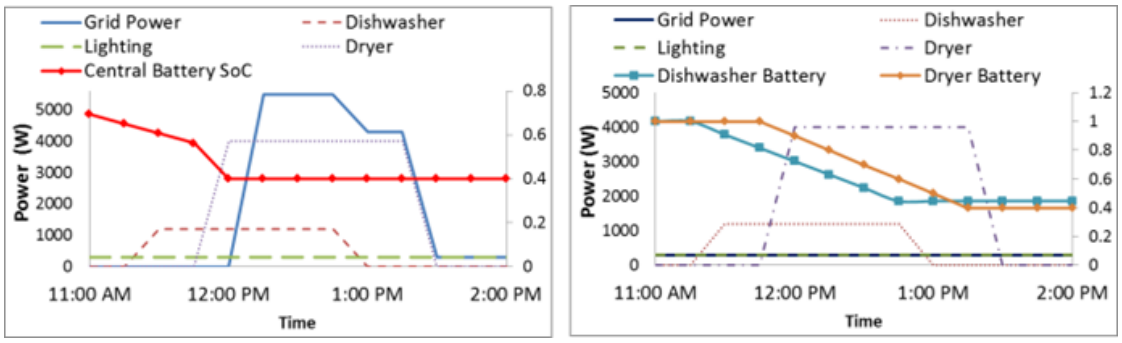

Figure 3.12: Centralized (left) vs. distributed (right) battery power use.

The results in Table 3.12. show that the total grid energy consumption drops by a factor of 1.5x, while battery consumption and green energy efficiency increase. By comparing the traces, as in Figure 3.12, we can see that large appliances that have a built-in battery are less susceptible to requiring grid energy, especially at times when the net load is high. The batteries store sufficient amount of energy for the connected appliances, delivering enough to prevent reliance on the grid, even when there is not enough instantaneous renewable energy available. The improvement in green energy efficiency comes from the fact that distributed batteries can be charged in an ad-hoc manner, providing a level of parallelism to charging that was not previously possible. Finally, by calculating the operational costs of each method, we show 36% improvement when leveraging distributed batteries over centralized.

Table 3.12: Centralized (fixed) vs. distributed battery results

Model	**Central Battery**	**Distrib. Batteries**
Total Grid Energy (kWh)	130.6	83.0
Green Energy Efficiency (%)	23.1	41.5
Total Battery Energy (kWh)	25.3	35.6
Grid Energy Cost ($)	33.2	21.1

3.3.8 Case 4: Cost Savings

By integrating cost calculations to the scheduler, HomeSim can evaluate the cost-benefit of centralized batteries, distributed batteries, and rescheduling appliances, as well as provide a comparison among the three for the same test case. Using the operational costs outlined in [93], we execute HomeSim for the same residence configuration, alternately using the three cases above for comparison. For rescheduled appliances, we extended the distributed battery case to determine additional benefit over the most financially viable case.

Table 3.13: Operational Costs of Centralized, Distributed, and Rescheduled Appliance cases

	Base Case	**Central Battery**	**Distributed Batteries**	**Rescheduled Appliances**
Avg. Monthly Cost ($)	89.2	73.14	69.81	62.96

Table 3.13 highlights the operational costs of each of the cases compared against the base case of no energy management. Incorporating any energy storage provides a cost reduction of 18%, which is further improved with distributed batteries. In this configuration, it is important to note that the savings with distributed batteries is only 5%, much less than the 36% improvement in Table 3.12, demonstrating that the technologies have variable impacts in different homes. Rescheduling further improves energy savings, with a net monthly energy reduction by 30%.

Similarly, we can incorporate capital costs into HomeSim, and extend the scheduler to calculate the recoupment of the cost of batteries and local solar generation. Using the capital cost numbers from battery manufacturers [15]and retail energy sources [93], HomeSim calculates the net energy savings weighed against the capital costs. Because our input data only lasts 2 weeks, we cycle the data until the savings exceed the costs. We compare the central and distributed battery cases as well as the case of distributed+rescheduled appliances. Finally, we also investigate the mixed case, including an additional 18.6 kWh central battery to the appliance-specific distributed batteries and rescheduling. This final case provides

insight into how savings scale with battery size.

Table 3.14: Recoupment Time (in years) for Centralized, Distributed, Rescheduled Appliance, and Mixed cases

	Central Battery	**Distributed Batteries**	**Rescheduled Appliances**	**Mixed**
Recoupment Time (yrs)	22.6	20.2	16.6	11.9

Table 3.14 compares these results, demonstrating very large recoupment time, over 22 years, in the centralized battery case. This scenario is the current state of the art, but is shown to have an unreasonably low cost-benefit. The distributed batteries case, while an improvement, has a similar duration. Rescheduled appliances, however, reduce the net-even time by 20%, and the introduction of an additional reduces the recoupment time to almost 50%. Since solar costs represent a large majority of the capital, increasing total battery capacity demonstrates good scaling in the time to net-zero. It is important to note that these experiments do not consider selling energy back to the grid, which is a further increase in savings, and also that after the recoupment time, all energy savings result in a net profit.

3.3.9 Case 5: Cost-aware Scheduling

Time-of-use (TOU) pricing is prevalent in European energy provision, and the concept is growing in relevance in the United States [73]. Integrating pricing information into the scheduler requires extending the algorithm in Figure 3.8 to further sort intervals by the lowest energy cost when grid energy is needed. Our comparison is between the reschedulable appliances scheduling algorithm in the previous case study, which had the best performance, to the algorithm with the inclusion of cost awareness. We compare the cost savings of two cases by maintaining the same building and battery configurations.

Table 3.15 compares reschedulable appliance scheduling to the inclusion of cost-aware scheduling. The results indicate that cost-aware scheduling has an almost negligible impact on grid energy savings, and no impact on green energy efficiency. Analyzing the output traces demonstrates the reason: the net benefit

Table 3.15: Reschedulable Appliance Scheduling vs. Cost-Aware Scheduling

	Rescheduled Appliances (RA)	**RA + Cost**
Avg. Weekly Grid Energy (kWh)	62.96	61.02
Avg. Weekly Green Energy Efficiency (%)	49.8	49.8

of a small amount of available green energy (which is free) is more favorable than an interval with lower cost. Therefore, it is more feasible to greedily use green energy than it would be to rely on a cheaper grid interval. In total, fewer than 10 jobs were rescheduled due to cost savings, resulting in a marginal reduction in grid energy, and almost no impact on green energy efficiency.

3.4 Extended Case Study: Optimized Residential Battery Usage

HomeSim enables comparing different complex energy scenarios to identify benefits. Electrochemical storage such as lead-acid, lithium-ion, and phosphate batteries allow residences to store locally generated energy for appliance use, as well as in general for load shifting [82], and peak power shaving [24].

However, batteries have nonlinear charging, discharging, and degradation properties that increase the difficulty of optimizing their usage. Within HomeSim, we integrated an open-form battery model. Previous works focus on the application or usage scheme, neglecting the impact of realistic battery degradation. In this case study, we develop a more accurate closed-form battery model for optimization and test it in simulation using HomeSim. We identify more than 2x error in estimated total savings that prior work claimed due to inaccurate battery modeling, but still provide a return on the investment of the battery with our new battery model.

3.4.1 Batteries in Residences

Several works have investigated battery integration in residential systems. The majority focus on integrating batteries as part of other objectives: peak power

shaving, voltage regulation, and cost and energy reduction. Ratnam et al. [82] use a quadratic program to minimize the energy needed from the grid. They utilize residential photovoltaics with a co-located battery. The storage afforded by the battery allows the residence to shift the grid energy profile, avoiding peak pricing. However, they assume an ideal battery model, not taking into account the nonlinear properties. Similarly, van de ven et al. [90] use a Markov Decision Process to optimize shifting stored energy to high-demand periods in houses, also using a linear battery model. Leadbetter et al. [66] use a battery storage system for peak power shaving, leveraging the model to determine the size of the system. They limit the battery state of charge (SOC) to between 15%-85% to minimize decay and extend cycle life, but do not account for the decay itself in the model.

Other works do attempt to account other nonlinear battery properties, including decay. Tant et al. [88] perform a multi-objective optimization, investigating the trade-off between voltage reduction and peak power shaving. They account for both battery degradation and capital cost. Due to the complexity of their ultimate model, they use a convex approximation. Aksanli et al. [28] generate a closed-form inequality for optimal battery configuration in residences using Coulomb counting model to constrain their battery use. Due in part to the complexity of Coulomb counting, however, they need to perform extensive simulation in order to reach their conclusions rather than a single-step optimization.

While there are several battery models that provide accuracy of battery characteristics, their complexity precludes use in optimization formulations, and most of the related work that uses batteries choose the inaccurate, if simpler, ideal model. Some works attempt to use the accurate models, but have to defer to approximations or simulation to draw conclusions. We aim to exploit some battery characteristics (depth of discharge level, charge/discharge rate) and approximate others (effective capacity) to provide a trade-off in the battery models, providing a closed-form optimization model that still preserves the important degradation characteristics of the battery.

3.4.2 Problem Formulation

Effective battery use relies on a practical scheme that balances battery charge and discharge cycles with the goal of leveraging savings into a return on the capital investment (ROI). This can be accomplished by reducing local energy consumption [88], smoothing peaks to avoid peak-power scenarios, [24], or time-shift usage to cheaper consumption periods [82]. All of these cases rely on varying utility energy prices, allowing storage to exploit high-priced peak power conditions and low-priced intervals. We can also accomplish this by buying, storing, and selling energy based on knowledge of changing utility energy prices. This approach enables us determine the true operational cost and lifetime of the battery beforehand and determine parameters for usage. Furthermore, in contrast to the battery usage methods shown in the related work, we aim to maintain an accurate battery state, including the degradation in capacity over time and the expected lifetime in our formulation.

Battery Cost and Return on Investment (ROI)

The ultimate goal of using the battery is to provide financial benefit. Another possibility for savings or recouping the investment is to treat varying prices as part of an energy market, selling during the high-priced periods and buying during the low-priced periods.

The *return on investment (ROI)* of the battery is the point at when the capital cost of the battery is recouped through the savings of selling stored battery energy. After this point, the battery has paid for itself, and any further savings are purely profit. While several works allow the battery to be charged/discharged at will within the constraints of an algorithm, the fact is that a battery's lifetime is limited: the manufacturer's cycle limit, in conjunction with the battery's charge/discharge cycles and degradation per cycle determines the lifetime of the battery, and ultimately, restricts any battery optimization.

In our approach, we base the ROI on the cycle life, as a battery cannot be used beyond this point. Additionally, as we aim for optimization over multiple years, we limit the amount of cycles consumed per year, to allow the battery

to last for the duration of a target lifetime. As a result, unlike previous work, we predetermine an ROI target time (e.g. 5 years) and optimize the use of the battery over that lifetime. In some very short target times (e.g. 1 year), it is impossible to buy/sell enough energy in order to recoup the investment, but iterating over different lifetimes enables us to optimize battery use to an extent where the user profits.

Time-of-Use Energy Market

The utility energy market is shifting from uniform pricing to more flexible strategies such as tiered pricing, which is already used by utilities to curtail overconsumption [24]. As the grid trends towards a fully distributed generation system, another interesting scenario is time-of-use pricing (TOU). This implementation is a retail parallel of the current wholesale pricing that is offered to utilities by independent system operators [10]. TOU pricing is a reflection of grid supply and demand, and varies regionally and seasonally. Utility operators leverage varying pricing in order to maintain grid stability, incentivizing consumers to increase/decrease energy usage to balance voltage deviation. Figure 3.13 shows the annual price distribution for Houston, San Diego, and Boston, demonstrating a regional variation in prices.

Retail energy load analysis demonstrates annual usage patterns [36], which are reflected by electricity unit price changes. These changes in usage create low/high points in the distribution and provide charge/discharge intervals to exploit for savings.

Real Battery Simplification

Several models for batteries have been proposed: electrochemical, electrical-circuit, analytical, and stochastic [58]; all of which allow complex use of batteries by varying charge/discharge rates, DoD, and adjusting the cycle life, effective capacity, and SoH. Complex, granular use of the battery allows for fine control of all the free variables, but the prescribed usage conditions from the related work can be accomplished by fixing the charge/discharge voltage and based on manufacturer

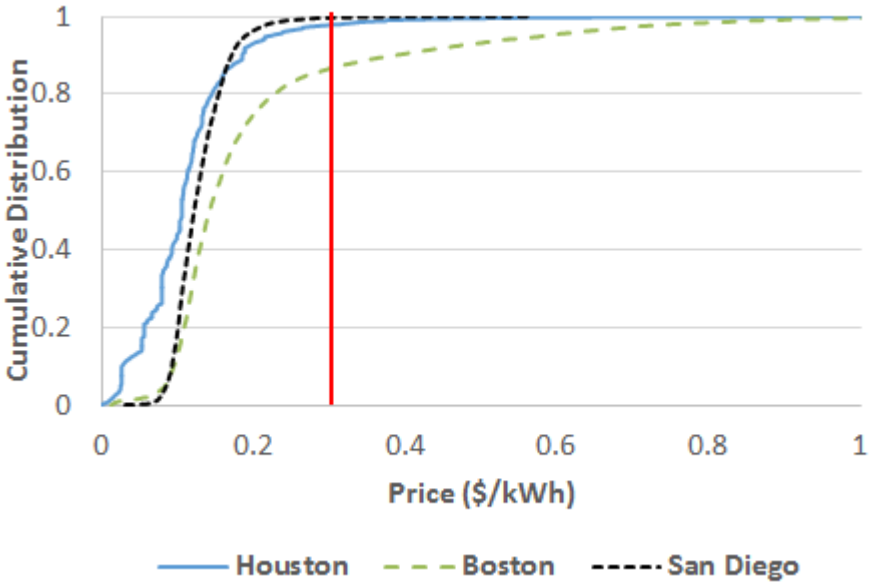

Figure 3.13: Cumulative density distribution of annual regional electricity price ($/kWh) for Houston, Boston, and San Diego.

cycle-life curves, and setting the maximum DoD, avoiding the trade-offs to battery lifetime. This allows the cycle life, the charge and discharge rates, and change in SoC to be fixed and predictable. This simplified battery scheme leaves only the effective capacity of the battery as the nonlinear change over its lifetime. As seen in Figure 3.6, the degradation in capacity under manufacturer-recommended conditions (e.g. 4.2V, 4.25V) offers a large linear region with smaller nonlinear outliers. As a result, we choose to approximate the change in effective capacity with a linear simplification. In the next section, we formulate a battery usage optimization problem using these conditions.

3.4.3 Optimization Problem

The overarching goal of the problem is to optimize the benefit of battery use over the specified duration (in our case, one year) by discharging battery capacity when provided with TOU pricing:

$$max \sum_{t} P(t) * d(t) - P(t) * c(t) \tag{3.7}$$

where $P(t)$ is the electricity unit price at time t, $d(t)$ is the energy discharged by the battery, and $c(t)$ is the charging energy. We defined the return on investment (ROI) as a predefined time limit Y: the target number of years for the battery cost to be recouped. We allow the battery's SoH to reach the end of life (SoH_{dead}) at the end of Y years.

Optimizing the profitability of battery usage over this time will determine if the initial cost can be recouped and if any profit can be gained in addition to the ROI. We also note that electricity pricing has an annual trend. Consequently, we divide the battery's total lifetime capacity D into the expected ROI time Y, as it maximizes the expectation of reaching recoupment. This limits our maximum annual discharge energy to $DY = D_{max}$. This restriction defines the inequality equation for the ILP problem:

$$\sum_{t} d(t) \leq D_{max} \tag{3.8}$$

The ILP's equality equation updates the battery capacity of the next iteration based on the charge and discharge events of the current iteration: is that the battery capacity at the next interval is dependent on the capacity on the current interval and the charge/discharge of the current interval:

$$B(t+1) - B(t) + d(t) - c(t) = 0 \tag{3.9}$$

where $B(t+1)$ is the remaining battery capacity for the next interval, and $B(t)$, $d(t)$, and $c(t)$ respectively represent the battery state, discharge, and charge energy of the current interval.

The remaining constraints limit the battery to the manufacturer-recommended specifications in order to minimize the impact on cycle life; namely, the charge and discharge limits (c_{nom} and d_{nom}, respectively), and the minimum and maximum charge on the battery (C_{min} and C_{max}). The latter limits correspond to the DoD

limits for the battery as specified by the manufacturer:

$$C_{min} \leq B(t) \leq C_{max}$$

$$0 \leq d(t) \leq d_{nom} \tag{3.10}$$

$$0 \leq c(t) \leq c_{nom}$$

While a battery's properties are strictly linear, the above equality equations and constraints hold on their own: the battery is limited to its depth of discharge, and the charge and discharge rates are limited to levels that do not adversely affect cycle time. However, even with the above constraints, the battery's effective capacity still decays over time. Specifically, C_{max} does not remain constant throughout the lifetime of the battery. Instead, we now consider it a variable that changes over time:$C_{max}(t)$, which corresponds to the effective capacity degradation. We modify the first inequality in Equation 3.10, changing C_{max} to $C_{max}(t)$. The additional equality equation updates the current effective capacity changes with every discharge event:

$$C_{max}(t+1) = C_{max}(t) - [\Delta SoH_{rate}] * d(t) \tag{3.11}$$

where ΔSoH_{rate} represents the rate in percentage reduction in the state of health per discharge energy. Because we use a linear approximation here, ΔSoH_{rate} is determined by the total change in SoH over the total number of cycles in the lifetime of the battery. Multiplying it with $d(t)$ allows the reduction to be proportional to the DoD in a particular cycle.

3.4.4 Experimental Setup

The goals are to determine the accuracy of the optimization outlined in the previous section, and to show how we can leverage our scheme to demonstrate an upper bound for reasonable ROI under realistic conditions. The following subsections outline the data and experiment setup for the battery, utility pricing, and verification of the accuracy of the optimization.

Battery

The properties of the battery are important for populating the constraints and variables in the ILP formulation. As such, we chose a real battery intended for grid-connected operation. The most typical chemical storage for these applications is lead-acid (LA), but the newer lithium-iron phosphate (LFP) batteries provide better current and voltage properties, and more important for our assumptions, a more stable degradation profile [24]. We select a battery from Balqon Corporation [10] that has the specs described in Figure 3.16.

Table 3.16: Balqon Lithium-Iron Phosphate Battery Properties

Property	**Value**
Rated cycle capacity	34kWh
# cycles	3000
Capital cost	$12,000.00
Voltage range	42-60V
Nominal voltage	48V
Current range	0-1000A
Nominal current	500A
Maximum DoD	0.4
Minimum SoH	0.4

We can calculate the battery's operational cost using the total usable energy of the battery over its lifetime and the capital cost:

$$\text{Operational Cost} = \frac{\text{Capital Cost}}{\sum_{t-1}^{\#\text{cycles}} C_{max}(t)} \tag{3.12}$$

With the specifications in 3.16, the operational cost of the Balqon battery is $0.24/kWh. This precludes consistent usage for local loads such as appliances because, as the distribution in 3.13 shows, at times it is cheaper to just use grid electricity rather than stored charge. Additionally, by taking appliances out of the equation, the focus remains on recouping the capital cost of the battery independently of usage patterns.

The other dependent factor for the ILP formulation is the target ROI lifetime, as it is used to calculate D_{max}. This in turn determines how aggressively the battery is cycled each year, since we expect to reach the lifetime of the battery

Figure 3.14: Balqon LFP Battery for Grid-Connected Residential Applications [10].

at the end of the target time. This is not a part of the formulation itself, so we instead iteratively provide longer deadlines to the optimization problem, from 1-10 years, by which point factors outside of charge cycling (i.e. chemical breakdown) accelerate the degradation of the battery.

Energy Costs

The appropriate use of TOU pricing is critical for correct battery scheduling. As previously mentioned, wholesale pricing shows a high-level correlation on a yearly basis, but within a year, there is significant variation in high and low prices. Additionally, while wholesale pricing is a true TOU system, retail pricing tends to be flat or tiered, with only a few utilities providing a true TOU scheme. To account for this, we scaled available wholesale pricing for different regions to local

retail prices, providing an emulation of retail TOU pricing.

We investigated three separate locations with different price profiles, retrieved from the 2013 record of local wholesale pricing independent system operators (ISOs): Boston [13], Houston [6], and San Diego [3]. Leveraging the yearly pattern, we extrapolate over the target lifetime by repeating the scaled annual costs each year. A summary of characteristics is shown in Table 3.17.

Table 3.17: Scaled retail pricing characteristics for different locations

Location	**Avg. retail energy price ($/kWh)**	**Standard Deviation ($/kWh)**
Houston	0.10	0.06
Boston	0.20	0.18
San Diego	0.13	0.03

Houston has the lowest retail price, in part due to very high energy availability. High summer temperatures contribute to moderate deviation in the costs throughout the year. In contrast, Boston has very high energy demand and more tightly constrained energy availability, leading to both the highest average price and deviation. San Diego strikes a middle ground, with very consistent usage and pricing: it is the median in average price, but the low deviation in consumption translates to the lowest deviation in price.

We optimize the battery over a full year's worth of pricing data, replicated for every subsequent year of the target time. While such a long-term optimization is somewhat infeasible, as knowing such long-term pricing is unrealistic, it does provide an upper bound for the optimization, and invites further analysis for different optimization schemes, such as weighting cycles based on weekly or monthly trends and predictions.

3.4.5 Results

Optimization Accuracy

The first set of experiments establishes the accuracy of the battery optimization proposed in this chapter. The goal of the formulation was to reduce the inaccuracy of an ideal battery model (represented by using only Equations 1-6).

We solve the LP problem, then compare the final SoH, calculated as the ratio of effective capacity to the rated capacity, against a simulation of the resulting charge/discharge using HomeSim [92]. Table 3.18 outlines the comparison of the error in SoH for all three locations with a 5-year ROI deadline.

Table 3.18: Mean absolute error for the decaying battery problem formulation compared against simulation

Location	**SoH Mean Absolute Error (MAE) (%)**
Houston	4.5
Boston	5.2
San Diego	3.8

In all three cases, the LP formulation maintains within 6% error of the simulation, demonstrating high accuracy. By fixing both the discharge rate and the DoD and accounting for the reduction in effective capacity, the formulation maintains a good model of a battery.

Linear vs. Capacity-Degrading Battery Model

We next compare the degrading battery formulation to the linear battery model used in the related work. This is the formulation represented by the naïve optimization in Section 4.2.4 compared with the incorporation of degradation introduced by Equation 3.11. The SoH is not an appropriate metric for comparison, since the linear model does not degrade. However, the overall calculation of the ROI between the two models represents an appropriate metric for comparison. For example, a subset of the ROI results for San Diego are shown in Table 3.19. As the ROI target becomes longer, the battery usage becomes less aggressive, and there are more opportunities for net savings (positive) as opposed to remaining in a deficit (negative).

The linear battery deviates from the decaying model from 3% up to 106% (for ROI of 5 years in Table 3.19), demonstrating over 2x error when disregarding the effective capacity decay.

Table 3.19: A subset of the annual savings over different ROI results for San Diego, comparing the linear vs. the decaying battery model

Target ROI (yrs)	**3**	**4**	**5**	**6**	**7**
Linear	\$-1056.79	\$-126.11	\$393.35	\$713.20	\$925.32
Decaying	\$-1321.56	\$-352.36	\$190.66	\$505.90	\$745.67
Linear model error (%)	20.1	64.2	106.8	41.2	24.2

Regional ROIs

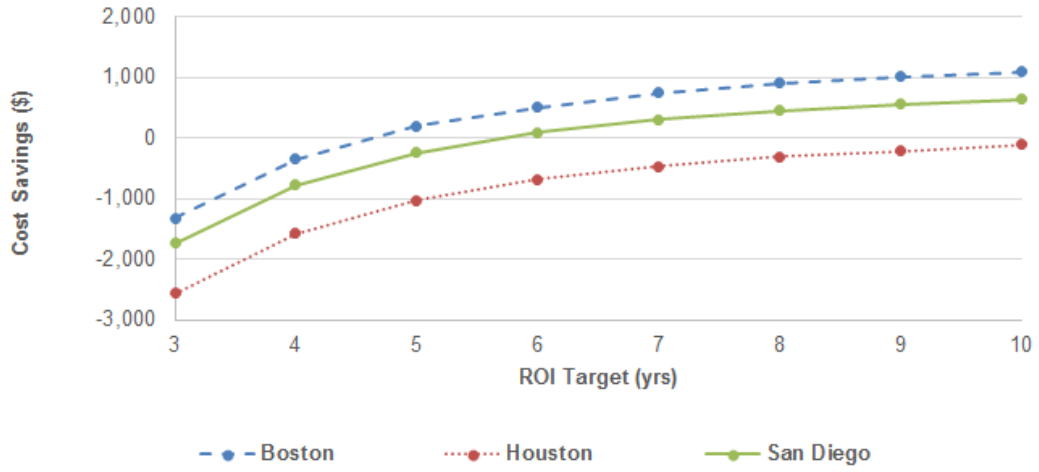

Figure 3.15: Annual battery-based savings for different ROI targets for San Diego (recouped in 6 years), Houston (never recouped), and Boston (recouped in 5 years).

Figure 3.15 illustrates the annual savings achieved by different ROI targets for the different cities in our experiments. The savings are calculated as the difference in battery-based profit and the amortized cost of the battery per year over the lifetime of the battery. Numbers below 0 indicate that the battery costs could not be recouped each year, and ultimately, over the battery life.

Boston demonstrates a return on investment of the amortized cost of the battery in 5 years. From that point onward, the consumer nets a profit, of over \$1,000/year with a 10-year ROI. The high average energy cost combined with the high deviation (see Figure 3.13) allowed many intervals where buying low and selling high could be exploited. As a result, it provided the quickest ROI time among the three test cases.

Houston presents an unsuccessful scenario for the optimization problem.

It combined the lowest energy pricing with low deviation. Even optimizing over a year, and extending the ROI target, and consequently, the lifetime, of the battery to 10 years could not provide enough savings to warrant the use of the battery, with a loss of $109/year. The flat, cheap utility costs were simply too low, and the overwhelming capital cost of the battery could not be overcome. The next section extends the case study to extrapolate to future decreases in battery costs.

San Diego provides a middle ground between the two other cities. While it too had low utility energy costs and the lowest deviation, the increase in the average cost over Houston provided enough opportunities when given longer ROI timelines. Consequently, the consumer recoups the cost of the battery in just under 6 years. Between 6-10 years, the battery provides a profit of $89-$636/year. Additionally, while the overall standard deviation was low, the few outlier peaks and valleys were significantly higher/lower than Houston, which, when combined with a longer ROI horizon, provided more lucrative selling points.

Decreasing Energy Costs

The results of the Houston case study in the previous subsection illustrate that the capital cost of batteries may remain prohibitively high for energy trading in some retail markets. However, battery technology improvements are driving these prices lower. We extrapolated this trend with the Balqon battery by reducing its operational expenses from 0.25 $/kWh over the battery's lifetime, the current state of the art, to 0.05 $/lifetime-kWh, in $0.05 increments. Figure 3.16 maps the resulting change in the net cost for Houston over both different lifetimes and the reducing cost of the battery.

Already, with the reduction in operational cost from 0.25 $/kWh to 0.20 $/kWh, we see the ROI being fulfilled in 8 years. As the price drops further, the ROI drops to as few as 4 years for 0.10 $/kWh. Similarly, ROI times for San Diego and Boston were reduced to as little as 1 year. This demonstrates the growing feasibility and profitability of energy trading in the near future.

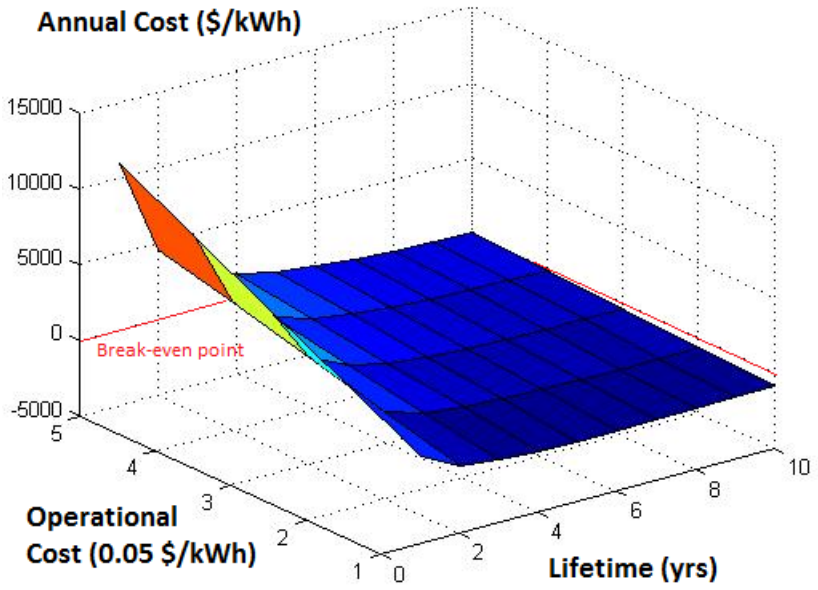

Figure 3.16: Annual cost of the battery under Houston TOU pricing when battery costs are reduced.

3.5 Conclusion

In this chapter, we introduce HomeSim, a powerful and extensible simulator for the increasingly relevant field of residential energy management. HomeSim provides a configurable simulation environment that is extremely versatile, to quantify and compare present and future improvements in residential energy consumption. We then investigate emerging technologies and procedures in residential energy management by performing a series of case studies targeting proposed improvements to residential energy management: the current state-of-the-art centralized battery, as well as distributed batteries, reschedulable smart appliances, and cost-aware scheduling using real data from instrumented houses. We further demonstrate the cost savings of each case by integrating capital and operational costs, as well as determine the time to recoupment for different technologies.

Finally, we illustrate an extensive end-to-end case study using HomeSim, first identifying inaccuracies in simplified battery models and developing a more accurate low-overhead battery model for linear battery scheduling optimization. We use HomeSim to simulate the battery and verify that our model is accurate,

with <5% error of simulation. We then formulate a linear programming solution using the new model and execute it in practical scenarios with real data, demonstrating an upper bound for the return on investment for the battery in as few as 5 years.

Chapter 3 contains material from Jagannathan Venkatesh, Bariş Akşanli, and Tajana Šimunić Rosing, "Residential Energy Simulation and Scheduling: A Case Study Approach", which appeared in Proceedings of the International Symposium on Computers and Communications (ISCC), 2013 [93]. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from Jagannathan Venkatesh, Bariş Akşanli, Jean-Claude Junqua, Philippe Morin, and Tajana Šimunić Rosing, "HomeSim: Comprehensive, Smart, Residential Electrical Energy Simulation and Scheduling", which appeared in Proceedings of the International Green Computing Conference (IGCC), 2013 [92]. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from Jagannathan Venkatesh, Shengbo Chen, Peerapol Tinnakornsrisuphap, and Tajana Šimunić Rosing, "Lifetime-dependent Battery Usage Optimization for Grid-Connected Residential Systems", which appeared in Proceedings of 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES) 2015 [95]. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Improving Residential Energy Modeling with User Context

In the previous chapters, we investigated residential energy consumption. We have so far dealt with static models and improvements - using the additionally available data as ground truth. However, this additional context is 1) not always readily available, and 2) strongly tied to the behavior and preferences of users [52] [26]. The advent of the Internet of Things (IoT)—the collection of sensing and actuation backed by the existing and growing Internet infrastructure [76]—can provide the context needed for user-driven residential energy automation. We now aim to use this additional user context to determine indiivdual appliance and house energy flexibility. Our approach to modeling, training, and generating this context in an efficient and scalable way is highlighted below.

Pre-IoT work in ubiquitous sensing still envisioned a level of compatibility and control over the sensors in the systems [46] and applications that used a manageable amount of raw sensor data. The number of available sensing and actuation devices has grown rapidly in the last few years [51], promising a truly pervasive sensing and actuating environment. In addition, ubiquitous connectivity and cloud storage have largely mitigated the primary research issues in the pervasive sensing fields. Reliability of communication and storage allows us to focus on the application layer: IoT applications operate in a world of changing inputs and available compute nodes as sensors and devices enter and exit an application's domain. The

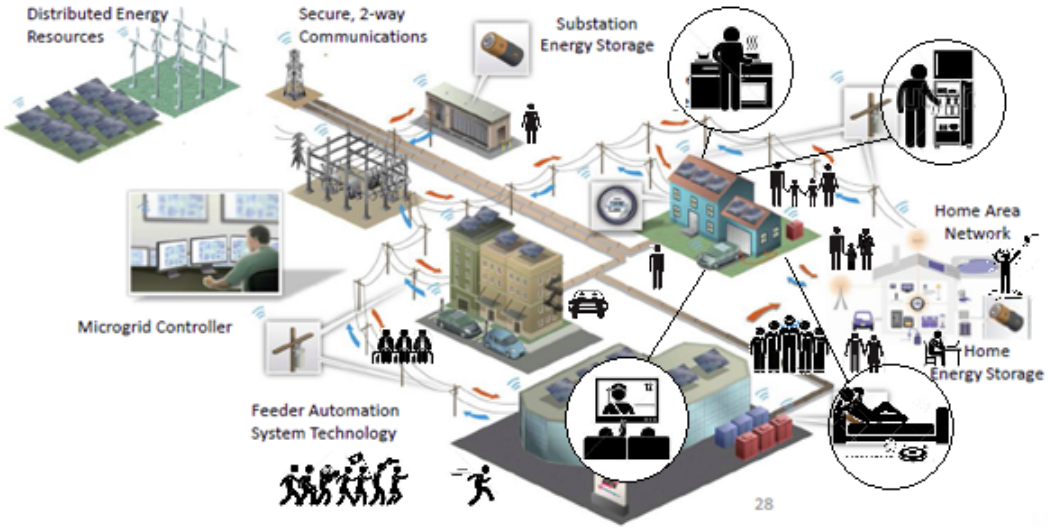

Figure 4.1: The residential smart grid is driven by user activities and preferences.

raw data in these applications will go through several levels of abstraction, combination, or distillation to produce a high-level description of the environment (and its users) with discrete, semantic states called *context*. Discrete, high-level context facilitates intuitive reasoning at the cost of raw data precision, and can be reused across applications.

These *context-aware* Internet of Things applications are ideally suited for determining user behavior for the residential smart grid: their main goal is to leverage the available data to drive automated actuation, such as in smart environments, distributed microgrids, or user-centric automation. They operate on dynamically changing ontologically-defined data called context data whose type, range, and sources are specified in an interface. However, current context-aware applications are still end-to-end implementations tightly coupled to the initial infrastructure and platforms, where each application maintains its own data and user interactions. As the number and heterogeneity of sensing devices and available compute nodes for each application changes, these implementations do not promote adaptation without complete redesign. Smaller, simpler functional units that provide intermediate steps towards an overall application can alleviate scalability issues. Additionally, the state of the art [54] [69] places the burden of processing in black-box applications. This is particularly inefficient when multiple applications need

to process the same data using similar computation (e.g. both workplace automation and home security can process a user's location and occupancy from various data sources in the same way). Furthermore, reliance on application-specific code squanders the potential for designing and reusing general-purpose machine learning for multiple context-aware applications.

In this chapter, we identify a novel approach to context-aware IoT applications and design a general-purpose functional unit (context engine), which drives data processing for a given output context variable. We identify the theoretical advantages of modular applications for scalability and reduction of compute redundancy, and exploit the specification of data through ontologies and the single-output design of context engine to drive general-purpose machine learning for IoT actuation and automation.

The context engines are then composed to form equivalent applications to their monolithic counterparts, separating the generation of each output into sequential context engines, each translating heterogeneous input data into high-level usable context. By sharing intermediate context across applications instead of redundantly using input data, the size of an application's input set is limited. Simpler and more numerous functional units means applications can be distributed among compute nodes, and applications are more distributable and parallelizable. Lower complexity in each functional unit promotes scalability, especially in the embedded and low-complexity computational units available in the IoT. Incorporating general-purpose machine learning simplifies application development as well: application creation is reduced to defining the input and output context variables for each context engine, and the infrastructure bears the responsibility for managing and processing the data. (e.g. automatically transforming heterogeneous environmental monitoring sources into location-specific air quality). The improvement in scalability may come at the cost of accuracy, as fewer inputs are used per context engine stage to generate intermediate output, but also provides the possibility to train the system on the intermediate data. We automate context-aware computing by leveraging general implementations of statistical machine learning that can easily be reused for other applications. While our approach may not match the

efficiency of a customized black-box application, it provides an advantage in the dynamic IoT space for general implementations of context applications. Specifically, we address:

1. Computational complexity improvement
2. Adaptability through minimization of compute redundancy
3. A processing model for context-aware applications
4. Scalability

We then apply the context engine towards two context-aware case studies. We first identify user behavior from wearable sensor data, as well as an environment-focused context application. These are characteristic context-aware IoT applications. Assuming the availability of user context through connected devices or applications such as the first case study, we envision the residential smart grid as a distributed, scalable context-aware application. We use heterogeneous data from different residences and user activities and scale the system up with more individual compute nodes and grid elements, demonstrating the potential for complexity reduction and scalability. We then demonstrate the impact of the addition of user context, with over 14% improvement in energy flexibility prediction accuracy and 12% reduction in annual grid energy cost.

4.1 Related Work

Pervasive sensors gather raw data from a diverse combination of data sources, including sensors and user-supplied or high-level context processed from mobile and computing devices. Analog sensor data has to be at least digitized and preprocessed before software can use it as meaningful input. In the Internet of Things, most data goes through several levels of abstraction, combination, or distillation to produce a description of the environment (and its users) with discrete, semantic states. This higher level context is used for visualization (e.g. quantified self [54], vehicular safety [67]) and actuation (smart spaces [40], ubiquitous computing [67],

medicine [83], e-learning [69] and user behavior tracking [11]). In exchange for raw data precision, discretized context facilitates intuitive reasoning and reuse across applications. These current context-aware applications are individual deployments that rarely share infrastructure, code, or data natively. Practically, this end-to-end development approach results in a disorganized data space, necessitating the use of ontologies to maintain a unified, regulated data representation.

Ontologies are formal data representations that provide a structure for describing the vast amount of unregulated, diverse IoT data [86]. The Web Ontology Language (OWL) is an early Internet ontology: now a standard that provides description of web relationships with sub-domains of objects (e.g. household appliances) encapsulated by the domains they live within (houses). Several systems have been designed using OWL: smart corporate spaces [40], homes [49] and semantic services (OWL-S) [86] [96]. The Context Modeling Language (CML) [74] identifies objects with their applications via the Object-Role Model, attributing all context data to a physical or virtual entity (the object) and provides a particular form of information associated with it (role). Crucially, CML allows for dynamically changing input data by partitioning the input space into context dimensions: subsets that can be used for application reasoning.

Finally, pervasive sensing and computing in the IoT is facilitated by learning and reasoning within applications to appropriately transform input data into output context and actuation. For example, when streaming data from human subjects, sliding windows of the continuous data must be smoothed and preprocessed before inputting into an analytic or modeling framework. K-means clustering is a prevalent way to automatically relate low-level data into high-level context [54]. Reinforcement learning (RL) invites users who are already involved in sensing and actuation to reinforce and guide the system towards better accuracy and intuitive actuation. Madhu et al. [69] use constraint reasoning to describe a daily plan and RL to find optimal customized reminders for a cognitively or orthotically impaired user. Rashidi et al. [81] perform unsupervised learning over low-level sensor event sequences to extract patterns that represent high-level activities. They focus on a specific implementation for the smart home over a known set of activities, but

we propose a framework and algorithms that can perform a similar level of data translation and actuation in a domain-independent manner.

4.1.1 Takeaways

From the related work, the goal of the IoT application layer is that it provides an interface between sensing and actuation in the IoT. A key takeaway is that applications operate in a dynamic space: mobile sensing devices (e.g. wearables) and compute units (e.g. mobile phones) enter and leave the domain of a particular application [32]. Some leverage ontologies to provide platform independent organization of applications: black boxes that transform input data into output data for a specific application [49]. Perera et al. [76] reinforce this view, providing a comprehensive overview of context-aware applications covering fifty publications over the last decade. They view applications as multi-input, multi-output (MIMO) computation units composing similar data transformations to obtain output context information.

We identify three major interrelated challenges with the current view:

1. There is significant processing redundancy: different applications using the same input may repeat the same computation. For example, user occupancy data serves both home security and grid automation applications, and may be independently computed by both. *We choose to instead expose the output of this computation for reuse.*

2. The complexity of IoT applications grows rapidly with input and output spaces. This in turn increases the computational cost of machine learning (ML) algorithms, whose complexity is dominated by the number of inputs. This in turn forces application-specific implementations that cannot be reused. *By reducing the number of inputs per functional unit and enforcing a single-output approach, our approach facilitates the use of ML.*

3. Without effective reuse of data and functionality, the scalability of IoT applications is severely limited. Large application functional units (see Figure 4.2 (left)) preclude a general approach to distributed computation, modularity,

and reduction of complexity. *Our approach focuses on modularity, which in turn creates applications that can be readily distributed or parallelized.*

4.2 Context Engine Design

We first design and implement an alternate view of IoT applications: a hierarchy of multiple-input-single-output (MISO) functional units called context engines to improve reasoning and scalability while reducing the data redundancy across applications, and accomplishing the same functionality as the previous monolithic multi-input multi-output (MIMO) units. In exposing intermediate data, we reduce the complexity and improve the scalability of other applications in the larger infrastructure. The improvement in scalability may come at the cost of accuracy, but we both quantify the additional error and illustrate how it can be minimized. We exploit the unique opportunity in IoT where reasoning and data is often replicated between different applications. Modularization generates intermediate context that can be shared among applications (see 4.2.(a). Furthermore, as the smaller, hierarchical functional units represent a simpler data translation compared to the overall computation of an application, we can implement a general machine learning algorithm to perform data transformation - from the input context to the output - and reduce application-specific code.

IoT applications consume data about both physical and virtual system entities. This data, from heterogeneous sources including sensors, social media, and even manually submitted by users is raw and noisy requires processing by applications to be filtered and distilled into usable information. Additionally, from the input data, applications need to extract context: high-level abstracted data. In the IoT, context tends to be human-centric classifications (e.g. location, activity) that are important to many different applications [76]. Black-box implementations of applications from raw data to output mask both types of processing output (preprocessing and common intermediate context) from other applications, which leads to redundancy in computation. Our proposal of a hierarchy of functional units in place of monolithic implementations trades off compactness for versatility. A hier-

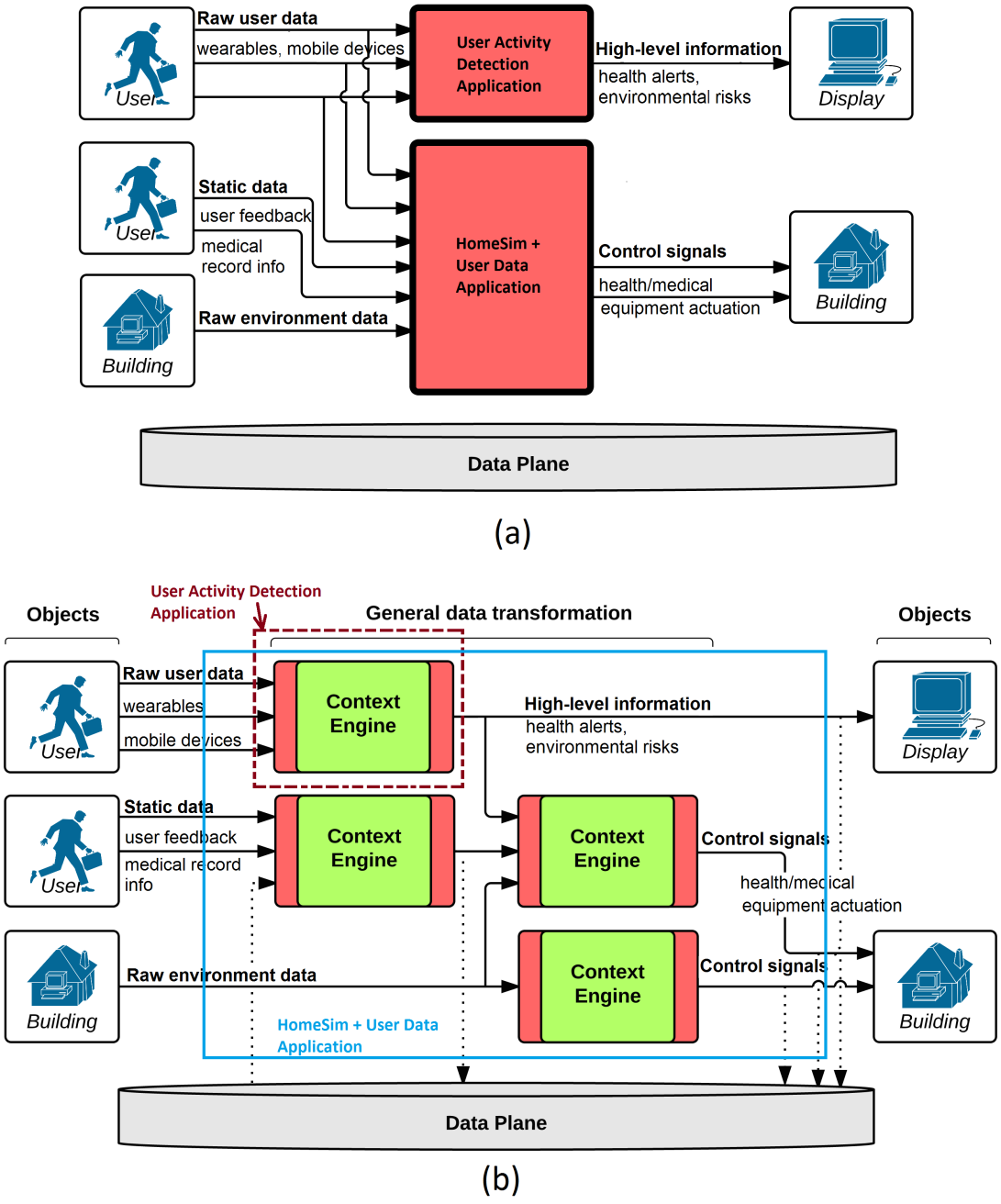

Figure 4.2: (a) The current state-of-the-art: monolithic end-to-end application implementation. (b) Our implementation: Applications publish intermediate context for reuse. Functional units (context engines) are multi-in-single-output, and each context engine performs a general statistical learning. For the above figures: red represents developer effort; green represents generalized data transformation provided by the context engine system.

archical approach breaks down a single application into multiple functional units, increasing organizational complexity. Although serializing the process can increase latency if a highly compact algorithm was expanded, it can also expose intermediate output for reuse by other applications, thus reducing compute redundancy in the system. We will prove that it also decreases overall compute complexity and enables system scalability, in terms of reduced input processing and reduced functional order when certain conditions are met . Additionally, splitting single-step applications into small functional units (each with fewer inputs, simpler logic) facilitates a generalized data transformation through machine learning.

4.2.1 Context Engine Architecture

We leverage ontologies to specify the interfaces to each of the context engines and, uniquely, to also drive data transformation. We define a context variable as the individual input or output data unit, and leverage the variable's space representation of ontologies to define the domain/range of the variable, and subsequently, the context engine:

- **Discrete context variables** must have a countable set of possible states. This is a requirement for the data processing algorithms to be able to map to input or output states

- **Continuous context variables** require a range of possible values, to allow the algorithms to perform clustering over the range of the variable. Different ontologies make allowances for multi-interval and other continuous representations.

While current monolithic applications may have internal modularity and parallelism, they are hidden from the rest of the system. The MIMO implementation of a current IoT application can be explicitly broken down into several context engines, which decompose its internal functionality into smaller, MISO functional units. The composition of the context engines delivers the same outputs as the original application (see 4.2(b)). This reduces the complexity of each context engine, as each performs less processing than the single-stage engine, and improves

the scalability, as each requires fewer inputs and produces fewer outputs. For IoT applications, this increases the overall versatility, as the now-visible functional units and the resulting dependency graph can be deterministically or automatically distributed and parallelized among available compute nodes by the IoT management system. Finally, the intermediate outputs of the distributed approach provide additional context that can be reused by other applications, reducing their computational complexity and consequently the compute redundancy of the system at large.

4.2.2 Generalized Data Transformation

Our approach, a modular multi-stage context engine, results in more functional units (FU) per application. An important consequence is that each FU that composes the application is a simpler translation of input data to a single output. This enables the use of a general data transformation in each context engine in place of application-specific code. Thus, a context-aware application can be created by specifying the inputs and output of each FU alone, and allowing the data transformation algorithm to incur the processing overhead generating and training a model based on input and output observations.

We leverage the ontologies that are already present in the current state of the art of IoT middleware. From a data standpoint, they regulate inputs and outputs of applications. Applications that participate in the system must enforce the ontology's specification: discrete variables must provide a set of possible states to populate the probabilistic condition tables; continuous variables must specify a valid range of values that can be clustered. We can exploit this ontological information for machine learning algorithms that clusters results based on the space of the input and output variables, as well as determines the training space and list of prior observations.

Matrix-based stochastic learning models express potential data dependencies as a system of equations. Some use predefined notions about the inputs to establish linear or nonlinear equations, while others start with a linear combination of the inputs whose coefficients are unknown. Over time, observed input and out-

put data is gathered until the coefficients can be trained and a model generated. Since complex relationships can exist among the input data for an IoT application, and a purely linear model may not be sufficient [30], several works [34] [50] [59] implement learning by considering higher orders and time correlation. In our current implementation, we leverage TESLA, a learning model originally designed for solar forecasting, as the data translation algorithm in our context engine. It provides efficient model generation: $O(n^{\alpha})$, where n represents the number of inputs and α represents the function order of the Taylor expansion. The generic function of this expansion is established as follows:

$$\sum_{i=0}^{n} C_i x_i \ \ (1^{st} \text{ order}), \quad \sum_{i=0}^{n}\sum_{j=0}^{i} C_{ij} x_i x_j \ \ (2^{nd} \ \text{ order}) \ \ \text{etc.} \tag{4.1}$$

where C_{ij} represents individual coefficients learned once observations are determined, and $x_0 = 1$ (the constant coefficient). The resulting equation is $\mathbf{A}x = \mathbf{B}$, where $\mathbf{A}$ is the row matrix of input observations; x is the column vector of coefficients, and $\mathbf{B}$ is the column vector of output observations, each entry correlating with the corresponding row of $\mathbf{A}$, and solved by least squares estimation.

One limitation to this model is that at least m independent observations are required for training, where $m = n$ for first-order, n^2 for second-order, and so on, which can become space-inefficient as the order increases. Finally, using the model is as simple as solving the equation using the learned coefficients and the input context, which produces the output context.

We demonstrate TESLA in this work because of its general formulation, versatility across different function orders, and applicability to context processing, but other statistical learning approaches exhibit similar properties: for example, Bayesian Networks [34], Hidden Markov Models (HMM), and Artificial Neural Networks (ANN) can leverage input and output domain spaces to conditional probability models and parameters that define preferred paths gu2013.

4.2.3 Integration with Ontologies

The context engine architecture we propose incorporates both modularity and general data transformation, significantly reducing application-specific and

implementation overhead. In addition to the overall application input and output context variables, we must identify the data flow and intermediate context required by the additional functional units. Existing context engine outputs that match the input needed by the application, that engine will be reused rather than defining and generating a new one. For example, Figure 4.3 identifies the context variable for GPS location for a particular object ("User1") using the context modeling language (CML).

```
<?xml version="1.0" ?>
<xcml:context-model xmlns:xcml="xcml">
    <xcml:context-dimension name="GPSLoc">
        <xcml:context-key name="id" value="User1"/>
        <xcml:context-key name="lat" type="float" min="-90" max="90"/>
        <xcml:context-key name="long" type="float" min="-180" max="180"/>
    </xcml:context-dimension>
</xcml:context-model>
```

Figure 4.3: Ontology specification for GPS data, with coordinates, source, and range.

If this variable specification is present, an application that require GPS location for that object can simply refer to this variable as input. If this variable is populated in the data store, either by a sensor or as the output of another application, changes to this variable will be applied. In particular, if deriving location is intermediate context from another application, it is now exposed for reuse without additional processing overhead by the current application. Intermediate context variables that are not already defined must be outlined using the ontology, specifying both the data source and the domain or range for input or output, respectively. Currently, ontological definition of inputs and outputs allows applications to retrieve and output data to the backend infrastructure. As we mentioned in previous sections, we additionally use ontologies to generate the constraints for the statistical learning algorithms. The application designer simply specifies the input and output ontologies for each context engine. The common data transformation records observations until there are enough to train the functional model for the order of computation, at which point it begins generating the output context for each successive input observation set.

4.3 Analysis

The sequential, hierarchical approach raises questions about the complexity overhead, latency, and accuracy of breaking down a possibly compact application into a composition of steps. We validate our approach by proving that the overall computational complexity of the architecture is actually reduced with a marginal impact on output accuracy.

4.3.1 Complexity

Theorem I: Dividing a context engine into multiple context engines decreases the total computational complexity of a nonlinear system.

Proof: We show that dividing the processing of N inputs from a single context engine to multiple context engines decreases the total computational complexity. We start with a general representation of a context engine: N number of inputs and a computational complexity order α for a maximum computational overhead N^α. We divide the single engine into two stages, where there are multiple engines with an arbitrary number of inputs of A. The number of engines of the first step becomes $\frac{N}{A}$. The second stage takes the outputs of the first stage and gives the final output. The total complexity overhead of this system is $\frac{N}{A}A^\alpha + \left(\frac{N}{A}\right)^\alpha$. We look for the conditions where the two-stage has a lower complexity than the single engine:

$$\begin{gathered}\frac{N}{A}A^\alpha + \left(\frac{N}{A}\right)^\alpha < N^\alpha \rightarrow A^{2\alpha-1}N + N^\alpha < A^\alpha N^\alpha \\ A^{\alpha-1}A^\alpha < N^{\alpha-1}\left(A^\alpha - 1\right) \rightarrow \left(\frac{N}{A}\right)^{\alpha-1}\left(1 - \frac{1}{A^\alpha}\right) > 1\end{gathered} \tag{4.2}$$

Although the selection of A is arbitrary, there are two limiting conditions: A must be an integer and the number of context engines must be an integer $\left(\frac{N}{A}\right)$. Thus, the minimum for A is 2 and the maximum is $\frac{N}{2}$, i.e. 2 engines. We do not consider $A = 1$ or $A = N$, as neither contribute to division of the single-stage context engine. The final inequality is the multiplication of two terms. The first term is minimized when $A = \frac{N}{2}$ and results in 2^{-1}. The second term is minimized

when $A = 2$ and results in $1 - 2^{-\alpha}$. This provides a lower bound for the result: $2^{\alpha-1}\left(1 - 2^{-\alpha}\right) = 2^{\alpha-1} - \frac{1}{2}$. If we prove that this lower bound satisfies the inequality, the multiplication result must also satisfy the inequality:

$$\left(\frac{N}{A}\right)^{\alpha-1}\left(1 - \frac{1}{A^{\alpha}}\right) > 2^{\alpha-1} - \frac{1}{2} > 1 \rightarrow a > log_2\ 3 \approx 1.6 \tag{4.3}$$

This proves that if the complexity order of the system is greater than 1.6 (e.g. for 2^{nd} and greater integer function orders), any arbitrary division of the single engine results in a decrease in computational complexity. The corollary to this theorem is:

Theorem II: The complexity of a system of context engines is minimized when each individual engine contains 2 inputs.

Proof: Theorem I shows that dividing an engine decreases complexity if the system has a complexity order greater than 1. The number of context engines is $\frac{N}{A}$, which gets its maximum value at $A = 2$.

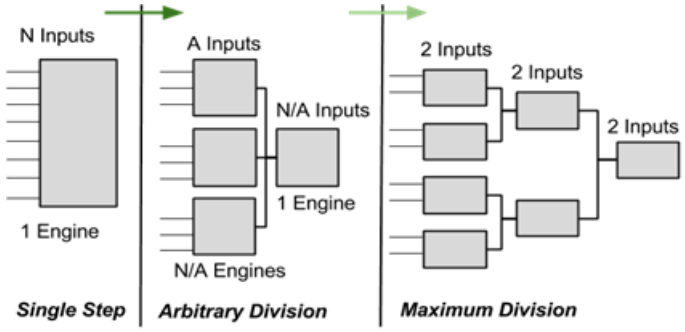

Figure 4.4: Breakdown of a single-step into lower-complexity equivalent reductions, with the minimum complexity occurring with maximum division (two-input engines on the right).

While context-aware applications do not necessarily fit perfectly into a system of two-input engines, as we reduce inputs into each context engine and increase the path from the initial input to output context, we reduce the overall system complexity.

4.3.2 Accuracy

We now investigate the accuracy change between sequential and consolidated applications. We begin with a general functional representation of data transformation from statistical learning: generating a model for data transformation as a polynomial of varying complexity and functional order, which can be solved to best-fit through techniques such as regression [30]. By providing the means to vary the inputs (ontology) and relationship (functional order), complex relationships between the inputs and output can be represented and trained. While a general formulation differs based on the function order and application, this example illustrates the accuracy change between 2-input context engines and a 4-input single-stage context engine, as in Figure 4.5.

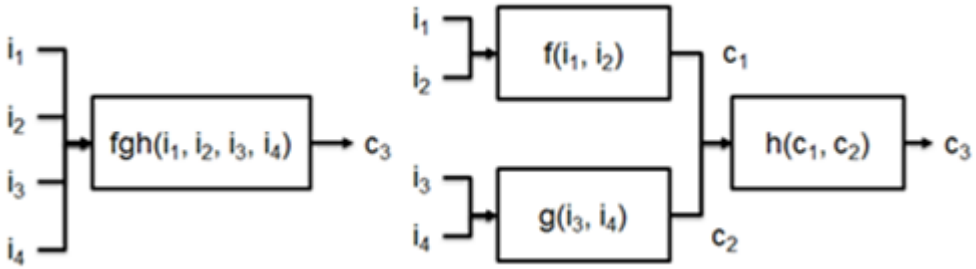

Figure 4.5: Functional comparison of sequential (left) and single-stage (right).

We can compare the two implementations through their respective transformation functions. We use a polynomial function as the general data transformation. The second-order Taylor expansion of f:

$$f(i_1, i_2) = f_0 + f_1 * i_1 + f_2 * i_2 + f_{11} * i_1^2 + f_{22} * i_2^2 + f_{12} * i_1 * i_2 \tag{4.4}$$

where f_{ij} are the corresponding coefficients. The other context engines (g, h, and fgh) have corresponding expansions. However, because h is composed of the outputs of f and g, it can be represented as a function of the f_{ij} and g_{ij} coefficients and the initial input variables i_1 to i_4. This is directly comparable to fgh, which is also a function of the initial inputs and a set of coefficients represented as λ_{ij} (e.g. $\lambda_0 + \lambda_1 * i_1 + ...$). $h(f, g)$ evaluates to fgh exactly except for the ratio of g_1

and g_2, which matches two different pairs of coefficients of fgh:

$$\frac{g_1}{g_2} = \frac{\lambda_{13}}{\lambda_{14}}, \quad \frac{g_1}{g_2} = \frac{\lambda_{23}}{\lambda_{24}} \tag{4.5}$$

This means that the sequential context engine will have the same accuracy as the single-stage only when the ratio of λ_{13} to λ_{14} matches the ratio of λ_{23} to λ_{24}. If the ratios do not match, 1 out of the 4 coefficients in this ratio cannot be represented, though the other 3 in the ratio as well as the remaining 7 coefficients in the equation are all represented accurately. This can be modeled as some error factor δ in the λ_{24} coefficient, which contributes $\delta * i_2 * i_4$ *truncation error* between the sequential and single-stage context engines.

We also quantify the impact of input signal noise on the sequential context engine compared to the single-stage approach. We model each input with zero-mean additive white Gaussian noise: $x_i + w_i$, a common expression of sensor noise [97]. The resulting noise coefficients propagate through the application. In the sequential case, f and g both propagate the original input noise to h. The truncation error of the sequential context engine is now compounded by additional noise:

$$\delta * i_2 * i_4 + \delta * w_2 * i_4 + \delta * w_4 * i_2 + \delta * w_2 * w_4 \tag{4.6}$$

The first term is the truncation error we previously quantified; the second and the third terms are the scaled Gaussian values due to noise; and the last term is a chi-squared distribution also derived from noise. The significance of the error terms is entirely dependent on the relationship between the cross-product input terms i_2 and i_4. If the output context is highly dependent on the cross products, the weight of the noise and truncation terms will pose significant error. From a system design perspective, simply selecting highly correlated input terms for context engines - an intuitive choice nonetheless - will mitigate truncation error, as the impact of the missing cross-coefficient terms is minimized.

4.3.3 Scalability

As previously mentioned, we envision IoT applications operating in an environment with dynamic computational ability. Specifically, there are different dis-

tributed compute nodes for sensing, infrastructure, and actuation, including sensor and actuation platforms, mobile devices, and backend storage and processing. The complexity arguments from the previous subsection show potential improvements even if we are confined to a single compute node. However, as the number of compute nodes grows, there is inherent scalability in the sequential context engine approach.

We leverage and extend the scalability definition from distributed systems [57] and IoT systems [89]: identifying the change in *speedup* under the conditions of 1) changing number of compute nodes for a given application (*strong scaling*) and 2) changing amount of input data (*load scaling*).

We define speedup for the context engine approach using the following piecewise function:

$$S(k,N) = kfor1 \leq k \leq N-1$$

$$\text{and} \tag{4.7}$$

$$S(k,N) = 1fork > N-1$$

where k is the number of cores (for strong scaling) and N is the number of inputs (for load scaling). Since we generalize the processing in each functional unit to the same algorithm, we deal with functional order as a general term representing the polynomial model's complexity and the number of inputs.

For *strong scaling*, the hierarchical application behaves like a distributed system, taking advantage of compute nodes as they are made available. However, even with maximum division (i.e. an application broken up into $N-1$ 2-input context engines), the speedup is ultimately capped: when more than $N-1$ compute nodes are made available, there are more free nodes than functional units. At best, some FUs can be reallocated to more capable nodes, but at this point, the system is already overprovisioned and will scale as the system is expanded.

Load scaling, or the increase in input data, is particularly important for IoT applications, as the growing amount of data in applications should be appropriately handled. An infusion of new input can be addressed by either:

1. Increasing the number of inputs to a single (or multiple different) context engines

2. Expanding the hierarchy with more low-input context engines.

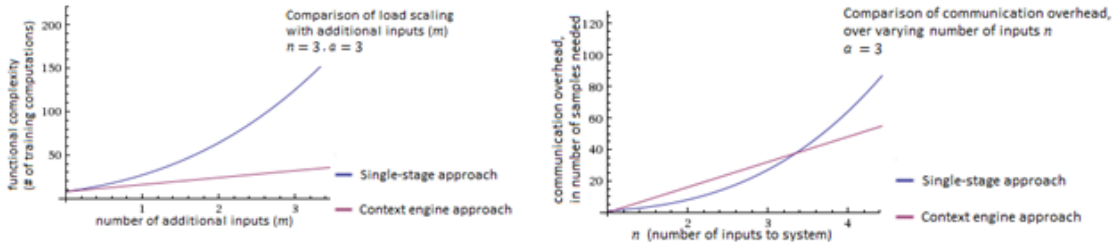

Figure 4.6: Comparison of scalability between the single-stage approach and the context engine, comparing growth in functional complexity with additional inputs (left) and communication overhead over the number of inputs (right).

In case 1, for every m additional inputs, the complexity of the updated context engine increases from n^α to $(n+m)^\alpha$. In contrast, for case 2, assuming a maximum division of input (a binary tree of two-input context engines), for every m inputs, we add at most $m-1$ new context engines, increasing the complexity to $(m+n-1)*2^\alpha$. As m increases, the complexity of case 1 grows much faster than case 2. Moreover, the second option falls in line with our goal of more modular applications. Load scaling, with a system growth of mn^α, represents linear growth in the modular context engine approach. Figure 4.6(left) illustrates this growth compared to the equivalent single-stage application's growth as m increases, with fixed n and $\alpha = 3$.

The increase in input data also affects the communication overhead of the application, another factor impacting scalability. Each functional unit must train its machine learning algorithm in order to generate appropriate output context. In the training phase of an n-input α-complexity single-stage application, the functional unit must receive n^α individual input samples and a corresponding n^α output samples from the source and sink devices to calculate the context engine's coefficients using TESLA. Similarly, a sequential application with maximum division (2-input context engines) requires $2(n-1)*2^\alpha$, or $2^{\alpha+1}*(n-1)$ input and output

samples. Figure 4.6(right) illustrates the communication overhead of the context engine approach vs. the consolidated approach over the number of inputs.

4.4 Case Study I: User Activity

For our case study, we investigate applications that leverage user activity prediction. User activity is important across several domains: connected/reactive spaces, the smart grid, social behavior understanding, etc. all provide output actuation (e.g. grid demand-response, home automation). We therefore investigate both applications that output user activity predictions and applications that use activity to provide output actuation for a particular space/domain. In particular, the latter application is used to determine the potential for a location to be used for activity or exercise.

4.4.1 Input Data

We use sensor data collected from wearable sensors from the UCSD Personal Activity and Location Measurement System (PALMS) [44], which provides high-fidelity wearable data such as from fitness trackers: hertz GPS and heart rate data, and 30Hz data from wrist and hip accelerometers for 40 individuals, with the activity annotated through observation of the individuals. The annotations fall into four categories: the activity; the posture of the participant; whether it constitutes social interaction; indoor/outdoor. The activity is chosen from a set: eating, TV, leisure, sports, exercise. Posture and indoor/outdoor are binary values associated with each activity.

4.4.2 Applications

We consider two applications: one specific to the users wearing the devices, and one specific to the spaces users are moving in. The user-centric application is activity prediction: translating raw sensor data into high-level activity definitions. The location-centric application is health potential the potential for a particular

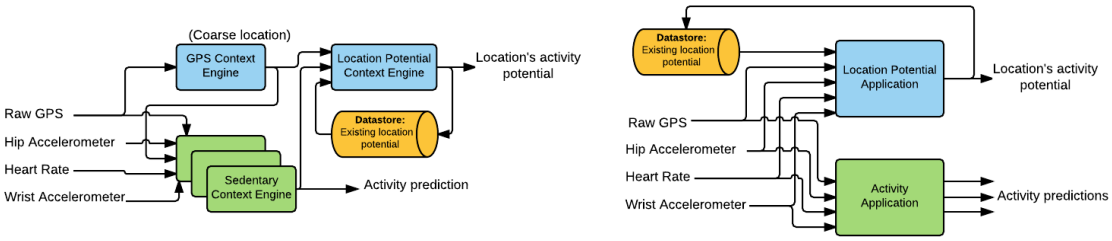

Figure 4.7: Sequential context engine applications (left) and equivalent consolidated applications (right) for user activity and location potential detection.

location to raise the user's heart rate. While the former application's output is specific to the user, the latter can help guide users towards better behavior: identify the possibility of taking stairs or improving their daily energy expenditure. This forms a consolidated IoT environment users whose sensor data impacts the spaces around them, and conversely, smart spaces that provide actuation to the users that enter them.

4.4.3 Data Translation and Outputs

The state of the art data translation is straightforward: separate applications take in all the available data as input, and produce the activity potential and activity prediction, respectively (see Figure 4.7(right)). Our approach modularizes the problem into three sets of context engines: generating coarse GPS location, detecting activities, and identifying each location's activity potential. In keeping with the MISO principle, we allocate a separate context engine for each activity. We also have separate activity detection engines for each of the 40 users, leveraging the fact that the users' fitness trackers are embedded devices that can detect personalized activity.

As GPS information is important, but the raw GPS data is too fine-grained for either application, we introduce a GPS context engine, which outputs a coarser latitude/longitude reading showing a larger physical space. This intermediate variable is defined as a latitude/longitude pair, albeit with less fine-granularity. The reference data to train this context engine is derived by spatially clustering the raw GPS data and using the northwest point of each cluster.

The output prediction is a boolean for each potential activity. Similarly, a location's activity potential is a boolean. Both values are trained using available annotated data, or ground truth, from PALMS. The location's activity potential is stored in the datastore (a key-value cloud database) with the location as the lookup key, and is fed back into the context engine as another input context.

4.4.4 Context Engine Setup

Figure 4.7 illustrates the configurations for both the sequential context engines (left) and the current state of the art single-stage application (right). The single-stage applications, as monolithic black-boxes, require all the input data. The sequential approach can be designed more judiciously. The GPS context engine's output supplants the raw GPS latitude/longitude data in both applications, though each activity's context engine requires the speed and satellite count from the raw GPS data. The GPS context engine, with one input, has $O(1)$ complexity.

The second stage of the sequential activity application transforms the available input from the original data sources and the GPS engine to a boolean representation of the respective activity. Each of these second-stage context engines have n inputs and a computational complexity of $O(n^{\alpha})$ for generating output, where a corresponds to the algorithm's function order. The third stage is the entire location potential application, as it only uses intermediate data produced by the other context engines (including feedback from itself). As activity detection context engines can grow in number as more activities are added, the computational complexity is $O(kn^{\alpha})$, where k is the number of detectable activities.

The single-stage applications (Figure 4.7 (right)) are used to compare complexity and accuracy against our approach. They are also run using a general data transform defined by a polynomial function order. Since both applications take in all the available inputs, their complexity is similar to the second-stage context engines: $O(n^{\alpha})$. As each context engine uses different inputs and generates different outputs, we test each with TESLA up to 3rd-order functions, after which accuracy improvements are marginal. From the PALMS dataset, we extract contiguous time-series data, interpolating each as necessary to provide correlated training and

test samples. To test the impact of the number of samples on functional order, we vary the number of samples up to 8,000 (two days), and test against 4,000 (one day) samples.

4.4.5 Results

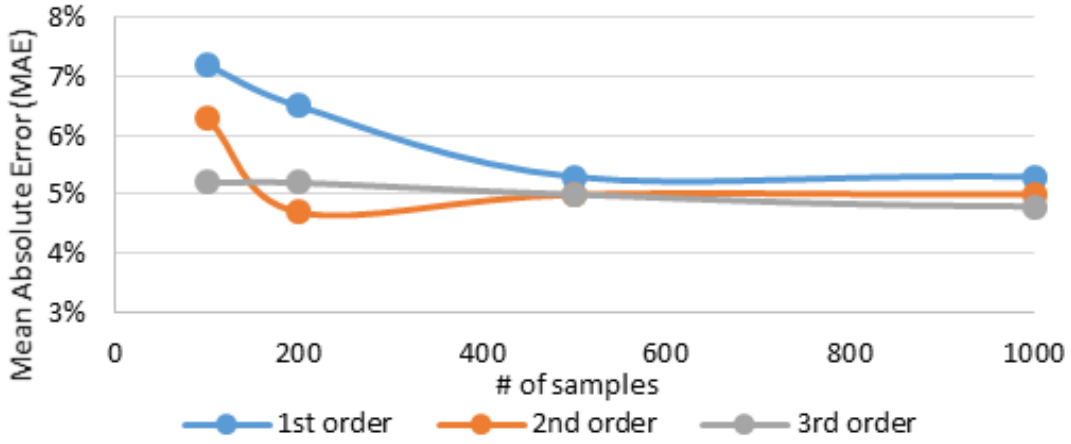

Figure 4.8: MAE of GPS context engine over function order and sample size

Complexity

We first investigate the computational complexity of each context engine group: The GPS context engine is the first stage for both sequential applications, as its output is consumed by following stages. The single engine was tested across different functional orders of the TESLA algorithm, where it performs well with first-order, gaining only marginal improvements for second- and third-order functions (Figure 4.8), for a complexity of O(1).

The activity context engines are used by both applications. Since some activities are more readily predictable than others, their functional orders vary. Table 4.1 lists the functional orders and accuracies for the different activities, and Figure 4.9 illustrates the change in the mean absolute error with each change in functional order. Some of the statistical learning results were inaccurate even when taken up to 3rd order. For example, *eating* shows marked similarities to *walking*,

Table 4.1: Functional orders used for each activity context engine.

Activity	Functional Order
Sedentary	2^{nd}
Walking	3^{rd}
Biking	2^{nd}
Exercise	2^{nd}
In Vehicle	2^{nd}
TV	3^{rd}
Eating	3^{rd}
Indoor/Outdoor	1^{st}

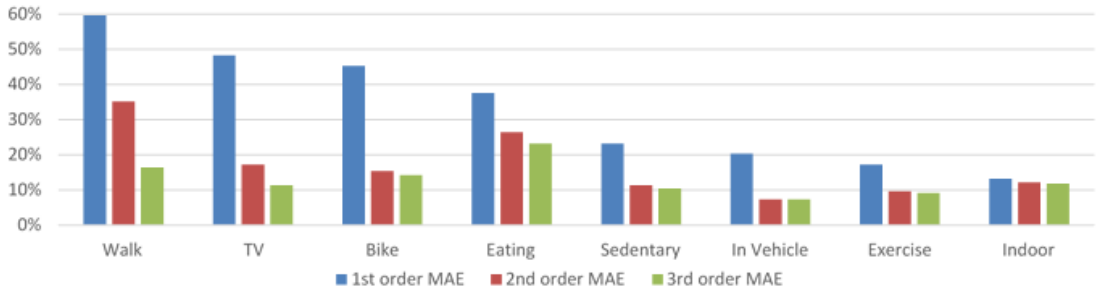

Figure 4.9: Mean absolute error (MAE) for each activity context engine across different function orders.

with the exception of speed, and as such, reports false positives even at 3^{rd} order. *Walking* incurs accuracy issues when indoors, due to reduced GPS reliability. This is compounded by the treatment of each set of correlated inputs as an individual instance, whereas *walking* on its own is better represented by a contiguous-time set of inputs represented together. TESLA is capable of treating the data as such by increasing the number of inputs to include those from time $t-1$ to $t-k$, and identifying the k value that provides the highest likelihood of being the boundary of an event. The algorithmic changes to determine the length of an event are outside the scope of this work, but the methodology can be found in [30].

In contrast, other activities can be determined with lower function order: most of the other activities are reasonably accurately predicted with 2^{nd} order polynomials, and determining that the user is *indoors* can be determined with first order alone. The trained TESLA coefficients show a high correlation between the number of satellites detected by the GPS signal and the error resulting from

poor coverage in outdoor areas. The worst-case complexity for the set of activity detection context engines is $O(kn^3)$.

The final stage of the sequential context engine approach is the location potential context engine, which consumes outputs generated by the GPS context engine and the activity context engines, to generate the potential for each location cluster. It incorporates the activity potential for the nearest known location found in the datastore (see Figure 4.7). This last input dominates the results, and a second order function, $O(n^2)$, is sufficient to generate accurate output (9.2% MAE).

Each application in the single-stage approach has exactly the same configuration: Using several TESLA algorithms on the same inputs to produce different outputs simulates general-purpose MIMO learning, with a time complexity of $O(kn^3)$. Even with the same worst-case complexity, sequential context engine approach presents an advantage in execution overhead. All context engines recalculate for the output whenever a new input observation is recorded. However, the second stage of the sequential context engine reacts only to changes in coarse location (output from the GPS context engine), which is much less frequent than the correspondingly subtle changes to the raw GPS data (Figure 4.7(left)). Table 4.2 highlights the number of computations performed by the single-stage and sequential context engines for the 4,000-input (one day) test set.

Table 4.2: Execution overhead based on iteration count for the context engines associated with the In-Vehicle activity

In-Vehicle	**Functional Order**		**Total Latency**
Prediction	1^{st} **stage** $O(1)$	2^{nd} **Stage** $O(n^2)$	
Sequential	3670	1823	0.24 sec
Single-Stage	—	3670	0.37 sec

The sequential application is comprised of more context engines, which all perform their own data transformations, and naturally performs more total computations than the single-stage approach. Still, it exhibits only 65% of the latency of the single-stage context engine. This is in part because of distributed computing: the first stage consists of data-independent context engines that can be parallelized, such as in low-level compute nodes that are already embedded in modern appliances. More importantly, because of the modularized functional

units, the sequential context engine offloads the processing of raw GPS to the low-overhead, very fast ($O(1)$) GPS context engine. The single-stage context engine, has no choice but to perform over 3000 $O(n^3)$ computations. Ultimately, the sequential application requires fewer than 50% of the time- and compute-intensive $O(n^3)$ computations as the single-stage implementation, and consequently, can complete the work 35% faster. This difference in latency exists across carries over to all other activity context engines, with a speedup of up to 2.6x for the Indoor activity engine, which has only a 1^st order function. Table 4.3 shows the latency of the context-engine-based applications normalized against their single-stage counterpart.

Table 4.3: Latency of the sequential applications grouped by the function order

Function Order (Context Engine Applications)	**Normalized Latency**
1st (Indoor)	0.38
2nd (Sedentary, Bike, Exercise, InVehicle)	0.49
3rd (Walk, TV, Eating)	0.65

Accuracy

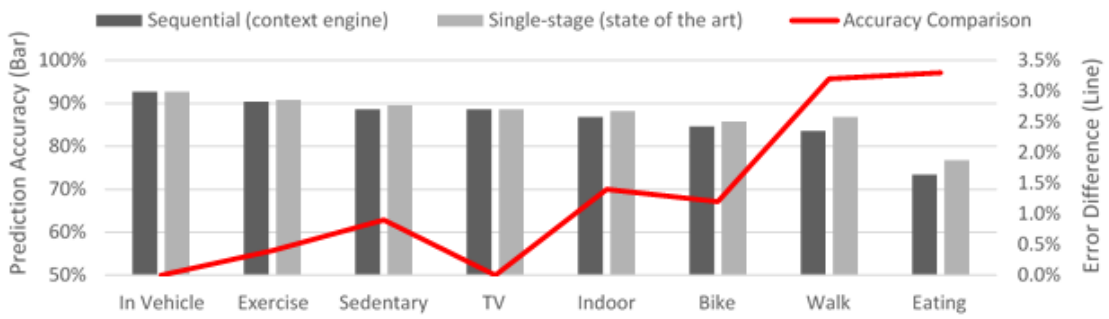

Figure 4.10: Accuracy comparison between the context engine and the single-stage activity applications (bar graph) with the delta in accuracy (line graph).

We demonstrated earlier that modularizing an application incurs truncation error. We now compare the accuracy changes in both applications between the sequential and single-stage counterparts. Figure 4.10 compares the accuracy of each of the individual activity context engines against the consolidated activity application. The application sets have very similar accuracy numbers, with the

highest error difference between them being 3.3% (for *eating*). This is explained by the high similarity in their input sets, with the only difference being the use of the intermediate output of the GPS context engine. Since only a small number of the original cross-coefficients in the statistical learning algorithm are missing, the resulting difference in error is low. In contrast, the consolidated location potential application is significantly different from the equivalent sequential context engine (see Figure 4.7(left) vs. (right)). This is an ideal example for comparison, because instead of using the raw input data, the context engine relies on only the *intermediate* data the outputs of the activity and GPS context engines, a key motivation for the modular context engine approach. Table 4.4 quantifies the accuracy comparison between the sequential and single-stage applications.

Table 4.4: Output accuracy comparison for Location Potential between the sequential context engine and the single-stage application

Application	**Location's Activity Potential - Prediction Accuracy**
Sequential	79.9%
Single Stage	74.2%

Despite using only intermediate data, which means that there are cross-coefficients missing in the sequential approach vs. the consolidated application, there is only a 5.7% reduction in output prediction accuracy. This reduction illustrates one of the conclusions earlier presented: organizing the inputs and outputs of the modular application appropriately reduces truncation error. Intuitively, the location's activity potential is related to the activities that are performed in that location, and as such, using the existing intermediate output provides reasonable accuracy compared to reusing (and performing similar computation) on the original data.

Application Extension

The location potential application shows how modularity can boost an IoT system with new applications. In the single-stage approach, an entirely new $O(n^3)$ application with $n = 5$ needed to be introduced to produce the same output, albeit slightly more accurately, as a $O(n^2)$ application with $n = 3$ using already-generated

output in the sequential approach. Growing an ecosystem of applications that make use of intermediate output from the existing applications reduces the computation needed for new applications. Furthermore, the smaller, more numerous context engines can be distributed more readily among distributed compute nodes in the IoT environment.

4.5 Case Study II: Context-Aware Residential Energy Management

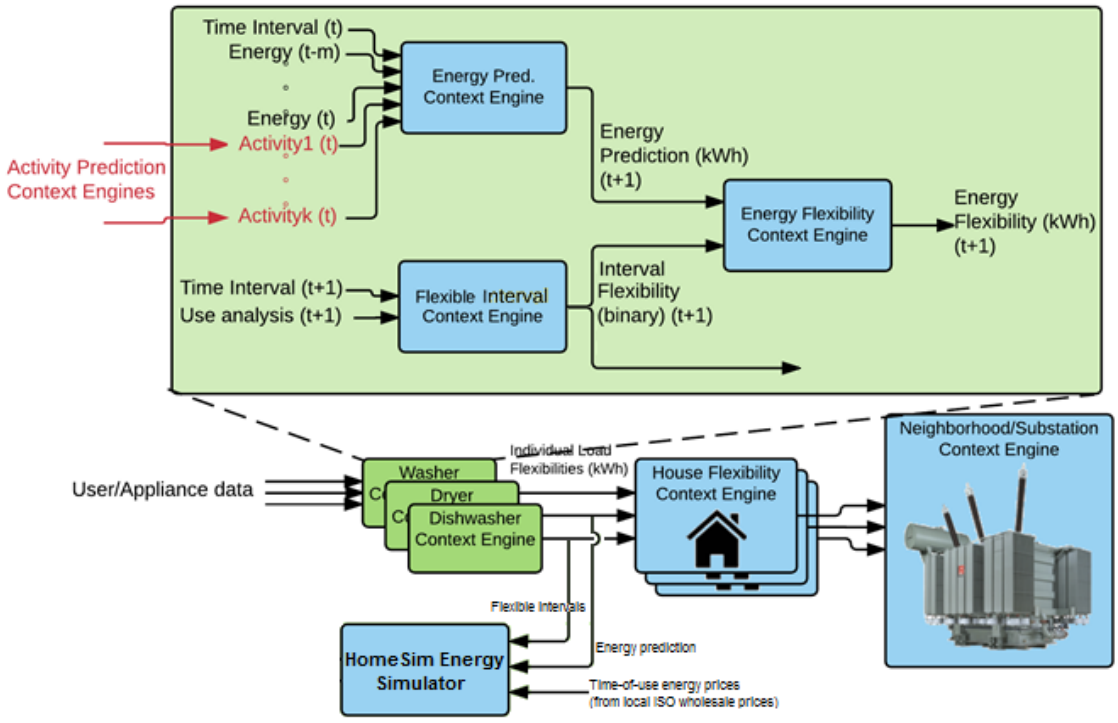

Figure 4.11: A context engine approach to residential energy management, with individual homes providing higher-level context in place of raw data, aggregated and passed . The outputs per house can vary depending on the types of sensors and actuators available to each unit.

For our second case study, we introduce the use of the context engine to scalably automate residences by predicting the energy flexibility of end-use elements in the residential smart grid. We then use this flexibility information and

usage prediction to reschedule appliances to save energy costs. Currently, utilities gather energy consumption from end-users through smart metering single-stage data processing system. User behavior can be used to further improve the accuracy of energy prediction [26]. This additional context, obtained from wearable and house sensors, vary in source, data, accuracy, format, etc. among the different users. In the current smart grid system, all this heterogeneous, additional data would need to be passed in directly to the utility, which in turn would use a redesigned application to provide energy prediction. This represents a significant increase in both communication and processing overhead. However, the context engine approach can be used to provide only the high-level context that the utility requires: energy prediction and the flexibility of the next interval (potential energy savings in kWh by shutting down loads). Furthermore, as the smart grid is naturally distributed, we can further break down data aggregation along the existing lines of power distribution: waystations, junction boxes, and substations, which have limited computation ability already (see Figure 4.11). The result is a multi-tier context-aware application that uses available residential data to determine the flexibility of the loads of a house, and further uses this generated context to determine the energy flexibility of a group of houses, a neighborhood, and ultimately, the residential sector. We demonstrate the feasibility of this approach by comparing it to the current state-of-the-art: sending all the raw data back to the utility for processing. We then investigate scalability by comparing 1) the speedup with the increasing problem size representing a neighborhood that grows over time, both in the number of sensors and houses and 2) the amount of data/communication overhead with increasing problem size. Finally, we connect our predictions back to HomeSim, using the *Flexible Interval* context engine to provide reschedulable timeframes for different appliances and the *Energy Prediction* context engine to generate individual appliance traces. We then simulate the houses to quantify the cost savings of appliance flexibility, taking into account more realistic, personalized deadlines and exploiting variable time-of-use pricing.

4.5.1 Context Engine Setup

In our approach, we first begin at the level of each individual end-use appliance within a house. Some appliances are less flexible (e.g. HVAC systems, refrigerators, and always-on devices) than others whose energy is dominated by direct user interaction (kitchen and laundry appliances, lighting, etc.). We exploit the advent of smart appliances with onboard embedded systems as potential nodes of computation. The goal in this first stage is to identify 1) user interaction with the appliance, if applicable, and 2) whether the usage of this appliance is flexible at a given time. These intermediate outputs are further used to predict the energy usage of the appliance in the next interval, and consequently, the predicted energy flexibility. The intermediate and final outputs are trained with ground truth as following:

- User interaction and activities are boolean values derived from analyzing the energy and/or water traces to find how appliances are used.
- Binary energy flexibility for appliances is derived from the distribution of use over time (see Figure 4.12. This is unique to each house due to differences in user behavior.

These first-stage context engines' outputs are further used to predict the usage of the appliance. While the energy usage alone has been previously used in time-series to predict future intervals' output, we can better learn the profiles of user-triggered appliances by also leveraging user context. An individual house can aggregate its flexibility prediction, passing it up to the next tier: junction boxes or substations, which in turn feed aggregated flexibility prediction to the overarching utility (see Figure 4(left)). While aggregated flexibility is useful for identifying the energy can be saved, our approach takes the next step and determines the individual loads that combine to provide this flexibility. This granularity is an innovation enabling the smart grid perform automated residential demand-response: feedback control signals to automate individual loads in a scalable manner.

4.5.2 Input/Intermediate Data

Our data is sourced from the Pecan Street database [5], a dataset that aggregates individual energy and water loads. In addition, a subset of houses provides basic information about the number and type of occupants. We selected and replicated houses that fall into one of the house types in Table 4.5 to represent a neighborhood with disparate amounts and types of data. The first-stage context

Table 4.5: The four different house types retrieved for the case study, with their constituent components.

House Type	Flex. Appl.	Inflex. Appl.	Add'l room-spec. appl.	Electric Vehicle	Water appl.	Water flow
A	X	X			X	
B	X	X	X		X	
C	X	X			X	X
D	X	X	X	X	X	X

engines need to be trained with ground truth for user interaction and binary flexibility of each interval. As Pecan Street does not provide this information directly, we define flexibility based on historical data about the appliances - each house show different usage patterns for each appliance, with each cluster having a range of start times. For each new appliance event, we assume its flexibility to meet that of the historical operation of that appliance instance. For example, Figure 4.12 illustrates the usage pattern of washing machines for House B, highlighting the aggregate number of instances at each time interval. The resulting clusters identify the windows of flexibility. A future appliance interval occurring at noon will have a flexibility range between 9:30 AM and 1:00 PM. Similarly, we record clusters for all appliances in all tested houses, generating unique, heterogeneous flexibility ranges that we use to represent different user preferences. Similarly, we associate flexibility to other appliances based on related research and the traces themselves. For example, the electric vehicle demonstrates three states: not plugged in, plugged in but not charging (nominal drain from charging circuit), and charging. The second and third states represent a flexible timeframe for flexible use. Table 4.6 highlights the other flexible appliances. Finally, to calculate grid energy cost, we use time-of-use pricing obtained from independent system operators across the United

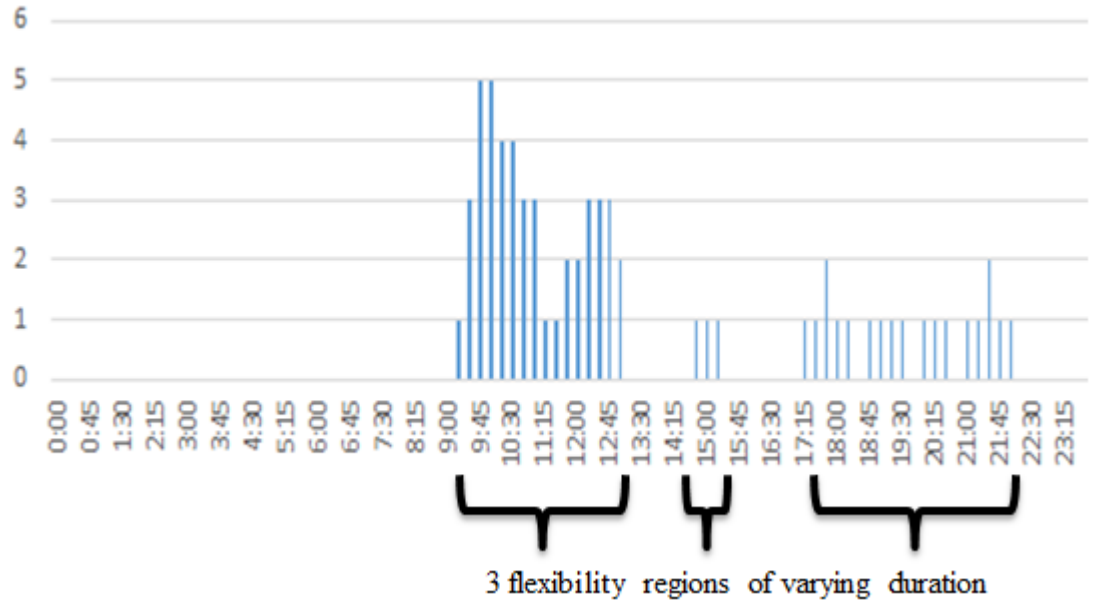

Figure 4.12: The aggregated instances of washing machine usage on Mondays in House B, illustrating 3 clusters of varying flexibility.

Table 4.6: Appliance flexibility parameters

Flexible Appliance	Justification
Clothes Washer/Dryer	Flexible usage patterns [33]
Dishwasher	Flexible usage patterns [56]
Electric Vehicle	Observed flexible charging in Pecan Street dataset
Lighting	Variation in light intensity [92]

States: CAISO for California [3], NEISO for New England [13], and ERCOT for Texas [6]. This allowed us to see the benefits of our prediction across different pricing schemes: Boston (high mean price, high standard deviation), San Diego (medium mean price, medium standard deviation), and Houston (low mean price, low standard deviation). Figure 4.13 illustrates the ISO wholesale prices scaled up to average retail levels, illustrating the differences among them.

4.5.3 Accuracy/Complexity

We first investigate the accuracy, comparing the sequential context engine approach to the single-stage state-of-the-art: a single node representing an aggre-

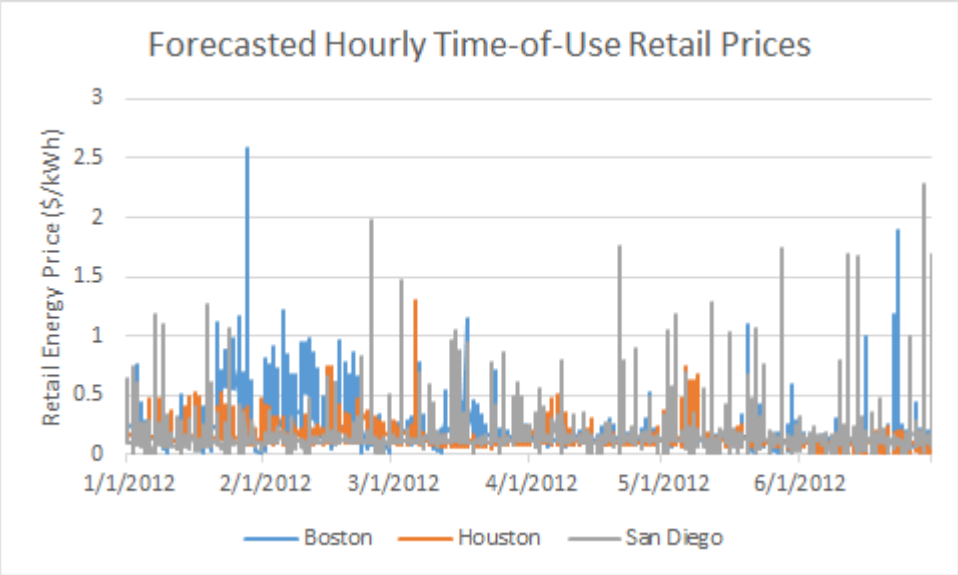

Figure 4.13: Wholesale electricity prices scaled up to retail residential pricing for different locations.

gator receiving all the raw traces from all houses and training over the aggregate flexibility. Table 4.7 highlights the mean absolute error (MAE) for both the context engines utilized by the two approaches. The ground truth that we used to calculate accuracy was the true appliance energy consumption from the Pecan Street traces. As the table shows, each of the sequential context engines within each house, pro-

Table 4.7: Average mean absolute error (MAE) for each context engine in single-stage and sequential approaches

Node Type (complexity)	**Single-stage MAE (%)**	**Avg. Context Engine MAE (%)**
A (3rd order)	—	27.15
B (3rd order)	—	14.23
C(3rd order)	—	9.81
D (3rd order)	—	6.16
Single-stage (3rd order)	26.94	
Context Engine Aggregator (1st order)		14.34

viding per-appliance energy flexibility, performed with less than 10% error for each appliance, and improving error as more user data is provided (# inputs for A < B < C < D). Conversely, the single-stage application, suffers worse error due to the

lack of training over each appliance's flexibility and user interaction. The single-stage engine requires the more complex 3rd-order computation to be performed by the aggregator, which scales poorly with more inputs. The complex-third order processing in our approach is handled closer to the edge by the embedded devices on the appliance-specific context-engines with fewer inputs and lower overall complexity. Thus, our approach at 1000 inputs performs 96x faster than the current state of the art, significantly reducing the output generation of a single-stage approach.

4.5.4 Scalability

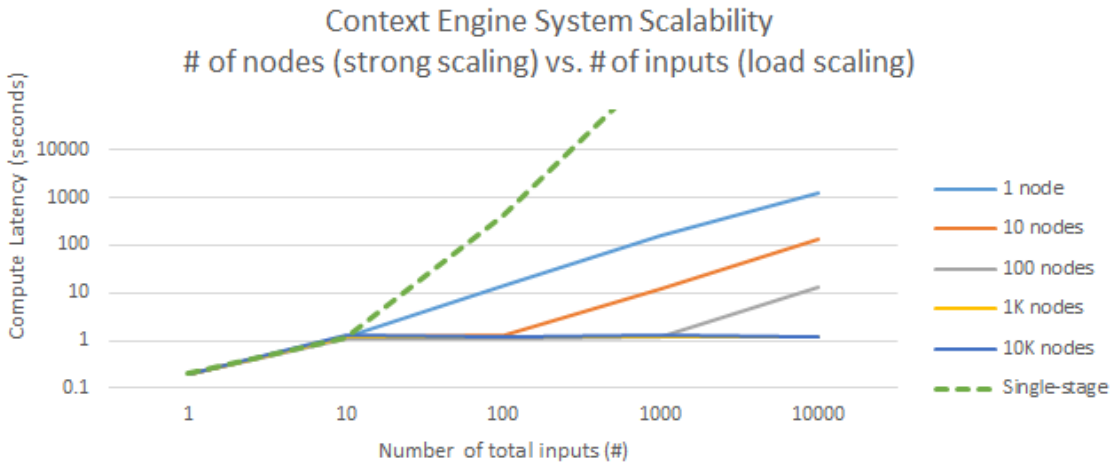

Figure 4.14: Scalability of the single-stage and sequential applications against the number of available compute nodes and the number of inputs.

We identify the latency of our application as both the number of available compute nodes and the number of inputs grows. Figure 4.14 illustrates the scalability on a log-log plot, varying the number of available compute nodes in the application and the total number of inputs. In order to investigate the theoretical upper bound, although we limit the total number of inputs, we maintain a ratio of 10 inputs per context engine, selecting only the appliance and user data that matches this ratio. Both applications initially exhibit similar latency when limited to the same number of nodes, but the context engine approach has the better potential to distribute the processing to the more numerous nodes being made

available. The sequential approach demonstrates linear growth with increased total inputs while there are enough available nodes. Eventually, there are not enough nodes, and a subset of computation must be serialized. The single-stage, without this benefit, scales exponentially with the input.

We also investigate the scalability of communication by investigating the amount of training data required for both the single-stage and the context-engine approaches. Both approaches require $O(n^{\alpha})$ training samples per functional unit. However, by limiting the n in the sequential set of context engines, and growing the number of context engines and the hierarchy instead, we are able to scale the amount of data needed for training linearly. The single-stage application, on the other hand, has a single functional unit of $O(n^3)$, and as n increases, grows exponentially with the number of inputs. Figure 4.15 illustrates the growth in communication in both cases.

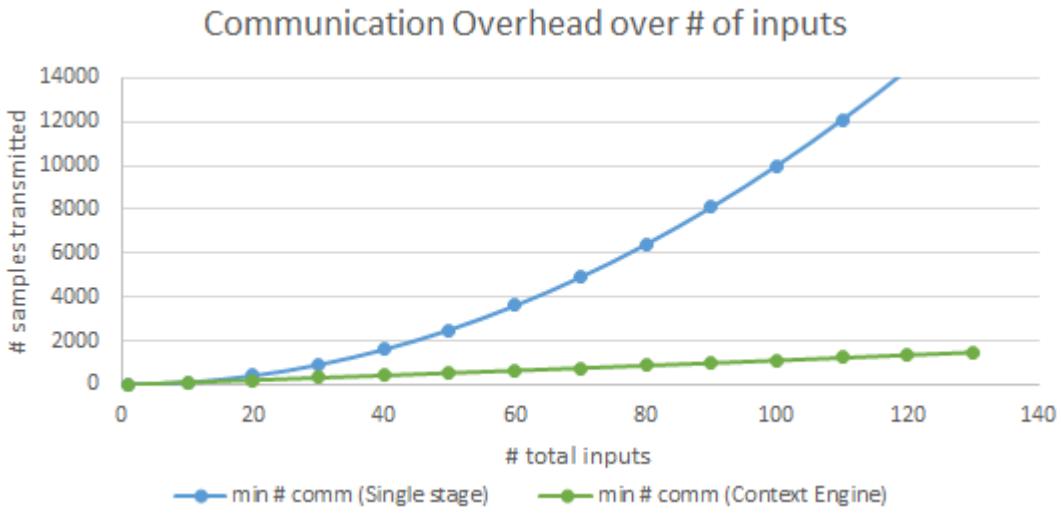

Figure 4.15: Communication overhead for training the single-stage and sequential applications as the number of number of inputs grows.

4.5.5 Grid Energy Savings

Finally, we connect our user activity and flexibility prediction back to our earlier work by integrating the context engine approach to residential grid energy management with HomeSim. In the previous chapter, we quantified the ability for

smart appliances to be rescheduled through awareness of time-of-use retail energy prices. While we previously used static flexibility information for each appliance (i.e. a fixed threshold for each appliance instance), we now have the ability to generate individual flexibility predictions using context engines (see Figure 4.11). To restate, we use historical appliance start times to generate a flexibility range for each appliance (see Table 4.6). We use HomeSim's reschedulable appliance scheduler, described in Table 3.5, which allows us to use our predicted flexibility and energy consumption to rescheduling flexible appliances. The goal is to move appliances to intervals where they can be more cheaply utilized. We use the outputs of our context engines — the predictions of energy consumption and flexibility — to generate new schedules for flexible appliances. We compare the results against having full knowledge of the appliance's consumption data and the assumed flexibility that was determined by historical usage. This is the same data that served as the ground truth in the previous set of results (Section 4.5.3).

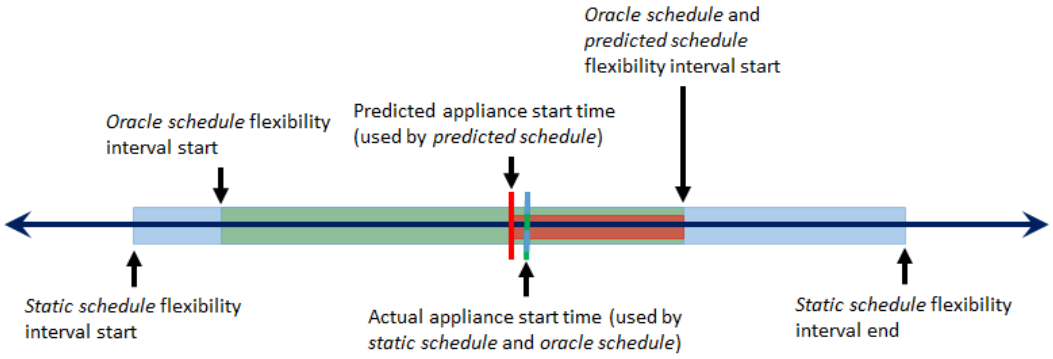

Figure 4.16: Appliance flexibility ranges for each of the test cases: *static*, *oracle*, and *predicted*.

Figure 4.16 illustrates the three cases we investigate:

- **Static schedule:** assumes a fixed schedule for each appliance. This is derived from Table 3.7. This case does not take into account differences derived from historical appliance traces.
- **Oracle schedule:** uses the ground truth derived from historical usage as the flexibility interval for each appliance. This varies from appliance to appliance

and from house to house. Since we are using the actual energy and flexibility traces, we also have the benefit of foresight: determining the full range of the flexibility interval before and after the actual appliance instance's start time.

- **Predicted schedule:** This is the real-time schedule determined solely by the output of context engine predictions. In addition to predicted flexibility, since our context engine only predicts one interval in advance, we only have the ability to use the remaining intervals of the flexibility range after the predicted appliance start time.

To generate the energy and flexibility predictions for each appliance, we obtain the output of the Flexible Interval and Energy Prediction context engines (Figure 4.11), respectively, for all subsequent intervals of the current day and pass this data into HomeSim. We assume knowledge of 24-hour time-of-use (TOU) pricing in order to facilitate rescheduling. This is data that is typically available on the wholesale sector from various ISOs [3] [13] [6]. As retail energy integrates TOU pricing as well, we expect to see similar forecasts.

Table 4.8: Residential cost savings of static flexibility vs. oracle knowledge of individual house flexibility vs. predicted flexibility

Case	**Static Schedule**	**Oracle Schedule**	**Predicted Schedule**
Base Appliance Annual Grid Cost	\$594.98	\$594.98	\$532.31
Rescheduled Appliance Annual Grid Cost	\$514.23	\$487.92	\$469.64
Cost Savings (\$)	\$107.06	\$80.75	\$62.67
Cost Savings (%)	18%	14%	12%
Appliance Deadlines Met (%)	90%	100%	100%

As Table 4.8 illustrates, we are able to use the predicted energy and flexibility values to generate a new schedule for flexible appliances. We obtain 12% annual cost savings for the rescheduled appliances using individually predicted flexibility values for each appliance. We further compare this to having oracle knowledge

of all appliances and their flexibilities — the ground truth that we used earlier to train the context engines — which generates 14% cost savings. The static schedule presented in the previous chapter (Table 3.7) demonstrates a further 4% savings, at 18% electricity cost saved.

Our predicted schedule's energy consumption is within 81% of the ground truth, and our savings prediction is only 2% less than that of the oracle. This error is due in part to energy prediction error, but also due because upon predicting an appliance's start time, we only have until the rest of the flexibility interval from the predicted start time to schedule the appliance (the *red* interval in Figure 4.16). The oracle, however, has *a priori* knowledge of the day's schedule, and can reschedule an appliance anytime within the flexible interval, including before the original start time (the *green* interval in Figure 4.16). Finally, the original static flexibility case study from Chapter 3 yields a further cost reduction primarily because of the increased range of flexible intervals (the *blue* interval in Figure 4.16). The static flexibility interval for the clothes washer is 12 hours (Table 3.7), but the flexibility interval generated by historical appliance use was shown to be 9.25 hours, with a median interval of 7.25 hours. This difference, observed over all flexible appliances, provides heterogeneity in user preferences at the expense of fewer opportunities to try to save on electricity cost. Comparing the static flexibility interval to the oracle's, we find that although it improves cost reduction by 32%, 10% of the statically rescheduled appliances actually fall outside their flexibility interval, missing users' perceived flexibility deadlines.

4.6 Conclusion

In this chapter, we establish the motivation for and design a novel approach to context-aware applications used in the Internet of Things. These applications are also of the type that will provide user information for residential grid applications. The current state of the art uses ontologies to identify the interfaces for ultimately monolithic and application-specific implementations [76]. This creates computational redundancy between applications and increases compute complex-

ity of the system as a whole. In contrast, we design a modular framework that exploits common computational processes between applications, exposing shareable intermediate context. We increase reusability and scalability while reducing computational complexity at a minor cost in accuracy. Decomposing a single-stage functional unit into a sequence of smaller ones also facilitates general data transformation. As an example, we implement a statistical learning technique that exploits ontological data to construct and train a model. We test our approach with two independent case studies, using the same approach to demonstrate the versatility of the context engine, as well as up to 65% latency reduction with minimal accuracy loss in both applications, all while exposing intermediate context for reuse. We subsequently integrate our context engines with HomeSim to demonstrate up to 9.5% energy savings by rescheduling appliances based on the engines' predictions. The result is a versatile approach and implementation for IoT applications that facilitate the use of machine learning while improving scalability and complexity of context-aware designs. With human-driven input data, this can efficiently and scalably learn and produce residential flexibility information for the smart grid.

Chapter 4 contains material from Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "A Modular Approach to Context-Aware IoT Applications", which appeared in Proceedings of the 1st Conference on Internet of Things Data and Implementation (IoTDI), 2016 [94]. The dissertation author was the primary investigator and author of this paper.

Chapter 4 contains material from Jagannathan Venkatesh, Barış Akşanli, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "Scalable IoT Application Design for Automated Learning", which was submitted for consideration in IEEE Software, Special Issue on Software Engineering for the Internet of Things, 2016. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Summary and Future Work

The smart grid is growing in complexity and versatility: new end-use elements, better analytics, and improved infrastructure. This enables the deployment of new scenarios, including distributed energy and storage resources, more complex and automated behavior of the end nodes, and automation of user-driven loads. While the goal of the improved grid is better energy efficiency and stability, different sectors have different approaches.

5.1 Thesis Summary

This thesis proposes the use of context—high-level, abstracted data—to improve the behavior of end users in different smart grid scenarios. Large consumers like data centers can integrate significant on-site renewable generation, but they must avoid missing the tight timing constraints of their workloads despite the variability in green energy output. We propose short term green energy prediction to efficiently use renewables without sacrificing quality of service. Meanwhile, small consumers such as residences are largely ignored due to their relatively insignificant individual contributions. However, the growth in complexity of residential nodes enables several new scenarios, which use additional user and environment context. These scenarios must be appropriately quantified to understand the financial and stability benefits. We introduce a new residential energy simulator to quantify and compare battery technologies, smart appliances, and trading battery capacity on

the energy market, and the cost savings of a more accurate closed-form battery state prediction. Finally, the diversity of use in the residential sector is driven by individual user preferences. However, the current smart grid infrastructure, which centrally collects and processes available meter data, does not scale to the level and diversity of users. We propose a new modular approach to context-aware applications, specifically those that use heterogeneous data, such as IoT applications. We leverage this approach for reduced computational redundancy, improved latency and scalability, and apply it to residential grid automation.

5.1.1 Renewable Energy and Context in Data Centers

Data centers are one of the important global energy consumers and carbon producers. However, their tight service level requirements prevent easy integration with highly variable renewable energy sources. Short-term green energy prediction can mitigate this variability. We first explored the existing short-term solar and wind energy prediction methods, and then leveraged prediction to allocate and migrate workloads across geographically distributed data centers to reduce brown energy consumption costs.

We demonstrated significantly higher (up to 3x) green energy efficiency despite the variability of the incoming energy, and developed a migration algorithm for performance maximization that delivered a 27% reduction in the completion time of batch workloads.

5.1.2 Modeling Residential Energy Management in the Smart Grid

Residential energy constitutes 38% of the total energy consumption in the United States. While the industrial and commercial sector has seen growth in grid-tied automation and control, the residential sector has largely been ignored because of the relative insignificance of the individual end-use loads, even as they grow in complexity and capability. Although a number of building simulators have been proposed, there are no residential electrical energy simulators capable of modeling

complex scenarios and exploring the tradeoffs in home energy management. We proposed HomeSim, a residential electrical energy simulation platform that enables investigating the impact of technologies such as renewable energy and different battery types. Additionally, HomeSim allows us to simulate and quantify different residential automation scenarios, including centralized vs. distributed in-home energy storage, intelligent appliance rescheduling, and outage management. We also develop a new formulation for battery usage based on a more realistic battery model, optimizing the benefit of discharging the battery. We design the scheme for the actual use of batteries in an energy-trading environment, considering the total cost of ownership and return on investment.

Using measured residential data, HomeSim quantifies different benefits for different technologies and scenarios, including up to 50% reduction in grid energy through a combination of distributed batteries and reschedulable appliances.

5.1.3 Improving Residential Energy Modeling with User Context

The Internet of Things envisions an infrastructure of sensing and actuation devices connected by a Web framework. This *context-aware computing* is especially useful for the residential smart grid, whose output is driven by the behavior of the constituent human users. However, the current state of the art in context-aware computing presents a different reality: complete end-to-end applications that are dependent on the sensors and actuators that they were designed for. Growing such an application with new input and output devices, or extending the existing infrastructure with new applications involves redesign and deployment. We proposed a modular approach to these context-aware applications, breaking down monolithic applications into an equivalent set of functional units, or context engines. By exploiting the characteristics of context-aware applications, context engines can reduce compute redundancy, computational complexity and scalability with a nominal impact on accuracy.

We applied these principles toward residential smart grid control, demonstrating more appropriate scalability than the current state of the art, in addition

to an accuracy improvement of 15% and a latency reduction of up to 35%.

5.2 Future Work

We have developed several tools and approaches to extend the field of grid automation, and our results can be applied to different aspects of smart grid problems and context-aware computing.

Our initial context engine results already demonstrate a significant improvement in predicting residential energy usage and flexibility (see Figure 5.1).

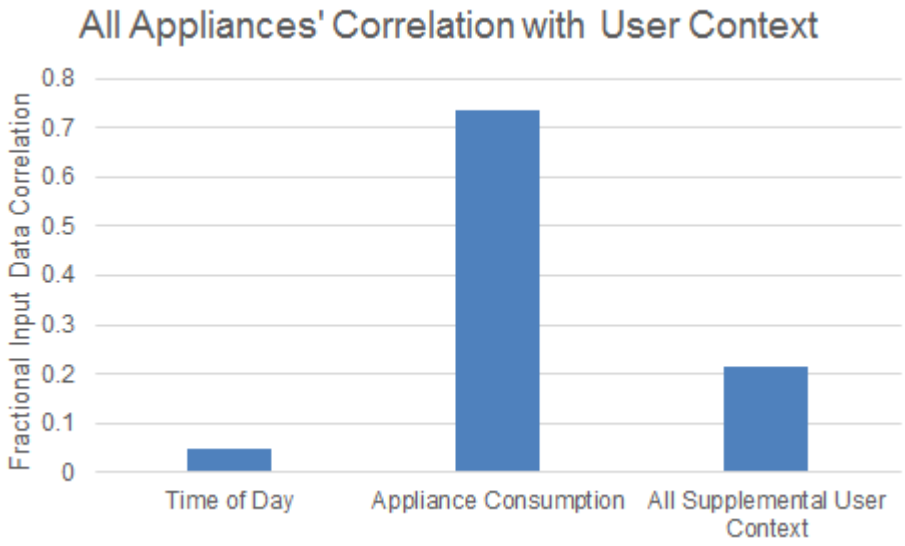

Figure 5.1: Correlation of energy prediction with time-of-day, appliance traces, and supplemental user context.

However, while we learn on all available inputs, we don't necessarily limit ourselves to only the *best* inputs. Preliminary correlation analysis has already shown the ability to mitigate accuracy losses by eliminating only the least significant inputs. Furthermore, limiting the number of inputs can improve the latency of each context engine.

Formalizing correlation analysis and automatically selecting the most appropriate inputs from a pool of available sources is useful for context-aware applications operating in an environment of changing data, as they can improve

efficiency as well as replace/update the existing algorithm as more accurate or reliable sources enter or leave the system.

Similarly, the context engine is currently employed with a number of machine learning algorithms, some of which are better suited for a particular application than the others, and all incurring their own costs, both in computational overhead and in accuracy and latency. This can be leveraged to define a multi-optimization problem for context-aware computing, trading off compute complexity, latency, and accuracy.

The ability to quantify these three characteristics, and more importantly, optimize for each is crucial to the real-life context-aware applications that these are used for: smart spaces, grid automation, and the Internet of Things. The result is a functional unit that can optimize for the most correlated data, train using the best available machine-learning technique, and generate the outputs required, as well as adapt to environment- and user-imposed performance constraints.

The context engine case study demonstrates the scalability and improved accuracy of predicting energy consumption and flexibility. Furthermore, it provides a currently unprecedented level of granularity in control of flexible appliances. However, this is only part of actually automating the residential grid. Adding distributed energy sources and storage to the system provides additional means of flexibility at different costs. Formalizing the overall cost model for each house to drive actuation based on the prediction can bring decision-making intelligence to each node. These local decision-makers and the now-available node-level flexibility information can be used in conjunction with grid information to drive actuation based on both the grid needs, user behavior, and locally-optimal node control. This can be suitably adjusted throughout the grid's hierarchy, scaling preferences and control up from a single house to a street, neighborhood, etc. up to the aggregate context of the residential sector.

Chapter 5 contains material from Jagannathan Venkatesh, Bariş Akşanli, Christine Chan, Alper Sinan Akyürek, and Tajana Šimunić Rosing, "Scalable IoT Application Design for Automated Learning", which was submitted for consider-

ation in IEEE Software, Special Issue on Software Engineering for the Internet of Things, 2016. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] Battery energy – what battery provides more? AllAboutBatteries. [Online]. Available: http://www.allaboutbatteries.com/Battery-Energy.html

[2] Borrego springs microgrid project. Green Energy Corp. [Online]. Available: http://www.greenenergycorp.com/borrego-springs-microgrid-project/

[3] California iso. California Independent System Operators. [Online]. Available: http://www.caiso.com/Pages/default.aspx

[4] California solar statistics. California Energy Commission. [Online]. Available: https://www.californiasolarstatistics.ca.gov/

[5] Dataport: A universe of energy data. Pecan Street, Inc. [Online]. Available: https://dataport.pecanstreet.org/

[6] Electricity reliability council of texas. ERCOT, Inc. [Online]. Available: http://www.ercot.com/

[7] Energy 101: Energy efficient data centers. Department of Energy. [Online]. Available: http://www.energy.gov/videos/energy-101-energy-efficient-data-centers

[8] Energy and technology. Center for Climate and Energy Solutions. [Online]. Available: http://www.c2es.org/category/topic/energy-technology

[9] Energy and technology. Center for Climate and Energy Solutions. [Online]. Available: http://www.c2es.org/category/topic/energy-technology

[10] Energy storage system - 36kwhr (3 week lead time). Balqon Corporation. [Online]. Available: http://www.balqon.com/store-2/#!/ /product/category=2860254&id=12477202

[11] Google now. Google, Inc. [Online]. Available: http://www.google.com/landing/now

[12] Heating and cooling no longer majority of energy use. U.S. Energy Information Administration. [Online]. Available: http://www.eia.gov/consumption/residential/

[13] Iso new england. ISO New England, Inc. [Online]. Available: http://www.iso-ne.com/

[14] Lead acid batteries. Samlex America, Inc. [Online]. Available: http://www.samlexsolar.com/learning-center/lead-acid-batteries.aspx

[15] Lfp batteries for customized power supply solutions. Jupiter Integrated Solutions, Inc. [Online]. Available: http://www.jisinc.net/products-battery.htm

[16] Nest: Home. Nest. [Online]. Available: https://www.nest.com/

[17] Rubis: Rice university bidding system. OW2 Consortium. [Online]. Available: http://rubis.ow2.org

[18] Savant home automation. Savant. [Online]. Available: https://www.savant.com/

[19] Simulation tool — opendss. EPRI - Electric Power Research Institute. [Online]. Available: http://smartgrid.epri.com/SimulationTool.aspx

[20] Wind power. Other World Computing. [Online]. Available: http://eshop.macsales.com/green/wind.html

[21] Wind-powered data centers. Green House Data. [Online]. Available: https://www.greenhousedata.com/green-data-centers

[22] Wind turbine power curves. Wind Power Program. [Online]. Available: http://www.wind-power-program.com/turbine_characteristics.htm

[23] M. M. Ahlers. (10) Style and comfort - it must be an energy saving house. [Online]. Available: http://www.cnn.com/2012/09/16/living/house-energy/index.html

[24] B. Akşanli, E. Pettis, and T. Šimunić Rosing, "Distributed battery control for peak power shaving in datacenters," in *Proceedings of the '13 Internation Green Computing Conference (IGCC)*, 2013.

[25] B. Akşanli, J. Venkatesh, and T. Šimunić Rosing, "Using datacenter simulation to evaluate green energy integration," *IEEE Computer 45*, 2012.

[26] B. Akşanli, J. Venkatesh, T. Šimunić Rosing, and I. Monga, *Computational Sustainability*. Springer, 2015, ch. Renewable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks.

[27] B. Akşanli, J. Venkatesh, L. Zhang, and T. Šimunić Rosing, "Utilizing green energy prediction to schedule mixed batch and service jobs in data centers," in *HotPower '11*, 2011.

[28] B. Akşanli and T. Šimunić Rosing, "Optimal battery configuration in a residential home with time-of-use pricing," in *Proceedings of IEEE SmartGridComm '13*, 2013.

[29] B. Akşanli, T. Šimunić Rosing, and I. Monga, "Benefits of green energy and proportionality in high speed wide area networks connecting data centers," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2012, pp. 175–180.

[30] B. O. Akyürek, A. S. Akyürek, J. Kleissl, and T. Šimunić Rosing, "Tesla: Taylor expanded solar analog forecasting," in *Proceedings of IEEE SmartGridComm '13*, 2014.

[31] K. Anderson. gridspice: A virtual platform for modeling, analysis, and optimization of the smart grid. [Online]. Available: http://code.google.com/p/gridspice/

[32] S. Bandyopadhyay, M. Sengupta, S. Maiti, and D. Subhajit, "A survey of middleware for internet of things," *Recent Trends in Wireless and Mobile Networks*, 2011.

[33] N. Banerjee, S. Rollins, and K. Moran, "Automating energy management in green homes," in *Proceedings of HomeNets '11*, 2011.

[34] A. Battaglini, J. Lilliestam, A. Haas, and A. Patt, "Development of supersmart grids for a more efficient utilisation of electricity from renewable sources," *Journal of Cleaner Production*, 2009.

[35] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-time battery models for system-level low-power design," in *IEEE Transactions on VLSI Systems*, 2001.

[36] G. Bredehoeft and E. Krall. (12) Increased solar and wind electricity generation in california are changing net load shapes. U.S. Energy Information Administration. [Online]. Available: http://www.eia.gov/todayinenergy/detail.cfm?id=19111

[37] N. Buchbinder, N. Jain, and I. Menache, "Online job-migration for reducing the electricity bill in the cloud," *Networking*, vol. 1, pp. 172–185, 2011.

[38] T. Carlon. Gridlab-d. [Online]. Available: www.gridlabd.org

[39] J. Carrasco, L. Franquelo, J. Bialasiewicz, E. Galvan, R. Guisado, M. Prats, J. Leon, and N. Moreno-Alfonso, "Power-electronic systems for the grid integration of renewable energy sources: A survey," *IEEE Transactions on Industrial Electronics*, 2006.

[40] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive environments," *The Knowledge Engineering Review*, vol. 18, no. 3, pp. 197–207, 2004.

[41] D. Crawley, J. Hand, M. Kummert, and B. Griffith, "Contrasting the capabilities of building energy performance simulation programs," *Building and Environment*, vol. 43, no. 4, pp. 661–673, 2008.

[42] A. de Almeida and P. Fonesca, "Residential monitoring to decrease energy use and carbon emissions in europe," REMODECE, Tech. Rep., 2015.

[43] D. Dondi, P. Zappi, and T. Šimunić Rosing, "A scheduling algorithm for consistent monitoring results with solar powered high-performance wireless embedded systems," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, 2011, pp. 259–264.

[44] K. Ellis, S. Godbole, S. Marshall, G. Lanckriet, J. Staudenmayer, and J. Kerr, "Identifying active travel behaviors in challenging environments using gps, accelerometers, and machine learning algorithms," in *Frontiers in Public Health*, 2014.

[45] K. Fehrenbacher. (July) Facebook is building a big wind-powered data center in texas. Fortune. [Online]. Available: http://fortune.com/2015/07/07/facebook-data-center-texas/

[46] M. Friedewald and O. Raabe, "Ubiquitous computing: an overview of technology impacts," *Telematics and Informatics*, vol. 28, pp. 55–65, 2011.

[47] G. Giebel, "The state-of-the-art in short-term prediction of wind power: A literature overview," Project ANEMOS, Tech. Rep., 2003.

[48] D. Gmach, J. Rolia, and C. Bash, "Capacity planning and power management to exploit sustainable energy," in *Conference on Network and Service Management (CNSM)*, 2010.

[49] T. Gu, X. Wang, H. Pung, and D. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of communication networks and distributed systems modeling and simulation conference*, 2004.

[50] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, 2013.

[51] J. Hammer and T. Yan, "Poster: A virtual sensing framework for mobile phones," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys)*, 2014.

[52] L. Hawarah, S. Ploix, and M. Jacomino, "User behavior prediction in energy consumption in housing using bayesian networks," *Artificial Intelligence and Soft Computing*, vol. 6113, pp. 372–379, 2010.

[53] C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[54] J.-H. Hong, S.-I. Yang, and S.-B. Cho, "Conamsn: A context-aware messenger using dynamic bayesian networks with wearable sensors," *Expert Systems with Applications*, vol. 37, no. 6, p. 46804686, 2010.

[55] E. Howland. What data centers' growing energy use means for utilities. [Online]. Available: http://www.utilitydive.com/news/what-data-centers-growing-energy-use-means-for-utilities/225139/

[56] T. Jamasb and M. G. Pollitt, *The Future of Electricity Demand: Customers, Citizens and Loads*. Cambridge University Press, 2011.

[57] P. Jogalekar and M. Woodside, "Evaluating the scalability of distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 6, pp. 589–603, 2000.

[58] M. R. Jongerden and B. R. Haverkort, "Which battery model to use?" *IET Software*, vol. 3, no. 6, pp. 445–457, 2009.

[59] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things," in *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, 2008.

[60] Z. J. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *SustKDD*, 2011.

[61] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D. Culler, and R. Katz, "Integrating renewable energy using data analytics systems: Challenges and opportunities," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2011.

[62] P. Kurp, "Green computing," *Communications of the ACM*, 2008.

[63] A. Kusiak, H. Zheng, and Z. Song, "Short-term prediction of wind farm power: A data mining approach," *IEEE Transactions on Energy Conversion*, vol. 24, no. 1, pp. 125–136, 2009.

[64] ——, "Wind farm power prediction: A data mining approach," *Wind Energy*, vol. 12, no. 1, pp. 275–293, 2009.

[65] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *In Proceedings of the 2010 International Green Computing Conference (IGCC)*, 2010.

[66] J. Leadbetter and L. Swan, "Battery storage system for residential electricity peak demand shaving," *Energy and Buildings*, vol. 55, pp. 685–692, 2012.

[67] S. Lee and K. C. Lee, "Context-prediction performance by a dynamic bayesian network: Emphasis on location prediction in ubiquitous decision support environment," *Expert Systems with Applications*, vol. 39, no. 5, p. 49084914, 2012.

[68] S. H. Low and K. A. Tang, "Recovery act - power minimization for networked datacenters," California Institute of Technology, Tech. Rep., 2011.

[69] S. K. Madhu, V. C. Raj, and R. M. Suresh, "An ontology-based framework for context-aware adaptive e-learning system," in *Proceedings of the International Conference on Computer Communication and Informatics (ICCI)*, 2013.

[70] C. Miller. Solar powered data centers. [Online]. Available: http://www.datacenterknowledge.com/solar-powered-data-centers/

[71] R. Miller. Solar power at data center scale. [Online]. Available: http://www.datacenterknowledge.com/archives/2009/06/16/solar-power-at-data-center-scale/

[72] ——. Sonoma project features large solar array. [Online]. Available: http://www.datacenterknowledge.com/archives/2009/07/27/sonoma-project-features-large-solar-array/

[73] A. Mishra, D. Irwin, P. Shenoy, J. Kurose, and T. Zhu, "Smartcharge: cutting the electricity bill in smart homes with energy storage," in *e-Energy*, 2012.

[74] M. Nebeling, M. Grossniklaus, S. Leone, and M. Norrie, "Xcml: providing context-aware language extensions for the specification of multi-device web applications," *World Wide Web (WWW)*, vol. 15, no. 4, pp. 447–481, 2012.

[75] T. Nguyen. (February) Electric car batteries can now power home appliances. SmartPlanet. [Online]. Available: http://www.smartplanet.com/blog/thinking-tech/electric-car-batteries-can-now-power-home-appliances/6339

[76] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications, Surveys, and Tutorials*, pp. 414–454, 2013.

[77] S. Peterson, J. Apt, and J. Whitacre, "Lithium-ion battery cell degradation resulting from realistic vehicle and vehicle-to-grid utilization," *Journal of Power Sources*, vol. 195, pp. 2385–2392, 2010.

[78] J. Piorno, C. Bergonzini, D. Atienza, and T. Šimunić Rosing, "Prediction and management in energy harvested wireless sensor nodes," in *Wireless VITAE*, 2009.

[79] C. W. Potter, A. Archambault, and K. Westrick, "Building a smarter smart grid through better renewable energy information," in *Proceedings of Power Systems Conference and Exposition*, 2009, pp. 1–5.

[80] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *Proceedings of the 2002 International symposium on Low Power Electronics and Design (ISLPED)*, 2002.

[81] P. Rashidi, D. Cook, L. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 527–539, 2011.

[82] E. Ratnam, S. Weller, and C. Kellett, "An optimization-based approach for assessing the benefits of residential battery storage in conjunction with solar pv," in *IREP Symposium*, 2013.

[83] M. Rudary, S. Singh, and M. E. Pollack, "Adaptive cognitive orthotics: combining reinforcement learning and constraint-based temporal reasoning," in *Proceedings of the 21st International conference on Machine Learning*, 2004.

[84] I. Sanchez, "Short-term prediction of wind energy production," *International Journal of Forecasting*, vol. 22, no. 43-56, 2006.

[85] K. Soon Ng, C.-S. Moo, Y.-P. Chen, and Y.-C. Hsieh, "Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries," *Applied Energy*, vol. 86, no. 9, 2009.

[86] S. Staab and R. Studer, *Handbook of Ontologies*. Springer Science and Business, 2010.

[87] L. Swan and V. I. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8, pp. 1819–1835, 2009.

[88] J. Tant, F. Geth, D. Six, P. Tant, and J. Driesen, "Multiobjective battery storage to improve pv integration in residential distribution grids," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 182–191, 2013.

[89] D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer Berlin Heidelberg, 2011, ch. An Architectural Approach Towards the Future Internet of Things, pp. 1–24.

[90] P. van de ven, N. Hedge, L. Massoulie, and T. Salonidis, "Optimal control of residential energy storage under price fluctuations," in *ENERGY '11*, 2011.

[91] J. Venkatesh, "Short-term solar and wind energy prediction for application in datacenters," Master's thesis, University of California, San Diego, 2012.

[92] J. Venkatesh, B. Akşanli, J.-C. Junqua, P. Morin, and T. Šimunić Rosing, "Homesim: Comprehensive, smart, residential electrical energy simulation and scheduling," in *Proceedings of 2013 IEEE International Green Computing Conference, year = 2013,*.

[93] J. Venkatesh, B. Akşanli, and T. Šimunić Rosing, "Residential energy simulation and scheduling: A case study approach," in *Proceedings of 2013 IEEE International Symposium on Computers and Communications, pages = 161-166, year = 2013,*.

[94] J. Venkatesh, C. Chan, A. S. Akyürek, and T. Šimunić Rosing, "A modular approach to context-aware iot applications," in *Proceedings of the 1st Conference on Internet of Things Data and Implementation (IoTDI)*, 2016.

[95] J. Venkatesh, S. Chen, P. Tinnakornsrisuphap, and T. Šimunić Rosing, "Lifetime-dependent battery usage optimization for grid-connected residential systems," in *Proceedings of 2015 Workshop on Modeling and Simula- tion of Cyber-Physical Energy Systems (MSCPES)*, 2015.

[96] W. Wang, "A comprehensive ontology for knowledge representation in the internet of things," in *11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

[97] S. Zahedi and C. Bisdikian, "A framework for qoi-inspired analysis for sensor network deployment planning," in *WICON '07*.

[98] T. Zhu, A. Mishra, D. Irwin, N. Sharma, P. Shenoy, and D. Towsley, "The case for efficient renewable energy management in smart homes," in *BuildSys '11*, 2011.

[99] M. Zimesnick. Average cost of solar panels and installation. [Online]. Available: http://renewableenergyindex.com/solar/cost-of-solar-panels