Dissertation

Resource Allocation Optimization in the Smart Grid and High-performance Computing

Submitted by

Timothy M. Hansen

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2015

Doctoral Committee:

Advisor: Howard Jay Siegel
Co-Advisor: Anthony A. Maciejewski

Siddharth Suryanarayanan
Thomas H. Bradley

Abstract

Resource Allocation Optimization in the Smart Grid and High-performance Computing

This dissertation examines resource allocation optimization in the areas of Smart Grid and high-performance computing (**HPC**). The primary focus of this work is resource allocation related to Smart Grid, particularly in the areas of aggregated demand response (**DR**) and demand side management (**DSM**). Towards that goal, a framework for heuristic optimization for DR in the Smart Grid is designed. The optimization problem, denoted *Smart Grid resource allocation* (**SGRA**), controls a large set of individual customer assets (e.g., smart appliances) to enact a beneficial change on the electric power system (e.g., peak load reduction). In one part of this dissertation, the SGRA heuristic framework uses a proposed aggregator-based approach. The aggregator is a for-profit entity that uses information about customers' smart appliances to create a schedule that maximizes its profit. To motivate the customers to participate with the aggregator, the aggregator offers a reduced rate of electricity called *customer incentive pricing* (**CIP**). A genetic algorithm is used to find a smart appliance schedule and CIP to maximize aggregator profit. By optimizing for aggregator profit, the peak load of the system is also reduced, resulting in a beneficial change for the entire system. Visualization techniques are adapted, and enhanced, to gain insight into the results of the aggregator-based optimization. A second approach to DR in the Smart Grid is taken in the form of a residential home energy management system (**HEMS**). The HEMS uses a non-myopic decision making technique, denoted partially-observable Markov decision process (**POMDP**), to make sequential decisions about energy usage within a residential household to minimize cost in a real-time pricing (**RTP**) environment. The POMDP HEMS

significantly reduces the electricity cost for a residential customer with minimal impact on comfort.

The secondary focus of the research is resource allocation for scientific applications in HPC using a dual-stage methodology. In the first stage, a batch scheduler assigns a number of homogeneous processors from a set of heterogeneous parallel machines to each application in a batch of parallel, scientific applications. The scheduler assigns machine resources to maximize the probability that all applications complete by a given time, denoted the makespan goal. This objective function is denoted *robustness*. The second stage uses runtime optimization in the form of dynamic loop scheduling to minimize the execution time of each application using the resources allocated in the first stage. It is shown that by combining the two optimization stages, better performance is achieved than by using either approach separately or by using neither.

The specific contributions of this dissertation are: (a) heuristic frameworks and mathematical models for resource allocation in the Smart Grid and dual-stage HPC are designed, (b) CIP is introduced to allow an aggregator profit and encourage customer participation, and (c) heuristics and decision-making techniques are designed and analyzed within the two problem domains to evaluate their performance.

Acknowledgments

I would like to thank my advisors, Howard Jay Siegel and Anthony A. Maciejewski, for their guidance and support. I would also like to thank my other committee members, Siddharth Suryanarayanan and Thomas H. Bradley, for their time and effort. Additionally, I would like to thank my co-authors and paper reviewers Ioana Banicescu, Edwin K. P. Chong, Florina M. Ciorba, Elaine Hale, Eric Jonardi, Arun V. Modali, Mark Oxley, Bryan Palmintier, Robin Roche, Srishti Srivastava, Kyle Tarplee, Peter M. Young, and Daniel Zimmerle for their valuable comments and contributions throughout the course of this research. The research in this dissertation was supported by Colorado State University (CSU), including the CSU George T. Abell Endowment and the ISTeC PhD Scholar, the National Science Foundation (NSF) under grant numbers CNS-0905339 and CCF-1302693, and the U.S. Department of Energy under Contract No. DE-AC36-08-GO28308 with the National Renewable Energy Laboratory (NREL). Parts of this research used the CSU ISTeC HPC System supported by NSF grant number CNS-0923386.

This dissertation is typset in LaTeX using a document class designed by Leif Anderson.

Dedication

I dedicate this dissertation to my loving mother, Sussie Eva Hansen. Without her love and support, I would not be where I am today.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction and Overview

Many resource allocation problems are heterogeneous in nature, such as allocating heterogeneous compute resources to incoming tasks. Optimal heterogeneous resource allocation, in general, is known to be NP-complete [4–6], leading to the use of resource allocation heuristics that try to find near-optimal solutions. This dissertation addresses two heterogeneous resource allocation problems. In Chapters 2 – 6, resource allocation in the Smart Grid is studied, denoted *Smart Grid resource allocation* (**SGRA**). Resource allocation in *high-performance computing* (**HPC**) is presented in Chapters 7 and 8. Concluding remarks and future directions of research are given in Chapter 9.

The primary focus of this work is resource allocation related to Smart Grid. As Smart Grids introduce profound changes in the operation of the electric power industry, the need for efficient and robust resource allocation algorithms arises, especially due to the increasingly stochastic nature of availability of highly dispersed resources. A framework for solving the SGRA problem using a heuristic approach (e.g., a genetic algorithm) is presented in Chapter 2. Similar challenges exist in resource allocation in the realm of computing. A comparison is drawn between SGRA and resource allocation in computing. Its application to a multi-agent-based distribution management system, used as an environment model, is also proposed. A path forward will conclude the chapter.

A for-profit aggregator-based approach to the SGRA problem is introduced and used in Chapter 3. The aggregator entity, using a given set of schedulable residential customer assets (e.g., smart appliances), must set a schedule to optimize for a aggregator profit. To encourage customer participation in the residential DR program, a new pricing structure

named customer incentive pricing is proposed. The aggregator profit is optimized using a proposed heuristic framework, implemented in the form of a genetic algorithm, that must determine a schedule of customer assets and the customer incentive pricing. To validate the heuristic framework, the optimization of a large-scale system consisting of 5,555 residential customer households and 56,642 schedulable assets is simulated using real pricing data over a period of 24-hours. By optimizing purely for economic reasons it is shown that the aggregator can enact a beneficial change on the load profile of the overall power system.

Real-time pricing (RTP) is a utility-offered dynamic pricing program to perform demand response. In such an RTP market, a customer can make changes in their energy usage behavior to drastically reduce their electricity bill. A home energy management system (HEMS) is an automated way for managing energy usage within the home in response to utility pricing signals, such as the RTP. Three new HEMS techniques are designed in Chapter 4 — one myopic approach and two non-myopic partially-observable Markov decision process (POMDP) approaches — for minimizing the household electricity bill in such an RTP market. In a simulation study in Chapter 4, the performance of the new HEMS methods are compared with a mathematical lower bound and the current status quo. The non-myopic POMDP approach can provide, at the high end, a 30% monthly savings. More modest results show an average of 10% savings over the status quo. A case is also made for RTP programs even in lieu of a HEMS or changes in energy usage.

With the influx of data in the emerging Smart Grid due to technologies such as smart meters and demand response programs, it is more difficult to analyze and discover relevant and interesting information. Visualization methods are adapted in Chapter 5 for quantifying and comparing the effectiveness and profitability of a given set of solutions to a demand response problem. Using these visualization methods, it becomes possible to answer: whether

or not the demand response plan worked effectively; at what times the demand response resulted in a profit or a loss; and, how multiple demand response solutions compare. The visualization methods are presented using data from a simulation of 5,555 customer households at 26 loadpoints using the Roy Billinton Test System mapped onto the city of Fort Collins, Colorado, USA.

As more Smart Grid technologies (e.g., distributed photovoltaic, spatially distributed electric vehicle charging) are integrated into distribution grids, static distribution simulations are no longer sufficient for performing modeling and analysis. GridLAB-D is an agent-based distribution system simulation environment that allows fine-grained end-user models, including geospatial and network topology detail. A problem exists in that, without outside intervention, once the GridLAB-D simulation begins execution, it will run to completion without allowing the real-time interaction of Smart Grid controls, such as home energy management systems and aggregator control. This lack of runtime interaction is addressed in Chapter 6 by designing a flexible communication interface, Bus.py (pronounced bus-dot-pie), that uses Python to pass messages between one or more GridLAB-D instances and a Smart Grid simulator. Chapter 6 describes the design and implementation of Bus.py, discusses its usefulness in terms of some Smart Grid scenarios, and provides an example of an aggregator-based residential demand response system interacting with GridLAB-D through Bus.py. The small scale example demonstrates the validity of the interface and shows that an aggregator using said interface is able to control residential loads in GridLAB-D during runtime to cause a reduction in the peak load on the distribution system in (a) peak reduction and (b) time-of-use pricing cases.

The secondary focus of the research is resource allocation for scientific applications in HPC using a dual-stage methodology. Scheduling parallel applications on existing or emerging computing platforms is challenging, and, among other attributes, must be efficient and robust. A dual-stage framework is proposed in Chapter 7 to evaluate the robustness of efficient resource allocation and dynamic load balancing of scientific applications in heterogeneous computing environments with uncertain availability. The first stage employs robust resource allocation heuristics, while the second stage incorporates robust dynamic loop scheduling techniques. The combined dual-stage framework constitutes a comprehensive framework that enables and provides guarantees for the robust execution of scientific applications in computing systems where uncertainty is caused by various unpredictable perturbations. The chapter reports on studies for determining the best techniques to be used for each stage that: (a) maximize the probability that the system makespan satisfies a deadline, and (b) minimize the system makespan for every given availability level in the system. The usefulness and benefits of the proposed framework are demonstrated via a small scale example.

The scheduling of moldable parallel applications to clusters of processors is challenging, where the number of processors on which a moldable application executes is decided by the scheduler. When the application execution times are stochastic in nature, and the availability of the resources is uncertain, this becomes an even greater challenge. A model is presented in Chapter 8 for the stochastic execution times of moldable parallel applications that are assigned to heterogeneous parallel resources, incorporating the change in execution times when applications are mapped to different numbers of processors. To account for the uncertainties in both application execution times and resource availability, a robustness model that combines the two sources of uncertainties is proposed. Using this robustness model, three novel

iterative-greedy heuristics are developed to allocate heterogeneous resources to batches of parallel applications to maximize the probability of completing by a designated time, called the makespan goal. To verify the performance of the proposed heuristics, a simulation study is conducted using different batch and system sizes. To showcase the benefit of using the proposed iterative-greedy heuristics, their performance is studied against two comparison heuristics. The five heuristics are evaluated against the upper bound on robustness.

CHAPTER 2

A Proposed Framework for Heuristic Approaches to Resource Allocation in the Emerging Smart Grid

2.1. Introduction

As power systems and information technologies are converging to revolutionize the way electricity is generated, delivered, managed, and consumed, new challenges arise, such as: how to integrate storage efficiently, how to use demand-response of residential customers to mitigate peak demand, etc. [8]. Succinctly this can be surmised as: how should resources be allocated in the emerging Smart Grid as the stochastic nature of the availability of resources (generation, storage, and loads) becomes prevalent?

Contrary to the transmission level, where relatively few numbers of assets, albeit rated large, are used, the Smart Grid is expected to revolutionize the distribution side, where a multitude of smaller assets will be available for controlled deployment. Distributed and intermittent renewable energy sources, distributed storage elements such as plug-in electric hybrid vehicles (**PHEVs**), and the ability to schedule loads require utilities to rethink the conventional procedures of scheduling and dispatching the resources.

A similar challenge exists for computer scientists in allocating resources for computing. Several similarities can be drawn between the challenges in the domains of Smart Grids and computing. In computing, it is beneficial to allocate tasks to machines that the tasks perform well in order to optimize some system performance measure. Similarly in the emerging Smart

This work was performed jointly with the full list of co-authors available in [7]. This work was supported by the University of Technology of Belfort-Montbéliard, a seed grant from the CSU Energy Supercluster, the National Science Foundation under grant number CNS-0905399, and the CSU George T. Abell Endowment.

Grid, the availability of the resources may also be stochastic in nature. For example, this may arise due to the uncertainty in task execution times as well as the sharing of machine resources [9]. These similarities indicate the possibility of adapting approaches used in the computing realm to the resource allocation problem in the emerging area of Smart Grids.

The main contribution of this chapter is to propose a framework for addressing the large-scale distributed *Smart Grid resource allocation* (**SGRA**) problem using heuristics adapted from resource allocation methods utilized in computing. Specifically, a genetic algorithm is used to showcase the framework for the heuristic approach.

Section 2.2 describes the environment model, in which a multi-agent-based distribution system model is presented. In Section 2.3, the resource allocation problem in Smart Grids is described. An example of how heuristics have been used in the field of computing is shown in Section 2.4. Section 2.5 proposes a framework for a heuristic approach to the resource allocation problem in the emerging smart grid. In Section 2.6, a path forward to continue with the framework proposed in this chapter is indicated.

2.2. Multi-Agent-Based Distribution System Model

2.2.1. Overview. Contrary to wide-area transmission systems that have a long history of automation and smart functionalities, most Smart Grid activities are focusing more on the automation of end-user distribution systems [10]. In such an emerging Smart Grid, a multitude of assets, including local generation sources, distributed energy sources, special loads such as PHEVs, and other schedulable loads, are available for control and deployment. It is imperative to schedule and deploy such assets properly, to minimize additional stress on the grid. These activities, usually called unit commitment and economic dispatch, are used in transmission systems [11, 12], but the algorithms and constraints utilized for transmission

systems are not well suited for large distribution systems with thousands of resources. It is in that regard that a multi-agent framework for a highly distributed distribution system is presented.

2.2.2. Multi-agent Modeling. Distribution systems connect the transmission system to end-users and, therefore, include a large number of customers with various profiles that are spread over the entire distribution network. Traditional power system modeling tools usually focus on transmission and utilize data that has been aggregated from the distribution system. However, as customers are expected to play an increasingly important role in Smart Grids, new tools that take into account the diversity and stochasticity associated with the end-user must be developed to comprehensively quantify and analyze the operation of the smart grid.

Multi-agent systems (**MASs**) offer a solution to study such large systems by creating a model for each element of the system (called an *agent*) and interconnecting them to create a MAS. A MAS is thus a group of agents interacting with each other and their environment [13]. This approach also enables modeling each element separately; e.g., each customer can be modeled, with individual dynamic profiles (loads, output of local generation, PHEV, etc.), and not just the aggregated or lumped static versions of the load.

At the same time, MASs help define the interactions between the individual elements (agents) of the system. This facilitates estimating the required communication infrastructure, enabling faster real-scale deployment, and is particularly relevant as Smart Grids rely on a highly dispersed and efficient communication.

2.2.3. Multi-agent Distribution System Architecture. A multi-agent model of a distribution system is thus proposed and serves as an environment model for the resource

allocation problem. In this model, each market player of the system is modeled as an agent. The independent system operator (**ISO**) is a non-profit entity that maintains the balance between supply (generation) and demand (system load) at the transmission level. Distribution system operators (**DSOs**) are connected to the ISO, and are responsible for distributing power to their customers. Each DSO has many physical assets, such as substations and feeders, that transfer electric energy to the end user (residential, commercial, or industrial). Typically the DSOs and ISOs also share assets at their point of common coupling – substations. Each end customer has traditional loads, and may also have specialized assets such as PHEVs, and local distributed generator (**DG**) assets that may be more-or-less controllable, such as photovoltaic (**PV**) systems or standby generation units, and often, controllable loads, including, for example, heating and ventilation equipment or lighting.

Every asset in the distribution system is expected to be under the control of an agent. Fig. 2.1 presents a typical smart distribution system that could be represented by the hierarchical structure of communication flow shown in Fig. 2.2. Aggregators can act as interfaces between end-users and DSOs, notably for PHEVs charge-recharge scheduling and for provision of certain ancillary services [14]. Additionally, large DGs and storage units may also be connected directly to substations.

Each agent is in charge of controlling the actuators of the asset associated with it, using inputs from other agents and from local measurements, and subscribing to local goals. An example of a local goal could be maximizing economic return by buffering energy between lower and higher prices specified by time-of-use (**ToU**) rates. Inputs from other agents may include set point requests from a central controller; upon receipt, the agent can decide to implement certain actions based on local objectives. Although this approach might seem contrary to traditional MAS concepts, it enables the developed system to account for communication

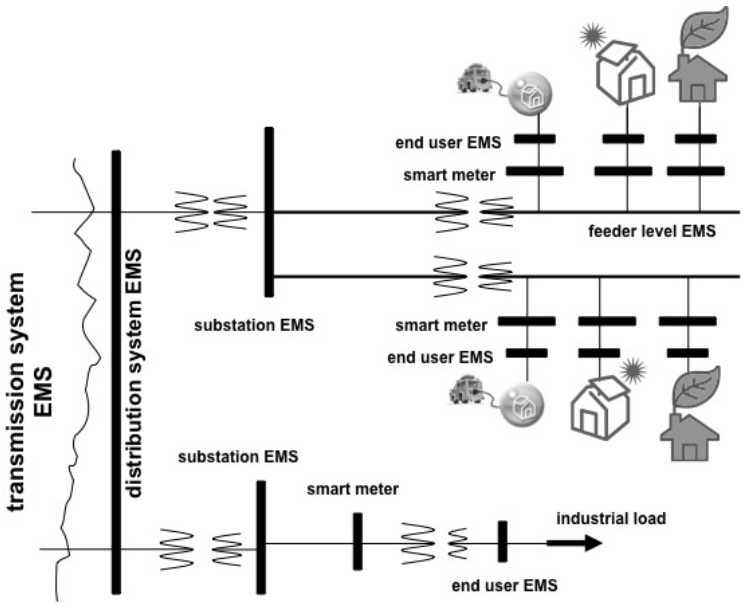


FIGURE 2.1. An illustration of an example smart distribution system with distributed assets.

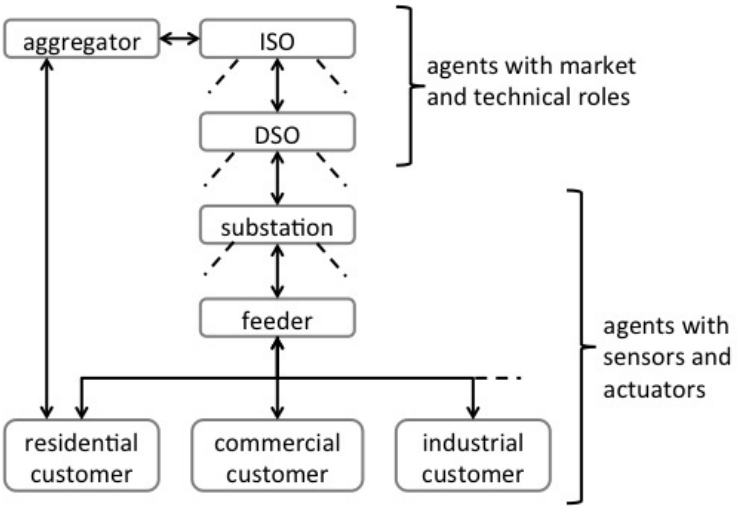


FIGURE 2.2. Architecture of the agent-based distribution system model. Dashed lines indicate connections with more agents of the same kind [15].

aspects that are an essential topic in smart grids. Based on the environment model shown in Fig. 2.2, algorithms to allocate resources (generation, storage, loads) efficiently need to be developed.

2.3. Resource Allocation in Smart Grids

2.3.1. Resource Allocation in Present Day Power Grids.

Resource allocation methods such as unit commitment (i.e., scheduling the use or non-use of generators a day in advance) and economic dispatch (i.e., optimizing the scheduled generator outputs) have been used for decades in electric grids at the transmission level for centralized assets [11]. As power grids evolve to include highly dispersed assets at the end-user domain available for control and deployment, the resource allocation problem may require some departures from the traditional techniques of optimization used for solving unit commitment and economic dispatch; such departures include heuristic optimization methods like genetic algorithms [16–19] and particle swarm optimization [20].

2.3.2. Smart Grid Resource Allocation Problem Complexity.

The resource allocation problem in Smart Grids is more complex than in traditional grids for the following reasons: (a) the number of schedulable assets in the decentralized distribution system is exceptionally large compared to the traditional centralized model of the grid, and (b) the stochastic nature of loads, generation, and storage. This stochasticity is due, in part, to the following mechanisms. First, at the transmission level, loads and DG are aggregated over a large number of units, which inherently reduces the short-term variability, while the variability in large units is well understood and characterized. For example, while loads vary in real-time with customers' activity, aggregate load can be forecast using well-known artificial intelligence techniques that have been used by utilities for decades [21].

Second, the behavior of distributed assets is part of a feedback loop, where monitoring and controlling behavior may influence future behavior. For example, a customer with a variable-rate utility contract such as ToU tariffs [8] may note that certain behaviors increase

or decrease their cost of electricity, and modify their behavior in response to the tariff's built-in incentives. The very Smart Grid technologies which allow control over the resources also provides the information required for customers to make these choices, thus increasing inherent variability.

Lastly, the increase of distributed renewable energy sources in the electric grid, especially those with intermittent inputs as wind and solar, introduces a shift from the status quo, where a smaller number of large, controllable centralized generators are dispatched concomitantly with greater certainty in output than renewable energy sources.

The deployment of storage in the grid – from PHEVs to large utility-scale storage – enables new energy management possibilities. However, maintaining the state-of-charge (**SOC**) of battery energy storage system units introduces a time-dependence; e.g., to be able to provide power during a demand peak, a unit has to charge as much as possible several hours in advance. In the case of PHEVs, the need for maintaining a certain SOC for enabling the primary function (transportation) of the asset introduces yet another constraint in the energy management scheme. Only PHEVs will be considered for storage in this paper, due to their distributed nature and relatively lower capital cost for the end-user arising from the dual use of this asset compared to dedicated energy storage devices. However, the proposed technique can be readily extended to storage units attached throughout the distribution system, whether they are large units at substations or PHEV-sized units distributed to homes.

While the end-user sector of the grid is undergoing unprecedented transformation through the "Smart Grid Initiative" [22], the transmission sector, which forms the backbone of the interconnected grid, is seeing reduced investments. Concomitantly, projected demand for electricity is expected to grow. This dichotomy is expected to result in reduced available

transmission capacity in the electrical grid [23] which may impact the system in the following ways: (a) increased bulk electricity prices, (b) reduced reliability and security of supply to the end-users, and (c) increased price differences between transmission nodes, with supply-constrained nodes seeing higher peak prices than less constrained nodes.

One of the ancillary services that the PHEV fleet could provide when functioning in the vehicle-to-grid (**V2G**) mode is the ability to locally supply the demand, thus alleviating the congestion scenario; however, for this ancillary service to provide distribution congestion, several things must happen: (a) high penetration of the PHEV fleet, (b) the willingness of end-user to allow centralized charging and discharging, (c) an infrastructure built on information exchange via control and communication, (d) the evolution of a fully deregulated retail electricity market that recognizes this ancillary service, and (e) support from vehicle manufactures, who are concerned about additional stress on the vehicle battery.

A more highly developed distributed resource is demand response (**DR**), typically implemented as programs that provide incentives to consumers for deferring or curtailing the local demand during peak system periods [24]. This is usually triggered by the service provider (i.e., the utility) based on information related to system reliability or market conditions. However, future development may rely on customer action, triggered by smart controllers monitoring customer preferences and real-time pricing signals from the utility. As a consequence of this evolution in the operation of emerging Smart Grids, newer algorithms for unit commitment and economic dispatch must be explored.

2.4. Heuristic-based Approaches to Resource Allocation in Computing

In a heterogeneous computing environment, a collection of machines that have different computational capabilities are utilized to execute tasks that have diverse computational requirements [25]. Because the environment is heterogeneous, each task will perform differently on each of the different machines. It is beneficial to allocate tasks to machines that they perform well on to optimize some system performance. In general, the problem of optimally allocating tasks to machines in a heterogeneous environment is known to be NP-complete [4–6], which leads to the use of heuristics.

The characteristics of each task, such as execution time, on each machine can be modeled either deterministically or stochastically. In the deterministic model, each task characteristic on each machine is given as a discrete value. In the stochastic model, the characteristics are represented as a probability mass function (**pmf**) [26] or a probability density function (**pdf**). In both models, the information for each task on each machine is assumed to be known beforehand.

Using the information about each task, the scheduling heuristics are used to optimize some system performance metric, such as minimize energy consumed or minimize system completion time. In the stochastic model, the resulting optimization would be a probability based on the pmfs of each task. Like in the unit commitment problem, both genetic algorithms [25, 27] and particle swarm optimizations [27] have been used. In addition, many other heuristics have been used such as Tabu [25, 27], simulated annealing [25], and k-percent best [28, 29].

Given the similarities between resource allocation in the realms of heterogeneous computing and the Smart Grid (which also involves heterogeneous resources), it makes sense to adopt similar heuristics to the emerging smart grid problem. The large, distributed nature

of the Smart Grid matches well with the high complexity of previously solved heterogeneous computing resource allocation problems. The stochasticity of resources in heterogeneous computing has been modeled as well [26] and could be adapted to the stochastic nature of the availability of resources in the Smart Grid.

2.5. Proposed Heuristic Framework for use in Smart Grids

2.5.1. Problem Formulation. The SGRA problem can be summarized as an optimization problem with the following objectives and constraints. The main objective is usually to minimize costs for design (for planning applications) and/or operations and maintenance (for DR applications). In the latter case, these costs usually include fuel costs, which do not apply for renewable energy sources. Additional objectives can include any subset of the following: (a) maximizing the share of renewable energy sources, (b) minimizing the total greenhouse gases emissions, (c) optimizing customer preferences, or (d) maximizing the reliability of the system. The system reliability may also be used as a constraint to meet, and can be considered as a robustness metric for the system [30]. To formulate the problem, let C_{tot} and E_{tot} be the total cost and total emissions for the system, respectively, $c_i(p_i)$ and $e_i(p_i)$ be the cost and the emissions, respectively, for the i^{th} asset producing an electrical power output of p_i, and I be the total number of assets, where an asset is either a conventional generator, a storage unit in the form of a PHEV, a renewable energy source, or a schedulable load. (1) describes an example with two such objectives: minimizing C_{tot} and E_{tot}.

$$min \begin{cases} C_{tot} = \sum_{i=1}^{I} c_i(p_i) \\ E_{tot} = \sum_{i=1}^{I} e_i(p_i) \end{cases} \tag{1}$$

These additional objectives can be handled either using a multi-objective Pareto-optimality based approach, in which each objective is a dimension of the Pareto front [31], or as a single-objective problem in which the additional objectives are transformed into constraints (e.g., by setting emissions or stability limits). Several constraints need to be met for the system to operate properly:

- Let p_g, p_s, p_{ls}, p_{lf}, p_res, and p_{cong} be the asset power for conventional generators, PHEVs, schedulable loads, fixed loads, renewable energy sources, and congestion needs, respectively. Similarly, let I_g, I_s, I_{ls}, I_{lf}, and I_res be the number of conventional generators, PHEVs, schedulable loads, fixed loads, and renewable energy sources, respectively. A balance between generation (supply) and the load (demand) has to be maintained at all times, as shown in (2).

$$\sum_{I_g} p_g + \sum_{I_s} p_s - \sum_{I_{ls}} p_{ls} = \sum_{I_{lf}} p_{lf} - \sum_{I_{res}} p_{res} + p_{cong} \tag{2}$$

- Let $p_{g,max}$, $p_{s,max}$, and $p_{ls,max}$, be the maximum power output of the conventional generators, PHEVs, and schedulable loads. Spinning reserve requirements should be met, as shown in (3).

$$\sum_{I_g} p_{g,max} + \sum_{I_s} p_{s,max} - \sum_{I_{ls}} p_{ls,max} \geq \sum_{I_{lf}} p_{lf} - \sum_{I_{res}} p_{res} + p_{cong} \tag{3}$$

- Let $p_{i,min}$ and $p_{i,max}$ be the minimum and maximum power output for asset i. The minimum and maximum operation range for each asset must be respected, as shown in (4) [12].

$$p_{i,min} \leq p_i \leq p_{i,max} \tag{4}$$

- Let Rd_i and Ru_i be be the ramp up and ramp down rates, respectively, for asset i. Let $\frac{dp_i}{dt}$ be the current ramp rate for asset i. The ramp rates of all conventional generators and PHEVs must be respected, as shown in (5) [12].

$$Rd_i \leq \frac{dp_i}{dt} \leq Ru_i \tag{5}$$

- Let Td_i and Ru_i be the current down and up times for generator i, respectively. Let $Td_{i,min}$ and $Tu_{i,min}$ be the minimum down and up times for generator i, respectively. The minimum up and down times for each generator must be met, as shown in (6) and (7) [12].

$$Td_i \geq Td_{i,min} \tag{6}$$

$$Tu_i \geq Tu_{i,min} \tag{7}$$

- For the energy storage elements considered here, i.e., the PHEVs, in addition to the power operation and ramp ranges, as shown in (4) and (5), they also have a few additional constraints. Let SOC be the usable state-of-charge, N_c be the number of daily battery cycles, and T be the charging target (i.e., the PHEV should be charged when the customer wants to use it, at time T). The constraints on SOC bounds, battery cycling, and charging are shown in (8)–(10), respectively.

$$SOC_{min} \leq SOC \leq SOC_{max} \tag{8}$$

$$N_c \leq N_{c,max} \tag{9}$$

(10) $$SOC_{t=T} = 100\%$$

- Let V_j be the voltage magnitude on bus j. Let V_{min} and V_{max} be minimum and maximum bounds on the bus voltage magnitudes. The proper operation of the system with regard to the bounded bus voltage magnitudes must be respected, as shown in (11).

(11) $$V_{min} \leq V_j \leq V_{max}$$

- Let f be the frequency of the system. Let f_{min} and f_{max} be minimum and maximum bounds on the system frequency. The proper operation of the system with regard to the frequency must be respected, as shown in (12).

(12) $$f_{min} \leq f \leq f_{max}$$

- Let S_k be the power flow on line k. Let $S_{k,max}$ be the maximum power flow on line k. The proper operation of the system with regard to the maximum power flow must be respected, as shown in (13).

(13) $$S_k \leq S_{max}$$

- The constraints resulting from the preferences set by customers for their assets, such as: the time for enabling DR and the set of loads available of scheduling should be met.
- The impact of DR on the customer should be as low as possible (i.e., it should ideally be as transparent as possible).

The problem at hand is thus non-linear, and the solution space may potentially reach unmanageable sizes for distribution systems [30]. Heuristics, such as genetic algorithms, are well suited for this kind of problem.

2.5.2. Genetic Algorithm Approach to Smart Grid Resource Allocation. To show how a heuristic approach to solving the resource allocation and scheduling problem in the Smart Grid could be used, a possible setup for a genetic algorithm is presented. As stated in Sections 2.3.1 and 2.4, a genetic algorithm has already been shown to solve both the unit commitment problem and the heterogeneous resource allocation and scheduling problem for computing. As such, the genetic algorithm is thought to be an apt choice to showcase the framework for a heuristic approach to resource allocation in the Smart Grid.

To properly utilize a genetic algorithm in different domains, the encoding mechanism of the genetic algorithm must be created to represent the optimization problem's variables. In the case of the Smart Grid, the variables in question are the on/off states of each of the assets, as well as their power output. The assets that we are modeling as controllable in this problem are the conventional generators, the PHEVs, and the schedulable loads. These represent the left hand side of (2) and (3). The values that we are assuming are fixed are the fixed loads, the uncontrolled renewable energy sources, and the congestion needs as requested by the ISO. The three fixed values represent the right hand side of the same equations.

At the lowest level in the genetic algorithm exists the *gene*. To model the SGRA problem, each gene represents an asset that is controllable, i.e. the values on the left hand side of (2) and (3). Let $\hat{u}_i$ be a vector whose j^{th} element is a binary value representing the on/off value of asset i at time $j * 0.25$ (i.e., the vector elements represent a 15-minute block of time). Let $\hat{o}_i$ be a vector whose j^{th} element is a real value representing the discrete output power of asset i at time $j * 0.25$. The gene of each asset is then comprised of a 96×2

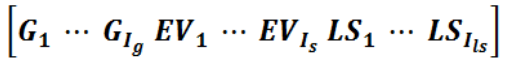


FIGURE 2.3. Chromosome representation for the SGRA framework.

matrix, $[\hat{u}\ \hat{o}]$, representing the on/off state and the output values for an asset over a 24 hour period. Additionally, each asset has a fixed availability vector, A_i, associated with it whose j^{th} element is a binary value representing whether or not a given asset is available at hour $j*0.25$. The availability vector is separate from the asset gene and is assumed to be provided by the consumer for each asset. Thus, the power output for asset i at time j is given by (14).

$$p(i,j) = A_i[j] \times \hat{u}_i \times \hat{o}_i \tag{14}$$

One entire solution to the SGRA problem is represented in a chromosome. The chromosome is made up of $I_g + I_s + I_{ls}$ genes, each representing one asset of the system. Let G_i be the gene for a conventional generator i, EV_i be the gene for PHEV i, and LS_i be the gene for schedulable load i. One chromosome, or solution, is shown in Fig. 2.3.

Each solution has a fitness value, or values, associated with it. These values are used to evaluate the chromosome in the dimensions that are trying to be optimized. For example, if trying to optimize for the values in (1) there would be a fitness value associated with both the C_{tot} and E_{tot} objectives (if using a multi-objective Pareto-optimality based approach). As stated before, the multi-objective optimization problem can be turned into a single objective optimization problem by optimizing over one objective representing the other objectives as constraints. Another way to accomplish this is to place weights on each of the objectives and combine them into a single fitness value.

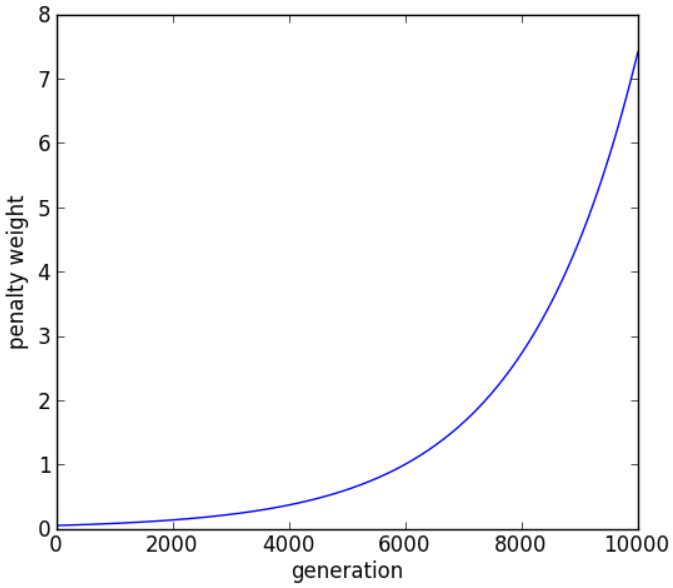


FIGURE 2.4. An example of how the penalty weight for violating a constraint will increase with the number of generations.

To accommodate for the constraints in the SGRA problem, penalty functions will be used. If a constraint is violated, a penalty will be included in the chromosome's fitness value. The penalty value is a function of the magnitude of violation and the current generation of the genetic algorithm. The reason that the penalty is a function of the generation is because at earlier generations it is beneficial to keep a variety of genetic material. Even if a chromosome violates a constraint, it might have a partial solution that performs well with respect to the objectives. A chromosome is kept in the population even though it may violate a constraint because it might be able to produce a child that performs well and fixes the constraint violation as it evolves. As the number of generations increases, however, the genetic algorithm is less likely to keep a chromosome that violates any of the constraints. In the final Pareto front (if multi-objective), none of the solutions should contain any constraint violations. An example of a possible penalty weight is shown in Fig. 2.4.

```
initialize population;
evaluate population;
while termination criterion not reached
{
      select solutions for next population;
      perform crossover and mutation;
      evaluate population;
}
```

FIGURE 2.5. Basic genetic algorithm procedure.

Let $F_a(x)$ be the fitness function associated with objective a for chromosome x, $F_a^{'}(x,t)$ be the fitness function, including penalty weights, associated with objective a for chromosome x in generation t, n_c be the number of constraints, v_b be a binary value representing whether or not constraint b is violated, m_b be the magnitude that constraint b is violated, and $X_b(t)$ be the penalty weight associated with constraint b in generation t (i.e., what is shown in Fig. 2.4). The fitness value being optimized with the genetic algorithm is shown in (15). Note that $F_a^{'}(x,t) = F_a(x)$ if no constraints are violated (i.e., $v_b = 0 \; \forall \; b = 1, ..., n_c$).

$$F_a^{'}(x,t) = F_a(x) + \sum_{b=1}^{n_c} v_b m_b X_b(t) \tag{15}$$

The basic genetic algorithm procedure is shown in Fig. 2.5 [32]. In the initial population, it is usually beneficial to have some type of genetic preconditioning in the form of seeding. This seeding uses some number of solutions in the initial population that are not generated at random. This can be done by running less computationally intensive heuristics to generate some initial seeds. In the case of the SGRA, it might be beneficial to precondition the population with some initial seeds that do not violate any constraints.

There are many different ways to perform crossover selection (such as tournament selection [32] or linear bias [33]), crossover, and mutation. For the purpose of the framework,

these will be left as generic genetic operators in the scope of this chapter. It is important to note that the crossover and mutation operators should take into account the change in power outputs from the changed assets to meet the power balance constraint in (2). It should be noted that this will most likely not be a trivial matter to produce crossover and mutation operators that will respect the power balance constraint.

A genetic algorithm is a valid heuristic approach to solving large-scale optimization problems and as such was used as an example. In addition to the ability to find near-optimal solutions, in one run of the genetic algorithm many solutions are found (equal to the population size) with different characteristics. In the SGRA problem, solutions might have similar fitness values, but one might have, for example, a much larger spinning reserve that might be beneficial to the system in question.

2.6. Path Forward

Going forward there are many aspects of the proposed framework to be explored. Perhaps the most obvious is to implement and evaluate the proposed genetic algorithm (including developing smart crossover and mutation operators) and against realistic distribution scenarios. In this light, other heuristics will be implemented as comparisons to the genetic algorithm. For practicality purposes as a day-ahead scheduler, the performance of the different heuristics should be explored as well as how their execution time scales with the size of the problem. It would also be interesting to explore the temporal and spatial stochasticity of the renewable energy sources, PHEVs, schedulable loads, and conventional generators. With this added stochasticity, metrics of robustness (as defined in [26]) would be useful for characterizing the system.

Certain asset characteristics extend beyond the 24-hour planning period, which open questions and approaches which support coarse, long-term optimization in conjunction with day-ahead scheduling. For example, emissions limits on generation units, storage capacity in pump-hydropower units, and thermal ride-through in commercial buildings all include multi-day dynamic.

As the applicability of any resource allocation algorithm to the Smart Grid domain has to be tested using power systems analysis software, a co-simulation framework introduced in [15] may be used. This unique test bed available at Colorado State University enables coordinated simulation of communication, control, and power system aspects of energy management systems.

CHAPTER 3

Heuristic Optimization for an Aggregator-based Resource Allocation in the Smart Grid

3.1. Introduction

According to the United States Department of Energy, since 1982 the growth in peak electricity usage has exceeded the growth in transmission capacity by almost 25% each year [35]. Furthermore, electricity sales in the residential sector in the United States is expected to grow 24% from the 2011 reference case to 2040 [36]. Given these trends, peak energy demands are expected to exceed the available transmission capability. This can be dealt with by increasing transmission capability, creating distributed generation, or curtailed load. As shown in [35], it is unlikely that additional spending will be allocated for increasing transmission capability, leading to research in the areas of distributed generation (**DG**) and, in the case of this chapter, curtailing load during peak hours.

In addition to the physical considerations, there is also an economical motivation. By curtailing load during peaks, electricity costs could be drastically reduced by eliminating the need for peaking power plants. In a study from [37], "a 5% reduction in peak demand during the California energy crisis of 2000–2001 would have reduced the highest wholesale prices by 50%." To reduce peak demands, the scheduling of customer appliances are intelligently coordinated away from the peak time, alleviating the peak demand, and offering a benefit to all parties.

This work was performed jointly with the full list of co-authors available in [34]. This work was supported by the National Science Foundation under grant numbers CNS-0905399 and CCF-1302693, the CSU George T. Abell Endowment, and the University of Technology of Belfort-Montbéliard.

Given both the physical and economical motivations, an aggregator-based residential demand response (**DR**) program is presented. The aggregator is a proposed for-profit entity in a deregulated market structure that interfaces a DR market (**DRX**) and a set of customers. The aggregator will possess information about the schedulable assets of the participating customers. In many, if not all energy markets, there is a minimum power rating required to bid into the market (e.g., 0.1 MW in PJM [38]). By aggregating the customer assets, the aggregator is able to enact a noticeable change on the overall system by scheduling the assets of many customers and bidding the aggregation of their assets, where a single customer would not be able to do so.

As presented in [39], incentives can influence customer behavioral changes. To encourage customers to participate with the aggregator on a daily basis, a new day-ahead price is proposed for electricity offered by the aggregator, in the form of customer-incentive pricing (**CIP**), to offset the customers' inconvenience of the aggregator controlling their assets. Additionally, if the inconvenience of rescheduling the load is not worth the reduced price, the customer may refuse the aggregator and instead pay the utility company for electricity.

The SGRA problem in this chapter is formally stated as given a set of customers and information about their respective assets, subject to customer constraints (i.e., availability of customer assets and customer incentive requirements), how can the aggregator find the *CIP* and *schedule of assets* to maximize aggregator profit? In this chapter, it is demonstrated that by optimizing solely for the profit of the aggregator, a change on the peak load of the system will be enacted because this is where most of the profit can be made due to the high cost of peak generators.

To solve the SGRA problem, concepts are borrowed from resource allocation in computing where tasks must be allocated to machines to optimize a performance metric, such as

completing all tasks as quickly as possible. It has been shown, in general, that such problems are NP-complete [4–6] and, as such, use heuristic optimization to find *near-optimal* solutions. Similarly in this chapter, heuristic optimization techniques are used to find near-optimal solutions to the SGRA problem in a time frame that is reasonable with the large number of assets considered for use as a day-ahead scheduler.

Related prior work on demand side management in Smart Grid has occurred in the areas of optimization and aggregation of end-user resources. The optimization of scheduling end-user resources has been approached as linear programming [40, 41], dynamic programming [42], and mixed integer programming [43]. Heuristic-based methods also have been used in the form of particle swarm optimization [44], evolutionary algorithms [45], and multi-agent systems [46]. However, increasing DR technologies and allowing retail customers direct access to wholesale market prices may increase the price-elasticity of demand, leading to increased volatility in power systems [47]. Aggregators are an intermediary entity that offer centralized coordination of many entities [14, 48–50]. The SGRA differs in that many more distributed residential customer assets are scheduled and a new time-variant customer pricing mechanism in the form of CIP in introduced, that exists in conjunction with the utility price, to encourage customer participation.

According to the California Energy Commission (**CEC**), residential loads are not easily controlled and need to be composed of a large portfolio of assets to provide a strategic DR product [51]. The CEC identified strategies to fulfill its DR requirements including: direct DR participation with the independent system operator (**ISO**), new market and auction mechanisms (e.g., the proposed DRX), improving customer willingness to participate, and the introduction of time-variant pricing. The work in this chapter directly addresses each of these strategies, offering direct DR participation through the customer-aggregator-DRX

relationship (see Fig. 3.1) and encouraging customer participation with the time-variant CIP mechanism.

In this chapter it is theorized that by using an aggregator placed between the customer and bulk power market the volatility in the power system can be reduced. In Chapter 2 (based on [7]), the use of a heuristic approach to the SGRA was hypothesized. In this chapter, the heuristic framework is designed and implemented using a simulation test bed of 5,555 customers to simulate the scheduling of over 56,000 consumer devices centrally controlled by the aggregator. This chapter makes the following contributions that is believed to be missing in prior work:

(a) A new customer pricing structure is proposed in the form of customer-incentive pricing to encourage customer participation in residential demand response.

(b) A heuristic optimization framework is designed to implement and solve the SGRA problem.

(c) An analysis of the heuristic framework using actual electricity pricing data and a large-scale simulation test system consisting of 5,555 customers and 56,642 schedulable assets is conducted.

(d) An aggregator-based approach for a residential demand response program for use in scheduling customer assets in a large-scale manner.

In the simulation study, by optimizing for profit, the aggregator was able to reduce the peak load of the 5,555 participating customers by 12.5%. It is demonstrated that this change benefits the customer of the aggregator (in the form of reduced cost of electricity for schedulable loads), the aggregator (in the form of a profit), and also those customers *not* participating with the aggregator (because the overall system peak is lowered as a common good).

The rest of the chapter is organized as follows. Section 3.2 describes the system model and the enabling technologies. In section 3.3, a heuristic framework and genetic algorithm implementation are presented. The setup for the simulation study is discussed in Section 3.4. Section 3.5 examines the simulation results. Section 3.6 concludes.

3.2. System Model

3.2.1. Cyber-physical System. The proposed cyber-physical system (**CPS**) for the aggregator-based residential DR program is shown in Fig. 3.1. On the right of Fig. 3.1 is the traditional power system and market structure that flows from the ISO to the distribution system operator (**DSO**) for delivering electricity to the residential customer. The left-hand side of Fig. 3.1 encapsulates our proposed residential DR program. The DRX is an ancillary market in a fully deregulated market structure that provides DR services to the ISO. The aggregator interfaces the DRX and the residential customer, and provides the positive attributes (e.g., load shifting, distributed storage) of the aggregated customer assets (e.g., distributed generation, electric vehicles) to the ISO. Each participating customer has a home energy management system (**HEMS**) that controls the assets, connected to a smart meter. The aggregator coordinates the use of the participating customer assets and brings the result (e.g., load reduction) to the DRX for market exchange. The aggregator and customer interactions will be expanded on in the following subsections.

For realizing the market interactions in Fig. 3.1, several enabling technologies are first expected to penetrate the electric power system, and are assumed to exist in this work. As previously mentioned, the retail electricity market must be fully deregulated, allowing for the customer to choose between suppliers. The control and communication infrastructure, including the requisite cyber-security, for the exchange of information and coordination of

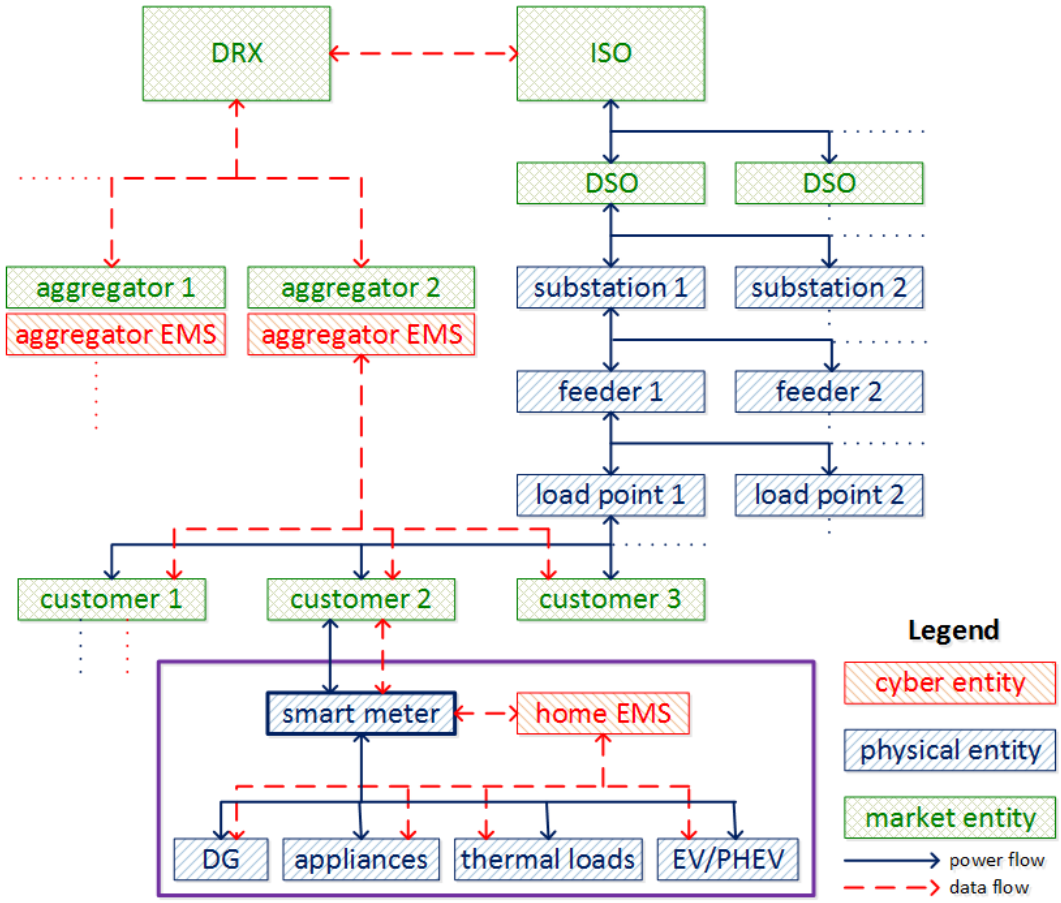


FIGURE 3.1. The architecture and communication for the cyber-physical system of the proposed aggregator-based residential demand response program.

customer assets must be developed and implemented. Lastly, the customer must be willing to participate with proper incentive.

3.2.2. AGGREGATOR. The aggregator is a for-profit market entity engaged in interacting with the customer and the bulk power market in a fully deregulated market structure. As shown in Fig. 3.1, the aggregator is situated between the DRX and the customer (in a fully deregulated market structure). Note that the DRX can exist in conjunction with existing deregulated market structures. The aggregator energy management system interacts with each of the customer HEMSs. In this chapterwe are only considering one aggregator entity, but it is expected that several aggregators may exist within the same distribution area.

The existence of an aggregator would depend on legislative policies, but this is beyond the immediate scope of this research.

The aggregator coordinates a set of participating customers, each with a set of schedulable loads. In this chapter, only schedulable loads in the form of smart appliances are being considered, but this approach could be extended to other types of assets such as DG, thermal loads (e.g., electric water heaters [52]), and PHEVs (in the form of vehicle-to-grid [53] or scheduling vehicle charging cycles [44]).

The scheduling problem is proposed as a day-ahead optimization. To make decisions, the aggregator requires information about the customer loads, the forecast utility pricing, and the forecast spot market pricing in the bulk electricity market. Using this information, the aggregator must find a *CIP* and a *schedule of loads* to maximize its profit. Because it is a day-ahead optimization, there are constraints on the execution time of the optimization technique used. This time constraint, along with the complexity of the scheduling problem (i.e., the class of problems is, in general, NP-complete) due to the large number of customer assets leads to the use of heuristics. Other objectives could be considered, such as minimizing the peak load, or considering multiple objectives in the form of a multi-objective optimization using Pareto-fronts [54]. In this chapter, only aggregator profit is optimized to demonstrate that a purely economic motivation will affect the desired change of reduced peak demand on the entire system.

CIP is a proposed pricing structure that the aggregator would offer all customers to allow the rescheduling of their loads. That is, instead of paying the utility company, the customer pays the aggregator the CIP for electricity. The customer paying the CIP for electricity to the aggregator at the time the asset has been rescheduled *to* is one part of the profit of the aggregator. The sum of these payments over all customers and all rescheduling events is

denoted S. The other two components to the aggregator profit are: (a) the aggregator selling a negative load to the spot market where the assets have been rescheduled *from* (denoted N), and (b) the aggregator buying spot market electricity where the assets have been rescheduled *to* (denoted B). This exchange is outlined in Fig. 3.2. The aggregator would, perhaps, need to enter into a leasing agreement with the utility company for the use of the distribution assets, but modeling this and other potential fixed costs are beyond the immediate scope of this chapter.

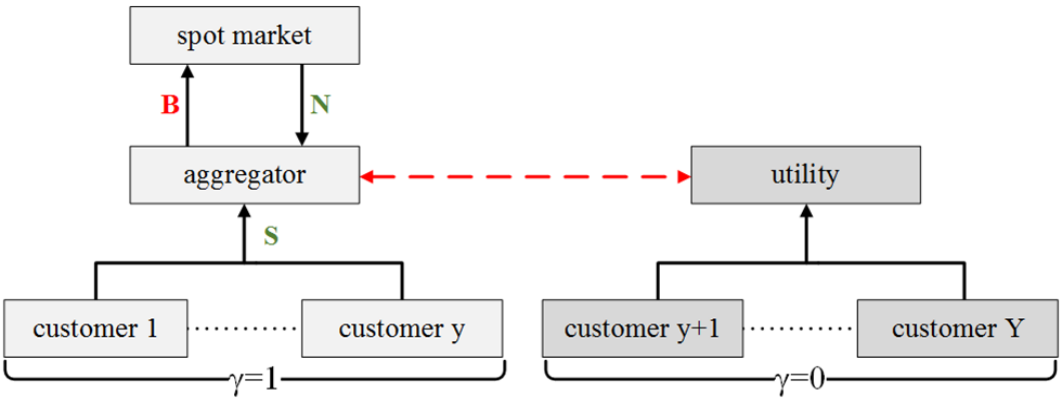


FIGURE 3.2. The money flow with respect to the aggregator, customer, spot market, and utility. The customer has a choice of electricity provider. Customers $\{1 \dots y\}$ pay the customer incentive pricing to the aggregator for their schedulable loads. Customers $\{y+1 \dots Y\}$ decide the customer incentive pricing is not worth the inconvenience and purchase electricity from the utility company. The solid arrows represent the money flowing in the system. The dashed red arrow indicates the possible need for a relationship between the aggregator and utility company, which is beyond the scope of this research.

3.2.3. CUSTOMER. Each customer under agreement with the aggregator has a set of schedulable loads. In this chapter, as mentioned in Subsection 3.2.2, only flexible, non-interruptible smart appliances, according to the definitions given in [55], are being considered. Each customer load has an *availability window* associated with it. The availability window describes the times during the day that a customer will allow their schedulable load to be rescheduled. In addition to the availability window, each customer has a pricing point that

must be met on each load to allow it to be rescheduled. That is, if the price reduction to be received from a rescheduled load at the given CIP is not worth the inconvenience to the customer, the customer may choose to not have their load rescheduled and instead pay the utility company for electricity for that load. This is shown in Fig. 3.2 as the set of customers $(y+1)$ to Y interacting with the utility instead of the aggregator. Only those loads that are agreed for DR between the customer and aggregator utilize the CIP. The base load and those loads not agreed upon will utilize the status quo of the utility company, e.g., real-time price and time-of-use. This choice of supplier is a powerful new tool for the customer and offers the customer an avenue to participate in the spot market (through the aggregator entity), which may reduce the customer electricity bill and offer freedom of choice.

3.3. Heuristic Framework

3.3.1. Framework.

3.3.1.1. *Overview.* The SGRA problem is solved in this chapter using a heuristic optimization framework, borrowed from concepts in resource allocation in computing, that finds near-optimal solutions to problems. Heuristic optimization methods are used because, in general, the class of problems is NP-complete. In this chapter, the heuristic framework is designed to be a day-ahead optimization, using a resolution of 15-minute intervals. This implies that a given heuristic would need to have a runtime of less than 24-hours to be useful. This also gives each vector 96 entries (96 15-minute intervals for a complete 24-hour period).

3.3.1.2. *Schedulable Loads.* To reschedule load, the aggregator requires information on the set of schedulable loads. These schedulable loads represent a subset of the system load. For each schedulable load i, the aggregator receives information from the customer on:

- δ_i, the runtime duration (in 15-minute intervals)

- p_i, the average power rating (in kW)
- t_{i_start}, the customer scheduled start time
- (A_{i_start}, A_{i_dur}), a 2-tuple that represents the customer-defined availability window for load i determined by the availability window start time, A_{i_start}, and the availability window duration, A_{i_dur}.

In this chapter, it is assumed that the aggregator knows the exact time a load will run (i.e., from $t_{i,start}$ for δ_i time intervals) if it is not rescheduled by the aggregator (i.e., the start time is deterministic).

3.3.1.3. *Aggregator.* Let $\boldsymbol{\lambda}$ be the CIP vector containing 96-elements, where each element λ_t gives the aggregator determined CIP at time interval t. In addition to the information about the schedulable loads, the aggregator possesses information on:

- $\gamma(i, \boldsymbol{\lambda}, t)$, a binary function that represents whether the customer will allow load i to be rescheduled to time t with CIP $\boldsymbol{\lambda}$ ($\gamma = 1$) or not ($\gamma = 0$)
- $s(t)$, the forecast spot market price of electricity in the bulk electricity market (in cents/kWh)
- $r(t)$, the forecast price of electricity from the utility company (in cents/kWh).

Because the customer also has access to the forecast utility price (e.g., real-time price and time-of-use), if the CIP, $\boldsymbol{\lambda}$, does not offer enough of a reduction in pricing to justify the inconvenience of rescheduling the load, the customer has the opportunity to refrain from participation, as represented by the binary function, γ. Therefore, the position of the aggregator is to find the following:

- L, the set of loads the aggregator is rescheduling
- t_{i_resch}, the rescheduled start time for load i
- $\boldsymbol{\lambda}$, the CIP vector

so as to maximize profit, given in the following subsection. Let I be the total number of schedulable loads. The cardinality of L is less than or equal to I (i.e., $|L| \leq I$) because the aggregator has information about all I schedulable customer loads, but it does not necessarily have to reschedule all loads.

3.3.1.4. *Objective Function.* The monetary exchange, representing the aggregator profit, is shown in Fig. 3.2. For the aggregator, let S be the *total income received* for selling electricity to customers, given by (16), N be the *total income received* for selling negative load to the spot market given by (17), and B be the *total cost paid* to the spot market for buying electricity given by (18). The exact payment received from N would depend on policy, such as the outcome of FERC Order 745 [56] and its future iterations; however, energy policy is not addressed in this work. It is assumed that the aggregator is a well-behaved agent that does not manipulate the market (such as by misrepresenting the sum of the negative load) and is paid the difference from a deterministic baseline load. The calculations for S, N, and B are given as:

$$S = \sum_{i \in L} \left[\gamma(i, \boldsymbol{\lambda}, t_{i\_resch}) \sum_{t=t_{i\_resch}}^{t_{i\_resch}+\delta_i-1} \frac{\lambda_t p_i}{4} \right], \tag{16}$$

$$N = \sum_{i \in L} \left[\gamma(i, \boldsymbol{\lambda}, t_{i\_resch}) \sum_{t=t_{i\_start}}^{t_{i\_start}+\delta_i-1} \frac{s(t) p_i}{4} \right], \tag{17}$$

$$B = \sum_{i \in L} \left[\gamma(i, \boldsymbol{\lambda}, t_{i\_resch}) \sum_{t=t_{i\_resch}}^{t_{i\_resch}+\delta_i-1} \frac{s(t) p_i}{4} \right]. \tag{18}$$

The forecast aggregator profit P is given as:

$$P = N + S - B. \tag{19}$$

The heuristic optimization problem is set up as follows:

$$\max_{t_{i\_resch} \forall i \in L, \boldsymbol{\lambda} = (\lambda_1, ..., \lambda_{96})} \quad P \tag{20}$$

subject to

$$A_{i\_start} \leq t_{i\_resch} \leq A_{i\_start} + A_{i\_dur} \qquad \forall i \in L \tag{21}$$

and

$$t_{i\_resch} \in \mathbb{Z} \qquad \forall i \in L \tag{22}$$

$$\lambda_t \in \mathbb{R} \qquad t = 1, ..., 96. \tag{23}$$

3.4. Simulation Setup

3.4.1. Overview. The following section describes parameters and models that are used to conduct the simulation study for analysis. The heuristic framework introduced above can be used with any optimization technique, utility pricing mechanism, customer behavior model, and set of customer smart appliances. Although the results show a profit for the aggregator in the considered distribution system, this does not indicate that an aggregator entity would be profitable in all distribution systems; however, the proposed framework can be used to determine this profitability using relevant data.

3.4.2. Genetic Algorithm. In this research, a Genitor [33] version of genetic algorithm (GA) is used to implement the heuristic framework. A GA is used as an example global search heuristic, but any optimization method can be used with the described framework. GAs have been shown to work well in many optimization problems, such as resource allocation in computing [57, 25, 58], economic dispatch [16], and unit commitment [18]. If multiple objectives are used, the GA can easily be extended to generate Pareto fronts, e.g., with NSGA-II [54, 59].

The implemented chromosome structure is broken into two parts, each with its own gene type, shown in Fig. 3.3. The first portion of the chromosome is dedicated to the CIP vector, $\boldsymbol{\lambda}$, containing 96 genes representing the price (in cents/kWh) for the corresponding 15-minute interval. The second portion of the chromosome represents the schedule of loads, containing one gene for each of the I customer schedulable loads. Let t_{i_sch} be a real value in the interval $[0, 1]$ representing the scheduled start time of load i. To obtain the time interval that each load i is scheduled, the following equation is used: $t_{i\_resch} = A_{i\_start} + t_{i\_sch} A_{i\_dur}$. If $t_{i\_resch} = t_{i\_start}$, then the load is not being rescheduled (i.e., $i \notin L$). The $[0, 1]$ representation of t_{i_sch} is used to avoid violating the customer-defined availability constraints of the loads given in (21).

Figure 3.3. The chromosome structure for the genetic algorithm. The genes $\lambda_1..\lambda_{96}$ represent the customer incentive pricing vector, one element for each 15-minute interval in the 24-hour period. The genes $t_{1\_sch}..t_{I\_sch}$ represent the schedule for the I customer loads that are schedulable.

The Genitor version of the GA has a few defining characteristics. In the initial population, no duplicates are allowed to prevent premature convergence. The Genitor is a

steady-state algorithm that maintains a ranked list of chromosomes (in our study, ranked by (19)), leading to implicit elitism, i.e., between generations, the best solutions are kept. In each generation, two parents are selected using the linear bias function (as defined in [33]) leading to the creation of two new children. The linear bias selection function requires a linear bias parameter that is a real value in the interval $(1, 2]$. A linear bias parameter of 1.5 means the best-ranked solution has a 50% greater chance of being selected than the median solution.

After two chromosomes are selected using the linear bias function, two search operators are applied: crossover and mutation. The former uses a two-point crossover performed on each of the two portions of the chromosome separately. After the crossover is performed, two new children are created. Within each child, every gene has a probability of mutation that will randomly generate a new value for that gene. These two new children are then evaluated in terms of the objective function (given in (20)), inserted into the sorted population, and the worst two chromosomes are trimmed, leading to a fixed population size. The complete algorithm is shown as pseudocode in Fig. 3.4. A parameter sweep was used to determine the

```
 1: initialize population
 2: order population by (19)
 3: repeat
 4:    select two chromosomes via linear bias
 5:    crossover creating two new chromosomes
 6:    mutation
 7:    insert children chromosomes
 8:    trim the two worst performing chromosomes
 9: until stopping criterion
10: return  best chromosome
```

FIGURE 3.4. Genitor algorithm.

best parameters to use for the GA in the scope of this problem. The population size was 100, the linear bias parameter was 1.4, and the probability of mutation was 0.01. The stopping

criteria was defined as 500,000 total iterations or 10,000 iterations without an increase in the objective function.

Let ω be a real value in the interval $[0, 1]$. To seed the CIP vector, $\boldsymbol{\lambda}$, in 50 chromosomes in the initial population, a seeding function was used, denoted $\sigma(t, \omega)$, for each time-window $t = 1, ..., 96$, given by (24). The schedule for the customer load was randomly generated for each seed. The 50 seeds were generated using values $\omega = \frac{n}{49}, n = 0, ..., 49$.

$$\sigma(t, \omega) = \begin{cases} \omega s(t) & s(t) \geq r(t) \\ \omega r(t) & s(t) < r(t) \end{cases} \tag{24}$$

The rest of the chromosomes in the initial population are randomly generated. For each gene in the CIP vector, representing the cost in cents/kWh at time t, a random value is generated in the interval $[0, \max(r(t), s(t))]$. For each gene in the schedule, representing the scheduled time of load i, a random value in the interval $[0, 1]$ is generated.

3.4.3. Pricing Data. The utility pricing and spot market pricing information used in the simulation were real data from Saturday July 9, 2011, obtained from ComEd Residential Real-time Pricing [1] and PJM [2], respectively. This data is given as 24 one-hour intervals. The day-ahead forecast pricing is given in Fig. 3.5(a) and the actual pricing is given in Fig. 3.5(b). The data in Fig. 3.5(a) is used by the GA to determine $\boldsymbol{\lambda}$ and the schedule of loads. In Section 3.5, we will evaluate the aggregator profit using the actual price data in Fig. 3.5(b) with the solution obtained using the forecast price data.

3.4.4. Customer.

3.4.4.1. *Customer Overview.* In the simulation study, 5,555 customers were considered. Each customer has a baseline load and a set of schedulable loads, as described in Subsection

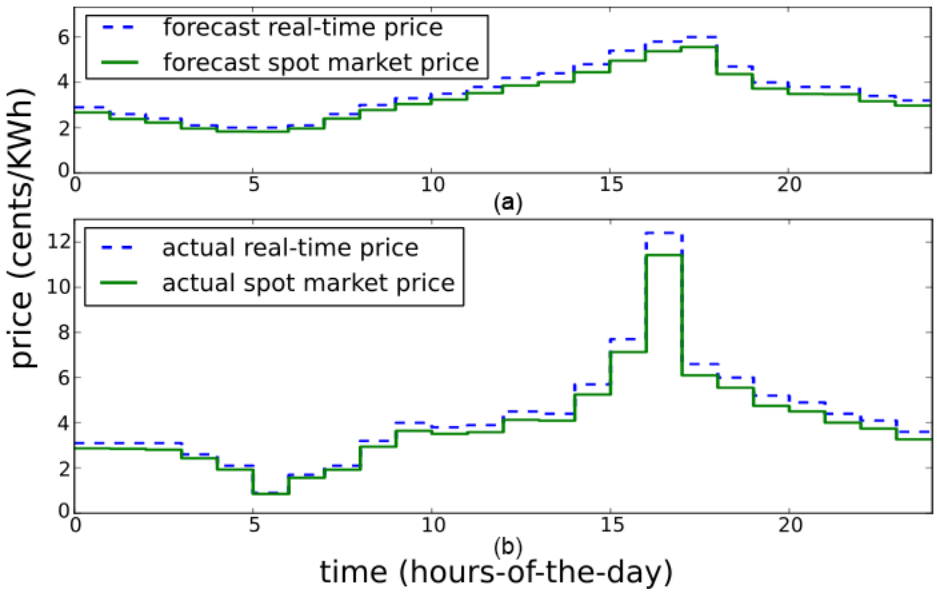


FIGURE 3.5. Real-time [1] and spot market pricing [2] from July 9, 2011. (a) The day-ahead forecast price. (b) The actual price.

3.4.4.3. When the aggregator wants to reschedule a customer load, the customer may veto (i.e., $\gamma = 0$) using the process described in the following subsection. In this case study, 56,642 loads were available to be rescheduled from the 5,555 customers.

3.4.4.2. *Customer Behavior.* A key assumption in the proposed DR methods is customer participation. The behavior of each customer is modeled for determining whether or not they will allow the aggregator to reschedule their smart appliances using the proposed α-model. In the α-model, each schedulable load i has an associated threshold metric for "customer comfort" in percent, α_i. Let c_{i_0} be the original cost of running load i at the utility real-time price and c_{i_sch} be the rescheduled cost of running load i at the CIP offered by the aggregator. For the owner of load i to allow it to be rescheduled (i.e., $\gamma = 1$), the inequality $c_{i\_sch} \leq \alpha_i c_{i\_0}$ must hold. This new model allows flexibility for the customer on a load-by-load basis. Additionally, the customer is always guaranteed (if its loads are used by the aggregator), to save $1 - \alpha_i$ times the cost of running load i compared to paying the real-time price. The user inconvenience of the rescheduling of loads is captured through the γ value as

opposed to the time dependent models in [60] and [61]. The customer γ values are private, and the aggregator is assumed to operate without receiving this information explicitly.

The coefficient-of-variation-based method is used to generate the α values for each load i, similar to generating task execution times for a heterogeneous suite of machines [62]. An analogous method of generating load α values for a heterogeneous suite of customers is offered. Let μ_a be the desired average load α value for all loads, σ_a be the desired coefficient-of-variation of the load types, and σ_c be the desired coefficient-of-variation of the customers within a load type. For each load type k (given from the rows of Table 3.1), a Gamma distribution is sampled with mean μ_a and standard deviation σ_a to obtain the mean α value for load type k, denoted $\mu_{a,k}$. For each customer that owns load type k, obtain α_i by sampling a Gamma distribution with mean $\mu_{a,k}$ and standard deviation σ_c. This gives similar α values for each type of load, and thus similar customer behavior. This approach was taken because it is assumed that customers will act similar regarding the use of load types (e.g., more flexible with laundry, less flexible with the TV).

A parameter sweep was performed on the input values μ_a, σ_a, and σ_c. A representative result is shown in Section 3.5 using the inputs $\mu_a = 0.75$, $\sigma_a = 0.10$, and $\sigma_c = 0.05$. In general, the magnitude of the CIP is sensitive and positively correlated to μ_a (i.e., as μ_a increases, the CIP proportionally increases with respect to the real-time price). Values of σ_a and σ_c are positively correlated with the noise level of the CIP.

3.4.4.3. *Customer Loads.* Two types of loads are assumed to be available for each customer in this study: baseline and schedulable (smart) appliances. The baseline load is divided into thermal, modeled as air conditioning [63] and electric water heaters [64], and

TABLE 3.1. Schedulable Smart Appliances

Penetration (%)	**Mean power** (kW)	**Power std. dev.** (kW)	**Duration** (15-minute intervals)	**Start mean** (hour)	**Start std. dev.** (hour)
70	0.5	.05	4	7	1
70	0.5	.05	4	14	3
70	0.5	.05	4	17	1
50	0.75	.1	3	7	1
50	0.75	.1	3	14	3
50	0.75	.1	3	17	1
30	1.0	.2	2	7	1
30	1.0	.2	2	14	3
30	1.0	.2	2	17	1
100	0.25	.01	8	7	1
100	0.25	.01	8	14	3
100	0.25	.01	8	17	1
10	1.5	.3	2	7	1
10	1.5	.3	2	14	3
10	1.5	.3	2	17	1
80	0.4	.05	6	7	1
80	0.4	.05	6	14	3
80	0.4	.05	6	17	1

other non-schedulable loads. The non-schedulable loads are probabilistically generated for each customer based on the data in [46], given in Appendix A as Table A.1.

A probabilistic model for 18 generic schedulable appliance types is given in Table 3.1. The penetration level gives the probability that an appliance is present for a given customer; if it is present, the rated power of the appliance, as well as the start hour, is obtained from a normal distribution. Values in Table 3.1 were chosen so that the total load reflects actual energy use of an average household. Similar to the non-schedulable loads, a set of schedulable loads corresponding to each customer is generated probabilistically using the data in Table 3.1.

Each probabilistically generated load i has an associated availability window, (A_{i_start}, A_{i_dur}), that describes the time-window that the customer has allocated for load i to be scheduled.

Recall that t_{i_start} is the originally scheduled starting time for load i. Let $\mathcal{U}(\delta_i, 96)$ be a uniform random variable in the interval $[\delta_i, 96]$. In this study, to generate the availability window for each load i, an interval of duration $\mathcal{U}(\delta_i, 96)$ is generated around the starting time t_{i_start}. That is, $A_{i\_dur} = \mathcal{U}(\delta_i, 96)$ and $A_{i\_start} = t_{i\_start} - \frac{A_{i\_dur}}{2}$.

3.5. Results

A total of 56,642 schedulable loads (i.e., $I = 56,642$) from the 5,555 customers were randomly generated using the data from Table 3.1. The schedulable customer loads correspond to 11.2% of the total energy used by the 5,555 customers. To capture the algorithm in steady state, a two-hour window was added to the start and end of the simulation. Any appliance load that occurs within these windows did not contribute towards the objective function (i.e., only the 24-hour window was used for the objective function calculation). The genetic algorithm ran for 375,000 iterations before terminating, taking 113 minutes on an Intel i7 4900MQ processor running at 2.8 GHz using a C++ implementation in Ubuntu Linux. The final objective value, i.e., forecast aggregator profit, was $P = \$813.92$ (based on Fig. 3.5(a)). When evaluated for the *actual* real-time and spot market pricing, the schedule determined by the genetic algorithm resulted in an aggregator profit of \$947.90 (based on Fig. 3.5(b)). This increase in profit from forecast to actual is because the actual spot market pricing at the peak period was much larger than forecast (as shown in Fig. 3.5), leading to an increase in profit from the N component of the profit function. From a customer standpoint, the total savings of all 5,555 customers was \$460.31 and \$794.93 when using the forecast and actual data, respectively, for the 24-hour period under consideration. The increase in savings is also due to the large increase in peak real-time price that the customer no longer has to pay. For the settlement of the customer DR, the aggregator uses the actual spot market price data.

The total customer savings implies an average saving of \$0.14 per customer with a range of savings between \$0.02 and \$0.33. This range is indicative of the possible monetary benefits from the customer being more flexible with their loads (in the availability of the load and the customer α values) and bringing more energy (i.e., a greater number of assets) to the aggregator to participate in DR. Although the average daily savings may appear small, in this chapter we are focusing on the viability of the aggregator.[1] In general, the aggregator makes less profit and the customer saves more when the μ_a value is decreased, and vice-versa.

Fig. 3.6 shows the change in the load before and after the optimization occurs. Fig. 3.6(a) compares the system load before and after the optimization. As hypothesized, if the aggregator entity optimizes purely for economic reasons, the overall change in the system peak load may be beneficial, as is the case in this study. The aggregator-based residential DR program was able to reduce the peak of the participating 5,555 customers by 12.5%, resulting in a 2.66 MW reduction at 4:45 p.m. In Fig. 3.6(b), the portion of the load that is schedulable is shown. The area under the curve is 11.2% of the total system energy, with 19.4% of the total load reschedulable at the peak. This figure shows in greater resolution when the rescheduling of customer loads occurs. Over half of the schedulable load at the peak is moved off-peak. The reason this value is not higher is due to the customer availability windows described in Subsection 3.3.1.2. Because of this constraint, not all of the load can be moved to off-peak hours. Fig. 3.6(c) explicitly shows the difference in load between the system before and after the DR. The green shaded regions with the "/" hashing are the areas that the load was reduced, corresponding to the reduction in the peak. In the other areas, shaded red with "\" hashing, the load was increased, corresponding to the load moving to off-peak hours. The

[1]Monetary benefits, however, are not the only reason that early adopters may want to participate. It has been shown that often customers are motivated by altruistic reasons, such as environmental benefits (i.e., "being green") [65].

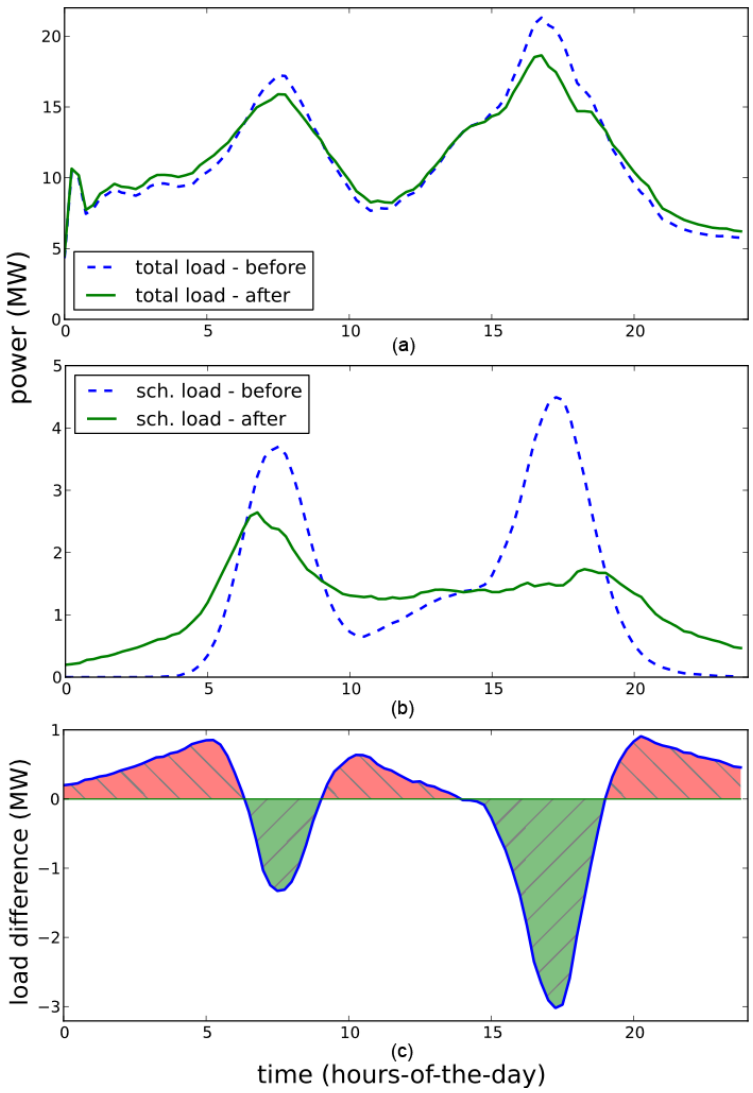


FIGURE 3.6. The change in load from before and after the aggregator demand response action. (a) The overall system load of the 5,555 customers. (b) The schedulable load. (c) The difference in load (i.e., after minus before).

large negative difference in load directly corresponds to the component of aggregator profit obtained by selling negative load, N, to the spot market. The positive difference in load is the portion of the load that contributes to the $S - B$ component of the aggregator profit function.

The customer pricing incentive obtained by the optimization, is shown in Fig. 3.7. Fig. 3.7(a) shows the incentive pricing compared to the forecast real-time and spot market prices. The CIP is lower than the forecast real-time price and, in general, the actual

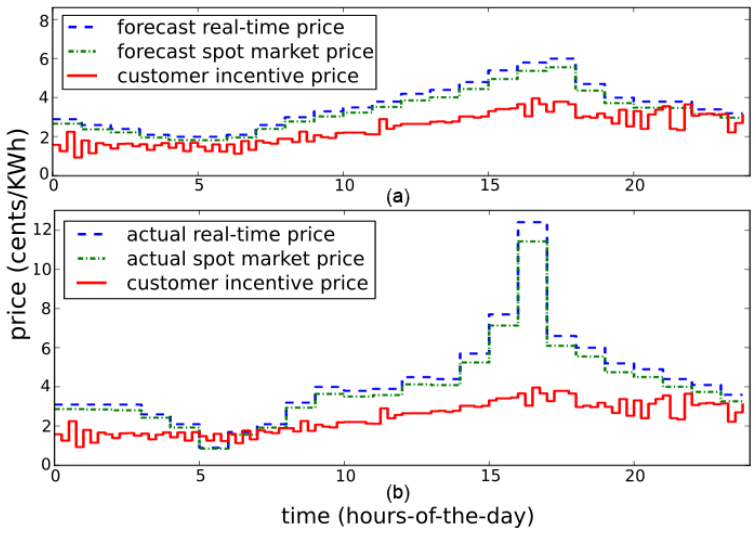


FIGURE 3.7. Real-time and spot market pricing compared to the customer incentive pricing. (a) The customer incentive pricing compared to the day-ahead forecast price. (b) The customer incentive pricing compared to the actual price.

real-time price (Fig. 3.7(b)). This indicates the customer receives a competitive, and reduced, rate of electricity for participating with the aggregator (including a hedge against the risk of large price spikes in the real-time price, such as at 4 p.m. in Fig. 3.7(b)).

To estimate the response of the spot market to the aggregator DR, a pseudo-spot-market-response is emulated. A sixth-order polynomial was fit to the PJM forecast spot market price for July 9, 2011 describing the spot market price (in cents/kWh) as a function of the load (in MW). The coefficients of the polynomial are given in Appendix B. The adjusted R^2 value for the polynomial to the data is 0.987, indicating a close fit. The l^2-norm of the difference between the price determined by the polynomial and the original forecast price is 0.516 cents/kWh. Note that this is a simple model for quantifying the changes the DR has on the market for July 9, 2011, and should not be generalized as a predictive tool.

The difference in system load due to the DR action (i.e., Fig. 3.6(c)) was added to the forecast PJM clearing load (available in [2]) and the polynomial fit was used to determine the resultant spot market price. The change in spot market price due to the DR is visualized

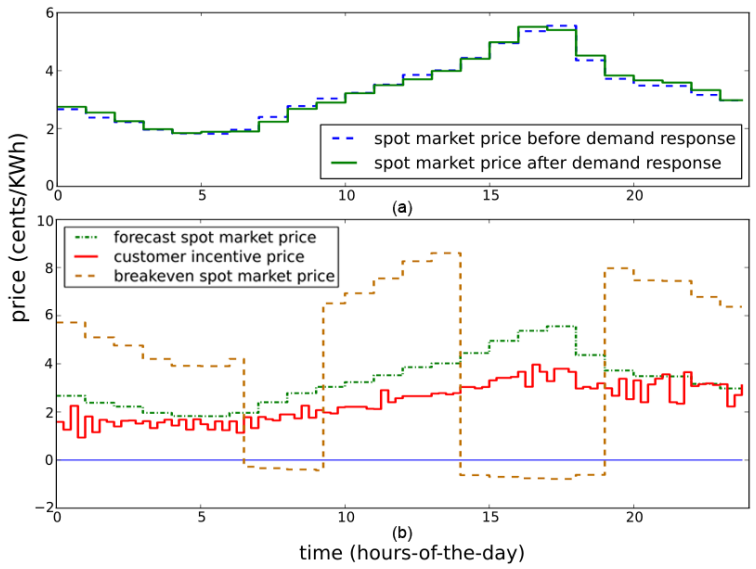


FIGURE 3.8. The results of the pseudo-market-response derived from the sixth-order polynomial regression model. (a) The predicted change in forecast spot market price as a result of the demand response. (b) The spot market price that will make aggregator break-even (no profit).

in Fig. 3.8(a). The l^2-norm of the difference between the emulated pseudo-spot-market-response and the forecast spot market price is 0.517 cents/kWh. When compared to the original l^2-norm of 0.516, this indicates the change in spot market price from the DR from *one* aggregator entity is small. However, when many aggregators exist within the purview of a single ISO, this market response will need to be investigated further.

To determine the breakeven point for the profit of the aggregator with the given DR, the forecast price was scaled until $P = \$0$. A scalar value, β, was applied in a positive manner at the times the load was increased and a negative manner at the times the load was decreased, the red and green areas in Fig. 3.6(c), respectively. This was done to increase the cost from the B term in the red shaded areas and to decrease the profit from the N term in the green shaded areas. The breakeven price was determined with $\beta = 1.141$, given in Fig. 3.8(b).

3.6. Conclusions

An aggregator-based residential demand response approach was proposed for scheduling residential customer assets. A customer incentive pricing (CIP) structure was proposed to compensate the customer for the inconvenience of rescheduling their assets. This new pricing structure gives the customer a near real-time choice of electricity supplier in a fully deregulated market scenario. A heuristic framework was designed to perform an optimization on the profit of the aggregator. To validate the heuristic framework, a system comprised of 5,555 customer households and 56,498 schedulable loads was simulated using a genetic algorithm implementation of the framework. The CIP found by the genetic algorithm was, in general, lower at all times than the customer would pay via real-time pricing. Despite this, the aggregator was able to make a profit by selling negative peak load to the spot market. This showed an example of optimizing for purely economical reasons in the form of aggregator profit, and enacting an overall change on the system peak load. This change benefits the customer of the aggregator (in the form of reduced cost of electricity for schedulable loads), the aggregator (in the form of a profit), and also those customers *not* participating with the aggregator (because the overall system peak is lowered as a common good).

CHAPTER 4

A Partially-Observable Markov Decision Process Approach to Residential Home Energy Management

4.1. Introduction

The United States Energy Information Administration predicts a 21% increase in residential electricity use from a 2012 reference case to the year 2040 [67]. Studies show that small and targeted reductions in peak demand can have large impacts on wholesale electricity prices [37]. Given that residential customers can account for over half of the system peak demand in summertime, such as in markets like the Electric Reliability Council of Texas (ERCOT) [68], residential demand response (DR) programs are attractive solutions for relieving the stress on the system and market.

Dynamic pricing programs are one such way to accomplish DR. These utility-offered programs, such as time-of-use (TOU) and real-time pricing (RTP), fluctuate the price of electricity throughout the day in accordance with system load levels to elicit a change in the consumption of electricity [69]. Residential customers can take advantage of these time-varying rates by changing electricity use to reduce their electricity bill. An automated method for changing electricity usage in response to time-varying price is a residential home energy management system (HEMS), a form of demand-side management (DSM). The challenges of an effective HEMS are (a) the uncertainty in the time-varying price of electricity, and (b) that as a customer, the benefit received from changing energy usage must exceed the

This work was performed jointly with the full list of co-authors available in [66].

inconvenience caused. To overcome these challenges and to maximize the benefit of the dynamic price, a HEMS is designed using a non-myopic sequential decision technique known as a partially observable Markov decision process (POMDP). The POMDP HEMS determines energy use at each point in time to minimize the electricity bill under the uncertainty of the time-varying price and customer comfort constraints.

The area of HEMS and DSM is an active research area [70, 71, 64, 72, 55, 73–79]. HEM has been approached through dynamic programming [72], stochastic optimization [55, 77], two-horizon algorithms [79], MILP [55, 73], convex programming with integer relaxation [75], and heuristic optimization [78, 34]. The HEMS optimize for cost [64, 55, 73, 75, 78, 79], user preference [64, 74], power caps [74, 77], and peak-to-average ratio [78]. The POMDP HEMS is a stochastic control process for making non-myopic decisions when the underlying state is uncertain. This delayed gratification approach to HEMS, combined with a continuously-updating RTP prediction method, results in significant cost savings in an RTP market. This work also presents a new method for accurately modeling appliance usage within a household. The considered time-period of one-month for quantifying results is also significantly longer than previous studies.

Markov decision processes (MDPs) have been used in other problems relating to power systems. [80] presents an MDP approach for commercial building energy management. Multiple energy systems (i.e., wind, photovoltaics (PV), combined cooling, heating, and power generation (CCHP), batteries) are jointly scheduled to match the load requirements of a commercial building to minimize cost. An MDP approach is used in [81] to schedule the use of residential pool pumps. The pool pump MDP is used to provide distributed ancillary service to the grid, such as self-supplied balancing reserves, without synchronization. A HEM unit plays a game with a central energy management unit that provides a dynamic price

to meet its energy needs within a desired budget in [82]. This work differs from previous power system MDP papers in that the *partially observable* framework is used and different decisions are made within the POMDP framework to exist at the purely customer-level in an existing RTP market. The POMDP HEMS presented here manages the use of flexible, non-interruptible smart appliances according to the definition in [55].

The primary contributions in this work are:

(a) The design of a non-myopic residential HEMS using a sequential decision technique (i.e., POMDP) that optimizes energy usage over a long time-horizon to minimize cost,

(b) The creation of a new appliance energy usage pattern based on queueing theory that models residential household usage for smart home simulations, and

(c) The comparison of the POMDP HEMS against three methods, including a new myopic algorithm, using month-long simulation studies.

The rest of the chapter is organized as follows. The system model and problem statement are described in Section 4.2. Section 4.3 introduces the design of the optimization methods. In Sections 4.4 and 4.5, the setup and results of the simulation study are presented. Concluding remarks are given in Section 4.6.

4.2. System Model

4.2.1. Overview. This work relates to changing the energy usage within a single residential household in response to dynamic pricing of electricity. The house exists in an RTP market, where the price for electricity varies every hour in response to demand. Flexible, non-interruptible smart appliances [55] are dynamically *arriving* (i.e., the residential customer wants to use the appliance) to be scheduled by the HEMS. The goal of the HEMS is

to minimize the total cost of electricity. Cyber-security (e.g., [83]) is a concern moving forward in the implementation of smart homes — and, more generally, the Internet-of-Things (IoT) — but it is out of the immediate scope of this work.

4.2.2. Appliance Model. For each appliance i that arrives at time t_{i-arr}, knowledge is assumed of the power rating in kW (p_i), the duration in hours (d_i), and the *start-time deadline* (t_{i-dead}). The start-time deadline is a customer provided input indicating when the latest the appliance can be started. This allows the customer to be more or less flexible depending on the specific appliance and the customer comfort. The goal of the HEMS is to find the start-time of each appliance ($t_{i-start}$).

For simulation purposes, t_{i-arr} must be determined for each appliance that accurately models residential energy usage. A novel probabilistic model is introduced for the usage pattern of a residential household based on queueing theory, specifically an $M_t/G/\infty$ queue. It is posited that a household can be modeled as an infinite computing server with appliance usage analogous to application arrivals. The $M_t/G/\infty$ queue states that applications arrive non-homogeneously with Markovian probability (i.e., Poisson distribution), generally distributed execution times, and infinite capacity [84]. In the smart home realm, the run-time of an appliance is analogous to the execution time of an appliance.

Let D be a random variable describing the duration of the set of appliances and $m(t)$ be the average number of running appliances at time t. The appliances arrive into the system according to a Poisson distribution with the time-varying rate $\lambda(t)$. Using the linear-with-time-shift (LIN-S) approximation of $m(t)$ from [84], the average number of appliances running in the household at time t is given as

$$m(t) = \lambda(t - \mathbb{E}[D])\mathbb{E}[D]. \tag{25}$$

To obtain the time varying rate of the Poisson distribution describing the appliance arrivals, solve for λ:

$$\lambda(t - \mathbb{E}[D]) = \frac{m(t)}{\mathbb{E}[D]}. \tag{26}$$

Because this is for simulation purposes, causality does not have to be assumed. Substitute the current time as $t \implies t + \mathbb{E}[D]$, then the required rate equation is given by

$$\lambda(t) = \frac{m(t + \mathbb{E}[D])}{\mathbb{E}[D]}. \tag{27}$$

To obtain a realistic number of simultaneously running appliances, $m(t)$ must be determined. Let L be a random variable describing the power rating of the set of appliances and $l(t)$ be the desired aggregate household load at time t. The average number of running appliances is then

$$m(t) = \frac{l(t)}{\mathbb{E}[L]}. \tag{28}$$

Substituting (28) into (27), the time-varying arrival rate of the appliances according to a Poisson distribution is determined by

$$\lambda(t) = \frac{l(t + \mathbb{E}[D])}{\mathbb{E}[L]\mathbb{E}[D]}. \tag{29}$$

An example generation of appliance arrivals is presented in Fig. 4.1. The generated usage pattern is very close to the desired usage pattern. The discrepancies can be explained because (a) the arriving appliances are a random process and inherently have some form of stochasticity, and (b) the LIN-S is an approximation of the average number of running

appliances and is not exact. The generated pattern is averaged over 500 samples to minimize the impact of (a).

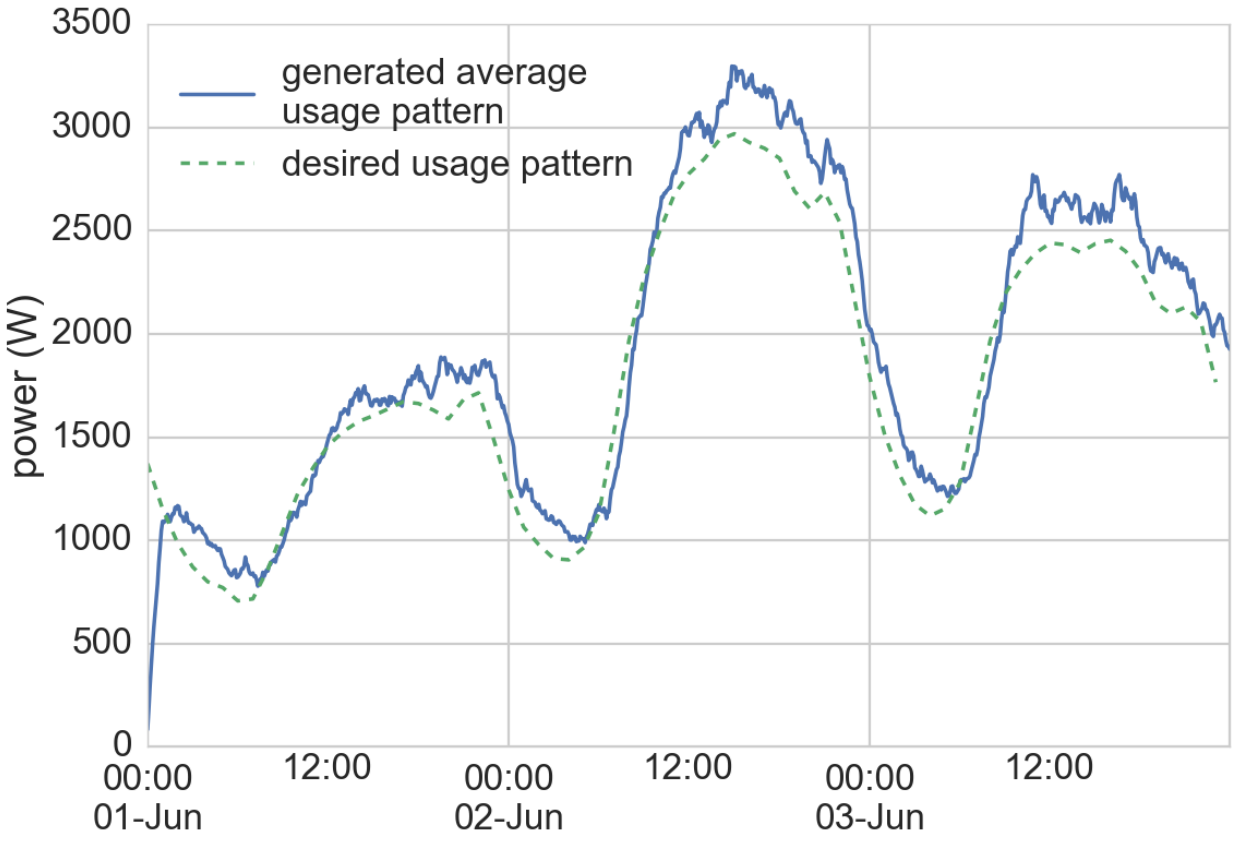


FIGURE 4.1. An example usage pattern generated by the $M_t/G/\infty$ queue model. The dashed green line represents the desired load ($l(t)$). The solid blue line is the generated usage pattern averaged over 500 samples.

4.2.3. REAL-TIME PRICING MODEL. The RTP market used in this study is modeled after the ComEd Residential RTP program [1]. In this market, the price of electricity changes every hour in response to the PJM real-time hourly market price. At approximately 4:30 p.m., a forecast for the next day's hourly prices are provided to the customer. At the start of each hour, the actual price of electricity for that hour is provided.

To better explain the market, a visual representation of the RTP market is given in Fig. 4.2. At 11:00 a.m. in Fig. 4.2, the price of electricity is known until noon. Additionally,

a forecast price is known for each hour until midnight. At 4:30 p.m., the price of electricity is still known until 5:00 p.m., but an additional 24-hours of forecast information is provided. To the left side of the red-dotted line in Fig. 4.2, the actual and forecast prices are much different. At time $(0, 1)$ and $(2, 3)$, the price of electricity is actually *negative*. This occurs when the demand for electricity is low compared to the available supply and it is cheaper for large generators to pay for the consumption of electricity than it is to shut the generator down and re-start it at a later time. The non-myopic HEMS can take advantage of this phenomenon to provide superior cost savings on the electricity bill.

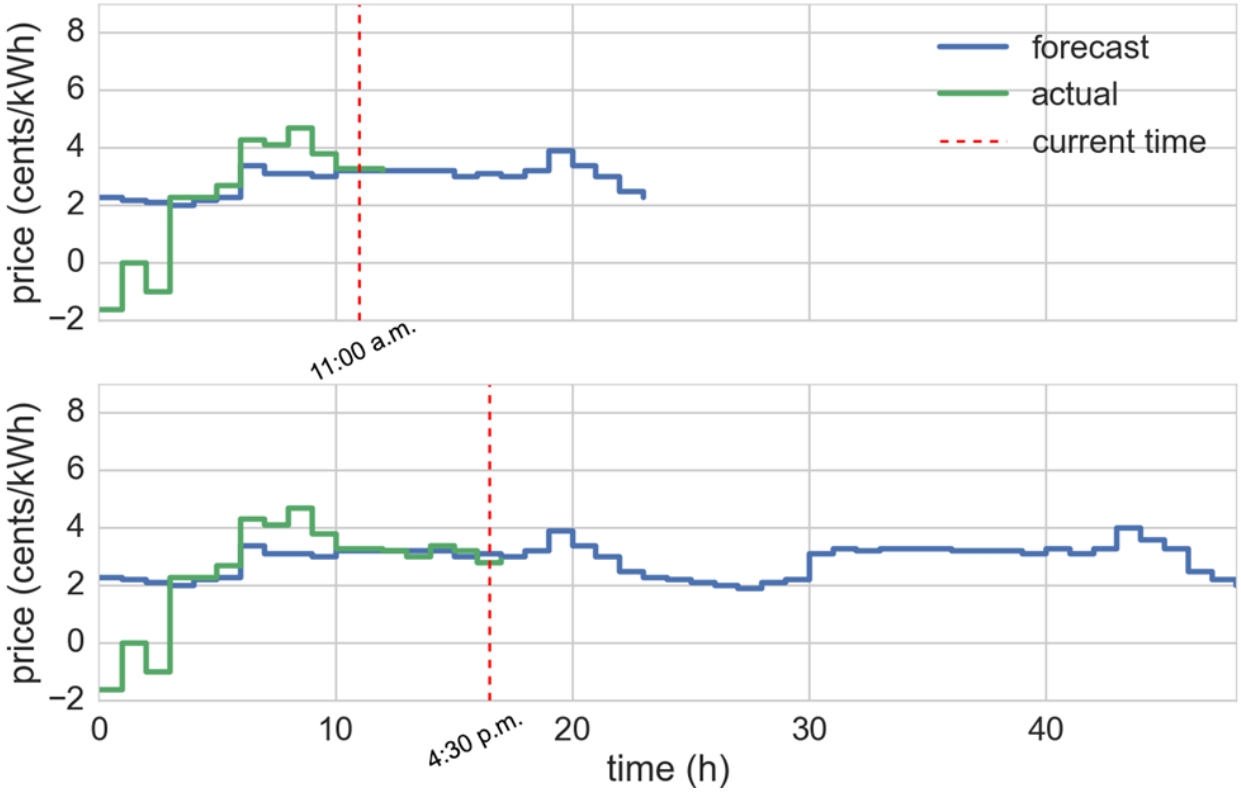


FIGURE 4.2. An example of the operations of the ComEd Residential RTP market [1].

At the current time t, let $c(t)$ be the cost of electricity (in cents/kWh) and $c_f(t, \tau)$ be the forecast price of electricity at time τ given the current forecast. The maximum forecast time,

τ_{max}, corresponds to the latest forecast provided by the utility (according to the operation in Fig. 4.2).

4.2.4. Problem Statement. In the residential house there is a vector of dynamically arriving appliances to be used by the customer at times $\hat{t}_{arr}$. Only information about appliances with an arrival time before the current time t is known. There is a corresponding vector with information about the start-time deadlines ($\hat{t}_{dead}$). Let the i^{th} element in the vectors correspond to the i^{th} appliance. The goal is to find the vector of start-times, $\hat{t}_{start}$, to minimize the total cost of using the appliances.

Let $C(\hat{t}_{start})$ be the cost to run appliances at the scheduled start-times, $\hat{t}_{start}$.

$$C(\hat{t}_{start}) = \sum_{i=1}^{|\hat{t}_{start}|} \int_{\hat{t}_{start}[i]}^{\hat{t}_{start}[i]+d_i} p_i c(t) dt \tag{30}$$

The formal problem statement is

$$\min_{\hat{t}_{start}} \quad C(\hat{t}_{start}) \tag{31}$$

s.t.

$$\hat{t}_{arr} \leq \hat{t}_{start} \leq \hat{t}_{dead}. \tag{32}$$

4.3. Optimization Methods

4.3.1. Overview. To compare the benefit of using a delayed gratification approach, a myopic comparison algorithm is designed, denoted minimum forecast cost (MFC). These new methods are evaluated against the status quo (appliances are used without regard to price) and the mathematical lower bound on cost.

4.3.2. IMMEDIATE. The status quo of energy usage within the household is to maximize personal comfort, often with little or no regard to price. In the simulation, this is analogous to using an appliance as soon as it arrives. The status quo optimization method, denoted *immediate* (imm), is then $\hat{t}_{start} = \hat{t}_{arr}$.

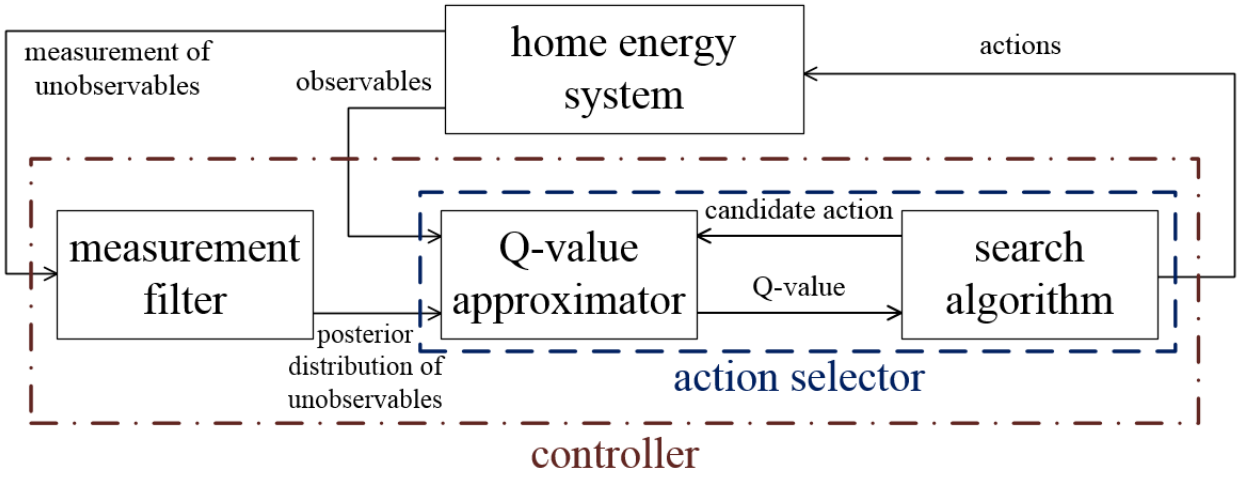


FIGURE 4.3. The general POMDP framework. The underlying belief state is split between *observables* and *unobservables*. The addition of unobservable portions of the state require the formulation of the POMDP, as opposed to an MDP. The unobservables are measured, and the conditional probability of their true state is determined with a measurement filter. The combination of the observable state and conditional probability of the unobservable state are used to select actions to take.

4.3.3. MINIMUM FORECAST COST. The obvious optimization approach is to run the appliances at the minimum forecast cost, as provided by the local utility. When an appliance i arrives at time t_{i-arr}, this optimization method, denoted MFC, schedules the appliance to run at the cheapest time over its duration, d_i, that satisfies $t_{i-start} \leq t_{i-dead}$. Let $C_i(t)$ be the cost of running appliance i at time t, given by (33).

$$C_i(t) = \int_t^{t+d_i} p_i c(t) dt \tag{33}$$

The *forecast* cost of running an appliance i at time t, if the current time is t_{cur}, is then:

$$C_{i-fore}(t) = \int_{t}^{t+d_i} p_i c_f(t_{cur}, t) dt. \tag{34}$$

The goal of the myopic MFC algorithm is, for each appliance i, to set $t_{i-sched}$ to the time that minimizes (34). Formally, MFC solves (35) for each appliance i.

$$t_{i-sched} = \underset{t}{\operatorname{argmin}}\, C_{i-fore}(t) \tag{35}$$

subject to

$$t_{cur} \leq t \leq \min\{\tau_{max} - d_i, t_{i-dead}\}. \tag{36}$$

4.3.4. Partially observable Markov Decision Process.

4.3.4.1. *Overview.* The POMDP optimization approach is adopted from [85] that is a non-myopic receding horizon control method that balances the trade-off between immediate knowledge and future performance (in this case, cost). By approaching the price of electricity as a stochastic input using historical information, the conditional probability (*posterior probability distribution*) of the future prices of electricity based on the current actual price and the error from the utility forecast can be determined.

To balance the trade-off between immediate and future decisions, Q-value approximation is used in the form of Bellman's equation [86]. For each appliance currently ready to run, the customer can take an action, a_i, from the set of possible actions A (i.e., $a_i \in A$). Let $\hat{a}$ be the vector consisting of the actions of the individual appliances (henceforth known as *the action*) to be determined by the HEMS, x be the current state, x' be the next state (after taking action $\hat{a}$), $R(x, \hat{a})$ be the immediate reward for taking action $\hat{a}$ in state x, and $V^*(x)$

be the optimal cumulative reward value over the time-horizon given an initial state x. The goal is to find the optimal action policy, $\pi^*(x)$, that maps states to actions to maximize the Q-value, $Q(x, \hat{a})$, given by

$$Q(x, \hat{a}) = R(x, \hat{a}) + \mathbb{E}[V^*(x')|x, \hat{a}]. \tag{37}$$

$\pi^*(x)$ is based on Bellman's principle [86], and given by

$$\pi^*(x) = \underset{\hat{a}}{\operatorname{argmax}}\, Q(x, \hat{a}). \tag{38}$$

The HEMS will take actions $\hat{a} = \pi^*(x)$ at each state x. A graphical overview of the POMDP framework is given in Fig. 4.3, adapted from [85].

In formulating the HEMS problem as a POMDP, there are two types of output from the home energy system: (a) the observables and (b) measurements of the unobservables. The measurements of the unobservables are filtered to determine the posterior distribution of the unobservables that, along with the observables, determine the *belief state*. At time t, let y_t represent the underlying state of the POMDP HEMS, $\hat{\Psi}_t$ be a vector of random variables describing the future prices of electricity, ϵ_t be the error between the utility forecast and current price of electricity, and H_t be the set of appliances ready to start. The underlying state is then $y_t = (c(t), \hat{\Psi}_t, \epsilon_t, H_t)$, where $\hat{\Psi}_t$ is unobservable. The measurement for future price, $\hat{\Psi}_t$, is the utility forecast price, $c_f(t, \tau)$, where $|\hat{\Psi}_t[\tau]| = \tau_{max}$. Given measurements $c_f(t, \tau)$, $P\big(\hat{\Psi}_t|c_f(t, \tau)\big)$ can be determined using a filtering method. Here, particle filtering [87] — a sequential Monte Carlo sampling method that maintains a set of representative samples (particles) — is used to calculate the posterior distribution.

The action set available for each appliance is to either run now or wait to run at a later time, i.e., $A = \{\text{run}, \text{wait}\}$. At each decision event at time t, $\hat{a}$ must be determined where $|\hat{a}| = |H_t|$ and each a_i corresponds to appliance $H_t[i]$, to maximize $Q(x, \hat{a})$. This decision is performed in the action selector of Fig. 4.3. The action selector is comprised of two parts: the Q-value approximator and a search algorithm. To approximate the Q-value, Monte Carlo sampling is used in conjunction with the particle filter and a rollout-based search algorithm.

4.3.4.2. *Particle Filter.* Two separate particle filters are used within the POMDP framework to create two distinct POMDP HEMS algorithms. The first, denoted *POMDP-Gauss*, keeps track of the mean and standard deviation of a Gaussian estimate of the error between the forecast and actual RTP at each hour of the day. The second, denoted *POMDP-GARCH*, keeps track of the parameters in an autoregressive (AR) process with a generalized autoregressive conditional heteroskedastic (GARCH) error process. Let the set of particles in the filter be denoted K.

For the POMDP-Gauss particle filter, an estimate can be constructed of the error from previous errors for each hour of the day. For each hour of the day h, there is an associated error $\delta_h \sim \mathcal{N}(\mu_h, \sigma_h)$, where $\mathcal{N}(\mu_h, \sigma_h)$ is a normal distribution with mean μ_h and standard deviation σ_h. To determine the initial mean and standard deviation, the sample mean and standard deviation from the previous N_d errors is calculated. Assuming the simulation start time t^0 starts at hour 0, and $h = t \mod 24$. At the current time t, given a measured observation $c_f(t, \tau)$, then each particle $\rho_i \in K$ is defined by

$$\rho_i = c_f(t, \tau) + \delta_h. \tag{39}$$

Assuming each particle has the same weight, the particle samples are used to estimate the conditional probability of the price $c(t + \tau)$ given $c_f(t, \tau)$.

The second filter, POMDP-GARCH, an AR process with GARCH error is used to model the error in RTP. The GARCH model has been used previously to estimate high variability in power markets [88]. In an AR process, the current output depends on a linear combination of prior values, plus some error term. At time t, let $c_{ar}(t)$ be the cost output of the AR process, k be the AR constant, $c_{ar}(t-i)$ be the i^{th} previous output, γ_i be the coefficient corresponding to $c_{ar}(t-i)$, m be the number of modeled coefficients, and ϵ_{t-ar} be the error. The AR process is given as

$$c_{ar}(t) = k + \sum_{i=1}^{m} \big(\gamma_i c_{ar}(t-i)\big) + \epsilon_{t-ar}. \tag{40}$$

The error in (40) is modeled as a GARCH process. The GARCH process is a specialized AR process that has a conditional variance based on prior inputs. At time t, let σ_t be the standard deviation and z_t be a Gaussian random variable with mean 0 and standard deviation of 1. The GARCH error process is given as

$$\epsilon_{t-ar} = \sigma_t z_t. \tag{41}$$

The term σ_t is itself a linear combination of prior inputs, hence the conditional heteroskedasticity of the GARCH process. At time t, let χ be the GARCH constant, σ^2_{t-i} be the i^{th} previous variance, ϕ_i be the coefficient corresponding to σ^2_{t-i}, P be the number of GARCH terms (i.e., prior variances), ϵ^2_{t-j} be the j^{th} previous square-error, q_j be the coefficient corresponding to ϵ^2_{t-j}, and Q be the number of ARCH terms (i.e., prior square-errors). A GARCH(P,Q) process is fully described by (41) and

$$\sigma_t^2 = \chi + \sum_{i=1}^{P} \phi_i \sigma^2_{t-i} + \sum_{j=1}^{Q} q_i \epsilon^2_{t-j}. \tag{42}$$

The AR + GARCH process to be used in the POMDP-GARCH is obtained by substituting (41) into (40), resulting as

$$c_{ar}(t) = k + \sum_{i=1}^{m} \left(\gamma_i c_{ar}(t-i)\right) + \sigma_t z_t. \tag{43}$$

At time t, the each particle $\rho_i \in K$ of the particle filter corresponding to POMDP-GARCH has its own list of the prior m AR outputs, P error variances, and Q square-errors. It uses this to determine the conditional probability of the next τ_{max} prices given $c_f(t, \tau)$.

4.3.4.3. *Action Selector.* The action selector, shown as the red dotted box in Fig. 4.3, consists of an estimate of the Q-value of taking a set of actions and an optimization algorithm that maximizes this value. The Q-function maps an action at a belief state to the Q-value. The HEMS problem is relatively unique in that an accurate cost of taking a given action is known and that the actions may be evaluated independently for their cost, reducing the search space at each belief state. Recall that (37) consists of the summation of the *immediate* reward for taking an action, $R(x, \hat{a})$, and the *expected future* reward, $\mathbb{E}[V^*(x')|x, \hat{a}]$ (also known as the expected reward-to-go). These map directly to the action set, $A = \{\text{run}, \text{wait}\}$.

Because each appliance action can be evaluated individually, the maximum number of evaluations is reduced from $|H|^2$ to $2|H|$. At state x, there is an estimate of the price of electricity through τ_{max}, denoted $c'(t), t_{cur} \leq t \leq \tau_{max}$. The cost of running appliance i at time t using $c'(t)$ is:

$$C'_i(t) = \int_t^{t+d_i} p_i c'(t) dt \tag{44}$$

To map this state x to a Q-value given an action a_i corresponding to appliance i,

$$Q_i(x, a_i) = \begin{cases} -C_i'(t_{cur}) & a_i = \text{run} \\ \max_t -C_i'(t) \quad t > t_{cur} & a_i = \text{wait} \end{cases}. \tag{45}$$

The costs of running the appliances are negated because the HEMS is a cost minimization problem. The total Q-value determined from the Q-function is

$$Q(x, \hat{a}) = \sum_{i=1}^{|H|} Q_i(x, \hat{a}[i]). \tag{46}$$

Note that the Q-value in (46) captures the effect of the current action $\hat{a}$ on the future of the system through (44). Recall from Bellman's principle that the future effect represented by the expected future reward depends on the optimal policy. Because the optimal policy is unknown, the usual approach is to approximate the Q-value. To approximate the Q-value, the method denoted *policy rollout* [85, 89] is used. In policy rollout, the optimal future reward V^* is replaced in (37) with the future reward associated with a base policy $V_{\pi base}$. This gives the rollout-approximate Q-value

$$Q_{\pi base}(x, \hat{a}) = R(x, \hat{a}) + \mathbb{E}[V_{\pi base}(x')|x, \hat{a}]. \tag{47}$$

This gives the rollout policy

$$\pi(x) = \underset{\hat{a}}{\text{argmax}}\, Q_{\pi base}(x, \hat{a}). \tag{48}$$

To compute the output of this policy, Monte Carlo sampling is used to estimate the expected future reward, $\mathbb{E}[V_{\pi base}(x')|x, \hat{a}]$, which works well with the output of the particle filter, where each particle represents a single sample of the unobservable state. For each appliance i, the

action a_i is determined at time t_{cur} as

$$a_i = \underset{A}{\operatorname{argmax}}\, Q_i(x, a_i) \tag{49}$$

where $C'_{i,j}(t)$ is the sample cost of appliance i in particle j, and

$$Q_i(x, a_i) = \begin{cases} |K|^{-1} \sum_{j=1}^{|K|} -C'_{i,j}(t_{cur}), & a_i = \text{run} \\ |K|^{-1} \sum_{j=1}^{|K|} \max_t -C'_{i,j}(t), & a_i = \text{wait}, \end{cases} \tag{50}$$

$$t_{cur} \leq t \leq \min\{\tau_{max} - d_i, t_{i-dead}\}.$$

The complete POMDP HEMS controller is given in Fig. 4.4.

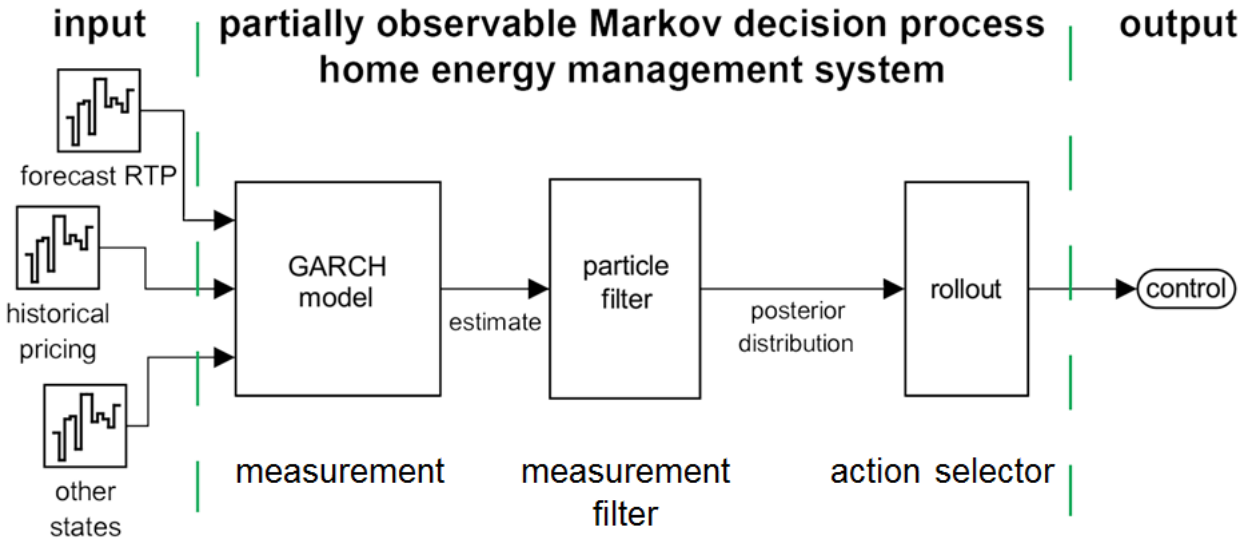


FIGURE 4.4. The POMDP-GARCH HEMS. The measurements of the unobservable state are provided by the utility-forecast cost. The GARCH process combined with the particle filter provides an estimate of the actual RTP. Along with the appliances ready-to-run, the output of the particle filter are used to determine which action to take for each appliance. The GARCH block can be replaced with the Gaussian-noise estimate to obtain the POMDP-Gauss HEMS.

4.3.5. Lower Bound. The lower bound (LB) can be calculated similarly to the MFC and POMDP rollout method in the previous subsections. If absolute knowledge of the system is assumed, i.e., the actual cost is known for each time interval of interest, then optimal scheduling decisions can be made with regards to cost minimization. The LB is obtained by scheduling each appliance i at the absolute minimum cost time given by

$$t_{i-sched} = \underset{t}{\operatorname{argmin}}\, C_i(t) \tag{51}$$

s.t.

$$t_{cur} \leq t \leq t_{i-dead}. \tag{52}$$

Note the difference in the scheduling constraint between (36) and (52). The MFC can only make decisions up to the forecast information provided by the utility where the LB assumes perfect information of the RTP up to each appliance start-time deadline, providing mathematical lower bound.

4.4. Simulation Setup

4.4.1. Overview. This section describes the models, input data, and simulation parameters used. The methods introduced up to this point can be used on any inputs for a HEMS, including dynamic pricing markets. To generalize the above methods for time-varying loads, p_i can be replaced by $p_i(t)$ in (30), (33), (34), and (44).

4.4.2. Appliance Types. For simulation purposes, abstract appliance types are modeled. Let N_a be the number of appliance types. Each appliance *type* j has a power rating in kW (p_j) and a random variable modeling the duration in hours (ϕ_j). The reason that the

duration is stochastic is that, although appliance run times may be similar between uses, the run time of an appliance usually has some variance. To generate p_j and ϕ_j for each appliance type, a Gaussian distribution and the coefficient-of-variation-based (CVB [34, 62]) method are used, respectively. Let $p_j \sim \mathcal{N}(\mu_p, \sigma_p)$, where μ_p and σ_p are the mean and standard deviation, in kW, of the distribution of appliance power ratings. Let $\phi_j \sim \mathcal{G}(\mu_{j-t}, \theta_t)$ where $\mathcal{G}(\mu_{j-t}, \theta_t)$ is a Gamma distribution with mean μ_{j-t} and coefficient-of-variation θ_t (where the relationship to the shape is θ_t^{-2} and the scale is $\mu_{j-t}\theta_t^2$). μ_{j-t} is itself generated by a Gamma distribution, $\mathcal{G}(\mu_d, \theta_d)$. The parameters N_a, μ_p, σ_p, θ_t, μ_d, and θ_d are determined empirically, and given in Appendix C.

4.4.3. Household Usage Pattern. Recall from (29) in Section 4.2.2 that the $M_t/G/\infty$ queue requires the desired load curve for the household, $l(t)$, as an input. To obtain a time-varying household load curve that accurately models the daily, weekly, and seasonal change in energy usage, the ComEd system load is scaled to match the load of a single household. Let $l_{sys}(t)$ be the ComEd system load at time t, obtained from [90]. Let $f(l)$ be a function that scales $l_{sys}(t)$ to $l(t)$, where $l(t) = f(l_{sys}(t))$. The function $f(l)$ is comprised of two parts: (a) normalizing the system load to $[0, 1]$, and (b) upscaling the normalized load to the desired $l(t)$. Let $S_{scale}(l)$ be the normalizing function for a given load l, given by

$$S_{scale}(l) = \frac{l - \min_t l_{sys}(t)}{\max_t l_{sys}(t) - \min_t l_{sys}(t)}. \tag{53}$$

Let o_{min} and o_{max} be the minimum and maximum household usage, in kW, respectively. The scaling function is then defined as $f(l) = o_{min} + S_{scale}(l)(o_{max} - o_{min})$. Defining the previous equation in terms of household load through time, $l(t) = o_{min} + S_{scale}(l_{sys}(t))(o_{max} - o_{min})$.

To obtain specific appliance arrivals from the appliance types and the $M_t/G/\infty$ queue, at a given time t, the next appliance arrival, i, is sampled from a Poisson distribution with rate $\lambda(t)$ to obtain t_{next}. This gives appliance i an arrival time of $t_{i-start} = t + t_{next}$. The appliance type j corresponding the the arrived appliance i is selected randomly from the N_a types with equal probability. This sets $p_i = p_j$ and d_i to a sample of ϕ_j. The start-time deadline, t_{i-dead}, is set by randomly sampling from the set of $t_{i-start} + \{1\text{h}, 2\text{h}, 4\text{h}, 8\text{h}\}$. The varying length of the deadlines represents the flexibility the customer may have with a given appliance.

4.4.4. Minimum Cost Time. Many of the proposed optimization techniques require scheduling an appliance to run at the minimum time. Because the price of electricity is constant per hour and the load of an appliance is constant, the search space can be reduced for determining the minimum cost time. If the run-time of an appliance is less than an hour, it is trivial to calculate the minimum cost. Let d_{res} be the residual duration of an appliance greater than the highest hour (i.e., for an appliance running 2 hours 20 minutes, $d_{res} = 20$ minutes). The scheduling times of interest are then t_{cur}, each hour h, and d_{res} before each hour for each h less than the appliance start-time deadline. This reduces the number of possible scheduling times from a continuous t to a small set of possible start-times.

4.4.5. Simulation Scenarios. For the simulation study, scenarios were chosen that represent one month of electricity usage to obtain a monthly electricity bill. The actual and forecast pricing data from ComEd [1] between 2007 and 2013 was used. The GARCH model was trained on the errors in 2007, the parameters can be found in Appendix D. For the POMDP-Gauss model, $N_d = 60$. Both particle filters used 1000 particles to estimate the posterior distribution of unobservables.

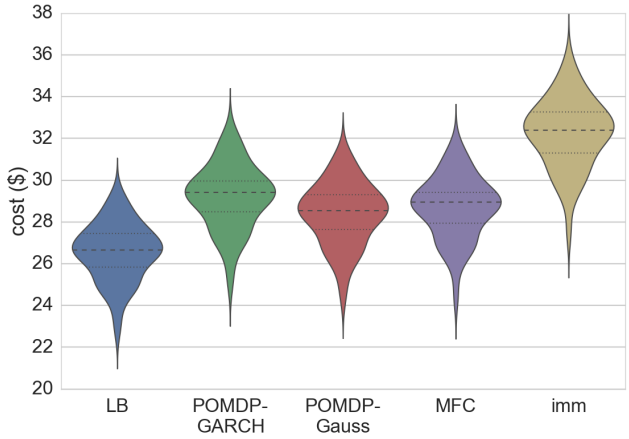
Three scenarios were chosen, each representing one month in the ComEd system. The scenarios (A, B, and C) correspond to the minimum, median, and maximum months in the input data in terms of root-mean-square (RMS) error between the forecast and actual price and are summarized in Table 4.1. Each scenario was run for 50 trials. For a given scenario, the actual and forecast RTP (from the ComEd data) and the scaled household load stay the same. Between trials, appliance types and appliance arrivals differ.

TABLE 4.1. Simulation Scenarios

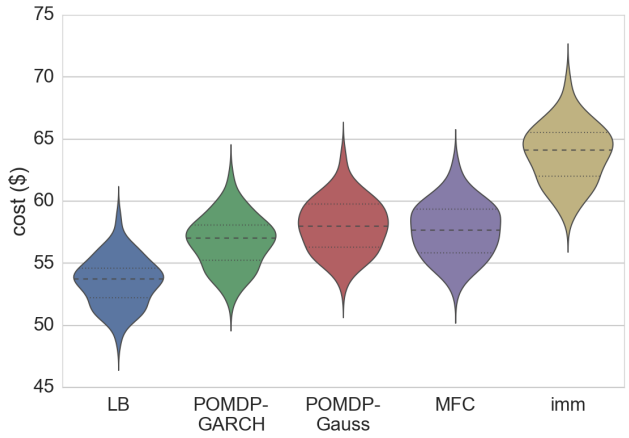
Name	**Month**	**RMS Error** (cents)	**Min. Price** (cents/kWh)	**Avg. Price** (cents/kWh)	**Max. Price** (cents/kWh)
A	Oct. 2009	0.72	-1.6	3.00	8.1
B	Jan. 2011	1.46	-7.5	3.93	20.7
C	June 2008	4.22	-21.1	5.94	48.7

4.5. Simulation Results

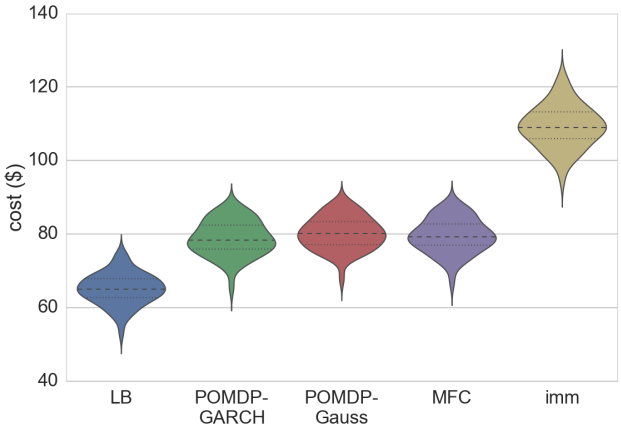
The different HEMS algorithms were evaluated against the three scenarios. Before conducting the simulations, it is expected that the POMDP methods to improve over the myopic MFC in situations where the forecast and actual price diverge (as in scenario C). Table 4.2 compares the average price of electricity paid by the customer during the scenario time periods between the different HEMS methods. The three intelligent HEMS methods are bounded by the LB and status quo. The MFC performs slightly better than POMDP-GARCH in scenario A, but slightly worse than POMDP-Gauss. The GARCH technique is well suited to signals with high variance, leading to a slightly worse prediction in the non-variable scenario A. The POMDP-Gauss method is able to take advantage of the delayed gratification over the myopic MFC. In general, as the RMS-error of the RTP increases, the POMDP-GARCH



(A)

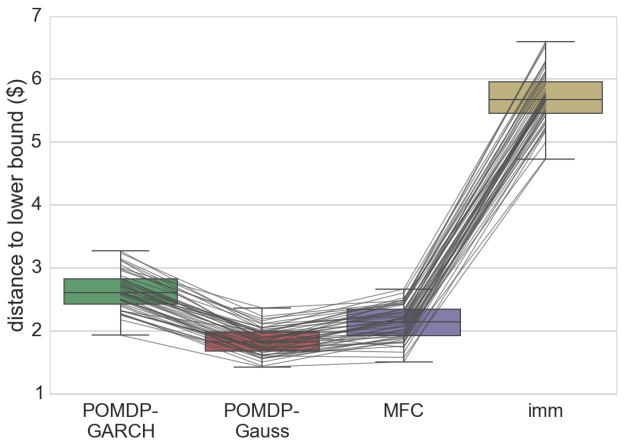


(B)

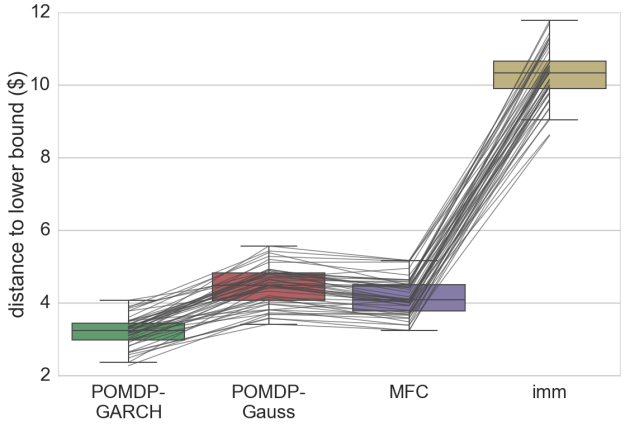


(C)

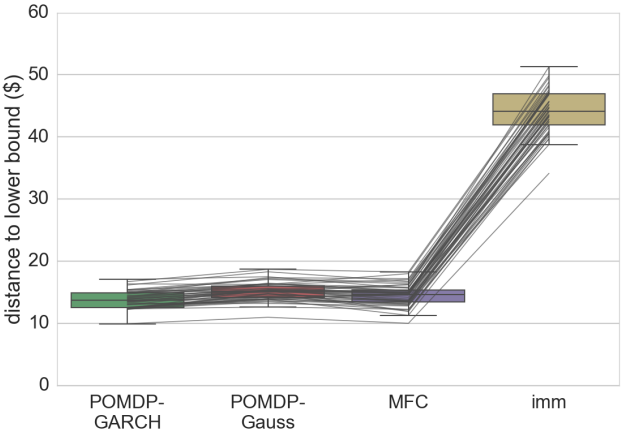
FIGURE 4.5. A comparison of the monthly electricity bills for each optimization method for scenarios A, B, and C, respectively. The violin plots show the probability density of each cost. The dashed line corresponds to the median cost and the dotted lines correspond to the quartiles.



(A)



(B)



(C)

FIGURE 4.6. A comparison of the distance to the lower bound for scenarios A, B, and C, respectively. The box plot shows the median, quartiles, and 10th percentiles. The lines between the box plots connect individual trials.

performs better relative to the other techniques. As the variance of the RTP signal increases, the Gaussian assumption made by POMDP-Gauss begins to degrade performance. It is important to note that *all* methods, including the status quo, resulted in a lower per-kWh price than the tariff-based ComEd pricing (∼7.5 cents/kWh).

TABLE 4.2. Average Price of Electricity (cents/kWh)

Name	**Energy** (kWh)	**LB**	**POMDP-GARCH**	**POMDP-Gauss**	**MFC**	**imm**
A	1037	2.56	2.82	2.74	2.77	3.10
B	1576	3.40	3.60	3.69	3.67	4.04
C	1524	4.27	5.18	5.26	5.22	7.14

The monthly electricity bills across scenario A, B, and C are presented in Figs. 4.5(a), 4.5(b), and 4.5(c), respectively. Each violin plot shows the probability distribution of obtaining a specific monthly electricity bill (the wider the figure, the higher the probability). The dashed line indicates the median cost and the dotted lines indicate the quartiles. The average monthly electricity bill on the tariff-based rate is $77.78, $118.20, and $114.30 for scenarios A, B, and C, respectively. As a customer, just opting-in to the RTP program with no change in energy usage offers the potential for significant savings. By combining an RTP program with a smart HEMS, the savings are improved. In general, the shapes of the distributions and coefficient-of-variation between the methods within the same scenario are similar, indicating a strong correlation between an individual trial and the monthly cost (within a trial, the appliance arrival patterns are the same and the overall energy usage is the same). This trial-cost correlation explains the large overlap between some of the distributions. The relative performance of the methods to the lower bound are presented in Fig. 4.6 where each box plot shows the median and quartiles, and the whiskers show the 10th percentile. The

lines between the optimization methods connect the individual trials. In general, the relative performance of a trial is similar between the methods (e.g., the median trial for one method is close to the median trial for another).

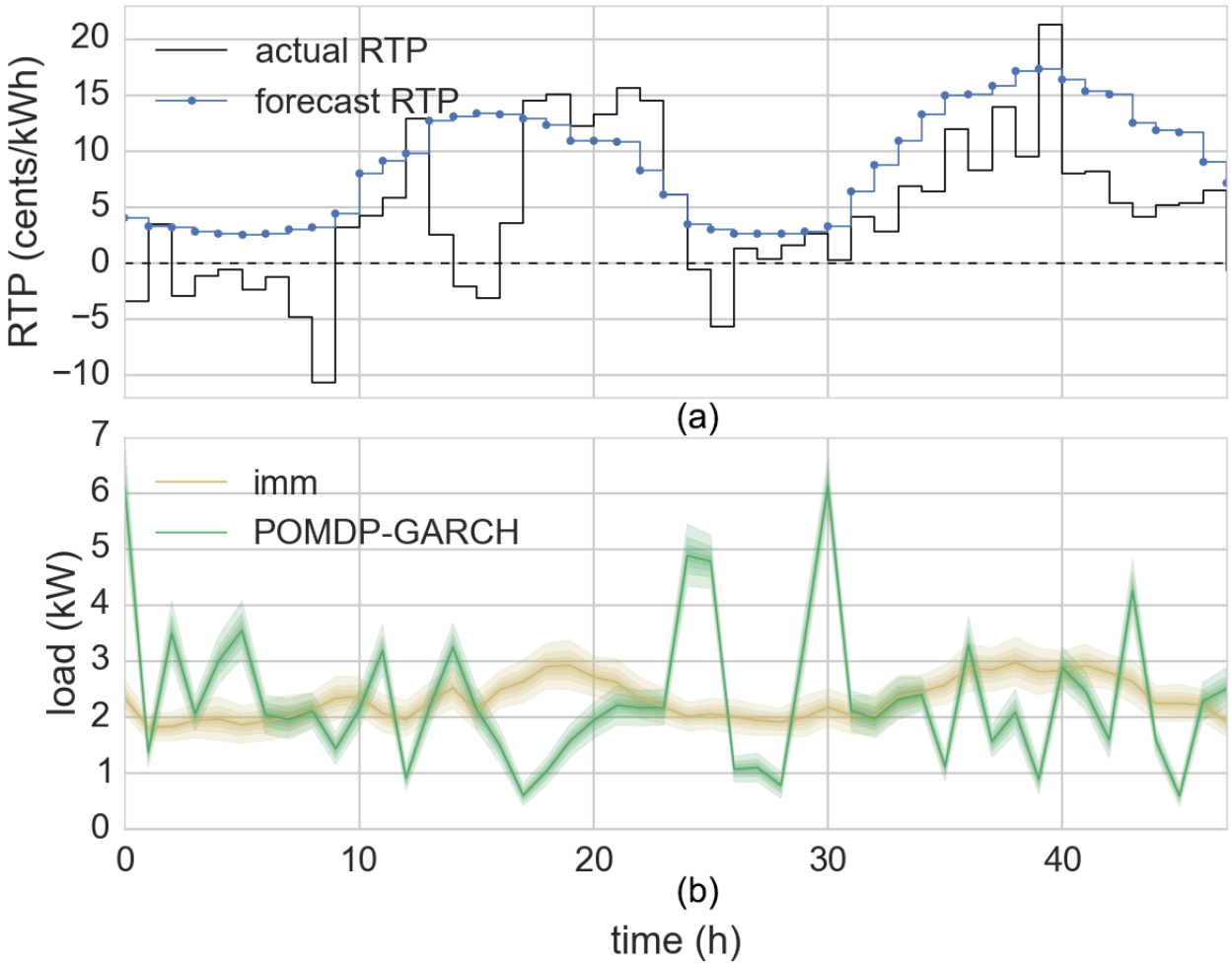


FIGURE 4.7. (a) A 48-hour sample of the actual and forecast RTP. (b) Statistical time-series of the household load compared between the immediate and POMDP-GARCH methods.

The change in energy usage in the household between the status quo and the POMDP-GARCH HEMS is presented in Fig. 4.7. Fig. 4.7(a) shows the difference between the actual and forecast RTP for a 48-hour period in scenario C. The curves in Fig. 4.7(b) present the HEMS response to the pricing signals. The cloud around the plot represents the statistical differences between trials. A drastic change in electricity usage occurs when switching to

a HEMS algorithm–the POMDP-GARCH in this case–from the status quo. The POMDP-GARCH HEMS actively prevents using appliances during high-price times, but also predicts low-price times, such as at $t = 25$ hours. It is interesting to note that the new peak is *worse* than the old peak. This is not a problem for a single 5 kW household, but if many households within a given area act in a similar way it may cause problems such as the rebound effect [15].

4.6. Conclusions

Reducing the peak demand of the electric power system provides benefits by reducing the cost of electricity by lowering the number of expensive generators needed. By reducing the peak, the capacity factor of dirty diesel-fired peaking generators can be reduced. Lastly, as peak demand increases, the available transmission capacity will also need to increase. By reducing the peak demand, building new transmission lines can be deferred; a costly, long-term project. Utilities are offering real-time pricing programs, passing ISO prices to customers, such as the ComEd residential real-time pricing program. Customers can take advantage of this real-time pricing to drastically reduce their monthly electric bill.

The partially observable Markov decision process is a promising non-myopic method for home energy management. On the high-end, the POMDP-GARCH HEMS resulted in a $30 cost savings over the status quo for June 2008. Even for more modest months, savings of 10% were shown. It should be noted that these large savings were obtained just using flexible appliances. As more asset types with more capabilities, such as electric vehicles and HVAC, are added, the savings can be expected to increase.

CHAPTER 5

A Visualization Aid for Demand Response Studies in the Smart Grid

5.1. Introduction

Each day, power systems engineers are inundated with information. As Smart Grid technologies, such as smart meters and DR programs, evolve and become more widely implemented, the amount of data available will increase drastically [92]. The abundance of data and information available makes it difficult to assess the state of power systems in a fast and user-friendly manner [93]. In the field of visualization, it is known that as the amount of data increases, it becomes more difficult to sift through the data to find key information and present it clearly [94]. By using proper information visualization techniques, it becomes easier for humans to recognize patterns and analyze information [95]. These facts, in part, led to the United States Department of Energy to recognize the importance of visualization in power systems [96]. This chapter attempts to address these issues in the form of new visualization techniques for DR programs.

The economic benefit of small reductions in peak load through DR programs is well understood [37, 97]. One method of DR that is currently being researched is an aggregator-based method that implements centralized control of many smaller-rated loads whose aggregation enables a noticeable change on the load profile of the power system [48, 49, 34]. We use data

This work was performed jointly with the full list of co-authors available in [91]. This work was supported by the National Science Foundation under grant numbers CNS-0905399 and CCF-1302693, and by the CSU George T. Abell Endowment. The datasets, example Python and Matlab GIS graphing code, and color-blind-friendly figures may be found at http://www.engr.colostate.edu/sgra/

involving the DR action of over 65,000 aggregated small-rated appliances from [34] to design new visualization techniques. The visualization techniques enable a better understanding of the effectiveness of the aggregator-based DR method. The effect of DR programs are commonly shown as the difference in load curves [34, 45]. A standard representation of a DR load curve is given in Figs. 5.1(a) and 5.2(a). We believe that this presentation of the data has some limitations, leading to our objective to develop user-friendly visualization tools for quantifying and comparing the effectiveness, the profitability, and the schedule for a given set of solutions to a demand response problem both temporally and spatially.

In the past, work on power system visualization has focused on "quick-look" information about the system in the form of graphic overlays on one-line diagrams. This includes spatial contours for visualizing power system voltage data [93], contingency analysis data [98], market power assessment [99], and power flows [100, 101]. Other research on spatial visualizations of power systems has occured using geographic information systems (**GIS**). In [102], the contours of real-time voltage magnitude and phase angle measurements, obtained from phasor measurement units, were overlaid on top of the contiguous United States. A neighborhood of photovoltaic (**PV**) arrays in Anatolia, California, and their real-time output were graphically overlaid on the Google Earth geographic area of Anatolia in [103]. The power-flow software PowerWorld allows the mapping of voltage contours and network flow information onto maps and exporting this to Google Earth. As far as I am are aware, no work has occurred to provide the same type of easy-to-understand information for DR programs. This makes it difficult to understand the large quantities of data created by DR programs to determine the effectiveness of the response. With regard to policymakers and other non-technical entities, it becomes more difficult to clearly explain the benefits of DR programs. In that regard, three new visualization techniques have been designed that, at a

glance, can provide more information about a set of DR solutions to both system operators and non-technical entities. The first two new techniques provide another temporal dimension of information without obfuscating the graph. The third technique adds a spatial component using GIS in addition to the temporal dimension.

The following contributions are made in this chapter:

(a) The design and description of two new temporal visualization techniques for a given set of solutions to a demand response problem that answer the following:
   (i) Did the demand response plan work effectively?
   (ii) When did the demand response entity (aggregator in this work) make a profit or loss?
   (iii) How does the difference between the forecast and actual price of the spot market affect the profit margin of the demand response entity?

(b) The creation of a spatial visualization of demand response using GIS that answers *where* in the distribution system did the demand response plan affect load.

(c) A discussion of how the visualization methods can be used to analyze the effectiveness of demand response optimization techniques.

The remainder of the chapter is organized as follows. The system model and associated data are provided in Section 5.2. In Section 5.3, the visualization techniques are described and analyzed. Concluding remarks are discussed in Section 5.4.

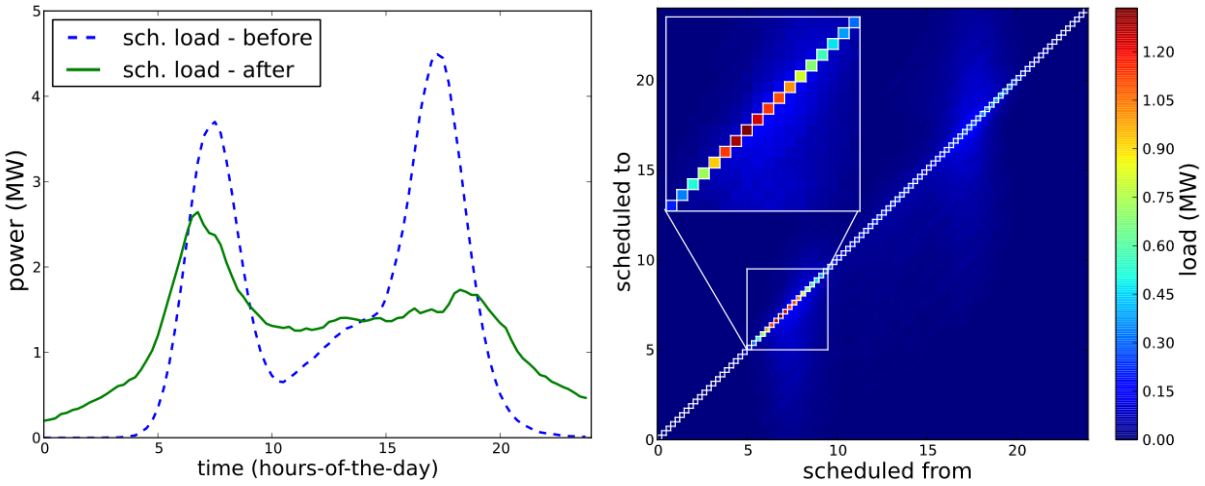
5.2. System Model

According to [47], by increasing DR technologies and directly allowing retail customers to access the wholesale market, the price elasticity of demand may increase, leading to an increased level of volatility in power systems. An aggregator, which is an intermediary entity

that offers the coordination of many entities centrally [48, 34], is an alternative to uncontrolled DR [34]. This chapter uses the same system model as Chapter 3 and is summarized below.

The work in Chapter 3 directly addresses the concerns of the CEC where the aggregator is presented as a for-profit entity that coordinates the schedule of a set of smart appliances belonging to a set of customers and brings the aggregated DR to the market. To encourage customer participation with the aggregator and to offset the inconvenience of a rescheduled appliance, a *customer incentive pricing* structure was proposed. The customer incentive price is a time-variant electricity rate structure that offers the customer a competitive rate of electricity, as determined by the aggregator, for those appliances they allow to be rescheduled as part of the DR action. If the rate is not worth the inconvenience of rescheduling the appliance, the customer is allowed to refuse the aggregator and instead pay the utility company real-time price for electricity (i.e., the status quo) at each time interval for each appliance in the DR. The aggregator-based residential DR program, denoted as Smart Grid Resource Allocation in [34], is formulated as an optimization problem where the objective function is to maximize the aggregator profit. The decisions the aggregator can make in order to maximize profit are the customer incentive pricing and the smart appliance schedule.

Recall from Chapter 3 that $\boldsymbol{N}$ is the *income received* by the aggregator for selling negative load to the spot market at the times the smart appliances were *rescheduled from*, $\boldsymbol{S}$ is the *income received* by the aggregator for selling electricity to the customer at the customer incentive price at the times the smart appliances were *scheduled to*, and $\boldsymbol{B}$ is the *cost paid* by the aggregator for buying electricity from the spot market at the times the smart appliances



(A) (B)

FIGURE 5.1. (a) Load curves of the schedulable load before and after the DR action in the system in the constrained case. A significant portion of the peak load is moved to off-peak hours in a valley-filling manner. This is a common way to present DR results. (b) A heat map showing the temporal source and destination of the schedulable load in the constrained case. The color of a given point at position (x, y) indicates the load moved from time x to time y, as represented by the accompanying color bar. The white box diagonal, i.e., $x = y$, indicates the amount of load that was not rescheduled. If you sum all loads at a given x-value across all y in (b), the total load will equal the load at the same x-value on the blue dotted line in (a). Similarly, the sum of load across all x will equal the green solid line. To highlight the fact that the magnitude of the load that is *not moved* from time 5-9 is much greater than the typical amount of load *moved* from any other time x to time y in (b), it is shown in greater detail.

were *scheduled to*. The aggregator profit, denoted as $\boldsymbol{P}$, is given by Eqs. 19 and 54.

$$\boldsymbol{P} = \boldsymbol{N} + \boldsymbol{S} - \boldsymbol{B} \tag{54}$$

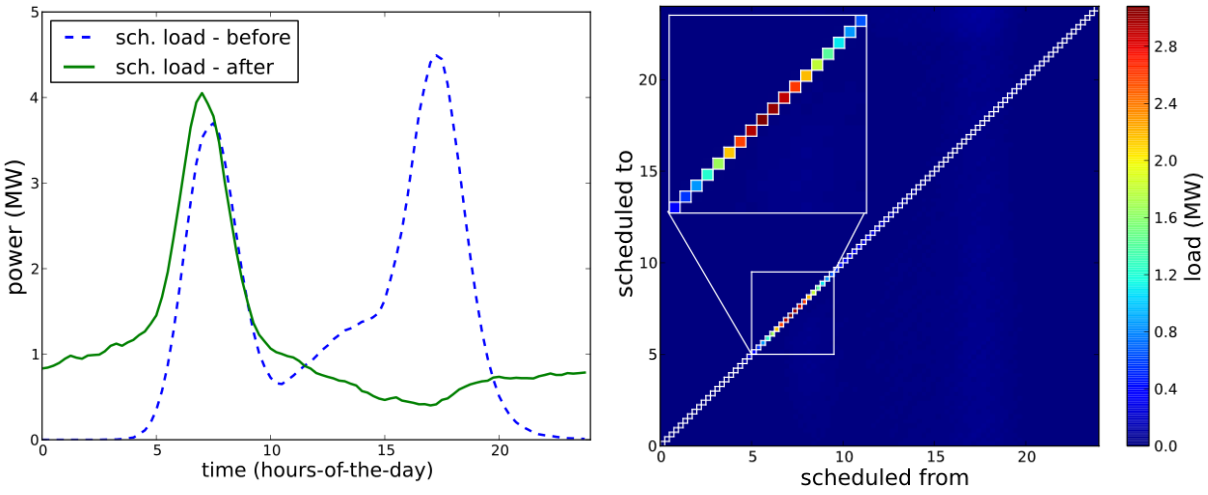
The optimization was implemented in the form of a genetic algorithm in Chapter 3 [34] based on a heuristic framework presented in Chapter 2 [7]. The simulation from Chapter 3 used a probabilistic method adapted from [46] (summarized in Appendix A) to generate 56,642 total smart appliances, amongst 5,555 customers, to be scheduled. The visualization methods presented in this paper use the data from the results of the genetic algorithm

optimization of these smart appliances and customer incentive pricing [34]. For comparison, there are two sets of data: one that is constrained on where loads can be rescheduled to and one that is unconstrained.

To map customers to a spatial location on the distribution network, the Roy Billinton Test System (**RBTS**) [104] is used. The RBTS is a standard six-bus test system with accurately modeled distribution assets that is commonly used for distribution network modeling and simulation [46, 105, 106]. For simulation purposes, RBTS Bus 5 is modeled, containing 26 loadpoints along four feeders for the 5,555 customers. The customers are probabilistically assigned to the loadpoints according to the probabilities in Table E.1 in Appendix E. The probabilities were calculated by normalizing the number of customers on each load point to the total number of customers provided in [104]. The nodes in the bus were mapped onto Fort Collins, Colorado, using the power line lengths described in the RBTS (the node coordinates are described in Table E.1). The DR from Chapter 3 was overlaid onto the loadpoints using the same 15-minute intervals over a period of 24-hours, creating a unique spatio-temporal DR visualization.

5.3. Demand Response Visualization

Demand response actions are usually shown as load data in two dimensions as a load curve, shown in Figs. 5.1(a) and 5.2(a). Although this provides a glimpse at the aggregate load before and after the DR action, there is information missing about what loads were scheduled when, how much profit was made, the spatial coordination of DR, etc. To overcome the deficiency of this missing information, three additional graph types are proposed that have one or more extra dimensions of information. The first is a heat map representation where the (x, y) coordinates indicate the magnitude of the event where smart appliances



(A) (B)

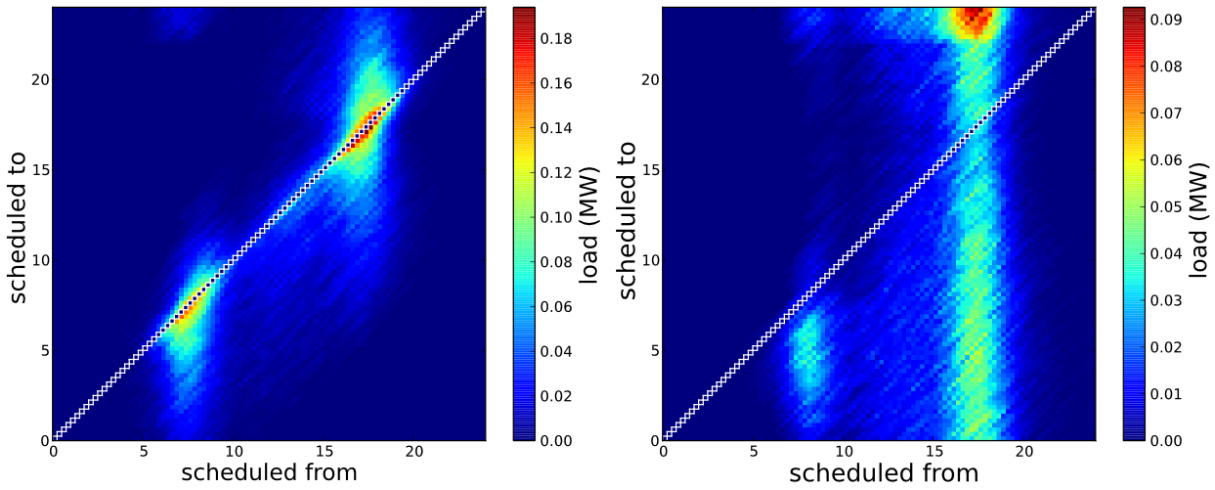
FIGURE 5.2. (a) Load curves of the schedulable load before and after the DR action in the system in the unconstrained case. (b) A heat map showing the temporal source and destination of the unconstrained schedulable load. To highlight the fact that the magnitude of the load that is *not moved* from time 5-9 is much greater than the typical amount of load *moved* from any other time x to time y in (b), it is shown in greater detail.

were rescheduled from time x to time y. The heat map representations can be seen in Figs. 5.1(b), 5.2(b), 5.3, and 5.5. The second type of graph is a three-dimensional (3D) representation where the coordinate (x, y, z) indicates the magnitude z of the DR event where smart appliances were rescheduled from time x to time y, shown in Fig. 5.4. The last type of graph represents the DR spatially using GIS, as shown in Fig. 5.7. The heat maps and 3D load curves were created using the Matplotlib library in Python, with the 3D graphs using the Mplot3d variant. Figures using color-blind-friendly color palettes were also created, although omitted due to space constraints. The GIS figures were created using the Keyhole Markup Language (**KML**) toolbox in MATLAB to generate overlays for Google Earth.

Fig. 5.1(b) contains the same information as Fig. 5.1(a), but it adds a new dimension of data using color as magnitude. Both figures are for the case where the smart appliances

have a constraint on what times they can be scheduled to in the DR action. The color at the (x, y) coordinate in Fig. 5.1(b) indicates the magnitude of the load moved from time x to time y. The white squares around the diagonal, i.e., $x = y$, correspond to the amount of load that was not rescheduled. In the case when $x > y$, i.e., below the diagonal, the load was scheduled for a time earlier than originally scheduled. Conversely, a value of $x < y$ indicates the load was scheduled for a later time. At a given original scheduled time of x, the larger the value of $|y - x|$, the further away the load was scheduled via the DR action. Most of the scheduling activity occurs around the peak times, which can be seen in the total schedulable load in Fig. 5.1(a), around 8:00 and 17:00. This is because there is a greater amount of load to be scheduled at those times. Because of the constraint on scheduling times, however, the distance from the original scheduled time is limited, leading to the smaller, but still noticeable, peak observed in Fig. 5.1(a) after the DR action. If you sum all loads at a given x-value across all y in Fig. 5.1(b), the total load will equal the load at the same x-value on the blue dotted line in Fig. 5.1(a). Similarly, the sum of load across all x will equal the green solid line. To highlight the fact that the magnitude of the load that is *not moved* from time 5-9 is much greater than the typical amount of load *moved* from any other time x to time y in Fig. 5.1(b), it is shown in greater detail.

Fig. 5.2 is similar to Fig. 5.1, except it relaxes the constraint on times to which the smart appliances can be rescheduled. This case is used as a comparison of constrained versus unconstrained scheduling. The total electric energy consumed in each heat map is the same (no load shedding), however the relaxation of the constraint on scheduling times leads to a stark difference in the distribution of the load. Fig. 5.2(a) shows that the first peak actually *increases* the amount of energy consumed. This occurs because in the case that was studied, the spot market and real-time price were lowest during the first peak.



(A) (B)

FIGURE 5.3. Heat maps showing only the load that was moved (so the diagonal, by definition, is zero) for (a) the constrained DR in Fig. 5.1(b); and (b) the unconstrained DR in Fig. 5.2(b). In general, the magnitudes of points in (b) are lower than the corresponding points in (a), as indicated by the labels on each color bar.

This may seem counter-intuitive because of the large load at that time, but recall that this is for residential households only and does not take into account commercial and industrial loads. This would most likely change in a residential-heavy distribution area such as that run by ERCOT in Texas where residential loads during summer account for approximately 50% of the total load [68]. To better represent the difference between the load curves in the constrained and unconstrained case in Fig. 5.1(b) and Fig. 5.2(b), respectively, only the load that was moved is plotted in Fig. 5.3. The load moved from the second peak is much larger in the unconstrained case and is spread throughout the day. Because of the spreading effect obtained by relaxing the constraint, the magnitude at each data point is lower in Fig. 5.3(b) than in the constrained case in Fig. 5.3(a) (as indicated by the different scales on the color bars).

The same information in Figs. 5.3(a) and 5.3(b) can be found in Figs. 5.4(a) and 5.4(b), respectively, but with a third dimension. Any point on the surface (x, y, z) gives the load

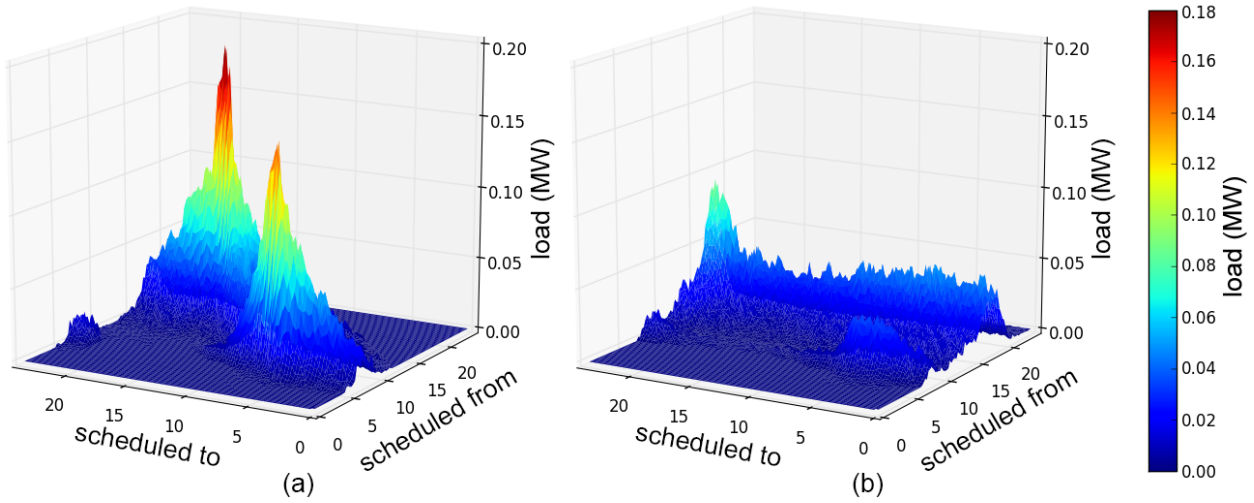


FIGURE 5.4. The 3D load graph shows the temporal displacement of the schedulable loads in both the (a) constrained and (b) unconstrained cases. Any point on the surface (x, y, z) gives the load z, in MW, displaced from time x to time y. Graphs (a) and (b) show the same information as Figs. 5.3(a) and 5.3(b), respectively, but with an extra dimension. The colors directly correspond to those in Fig. 5.3(a).

z, in MW, displaced from time x to time y. By using the z-axis in addition to color, this presentation of data is better for determining the disparity in magnitude of the schedule at a glance. The color information directly correlates to the heat map in Fig. 5.3(a). In this presentation, the difference in magnitude between the constrained and unconstrained cases is more apparent than in Fig. 5.3. By relaxing the constraint on what times smart appliances can be scheduled to, there is near uniformity along the *scheduled to* axis around time 20 in Fig. 5.4(b), indicating the magnitude of the appliance loads are near-uniform throughout the day. Because of the magnitude of the load not moved, for presentation purposes, the z values of the four datapoints closest to each $x = y$ datapoint are averaged to create a smooth graph.

The graphs in Fig. 5.5 are similar to those in Fig. 5.3, but instead of showing load information they show profit information for the constrained case. The color at a point (x, y) in Figs. 5.5(a) and 5.5(b) show the profit, in USD, from the smart appliances with

start times rescheduled from time x to time y. The white dotted diagonal line show the times when $x = y$. There is no profit made at these times because the smart appliances were not rescheduled, therefore there is no income or cost associated. Fig. 5.5(a) shows the profit when calculated using the forecast for both the spot market and real-time price information. The former is obtained from the bulk electricity spot market and the latter is a dynamic pricing scheme from the local distribution company. In this work, actual dynamic pricing and spot market pricing data was used from Saturday July 9, 2011 [1, 2]. This is the data that the aggregator would use for optimization in the day-ahead scheduler. Fig. 5.5(b) shows the profit when the appliance schedule and customer incentive price are evaluated using the actual spot market price. The background color (i.e., when the magnitude is equal to zero) in Figs. 5.5(a) and 5.5(b) is lighter than in Figs. 5.3(a) and 5.3(b) because the color scale contains negative values (i.e., a loss is experienced). Because it can be difficult to pick out the differences between Figs. 5.5(a) and 5.5(b), Fig. 5.5(c) shows the *difference* between the actual profit and forecast profit. The color at point (x, y) shows the difference in the aggregator profit between using the actual and forecast spot market pricing information from the smart appliances rescheduled from time x to time y. The black dotted diagonal line shows the times when $x = y$. The red areas show the rescheduling events where the aggregator made more profit than forecast while the blue areas show those events where less profit was received than expected.

Additional insight into Fig. 5.5(c) is provided by Fig. 5.6, which shows the difference in the spot market price between the actual and forecast values. The red areas in Fig. 5.5(c) are vertically skewed around 16:00, i.e., they correlate to smart appliances scheduled *from* 16:00 to any other time during the day. As shown in Fig. 5.6, this time directly relates to a large increase in the actual spot market price compared to the forecast price. Because these loads

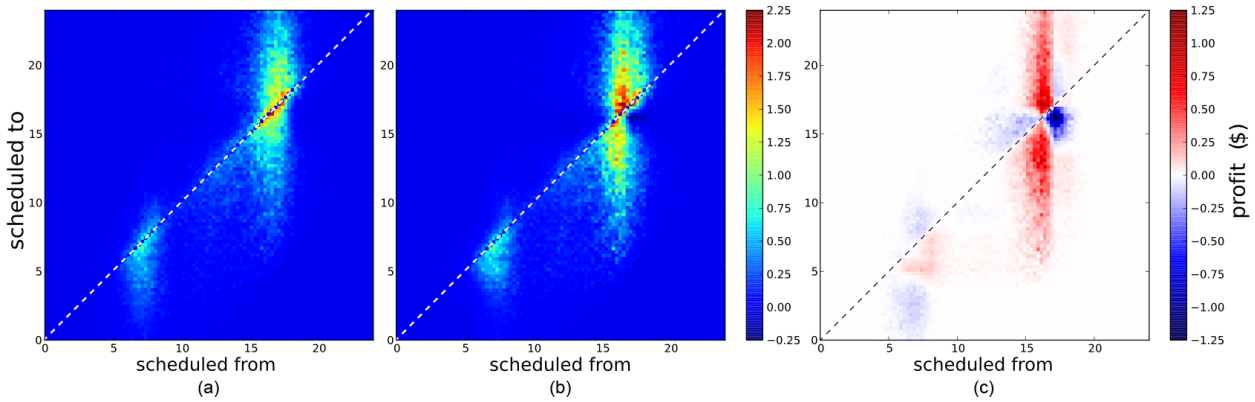


FIGURE 5.5. Heat map of times that the aggregator made a profit in the constrained case. Graph (a) gives the aggregator day-ahead forecast profit when using forecast price information. In graph (b), the *actual* aggregator profit is shown by replacing the forecast information with the actual pricing. To emphasize the contrast between graphs (a) and (b), graph (c) shows the difference between the actual profit and the forecast profit (i.e., actual minus forecast). The color at position (x, y) on the heat map in graphs (a) and (b) gives the profit made, in USD, from moving the loads from time x to time y. The color in graph (c) gives the difference in profit between using the actual and forecast pricing information, in USD, from moving the loads from time x to time y. The red areas in (c) indicate where the aggregator made more profit than forecast, while the blue areas indicate less profit. The white dotted diagonal line in graphs (a) and (b), as well as the black dotted diagonal line in graph (c), show where $x = y$.

are moved *from* these times, they are part of the $\boldsymbol{N}$ term in (54), leading to an increased income from selling these smart appliances as negative loads as part of the DR. Conversely, the blue areas in Fig. 5.5(c) are horizontally skewed around times 16:00 and vertically skewed around 6:00. The horizontally skewed decrease in profit is due to electricity purchased at a higher spot market price, leading to increased cost from the $\boldsymbol{B}$ term. The vertically skewed profit decrease at time 6 occurs because of a reduction in spot market price, leading to a decreased profit from selling the negative load represented by the $\boldsymbol{N}$ term. The densest areas of blue are below the diagonal line, indicating that the smart appliances resulting in this loss were scheduled earlier in the day.

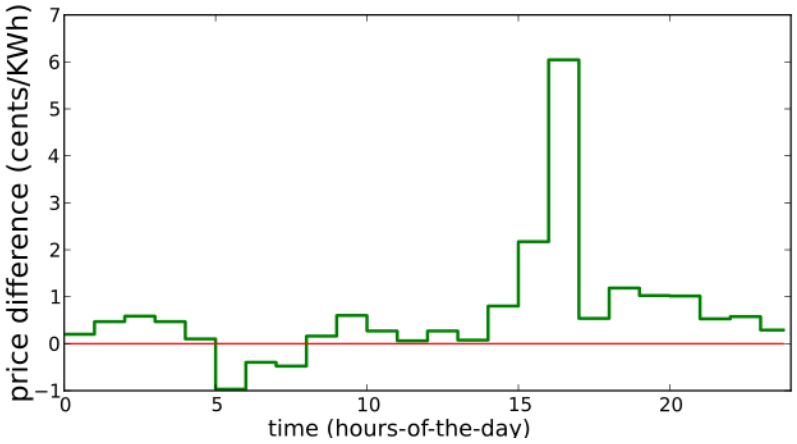
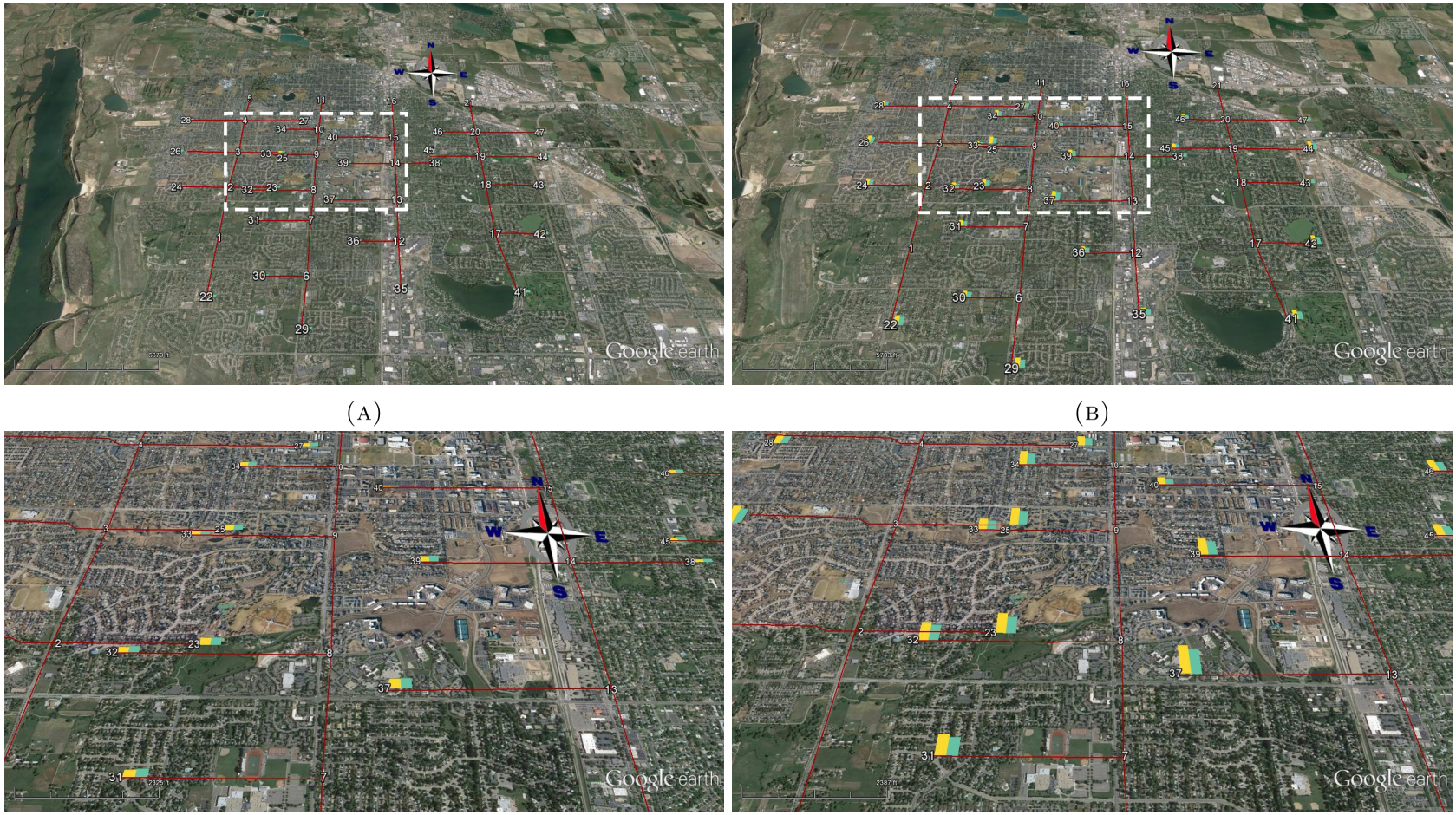


FIGURE 5.6. The price difference between the actual and forecast spot market prices [2] is given as the solid bold green line. The solid red horizontal line indicates no difference. If the difference curves are positive, i.e., above the red line, the price of electricity is more than forecast. If negative, the price of electricity is less than forecast.

The plots in Fig. 5.7 show the spatial variation of the DR using GIS overlays. Four figures are displayed to represent the entire 24-hour period, but a time-varying movie can be found with the code at the SGRA website[1]. The top two figures show the entire RBTS bus and the bottom two figures zoom in on an area of interest. The left two figures show an off-peak time (10:00-10:15) and the right two figures show a peak time (16:45-17:00). The red lines show the branches between the nodes of the RBTS bus, with the white and black numbers describing the node numbers (GIS coordinates for the nodes are described in E). The nodes that have the side-by-side bar graphs are the customer loadpoints with the yellow bar on the left representing the load before the DR action and the blue-green bar on the right representing the load after the DR action. A scale and compass rose are presented in the bottom-left and top-right corners, respectively, to provide orientation. As shown in Fig. 5.7(d), the DR action is non-uniform in space. Comparing Fig. 5.7(c) to Fig. 5.7(d), the DR action is also non-uniform in time (also shown in the heat maps and 3D load graphs).

[1]http://www.engr.colostate.edu/sgra/



(A) (B)

(C) (D)

FIGURE 5.7. GIS overlays of a DR action for the RBTS system mapped onto Fort Collins, Colorado, in Google Earth. The top two figures show the entire RBTS bus and the bottom two figures zoom in on an area of interest. The dashed box in the top two figures indicate the zoomed area of the bottom two figures. The left two figures show an off-peak time (10:00-10:15) and the right two figures show a peak time (16:45-17:00). The side-by-side bar graphs describe the load change from the DR action with the yellow bar on the left representing the load before the DR action and the blue-green bar on the right representing the load after the DR action.

Without properly mapping the DR information onto a spatial system, such as GIS, these unique spatio-temporal characteristics are lost.

In addition to providing more information to the system operator, these new visualization methods also can help understand how the optimization technique, in this case the genetic algorithm [34], is performing. By examining the load graph in Figs. 5.1(a) and 5.2(a), one only gets the aggregate information about the load and DR action. The optimization technique, however, makes decisions on individual loads, each with their own initial schedule and constraints. The heat map and 3D graphs help show at a finer grain of detail the decisions being made by the genetic algorithm and their impact on schedules and profits, while the GIS graphs show the spatio-temporal characteristics of the decisions.

5.4. Conclusions

Three visualization methods (heat maps, 3D load graphs, and GIS) were adapted for a given set of solutions to a demand response problem. Using these visualization methods, it becomes possible to answer: whether or not the demand response plan worked effectively; at what times the demand response resulted in a profit or a loss; how multiple demand response solutions compare; and where in the distribution system the demand response actions occurred. This allows greater insight into the effectiveness and profitability of the demand response programs and the effect of optimizing for the forecast price and data. The three visualization methods were examined in depth, describing what is being shown and the usefulness of each method. The figures shown here are not exhaustive and represent a subset of the visualization capabilities of the heat map, 3D load graph, and GIS techniques.

CHAPTER 6

Bus.py: A GridLAB-D Communication Interface for Smart Distribution Grid Simulations

6.1. Introduction

In Title XIII of the Energy Independence and Security Act of 2007, the U.S. Congress set forth the tenets of modernizing the electricity grid through the Smart Grid initiative [108]. Some of these tenets include the deployment, development, and integration of distributed energy resources, "smart" technologies and appliances, and advanced storage devices. The integration of these technologies requires new modeling and simulation tools, but it is difficult for a single tool to model all power systems domains in adequate detail. This leads to the use of *co-simulation* tools where multiple individual tools, each modeling a single domain in detail, interact while running simultaneously [15].

This work introduces Bus.py (pronounced bus-dot-pie), an abstract software transmission bus interface that facilitates the co-simulation of tools, e.g., customer home energy management systems and the distribution feeder. One such simulation tool is GridLAB-D: a flexible distribution system simulator that allows fine-grain modeling of distribution assets from the substation transformer down to the individual household [109]. This fine-grained modeling of the distribution system makes it an ideal tool to perform Smart Grid technology studies at

This work was performed jointly with the full list of co-authors available in [107]. This work was supported by the U.S. Department of Energy under Contract No. DE-AC36-08-GO28308 with the National Renewable Energy Laboratory. See Bus.py project updates at http://www.engr.colostate.edu/sgra/.

the distribution level. Bus.py leverages the fine-grained modeling and inter-process communication of GridLAB-D to enable a myriad of future-grid scenarios. To demonstrate Bus.py's effectiveness for performing co-simulation studies, two illustrative examples are presented.

In the last few years, there has been relevant related work in the area of electric power grid co-simulation tools, specifically GridLAB-D. GridMat is a Matlab-GridLAB-D interface for residential controllers for use in a residential microgrid [110]. GridMat includes a similar GridLAB-D control interface for residential control only. In addition to residential control, Bus.py enables additional use cases, such as transmission-distribution integration through the use of high-performance computing. In [111], another Matlab-GridLAB-D co-simulation framework was introduced. This work differs in that Bus.py offers dynamic interaction with GridLAB-D while [111] performs all optimization offline to be used as a static GridLAB-D simulation. GridSpice [112] is a GridLAB-D cloud infrastructure that enables the deployment of many GridLAB-D instances to the Amazon cloud for the purpose of transmission-level power analysis with MATPOWER [3]. In contrast, Bus.py allows the dynamic simulation of multiple grid use-cases without relying on a cloud infrastructure, but can be enabled by many computing platforms (e.g., personal computer, high-performance computing). In this work, the following unique contributions are made.

(a) The design of Bus.py, a software transmission bus interface for use in Smart Grid co-simulation studies,

(b) the creation of a flexible communication interface with the distribution simulator GridLAB-D,

(c) a discussion of the usefulness of the interface, and

(d) a demonstration of Bus.py interacting with GridLAB-D simulating a small set of customers on a distribution feeder and an aggregator entity.

The rest of this chapter is organized as follows. Section 6.2 describes the Bus.py interface. The usefulness of Bus.py is presented in Section 6.3 by illustrating how one can simulate the control of multiple residential households with a residential aggregator using Bus.py and GridLAB-D. Concluding remarks are given in Section 6.4.

6.2. Bus.py

6.2.1. Overview. The name Bus.py derives from the fact that its purpose is to emulate a transmission-level bus. To accomplish this goal, an abstract interface is provided in Python that includes inter-process communication to the distribution simulator GridLAB-D. To facilitate interactive simulation, GridLAB-D currently provides an HTTP server for inter-process communication. Other provided implementations of the Bus.py interface are a constant load bus, time-series load bus, and resistance-based load bus (where a Thévenin equivalent resistance is used in conjunction with Ohm's law to determine the load at a bus). This chapter focuses on the implementation of the Bus.py interface with GridLAB-D, described in the following subsection. A few interesting co-simulation use cases, made possible with Bus.py, are given in Subsection 6.2.3.

6.2.2. Interface. The principle of Bus.py is a flexible, easy-to-use interface to enable the co-simulation of electric power system tools. Pseudocode that describes the operation of Bus.py with a generic co-simulator (e.g., microgrid controller, HEMS controller) is presented in Fig. 6.1. Bus.py has four main functions: load_bus, start_bus, transaction, and stop_bus (given as lines 1, 3, 6, and 9 in Fig. 6.1, respectively), each described in detail below.

The **load_bus** function reads from a bus input file all relevant parameters for Bus.py to be used during the co-simulation process. The input file will specify which type of bus will be modeled (e.g., a GridLAB-D bus), simulation time information (i.e., start time, stop

```
 1: Bus = load_bus(input_file)
 2: cosimulator.initialize()
 3: Bus.start_bus()
 4: repeat
 5:    bus_inputs = cosimulator.optimize()
 6:    bus_outputs = Bus.transaction(bus_inputs)
 7:    cosimulator.process(bus_outputs)
 8: until Bus.finished
 9: Bus.stop_bus()
10: cosimulator.postprocess()
```

FIGURE 6.1. Bus.py pseudocode with an abstract co-simulator process.

time, and timestep), and any other relevant parameters for that bus type. The input file is specified using the JavaScript Object Notation (JSON), an easy-to-read set of key-value pair strings. Load_bus will return a Bus object to be used for the co-simulation.

Once a Bus object is loaded and the co-simulator is initialized, **start_bus** will start the bus co-simulation environment. In the case of a GridLAB-D bus, this will start a GridLAB-D simulator process and start its HTTP server for inter-process communication with the Bus.py interface.

After the co-simulation environment is started with start_bus, the main simulation loop begins (lines 4–8 in Fig. 6.1). Each iteration of the loop represents one timestep in the simulation. The basic order of operation at timestep t is: (1) obtain the inputs to the Bus for timestep t from the co-simulator, (2) perform a transaction with Bus, and (3) process the outputs of Bus using the co-simulator. The **transaction** function passes the inputs to the Bus, steps time forward, and returns the specified outputs. For GridLAB-D, this will (1) send key-value pairs (e.g., customer1.load=10 kW) to GridLAB-D using HTTP, (2) step GridLAB-D forward one simulation timestep, and (3) request and return the GridLAB-D simulated output (e.g., substation power). *The single transaction function simplifies the communication with GridLAB-D to present a powerful co-simulation tool.*

After the stop time of the simulation is reached (line 8 in Fig. 6.1), the simulation loop will end. The **stop_bus** function will stop any associated processes/files used by the Bus object (e.g., stop the GridLAB-D process in the case of a GridLAB-D Bus). Once the main simulation loop ends and Bus is stopped, it may be useful to perform post-processing with the co-simulator, such as visualization of the time-series output.

6.2.3. Use Cases. The following subsection will describe a non-exhaustive list of possible use cases for Bus.py-enabled co-simulation. The first use case is the co-simulation of a transmission-level simulator, such as MATPOWER [3], and a Bus instance at each bus in the transmission system. This interaction is depicted in Fig. 6.2. The physical simulation of the bulk power system and distribution systems have traditionally been modeled as separate with simplified representations of their interaction in each individual simulation. As more Smart Grid technologies (e.g., demand response technologies, distributed energy resources) are implemented in the distribution system, these separate simulations may no longer be adequate, such as in the case with bi-directional power flow resulting from a high penetration of distributed photovoltaic generation [113]. *Bus.py enables bi-directional power flow studies resulting from increased distributed energy resources.*

A second use case is presented in Fig. 6.3. In this use case, many customer HEMS are optimizing and interacting on a single distribution feeder, modeled by GridLAB-D. Because each house is located at a separate physical location on the distribution feeder, each will have slightly different physical interactions with said distribution feeder (e.g., different voltage levels, different phases). Increasing the number of integrated Smart Grid technologies and allowing retail customers direct access to market prices may increase the price-elasticity of demand, leading to an increased volatility in power systems [47]. If many HEMS are simulated individually (i.e., without modeling the effect of the distribution grid), this volatility

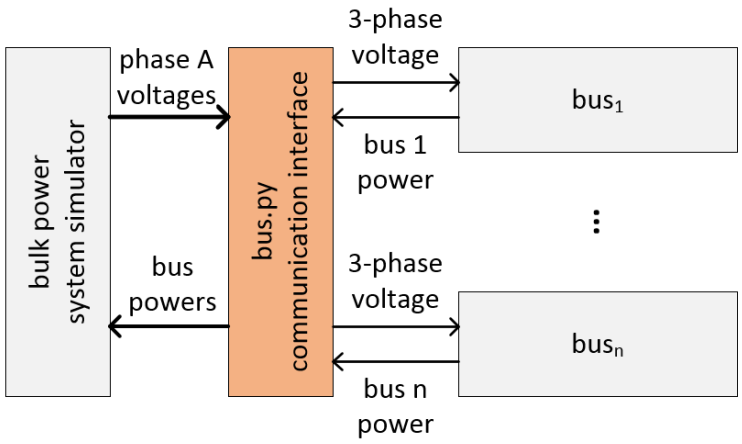


FIGURE 6.2. Bus.py interfacing a bulk power simulator, such as MATPOWER [3], and many Bus instances (e.g., GridLAB-D, time-series). This scenario could be used to integrate transmission and distribution system simulators.

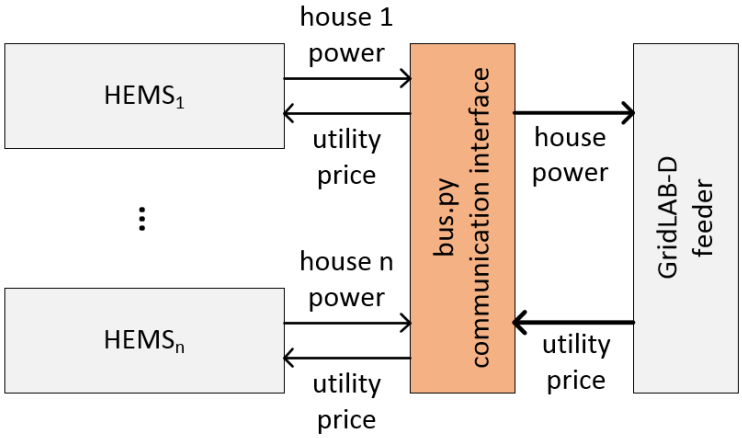


FIGURE 6.3. Bus.py interfacing a GridLAB-D feeder with many customer home energy management systems (HEMS). This scenario could be used to determine the effect of many HEMS on a distribution system.

may not be properly modeled. *Bus.py allows the study of the physical responses on distribution feeders from a multitude of customer HEMS and other Smart Grid technologies.* Additionally, because the distribution system properties (e.g., voltage, retail prices in a real-time pricing environment) change through time, a static simulation of customer HEMS is not sufficient. *Bus.py facilitates the dynamic co-simulation of Smart Grid technologies and their respective distribution feeder.*

This list of use cases is not exhaustive but is meant to illustrate the usefulness of the Bus.py interface to enable the co-simulation of tools. Additional use cases could include market (with a bulk power market simulator such as FESTIV [114]) and communication (with a communication simulator such as ns-3 [115]) co-simulations.

6.3. Residential Aggregator Demand Response

6.3.1. System Model. To demonstrate the usefulness of the Bus.py interface, two illustrative co-simulation examples are presented. The first is a common load shifting problem to reduce the peak system load, presented in Subsection 6.3.3. The second example quantifies the effect of time-of-use (ToU) pricing on the system load profile, presented in Subsection 6.3.4. In both examples, an aggregator-based residential demand response is used (different from the one described in Chapter 3), described in detail below and presented in Fig. 6.4.

On a given distribution feeder, $\boldsymbol{N}$ customer households are modeled. Each household i has a set of $\boldsymbol{n_i}$ loads (e.g., smart appliances) that are made available to an aggregator for rescheduling. For each load j of customer i, the aggregator is assumed to know:

- $\boldsymbol{p_{ij}}$ – the average load, in kW,
- $\boldsymbol{\delta_{ij}}$ – the duration of the load's operation,
- $\boldsymbol{A_{ij,min}}$ – the start of the customer-defined rescheduling window, and
- $\boldsymbol{A_{ij,max}}$ – the end of the customer-defined rescheduling window.

The time period from $A_{ij,min}$ to $A_{ij,max}$ is a customer-defined rescheduling window that is used to take into account customer comfort [65]. The aggregator-based residential demand response is set up as an optimization problem. For each customer load, the aggregator must find the rescheduled time, $\boldsymbol{T_{ij}}$, subject to $A_{ij,min} \leq T_{ij} < A_{ij,max}$, that optimizes an objective

function (described in Subsections 6.3.3 and 6.3.4). To perform the optimization, a genetic algorithm is used, described in the following subsection.

The simulation is set up for a 24-hour period with a time step, $\boldsymbol{\Delta t}$, of 15-minutes (i.e., 96 simulation periods) occurring on June 1, 2012. For the GridLAB-D inputs, 206 houses were placed on a Pacific Northwest National Laboratory taxonomy feeder (using the method from [116]) representing a lightly populated rural area (R4-25.00-1) [117]. Weather data for Charlotte, North Carolina, was used to match the location of the taxonomy feeder in the non-coastal southeast United States. Each customer has between one and four schedulable loads leading to a total of 543 loads. Let $\mathcal{N}(\mu, \sigma)$ represent a normal distribution with mean μ and standard deviation σ. Each load has the duration and power generated randomly from $\mathcal{N}(4, 2)$ and $\mathcal{N}(0.8, 0.2)$, respectively. A baseline load is determined by randomly assigning each load to start at one of the 96 simulation periods. The duration of the customer-defined rescheduling window is randomly generated around this baseline time for each load from $\mathcal{N}(16, 4)$. Note that these values are used for simulation purposes only and do not have any bearing on the usefulness of the Bus.py interface.

6.3.2. Heuristic Approach. In this example, a Genitor [33] version of genetic algorithm (GA) implemented in Python was used to perform the aggregator optimization in conjunction with Bus.py. The GA is a global search heuristic that has been shown to work well in many optimization problems in power systems, such as economic dispatch [16] and unit commitment [18]. A gene within the chromosome represents an individual schedulable load. Let $\boldsymbol{s_{ij}}$ be a real value in the interval $[0, 1]$ representing the T_{ij}. To obtain the time interval that each load was scheduled, the following equation was used: $T_{ij} = A_{ij,min} + s_{ij}\,(A_{ij,max} - A_{ij,min})$. The $[0, 1]$ representation of s_{ij} was used to avoid violating the customer-defined rescheduling window constraint of the loads [34].

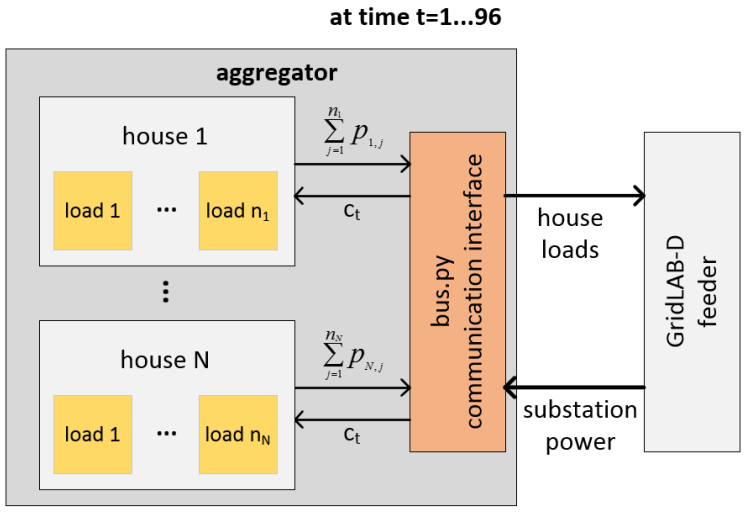


FIGURE 6.4. The proposed aggregator system model. At each 15-minute time step, the aggregator determines the total schedulable load at each customer household and passes the information of the loads to GridLAB-D through the Bus.py interface. The aggregator then requests the substation apparent power following the scheduling of the loads to verify and quantify the change in load.

The Genitor is a steady-state algorithm that maintains a ranked list of chromosomes, leading to implicit elitism, i.e., between generations, the best solutions are kept. In each generation, two parents are selected using the linear bias function (as defined in [33]) for crossover. The linear bias selection function requires a linear bias parameter that is a real value in the interval $(1, 2]$. A linear bias parameter of 1.5 means the best-ranked solution has a 50% greater chance of being selected than the median solution. A two-point crossover method is used. The mutation operator on a gene will randomly generate a new s_{ij} between $[0, 1]$. In this simulation study, the genetic algorithm runs for 10,000 iterations using 50 chromosomes with a probability of mutation of 0.05 and a linear bias of 1.5. Note that these values are used for simulation purposes only and do not have any bearing on the usefulness of the Bus.py interface.

6.3.3. PEAK LOAD MINIMIZATION. The first illustrative example involves peak load minimization through load shifting using Bus.py as in Fig. 6.4. Load or demand shifting is

a well-known demand response technique [118, 119]. Let $\boldsymbol{\kappa_t}$ be the known fixed load of the distribution system at time t in kW (i.e., the unschedulable load). The peak load version of the aggregator-based residential demand response problem is defined as: find $T_{ij} \forall i, j$, subject to $A_{ij,min} \leq T_{ij} < A_{ij,max}$, to minimize:

$$\max_{t=1...96} \kappa_t + \sum_{i=1}^{N} \sum_{j=1}^{n_i} \begin{cases} p_{ij} & T_{ij} \leq t < (T_{ij} + \delta_{ij}) \\ 0 & else \end{cases} . \tag{55}$$

The resulting substation apparent power throughout the simulation period is presented in Fig. 6.5. Because the optimization was peak reduction, the peak load between 3:00 to 6:00 pm is shown in more detail in the inset in Fig. 6.5. The solid blue line represents the baseline load of the system (i.e., the system load in the absence of the aggregator demand response). The dashed green line represents the substation apparent power, in kVA, after the aggregator-based residential demand response was performed. The peak system power at 5:15 pm was reduced by 19.2 kVA, the total schedulable load available for demand response at that time. This corresponds to a 2.5% decrease in peak system load, which aligns with the Federal Energy Regulatory Commission (FERC) expectations for demand response in the residential sector [120].

6.3.4. Customer Cost Minimization. The second illustrative example involves total customer cost minimization of the schedulable demand response loads (i.e., smart appliances in this work) in a ToU market using Bus.py as in Fig. 6.4. The effect of ToU on load profiles is a common optimization problem [121]. Let $\boldsymbol{c_t}$ be the cost of electricity at time t in \$/kWΔt (where Δt was 15-minutes). Because the distribution feeder modeled is using inputs for North Carolina, the ToU pricing from Duke Energy in North Carolina was used [122]. Note that only the energy charge was considered. Modeling the monthly demand charge would

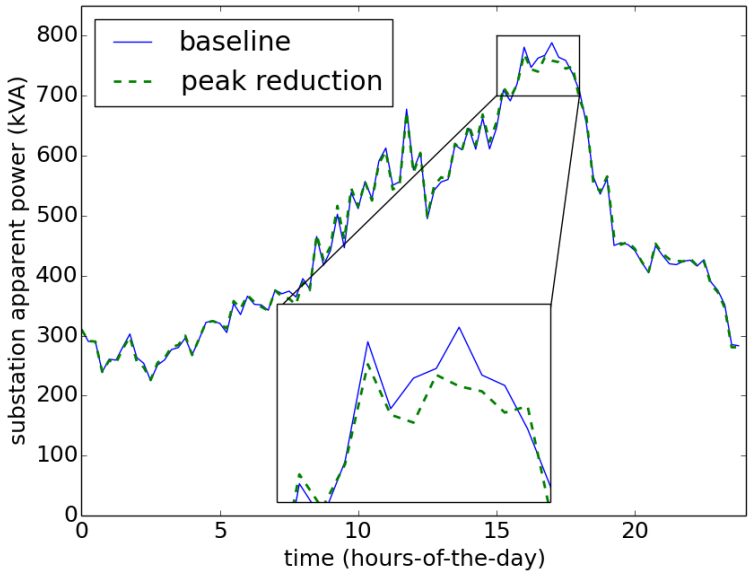


FIGURE 6.5. The comparison of the substation apparent power, in kVA, between the baseline (solid blue line) and aggregator demand response (dashed green line) cases resulting from peak minimization. Because the optimization was peak minimization, the peak of the system is shown in more detail in the inset.

require a longer simulation period, which is possible using Bus.py but not shown here for brevity. The energy cost minimization version of the aggregator-based residential demand response problem is defined as: find $T_{ij} \forall i, j$, subject to $A_{ij,min} \leq T_{ij} < A_{ij,max}$, to minimize:

$$\sum_{t=1}^{96} c_t \sum_{i=1}^{N} \sum_{j=1}^{n_i} \begin{cases} p_{ij} & T_{ij} \leq t < (T_{ij} + \delta_{ij}) \\ 0 & else \end{cases} . \tag{56}$$

The resulting load of *only the schedulable loads*, which represents 450 kWh of the distribution feeder, throughout the simulation period is presented in Fig. 6.6. The solid blue line represents the baseline schedulable loads of the customers. The dashed green line represents the customer schedulable loads, in kW, after the aggregator-based residential demand response was performed while optimizing for the minimization of customer energy cost. The solid black curve shows the ToU pricing used in the optimization, with the corresponding y-axis values on the right, in cents/kWh.

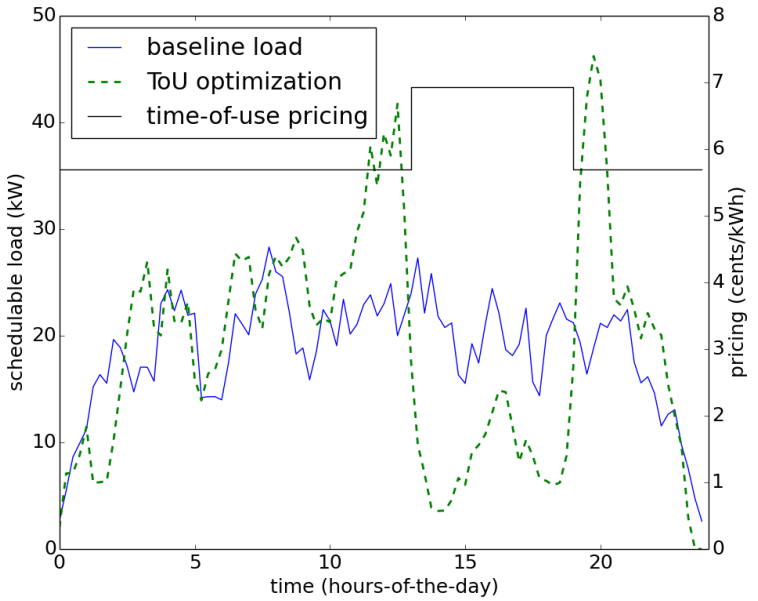


FIGURE 6.6. The comparison between the load, in kW, of the customer schedulable loads made available to the aggregator between the baseline (solid blue line) and aggregator demand response (dashed green line) cases resulting from customer cost minimization. The time-of-use pricing used is given as the solid black curve.

During the ToU peak-pricing period, from 1:00 to 7:00 pm, the total schedulable loads of all the customers pushed off-peak and resulted in a reduction from 123 kWh to 52 kWh. This makes sense because the only way to reduce customer cost in the ToU pricing scheme is to move load from on-peak to off-peak. The reason that *all* schedulable load was not moved off-peak is because of the customer-defined rescheduling window, from $A_{ij,min}$ to $A_{ij,max}$. It is interesting to note the resulting rebound effect—the change in the consumption pattern of electricity from the changing cost of electricity [123]—that occurs on either side of the transition from off-peak to on-peak pricing at 1:00 pm and at 7:00 pm. In our problem, the total energy remains the same because the loads are just shifted, but if thermal loads were considered sometimes *more* or *less* total energy is consumed because of this effect [124].

6.4. Conclusions

Bus.py is an abstract software transmission bus interface that facilitates the co-simulation of electric power system tools. As more Smart Grid technologies are implemented in the distribution system, it becomes infeasible for a single tool to simulate all electric power system domains at a detailed level. Bus.py enables the dynamic simulation of multiple electric power systems tools, such as GridLAB-D and customer home energy management systems. The Bus.py interface interaction with GridLAB-D, a distribution system simulator, was described. The existence of Bus.py enables the co-simulation of transmission and distribution systems, customer home energy management systems and the distribution system, as well as many other use cases. A demonstration of residential demand response resulting from multiple residential homes on a single distribution feeder through an aggregator in a single distribution area was achieved using the new Bus.py interface with GridLAB-D. The demonstration consisted of two examples: (a) system peak minimization and (b) customer cost savings in a time-of-use pricing scheme. The residential demand response resulted in (a) a 2.5% peak reduction and (b) an on-peak reduction from 123 kWh to 52 kWh, respectively.

CHAPTER 7

A Combined Dual-stage Framework for Robust Scheduling of Scientific Applications in Heterogeneous Environments with Uncertain Availability

7.1. Introduction

The rapid development of computing technology has increased the complexity of computational systems and their ability to solve large, and more complex, scientific problems. These computing systems often are heterogeneous and operate in uncertain environments, consisting of computing resources that differ in number and availability over time. Machine availability is the percentage of the machine's total computational resource that can be used for a given application. A machine is said to be loaded when its availability is less than 100%.

Scientific applications express the solutions to complex scientific problems, which often are data-parallel and contain large loops. The execution of such applications in heterogeneous computing environments is computationally intensive and exhibits an irregular behavior, in general due to variations of algorithmic and systemic nature [125]. Distribution of input data and variations of algorithmic nature cause intrinsic imbalance, while variations of systemic nature cause extrinsic imbalance [126]. Load imbalance in computationally intensive

This work was performed jointly with the full list of co-authors available in [9]. This work was supported by the German Research Foundation in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing," the National Science Foundation (NSF) under grant numbers CNS-0905399 and NSF IIP-1127978, and the CSU George T. Abell Endowment.

scientific applications is often their major performance degradation factor [125, 126]. Traditionally, solutions that address load imbalance in scientific applications involve dynamic data and/or work re-distribution.

The problem statement for this chapter has two components. First, given a collection of applications with uncertain input data and a heterogeneous computing system with uncertain availability, how can resources be assigned to maximize the probability that applications can complete by a common deadline? Second, given this allocation of resources, how can we minimize the makespan for this collection of applications?

The work presented herein demonstrates that using robust *resource allocation* (**RA**) heuristics [127] *and* application load balancing via *dynamic loop scheduling* (**DLS**) techniques, in concert, will enhance the execution of computationally intensive scientific applications in uncertain heterogeneous systems. The goal of this chapter is to assign applications to heterogeneous computing systems and execute them in such a way that all applications complete before a common deadline, and their completion times are robust against uncertainty in input data *and* system availability.

To accomplish this goal, the approach proposed herein is to divide the execution of scientific applications on heterogeneous computing systems into two stages, as outlined in Fig. 7.1:

- Stage I *initial mapping*–resources are allocated to each application according to a given robust RA policy.
- Stage II *runtime application scheduling*–the execution of each application is optimized, for the set of resources allocated in the previous stage, according to a given robust application scheduling strategy.

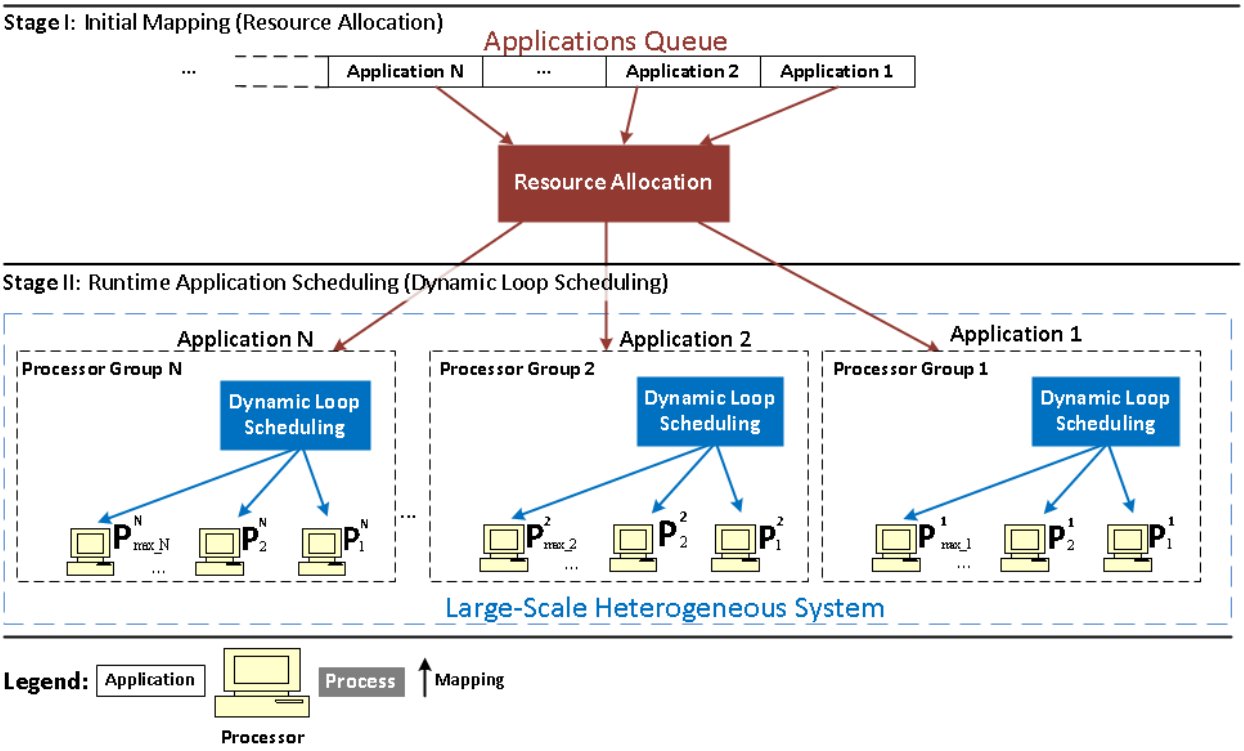


FIGURE 7.1. Schematic illustration of the proposed dual-stage framework. A resource allocation heuristic is employed in stage I to assign each application from a batch of N applications in the queue to one of the N groups of processors of a large-scale heterogeneous system. Dynamic loop scheduling techniques are used in stage II for runtime scheduling of each application onto the processors of their respective assigned group.

Initial mapping (**IM**) can be defined as the problem of finding a mapping of a batch of applications onto a set of resources to maximize robustness against uncertain input data and system availability. Robustness here is defined as the probability that applications are completed on the allocated resources by a common deadline [26].

Motivation for Stage I. The motivation for solving the IM problem via robust RA is to avoid the runtime resource reallocation problem, i.e., reallocating resources already assigned to applications to avoid violations of the performance objective. The robustness of an RA can be quantified as the joint probability that all applications will complete by their deadline given the uncertain input data and system availability.

Motivation for Stage II. Just as in stage I, uncertain runtime availability of resources allocated to an application, as well as uncertain input data, are known sources of uncertainty in stage II and may impact the applications execution times. The motivation for this stage is based on the assumption that a specific runtime application scheduling (**RAS**) policy exists that avoids the runtime resource reallocation problem and that satisfies the stated performance objective, while possibly allowing a larger degree of uncertainty in input data and system availability.

RAS is defined as the problem of selecting the DLS technique for dynamic load balancing of applications during their execution on the resources already allocated in stage I that maximizes robustness against uncertain input data and system availability [128, 129], defined as the maximum allowable decrease in the expected availability of the resources allocated in stage I before a performance objective violation occurs.

Employing a robust DLS technique for an application during stage II will allow the application to begin and complete its execution on the same set of resources that have been allocated during stage I, while in case of perturbations, only iterations (of the same application) will be migrated between the processors of the allocated resource set.

Usefulness. The usefulness of the proposed combined dual-stage framework is based on the following hypothesis: using an intelligent approach in both stages will result in better overall system performance than using an intelligent approach for either stage in isolation or neither. The dual-stage framework allows investigation of the overall degree of tolerable uncertainty, such that the desired performance objective is satisfied, for each application individually *and* the entire collection of applications running on the heterogeneous computing system.

Contribution. The main contribution of this chapter is the design of an intelligent two-stage framework to solve the problem of allocating resources to applications to maximize the probability that the applications can complete by a common deadline given uncertainty in the input data and system availability, including developing a mathematical model of this environment.

The next section presents a review of related work, RA heuristics, and DLS techniques. The proposed combined dual-stage framework is described in Section 7.3. The usefulness and benefits of the proposed framework are demonstrated via a small scale example in Section 7.4. The conclusions and insights into future work are summarized in Section 7.5.

7.2. Related Work

7.2.1. Overview. This work spans two major research areas: resource allocation and dynamic application scheduling in heterogeneous computing systems. The following is a brief summary of some work relevant to both research areas.

A framework for resource allocation and task scheduling is proposed in [130] for efficient allocation of heterogeneous grid resources to resource-intensive applications that minimizes their makespan and allocates the minimum number of resources. This approach is static and based on the current state of the grid resources. In contrast to the work in this chapter, the resource allocation and application scheduling strategies are intertwined, application tasks are assumed to take one unit of time, and no source of uncertainty is considered.

The problem of mixed resource allocation and task scheduling has been addressed via a constrained-based approach as a temporally constrained and a resource constrained problem [131]. Time constraints can be limit times and precedences, while resource constraints concern allocation and sharing. Constraints propagation mechanisms have been proposed

that led to the removing of some task assignments, or that determined inconsistent allocations between pairs of tasks. In contrast to the work in this chapter, it is assumed that the durations of the unassigned tasks are correlated and constant, and that resources are homogeneous.

Extensions to application and performance models used in compile-time task and resource allocation have been proposed that capture applications with statistical variations in execution times and task dependencies [125]. In contrast to the work in this chapter, the focus is on compile-time mapping of single applications onto homogeneous multiprocessor platforms.

The approach proposed herein to satisfy the stated performance objective in the presence of uncertainties is to divide the execution of scientific applications into two stages: stage I employs a RA heuristic to allocate a set of resources to every application, while in stage II, DLS techniques are employed (one for each application) to ensure an effective application execution on the set of resources allocated in stage I. Existing work on RA and DLS is reviewed next.

7.2.2. Review of RA Heuristics. In general, the resource allocation and the application scheduling problems are both known to be NP-complete [4–6], which leads to the use of scheduling heuristics. In the stochastic resource allocation model, the historical computing time of each task to be run on each of the machines in the system is said to be known beforehand and is represented as a probability mass function (**pmf**) [26]. The use of pmfs allows for the calculation of the probability of a machine finishing its tasks before a specified time. Because each of the machine's execution times are independent, the overall probability of the system completing by a specified time can be obtained by their joint probability.

Due to the fact that the example given later in the paper (in Section 7.4) is illustrative and represents a small scale case, no particular RA heuristic is actually needed, as the optimal allocation can be determined exhaustively. As a comparison metric between possible resource allocations, a simple load balancing technique is used, in which each application is allocated an equal number of resources.

7.2.3. Review of DLS Techniques. The most efficient dynamic load balancing approach for improving the performance of scientific applications employs DLS. This approach is effective in scientific applications that contain computationally intensive parallel loops. The DLS techniques are based on probabilistic analyses and ensure a high performance execution of the applications. Using DLS, a new size for the next chunk of ready-to-be-executed loop iterations is computed at runtime, and thereupon offered for execution to the first processor that finished executing other assigned chunks.

DLS methods provide two alternative approaches, *non-adaptive* and *adaptive*, for achieving good load balancing on variably loaded resources, as well as for executing iterations with varying execution times. Most of the techniques described in [132] are based on probabilistic analyses and are non-adaptive. Other non-adaptive techniques, not contained in the above survey, include fractiling and weighted factoring (**WF**) [133]. Subsequent efforts resulted in more elaborate techniques, called adaptive. A few examples include adaptive weighted factoring (**AWF**), and its variants: AWF-batch (**AWF-B**) and AWF-chunk (**AWF-C**), and adaptive factoring (**AF**) [134]. Most of these methods are derived from factoring (**FAC**) [135], and hence, employ rules similar to those of FAC to achieve dynamic load balancing.

The above adaptive methods are also based on probabilistic analyses. Their goal is to achieve the best possible scheduling that optimizes application performance (minimizing the makespan) via dynamic load balancing. The adaptive DLS techniques use a combination of

runtime information about the application (e.g., input data) and the system (e.g., availability) to (i) predict the system capabilities for the next computational assignments, or (ii) to estimate the time the remaining application iterations will require to finish execution. These techniques dynamically compute the size of chunks (a collection of iterations) at runtime, such that they are completed within their optimal time with high probability.

The DLS techniques considered in stage II in this work are two non-adaptive methods, FAC and WF, and two adaptive methods, AWF-B and AF. The usefulness of the proposed dual-stage framework is not limited to this choice of DLS techniques. Due to space limitations, the interested reader is referred to the appropriate references for further details of the above DLS techniques.

7.3. A Combined Dual-stage Framework

7.3.1. Uncertainty and Performance Objective. A novel combined dual-stage framework (**CDSF**) is proposed herein, with the goal of assigning applications to heterogeneous computing systems and executing them in such a way that all applications complete before a common deadline, and their completion times are robust against uncertain input data *and* system availability. The robust execution involves two stages: *initial mapping* using robust RA heuristics, in stage I, and *runtime application scheduling* using robust DLS techniques, in stage II. The CDSF provides a certain level of guarantee for satisfying the stated performance objective against uncertainties.

Uncertainty: The uncertainty is assumed to be caused by a *2-tuple* of perturbation parameters [127] (π_1, π_2), in which π_1 pertains to stage I, while π_2 pertains to stage II.

The execution time of each application is considered stochastic due to its dependence on input data. More specifically, the execution time of each application is modeled as a

random variable and assume that we are given a pmf describing the possible execution time values and their probabilities for each combination of application and processor type. That is, the execution time of each application i when executed alone on a single, unloaded (fully dedicated) processor of type j is modeled as a random variable. The list of applications that the user may select from is assumed limited to a set of frequently requested algorithms such as may be found in companies or government research lab environments [26]. Consequently, the execution time random variable for each application is assumed to be well characterized. That is, a pmf is assumed to be available for each application execution time random variable on each processor type (determined by historical, experimental, or analytical techniques [136, 137]). In addition, each application execution time is assumed independent, i.e., there is no inter-application communication. Similarly, the system availability for each processor type j is modeled as a random variable, $\alpha_{\mathbf{j}}$, with a given pmf describing the possible system availability percentages and their probabilities, generated using historical usage data of the heterogeneous computing system.

Let $\hat{\epsilon}$ be a matrix where the $(i,j)^{th}$ element is a random variable modeling the execution time for application i on processor type j, as described above. $\hat{\mathbf{A}}$ is a vector where the j^{th} element is α_j, denoting the availability of processors of type j, also described above. The perturbation parameter for stage I, π_1, is given by $\pi_1 = \{\hat{\epsilon}, \hat{\mathbf{A}}\}$. If a given application is executed on a single processor of a given type, its computation can be modeled based on $\hat{\epsilon}$ and $\hat{\mathbf{A}}$. However, because each application will be executed using parallelism, its computation time is more complex, and its modeling is described in Section 7.4.

Let $\mathbf{\Lambda}$ be the system load fluctuation at runtime [128, 129, 133]. Given $\mathbf{A} = 1 - \mathbf{\Lambda}$ as the *runtime* system availability in stage II, the perturbation parameter for stage II is $\pi_2 = \{\mathbf{A}\}$. In general, the runtime system availability can be higher or lower than the expected system

availability, i.e., $\mathbf{A} \neq \mathbf{E}[\hat{\mathbf{A}}]$. A system is said to be loaded when it is less than 100% available. For example, a system having a load of 30% during a certain period of time, is said to have a $100\% - 30\% = 70\%$ availability for that period of time.

The uncertainty in this chapter is assumed to be caused by the 2-tuple $(\pi_1, \pi_2) = (\{\hat{\epsilon}, \hat{\mathbf{A}}\}, \{\mathbf{A}\})$.

Performance objective: The performance objective has two components, called performance features [127]. These performance features are expressed as a *2-tuple*, denoted (ϕ_1, ϕ_2), in which ϕ_1 is the performance feature related to stage I, and ϕ_2 is the performance feature of interest in stage II.

Let $\mathbf{T}$ be the system makespan, defined as the completion time for an entire collection (or batch) of applications, determined by the maximum of the actual finishing times of all machines for all applications. $\mathbf{T}$ represents the time when the next batch of applications will require resources given an RA heuristic used in stage I and a set of DLS techniques (one for each application) used in stage II. Let $\boldsymbol{\Delta}$ be the system deadline. Then ϕ_1 is defined as $\mathbf{Pr}(\mathbf{T} \leq \boldsymbol{\Delta})$ given $\pi_1 = \{\hat{\epsilon}, \hat{\mathbf{A}}\}$, and $\phi_2 = \{\mathbf{T}\}$ given $\pi_2 = \{\mathbf{A}\}$.

Given a batch of parallel scientific applications executing on the resources of a heterogeneous system, the performance objective in this work is given by the 2-tuple $(\phi_1, \phi_2) = (\{\mathbf{Pr}(\mathbf{T} \leq \boldsymbol{\Delta})\}, \{\mathbf{T}\})$, and is described as: (1) maximize the probability that all applications complete before the system deadline, i.e., `maximize` ϕ_1 given $\hat{\epsilon}$ and $\hat{\mathbf{A}}$ (Stage I); and (2) minimize the system makespan that satisfies the deadline for every given availability in the system, i.e., `minimize` ϕ_2 given $\mathbf{A}$ (Stage II).

7.3.2. Outline of the Proposed Framework. The proposed CDSF for robust execution of scientific applications on heterogeneous uncertain computing systems is schematically illustrated in Fig. 7.2.

Initial mapping conducted in stage I is the problem of finding a static mapping (i.e., one found in an offline planning phase) of a batch of applications onto a set of resources to maximize robustness of the allocation against uncertain input data and system availability, by maximizing the probability that all applications will complete before the deadline, given a pmf for system availability $\hat{\mathbf{A}}$. Runtime application scheduling carried out in stage II is the problem of finding a dynamic scheduling policy for each application that minimizes the parallel time to complete of an application for every given runtime system availability $\mathbf{A}$.

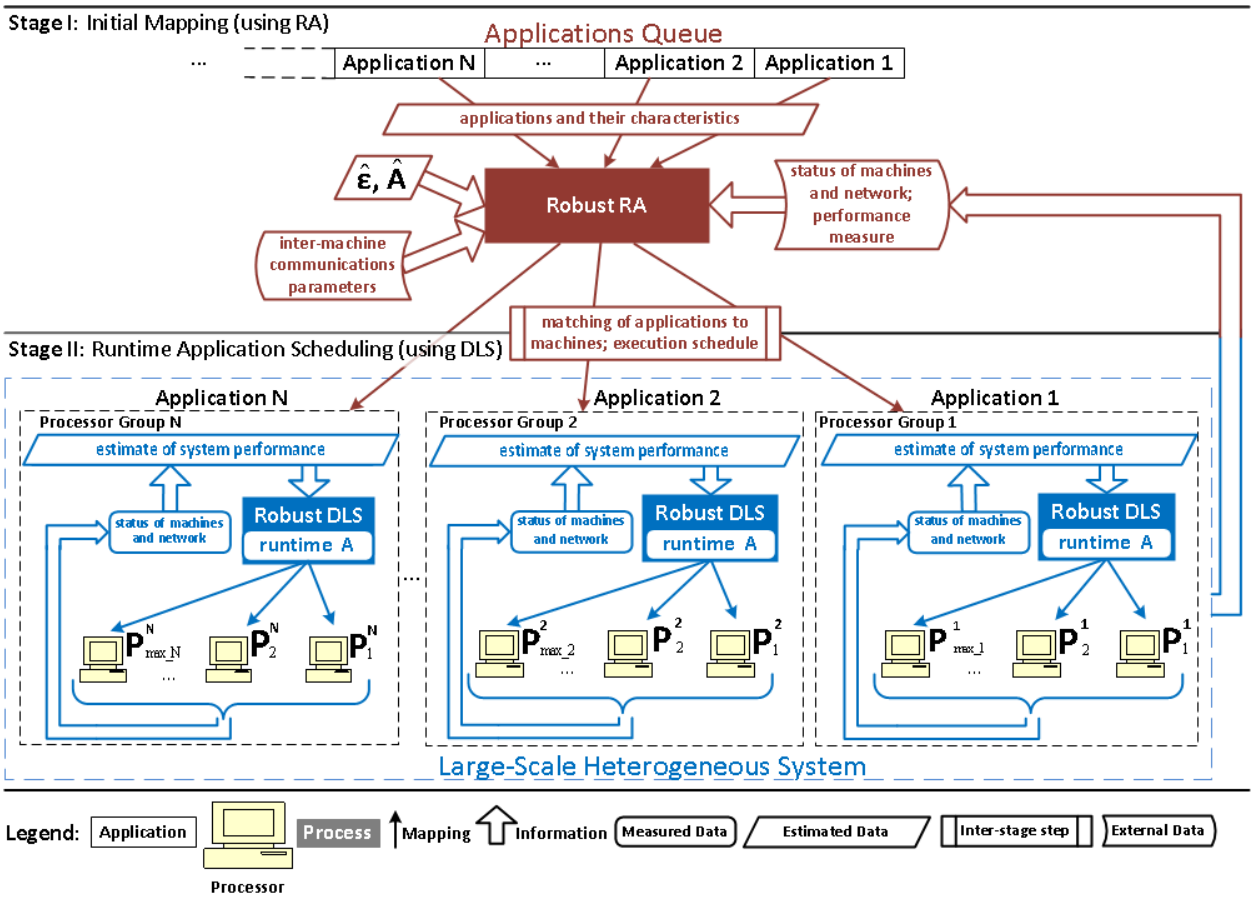


FIGURE 7.2. Schematic illustration of the proposed combined dual-stage framework: a robust resource allocation heuristic is employed in stage I, and robust dynamic loop scheduling techniques are employed in stage II. A number of N applications are mapped onto N groups of processors, which compose into a large-scale heterogeneous system with $\sum_{i=1}^{N} max_i$ processors.

7.3.2.1. *Stage I – Initial Mapping.* Scientific applications arrive at random intervals in the queue of a resource manager, in view of assignment for execution onto any one of a group of resources of a heterogeneous computing system. The applications queue consists of different scientific applications, which can represent different instances of the same application.

As the applications arrive, their assignment to available resources is made in batches. After assignment, an application is placed in the input queue of the resource designated as coordinator (master) of the assigned group of resources. Any required data are staged at the master, in advance of application execution. Let N be the number of applications in the batch. Each application is assumed to be *data parallel* (with no interprocessor communications needed) and to contain large computationally intensive parallel loops.

Robust heuristics are employed for the IM, and the intention is to conduct studies to determine the best heuristic to use in this stage. The best heuristic will provide the most robust mapping of groups of resources to applications, i.e., maximize the probability that an application completes before $\boldsymbol{\Delta}$, assuming a certain system availability $\hat{\mathbf{A}}$.

The resource allocation actions are pre-planned before the actual execution of the applications begins and the goal is to minimize (or to prevent) the immediate effects of uncertain perturbation in $\hat{\epsilon}$ and $\hat{\mathbf{A}}$ on the system makespan $\mathbf{T}$, such that $\phi_1 = \{\mathbf{Pr}(\mathbf{T} \leq \boldsymbol{\Delta})\}$ is maximized. Regardless of the type of allocated resources, once an allocation decision has been made, it cannot be adjusted for a currently executing application. Perturbations during the actual execution of applications are expected and addressed (or compensated for) in stage II via the use of robust DLS techniques.

Let max_i, $i = 1, N$ be the number of resources allocated to application i, and $T^{exp}_{max_i,i}$ be the expected time to complete of application i on max_i processors.

7.3.2.2. *Stage II – Runtime Application Scheduling.* Each application from the current batch of N applications is executed on its group of resources allocated in stage I. A robust DLS technique from the set {FAC, WF, AWF-B, AF} [128, 129, 133] is employed to define the rules for executing an application at runtime. The intention is to conduct studies to determine the best DLS technique to employ for each application in the batch, such that the completion time of an application is minimized for every given runtime system availability $\mathbf{A}$, and consequently, the system makespan is smaller than or equal to the deadline. A single DLS technique may be employed for several applications as several distinct instances of the particular DLS technique. In general, the runtime system availability is expected to be different than the estimated system availability. In this work, it is assumed that $\mathbf{A} \leq \mathbf{E}[\hat{\mathbf{A}}]$.

The most robust DLS technique will provide the best runtime scheduling decisions for executing an application on the allocated group of processors that minimize the system makespan while tolerating a larger degree of perturbation in system availability than the one assumed in stage I. The goal of the robust DLS technique is to detect any runtime perturbation in system availability as soon as it occurs, and to take appropriate scheduling decisions for the remaining unexecuted application iterations. Stage II can, thus, be considered a runtime approach for the detection and recovery from the uncertain effects of the perturbation expected to occur in $\mathbf{A}$, on the performance feature ϕ_2 described earlier.

To guide the scheduling decisions at runtime and to tune the performance of an application, the DLS techniques use runtime estimations of the time required to compute loop iterations. These times are determined using probabilistic analyses and are influenced by the application input data and the availability to compute of the resource executing the iteration(s). The execution of an application using a DLS technique is non-preemptive, and, therefore, the choice of the DLS technique cannot be changed during runtime.

The overhead associated with employing a robust DLS technique is higher than that of a robust RA heuristic. The actions are not pre-planned and are taken dynamically during the application execution, as soon as perturbation occurs. The benefits are expected to, and in general do, compensate the overhead of employing robust DLS techniques.

7.3.3. Questions regarding the CDSF robustness. To claim robustness for a system, the following questions must be answered [138]:

1. What behavior of the system makes it robust?

Answer: The system considered in this work is robust if all applications complete before a common deadline $\mathbf{\Delta}$, given uncertainty in input data (which impacts application execution time) and system availability. The system robustness is achieved via the CDSF employing robust RA and robust DLS in two consecutive stages.

A robust RA heuristic is one that is capable of maximizing the probability that all applications complete before the deadline. A DLS technique is said to be robust if it facilitated the execution of an application in the smallest amount of time, *and* if this time satisfies the deadline when the runtime system availability may vary from the one assumed initially.

2. What uncertainties is the system robust against?

Answer: Given uncertain variations in input data and system availability, application execution times are a known source of uncertainty in the system, and may have a significant impact on the stated performance objective. The uncertainty against which the system considered in this work is assumed to be robust is the 2-tuple (π_1, π_2).

3. How is the system robustness quantified?

Answer: The robustness of the system using the CDSF can be quantified as the joint robustness of the initial mapping in stage I and the runtime application scheduling in stage II. The robustness of stage I is quantified as the joint probability of all applications completing

by the common deadline, i.e., ϕ_1. Let $\mathbf{A_i}$ be the pmf for a given case study. The robustness of stage II is quantified as the percent decrease in weighted system availability that can be tolerated by *all* applications without violating the deadline, i.e., $1 - (\mathbf{E}[\mathbf{A_i}]/\mathbf{E}[\hat{\mathbf{A}}])$ for which $\mathbf{T} \leq \mathbf{\Delta}$.

Let Ψ_1 be the largest robustness value of stage I. Also, let Ψ_2 be the largest robustness value of stage II. The system robustness is quantified as the 2-tuple (Ψ_1, Ψ_2).

7.4. Usefulness of Proposed Framework

7.4.1. System Setup. The assessment of the usefulness of the proposed CDSF requires an investigation of the impact of the different possible RA heuristics and DLS techniques on the performance objective of interest. A small scale example is provided next to illustrate the usefulness of the proposed CDSF. The data that was chosen for this example was used to demonstrate the efficacy of the CDSF. The relevant assumptions for this example are described below.

Consider a heterogeneous system with twelve processors of two types, i.e., $j = 1, 2$: four processors of type 1, and eight processors of type 2. Processors of type 1 are assumed to have a different computational capacity and availability than processors of type 2. Case 1 in Table 7.1 describes the system availability as it was historically collected and aggregated over a given period of time, to form the expected system availability $\hat{\mathbf{A}}$ used in stage I, and is taken as a reference case. Cases 2–4 correspond to systems with decreased weighted availability compared to the reference case, i.e., $\mathbf{E}[\mathbf{A_1}] > \mathbf{E}[\mathbf{A_2}] > \mathbf{E}[\mathbf{A_3}] > \mathbf{E}[\mathbf{A_4}]$. Let p_j be the number of processors of type j, and e_j be the expected availability of processor type

j. The weighted system availability can be calculated as shown in (57).

$$\frac{\sum_{j=1}^{2}(p_j)(e_j)}{\sum_{i=1}^{3} max_i} \tag{57}$$

TABLE 7.1. Processor Availabilities by Type and Weighted System Availabilities. Case 1 corresponds to $\hat{\mathbf{A}}$. Square brackets indicate $1 - (\mathbf{E}[\mathbf{A_i}]/\mathbf{E}[\hat{\mathbf{A}}])$.

	Processor	**Availability** (%)	**Probability** (%)	**Expected Availability** (%)	**Weighted System Availability** (%)
Case 1 ($\mathbf{A}_1 = \hat{\mathbf{A}}$)	Type 1	75	50	87.50	75.00
		100	50		
	Type 2	25	25	68.75	
		50	25		
		100	50		
Case 2 ($\mathbf{A}_2$)	Type 1	50	90	52.50	53.87 [28.17]
		75	10		
	Type 2	33	45	54.55	
		66	45		
		100	10		
Case 3 ($\mathbf{A}_3$)	Type 1	52	50	60.58	51.92 [30.77]
		69	50		
	Type 2	17	25	47.60	
		35	25		
		69	50		
Case 4 ($\mathbf{A}_4$)	Type 1	33	75	41.25	50.42 [32.77]
		66	25		
	Type 2	20	50	55.00	
		80	25		
		100	25		

A batch of $N = 3$ applications is considered, having different sizes and serial/parallel component ratios (see Table 7.2). Serial iterations can only be executed on a single processor and parallel iterations can be executed on multiple processors of the same type. The system

deadline is $\mathbf{\Delta} = 3,250$ time units, and was chosen to help illustrate the differences between using intelligent stages versus naïve stages in the dual-stage framework.

TABLE 7.2. Characteristics of a batch of applications

App.	**# Serial iterations**	**# Parallel iterations**	**% Serial iterations**	**% Parallel iterations**
1	439	1024	30	70
2	512	2048	20	80
3	216	4096	5	95

The single processor execution times for each of the three applications on each of the two processor types are represented as pmfs; this is the $\hat{\epsilon}$ used in stage I. For this study, the pmfs were generated by sampling a normal distribution with the mean values (μ) shown in Table 7.3. Each normal distribution used a standard deviation (σ) of one-tenth of its mean value, i.e., $\sigma = \frac{1}{10}\mu$. These values were considered to be the expected serial times required for each application to execute on one processor of a given type and were chosen to highlight the usefulness of the proposed CDSF.

It is assumed that all applications must be assigned and that they are assigned to a power-of-2 number of processors of one type. For application i on processor type j, let T_{ijx} be the time of pulse x in the pmf, s_{ij} and p_{ij} be the serial and parallel fractions, respectively, and n_{ij} be the number of processors. Let T_{ijxn} be the pulse in the parallel execution time pmf of application i assigned to n_{ij} processors of type j. This specific parallel execution time pmf is obtained by recalculating each pulse of the single processor execution time pmf according to (58).

$$T_{ijxn} = (s_{ij}T_{ijx}) + (p_{ij}T_{ijx})/n_{ij} \tag{58}$$

For each pulse x, the time associated with T_{ijxn} will differ from T_{ijx}, while the probability will remain the same.

TABLE 7.3. Normal distribution mean values for single processor execution times of each application on each processor type

Processor	$T^{exp}_{1,1}$	$T^{exp}_{1,2}$	$T^{exp}_{1,3}$
Type 1	1,800	2,800	12,000
Type 2	4,000	6,000	8,000

Once the pmf modeling the parallel execution time of an application on a certain number of processors of one type is calculated, it is multiplied with the pmf modeling the historical system availability ($\hat{\mathbf{A}}$) of processors of that type (α_j), to determine the pmf used in stage I to calculate the resource allocation robustness values. To calculate the probability that for a given resource allocation, an application will meet the common deadline $\mathbf{\Delta}$, each pulse in this resulting PMF corresponding to a time *less* than the deadline is summed together. Due to the fact that each application's finishing times are independent, the probability that the entire system will complete by the common deadline is given their joint probability of completing by $\mathbf{\Delta}$.

To demonstrate the benefits and usefulness of the proposed CDSF for allocating the twelve heterogeneous processors of two types, executing with uncertainties shown in Table 7.1, to the three applications, four scenarios have been identified: 1) naïve IM−naïve RAS, 2) *robust* IM−naïve RAS, 3) naïve IM−*robust* RAS, and 4) *robust* IM−*robust* RAS.

In all scenarios, the IM problem is solved assuming a system availability equal to $\hat{\mathbf{A}} = \mathbf{A}_1$, while the RAS problem is solved assuming the runtime system availability $\mathbf{A}$, is one value from the set $\{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4\}$.

In naïve IM, a simple load balancing technique is used to allocate an equal share of the available processors to each application. The load balancing allocation with the highest

probability that all applications will complete before the deadline was chosen. In the system as described by the example, the load balancing technique allocated resources as described in Table 7.4. Given this resource allocation, the system has a 26% chance of completing the applications before the deadline, i.e., $\mathbf{Pr}(\mathbf{T} \leq \mathbf{\Delta}) = 26\%$.

TABLE 7.4. Resource allocation for naïve and robust IM

RA	**App.** i	**Proc. type** j	**# Procs** max_i
naïve IM	1	2	4
	2	1	4
	3	2	4
robust IM	1	1	2
	2	1	2
	3	2	8

In the robust IM case, all possible resource allocations are compared and the one with the highest probability of all applications completing before the system deadline is chosen. This results in resources being allocated as shown in Table 7.4. Given this resource allocation, the system has a 74.5% chance of completing the applications before the deadline, i.e., $\mathbf{Pr}(\mathbf{T} \leq \mathbf{\Delta}) = 74.5\%$. It is important to note that this exhaustive search for the robust resource allocation is only feasible in the case of the small demonstrative example. More advanced and scalable RA heuristics are required for larger problem sizes, and our future work will include designing such robust RA heuristics.

Given the two resource allocations, naïve and robust, each application's expected completion time calculated in stage I is shown in Table 7.5. These values were obtained by taking the expected value of the pmf relating to the assigned resources for each application. It is interesting to note that $T^{exp}_{max_2,2}$ is larger in the robust IM than in the naïve IM.

TABLE 7.5. Parallel pmf estimated values of applications completion times for naïve and robust IM (in time units)

RA	$T^{exp}_{max_1,1}$	$T^{exp}_{max_2,2}$	$T^{exp}_{max_3,3}$
naïve IM	3800.02	1306.39	4599.76
robust IM	1365.46	1959.59	2699.86

In naïve RAS, straightforward parallelization is employed for each application to schedule its iterations in equal shares, which are then assigned to processors in a single step. This technique is called STATIC.

In robust RAS, a DLS technique from the set {FAC, WF, AWF-B, AF} is employed to execute an application on its allocated resources. The DLS techniques implement dynamic load balancing mechanisms based on probabilistic analyses to ensure minimal impact of runtime uncertainties on the application performance. For a given application and a runtime system availability, the DLS technique resulting in the smallest parallel execution time that satisfies the system deadline is considered best.

The usefulness of the proposed CDSF is based on the hypothesis that any of the first three scenarios will result in solutions that tolerate much *less* perturbations variations in the overall system, and therefore, are *less* robust. Thus, the fourth scenario (robust IM−robust RAS) is expected to be superior to the first three scenarios. The CDSF allows investigation of the overall degree of tolerable uncertainty for which the stated performance objective is satisfied, at the level of each individual application and for the entire batch of applications executing on the heterogeneous system.

7.4.2. Example Scenarios.

7.4.2.1. *Scenario 1 – Naïve IM−naïve RAS.* In this scenario, each stage employs naïve heuristics to allocate resources to each of the three applications, namely simple load balancing and STATIC, respectively. The application execution times are shown in Fig. 7.3.

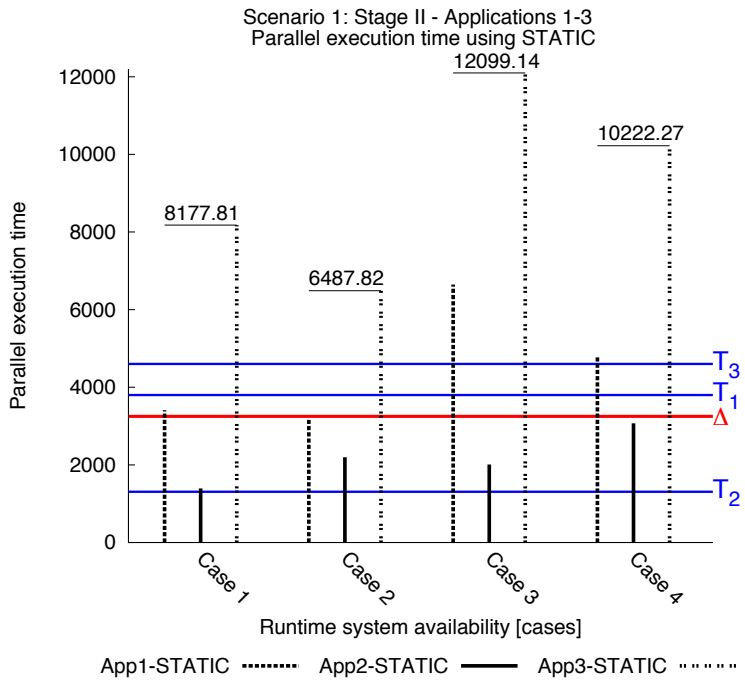


FIGURE 7.3. Scenario 1) Stage I: resource allocation using simple load balancing, Stage II: straightforward parallelization using STATIC. $\boldsymbol{\Delta} = 3,250$ time units is the system deadline. $\mathrm{T}_1 = 3,800.02$ time units, $\mathrm{T}_2 = 1,306.39$ time units, and $\mathrm{T}_3 = 4,599.76$ time units where $T_i = T^{exp}_{max_i,i}$ (see Table 7.5).

For the given resource allocation and application scheduling, $\phi_1 = 26\%$ and $\phi_2 > \boldsymbol{\Delta}$ for all system availability cases. This scenario shows that a naive RA policy in stage I and a straightforward parallelization in stage II cannot prevent the violation of the system deadline given the system availability cases considered. Therefore, the system is not robust.

7.4.2.2. *Scenario 2 – Robust IM−naïve RAS.* The interest in this scenario is to investigate how much perturbation can be tolerated when only the robust IM ensures a certain level of robustness for all applications. Thus, an optimal RA heuristic is employed in stage I to allocate resources to each application, such that they all complete before $\boldsymbol{\Delta}$, with a probability of 74.5% assuming the system availability equals $\hat{\mathbf{A}}$ (see Table 7.1). The use of robust RA is recommended when the assumptions made using a naïve RA are not sufficient to guarantee the satisfaction of $\boldsymbol{\Delta}$, as is the case in the previous scenario. Each application is parallelized using STATIC in stage II, and their execution times are plotted in Fig. 7.4.

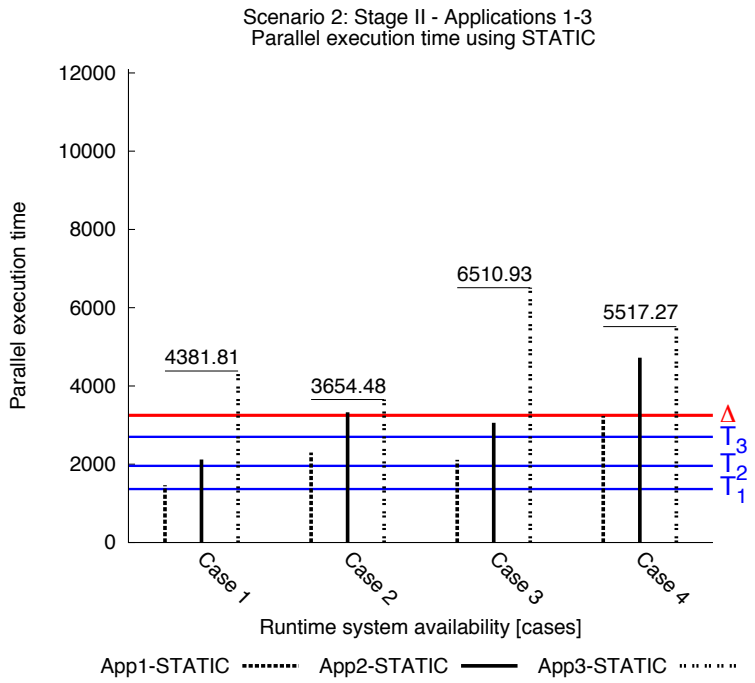


FIGURE 7.4. Scenario 2) Stage I: resource allocation using optimal RA, Stage II: straightforward parallelization using STATIC. $\mathbf{\Delta} = 3,250$ time units is the system deadline. $\mathrm{T}_1 = 1,365.46$ time units, $\mathrm{T}_2 = 1,959.59$ time units, and $\mathrm{T}_3 = 2,699.86$ time units where $T_i = T^{exp}_{max_i,i}$ (see Table 7.5).

Given the robust resource allocation, $\phi_1 = 74.5\%$. This means that the system makespan has a higher probability of meeting the deadline when $\pi_2 = \mathbf{E}[\mathbf{A_1}]$ than in the previous scenario. However, Fig. 7.4 shows that the performance of each application using STATIC degrades with decreasing system availability *after* the RA decisions have been taken in stage I, and $\phi_2 > \mathbf{\Delta}$ for all four system availability cases. Thus, it can be stated that the system is not robust.

7.4.2.3. *Scenario 3 – Naïve IM−robust RAS.* In this scenario, the interest is to investigate how much perturbation can be tolerated when only the DLS policy ensures a certain level of robustness for each application. Thus, the allocation decisions are made in stage I according to a naïve RA heuristic. A robust DLS technique, i.e., FAC, WF, AWF-B, or AF, is employed in stage II, and uses knowledge obtained during the execution of the application about the system, to guide the scheduling decisions in such a way that the performance of the

application suffers a minimal degradation with varying (decreasing) system availability. The application execution times for this scenario are illustrated in Fig. 7.5.

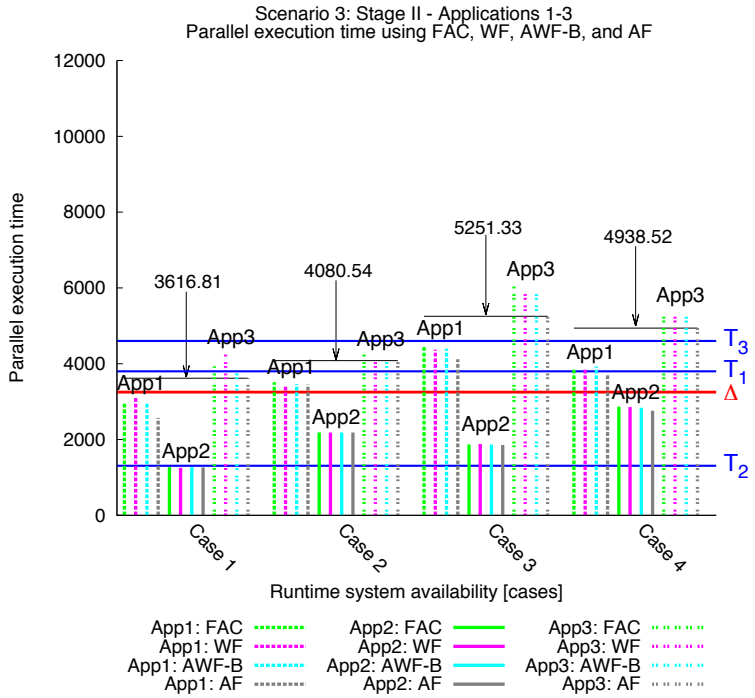


FIGURE 7.5. Scenario 3) Stage I: resource allocation using simple load balancing, Stage II: robust DLS using FAC, WF, AWF-B, and AF. $\mathbf{\Delta} = 3,250$ time units is the system deadline. $T_1 = 3,800.02$ time units, $T_2 = 1,306.39$ time units, and $T_3 = 4,599.76$ time units where $T_i = T^{exp}_{max_i,i}$ (see Table 7.5).

For the naïve resource allocation the probability of all applications completing before $\mathbf{\Delta}$ is $\phi_1 = 26\%$. Even when the most robust DLS technique is used in stage II, one can note that certain applications finish earlier than others. This causes the violation of the system deadline, as it is the case for application 3 in case 1, and applications 1 and 3 in cases 2-4. Given that $\phi_2 > \mathbf{\Delta}$ for all four system availability cases, a more robust RA heuristic is, hence, required in stage I to complement the robust DLS technique used in stage II. It can be stated that the system in this scenario is not robust.

7.4.2.4. *Scenario 4 – Robust IM−robust RAS.* The previous two scenarios show an improvement over the first scenario. This improvement is, however, insufficient to ensure that

the system deadline is met for all applications. Therefore, in this scenario the largest amount of tolerable perturbation in system availability is considered, while ensuring that the system deadline is met for the *entire* batch of applications. This scenario, is also referred to as the scenario that best illustrates the usefulness of the proposed CDSF.

The robust IM from the second scenario (robust IM−naïve RAS) is employed in stage I to allocate a more suitable set of resources to each application than in scenarios 1 and 3. Just as in scenario 3, robust DLS algorithms are employed in stage II to minimize the impact on the application performance assuming unforeseen variation in the system availability, and to support the probability given by the robust IM in stage I that all applications complete before the system deadline.

The application execution times for this scenario are shown in Fig. 7.6. One can note that the system deadline is met for all applications when the weighted system availability decreased by 28.17% (case 2) and 30.77% (case 3), respectively, compared to that assumed in stage I (case 1). When the weighted system availability decreased by 32.77% (case 4), the deadline is met for application 1, while it is violated for application 2 using any DLS technique and for application 3 using FAC, WF or AWF-B. This indicates that in case 4 and for application 3, AF is more robust than FAC, WF, or AWF-B when executing on the resources of type 2 allocated in stage I, with an expected availability of 55% for this processor type. Therefore, the system is said to be robust for system availability cases 1-3, while it is not robust for case 4, and the robustness value for stage I is $\Psi_1 = 74.5\%$.

It is important to note that the above observations are valid only for the combination of type 1 and type 2 processors availabilities shown in Table 7.1 (third column), and not for any general combination that may result in the weighted system availability values in Table 7.1 (sixth column).

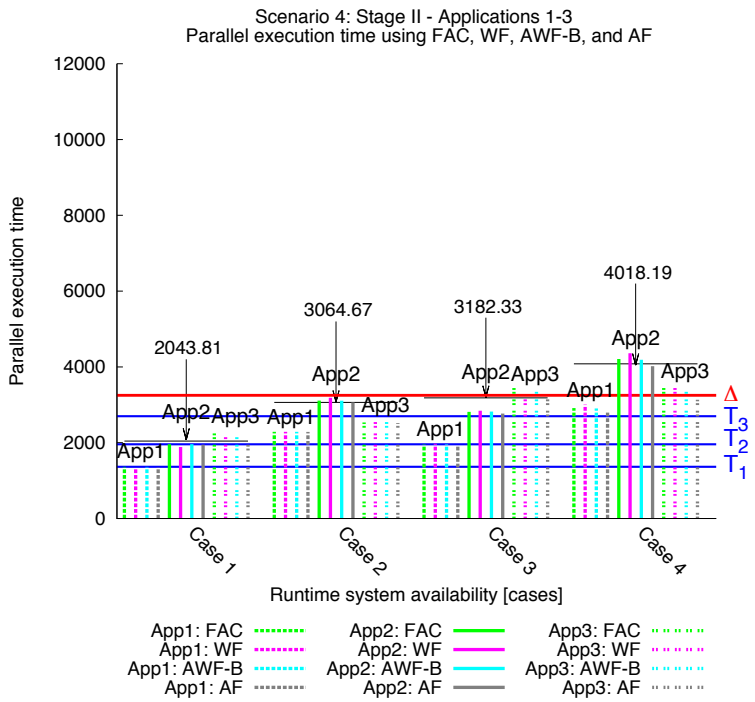


FIGURE 7.6. Scenario 4) Stage I: resource allocation using optimal RA, Stage II: robust DLS using FAC, WF, AWF-B, and AF. $\mathbf{\Delta} = 3,250$ time units is the system deadline. $T_1 = 1,365.46$ time units, $T_2 = 1,959.59$ time units, and $T_3 = 2,699.86$ time units where $T_i = T^{exp}_{max_i,i}$ (see Table 7.5).

TABLE 7.6. Scenario 4) DLS techniques providing best application performance and meeting the system deadline for all cases of system availability

Application	Case 1	Case 2	Case 3	Case 4
1	WF	AF	**AF**	AF
2	WF	WF	**AF**	–
3	AF	AF	**AF**	AF

Table 7.6 shows the DLS techniques that result in the best application performance and that at the same time satisfy the system deadline. It can be noted that from all the cases considered in Table 7.1 the largest tolerable decrease in overall system availability compared to the reference case is 30.77% (case 3) and the system deadline is met for all applications. The most robust DLS technique in this case is AF for all applications (cf. Table 7.6, fourth column), and, hence, the robustness value for stage II is $\Psi_2 = 30.77\%$.

The system robustness for this scenario is quantified as $(\Psi_1, \Psi_2) = (74.5\%, 30.77\%)$.

7.5. Conclusions

The goal of this chapter was to study the allocation of resources to applications and the completion of their execution before a system deadline in the presence of uncertainty in input data and in system availability. A CDSF has been proposed towards this goal. The framework enables the robustness of resource allocation used in a first stage to be enhanced via the use of dynamic loop scheduling techniques used in a second stage. The usefulness of the framework has been demonstrated via a small scale, illustrative example.

Extensions to this work may consider the impacts of employing previously developed static [26] and dynamic [139] stochastic resource allocation heuristics in stage I, and other DLS techniques in stage II [134]. Designing novel robust and scalable resource allocation heuristics to be employed in stage I is also a noteworthy extension to the present work. A study of the factors to be considered in guiding the choice of heuristics used in either stage is another potential extension of interest to the current work. Exploring the possible correlation between the availabilities for different processor types on the overall robustness of the system is also of interest for future work because it would help in better quantifying the system robustness.

In Chapter 8, a larger scale problem is used to demonstrate the necessity of more advanced RA heuristics in stage I. This larger scale problem incorporates more applications, i.e., in a larger batch or in multiple batches, on a larger computing system, i.e., one with more processors and processor types. Probabilistic studies are performed on this larger problem to determine the benefit of Stage I on a range of application and system parameters.

CHAPTER 8

Heuristics for Robust Allocation of Resources to Parallel Applications with Uncertain Execution Times in Heterogeneous Systems with Uncertain Availability

8.1. Introduction

Today's computing systems are often heterogeneous in nature, comprised of machines with differing computational capabilities to satisfy the diverse computational requirements of applications [141, 142]. For example, a mixture of general purpose and programmable digital machines, along with application-specific systems-on-a-chip, were shown to solve parallel jobs with real-time constraints [143, 144]. The scheduling of applications on such heterogeneous systems has been shown, in general, to be NP-complete [5]. Scheduling decisions become more difficult in systems with uncertain processor availability (this can be due to system jitter/noise [126], or the time sharing of resources [145]) and with application execution times that are modeled as stochastic due to uncertain input data [26].

A batch of scientific *moldable* parallel applications with stochastic execution times are considered, where a moldable parallel application is one that differs in execution time as a function of the numbers of processors (determined by the scheduler) on which it executes.

This work was performed jointly with the full list of co-authors available in [140]. This work was supported by the National Science Foundation (NSF) under grant numbers CNS-0905399, CCF-1302693, and IIP-1034897; the CSU George T. Abell endowment; and the German Research Foundation in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing." This research utilized the CSU ISTeC Cray HPC System supported by NSF Grant CNS-0923386.

These applications need to be allocated resources from a set of heterogeneous processor types, where the processor types are heterogeneous in both their computational capabilities (affecting the application execution times) and the number of processors available. All applications in the batch should be scheduled in such a way as to finish executing close to a given *makespan goal* (a soft version of the deadline from Chapter 7).

To allocate resources to applications, a new batch scheduler is proposed. The batch scheduler must allocate resources in the presence of the two uncertainties of application execution times and system availability. To minimize the impact of the two sources of uncertainty on achieving the makespan goal, our resource allocations should be robust against these uncertainties. To accomplish this goal, a model that combines the impact on performance of two sources of uncertainties into a single performance metric of *robustness* [138] is introduced, where in this chapter robustness is defined as the probability that a batch of applications finishes by the given makespan goal. Three iterative-greedy resource allocation heuristics that use this measure of robustness were designed. The allocation decisions that need to be made for each application are: (1) on what processor type to run, and (2) on how many processors of a given type to run. The resource allocation heuristics are designed to maximize robustness by using stochastic knowledge of the uncertain execution times and uncertain system availability to intelligently allocate processors to applications.

This chapter is based on the first stage of the dual-stage optimization framework introduced in Chapter 7 (based on[9]). In the first stage, which is the focus of this chapter, a batch of applications is allocated resources from a set of heterogeneous processor types. The second stage, which is not included in this chapter, performs fine-grain runtime optimization for each application given the allocated resources from the first stage. The system and the flow of information is shown in Fig. 8.1. In this chapter, novel resource allocation heuristics

are designed and evaluated. The heuristics presented here utilize a more realistic parallel execution time model and a much larger system than those discussed in Chapter 7.

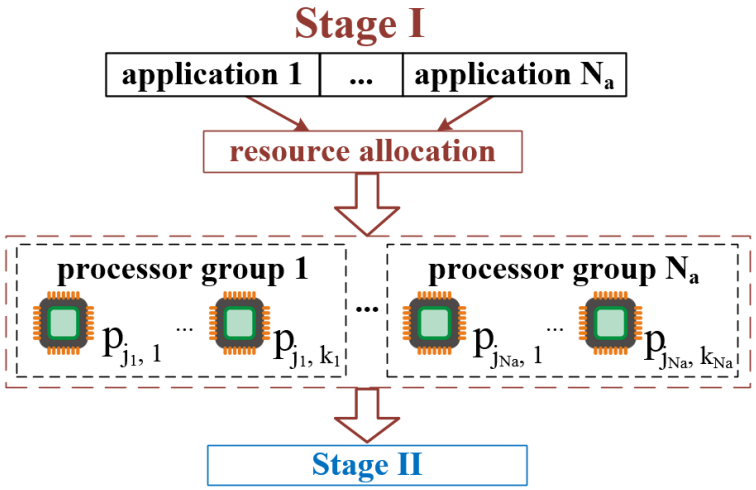


FIGURE 8.1. Dual-stage optimization framework with a focus on Stage I. In the first stage, a batch of N_a scientific moldable parallel applications are allocated resources from heterogeneous processor types according to a given resource allocation heuristic. In Stage II, a runtime optimization is performed for each application using the allocated resources from Stage I.

Related prior work on resource allocation and scheduling for heterogeneous systems has occurred in the areas of heuristic optimization and modeling. Uncertainties in application execution times were taken into account in [146–148]. These uncertainties lead to robustness as a performance measure (e.g., [26, 138]). The uncertainty in the availability of computing resources was studied as system noise [126] and operating system (**OS**) overhead [149]. Iterative-greedy heuristics have been shown to perform well for scheduling problems [150] and balanced resource allocation techniques are often used in practice [151]. This chapter differs from previous work in that new heuristics are designed that take into account the uncertainty in system availability as well as the uncertainty in application execution times.

In this chapter, the following contributions are made:

- The design of a model for moldable parallel applications with stochastic execution times running in a heterogeneous computing environment.
- A new robustness model and measure dealing with, and combining, two sources of stochastic uncertainties for use in the resource allocation of processors to applications and the performance evaluation of said resource allocations.
- The design and analysis of three novel iterative-greedy heuristics across three different platforms of a varying number of processor types compared to two reference heuristics and an upper-bound.

The rest of the chapter is organized as follows. The system model is described in Section 8.2. In Section 8.3, the developed heuristics are presented in detail. The parameters used for the analysis are given in Section 8.4 with the analysis results being shown in Section 8.5. Finally, concluding remarks and a brief description of future work are summarized in Section 8.6.

8.2. System Model

8.2.1. Batch Scheduler. The proposed model for the batch scheduler is given in Fig. 8.2. There are three times of interest shown: the time the assignment of resources for batch z starts (t_0), the time batch z starts executing (t_1), and the time that batch z finishes executing (t_2). Thus, the resource allocation heuristic executes from t_0 to t_1. Without a loss of generality, a workload of one batch is assumed so the subscript denoting the batch number will be dropped from the notation. This research is applicable for any number of subsequent batches.

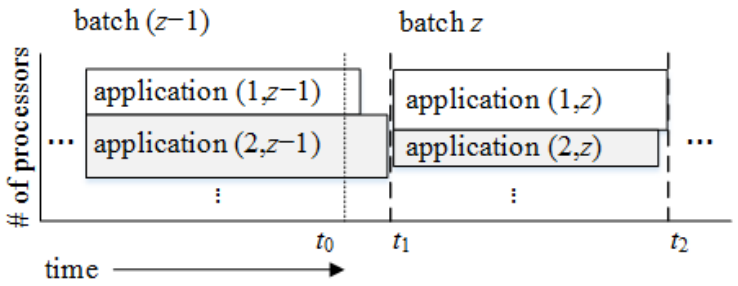


FIGURE 8.2. Proposed batch scheduler model. At some time t_0, the applications in batch z will be assigned resources using a given heuristic. At t_1, the last application of batch $(z-1)$ finishes executing and batch z can begin executing using the resources allocated by a given heuristic. Time t_2 denotes when batch z finishes executing and batch $(z+1)$ begins executing, *ad infinitum*.

Given a batch of $\boldsymbol{N_a}$ moldable parallel applications, a scheduling heuristic is used to allocate computing resources to each application. The applications are assumed to be independent and without precedence constraints. The allocation decision that needs to be made for application i is twofold. First, application i must be assigned to one of $\boldsymbol{N_p}$ processor types (denoted processor type j). Second, the application must be allocated a number of processors, k, of processor type j. Let $\boldsymbol{I}$ be a vector of length N_a representing a complete resource allocation, where the i^{th} entry is a (j,k) tuple that represents application i being assigned to k processors of type j (i.e., $I[i] = (j_i, k_i)$).

To avoid fragmentation of the system resources, all N_a applications start executing at the same time (i.e., each batch of applications can leverage the ability of the entire set of system resources), shown as t_1 in Fig. 8.2. In addition to avoiding fragmentation, other scenarios in which this holds true are in scatter-gather operations and MapReduce [152]. Because all applications in the batch start executing at the same time, the next batch of applications cannot start executing until the current one is finished (t_1 and t_2 in Fig. 8.2). This implies that the scheduling heuristics should try to allocate resources so applications

finish at approximately the same time to prevent the waste of system resources (i.e., idle machines).

8.2.2. Applications.

8.2.2.1. *Heterogeneity.* The execution time for each application i in the batch of applications differs across heterogeneous processor types. For a fixed number of processors, if processor type A is faster than processor type B for a given application, it is not necessarily true that processor type A is faster than processor type B for *all* applications. The application execution time distributions are assumed to be known *a priori*. This information, in practice, can be obtained by analytical, historical, or experimental techniques [26, 137].

8.2.2.2. *Parallel Model.* We use Downey's parallel speedup model [153] to describe how the execution times of real applications change as a function of the number of processors allocated. Given k processors, the speedup of an application is denoted $S(k)$. If $S(k) = x$ for a given resource allocation, application i will execute x times as fast in parallel on k processors ($k > 1$) of type j than if it was run serially (i.e., $k = 1$). This model takes into account the different finishing times of each processor and the execution time of the application is determined by the longest running processor.

8.2.3. Uncertainties.

8.2.3.1. *Application Execution Time.* Each application on a given processor type has an uncertain execution time, for example because of differing input data. Because the system is modeled as heterogeneous, each application has a probability distribution describing its execution time on each processor type. These execution time distributions are assumed to describe the *serial execution times* of the applications. Let $\boldsymbol{\mathcal{T}_{i,j}}$ be a random variable describing the serial execution time of application i on processor type j. To obtain the

parallel execution time distribution, the Downey model is used by scaling the time axis of the probability distribution of $\mathcal{T}_{i,j}$ by $\frac{1}{S(k)}$.

8.2.3.2. *System Slowdown.* In addition to the uncertainty in application execution times, the system availability of each processor type is uncertain. The set of system resources available are N_p *processor types*, where processor type j has $\boldsymbol{n_j}$ processors. Each processor type is assumed to have an associated *system slowdown* (defined as the reciprocal of the system availability). This could be due to OS jitter or system noise [126], or the time sharing of resources [145].

Because the slowdown of a processor type is uncertain, slowdowns are modeled probabilistically. That is, given historical data of a processor type, a probability distribution describing the likelihood of a given system slowdown occurring is obtained. Let $\boldsymbol{\mathcal{S}_j}$ be a random variable describing the slowdown of processor type j. A system slowdown of x will scale the execution time of the application by x.

8.2.3.3. *Combining Uncertainties.* Let $\boldsymbol{\mathcal{X}_{i,j,k}}$ be a random variable describing the final execution time distribution of application i running on k processors of type j. To obtain the *final execution time distribution*, multiply the *parallel execution time distribution* by the *system slowdown distribution* [9], $\mathcal{S}_j$, as defined by (59). Once $\mathcal{X}_{i,j,k}$ is obtained, it can be used to calculate the probability that a given resource allocation will result in application i finishing by a given time on k processors of type j. The multiplication of the distributions was performed discretely in the simulation code, not using a closed form solution (as one does not exist for the types of distributions used).

$$\mathcal{X}_{i,j,k} = \mathcal{S}_j \frac{\mathcal{T}_{i,j}}{S(k)} \tag{59}$$

8.2.4. Robustness. A "robust" resource allocation is defined as a resource allocation that mitigates the effect of uncertainties on a given performance objective. To claim robustness for a system, the following three questions must be answered [138]: **(1)** What performance feature makes the system robust? **(2)** What uncertainties is the system robust against? **(3)** How is robustness quantified?

To answer the robustness questions, the *makespan goal*, $\boldsymbol{\Delta}$, is defined as a target time for all applications to attempt to complete executing by, as well as a time used to calculate the probability that a given resource allocation will complete by a given Δ. Therefore, the performance feature that makes the system robust is applications completing by the makespan goal, Δ. The system is robust against uncertainties in application execution times and the uncertainties in system slowdown.

Let $\boldsymbol{P(i,(j,k))}$ be the probability that application i allocated resources (j,k) completes by the makespan goal Δ, obtained by evaluating the cumulative distribution function (**cdf**) of $\mathcal{X}_{i,j,k}$ at Δ. The robustness of the resource allocation I, denoted $\Psi(I)$, is quantified in (60).

$$\Psi(I) = \min_{i=1..N_a} P(i, I[i]) \tag{60}$$

8.2.5. Formal Problem Statement. A batch of N_a moldable parallel applications needs to be allocated resources from N_p heterogeneous processor types, where processor type j has n_j processors. The parallel characteristics and serial execution time distribution for each application i on each processor type j is known. The system slowdown distribution for each processor type j is also known. Within the constraint of an allocation not exceeding the total number of processors of each type and the constraint that each application can only be assigned processors of one type, the goal of the resource allocation heuristics is to

find a resource allocation I to maximize $\Psi(I)$, given in (61) as the performance objective $\boldsymbol{\rho}$.

$$\rho = \max \Psi(I) = \max \min_{i=1..N_a} P(i, I[i]) \tag{61}$$

8.3. Heuristics

8.3.1. Processor Balance.

8.3.1.1. *Overview.* The goal of the *processor balance* heuristics is to give equal resources (i.e., processors) to each application. Let the total number of processors in the system be $\boldsymbol{\tau}$ (i.e., $\tau = \sum_{j=1}^{N_p} n_j$) and let $\boldsymbol{M}$ be the average number of processors per application in the system (i.e., $M = \frac{\tau}{N_a}$). It is assumed that M is an integer value and each n_j is a multiple of M for each processor type. These assumptions allow the comparison of the proposed robustness floor heuristics to the processor balance heuristics, but the assumptions are not necessary for the robustness floor heuristics.

The processor balance heuristics will then split the total number of resources into N_a blocks of M processors, where each block is comprised of processors of a single type. This reduces the dimensionality of the problem to just assigning an application to a processor type. The two variants of how this assignment is accomplished are *random* and *smart*. These two heuristics will be used as a comparison to the performance of the robustness floor heuristics, as a layperson might assign a fair share of resources to each user [151].

8.3.1.2. *Random.* The *processor balance – random* (**PB-R**) variant of the processor balance heuristic assigns the N_a applications to the N_a groups of processors randomly. That is, for each of the N_a groups of M processors, randomly select an application to be assigned. The application and processor group are removed from further allocation decisions and the process is repeated until no applications remain.

8.3.1.3. *Smart.* Unlike the PB-R variant, *processor balance – smart* (**PB-S**) greedily assigns applications to processor groups. Each of the N_a groups of M processors has an associated processor type j. Randomly select a group and assign the application i that maximizes $P(i,(j,M))$. The application and processor group are removed from further allocation decisions and the process is repeated until no applications remain.

8.3.2. Robustness Floor.

8.3.2.1. *Overview.* The proposed robustness floor algorithm, shown as pseudocode in Fig. 8.3, is an iterative-greedy heuristic with three variants. For each algorithm iteration l, the robustness floor algorithm performs a binary search over the minimum value of robustness an application should achieve, $\boldsymbol{\Omega_l}$ (i.e., each application should have at least a probability of Ω_l of completing by the makespan goal). Let $\boldsymbol{\Pi(\Omega_l)}$ be a function that returns a matrix where the i,j element gives the minimum number of processors k required for application i on processor type j to meet $P(i,(j,k)) \geq \Omega_l$. Also let $\boldsymbol{greedy(\Pi(\Omega_l))}$ be a greedy heuristic that takes the application-processor pairings from $\Pi(\Omega_l)$ and returns a complete resource allocation I. These greedy heuristics are described in Subsections 8.3.2.2 to 8.3.2.4. After the greedy heuristic returns a resource allocation I, the function $\boldsymbol{v(I)}$ returns true or false depending on whether I is valid (i.e., every application is assigned a set of resources). If $\boldsymbol{\lambda_l}$ is the current binary search residual, defined as $\Omega_l - \Omega_{l-1}$, the next robustness floor value will be changed by $\frac{\lambda_l}{2}$. If $v(I)$ returns true, then the robustness floor of the next iteration $\Omega_l = \Omega_l + \frac{\lambda_l}{2}$, but if $v(I)$ returns false, $\Omega_l = \Omega_l - \frac{\lambda_l}{2}$. Finally, let $\boldsymbol{\Lambda}$ be the minimum residual considered (i.e., when $\lambda_l < \Lambda$, stop iterating).

8.3.2.2. *Min-Min Processors.* The min-min processors greedy heuristic is a two-stage greedy heuristic (e.g., [25, 29]) that describes one of the *greedy* functions from line 4 in Fig. 8.3. In the first stage, for each application i, find the processor type j in row i of the matrix

1: $\Omega_l = 1.0$
2: $\Omega_{l-1} = 0.0$
3: **repeat**
4: $I = greedy(\Pi(\Omega_l))$
5: $\lambda_l = \Omega_l - \Omega_{l-1}$
6: $\Omega_{l-1} = \Omega_l$
7: **if** $v(I) ==$ **true then** $\Omega_l = \Omega_l + \frac{\lambda_l}{2}$
8: **else if** $v(I) ==$ **false then** $\Omega_l = \Omega_l - \frac{\lambda_l}{2}$
9: **until** $\lambda_l < \Lambda$
10: **return** I

FIGURE 8.3. Robustness floor algorithm

returned by $\Pi(\Omega_l)$ that uses the minimum number of processors. In the second stage, from the application to processor type pairs found in the first stage, assign the application, $\boldsymbol{i_{min}}$, to the processor type, $\boldsymbol{j_{min}}$, that uses the minimum number of processors, $\boldsymbol{k_{min}}$. Remove application i_{min} from further allocation decisions and decrement the number of processors of type j_{min} by k_{min}. Repeat the two stages until all applications are assigned (i.e., $v(I) =$ true) or until there are not enough remaining processors to make any more allocations (i.e., $v(I) =$ false).

By utilizing the min-min processors greedy heuristic in step 4 of the robustness floor algorithm, the *robustness floor min-min* (**RF Min-Min**) heuristic is formed.

8.3.2.3. *Min-Max Processors.* The min-max processors heuristic is a two-stage greedy heuristic similar to the min-min processors heuristic in that the first stage is the same. In the second stage, however, from the application to processor type pairs found in the first stage, assign the application, $\boldsymbol{i_{max}}$, to the processor type, $\boldsymbol{j_{max}}$, that uses the maximum number of processors, $\boldsymbol{k_{max}}$. Remove application i_{max} from further allocation decisions and decrement the number of processors of type j_{max} by k_{max}. Repeat the two stages until all applications are assigned (i.e., $v(I) =$ true) or until there are not enough remaining processors to make any more allocations (i.e., $v(I) =$ false).

The intuition behind assigning those applications that need more processors first is that as more applications are assigned, it is harder to find room for those requiring more processors to meet the robustness floor. By utilizing the min-max processors greedy heuristic in step 4 of the robustness floor algorithm, the *robustness floor min-max* (**RF Min-Max**) heuristic is formed.

8.3.2.4. *Duplex.* The final greedy heuristic is a combination of the min-min and min-max processors heuristics, referred to as duplex. At step 4 in Fig. 8.3, duplex runs both min-min and min-max processors at each iteration and returns I such that the performance objective, ρ, is maximized. By utilizing the duplex greedy heuristic in step 4 of the robustness floor algorithm, the *robustness floor duplex* (**RF Duplex**) heuristic is formed.

8.4. Simulation Parameters

8.4.1. Overview. The following section describes parameters that are used only to conduct a simulation study for analysis. The techniques introduced above can be used for any real system. Regardless of the input data, a system administrator utilizing these techniques would need to evaluate their effectiveness for their exact system. The performance from the simulation analysis does not imply the same performance on *all* systems.

8.4.2. Application Parameters. To model the stochastic execution times, Gamma distributions are used [147]. Gamma distributions were chosen to represent the application execution times as they are non-negative and their shape is flexible, allowing the representation of execution time distributions of a myriad of different applications. To generate the serial execution time distributions for each application on each processor type, the Coefficient of Variation (**COV**) method was used [62] to obtain the mean and standard deviation

of each application on each processor type. To generate the parallel characteristics for the Downey model for each application, the distributions from [154] are used.

8.4.3. System Slowdown. To represent the slowdown of a given processor type, a modified form of a Beta distribution was used because it is a flexible distribution on the interval [0,1] that was similar to the shape of small scale slowdown studies we conducted on actual systems. It can be used to model the *system availability*, defined as the inverse of the system slowdown, where in this context 0 corresponds to no availability and a slowdown of ∞, 1 corresponds to a fully available system and no slowdown, and a processor type that is 50% available would have a slowdown of 2. To use the Beta distribution as a model for the system slowdown, the reciprocal of the x-axis is used (i.e., the x-axis is now on the interval $[1, \infty)$ instead of $[0, 1]$). We denote the *overall system slowdown* as $\boldsymbol{\Gamma}$, defined as the weighted average (weighted by the number of processors for each type) of the mean system slowdown of the processor types.

8.4.4. Makespan Goal. Let $\boldsymbol{\mu_{i,j,k}}$ be the mean execution time of application i using k processors on processor type j. The calculation of the makespan goal, Δ, is described in (62).

$$\Delta = \frac{\sum_{i=1}^{N_a} \sum_{j=1}^{N_p} \mu_{i,j,M}}{N_a N_p} \tag{62}$$

The intuition behind using (62) is that it represents an average of the applications' performance in the given heterogeneous system. The inner summation averages the mean execution time of an application across all processor types. The outer summation averages across all applications. This allows different scenarios (e.g., number of applications, number of processor types) to be compared by using the same makespan goal calculation. The determination

of the makespan goal will have an impact on the resource allocation determined by the heuristics and needs to be explored in the future, but is out of the scope of this chapter.

8.4.5. Upper Bound on Robustness. Because robustness is defined across all applications as the minimum probability $P(i,(j,k))$ that an application i in resource allocation I completes by the makespan goal Δ, it is possible to find an upper bound on robustness based upon the worst performing application. The upper bound (given in (63)), $\boldsymbol{B}$, states that for each application i, find the processor type j that maximizes its probability of completing by the makespan goal if it is given *all* processors of that type (i.e., n_j). Out of all of those probabilities, the application that has the minimum probability of completing by the makespan goal sets the upper bound on system robustness.

$$B = \min_{i=1..N_a} \max_{j=1..N_p} P(i,(j,n_j)) \tag{63}$$

8.5. Simulation Results

For the simulation analysis, three different batch sizes (i.e., N_a) of 8, 32, and 128 applications are considered. The total number of processor types (i.e., N_p) explored for each batch size were $\{1,2,4,8\}$, $\{1,2,4,8,16\}$, and $\{1,2,4,8,16,32\}$, respectively. The total number of processors in the system (i.e., τ) with batch sizes of 8, 32, and 128 were 64, 256, and 1024, respectively, where n_j varies between types. The overall system slowdown, Γ, was broken into categories of low, mixed, and high corresponding to Γ of 1.1, 1.36, and 1.6, respectively. Each scenario (where a scenario is a batch size, system size, and system slowdown category) was run for 48 trials for each of the five heuristics with the 25th, median, and 75th quartiles shown. The plus symbols show all trials outside of the 25th and 75th quartiles. Between trials, the application characteristics and the makespan goal differed. The stopping criterion

for the robustness floor heuristics, Λ, was set to 0.01. This led to each robustness floor variant running for seven iterations ($\lceil \log_2(\Lambda^{-1}) \rceil$).

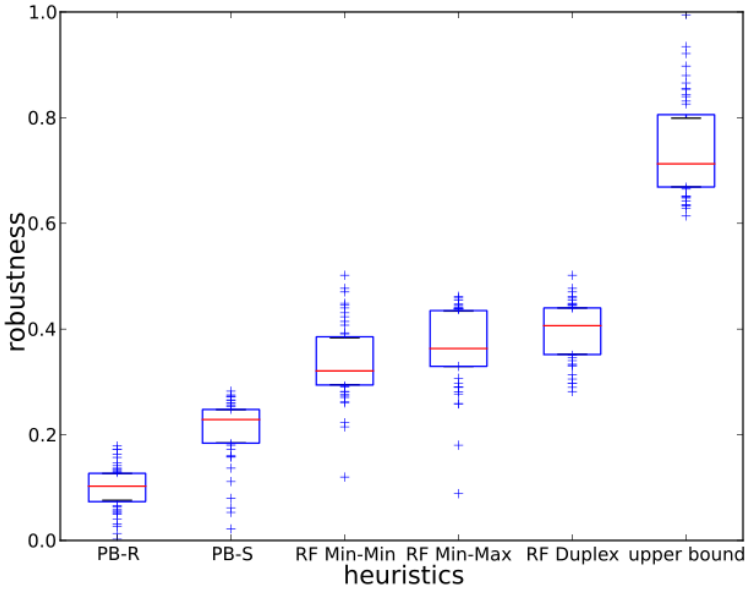


FIGURE 8.4. A typical result in the comparison of the five heuristics and the upper bound. The batch size was 32 applications with four processor types with high system slowdown. The box plot shows the distribution of 48 trials with the 25th, median, and 75th quartile trials represented by the box. All trials outside of the 25th and 75th quartile are shown with the plus symbol.

A typical result is presented in Fig. 8.4. This figure shows the robustness compared between the five heuristics and the upper bound. The batch size was 32 applications with four processor types with high slowdown and 256 total processors. The two processor balance heuristics do not perform as well as the three robustness floor heuristics. This is because the processor balance heuristics only make decisions on which processor type to allocate an application to, where the number of processors per application is fixed as M. The PB-S heuristic performs better than the PB-R heuristic because at each allocation decision it assigns the application that will have the highest probability of completing by the makespan goal where PB-R makes random allocation decisions. Out of the robustness floor heuristics,

RF Min-Max, in general, performs better than RF Min-Min. This is because as more applications are allocated, there is less room and it becomes harder to assign the applications that need more processors. RF Min-Max assigns these larger applications first, leading to better performance in most cases. RF Duplex will always perform at least as well as RF Min-Min and RF Min-Max because it runs both greedy algorithms at each iteration. By using intelligent calculation optimizations, running this algorithm only requires 5 to 10% longer than either RF Min-Min and RF Min-Max. The upper bound is not overlapped by any of the heuristics because, in general, it is not achievable as it assumes an application occupies an entire processor type. This does not leave enough resources for the remaining applications to have an ample opportunity of completing by the makespan goal.

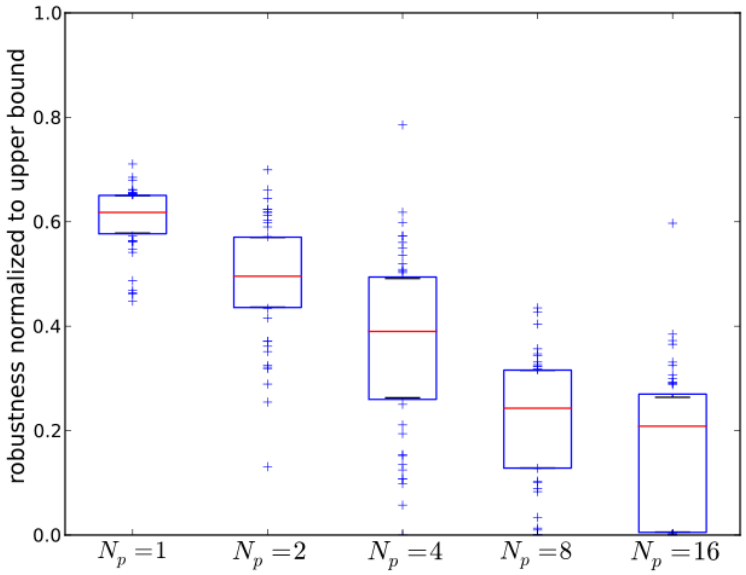


FIGURE 8.5. The trend in performance, in terms of B_{norm}, when increasing the number of processor types in the system. A B_{norm} value of zero indicates that the robustness exactly equals the upper bound (i.e., $\Psi(I) = B$). The batch size was 32 applications and the system had a low system slowdown. The number of processor types varied between 1, 2, 4, 8, and 16.

Because RF Duplex was the best performing heuristic in all scenarios with respect to robustness, the rest of the discussion is focused on it. In the simulations, the upper bound

was noticed to remain mostly constant for any given number of processor types. This is because the upper bound is only set by one application on one processor type, therefore the number of processor types does not matter in that calculation, but rather the heterogeneity and performance of a single processor type in the system. Where the number of processor types does matter, however, is with the heuristics. The more processor types there are, the more the heuristics can leverage the benefit of the heterogeneity in the system.

The heuristics leveraging the heterogeneity in the system is apparent in Fig. 8.5. Let $\boldsymbol{B_{norm}}$ be difference between the upper bound and the robustness, normalized by the upper bound (i.e., $B_{norm} = \left(\frac{B-\Psi(I)}{B}\right)$). As B_{norm} approaches zero, the performance of the resource allocation approaches the upper bound. Fig. 8.5 shows how B_{norm} changes when the number of processor types are varied in a system with a batch size of 32 and low system slowdown. The increase in the number of processors in the system leads to better performance by RF Duplex with respect to the upper bound.

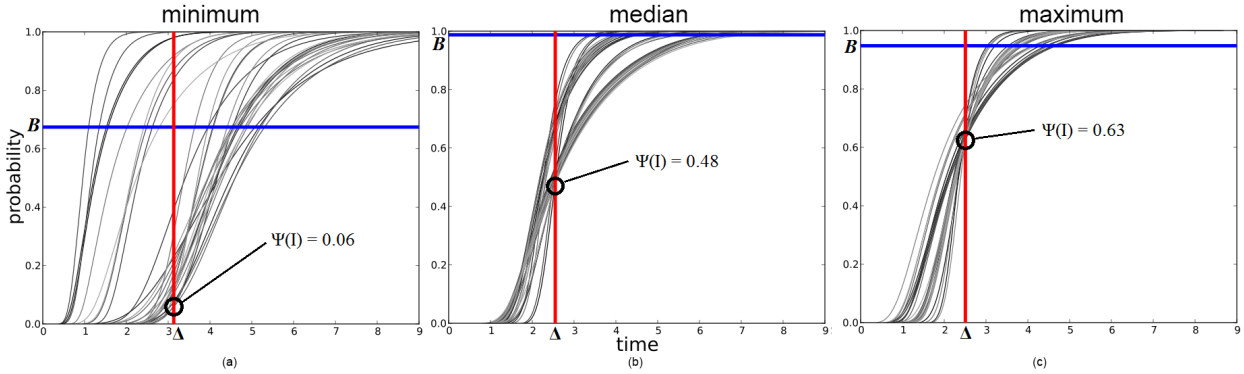


FIGURE 8.6. Three complete resource allocations for the RF Duplex heuristic. Each graph shows the cumulative distribution function of each application in the batch. The vertical red line shows the makespan goal. The horizontal blue line shows the upper bound on robustness. Graph (a) shows the trial that had the worst performance in terms of robustness. Graph (b) shows the median trial and graph (c) shows the best performing trial. The results are for a batch size of 128 applications with eight processor types with mixed slowdown.

Fig. 8.6 is different than the previous figures in that each graph only shows one complete resource allocation. The graphs show the cdf of all the allocated application execution times. The vertical red line shows the specific makespan goal of the trial while the horizontal blue line shows the upper bound. The results are for a batch size of 128 applications with eight processor types with mixed slowdown. The three graphs, from left to right, show the worst performing trial (Fig. 8.6.a) in terms of robustness, the median trial (Fig. 8.6.b), and the best performing trial (Fig. 8.6.c).

In the worst performing trial, the RF Duplex heuristic does a poor job of setting the applications to finish near the makespan goal. This could be due to the specific parameters of the applications in that particular trial. The upper bound is much lower in the worst-performing trial in comparison to the others, meaning that a high robustness measure for that set of applications would be harder to achieve. In the median and maximum graphs, RF Duplex sets the applications' probability of completing at the makespan goal to be close to the same.

8.6. Conclusions

A robustness metric was presented that combines the uncertainties of moldable parallel applications with stochastic execution times and heterogeneous resources with uncertain availability. Using knowledge of the parallel characteristics of the application in conjunction with the robustness metric, three iterative-greedy heuristics were designed and studied through simulation. In practice, the RF Duplex heuristic should be used. For a small increase in computation time, it combines the benefits of RF Min-Min and RF Min-Max.

In the future, additional resource allocation heuristics will be designed, implemented, and analyzed. The sensitivity of the performance of the heuristics to the setting of the

makespan goal will be explored. Last, as in Chapter 7, the resource allocation techniques will be combined with a second stage that implements dynamic loop scheduling, a suite of runtime performance optimization techniques [133].

CHAPTER 9

Conclusions and Future Work

9.1. Conclusions

Two different areas of resource allocation optimization were explored: Smart Grid and High-performance computing. The primary area of research in this preliminary dissertation was in Smart Grid resource allocation with a heuristic framework being presented in Chapter 2. Smart Grid resource allocation was studied from the perspective of the aggregator entity in Chapters 3 and 5. The aggregator in this dissertation is a for-profit entity that interfaces the electricity market and the end-user (customer), allowing the end-user the chance to participate in a fully deregulated electricity market by offering control of their assets to the aggregator. To encourage customer participation, a new pricing mechanism called *customer incentive pricing* was created. Customer incentive pricing is an alternative pricing structure for the customer if they allow their assets to be controlled by the aggregator. Results in Chapter 3 show that this new pricing structure is, in most cases, cheaper than the alternative distribution company price. The aggregator-based Smart Grid resource allocation of customer smart appliances was formulated as an optimization problem (optimizing for aggregator profit) and implemented using a genetic algorithm in Chapter 3. The genetic algorithm was run using real pricing data from a distribution company and independent system operator and it was shown that the aggregator is able to make a profit, participating customers saved money, and the peak load of the system was reduced as a common good. Two new visualization methods for demand response programs were presented in Chapter 5. The visualization methods, load heat maps and three-dimensional load curves, were used to analyze the results from the aggregator-based Smart Grid resource allocation. The analysis

allowed greater insight into whether or not the demand response plan worked effectively; at what times the demand response resulted in a profit or a loss; and how multiple demand response solutions compared.

A customer-approach to the SGRA problem was presented in Chapter 4. Three new home energy management systems were designed, one myopic (denoted minimum forecast cost) and two non-myopic approaches based on a partially observable Markov decision process decision making framework. Results in Chapter 4 show significant savings in monthly electricity cost by using the proposed home energy management systems in a real-time pricing market. To enable the co-simulation of electric power system simulation tools, Bus.py (a Python-based abstract software transmission bus interface) was designed and presented in Chapter 6.

The secondary area of research was resource allocation for parallel scientific applications in high-performance computing. A combined dual-stage framework was presented in Chapter 7 for allocating heterogeneous compute resources with uncertain performance to parallel scientific applications to maximize their performance. In the first stage, batch resource allocation heuristics were used to allocate processors to parallel applications to maximize the probability that they all finish by a given deadline. The second stage performs a finer-grain optimization during runtime to minimize the application execution times given the resources allocated from the first stage. A small scale example was given to show the benefit of the dual-stage approach. The problem was improved and the first stage was studied in greater depth in Chapter 8. A mathematical model was designed for the stochastic execution times of moldable parallel applications that are assigned to heterogeneous parallel resources, incorporating the change in execution times when applications are mapped to different numbers of processors. A metric for robustness was proposed that combined the uncertainty in application execution times and the uncertainty in machine performance.

Three novel iterative greedy heuristics were designed to optimize for the robustness and their benefit was shown by comparing to two heuristics from literature across a myriad of system sizes.

9.2. Future Work

The future directions of this work lie in combining the system and end-user approaches. To accomplish this, different tools from the bulk power market to the individual end-user appliance must be co-simulated (enabled by Bus.py from Chapter 6). More end-user assets with enhanced capabilities, such as distributed generation, electric vehicles, and thermal loads (including their inherent uncertainty) should be considered and modeled.

In Chapter 3, a for-profit aggregator-based CPS was presented. Future work can extend the CPS market structure, where each aggregator submits bids to a day-ahead DR market that co-exists with the bulk power market. The ISO uses these inputs to determine the day-ahead spot market prices and perform unit commitment and economic dispatch.

In Chapter 3, only one aggregator entity on one distribution system was considered. In future work, the complex problem of multiple-interacting aggregators will be addressed. The aggregators are spatially dispersed, existing on the distribution level, across a transmission network. To fully quantify the meaningful impact of DR programs across a large geographical area and over long time periods, unique HPC simulation tools will be used. These tools, including Bus.py, enable the parallel co-simulation of the transmission and distribution networks using MATPOWER and GridLAB-D, respectively.

The use of HPC is critical to the tractability of the future CPS studies. Fig. 9.1 shows the proposed multi-core HPC simulation test bed. All transmission-level computations (i.e., power flow, markets, system information) occur at the master core. Each transmission

bus simulates the associated distribution system using a GridLAB-D instance—each on a separate core, communicating with the master core. Each distribution system contains one or more aggregator simulations—each on a separate core. Let B be the number of load buses and D_i be the number of aggregators on bus i. The total number of cores required for the simulation are $1 + B + \sum_{i=1}^{B} D_i$.

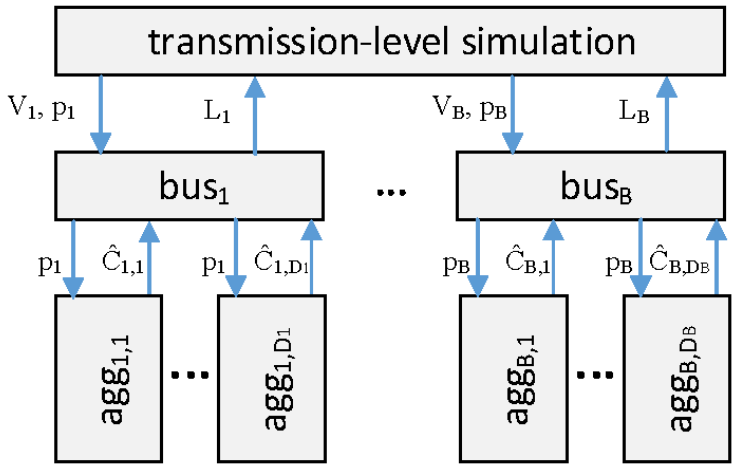


FIGURE 9.1. The high-performance computing architecture and communication for the parallel simulation of the CPS. The transmission-level simulator communicates voltages (V_i), locational marginal price (p_i), and load (L_i) to and from each bus i. Each distribution bus i passes p_i to each aggregator j and receives a set of asset controls $\hat{C}_{i,j}$.

The spatial network considerations are locational marginal pricing, bus voltages, and line flows. The transmission system considered is the IEEE 118-bus test case. Data for generators, bus loads, and weather come from real sources. Locational marginal pricing is determined by the bulk power market with a simulated real-time pricing program at each bus.

Another future consideration is sustainability in electric power systems. The underlying premise of sustainability is "that economic well-being is inextricably linked to the health of the environment and the success of the world's communities and citizens" [155]. According to [156], the three pillars of sustainability are: economics (profits), environment

(planet), and society (people). In 1999, a federal advisory committee propounded that environmental sustainability is the cornerstone of economic and societal sustainability [157]. In future work, methods for environmental sustainability through deferral of transmission infrastructure investments and reduction of the output of dirty diesel peaking units, and economic sustainability through the aggregator DR programs should be explored. Two new metrics for quantifying the long-term wide-area impact of such programs on sustainability are proposed here.

The first proposed sustainability metric quantifies the reduction in peak load through the use of DR. Because, in a real power system, the peak before a DR action cannot be directly measured, using the reduction in capacity factor of peaking generators (typically running on "dirty" fuels such as diesel) is proposed as a proxy measurement for the environmental benefit of peak reduction. In terms of peaking generators, to reduce the wholesale cost of electricity and reduce environmental emissions, the capacity factor of the peaking generators should ideally be zero. Let $C_{pk,ref}$ and $C_{pk,\tau}$ be the capacity factors of the peaking generators for a reference case (i.e., before DR was implemented) and the case of interest, respectively. The proposed metric that addresses the environmental resilience pillar of sustainability is given by $C_{pk,ref} - C_{pk,\tau}$ and should be maximized. The time interval considered for the reference case and case of interest should be the same.

The second proposed sustainability metric is designed from the perspective of customer economic well-being. Because there is information about the cost of electricity the customer *would* have paid if not participating with the aggregator from Chapter 3 (i.e., the local utility's price), it is possible to calculate the customer savings. Let $\gamma(i, d)$ indicate whether (i.e., 1) or not (i.e., 0) asset i was rescheduled on day d, H be the number of days of interest, I be the total number of assets, and $c_{i,d,0}$ and $c_{i,d,sch}$ be the cost of using asset i on day d

at the local utility price and customer-incentive price, respectively. The cost terms, $c_{i,d,0}$ and $c_{i,d,sch}$, are calculated using information about asset power, asset duration, and cost of electricity. The proposed metric that addresses the economic demand pillar of sustainability is quantified as

$$\sum_{d=1}^{H} \left[\frac{\sum_{i=1}^{I} \left[\gamma(i,d) c_{i,d,sch} + (1 - \gamma(i,d)) c_{i,d,0} \right]}{\sum_{i=1}^{I} c_{i,d,0}} \right]. \tag{64}$$

Note this is an aggregate measurement for all customers, but it could easily be calculated for each individual customer.

References

[1] ComEd. ComEd residential real-time pricing program. Accessed: Aug. 18, 2013. [Online]. Available: https://rrtp.comed.com/live-prices/

[2] United States Federal Energy Regulatory Commission. PJM daily report archives. Accessed: Jan. 14, 2014. [Online]. Available: http://www.ferc.gov/market-oversight/mkt-electric/pjm/pjm-iso-archives.asp

[3] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

[4] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, Nov. 1989.

[5] E. G. Coffman, *Computer and Job-Shop Scheduling Theory.* John Wiley & Sons, New York, NY, 1976.

[6] O. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of the ACM*, vol. 24, no. 2, pp. 280–289, Apr. 1977.

[7] T. Hansen, R. Roche, S. Suryanarayanan, H. J. Siegel, D. Zimmerle, P. M. Young, and A. A. Maciejewski, "A proposed framework for heuristic approaches to resource allocation in the emerging Smart Grid," in *IEEE PES International Conference on Power Systems Technology (POWERCON 2012)*, Oct. 2012, 6 pp.

[8] M. G. Simoes, R. Roche, E. Kyriakides, A. Miraoui, B. Blunier, K. McBee, S. Suryanarayanan, P. Nguyen, and P. Ribeiro, "Smart-grid technologies and progress in Europe and the USA," in *Energy Conversion Congress and Exposition (ECCE 2011)*, Sep. 2011, pp. 383–390.

[9] F. M. Ciorba, T. Hansen, S. Srivastava, I. Banicescu, A. A. Maciejewski, and H. J. Siegel, "A combined dual-stage framework for robust scheduling of scientific applications in heterogeneous environments with uncertain availability," in *21st Heterogeneity in Computing Workshop (HCW 2012) in the proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, May 2012, pp. 193–207.

[10] H. E. Brown, S. Suryanarayanan, and G. T. Heydt, "Some characteristics of emerging distribution systems under the Smart Grid Initiative," *The Electricity Journal*, vol. 23, no. 5, pp. 64–75, Jun. 2010.

[11] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*. Wiley New York, 1996.

[12] D. S. Kirschen and G. Strbac, *Fundamentals of Power Systems Economics*. Wiley Online Library, 2004.

[13] J. Ferber, *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Reading, 1999.

[14] C. Quinn, D. Zimmerle, and T. H. Bradley, "The effect of communication architecture on the availability, reliability, and economics of plug-in hybrid electric vehicle-to-grid ancillary services," *Journal of Power Sources*, vol. 195, no. 5, pp. 1500–1509, Mar. 2010.

[15] R. Roche, S. Natarajan, A. Bhattacharyya, and S. Suryanarayanan, "A framework for co-simulation of AI tools with power systems analysis software," in *23rd International Workshop on Database and Expert Systems Applications (DEXA)*, Sep. 2012, pp. 350–354.

[16] D. C. Walters and G. B. Sheble, "Genetic algorithm solution of economic dispatch with valve point loading," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1325–1332, Aug. 1993.

[17] P. Chen and H. Chang, "Large-scale economic dispatch by genetic algorithm," *IEEE Transactions on Power Systems*, vol. 11, no. 4, pp. 1919–1926, Nov. 1995.

[18] S. A. Kazarlis, A. G. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 11, no. 1, pp. 83–92, Feb. 1996.

[19] T. T. Maifeld and G. B. Sheble, "Genetic-based unit commitment algorithm," *IEEE Transactions on Power Systems*, vol. 11, no. 3, pp. 1359–1370, Aug. 1996.

[20] Z. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1187–1195, Aug. 2003.

[21] G. Gross and F. D. Galiana, "Short-term load forecasting," *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1558–1573, Dec. 1987.

[22] 110th Congress of the United States of America, Energy Independence and Security Act of 2007, "Title XIII – Smart Grid," Dec. 2007.

[23] S. Suryanarayanan, F. Mancilla-David, J. Mitra, and Y. Li, "Achieving the Smart Grid through customer-driven microgrids supported by energy storage," in *2010 IEEE International Conference on Industrial Technologies (ICIT)*, Mar. 2010, pp. 884–890.

[24] United States Department of Energy. (Apr. 2011) Energy Incentive Programs. [Accessed: May 13, 2012]. [Online]. Available: http://www1.eere.energy.gov/femp/financing/energyincentiveprograms.html

[25] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hengsen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, June 2001.

[26] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel, "Stochastic robustness metric and its use for static resource allocations," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1157–1173, Aug. 2008.

[27] L. D. Briceño, H. J. Siegel, A. A. Maciejewski, Y. Hong, B. Lock, M. N. Teli, F. Wedyan, C. Panaccione, C. Klumph, K. Willman, and C. Zhang, "Robust resource allocation in a massive multiplayer online gaming environment," in *4th International Conference on the Foundations of Digital Games (FDG 2009)*, Apr. 2009, pp. 232–239.

[28] F. Xhafa, L. Barolli, and A. Durresi, "Immediate mode scheduling in grid systems," *International Journal of Web and Grid Services*, vol. 3, no. 2, pp. 219–236, June 2007.

[29] M. Maheswaran, S. Ali, H. J. Siegel, D. Hengsen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 107–131, Nov. 1999.

[30] H. E. Brown, S. Suryanarayanan, S. A. Natarajan, and S. Rajopadhye, "Improving reliability of islanded distribution systems with distributed renewable energy resources," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2028–2038, Dec. 2012.

[31] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[32] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[33] D. Whitley, "The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best," in *3rd International Conference on Genetic Algorithms*, June 1989, pp. 116–121.

[34] T. M. Hansen, R. Roche, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "Heuristic optimization for an aggregator-based resource allocation in the Smart Grid," *IEEE Transactions on Smart Grid*, 10 pages, accepted 2015, to appear. DOI: 10.1109/TSG.2015.2399359.

[35] United States Department of Energy, "The Smart Grid: An introduction," 2009.

[36] United States Energy Information Administration, "Annual energy outlook 2013," Apr. 2013.

[37] J. Osborne and D. Warrier, "A primer on demand response–The power grid: Evolving from a dumb network to a Smart Grid," *Thomas Weisel Partners Equity Research*, Oct. 2007.

[38] PJM Forward Market Operations, "Energy and Ancillary Services Market Operations," June 2014.

[39] P. C. Stern, "Information, incentives, and proenvironmental consumer behavior," *Journal of Consumer Policy*, vol. 22, no. 4, pp. 461–478, Dec. 1999.

[40] X. Chen, T. Wei, and S. Hu, "Uncertainty-aware household appliance scheduling considering dynamic electricity pricing in smart home," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 932–941, Mar. 2013.

[41] A. Agnetis, G. de Pascale, P. Detti, and A. Vicino, "Load scheduling for household energy consumption optimization," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2364–2373, Dec. 2013.

[42] T. H. Chang, M. Alizadeh, and A. Scaglione, "Coordinated home energy management for real-time power balancing," in *IEEE Power and Energy Society General Meeting*, July 2012, 8 pp.

[43] C. Chen, J. Wang, Y. Heo, and S. Kishore, "MPC-based appliance scheduling for residential building energy management controller," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1401–1410, Aug. 2013.

[44] M. A. A. Pedrasa, T. D. Spooner, and I. F. MacGill, "Coordinated scheduling of residential distributed energy resources to optimize smart home energy services," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 134–143, Sep. 2010.

[45] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in Smart Grid using heuristic optimization," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1244–1252, Sep. 2012.

[46] R. Roche, *Agent-based Architectures and Algorithms for Energy Management in Smart Grids*. Ph.D. dissertation. Université de Technologie de Belfort-Montbéliard, Belfort, France, Dec. 2012.

[47] M. Roozbehani, M. A. Dahleh, and S. K. Mitter, "Volatility of power grids under real-time pricing," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1926–1940, Nov. 2012.

[48] C. Quinn, D. Zimmerle, and T. H. Bradley, "An evaluation of state-of-charge limitations and actuation signal energy content on plug-in hybrid electric vehicle, vehicle-to-grid reliability, and economics," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 483–491, Mar. 2012.

[49] L. Gkatzikis, I. Koutsopoulos, and T. Salonidis, "The role of aggregators in Smart Grid demand response markets," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1247–1257, July 2013.

[50] A. Pratt, "A holistic approach to energy management, and experimental results from a proof-of-concept residential management system, implemented in a single-family residence, as well as subsequent approaches for managing energy across a group of residences in a neighborhood, and for a group of commercial buildings assembled into a microgrid," presented at the *IEEE PES General Meetings*, July 2013.

[51] California Energy Commission. 2013 Integrated Energy Policy Report. Accessed: Mar. 28, 2014. [Online]. Available: http://www.energy.ca.gov/2013_energypolicy/

[52] P. Du and N. Lu, "Appliance commitment for household load scheduling," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 411–419, June 2011.

[53] E. Sortomme and M. A. El-Sharkawi, "Optimal scheduling of vehicle-to-grid energy and ancillary services," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 351–359, Mar. 2012.

[54] R. Friese, T. Brinks, C. Oliver, A. A. Maciejewski, and H. J. Siegel, "Analyzing the trade-offs between minimizing makespan and minimizing energy consumption in a heterogeneous resource allocation problem," in *2nd International Conference on Advanced Communications and Computation*, Oct. 2012, pp. 81–89.

[55] Z. Chen, L. Wu, and Y. Fu, "Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1822–1831, Dec. 2012.

[56] United States Federal Energy Regulatory Commission, "FERC Order 745: Demand response compensation in organized wholesale energy markets," Mar. 2011.

[57] L. D. Briceño, H. J. Siegel, A. A. Maciejewski, M. Oltikar, J. Brateman, J. White, J. Martin, and K. Knapp, "Heuristics for robust resource allocation of satellite weather data processing on a heterogeneous parallel system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1780–1787, Nov. 2011.

[58] F. D. Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Computers and Operations Research*, vol. 22, no. 1, pp. 15–24, Jan. 1995.

[59] D. Kalyanmoy, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[60] L. Gkatzikis, T. Salonidis, N. Hegde, and L. Massoulié, "Electricity markets meet the home through demand response," in *IEEE Conference on Decision and Control*, Dec. 2012, pp. 5846–5851.

[61] T. Bapat, N. Sengupta, S. K. Ghai, V. Arya, Y. B. Shrinivasan, and D. Seetharam, "User-sensitive scheduling of home appliances," in *ACM SIGCOMM Workshop on Green Networking*, Aug. 2011, pp. 43–48.

[62] S. Ali, H. J. Siegel, M. Maheswaran, D. Hengsen, and S. Ali, "Representing task and machine heterogeneities for heterogeneous computing systems," *Tamkang Journal of Science and Engineering,* Special Tamkang University 50th Anniversary Issue, vol. 3, no. 3, pp. 195–208, Nov. 2000. Invited.

[63] N. Lu and Y. Zhang, "Design considerations of a centralized load controller using thermostatically controlled appliances for continuous regulation reserves," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 914–921, June 2013.

[64] P. Du and N. Lu, "Appliance commitment for household load scheduling," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 411–419, June 2011.

[65] A. Zipperer, P. A. Aloise-Young, S. Suryanarayanan, R. Roche, L. Earle, D. Christensen, P. Bauleo, and D. Zimmerle, "Electric energy managemeny in the smart home: Perspectives on enabling technologies and consumer behavior," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2397–2408, Aug. 2013.

[66] T. M. Hansen, E. K. P. Chong, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "A partially-observable Markov decision process approach to residential home energy management," *under preparation for journal*, May 2015.

[67] United States Energy Information Administration, "Annual energy outlook 2014," May 2014.

[68] A. Zipperer, P. Aloise-Young, and S. Suryanarayanan, "On the design of a survey for reconciling consumer behaviors with demand response in the smart home," in *proceedings of the North American Power Symposium (NAPS), 2013*, Sep. 2013, 6 pp.

[69] M. H. Albadi and E. El-Saadany, "A summary of demand response in electricity markets," *Electric Power Systems Research*, vol. 78, no. 11, pp. 1989–1996, 2008.

[70] A. J. Conejo, J. M. Morales, and L. Baringo, "Real-time demand response model," *Smart Grid, IEEE Transactions on*, vol. 1, no. 3, pp. 236–242, 2010.

[71] A. H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 120–133, Aug. 2010.

[72] H. Tischer and G. Verbic, "Towards a smart home energy management system–A dynamic programming approach," in *2011 IEEE PES Innovative Smart Grid Technologies Asia (IGST)*, Nov. 2011, 7 pp.

[73] T. Hubert and S. Grijalva, "Modeling for residential electricity optimization in dynamic pricing environments," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2224–2231, Dec. 2012.

[74] M. Pipattanasomporn, M. Kuzlu, and S. Rahman, "An algorithm for intelligent home energy management and demand response analysis," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2166–2173, Dec. 2012.

[75] K. M. Tsui and S. C. Chan, "Demand response optimization for smart home scheduling under real-time pricing," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1812–1821, Dec. 2012.

[76] Y. Guo, M. Pan, Y. Fang, and P. P. Khargonekar, "Decentralized coordination of energy utilization for residential households in the Smart Grid," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1341–1350, Sep. 2013.

[77] Z. Yu, L. Jia, M. C. Murphy-Hoye, A. Pratt, and L. Tong, "Modeling and stochastic control for home energy management," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2244–2255, Dec. 2013.

[78] Z. Zhao, W. C. Lee, Y. Shin, and K. Song, "An optimal power scheduling method for demand response in home energy management system," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1391–1400, Sep. 2013.

[79] M. Beaudin, H. Zareipour, A. K. Bejestani, and A. Schellenberg, "Residential energy management using a two-horizon algorithm," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1712–1723, July 2014.

[80] Q. Jia, J. Shen, Z. Xu, and X. Guan, "Simulation-based policy improvement for energy managemeny in commercial office buildings," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2211–2223, Dec. 2012.

[81] S. Meyn, P. Barooah, A. Bušić, Y. Chen, and J. Ehren, "Ancillary service to the grid using intelligent deferrable loads," *arXiv preprint arXiv:1402.4600*, 30 pp., Feb. 2014.

[82] S. Misra, A. Mondal, S. Banik, M. Khatua, A. Bera, and M. S. Obaidat, "Residential energy management in Smart Grid: A Markov decision process-based approach," in *Green Computing and Communications (GreenCom)*, Aug. 2013, pp. 1152–1157.

[83] W. Wang and Z. Lu, "Cyber security in the Smart Grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344–1371, Apr. 2013.

[84] S. G. Eick, W. A. Massey, and W. Whitt, "$M_t/G/\infty$ queues with sinusoidal arrival rates," *Management Science*, vol. 39, no. 2, pp. 241–252, Feb. 1993.

[85] E. K. P. Chong, C. M. Kreucher, and A. O. Hero III, "Partially observable Markov decision process approximations for adaptive sensing," *Discrete Event Dynamic Systems,* special issue on Optimization of Discrete Event Dynamic Systems, vol. 19, no. 3, pp. 377–422, Sep. 2009.

[86] R. E. Bellman, *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[87] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[88] R. C. Garcia, J. Contreras, M. van Akkeren, and J. B. C. Garcia, "A GARCH forecasting model to predict day-ahead electricity prices," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 867–874, May 2005.

[89] D. P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, 1995.

[90] PJM. PJM estimated hourly load. Accessed: Apr. 30, 2015. [Online]. Available: http://www.pjm.com/markets-and-operations/energy/real-time/loadhryr.aspx

[91] T. M. Hansen, S. Suryanarayanan, A. A. Maciejewski, H. J. Siegel, and A. V. Modali, "A visualization aid for demand response studies in the Smart Grid," *The Electricity Journal*, vol. 28, no. 3, pp. 100–111, Apr. 2015.

[92] H. Farhangi, "The path of the Smart Grid," *IEEE Power and Energy Magazine*, vol. 8, no. 1, pp. 18–28, Jan. 2010.

[93] J. D. Weber and T. J. Overbye, "Voltage contours for power system visualization," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 404–409, Feb. 2000.

[94] S. K. Card, *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco, CA, 1999.

[95] C. Ware, *Information Visualiztion: Perception for Design*. Elsevier, Waltham, MA, 2012.

[96] United States Department of Energy, "2008 visualization and controls agenda," in *Department of Energy Peer Reviews*, Oct. 2008.

[97] M. H. Albadi and E. F. El-Saadany, "A summary of demand response in electricity markets," *Electric Power Systems Research*, vol. 78, no. 11, pp. 1989–1996, Nov. 2008.

[98] Y. Sun and T. J. Overbye, "Visualizations for power system contingency analysis," *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 1859–1866, Nov. 2004.

[99] T. J. Overbye, J. D. Weber, and K. J. Patten, "Analysis and visualization of market power in electric power systems," *Decision Support Systems*, vol. 30, no. 3, pp. 229–241, Jan. 2001.

[100] T. J. Overbye, P. W. Sauer, C. M. Marzinzik, and G. Gross, "A user-friendly simulation program for teaching power system operations," *IEEE Transactions on Power Systems*, vol. 10, no. 4, pp. 1725–1733, Nov. 1995.

[101] J. D. Glover, M. S. Sarma, and T. Overbye, *Power System Analysis and Design, Fifth Edition*. Cengage Learning, Stamford, CT, 2011.

[102] G. Zhang, S. Lee, R. Carroll, J. Zuo, L. Beard, and Y. Liu, "Wide area power system visualization using real-time synchrophasor measurements," in *IEEE Power and Energy Society General Meeting*, July 2010, 7 pp.

[103] J. Bank and J. Hambrick, "Development of a high resolution, real time, distribution-level metering system and associated visualization, modeling, and data analysis functions," National Renewable Energy Laboratory, Golden, CO, Tech. Rep., May 2013.

[104] R. Billinton and S. Jonnavithula, "A test system for teaching overall power system reliability assessment," *IEEE Transactions on Power Systems*, vol. 11, no. 4, pp. 1670–1676, Nov. 1996.

[105] H. E. Brown, S. Suryanarayanan, S. A. Natarajan, and S. Rajopadhye, "Improving reliability of islanded distribution systems with distributed renewable energy resources," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2028–2038, Dec. 2012.

[106] J. Giráldez, R. Roche, S. Suryanarayanan, and D. Zimmerle, "A linear programming methodology to quantify the impact of PHEVs with V2G capabilities on distribution systems," in *2013 IEEE Green Technologies Conference*, Apr. 2013, pp. 8–15.

[107] T. M. Hansen, B. Palmintier, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "Bus.py: A GridLAB-D communication interface for smart distribution grid simulations," in *IEEE Power and Energy Society General Meeting 2015*, 5 pages, July 2015.

[108] 110th Congress of the United States of America, Energy Independence and Security Act of 2007, "Title XIII – Smart Grid," Dec. 2007.

[109] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "GridLAB-D: An open-source power systems modeling and simulation environment," in *IEEE PES Transmission and Distribution Conference and Exposition*, Apr. 2008, 5 pp.

[110] M. A. Al Faruque and F. Ahourai, "GridMat: Matlab toolbox for GridLAB-D to analyze grid impact and validate residential microgrid level energy management algorithms," in *IEEE PES Innovative Smart Grid Technologies Conference (ISGT 2014)*, 2014, 5 pp.

[111] D. Wang, B. de Wit, S. Parkinson, J. Fuller, D. Chassin, and C. Crawford, "A test bed for self-regulating distribution systems: Modeling integrated renewable energy and demand response in the GridLAB-D/MATLAB environment," in *IEEE PES Innovative Smart Grid Technologies Conference (ISGT 2012)*, 2012, 7 pp.

[112] K. Anderson, J. Du, A. Narayan, and A. E. Gamal, "GridSpice: A distributed simulation platform for the smart grid," in *IEEE 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, May 2013, 5 pp.

[113] K. Turitsyn, P. Šulc, S. Backhaus, and M. Chertkov, "Distributed control of reactive power flow in a radial distribution circuit with high photovoltaic penetration," in *IEEE PES General Meeting 2010*, July 2010, 6 pp.

[114] E. Ela, M. Milligan, and M. O'Malley, "A flexible power system operations simulation model for assessing wind integration," in *IEEE PES General Meeting 2011*, July 2011, 8 pp.

[115] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. B. Kopena, "Network simulations with the ns-3 simulator," presented at SIGCOMM, 2008, [Online] Abstract available: http://goo.gl/25vuwr.

[116] D. Pinney, "Costs and benefits of conservation voltage reduction," National Rural Cooperative Association, Arlington, VA, Tech. Rep., Nov. 2013. [Online]. Available: http://goo.gl/J2gnn5

[117] K. P. Schneider, Y. Chen, K. P. Chassin, R. Pratt, D. Engel, and S. Thompson, "Modern grid initiative: Distribution taxonomy final report," Pacific Northwest National Laboratory, Richland, WA, Tech. Rep., Nov. 2008. [Online]. Available: http://goo.gl/IJIAkf

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

[119] K. Spees and L. B. Lave, "Demand response and electricity market efficiency," *The Electricity Journal*, vol. 20, no. 3, pp. 69–85, Apr. 2007.

[120] Federal Energy Regulatory Commission, "Assessment of demand response and advanced metering," Dec. 2012. [Online]. Available: http://goo.gl/gJhV2v

[121] H. Aalami, G. R. Yousefi, and M. P. Moghadam, "Demand response model considering EDRP and ToU programs," in *IEEE PES Transmission and Distribution Conference and Exposition*, Apr. 2008, 6 pp.

[122] Duke Energy Carolinas, LLC. Residential service Time-of-Use. Accessed: Nov. 15, 2013. [Online]. Available: http://goo.gl/q3K8Lt

[123] P. H. G. Berkhout, J. C. Muskens, and J. W. Velthuijsen, "Defining the rebound effect," *Energy Policy*, vol. 28, no. 6, pp. 425–432, June 2000.

[124] W. J. Cole, K. M. Powell, E. T. Hale, and T. F. Edgar, "Reduced-order residential home modeling for model predictive control," *Energy and Buildings*, vol. 74, pp. 69–77, May 2014.

[125] N. R. Satish, "Compile Time Task and Resource Allocation of Concurrent Applications to Multiprocessor Systems," Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, USA, 2009.

[126] C. Boneti, R. Gioiosa, F. J. Cazorla, and M. Valero, "Using hardware resource allocation to balance HPC applications," *Parallel and Distributed Computing*, Ros Alberto, Ed. InTech, 2010.

[127] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, July 2004.

[128] I. Banicescu, F. Ciorba, and R. L. Cariño, "Towards the robustness of dynamic loop scheduling on large-scale heterogeneous distributed systems," in *International Symposium on Parallel and Distributed Computing (ISPDC 2009)*, June 2009, pp. 129–132.

[129] S. Srivastava, I. Banicescu, and F. Ciorba, "Investigating the robustness of adaptive dynamic loop scheduling on heterogeneous computing systems," in *Parallel and Distributed Scientific and Engineering Computing Workshop 2010,* in the Proceedings of the *International Parallel and Distributed Processing Symposium (IPDPSW-PDSEC 2010)*, May 2010, pp. 1–8.

[130] A. Aziz and H. El-Rewini, "Grid resource allocation and task scheduling for resource intensive applications," in *International Conference on Parallel Processing Workshops (ICPP 2006 Workshops)*, Aug. 2006, 8 pp.

[131] M.-J. Huguet and P. Lopez, "Mixed task scheduling and resource allocation problems," in *International Conference on Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming (CPAIOR 2000)*, Mar. 2000, pp. 71–79.

[132] A. Hurson, J. Lim, K. Kavia, and B. Lee, "Parallelization of DOALL and DOACROSS loops: A survey," *Advances in Computers*, vol. 45, pp. 53–103, 1997.

[133] I. Banicescu and R. L. Cariño, "Addressing the stochastic nature of scientific computations via dynamic loop scheduling," *Electronic Transactions on Numerical Analysis*, vol. 21, pp. 66–80, Oct. 2005.

[134] R. L. Cariño and I. Banicescu, "Dynamic load balancing with adaptive factoring methods in scientific applications," *The Journal of Supercomputing*, vol. 44, pp. 41–63, Apr. 2008.

[135] S. F. Hummel, E. Schonberg, and L. E. Flynn, "Factoring: A method for scheduling parallel loops," *Communications of the ACM*, vol. 35, no. 8, pp. 90–101, Aug. 1992.

[136] Y. A. Li, J. K. Antonio, H. J. Siegel, M. Tan, and D. W. Watson, "Determining the execution time distribution for a data parallel program in a heterogeneous computing environment," *Journal of Parallel and Distributed Computing*, vol. 44, no. 1, pp. 35–52, July 1997.

[137] L. Wasserman, *All of Statistics: A Concise Course in Statistical Interference*. New York, NY: Springer Science+Business Media, 2005.

[138] S. Ali, A. A. Maciejewski, and H. J. Siegel, "Perspectives on robust resource allocation for heterogeneous parallel systems," *Handbook of Parallel Computing: Models, Algorithms, and Applications*, S. Rajasekaran and J. Reif, Ed. Boca Raton, FL: Chapman and Hall/CRC Press, 2008.

[139] J. Smith, E. K. P. Chong, A. A. Maciejewski, and H. J. Siegel, "Stochastic-based robust dynamic resource allocation in a heterogeneous computing system," in *International Conference on Parallel Processing (ICPP 2009)*, Sep. 2009, pp. 188–195.

[140] T. M. Hansen, F. M. Ciorba, A. A. Maciejewski, H. J. Siegel, S. Srivastava, and I. Banicescu, "Heuristics for robust allocation of resources to parallel applications with uncertain execution times in heterogeneous systems with uncertain availability," in *2014 International Conference on Parallel and Distributed Computing (ICPDC'14),* in the proceedings of the World Congress of Engineering 2014 (WCE2014), July 2014, 6pp.

[141] M. M. Eshaghian, *Heterogeneous Computing.* Artech House Publishers, 1996.

[142] S. Ali, T. D. Braun, H. J. Siegel, A. A. Maciejewski, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "Characterizing resource allocation heuristics for heterogeneous computing systems," *Advances in Computers Volume 63: Parallel, Distributed and Pervasive Computing*, Ali R. Hurson, Ed. Elsevier, 2005.

[143] X. Qin and H. Jiang, "A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters," *Journal of Parallel and Distributed Computing*, vol. 65, no. 8, pp. 885–900, Aug. 2005.

[144] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," *Journal of Parallel and Distributed Computing*, vol. 4, no. 2, pp. 175–187, Feb. 1993.

[145] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU availability of time-shared Unix systems on the computational grid," *Cluster Computing*, vol. 3, no. 4, pp. 293–301, Dec. 2000.

[146] D. J. Robb and E. A. Silver, "Probability density functions of task-processing times for deterministic, time-varying processor efficiency," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1049–1052, Nov. 1990.

[147] S. R. Lawrence and E. C. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain," *Journal of Operations Management*, vol. 15, no. 1, pp. 71–82, Feb. 1997.

[148] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user estimates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 789–803, June 2007.

[149] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick, "System noise, OS clock ticks, and fine-grained parallel applications," in *19th International Conference on Supercomputing*, June 2005, pp. 303–312.

[150] R. Ruiz and T. Stützle, "An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1143–1159, June 2008.

[151] *Maui Scheduler^{TM} Administrator's Guide*, Adaptive Computing Enterprises, Inc. [Online]. Available: http://docs.adaptivecomputing.com/maui

[152] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[153] A. B. Downey, "A parallel workload model and its implications for processor allocation," *Cluster Computing*, vol. 1, no. 1, pp. 133–145, May 1998.

[154] W. Cirne and F. Berman, "Using moldability to improve the performance of supercomputer jobs," *Journal of Parallel and Distributed Computing*, vol. 62, no. 10, pp. 1571–1601, Oct. 2002.

[155] J. Schwarz, B. Beloff, and E. Beaver, "Use sustainability metrics to guide decision-making," *Chemical Engineering Process*, vol. 98, no. 7, pp. 58–63, 2002.

[156] R. Hansmann, H. A. Mieg, and P. Frischknecht, "Principal sustainability components: Empirical analysis of synergies between the three pillars of sustainability," *International Journal of Sustainable Development & World Ecology*, vol. 19, no. 5, pp. 451–459, June 2012.

[157] K. N. Johnson, J. Agee, R. Beschta, V. Dale, L. Hardesty, J. Long, L. Nielsen, B. Noon, R. Sedjo, M. Shannon, R. Trosper, C. Wilkinson, and J. Wondolleck, "Sustaining the people's lands – recommendations for stewardship of the national forests and grasslands into the next century," *Journal of Forestry*, vol. 97, no. 5, pp. 6–12, May 1999.

APPENDIX A

Baseline Customer Loads

As described in Section 3.4.4.3, the non-schedulable baseline loads are generated using Table A.1 from [46]. The penetration level indicates the probability that a given customer owns a given load. If a customer *does* own a load, the rating is determined using a normal distribution, $N(\mu, \sigma)$, with mean μ and standard deviation σ. The start time of the load is determined by sampling either a normal distribution or a uniform distribution, $U(a_1, a_2)$, between a_1 and a_2. If the load is run multiple times in the day (i.e., occurrences > 1), then the start time is sampled from each of the corresponding distributions. If the load is continuous, the rating is assumed to be constant throughout the day.

TABLE A.1. Baseline Appliance Model Parameters [46]

Appliance	**Penetration** (%)	**Rating** (kW)	**Occur-rences**	**Start time** (hour)	**Duration** (15-minute intervals)
Electric cooker	100	N(3.00,1.00)	2	N(12,1), N(18,1)	3
Water kettle	95	N(2.00,0.20)	2	N(8,1), U(14,18)	1
Dishwasher	95	N(1.60,0.10)	1	N(14,2)	4
Coffee maker	80	N(1.00,0.20)	2	N(7,1), U(14,18)	2
Microwave	85	N(1.00,0.10)	3	N(8,1), N(12,2), N(18,2)	1
Toaster	92	N(1.00,0.10)	1	N(9,1)	2
Fridge-freezer	100	N(0.13,0.03)	–	–	Continuous
Freezer	45	N(0.10,0.03)	–	–	Continuous
Other kitchen appliances	95	N(0.50,0.10)	3	N(8,1), N(13,2), N(19,2)	1
Washing machine	99	N(1.60,0.20)	1	U(8,20)	4
Laundry-dryer	80	N(2.50,0.25)	1	U(10,22)	5
Vacuum cleaner	97	N(1.50,0.50)	1	U(10,20)	2
Hair dryer	80	N(2.00,0.10)	2	N(8,1), N(18,2)	1
Circulation pump	100	N(0.05,0.01)	–	–	Continuous
TV 1	99	N(0.15,0.03)	2	N(13,5), N(19,2)	16
TV 2	80	N(0.15,0.03)	2	N(13,5), N(21,2)	8
PC 1	84	N(0.15,0.03)	1	N(14,7)	16
PC 2	40	N(0.15,0.03)	1	N(14,7)	8
Hi-fi system	81	N(0.03,0.01)	1	N(19,3)	4
Other goods	100	N(0.05,0.03)	–	–	Continuous
Lighting	100	N(0.20,0.05)	2	N(7,1), N(19,3)	12

APPENDIX B

Market Response Approximation

This appendix describes the market response from Chapter 3. Let $s_p(x)$ be the predicted spot market price in cents/kWh for a given load x in MW. The sixth-order polynomial approximation of $s_p(x)$ for ComEd within PJM for July 9, 2011, is given in (65). Note that the regression fit is empirical over the domain $x = [69200, 112000]$ MW and should not be used for loads outside of these values.

$$\begin{aligned} s_p(x) = \quad & 1.84 \times 10^{-26} x^6 - 9.60 \times 10^{-21} x^5 + \\ & 2.07 \times 10^{-15} x^4 - 2.38 \times 10^{-10} x^3 + \\ & 1.52 \times 10^{-5} x^2 - 0.516x + 7230 \\ & \text{for} \quad 69200 \leq x \leq 112000 \end{aligned} \tag{65}$$

APPENDIX C

Parameters for Appliance Type Generation

For this simulation study, we set $N_a = 10$, $\mu_p = 1$ kW, $\sigma_p = 0.25$ kW, $\theta_t = 0.05$, $\mu_d = 1$ hour, and θ_d =0.5. This corresponds to an average appliance load of 1 kW and duration of 1 hour. The coefficient-of-variation between appliance types (θ_d) is relatively large, while the variation within an appliance type is small (θ_t). This is because there is usually a large variation in the run times of different appliances, but when using the same appliance it usually runs for similar times. For scaling the load, we set $o_{min} = 0.5$ kW and $o_{min} = 6$ kW. This implies that at the ComEd system minimum, the house will be drawing approximately 500 W, and at the ComEd system maximum the house will be drawing approximately 6 kW. These parameters may be changed to simulate differently rated households.

APPENDIX D

AR + GARCH Model

For our AR+GARCH model for the POMDP-GARCH HEMS optimization method, we set $m = 504$, $P = 1$, and $Q = 3$. These parameters come from previous research on power system markets [88]. The values for the coefficients, provided in Table D.1, were determined by minimizing the log-likelihood function using MATLAB. If a coefficient is missing, then the corresponding value is 0.

TABLE D.1. AR + GARCH Parameters

AR Coefficient	**Value**	**GARCH Coefficient**	**Value**
k	-0.070	χ	0.960
m_1	0.435	ϕ_1	0.064
m_2	0.072	q_1	0.578
m_3	0.074	q_2	0.148
m_4	0.012	q_3	0.059
m_5	0.033		
m_6	0.011		
m_{22}	0.031		
m_{23}	-0.025		
m_{24}	0.065		
m_{25}	0.007		
m_{26}	-0.013		
m_{27}	-0.032		
m_{47}	0.025		
m_{48}	0.009		
m_{71}	-0.003		
m_{72}	0.018		
m_{73}	-0.001		
m_{95}	-0.008		
m_{96}	0.011		
m_{97}	-0.003		
m_{119}	-0.014		
m_{120}	0.015		
m_{121}	0.008		
m_{143}	0.025		
m_{144}	-0.010		
m_{145}	0.022		
m_{167}	0.002		
m_{168}	0.038		
m_{169}	0.004		
m_{170}	-0.020		
m_{191}	-0.027		
m_{192}	0.029		
m_{193}	-0.015		
m_{215}	-0.018		
m_{216}	0.032		
m_{217}	0.004		
m_{239}	0.002		
m_{240}	-0.021		

AR Coefficient	Value	GARCH Coefficient	Value
m_{241}	-0.018		
m_{264}	0.010		
m_{287}	-0.067		
m_{288}	0.038		
m_{311}	0.037		
m_{312}	0.015		
m_{322}	0.002		
m_{336}	0.030		
m_{337}	-0.021		
m_{338}	-0.006		
m_{358}	-0.025		
m_{359}	0.014		
m_{360}	-0.003		
m_{408}	-0.029		
m_{503}	-0.012		
m_{504}	0.001		

APPENDIX E

Customer Load Points and GIS Data

The customer load points for Chapter 5. Table E.1 describes the physical location of each of the numbered nodes from the RBTS bus mapped onto Fort Collins, Colorado (in Fig. 5.7). The coordinates are given in degree-minute-second format. The buses 22–47 are nodes that contain customer loads, with the probability that each of the 5,555 customers exist on a specific node given by "Prob." The buses 1–21 do not contain customer loads and as such do not have an associated probability of containing customers.

TABLE E.1. Customer Load Points

Bus	Latitude	Longitude	Bus	Latitude	Longitude	Prob.
1	N40°32'40.10"	W105°6'53.12"	22	N40°31'58.21"	W105°6'53.12"	0.0335
2	N40°33'21.94"	W105°6'53.12"	23	N40°33'21.94"	W105°6'18.77"	0.0434
3	N40°33'55.63"	W105°6'53.12"	24	N40°33'21.94"	W105°7'37.52"	0.0503
4	N40°34'29.15"	W105°6'53.12"	25	N40°33'55.63"	W105°6'18.97"	0.0434
5	N40°34'54.74"	W105°6'53.12"	26	N40°33'55.63"	W105°7'47.86"	0.0503
6	N40°32'12.37"	W105°5'44.32"	27	N40°34'29.15"	W105°5'58.46"	0.0380
7	N40°32'54.00"	W105°5'44.32"	28	N40°34'29.15"	W105°7'47.68"	0.0380
8	N40°33'19.85"	W105°5'44.32"	29	N40°31'38.50"	W105°5'44.32"	0.0353
9	N40°33'53.83"	W105°5'44.32"	30	N40°32'12.37"	W105°6'18.34"	0.0281
10	N40°34'19.89"	W105°5'44.32"	31	N40°32'54.00"	W105°6'28.84"	0.0353
11	N40°34'53.75"	W105°5'44.32"	32	N40°33'19.85"	W105°6'38.85"	0.0353
12	N40°32'38.08"	W105°4'36.43"	33	N40°33'53.83"	W105°6'28.57"	0.0353
13	N40°33'11.69"	W105°4'36.43"	34	N40°34'19.89"	W105°6'18.60"	0.0503
14	N40°33'45.66"	W105°4'36.43"	35	N40°32'4.20"	W105°4'36.43"	0.0281
15	N40°34'11.40"	W105°4'36.43"	36	N40°32'38.01"	W105°5'10.92"	0.0335
16	N40°34'52.93"	W105°4'36.43"	37	N40°33'11.69"	W105°5'30.84"	0.0503
17	N40°32'44.10"	W105°3'21.60"	38	N40°33'45.66"	W105°4'1.92"	0.0281
18	N40°33'25.32"	W105°3'21.60"	39	N40°33'45.66"	W105°5'20.95"	0.0434
19	N40°33'52.33"	W105°3'21.60"	40	N40°34'11.40"	W105°5'31.15"	0.0281
20	N40°34'17.85"	W105°3'21.60"	41	N40°32'2.26"	W105°3'11.60"	0.0353
21	N40°34'51.79"	W105°3'21.60"	42	N40°32'44.18"	W105°2'47.19"	0.0434
			43	N40°33'25.32"	W105°2'37.45"	0.0335
			44	N40°33'52.33"	W105°2'26.82"	0.0503
			45	N40°33'52.33"	W105°4'6.23"	0.0335
			46	N40°34'17.85"	W105°3'55.85"	0.0380
			47	N40°34'17.85"	W105°2'22.90"	0.0380